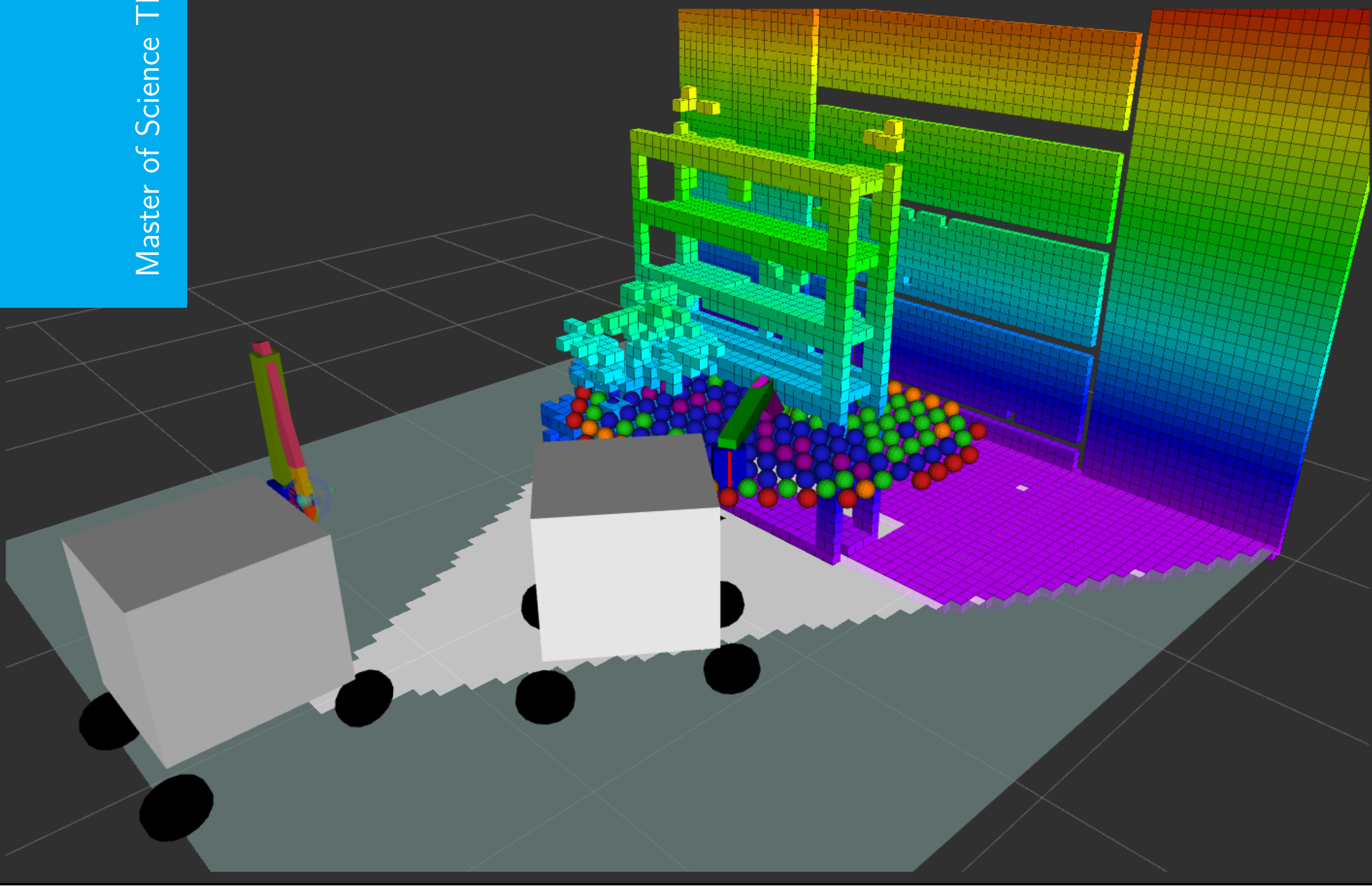
Robot Placement for Mobile Manipulation in Domestic Environments

Hemang Chawla

Master of Science Thesis



Delft University of Technology

Delft Center for Systems and Control

Robot Placement for Mobile Manipulation in Domestic Environments

Master of Science Thesis

For the degree of Master of Science in Biomechanical Design - Biorobotics at Delft University of Technology

Hemang Chawla

November 10, 2017

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Robot Care Systems (RCS). Their cooperation is hereby gratefully acknowledged.

Abstract

The development of domestic mobile manipulators for unconstrained environments has driven significant research recently. Robot Care Systems has been pioneering in developing a prototype of a mobile manipulator for elderly care. It has a 6 degrees of freedom robotic arm mounted on their flagship robot LEA, a non-holonomic differential drive platform. In order to utilize the navigation and manipulation capabilities of such mobile manipulators, robot placement algorithm that computes a favorable position and orientation of the mobile base is sought, which enables the end effector to reach a desired target. None of the existing approaches perform robot placement while ensuring a high chance of successful planning to target through a short path, while accounting for sensing and actuation errors typical in real world scenarios. This thesis presents a novel robot placement algorithm DeCOWA (Determining Commutation configuration using Optimization and Workspace Analysis) with these characteristics. Since the approach to robot placement is dependent upon the kind of mobile manipulation, a comparative study of sequential and full body methods is performed with respect to criteria important in domestic settings. Sequential mobile manipulation is found to be most suitable, for which a modular mobile manipulation framework encompassing motion planning and robot placement is presented. With sequential mobile manipulation, the ability to successfully reach a target depends upon the kinematic capabilities of the arm. Accordingly, robot placement with DeCOWA determines a favorable location for the arm, and corresponding platform orientation. To find the position of arm's base, an offline manipulator workspace analysis is performed generating the Inverse Reachability and Planability maps. During online use, these maps are combined into an Inverse Fusion Map that ranks different locations based on the ability of the arm placed there to find a successful and short motion plan to target. This map is filtered to generate a set of feasible locations at the arm's height. Through a ranked iterative search, a suitable collision free arm location is determined followed by minimization of the platform distance from robot's current pose. This approach is evaluated against an unbiased random placement of robot near the target using a sample set of twenty scenes mimicking domestic settings. It is found that DeCOWA is able to generate commutation configurations in fraction of a second, that lead to a high planning success rate, a short path length, and account for goal tolerance of navigation. Also, its modularity allows to use several planability metrics, making it useful for domestic application.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr. Martijn Wisse for being my supervisor for this endeavor of master thesis, and for his elaborate feedback on my thesis progress and writing. I also extend my gratitude to Dr. Pieter Jonker and Dr. Maja Rudinac for providing me the opportunity to pursue this research at Robot Care Systems.

I would like to thank my supervisor Aashish Vatsyayan for his guidance throughout the thesis. Your discipline and vision of technology research has been really inspiring through the year. I also extend heartfelt gratitude to my mentors Dr. Gabriel Lopes and Dr. Aswin Chandarr for their support and ideation during brainstorming sessions.

I must especially thank Abhijit Makhal whose research in robot placement became an inspiration for me to pursue this thesis.

I also extend my deepest gratitude to my friends Aseem Dua, Salil Apte, and Sofia Danai for their help in reviewing, reformulating, and completing this thesis. Without Salil's help in structuring this report and Aseem's feedback on my writing, this report would not have been possible. The discussions with Aseem on different metrics to be used for the algorithm aided a lot in developing a scientific scrutiny of my own research. I am also extremely thankful to Divyam Rastogi for patiently helping me when I found myself stuck on some things, and for his moral support.

I also thank Dimitrios, Geetank, Toby, and Lotte for their valuable feedback on my research and presentation. I highly appreciate the talks that I had with everyone at RCS during the course of this project.

This section is incomplete without acknowledging the support I have had from my family, whose faith in me has always been invaluable.

Thank you!

Delft, University of Technology
November 10, 2017

Hemang Chawla

Chapter 1

Introduction

The vision of domestic service robots assisting humans in our daily personal lives has been part of the human endeavor for a long time. With recent advances, this dream is being shaped into reality. Robots like Roomba are assisting to clean the floors [1], Functional Robot arm with user-frIENdly interface for Disabled people (FRIEND) is being used to aid people with functional impairments [2]. Such robots can especially be used to care for the elderly and people with physiological ailments (eg. wheelchair robot EL-E [3]), and for providing therapeutic assistance (eg. PARO [4], the seal robot to dementia patients). Several of these robots provide physical assistance in fetching and lifting of objects, provide personal care in form of keeping track of schedules, and help felicitating contact with family and friends. With transition from industry to households, service robots have the potential to propel humanity forward with significant impact to society, economics, and science. However, affordable, and effectively usable personal robots are still under development in scientific institutes around the globe. As per the International Federation of Robotics (IFR) [5], there has been a rise in the number of professional service robots over the years. Yet, the domestic robots in the market are limited to primarily vacuum cleaning, lawn mowing, and window cleansing robots [6]. Hence, there is a considerable scope for research to bring these robots into widespread use.

One of the bottlenecks to domestic mobile manipulation is the ability to find a suitable robot placement, i.e. the *determination of position and orientation of the robot platform at which the mobile manipulator can reach a desired target.* The selection of this robot placement also affects the quality of motion planning for the mobile manipulator. Unsuitable placement may render the target unreachable, or may imply a non-optimal motion to the target. The system must be designed considering requirements of domestic users, and the challenges posed by an unconstrained environment. Existing methods are unable to account for all these important criteria. This thesis proposes a novel generalizable algorithm to robot placement, while accounting for errors in actuation and sensing typical of real world application.

In this chapter an overview to personal/domestic service robots with attention towards geriatric care will be presented. It further discusses the challenges in developing mobile manipulation systems for domestic environments with focus on the challenge of robot placement.

Thereafter, the context and objective of the thesis are presented. Finally, the approach to solving the problem and relevant thesis contributions are described.

1-1 Personal Service Robots

The IFR defines a personal service (or domestic) robot as *"a service robot used for a non-commercial task, usually by lay persons. Examples are domestic servant robot, automated wheelchair, personal mobility assistant robot, and pet exercising robot."* They are often equipped with different capabilities such as voice recognition, camera vision, laser scanners and so on to navigate through homes and perform various tasks. Unlike the industrial robots, these do not focus on precision and speed. Instead, their goal is safe loco-manipulation so that humans can feel comfortable interacting with them. According to the IFR, over 31 million service robots will be sold worldwide between 2016 and 2019 [5] .

Service Robots for Geriatric and Assistive Care: One of the key utilities of service robotics is in the domain of geriatric care. With European society growing to be an aging society, the elderly population needs assistance with their day to day activities. Based on report of the Dutch Central Agency for Statistics, the percentage of aged people (age > 65) will increase to 21.7% in 2025, and to 25.3% by 2035 [7]. While walking, getting up and down the stairs, remembering things etc. are challenges for old people, the number of health-care personnel that can assist the elderly are reducing [8]. These tasks are even more difficult for those who may suffer from Alzheimer's, muscle dystrophy, arthritis etc. Personal robots for elderly care have been proposed as a viable solution to this issue. They help the users in remaining self-reliant and allow them to continue living in their own homes by providing a form of all-day personal care. Robots like Care-o-bot [9], wheelchair mounted robot arms like FRIEND [2], stroller based Lean Empowering Assistant (LEA) [10] etc are efforts in this direction.

Mobile Manipulators: Several of these geriatric care robots come under the category of Assistive Mobile Manipulators. A mobile manipulator combines the mobility of the base platform and the manipulability of the arm in a common system. This increases the workspace of the robot and makes a variety of tasks achievable. While the mobile base can be composed through multiple ways (such as wheels, legs, fins, or rotors for flying), this thesis will use a wheeled mobile manipulator as a test platform. These two components of a mobile manipulator can either be considered separately, or acting in coordination, or the whole robot can be treated as a single entity. Certain unique problems arise when a robotic manipulator and a wheeled platform are put together on a single system. Often, to reduce the complexity, sequential mobile manipulation is utilized [11] [12]. The approach to mobile manipulation affects the design of the robot placement algorithm. In contrast, the coordinated motion of the platform and the arm allows for reaching more constrained locations [12]. In chapter 2, the various approaches to mobile manipulation planning will be contrasted to select the most suitable method for domestic applications.

Based on [17], some of the challenges for mobile manipulators in domestic settings are summarized as follows:

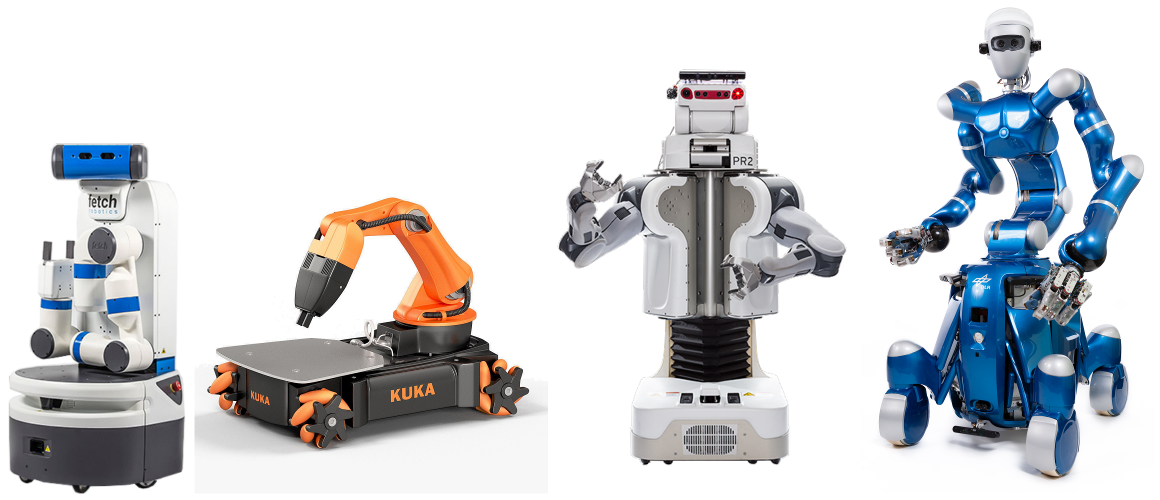


Figure 1-1: State-of-Art Mobile Manipulators(from left to right): FETCH [13], KUKA Youbot [14] , PR2 [15], Rollin' Justin [16]

- Unstructured environment: Design of homes, and placement of objects vary. There is multiplicity of tasks and variability within them. manipulate different kinds of objects such as glasses, plates, bottles, books etc which may be kept in different places, such as on a table, inside a cabinet, upside down.
- High dimensional configuration space: The degrees of freedom of the platform and that of all the arms attached to it make their coordination a complex task.
- Uncertainty in world model, sensing, and execution: Unlike a fixed factory environment where all scenes maybe known, the unstructured scene of a domestic environment is captured through sensors which have inaccuracies.
- Dynamic obstacles: People, pets, other robots move around in the house and the mobile manipulator is expected to adapt to this dynamic environment. This is different from factories where the robots work within a fixed environment within a cage.
- Human comfort: The presence of people require special consideration in order to make the robot motion appear user-friendly and suitable for interaction if required.

Figure 1-1 shows some of the state-of-the-art mobile manipulators. Robots like FETCH [13] have found use in warehousing, Justin Rollin' has been designed for assisting astronauts [18], PR2 and Youbot are used for research in domestic [19] and human-robot cooperative applications [20] respectively. However, mobile manipulators for domestic use are not wide spread.

Research competitions like RoboCup@Home League [21] focuses on improving service and assistive robot technology so that they can be used for personal domestic applications in the future. In this competition, objects to be manipulated are kept at predefined locations within 15cm of an edge of a flat surface. Furthermore, there is at least a 5cm space around each object to ease the reaching and grasping [22]. Based on these simplifications, the robot can place itself at a predetermined distance from the edge in front of the object in order to reach it.

Other methods used for robot placement include using a laser pointer [3], or a Graphical User Interface (GUI) [23] to select the location. One of the state-of-the art approaches to robot placement is based on workspace analysis [24] of the robotic arm. This approach computes the reaching capabilities of the manipulator for robot placement but does not consider the length of motion to target, and the errors in reaching the decided location typical of a domestic setting.

Therefore, a novel robot placement algorithm that considers aforementioned aspects is deemed necessary for improving robotic utility in home environments. Such an algorithm may also be used in robot placement for humanoids in competitions such as the Defense Advanced Research Projects Agency (DARPA) robotics challenge [25].

Having discussed the state-of-the-art in mobile manipulation, the next section describes the organization in which this thesis is conducted.

1-2 Robot Care Systems

Among the robots being designed for senior citizens in Europe is Lean Empowering Assistant (LEA) developed by Robot Care Systems (RCS), the Hague, Netherlands. It is a certified class B electronic product at the intersection of medical and personal service robots. Designed keeping afford-ability in mind, LEA consists of a mobile platform, and a low cost arm (Figure 1-2). LEA aids the elderly in walking with the correct posture and in avoiding obstacles. It reminds and keeps track of the daily progress of the user in aspects of medication and exercise. It is also used to keep in touch with family and friends, and in establishing contact with caregivers in case of an emergency. The robot can also move autonomously, if the user so desires.

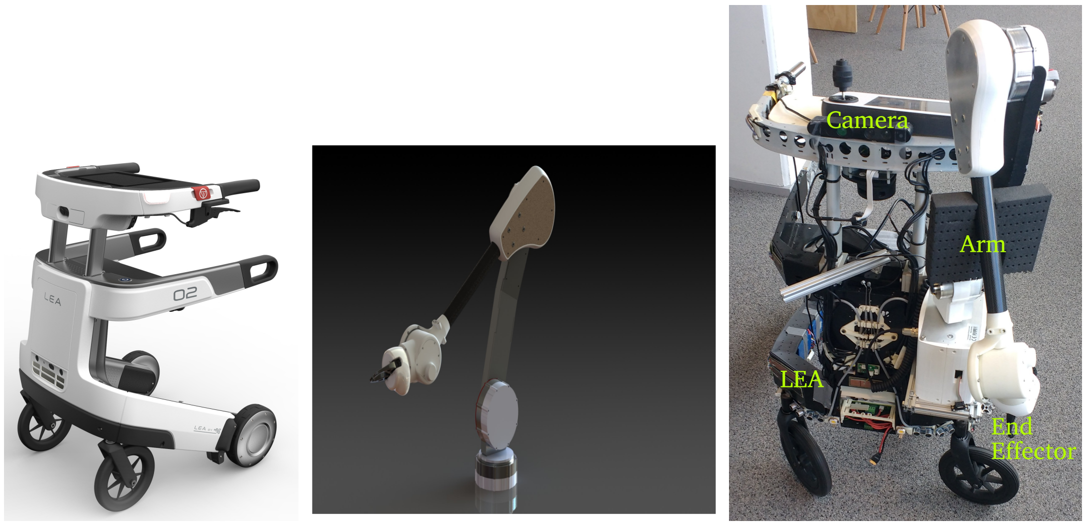


Figure 1-2: Model of LEA mobile base (left), and 6 d.o.f. arm (center), LEA with arm prototype.
Note: Figure is not to scale.
Source: Robot Care Systems

RCS now aims to develop a mobile manipulation system that autonomously brings the user objects kept at different rooms in a house. Presently, the RCS arm is able to reach a pre-grasp pose looking at a static scene through a Red Green Blue - Depth (RGB-D) camera. The user selects an object on the computer screen based on grab-cut segmentation [26]. Then the arm plans a path to a point in front of the object while avoiding obstacles for all links using a motion planner. With this system, once the arm is attached and integrated with the stroller, the user will have to walk to the object of choice and select the target as shown in the screen. While this is useful, it does not satisfy the potential that an autonomous personal robot may be capable of. Therefore, to merge the autonomous navigation capabilities of LEA with manipulation it is pertinent to design a robot placement algorithm within a mobile manipulation framework.

In order to formalize the mobile manipulation task being considered, the next section provides a mathematical formulation for a differentially driven wheeled mobile manipulator reaching a target pose.

1-3 Mathematical Formulation

Consider a non-holonomic mobile manipulator with n degrees of freedom. Of these, let the arm be of n_m degrees of freedom. Let the platform (also referred to as base) be of n_b degrees of freedom. Therefore $n = n_b + n_m$ are the degrees of freedom of the robot.

For the case of a wheeled mobile manipulator like LEA, $n_b = 3$, i.e. x_b, y_b, θ_b. Where x_b is the x coordinate of the platform, y_b is the y coordinate of the platform, and θ_b is the orientation of the platform. And $n_m = 6$.

A quasi-static form of such a non-holonomic system is described by

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \mathbf{u}), \tag{1-1}$$

$$G(\mathbf{q_b})\dot{\mathbf{q}}_b = 0. \tag{1-2}$$

where **boldface** letters denote a vector. Here $\mathbf{f}$ may be discontinuous but is measurable and bounded. $\dot{\mathbf{q}}$ is time-derivative of the configuration coordinates $\mathbf{q}$, and $\mathbf{u}$ is the control inputs to the system. Now,

$$\mathbf{q} = \begin{bmatrix} \mathbf{q_b} \\ \mathbf{q_m} \end{bmatrix} \tag{1-3}$$

composed of configuration of the base $\mathbf{q_b}$ and that of the manipulator $\mathbf{q_m}$. For the case of LEA and arm,

$$\begin{aligned} \mathbf{q_b} &= \begin{bmatrix} x_b & y_b & \theta_b \end{bmatrix}^T, \\ \mathbf{q_m} &= \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^T. \end{aligned}$$

Similarly $\mathbf{u}$ represents the number of control inputs that are admissible to the mobile manipulator,

$$\mathbf{u} = \begin{bmatrix} \mathbf{u_b} \\ \mathbf{u_m} \end{bmatrix}. \tag{1-4}$$

Here the base velocity $\mathbf{q_b}$ is subject to k non-holonomic constraints as shown in Eq. (1-2) with $G(\mathbf{q_b})$ being a full rank $(k \times n_b)$ Pfaffian constraints matrix.

While a generic non-holonomic system may be subject to k non-holonomic constraints, a differentially driven wheeled mobile platform (assuming a unicycle model) has one non-holonomic constraint

$$\begin{aligned} &[-sin(\theta_b)\ cos(\theta_b)\ 0] \cdot \dot{\mathbf{q_b}} = 0 \\ &\Leftrightarrow -\dot{x_b} sin(\theta_b) + \dot{y_b} cos(\theta_b) = 0. \end{aligned} \tag{1-5}$$

Therefore, the platform can move in two directions given by the columns of the matrix in the drift-less system,

$$\dot{\mathbf{q}}_\mathbf{b} = \begin{bmatrix} cos(\theta_b) & 0 \\ sin(\theta)_b & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u_b}.$$

For the manipulator,

$$\dot{\mathbf{q}}_\mathbf{m} = \mathbf{u_m}.$$

Hence,

$$\begin{aligned} \mathbf{q_m} \in & \quad \mathbb{R}^{n_m} = \mathbb{R}^6, \\ \mathbf{q_b} \in & \quad \text{SE}(n_b - k) = \text{SE}(2), \\ \mathbf{u} \in & \quad \mathbb{R}^{n-k} = \mathbb{R}^{n-1} = \mathbb{R}^8. \end{aligned}$$

The initial condition of the robot is given to be

$$\mathbf{q}(0) = \mathbf{q_0}, \tag{1-6}$$

$$\dot{\mathbf{q}}(0) = \mathbf{0} \tag{1-7}$$

implying the robot is static at the initial position.

Moreover, if the arm of the mobile manipulator is assumed to be folded initially, we get

$$\mathbf{q_m}(0) = \mathbf{0}.$$

The mechanical, and the actuation limits are given by in-equalities

$$\mathbf{q_{min}} \leq \mathbf{q} \leq \mathbf{q_{max}}, \tag{1-8}$$

$$\mathbf{u_{min}} \leq \mathbf{u} \leq \mathbf{u_{max}} \tag{1-9}$$

where $\mathbf{q_{min}}, \mathbf{u_{min}}$, and $\mathbf{q_{max}}, \mathbf{u_{max}}$ are the lower and upper mechanical and actuation limits respectively.

Also, the configuration space of the mobile manipulator is divided into mutually exclusive $\mathcal{C}^{free}$ and $\mathcal{C}^{obst}$. Here $\mathcal{C}^{free}$ represents the free configuration space of the robot, and $\mathcal{C}^{obst}$

represents the space with obstacles. Also, $\mathcal{S} \subset (\mathcal{C}^{free} \cup \mathcal{C}^{obst})$ is the subspace defining the singular positions of the arm of the mobile manipulator.

The end effector pose $\mathbf{p}$ is related to the configuration $\mathbf{q}$ by the kinematic map,

$$\mathbf{p} = h(\mathbf{q}) \mid \mathbf{p} \in \mathcal{P} \subseteq \mathrm{SE}(3).$$

Particularly, the objective is for the end effector to reach a target pose $\mathbf{p}^* \in \mathcal{P}$. With this the problem statement is formulated:

> **To compute a trajectory $\Omega(\mathbf{q}, t)$ such that $\Omega \in \mathcal{C}^{free}, \Omega \notin \mathcal{C}^{obst} \cup \mathcal{S}$. Moreover, it must satisfy the non holonomic constraint in Eq. (1-5), the limits in Eq. (1-8) and Eq. (1-9), for the given initial conditions Eq. (1-6) and Eq. (1-7).**

This constitutes the planning and control problem for mobile manipulators. Similar to [11], this is divided into the following sub-problems (Figure 1-3):

Robot Placement This deals with determining the goal pose of a mobile manipulator's platform such that the end-effector attached to the arm is able to reach a target pose, while satisfying certain criteria. This is the final location of the platform, $\mathbf{q_b}^{goal}$ in the trajectory $\Omega(\mathbf{q}, t)$ that will be computed. Based on kinematic redundancy of the mobile manipulator, the space of possible solutions is given by $\mathcal{B} = \{\mathbf{q} \in R^{n_m} \times \mathrm{SE}(n_b - k) : h(\mathbf{q}) = \mathbf{p}^*\}$. The solution must also lie in $\mathcal{C}^{free}$ and not in $\mathcal{C}^{obst}$ or $\mathcal{S}$.

Motion Planning Motion planning is composed of path planning, and time parametrization of this path. Path planning implies determining the best way to reach a desired pose, or configuration from the starting pose/configuration while avoiding self -collisions and obstacles. It describes the relation of $\Omega(\mathbf{q}, t)$ with $\mathbf{q}$. Among the infinitely many paths possible to reach a given pose, the best is decided based on a chosen metric such as energy required to traverse, distance traveled, smoothness of path, and so on. Parameterizing the generated path as a function of time allows to compute relevant control inputs that drive a robot from one pose to another.

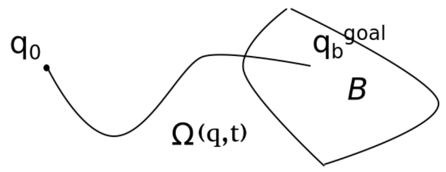


Figure 1-3: Problem tackled in the thesis. From $\mathbf{q_0}$ the trajectory $\Omega(\mathbf{q}, t)$ is used to traverse the robot to $\mathcal{B}$ where $h(\mathbf{q}) = \mathbf{p}^*$. Robot placement finds $\mathbf{q_b}^{goal}$ in $\mathcal{B}$.

As explained previously this thesis is primarily concerned with Robot Placement part of mobile manipulation. The thesis objective is therefore described in the next section.

1-4 Thesis Objective

The goal of the thesis is

> **To design a robot placement algorithm for mobile manipulation within domestic environments.**

Apart from *safety* and *modularity*, the desired algorithm must ensure:

- *High chance of successful motion planning to target*
- *Fast and smooth motion to target*
- *Low computation time*
- *Account for possible errors in actuation or sensing*

The aforementioned criteria are ranked decreasingly in importance based on their order of description. Presently, no robot placement algorithm in literature accounts for all of the above factors. Therefore, the proposed algorithm would be considered successful when it performs better than random placement of the robot near the target object. Random placement implies $\mathbf{q_b}^{goal}$ is chosen from a uniform random distribution of poses around the target object within a radius equal to the distance from robot's base to its end effector at maximum extension of the arm.

The chance of successful planning depends on the motion planner used as well as the placement of the robot relative to the target. The mobile manipulation framework must allow for selection of suitable planners for successful planning, and the robot placement should ensure a *success rate better than that of random placement* when using the same motion planner. Success rate SR_{MP} is measured over repeated trials as

$$SR_{MP} = \frac{\text{No. of successful plans}}{\text{No. of trials}} \times 100\%$$

The fastness of the motion depends on the path length of the platform as well as the arm. Since having a successful plan is more important than having a short motion, optimizing the path length of the platform is considered less important. A successful robot placement would have an average *arm path length less than that achieved through random placement.* The average path length PL is given by

$$PL = \frac{\sum_{i=1}^{\text{Successful Trials}} PL_i}{\text{No. of successful Trials}}.$$

The computation time is a composition of the time required for robot placement as well as motion planning. The state of the art robot placement algorithms such as [27], [28] take nearly *750ms* on an average using a 3.0 GHz Core i5/i7 PC within a single-threaded application. Therefore, the computation time for robot placement will be compared against this benchmark. The average time τ for computing robot placement is given by

$$\tau = \frac{\sum_{i=1}^{\text{Successful Trials}} \tau_i}{\text{No. of successful Trials}}.$$

It is desired that the system is able to accommodate *kinect sensing error with a maximum value of 4cm* [29]. Also, the system must account for an platform actuation uncertainty given by assumed Gaussian distributions based on the locomotion goal tolerance [30]. The uncertainty is therefore represented by normal distributions $\mathcal{N}(0, \frac{20}{3}\text{cm})$ in the x and y axis and $\mathcal{N}(0, \frac{0.05}{3}\text{rad})$ for the yaw.

Finally, it will be assumed that the robot is already in the same room as the desired object, and can visualize it through the setup shown in Figure 1-2. Therefore, the problem considered in this thesis is limited to the range where a RGB-D camera is able to visualize the target object (up to 5m) [29]. Furthermore, the target to be reached is taken to be the same as the pre-grasp pose for the mobile manipulator.

1-5 Approach

In order to design a suitable robot placement algorithm for domestic application, an appropriate mobile manipulation method must be selected first. Chapter 2 compares three state-of-the-art approaches to mobile manipulation, viz. sequential, full body configuration space, and full body task space mobile manipulation. The choice of this method affects the design of the robot placement algorithm. Unlike sequential mobile manipulation, full body approaches require that motion planning is necessarily considered in combination with robot placement. It is shown that for domestic environments, sequential approach to mobile manipulation is most appropriate. The sequential approach is relatively safe, requires low computation time, and can recover from errors in actuation of the platform. A modular framework for sequential mobile manipulation is implemented for LEA as elaborated in Chapter 3 to be used for assessing the proposed robot placement algorithm. Robot Operating System (ROS) [31], is used along with MoveIt! for motion planning of the robot. An inflated Universal Robot Description Format (URDF) [32] model of the robot allows to account for sensing errors during collision checking. The modular nature of this framework allows it to be used for any other domestic mobile manipulator like LEA as well.

When using the sequential mobile manipulation approach, the solution obtained through robot placement is called commutation configuration. At this configuration of the platform, the robot switches from locomotion to manipulation. In chapter 2 the state of the art in robot placement is analyzed. Various approaches have been implemented in literature ranging from those based on optimization, learning, and pre-computing the inverse kinematic capabilities of the manipulator. Learning methods [28], [33] for robot placement are specific to the scene, and hence can not be easily generalized. Optimization approach to robot placement has only found use in simulation due to high computation time [11]. Pre-computation of kinematic capabilities in form of a reachability map [24], and its inversion [27] was proposed as a viable approach to finding the best robot placement. Reachability analysis describes the kinematic capabilities of a manipulator in its workspace. The workspace is discretized into 3D [24] or 6D [27] voxels. Each voxel is assigned a quality metric based on a chosen criterion such as manipulability [34]. According to literature, the 6D reachability map can be inverted to generate a set of robot placements from which a pose at origin can be reached. The location having the highest quality metric is selected as the goal to which the robot must navigate. However inverse reachability maps based on the manipulability of the robot don't reflect the expectancy of a successful or a short motion. These methods also ignore the uncertainties present in a domestic environment.

In this thesis, a novel algorithm **De**termining **C**ommutation-configuration using **O**ptimization and **W**orkspace **A**nalysis (DeCOWA) is presented. The reachability map based approach of workspace analysis is modified to select the robot placement for high chance of successful planning while accounting for possible errors in reaching the chosen goal accurately. To do so, the problem is segregated into finding the best location for the arm and thereafter determining the orientation of the platform. The location for the arm is chosen through the inversion of arm's 3D reachability map (unlike the 6D reachability inversion). This size of these 3D voxels is decided based upon the actuation uncertainty of the platform. In a similar fashion, a new indicator in form of inverse *planability map* is utilized along with the inverse reachability map as a measure for shortness of motion. This map is created by inverting the map of minimum path length between arm's start configuration and different voxels. These maps are fused online (during the real-time robot placement process) to generate a set of locations where the arm has a high chance of successfully planning a short path. These voxels are iteratively searched in a decreasing order of their quality to find a collision free arm location. For the selected arm location, a collision free orientation of the platform is found by Nelder Mead minimization [35] of distance to the current platform position. Since most of the computation is done off-line, the online (i.e. during use) generation of robot placement is computationally fast. The proposed approach is generic and can be applied to different mobile manipulators in all kinds of scenes.

1-6 Thesis Outline

The remainder of this theses is structured as follows:

In Chapter 2 the state-of-the-art in robot placement and mobile manipulation motion planning are compared against the metrics important for domestic environments.

In Chapter 3 a framework for sequential mobile manipulation is provided explaining the different modules that make up the system.

Thereafter, the proposed algorithm for robot placement is elaborated in detail in Chapter 4.

In order to evaluate the robot placement method, a set of experiments and their results are described in Chapter 5. The chapter also discusses the implications of these results and also notes the limitations of the proposed approach.

Finally, Chapter 6 provides a summary of the thesis and concludes with providing directions for future research.

Chapter 2

State-Of-The-Art in Domestic Mobile Manipulation

This chapter examines different approaches to mobile manipulation available in literature, and selects the one suitable for home environments.This affects the comparison and design of robot placement approaches. In order to do so, the next section discusses the criteria important for robotic application in *domestic settings.* In section 2-2 three approaches to mobile manipulation are contrasted based on these criteria. Such a comparison has not been performed previously in literature. Finally, in section 2-3 the literature on robot placement is examined to note its limitations, and demonstrate a need for a novel robot placement algorithm.

2-1 Criteria for Domestic Environments

The criteria considered important for mobile manipulation in domestic environments are elaborated in this section.

Applicability to unconstrained environment Domestic environments are composed of changing scenes unlike the factory where manufacturing setups and assembly lines are fixed. Therefore, domestic environments are primarily characterized by the lack of constraints i.e. things may be moved from one place to another, or even change positions unintentionally. The environment is constituted by objects of different shapes, sizes, appearances and so on. Accordingly, a robotic system must be able to manipulate various objects at different locations as well as heights. As shown in Figure 2-1, a mobile manipulator is able to combine mobility of the robot platform and the manipulability of the arm in a common system. This increases the workspace of the robot and makes a variety of tasks achievable. Hence, being able to successfully complete the desired objective in unconstrained environments becomes a fundamental criterion in domestic environments. Since all motion planning methods are designed for finding paths in different scenes,the robot placement method must also be able to generate robot placement in different variety of scenes. *This criteria is primary in comparing approaches to robot placement* (discussed in section 2-3-1).

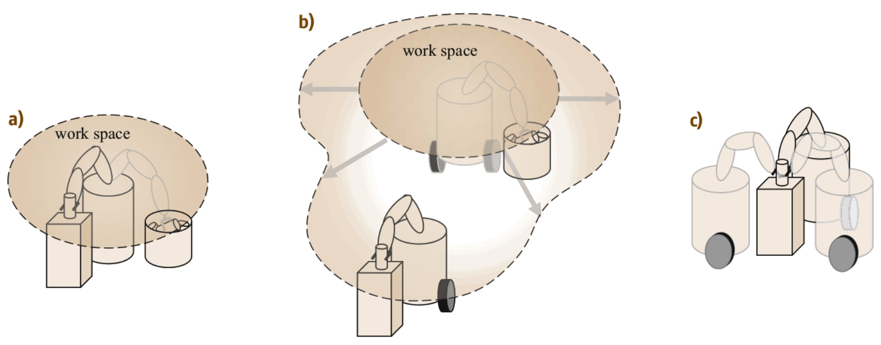


Figure 2-1: Increased workspace and redundancy of mobile manipulators: (a) A manipulator on a fixed base; (b) increased work space of a mobile manipulator; (c) redundancy due to the added mobility.
Source: [6]

Safety At the expense of this increased workspace is a greater focus on safety during motion. The robot must not hit objects in the environment while moving around. To do so, the robot must be able to sense its environment, detect the shape and size of obstacles, and plan motions that avoids them. Hitting inanimate objects in the house may even imply damaging the mechanical components of the robot itself. Collision with humans or pets may lead to physical injuries. Therefore, sensors such as RGB-D cameras, lasers scanners, and so on are used to generate a map of the environment, and wheel encoders, inertial measurement units (IMU) are used to simultaneously localize the robot. This map is then utilized during motion planning to generate paths that avoid collisions with obstacles. Safety is also important in determination of a collision free robot placement. However, *the primary use of this criterion is in selection of the mobile manipulation planning method.*

User comfort Furthermore, presence of humans in the house implies considering safety alone is not sufficient. The comfort of users is also an important factor for a marketable robot like LEA. Comfort of users in presence of robots is defined as "*the absence of annoyance and stress*" [36]. Robotic systems lead to stress when the robot gets too close to humans, or when the robot moves in a way that is random and unpredictable for the user. Therefore, user comfort to robotic applications focus on two aspects viz. proxemics [37] and motion quality. The proxemics based approaches focus on maintaining a certain distance to humans in order to not surprise them or indicate a possible collision. While such criteria have found use in developing algorithms for robot placement (eg. [38]) as well as motion planning (eg. [38], [39]), they fundamentally require identification and classification of humans amongst the perceived obstacles in a scene. Since this technology does not exist in LEA, the approaches to increasing user comfort through proxemics cannot be used in this thesis.

To incorporate user comfort in motion planning and execution, it has been found that energy efficient motions appear natural to humans [36]. Such motions increase user comfort because it allows them to interpret and predict robot's objective. Therefore, motion planners that generate smooth and consistent paths must be preferred. Different planners offer different smoothness and repeatability of generated plans. The success rate of planners also vary according to the scenes. Therefore a single planner cannot be deemed as the best for all scenarios. Accordingly, the approach to mobile manipulation planning must provide a choice of planners from which a suitable one can be selected based on the scene.

Accounting for uncertainty For robotic applications in the real world, it is pertinent to discuss the inherent uncertainty. Considering uncertainty is important because sensors used are not able to perfectly capture information about the environment. For example, an RGB-D camera is used to get images as well as the depth of obstacles. This allows to construct a 3D model of the environment in form of an octomap [40]. Due to uncertainty in sensing [29], the positions of objects in the octomap may not be accurate. In this thesis only the sensing errors due to uncertainty of Kinect camera will be considered explicitly. Furthermore, there is uncertainty in actuation of the robot. Due to Brockett's theorem [41], pose stabilization of the mobile platform for non-holonomic robots like LEA is a hard problem. This implies there will always be a difference in the pose reached by the robotic platform and the desired goal. Selecting appropriate mobile manipulation planning framework must be done considering this limitation.

Low computation and execution time Finally, the mobile manipulator must be able to complete the objective in a sufficiently fast manner. Task duration includes the time required to compute the path as well as its execution. Computation time is affected by the choice of motion planner and the approach used for robot placement. Planners that utilize optimization are generally slower as compared to those using randomized motion planning approaches [42]. The execution time of the motion is also dependent on the chosen robot placement as explained before as well as the length of the computed path.

In summary, considering safety and applicability to an unconstrained environment are primary criteria for domestic mobile manipulation. User comfort, being able to account for uncertainty, and fast computation and execution time are also factors that are deemed necessary. These will be used in the following sections to select a mobile manipulation planning approach, and analyze the available robot placement methods in literature.

2-2 Approaches to Mobile Manipulation Planning

Motion Planning for mobile manipulators may be classified into two categories. These are **sequential mobile manipulation** and **full body mobile manipulation**. The full body mobile manipulation is further divided into approaches that plan in the configuration space of the robot, and those that compute paths in the task space. In this section these three kinds of mobile manipulation approaches (illustrated in Figure 2-2) are compared in their ability to satisfy the aforementioned criteria.

Sequential Planning: Generally, mobile manipulation is approached through segregated motions of the base and arm. Firstly a path for the mobile base is computed and a local controller

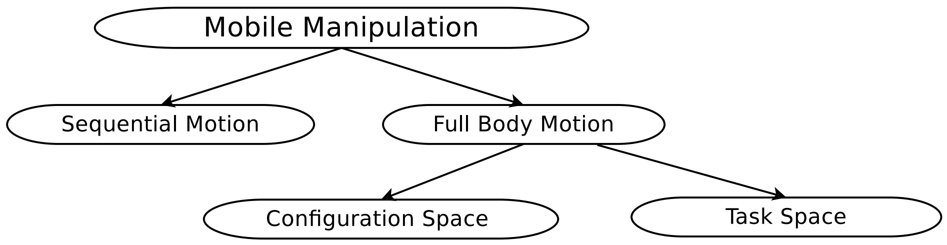


Figure 2-2: Classification of approaches to mobile manipulation planning

is used to traverse this path. Once the designated goal pose for the base is reached, a path for the manipulator is planned such that the end effector reaches its target pose. This path is time parameterized for generating velocity commands using approaches such as iterative parabolic time parameterization [43]. In this approach, motion of the base is considered as *coarse motion* and that of the arm as *fine motion*. Together they constitute the motion plan for the whole robot. This allows the standard techniques used in planning and control of base (eg. [30] [44]) and arm (eg. [45] [46]) to be used individually. *The configuration of the base where transition from base motion to arm motion takes place is called commutation configuration.*

Configuration Space Full Body Planning: With configuration space motion planning, the path is computed in the space of joints describing the robot state. For a 9 dof robot like LEA, path for the 9 dof joints must be planned simultaneously. For mobile manipulators that have omni-directional bases, i.e. can move in x, y and θ coordinates without any non-integrable constraints, the coordinates in SE(2) space can be transformed to those in $\mathbb{R}^3$ [47]. After this transformation, planners that are used for generating paths for a manipulator can also be utilized for planning paths for a mobile manipulator [11] [48]. Figure 2-3 shows an example for whole body motion for a mobile manipulator computed in RViz [49] using Rapidly Exploring Random Trees (RRT) Connect planner [50]. For non-holonomic mobile manipulators, the motion primitives as used in search based planning are utilized for motion planning [51].

Task Space Full Body Planning: Task space planning methods plan in the space describing the task. For reaching a desired end effector pose, the task space dimensionality is six, irrespective of the degrees of freedom of a robot. However, the kinematic redundancy must be resolved by methods such as hierarchal control [52] in order to generate relevant control inputs. This approach is based on the understanding that full body paths for mobile manipulators often have a lower dimensional structure rooted in the end effector path [53]. Unified motion planning and control allows the plans generated to be associated with feedback controllers in form of overlapping funnels. In this approach, a sampling based motion planner such as probabilistic roadmap planner [54] is used to compute the end effector path while connecting the nodes through controllers based on artificial potential field [55]. An example of such an approach is the Elastic Roadmap (ERM) Planner [56].

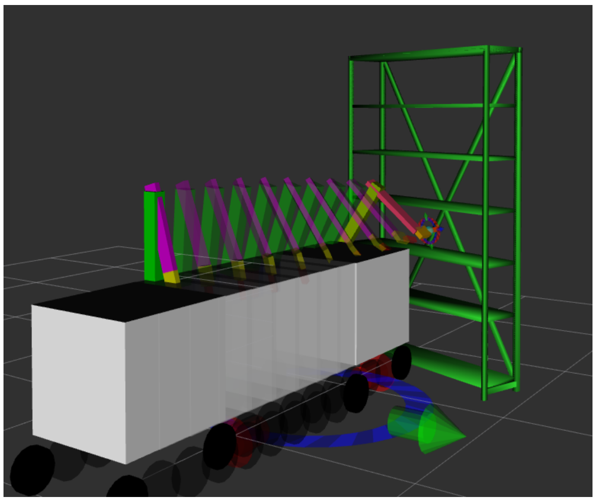


Figure 2-3: Whole-body motion planning for mobile manipulator to reach target pose in a shelf. Performed using MoveIt!

2-2-1 Comparison

The previous section described the mobile manipulation approaches in domestic mobile manipulation. Selection of the appropriate method of the thesis will affect the design of the robot placement approach. If the sequential mobile manipulation approach is deemed favorable, robot placement must ensure a collision free path to the target for the arm. If the configuration space full body approach is selected, the motion of the arm and the platform is considered together. Therefore, the robot placement must ensure a successful collision free path to the chosen location for the arm as well as the platform. With task space planning , the robot placement is performed along with the motion planning of the whole body during the execution of the motion. This section compares the three categories of mobile manipulation for a robot such as LEA.

Table 2-1: Summary of comparison between different aproaches to Mobile Manipulation planning. (Green = Good, Orange = Conditional, Red = Bad). Sequential Mobile Manipulation is found to be the most suitable approach

Criterion	**Sequential**	**Full Body Config Space**	**Full Body Task Space**
Safety	Fewer parts in motion	All parts in motion	All parts in motion, Requires extensive sensors
Computation Time	Platform: 3D planning Arm: Config. Space Planning	High dimensional	Cartesian space
Accounting for Uncertainty	Segregated motion	Coordinated control	Coordinated control
Comfort	Planner choices	Few planners	Controller design

- **Safety:** In terms of safety, sequential mobile manipulation implies less number of parts; either the arm (6 dof) or the platform (3 dof), are in motion at a given time. This leads to smaller chance of collision with other objects as compared to full body motion [12].

Furthermore, in case of task space full body motion, the path of the joints is computed during execution using a reactive controllers. Therefore, an extensive array of sensors on the arm is required for safety. This makes full body mobile manipulation less favorable as compared to sequential mobile manipulation in terms of safety.

- **Computation and Execution Time:** The sequential mobile manipulation planning time depends on the planners used. The platform motion planning generally utilizes A* [57] on a 2D costmap grid [58] and is therefore fast. For the arm, the motion planners using randomization (such as RRT Connect) are faster than those based on optimization (such as Stochastic Trajectory Optimization for Motion Planning (STOMP)). On the contrary, full body configuration space planning is high dimensional and therefore computationally complex [51]. The planning for the task space full body motion is the fastest as it is performed in low dimensional cartesian space. In conclusion both task space full body motion planning and sequential mobile manipulation are computationally faster than fully body configuration space planning. Among these two approaches, task space full body motion has a lower execution time due to all joins being in motion simultaneously.

- **Uncertainty:** In sequential mobile manipulation, path of the arm is planned only after the platform has reached its destination. Any error in platform location can be compensated for by the arm's path ensuring the end effector reaches the desired target. Also, any error in sensing the target's position can be addressed through sequential mobile manipulation because the target can be recomputed with a greater accuracy before planning the arm motion. On the contrary full body motion implies any error in the platform affects the path of the arm as well. Therefore coordinating controllers between the platform and the arm such as [59] must be utilized. In conclusion, the sequential mobile manipulation is best suited for environments with uncertainty.

- **User Comfort:** The user comfort with robot motion depends on the kind of planner used. Motion planners such as STOMP and those part of Search Based Planning Library (SBPL) [60] generate smooth paths consistent over repeated trials. While sequential mobile manipulation allows to choose from different kinds of planners for the platform as well as the arm, the configuration space full body planning has limited lattice based approaches for smooth motion [51]. The task space motion planners such as ERM depend upon randomization, and therefore generate non-smooth paths for the end effector. Furthermore the comfort with motion of the joints depends upon the design of the controller used. In conclusion, sequential mobile manipulation allows the maximum variety of planners to choose from for generating smooth motion plans.

Table 2-1 shows a summary of comparison between the three kinds of mobile manipulation approaches considered. Since safety was deemed the most important criteria, it can be seen that the sequential mobile manipulation planning is the most suitable. This conclusion is further buttressed by considering the remaining criteria as well.

2-3 Robot Placement

In the previous section, the sequential mobile manipulation approach was selected as most suitable for domestic environments. In order to compute the path for the platform it is necessary to determine its goal pose, i.e. the commutation configuration. When the robot is standing at this commutation configuration, the end effector should be able to reach the desired target.

Therefore, the success of a mobile manipulator in grasping an object is dependent upon positioning of the arm in physical space near the object. The planning success also depends on the particular scene and the kinds of obstacles through which the planning is done. For different kinds of scenes, the mobile platform must be positioned in different ways.

With the variety of task a mobile manipulator must perform in context of varying scenes, the purpose of robot placement becomes apparent. Consider the example of picking up a glass of water. The glass must be lifted such that the water inside it does not spill. The dexterity of the arm with respect to the task is dependent upon the positioning of its platform as well. If the robot is positioned too far, it may not reach this glass. Same is the case when it is too close. Even if the pre-grasp pose is reachable i.e. an inverse kinematics solution exists, planning a path from the current configuration of the arm to this desired configuration may be hard. The execution of the grasp and the following tasks is constrained by the controller on the arm, and the particular scene. Incorrect positioning of the platform may imply that a path to this pre-grasp pose cannot be planned. Even if this pre-grasp pose is reached the arm configuration may be such that the further manipulation of end effector to grasp and manipulate the object with given constraints may not be feasible.

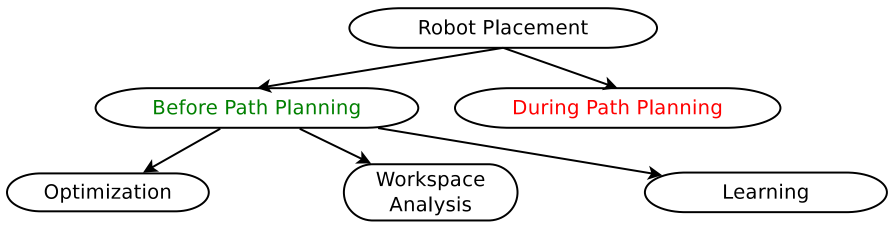


Figure 2-4: Classification of Approaches to Robot Placement

Robot placement methods in literature can be classified into four categories viz. those based on workspace analysis of the manipulator; those utilizing optimization techniques; those relying on machine learning, and those computing the robot placement through path planning. This is illustrated in Figure 2-4.

Optimization Approach: In [11], a co-evolutionary optimization approach is adopted to generate robot placement. The algorithm maximizes the manipulability of the arm, and distance of the robot from nearby obstacles through a measure of clutter. Since the problem of finding the optimal base position with the aforementioned metrics is highly non-linear, evolutionary algorithms are used to avoid local minima. In this kind of optimization based approach, the search space is limited to an annulus around the object of desire. The inner radius of this annulus is given by size of the mobile base. And the outer radius is limited to the maximum possible distance of end effector from the mobile base. One of the researches that use this approach is [12].

Workspace Analysis Approach: Workspace analysis approaches discretize the environment around the manipulator in form of 3D (regions) or 6D (poses) voxels. In [24], a capability map is proposed which quantifies voxels based on the ratio of number of poses that can be reached to the number of poses considered. In mathematical terms,

$$R = D/N$$

where R is the reachability of a voxel, D is the number of poses reached, and N, the number of poses considered in a voxel.

This map can be utilized to generate the robot placement for reaching targets or executing constrained linear trajectories [61] [62]. A Graphics Processing Unit (GPU) based monte-carlo forward-kinematics approach to generating reachability maps for serial manipulators is presented in [63]. This method uses simplified online collision checking considering obstacles as spheres and the robot links as cuboids. [64] presents a hybrid to compute the reachability map by combining the forward kinematics approach with the inverse kinematics approach.

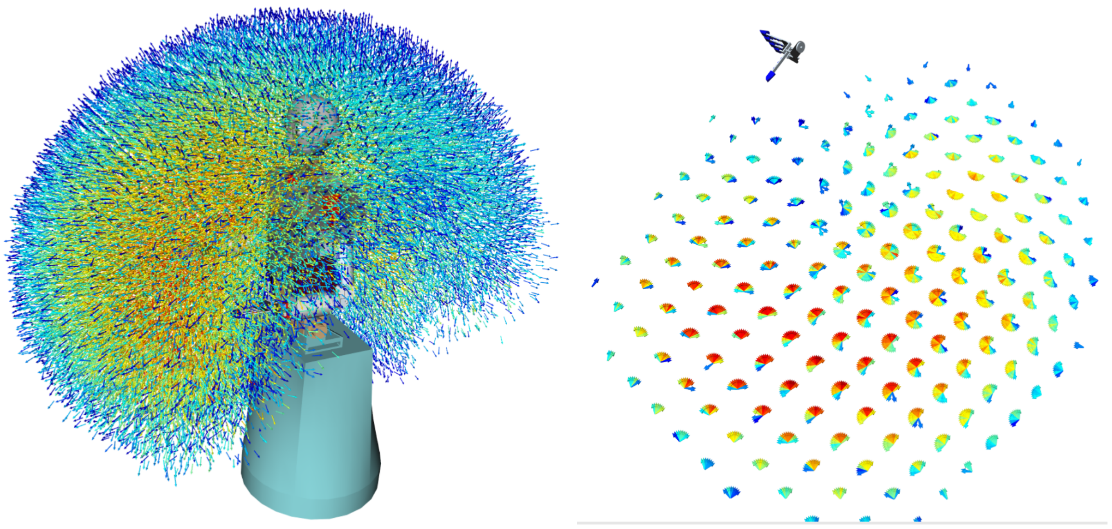


Figure 2-5: 6D Reachability (left) and Inverse Reachability Map (right) for ARMAR-III robot. (Blue: Bad, Red: Good). The robot placement is chosen from the poses. The limitation of this approach is in choosing the pose with highest value when that pose may not be reached accurately. Instead choosing regions with more number of poses lead to higher chance of successful planning to target.
Source: [27]

Similarly, [27] proposes a 6D reachability map that assigns quality values based on the manipulability [34] of its inverse kinematics (IK) configuration. This reachability map is inverted in order to generate possible set of poses from which the robot can reach a given target. To invert the map, a platform to end-effector transformation is computed. This transformation is inverted find poses in space that are candidates for robot placement. Since the robot must stand on the ground, this inverse reachability map is sliced at the ground to generate a list of possible commutation configurations for the robot. Figure 2-5 shows an example of the reachability and the sliced inverse reachability map for the ARMAR-III robot [65]. An iterative search of these poses is utilized to find a collision free robot placement.

Learning Based Approaches: A concept of action relate places (**ARPlaces!**) is introduced in [66] to generate possible set of robot placements through learning for a given scene. The robot perceives the scene through its sensors and performs the locomotion plus manipulation task multiple times in order to generate the region in which the robot can be placed. The **ARPlaces!** are able to account for uncertainty in perception of the target, but require re-training when obstacles are introduced in the environment, or the scene changes. On the contrary, [28] updates the possible robot placements online in presence of obstacles through training for a set of features describing these obstacles in the environment. Thereafter, updated Ease of Reachability Score (ERS) are used for robot placement.

Robot Placement during Path Planning: The previous approaches do not account for the existence of a valid path to the chosen robot placement. In [67], randomized motion planners such as RRT Connect are used to compute the goal during the path planning itself. This allows for choosing a robot placement that is collision free and minimizes a specific metric for the path to this goal. These methods are applicable along with task space motion planning. Since it has been concluded that sequential mobile manipulation is the favorable approach in domestic settings for LEA, this method cannot be used and will not be part of analysis in the next section.

2-3-1 Comparison and Limitations

In this section the limitations of aforementioned robot placement approaches are discussed against the criteria important in domestic settings. Primarily, the approaches will be compared against the criteria listed in section 1-4. A summary of this comparison is provided in Table 2-2.

Table 2-2: Analysis of robot placement approaches in literature against desired criteria. Red: Bad, Orange: Limited Use, Green: Good. None of the methods are able to satisfy all the criteria.

Criterion	**Optimization**	**Workspace Analysis**	**Learning**
Successful Planning	Yes	Yes	Yes (Scene)
Motion Quality	No	No	No
Computation Time	High	Low	Low
Account for Uncertainty	No	No	No

- **Successful Planning:** Optimization approaches focus on maximizing the distance from nearby obstacles and reaching the target pose with a high manipulability. Both these factors allow this approach to increase the chance of finding a successful path at the robot placement in a cluttered environment. Workspace Analysis approaches that use inverse reachability maps are able to compare the possible list of robot placements against the manipulability of target configuration. Since higher manipulability implies more number of ways to move to/from that configuration, there are more ways to plan a path to the target. This way the inverse reachability approaches in literature account for successful planning. While learning based approach like **ARAPlace!** are used to find robot placement that maximizes chances of successful planning, the learning cannot be transferred to a different scene. Ease of Reachability Score (ERS) score approach improves over this aspect. However, it cannot be extended to real life use due to assumptions made about the kind of obstacles, and the features requiring accurate knowledge about the scene. Furthermore, the generalization of proposed features to different scenes is yet to be studied. Since applicability to an unconstrained environment is a primary factor as stated in section 2-1, the learning based approaches cannot be used.

- **Motion Quality (Short Motion):** The methods discussed above only consider the possible list of robot placements on ground against against an indicator for finding a successful plan to that target. However, robot placement also affects the length of path taken by the arm as well as the platform for reaching the target. This is an open area for research and a *planability map* is proposed in chapter 4 towards reaching this ability.

- **Computation time:** The optimization approaches involves a huge amount of collision checking and IK computations. Therefore the computation time for these methods is high. Contrarily, the workspace as well as Learning Based Approaches pre-compute a lot of information about the capabilities of the robot offline (pre-compute). Hence, these methods are able to quickly compute the robot placement online. [Note: The numerical computation time is not being compared due to differences in scenes as well as computing powers of the machines used by researchers. Yet, these conclusions are generally applicable].

- **Account for Uncertainty:** None of the methods apart from **ARAplace!** consider uncertainty of actuation. Optimization approaches compare different robot placements for manipulability and distance from obstacles. Similarly, the 6D inverse reachability approach compares poses as shown in Figure 2-5. This disregards the fact that the robot will never be able to accurately reach the exact commutation configuration. Instead it may be useful to compare regions from where the robot can reach the target. Furthermore, these approaches compare the complete platform poses directly against manipulability, whereas manipulability is a property describing the arm configuration, not the complete system.

In conclusion workspace analysis based approaches are found to be preferred in current use. However, none of the methods are able to satisfy all the desired criteria.

2-4 Summary

This chapter provided an overview of the state-of the art in mobile manipulation for domestic environments. Applicability to unconstrained environment and safety were deemed as primary factors for domestic use.

Based on these criteria, the sequential mobile manipulation planning was found to be most suitable for this thesis. It is safer as compared to full body approach due to less number of parts being in motion at a time. Further, the segregation of platform and arm motion implies error in actuation of the platform does not affect the motion of the arm, and eventual reaching of the target.

Thereafter, the recent literature in robot placement was examined. It was concluded that none of the existing approaches fulfill all the requirements. However, the workspace analysis approach is the most preferred due to applicability in diverse unconstrained environments with a low online computation time.

The next chapter will discuss the design of a sequential mobile manipulation framework for domestic environments. This will later be used in the evaluation of the proposed robot placement algorithm in Chapter 4.

Chapter 3

Sequential Mobile Manipulation Framework

In the previous chapter it was concluded that sequential mobile manipulation approach is the most suitable one for domestic settings. A framework for motion planning and control of mobile manipulators like LEA is presented in this chapter. An overview of the state machine is given and the role of each building block is discussed. The framework also takes into account the real world limitations due to noise and unpredictable changes in the environment. This framework will later be utilized in evaluating the proposed robot placement algorithm. Figure 3-1 illustrates the desired mobile manipulator system. The framework has been designed for the sub-mobile manipulation part of the illustration, when the robot is already in the same room as the desired object. First, the robot uses sensors to localize the object with respect to its current location. This is used to compute a target pose. For the scene and the given target pose, a commutation configuration is generated using robot placement algorithm (discussed in the next chapter). The robot navigates to the selected platform goal, and unfolds the arm to a pre-set configuration. From this configuration, the arm plans a path to the target. In simulation, the target can be assumed as already known, whereas the module on LEA computes it through object segmentation and tracking [68].

Since there are multiple challenges in designing a practical mobile manipulator, utilizing a safe and modular framework is important:

- Safety: This is of utmost concern when working in domestic environment. Fail safes must exist such that the system stops in case of errors or emergencies. This is in accordance with the criteria set in section 2-1.
- Modularity: The different components of sensing, robot placement, locomotion, and manipulation must be independent so that they can be replaced with different approaches as desired. For example, this allows to select different planners for improving user comfort.

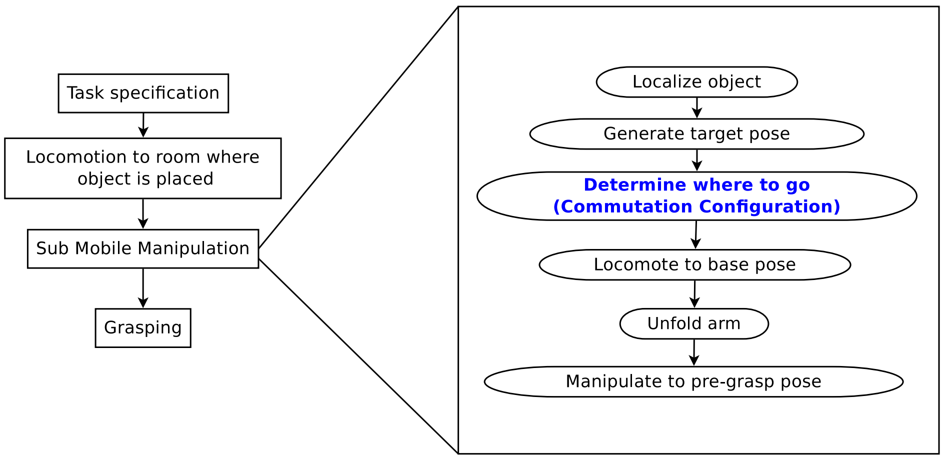

Task specification
Locomotion to room where object is placed
Sub Mobile Manipulation
Grasping
Localize object
Generate target pose
Determine where to go (Commutation Configuration)
Locomote to base pose
Unfold arm
Manipulate to pre-grasp pose

Figure 3-1: Approach to Sequential Mobile Manipulation. The scope of the thesis is within the sub-mobile manipulation problem when the robot is in the same room as the target. Robot placement is used to determine the commutation configuration.

3-1 Framework

Figure 3-2 shows the overall modular mobile manipulation framework for LEA and arm. The framework has been developed in Linux using ROS [31]. The code has been written in C++ and utilizes MoveIt! for motion planning of the arm. ROS Navigation Stack (`costmap_2d` [58]) and Octomap (`octomap_server` [69]) is used for building 2D and 3D maps of the environment. The motion planning for the base can be done using the ROS Navigation stack (currently implemented in LEA), though a simplified method based on parabolic paths and Proportional-Integral-Derivative (PID) control is adopted for simulation. For simulation purposes, Virtual Robot Experimentation Platform (V-REP) [70] is utilized. The system receives information from the 3D camera in form of RGB-D data. This information is utilized to generate the target pose using Tracking-Learning-Detection (TLD) tracker [71] and object segmentation. This is part of the already existing modules with RCS. For this thesis it can be assumed that the target pose is already known.

The state machine is the main component where the decision making takes place. This block computes the robot placement using the target received and scene perceived through its sensors. Thereafter, this platform goal is sent to the motion planner for the base. Using the 2D costmap generated and the odometry computed through simultaneous localization and mapping (SLAM) [72], a path to this goal is computed. Once the robot has reached the goal location, the motion planning for the arm is begun. This takes the end effector to the target pose. The outputs of this state machine are the trajectories of the base and the arm. The present location of the arm and the platform are used as feedback for the base local planner and the PID encoders in the arm. If only the simulation is to be performed, V-REP is used to simulate real-world like settings. This helps in conducting experiments (chapter 5) to evaluate performance of algorithms before testing on the real robot.

The motion commands to the real robot are sent through the driver for the arm and the controller for the base. The output for the base motion planning is the linear and angular velocities of the platform. These are converted to wheel velocities by the controller which

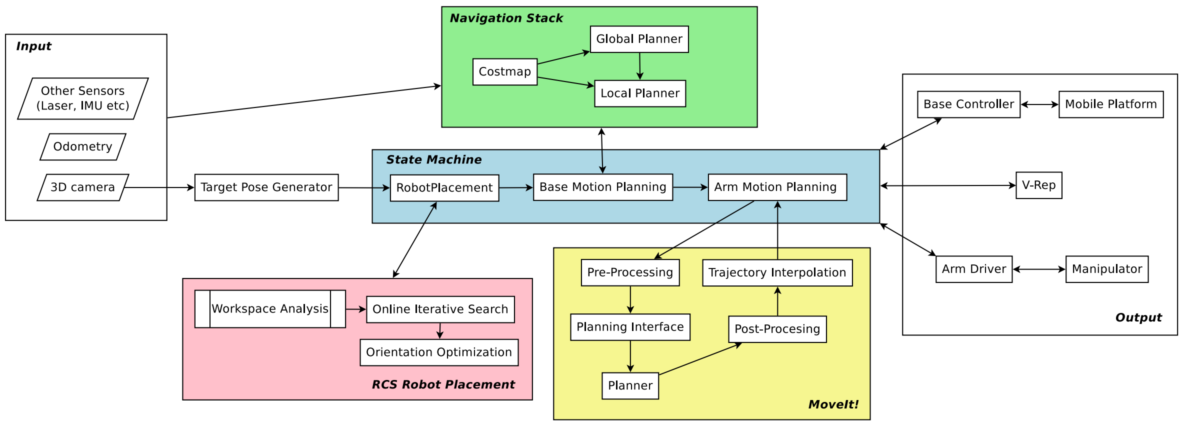


Figure 3-2: Overall Mobile Manipulation Framework. The sensor input is fed into the state machine. The state machine uses the proposed robot placement algorithm DeCOWA, navigation stack and MoveIt!. The output of robot placement is the goal position and orientation for the platform to be reached. This is used by Navigation stack to plan a path for the platform. This block outputs the linear and angular velocity for the platform. Once the robot reaches this location, the arm motion is planned in MoveIt!. It outputs joint position targets that manipulates the arm to the target pose.

leads the robot to the goal. For the motion execution of the arm, position commands are utilized. Position commands sent to the arm are executed using a local PID controller for joints. The arm communicates with the computer using the Controller Area Network (CAN) protocol [73].

In the following section, role of each of these modules is examined. Thereafter a flowchart of decision making for mobile manipulation is provided.

3-2 Building Blocks

3-2-1 Robot Hardware and Modeling

In order to compute paths, and determine robot placement, it is necessary to have the model of the robot. Therefore, a URDF [32] format model is constructed to represent the robot as shown in Figure 3-3. The figure shows the 6 links of the robot attached to the box shaped mobile platform. The shape and size of the arm is in accordance with the specifications as given in appendix A. The model has two primary characteristics:

- Simplification: The model is a simplified version of the real robot geometry. This leads to faster collision checking during path planning as basic shapes instead of complex meshes are considered.
- Inflation: During collision checking using the Flexible Collision Library (FCL) [74], the model is inflated by a fixed amount using MoveIt!. This adds buffer distance between the robot and the obstacles. The buffer distance allows to account for inaccuracies in 3D mapping of the environment while collision checking. According to [29], the maximum error in measuring the point cloud is 4cm and occurs in the depth axis. Therefore, an

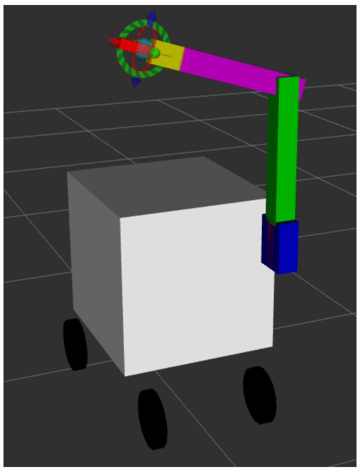


Figure 3-3: Universal Robot Description File Model of LEA with arm. The box represents the LEA mobile platform standing on two rear motor wheels, and two front caster wheels.

inflation of 4cm provides a reasonable buffer against uncertainties of the kinect sensor. One can further include uncertainties due to extraneous causes by taking a factor of safety equal to 1.5. Accordingly,

$$\text{Inflation} = K \times \max(\epsilon_{kinect})$$

where K is the factor of safety, and ϵ_{kinect} is the error in measuring the position of an object. Therefore,

$$\text{Inflation} = 1.5 \times 0.04\text{m} = 0.06\text{m}$$

3-2-2 Robot Operating System

For the realization of the framework, ROS was used which is a flexible framework for writing robot software [31]. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Figure 3-4 shows a representative framework for communicating with the robot via ROS where the robot acts as a server and the laptop is a client connected via Transmission Control Protocol (TCP) Internet Protocol (IP) routing. They have a common ROS master the controls the communication flow.

This allows for concurrent resource handling implying shared control of sensors, actuators, and robust information flow. It also allows for inter-platform interoperability implying applicability to multiple robots. Furthermore, ROS is modular and hence allows to replace component modules if needed. Finally, the open-source nature of ROS makes it an attractive platform for robot development.

In the thesis, communication between the modules is controlled through the subscriber, publishers, and nodes. The state machine module subscribes to topics publishing the sensory and odometry information. Thereafter the generated paths are published on topics. The `move_base` node is uses this information to compute the path for the mobile base. And the

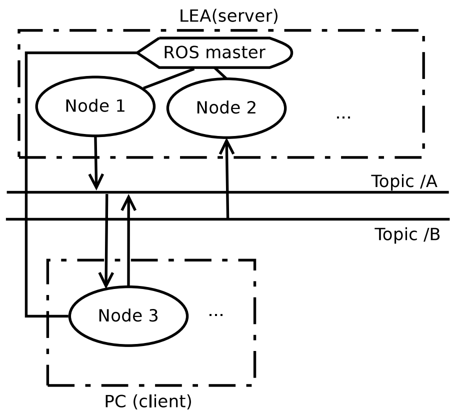


Figure 3-4: ROS communication framework

`move_group` node computes the path for the arm. Those topics are subscribed by another node that implements the velocity commands and the joint positions in V-REP or the robot drivers.

Having described how the different modules communicate with each other, the next subsections will briefly discuss the modules for motion planning of the base and the arm.

3-2-3 Base Motion Planning

As stated earlier, the standard way of mobile motion for wheeled platforms like LEA uses the ROS Navigation stack, specifically the node `move_base`. In this subsection the standard approach to motion planning of the platform is discussed. Thereafter another approach that finds a parabolic trajectory to the goal and traverses it using PID control is presented. Since either of these methods can be used interchangeably for base motion planning, it showcases the modular nature of the mobile manipulation framework.

ROS Navigation Stack Navigation stack consists of the components required to move a mobile robot while avoiding obstacles. Given a cost map, the current and desired goal locations of the robot, a global planner generates the path in 2D discretized grid space. This path generation is done using standard node based planners such as A* or Dijkstra, and does not take into account the non-holonomic nature of the robot.

Hence, a local planner is needed that accounts for non-holonomic constraints. This planer modifies the path locally to make it feasible such that it stays in a neighborhood of designated path. For the local planner Trajectory Rollout and Dynamic Window Approach (DWA)[75] are popularly used.

ROS navigation stack is very useful in computing paths for a mobile robot over long distances across multiple rooms in a house. However, for shorter distances, the global planner generated path may only be traversed approximately. This implies that often the robot can only reach a neighborhood of desired goal configuration. This is especially true for robots like LEA where the front caster wheels make accurate path following or pose stabilization more challenging. Therefore a tolerance value to the goal pose is used. The goal tolerance in x and y axis is set at 0.1m as default. Similarly, the default tolerance for goal yaw is 0.05 rad. The tolerance values will be used in the next chapter while developing the robot placement algorithm. The approach to robot placement has to be designed considering this limitation of the navigation stack.

Parabolic Path with PID control In this approach a parabolic path between the starting pose of the mobile platform and the goal is generated. The path is computed such that the goal is at the vertex of the parabola. A parabola is used as it provides an unambiguous trajectory of the robot for any given starting and intermediate pose, unlike other conic sections. This is shown in Figure 3-5.

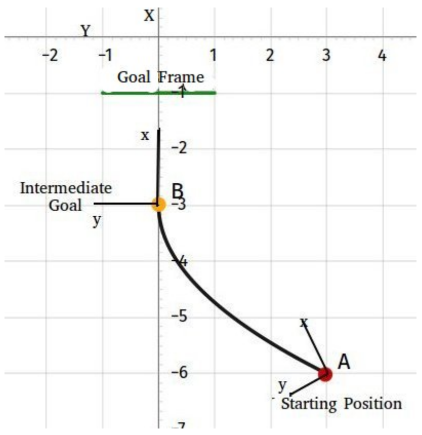


Figure 3-5: Parabolic path to target in goal's coordinate frame

To traverse the parabolic path, angular and linear velocities for the robot have to be generated. These velocities are output from the state machine shown in Figure 3-2. To do so, a simple PID controller is designed with velocity clipping as shown in Figure 3-6.

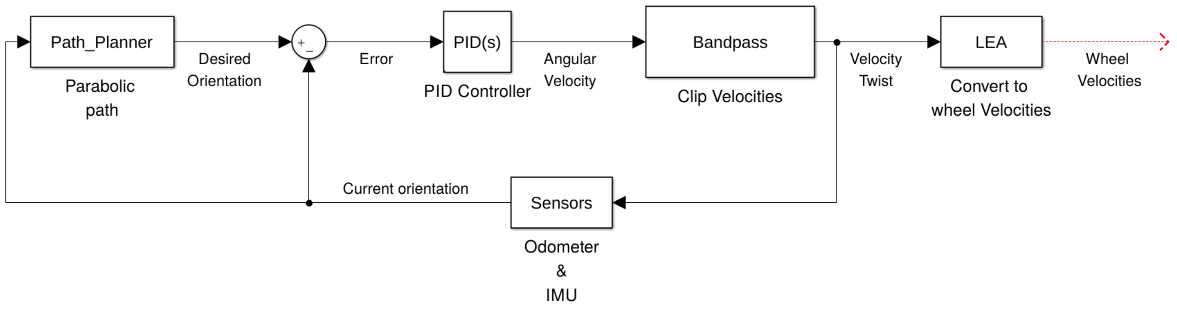


Figure 3-6: P(I)D control approach to motion planning of the base

Appendix B provides a mathematical formulation for generating and traversing this parabolic path.

3-2-4 Arm Motion Planning

The motion planning for the arm is done for a highly unconstrained domestic environment. MoveIt! is a software platform that allows to solve motion planning problems for manipulators. While the Open Motion Library [46] is generally utilized with MoveIt!, other motion planners may also be incorporated in the modular framework. Three motion planning libraries are incorporated in the framework using the `planning_family_manager` of Search Based Planning Library (SBPL) [60] as an interface with MoveIt!.

- **Open Motion Planning Library (OMPL):** OMPL is one of the most widely used planning libraries. Sampling based, combinatorial, graph based, optimization and control based motion planning algorithms are part of this library.
- **Search Based Planning Library (SBPL):** SBPL is a library of planners specifically using graph search methods to compute paths over discretized space generated through motion primitives.
- **Stochastic Trajectory Optimization for Motion Planning (acsSTOMP):** This planner is specifically designed to generate smooth motion plans through stochastic optimization. Here, smoothness of a plan is defined using the sum of squared accelerations along the trajectory. It generates a simple trajectory between two configurations and multiple perturbations of the same to optimize the cost function.

Table 3-1 provides a summary of advantages and disadvantages of using each of these libraries. Based on the scene, a particular planner can be chosen from the set of libraries to generate a successful and comfortable motion plan.

Table 3-1: Strengths and Weaknesses of motion planning libraries

Planning Library	**Advantages**	**Disadvantages**
OMPL	- Faster planning - Diverse set of planners - Can be used for non-holonomic planning	- Randomized paths possible - Poor orientation constraint handling
SBPL	- Consistent motion generation - Motion primitives can be used - Bounded optimization of cost function	- High planning time - Poor obstacle avoidance
STOMP	- Smooth motion plans - Effective constraint handling	- High planning time - Only useful for local planning

3-2-5 Simulation Environment

Before implementing the framework on the robot, it was tested on a simulation platform. For simulation purposes, V-REP was utilized which is a tool by Coppelia Robotics [76]. This was chosen since it provides the ability to model the environment similar to those at homes by adding obstacles, household items, and sensors on the robot. Also, certain Application Programming Interfaces (APIs) such as ROSInterface [77] and ROSPlugin [78] allow V-REP to interface with ROS nodes and vice verse. V-REP was selected over Gazebo [79] since it is more intuitive, offers easy integration with ROS, has more number of standard household items in its library used to build a scene similar to real world, and is less hardware intensive [80]. For working in the simulation environment, the joint states and the odometry information is taken from V-REP which does not include measurement errors. Figure 3-7 shows an example of simulated environment in V-REP.

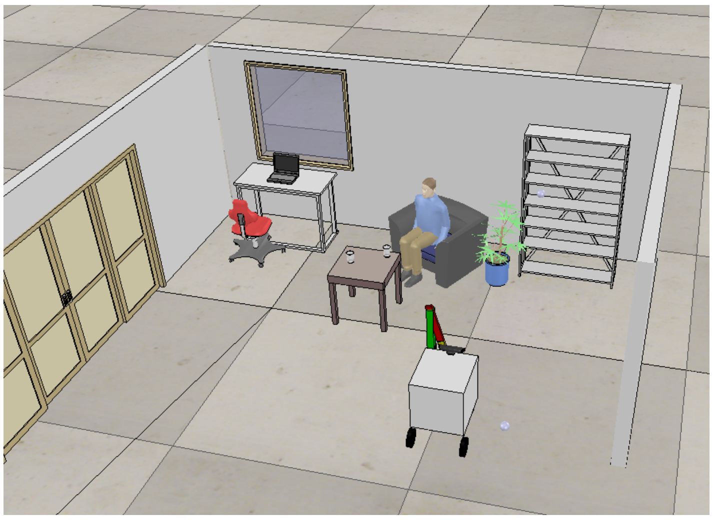


Figure 3-7: Simulation Envrionment of V-REP

3-3 Overview

To supervise the decision making in mobile manipulation, state machine as shown in Figure 3-2 is used.

- Robot Placement: This process determines the commutation configuration for the mobile manipulator. If the robot is not able to find a suitable commutation configuration, the process stops and returns failure.
- Compute Path and Navigate to Platform Goal: The robot uses the base motion planners to compute a path to the goal location. If a path is found, then the robot navigates to the commutation configuration. Otherwise, robot placement is attempted again.
- Unfold: After reaching the desired commutation configuration, the arm unfolds to a set configuration. This can be done using pre-determined joint commands and does not require motion planning.
- Select Planner: Using the planners family manager, any one of the suitable planners may be selected. Usually this is pre-specified in the code. However, a contextual awareness approach was implemented for the arm in [68] that selects the appropriate planner based on the scene.
- Compute Path and Manipulate: Once the planner is selected, it is used to compute the path for the arm and manipulate to the desired target. If the motion planning is not successful, a new pre-grasp pose may be recomputed and the process of motion planning is repeated for a fixed number of times.
- Grasp: Once the pre-grasp pose has been reached, the state to navigate to is grasp. If this is unsuccessful, a new pre-grasp pose must be selected.
- Place: Once the object is grasped, the sequence to place it on the robot is initiated.

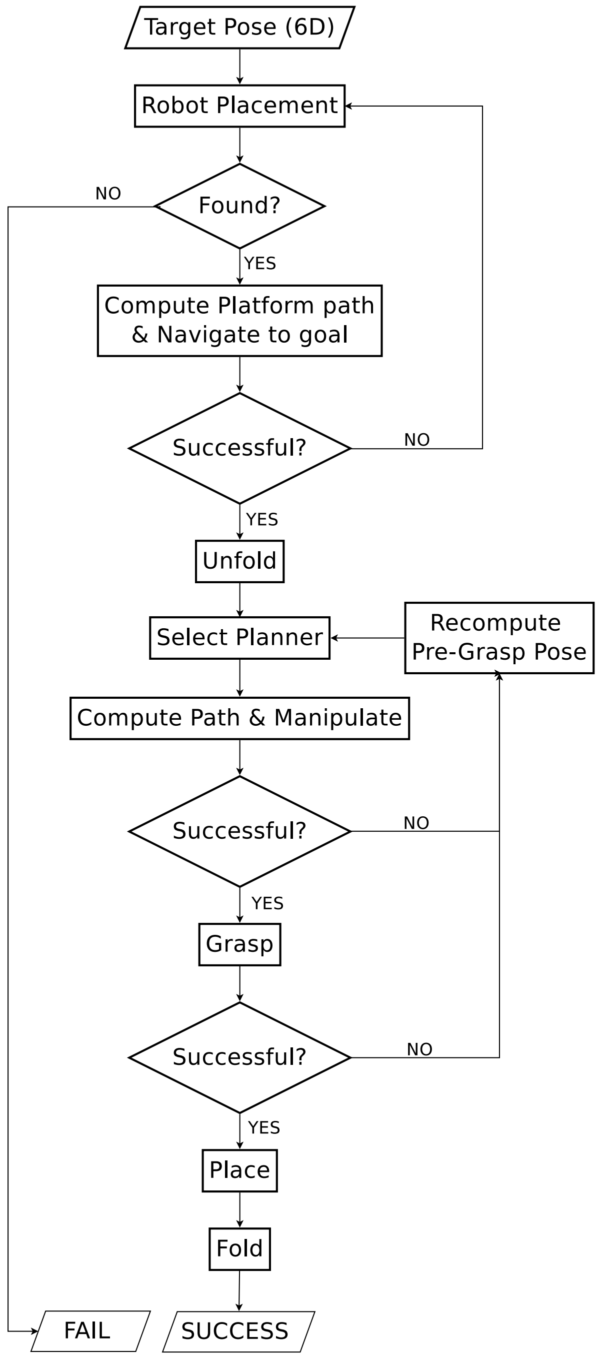


Figure 3-8: Flowchart describing the state machine for mobile manipulation

- Fold: Once the object has been picked up and placed atop the robot, the arm is folded back.

In this thesis, it is assumed that the pre-grasp pose is known. The focus is on determining the robot placement and using it to move the end effector to the chosen target. In each of these states, if there is an unexpected exception, the robot comes to a stop. The problem of grasping is a separate field of research. Grasping depends on the kind of object, the kind of gripper being used, and the available grasp poses. This is not part of the present research. Furthermore, it must be noted that the propsed framework does not respond to dynamic obstacles. The path is computed for the scene at that instant. During motion, the environment is not re-analyzed. Therefore it may even happen that the chosen commutation configuration is no longer available while the platform is in motion. This can be considered in future works.

3-4 Summary

In this chapter a mobile manipulation framework for domestic robots like LEA is presented, and its modules are discussed. The state machine is composed of robot placement method, the motion planner for the base, and the motion planner for the arm. These interact with the robot through drivers using sensors to perceive information and make decisions. In order to increase safety of the robot against sensor uncertainty, a simplified and inflated URDF model is used. It is shown that the system is modular and allows for different motion planners for the platform and the arm to be used. This modularity is useful in selecting planners that produce comfortable paths for the users. Also, the framework allows to recompute the robot placement in case a path to the chosen commutation configuration is not found.

Having presented an overview to the developed mobile manipulation framework, the next chapter elaborates the robot placement algorithm for use in domestic environments.

Chapter 4

DeCOWA: Robot Placement Algorithm

In the previous chapter the mobile manipulation framework for domestic robots was developed. All the modules of the state machine were discussed except for robot placement. This chapter elaborates the proposed robot placement algorithm, named **DeCOWA** i.e. **De**termining **C**ommutation-configuration using **O**ptimization and **W**orkspace **A**nalysis. To recap, it is desired that the robot placement must be such that there is a high chance of successful planning to the target pose; the path taken to the target is short; the method accounts for possible errors in sensing and actuation; while maintaining a low computation time.

The input available to generate this commutation configuration is the planning scene as visualized through the sensors, the URDF model of the robot, the end effector target pose, and the aforementioned set of criteria. This is illustrated in Figure 4-1.

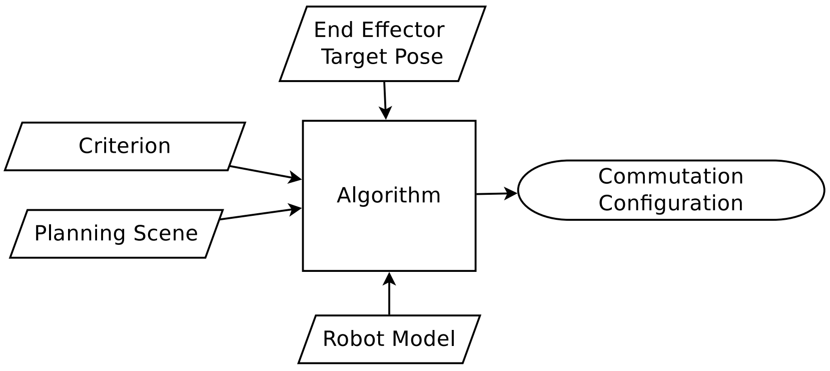


Figure 4-1: Inputs and outputs of the desired robot placement algorithm

In the next section an overview of the proposed algorithm is provided. Thereafter each of its components is discussed in detail.

4-1 Overview

DeCOWA is composed of two parts as shown in Figure 4-2. In the first part a favorable location for placing the robotic arm is determined. Thereafter, the best orientation of the platform at this location is computed. Unlike existing approaches, it is recognized that the success of manipulation is dependent only upon the location of the arm and not on the orientation of the platform since it is static. However, any error in actuation of the platform, ergo its accuracy in reaching the commutation configuration implies that the robotic arm will also not be able to reach the chosen location accurately. Therefore, to decide the arm location, different regions in space are compared for possible placement rather than individual poses.

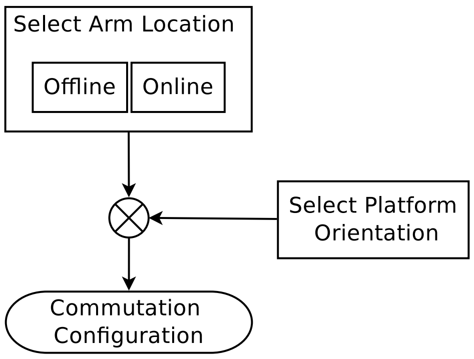


Figure 4-2: Composite approach to robot placement. Determining arm location followed by finding platform orientation. Procedure for selecting arm location is split into offline and online components.

The procedure to determine a favorable arm location is further divided into two parts; offline and online. The offline part focuses on computing the *reachability map* of the manipulator. The quality of 3D voxels in the reachability map represent the number of ways the robot can reach the particular region. Similar to this map, a *planability map* is constructed that indicates the minimum path length (measured through total joint motion) to each voxel. By considering the reachability or planability as a probability of the voxel being reachable with a short path, these maps can be interpreted as probability distributions. Thereafter, these maps are inverted to generate inverse metric maps. The inverse reachability map contrasts different regions based on the number of ways a manipulator placed in that voxel can reach the origin. The inverse planability map acts as an indicator for comparison of minimum path length of the arm to origin from starting configuration.

In order to consider the path length of manipulation as well as a high success rate during robot placement, the inverse reachability and planability maps are fused into an inverse fusion map online. Thereafter, this map is iteratively searched for a collision free arm location in decreasing order of its quality, where it can reach the target. For such an arm location, the platform orientation is determined by minimizing the distance from the current robot position. This allows to locally minimize the amount of platform motion while ensuring successful planning for the arm. As soon as a collision free inverse kinematic solution for the arm and platform orientation is found, the algorithm terminates. Since the offline component contains substantial computation, it allows for rapid online generation of the commutation configuration $\mathbf{q_b}^{goal}$.

In the following sections each of these parts are discussed in detail. First, the offline component of robot placement is elaborated. Thereafter, the online utility of this data to determine the commutation configuration is presented.

4-2 Offline Computation

In this section the offline component of selecting the arm location is detailed. Pre-computation of reachability, planability, and inverse fusion maps is elaborated.

4-2-1 Reachability Map

The reachability maps reflect the working capabilities of a manipulator. It is a qualification of regions around a manipulator based on its geometry and kinematics. The reachability map for a manipulator needs to be generated only once for a given resolution. To generate this map, the method from [24] is adopted. Part of the code has been adopted from [81]. Unlike [81], joint limits and self collisions are also considered in this thesis while generating reachability maps. The following paragraphs explain the procedure for creating this map (as illustrated in Figure 4-3).

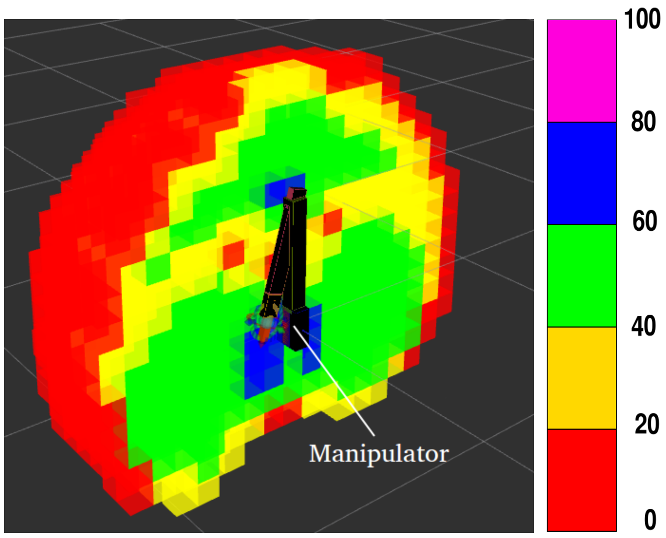


Figure 4-3: Disected Reachability map for arm at resolution 0.1. The figure shows the robotic arm in the center in black. The cubic voxels around it depict the reachability map with colors ranging from Red = Low Reachability (<20%) to Blue = High Reachability (> 80%). The exterior voxels have < 20 reachability, which increases on moving closer to center.

Voxelization: The first step is voxelization of the arm's workspace. Based on the maximum reach of the manipulator, a cubical envelope to the workspace is considered. The region within this envelop is discretized into smaller voxels using octree [82], a hierarchical data structure enabling spatial partitioning, searching, and downsampling of the voxels. In order to select the size of the voxels, it is important to understand the philosophy behind utilizing 3D voxels. The purpose of using 3D instead of 6D voxels is to account for pose stabilization errors of the mobile base. From chapter 3, it is known that the DWA local planner [75] has a default goal tolerance of 0.1m. This means as soon as the mobile platform is within this range of the goal, the motion execution is considered successful and complete. Since the base of the arm is rigidly attached to a mobile manipulator, the error in positioning the arm would be the same as the error in platform reaching the commutation configuration. Therefore, it is decided to voxelize the workspace into cubes of 0.1m each. In general, the resolution of these maps is a free parameter and left up to user's choice. With a smaller voxel size, the robot placement would be much finer. However, as can be seen in Figure 4-5, the computation time increases exponentially with greater discretization. This is because the number of reachable poses and voxels increase with smaller voxel size as depicted in Figure 4-4.

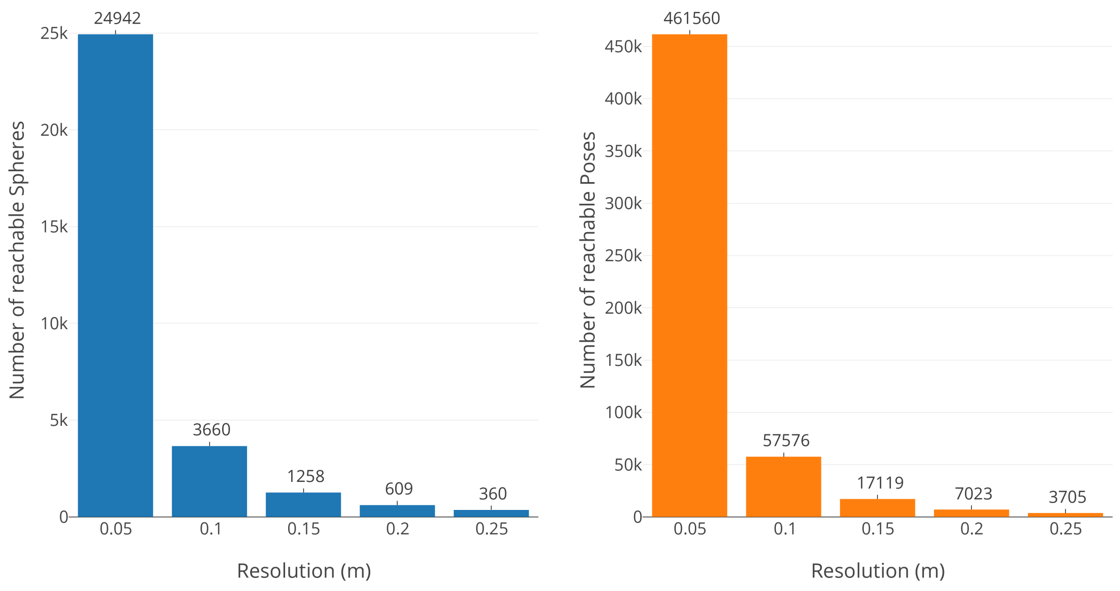


Figure 4-4: Reachability Map characteristics for different resolutions. Left: The number of reachable voxel/spheres against map resolution. Right: The number of reachable poses against map resolution. Shows exponential increase in size of reachability map with decrease in voxel size.

Sampling poses in the workspace: Once the workspace has been voxelized, the next step is to sample the number of poses that the manipulator can reach in each of these voxels. The number of poses reached is used to evaluate a voxel's quality. There are two possible approaches to sampling poses. The first method involves monte-carlo sampling [83] in the configuration space of the arm, and thereby generating poses using forward kinematics (FK). The number of hits for each voxel would indicate the volume of joint space that maps to the particular voxel. The second approach is to consider a fixed set of poses in each voxel and then use inverse kinematics (IK) to compute the ratio of poses reached to the poses considered in each voxel. The forward kinematics approach suffers from two drawbacks. The first drawback is that it requires a large number of forward simulations in order to generate enough data

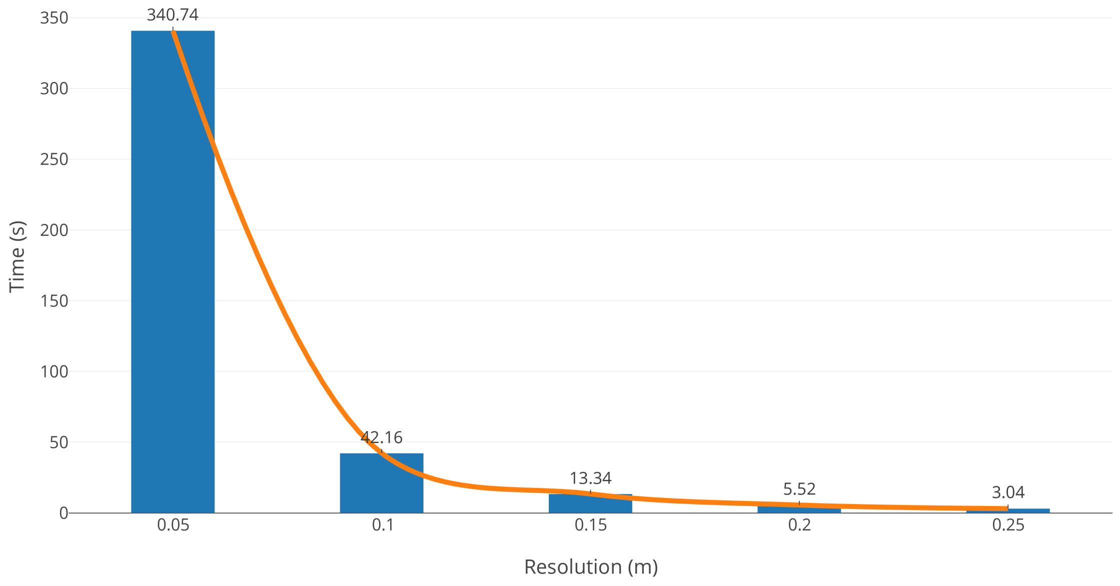


Figure 4-5: Average computation time for Reachability Map at different resolutions. Orange curve shows interpolated computation time for these resolutions. An exponential trend is observed.

that can be useful for drawing comparative between the voxels. Secondly, even after a large number of monte-carlo simulations, the singular configurations are not handled appropriately by this approach [27]. This implies most of the FK sampling would miss out on voxels in the outer regions of the workspace. On the contrary, IK based approach is able to compute the quality for all the voxels. Therefore, the second approach of using IK sampling has been chosen for DeCOWA.

For each voxel, a sphere having diameter equal to the map resolution is created in it. Thereafter, a uniform number of points are sampled on the sphere using [84]. For each point on the sphere, a pose towards the center of the voxel is considered as shown in Figure 4-6. Using IKFast [85] library for inverse kinematics, the poses reached by the arm corresponding to the voxel center are stored in a datastructure. This is used to compute the reachability of the voxel.

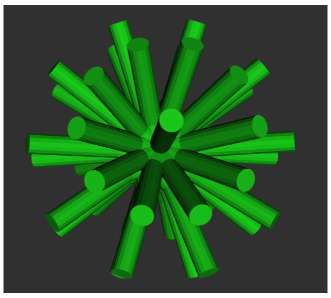


Figure 4-6: Sampling of poses on sphere pointing to the center of a voxel. A total of 50 poses per sphere/voxel are used. Using IK evaluations to each of these poses, the reachability of the voxel is calculated.

Measuring Reachability: Reachability of a voxel is given by the ratio of number of poses reached by the manipulator to the number of poses considered.

$$R_i = \frac{D_i}{N} \cdot 100 \ \forall \ i = 1 \ldots V$$

where D_i is the number of poses in the ith voxel sphere for which an inverse kinematics solution is found, N is the number of poses considered in each voxel, and R_i is the reachability of the voxel v_i. In this thesis, the value of N has been taken to be 50 poses. The total number of voxels is given by V. Finally each voxel center is associated with its reachability and the reachable poses and stored. Algorithm 1 summarizes the method for creating a reachability map. This map is stored in form of the hierarchical data file (hdf5)[86].

Algorithm 1 Generating Reachability Map

```
1:  create $V$ voxels
2:  for each voxel $v_i$ in $V$ do
3:      create sphere $s_i$
4:      uniformly sample $N$ poses on sphere $s_i$
5:      for each pose $p_j$ on $s_i$ do
6:          find IK solution to $p_j$
7:          if IK solution found then
8:              store $<s_i, p_j>$ in hdf5
9:          end if
10:     end for
11:     measure reachability $R_i$ for sphere $s_i$
12:     store $<s_i, R_i>$ in hdf5
13: end for
```

As an example, Figure 4-3 had shown the reachability map computed for the RCS arm at resolution of 0.1m . It can be seen that the map is symmetric about the zx and zy planes. The voxels at the outer layer of the map are red, i.e have lower reachability because the arm is near its maximum reach. For another URDF, the map will be different based upon its joint limits and kinematics. However, since most arms are symmetric about one of the axes, the same will be the case with the generated reachability maps.

4-2-2 Planability Map

Based on the requirement of having a short manipulation path as elaborated in chapter 2, a new indicator of such quality i.e. planability is proposed inspired from the reachability map. For the creation of a planability map, the method for voxelization and sampling poses in the workspace remains the same. However the measure of planability is different than that of reachability.

Since the duration of motion is dependent on the path length, the minimum total joint motion to each reachable pose is computed and stored as the cost of each reachable 6D pose. This is computed in free space using an optimizing planner such as STOMP that generates smooth and consistent trajectories. On experimentation, it was found that in free space, STOMP would always generate shortest paths of same length with negligible variance of $\mathcal{O}(10^{-3})$ rad.

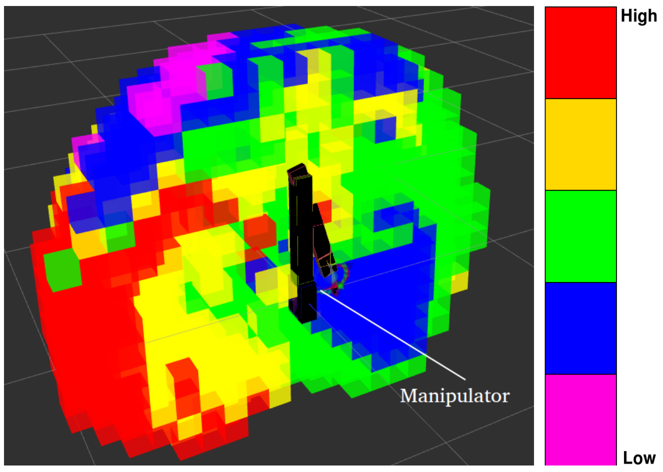


Figure 4-7: Disected Planability map for arm at resolution 0.1m. Note: The costs have been scaled to lie between 0 rad and 100 rad for visualization. Colors range from Red = High cost (>80 rad) to Magenta = Low cost(<20 rad) The arm is denoted in black in the center.

In order to use STOMP to generate the planability map, its parameters were tuned as follows.

- `Num_time_steps`: Represents the number of joint configurations per trajectory. Increasing this number makes the trajectory smoother at the expense of higher computational cost. Since the planning is done in free space, this was set to 20, a relatively low value.
- `Max_iterations`: The maximum number of optimizing iterations STOMP is allowed to perform before finding a collision free trajectory, after which it will report failure. This was set to 10 even though STOMP was able to find optimum path within 1 to 2 iterations in free space.
- `Num_rollouts_per_iteration`: The number of noisy candidate trajectories considered in each optimization iteration so as to find a better trajectory. This was set to 5 since the space is empty and there is no need to perturb trajectories for obstacle avoidance.
- `Noise_coefficients`: These coefficients are used to randomize the joint values during the generation of noisy candidate trajectories for optimization. These were set at 0.1 rad for each joint to generate smooth paths.

Mathematically the path length in joint space is given by,

$$c_i = \sum_{s=1}^{k-1} ||(\theta_{\mathbf{k+1}} - \theta_{\mathbf{k}})|| \; \forall \; k \; path \; segments. \tag{4-1}$$

Here $\theta_{\mathbf{k}}$ represents the vector of kth path segment, and $||\cdot||$ represents its norm.

Using this, the cost of a voxel in manipulator's workspace is given by the average cost of all the reachable poses. In mathematical terms,

$$C_i = \frac{\sum_{j=1}^{D_i} c_j}{D_i} \; \forall \; i = 1 \ldots V \tag{4-2}$$

where C_i is the cost of voxel v_i, D_i is the number of poses reachable in the voxel, and c_j is the cost to reach pose p_j in voxel v_i. Algorithm 2 summarizes the steps taken to generate the planability map.

Algorithm 2 Generating Planability Map

```
 1: create $V$ voxels
 2: for each voxel $v_i$ in $V$ do
 3:     create sphere $s_i$
 4:     uniformly sample $N$ poses on sphere $s_i$
 5:     for each pose $p_j$ on $s_i$ do
 6:         find IK solution to $p_j$
 7:         if IK solution found then
 8:             compute path from start configuration to $p_j$
 9:             if planning successful then
10:                 compute path length $c_j$ for reaching $p_j$
11:                 store $<p_j, c_j>$ in hdf5
12:                 store $<s_i, p_j>$ in hdf5
13:             end if
14:         end if
15:     end for
16:     measure planability cost $C_i$ for sphere $s_i$
17:     store $<s_i, C_i>$ in hdf5
18: end for
```

The planability map for the RCS arm with resolution of 0.1 is shown in Figure 4-7. It can be seen that the voxels near the starting configuration have a lower path length. As one moves backward, the first joint will be moved which would lead to huge cost of motion. Therefore, the voxels on the back have more higher cost. Figure 4-8 shows the number of spheres and poses for the planability map at different resolutions. It can be seen that as the discretization decreases, the data to be stored reduces exponentially. Similarly, Figure 4-9 shows the amount of time required to compute the planability map in minutes. With a decrease in discretization, the time required to compute the planbility map reduces exponentially. It must be noted that while the computation of reachability maps can be completed in a few seconds to some minutes, computing the planability map takes some minutes to several hours.

4-2-3 Inverse Metric Maps

In order to compute the possible set of arm locations from which the target pose is accessible, the inverse reachability approach [27] is extended to regions (3D voxels) in space. This allows to accommodate for actuation uncertainties affecting practical utility of the algorithm. Similarly, the inverse planability map is built to find locations in space from which the minimum joint motion to target is small. These maps are referred as Inverse Metric Maps, as they can be generalized to other metrics of motion quality as well (eg. minimum energy consumption). The following subsection details the creation of an inverse reachability map.

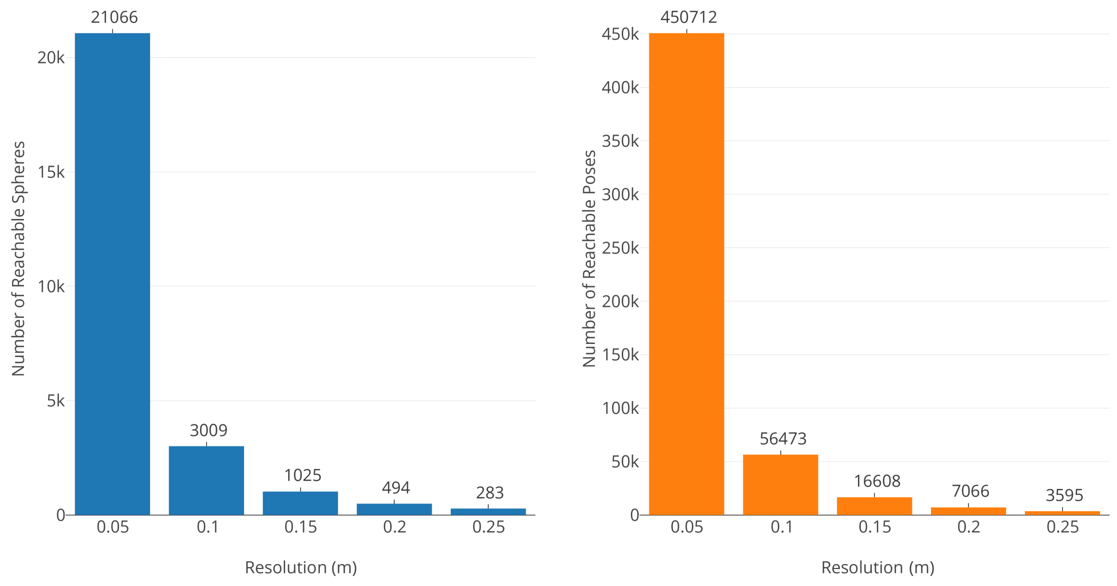


Figure 4-8: Planability Map characteristics for different resolutions. Left: The number of voxel/spheres to which a plan was found against map resolution. Right: The number of poses to which a plan was found against map resolution. Shows exponential increase in size of planability map with decrease in voxel size.

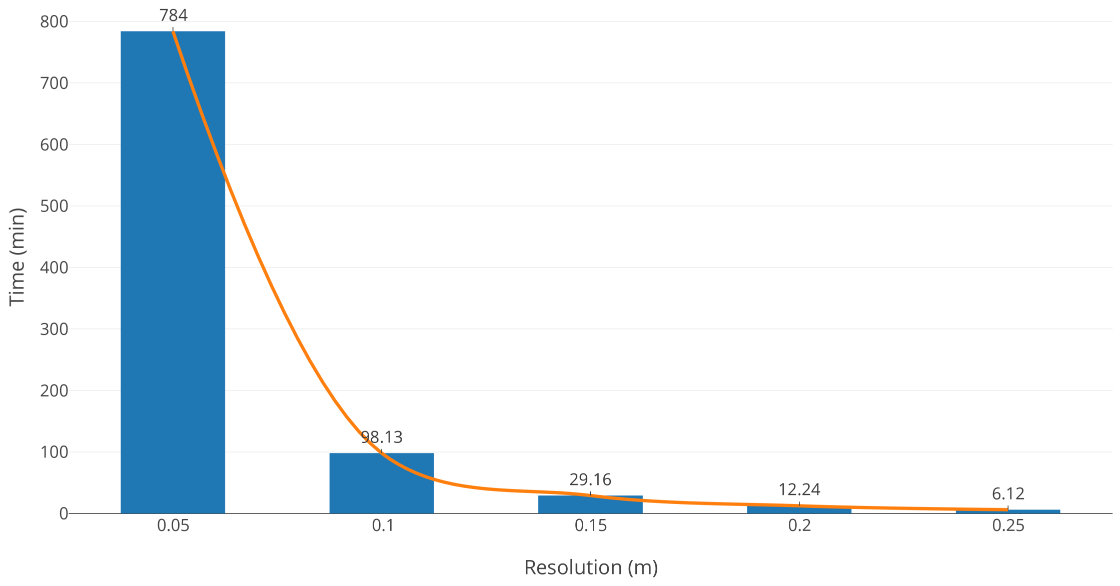


Figure 4-9: Average computation time for Planability Map at different resolutions. Orange curve shows interpolated computation time for different resolutions. An exponential trend is observed.

Inverse Reachability Map (IRM)

The inverse reachability map is computed by inverting the affine transformation from origin to the all the reachable poses. This generates a set of poses in space from where the arm is capable of reaching the origin.

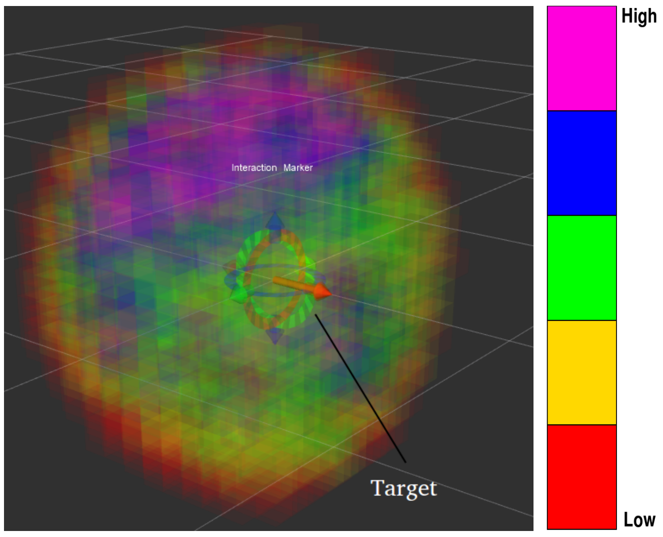


Figure 4-10: Inverse Reachability map for arm at resolution 0.1m. The center of image shows the target pose at origin. The colored voxels around it indicate the number of ways of reaching the target when the arm is placed in the voxel. Colors range from Red = Low Success Likelihood ($< 20\%$) to Magenta = High Success Likelihood ($> 80\%$) in reaching pose at origin.

The first step to creating the inverse reachability map is the same as that of creating the reachability map. The workspace is voxelized at the resolution of the reachability map (eg. 0.1m). Thereafter, for each pose p_j that was reachable, a transformation T^{EE}_{origin} is found, using the end effector coordinate and orientation.

$$T^{EE}_{origin} = \{x, y, z, \rho, \psi, \phi\}.$$

Here, *origin* is the odometry global frame (taken as $\mathbf{0}$), and EE is the frame of the end effector. This transformation is inverted to get

$$T^{origin}_{EE} = (T^{EE}_{origin})^{-1}$$

T^{origin}_{EE} is used to get the poses from which the arm can reach the origin. Next, these poses must be combined into a 3D inverse reachability map, in order to account for actuation uncertainty. In order to do so, the following steps are taken:

- The translation components (x, y, z) of the poses are stored as point cloud [87].
- Point Cloud Library's (PCL) nearest neighbor (NN) search is used to find the number of points that fall within a given voxel [88].

Based on this information the inverse reachability score is given by,

$$R_i^{inverse} = \frac{D_i^{inverse}}{N}, \forall\ i = 1 \dots V$$

where N is the maximum number of points that can lie within the voxel, and $D_i^{inverse}$ is the number of points found to be within the voxel. This ratio indicates the likelihood of finding a successful path to the origin when the arm is placed at this location. In other words, higher $R_i^{inverse}$ implies the reachability of voxel at origin is high when the arm is placed at voxel v_i.

Algorithm 3 Generating Inverse Reachability Map

```
1: create $V$ voxels
2: read reachability map stored in memory
3: **for** each reachable pose $p_j$ in reachability map **do**
4:     compute origin to end-effector transform $T_{origin}^{EE}$
5:     invert transform to compute $T_{EE}^{origin}$
6:     store translation part of inverse transform in a point cloud *pcl*
7: **end for**
8: **for** each voxel $v_i$ in $V$ with voxel center $s_i$ **do**
9:     use nearest neighbor within voxel search to identify number of *pcl* points within $v_i$
10:    compute inverse reachability score $R_i^{inverse}$
11:    store $<s_i, R_i^{inverse}>$
12: **end for**
```

Algorithm 3 summarizes the method for computing the inverse reachability map. It must be noted that the success of a manipulator in reaching an end effector target is dependent upon the scene as well as the planner used. However, when the arm is placed at a location of high inverse reachability, there are more number of ways to reach the origin which indicates to a higher chance of planning success in presence of obstacles as well. Figure 4-10 shows the inverse reachability map computed for the RCS arm with resolution 0.1m. It is interesting to note that the shape of the inverse reachability map is not a sphere. This is understood since the specific pose at origin cannot be reached when the arm is placed further ahead of the origin.

Inverse Planability Map (IPM)

The previous subsection described the procedure to compute the inverse reachability map. Next, the method of generating the inverse planability map is discussed. To compute the inverse planability map, the planable poses are used to construct transformation from origin to the end effector, in the same way as was done for the inverse reachability map. This transformation is inverted to compute the locations where the arm can be placed in order to reach the origin.

The important distinction from inverse reachability map is that each of these points (translation components of inverse transform) is assigned a cost value associated to the original pose. Then the cost of a voxel of inverse planability map is given by

$$C_i^{inverse} = \frac{\sum_{j=1}^{D_i^{inverse}} c_j}{D_i^{inverse}} \;\forall\; i = 1 \dots V \tag{4-3}$$

where $C_i^{inverse}$ is the cost value associated with the voxel v_i, $D_i^{inverse}$ is the number of points in the voxel, and c_j is the cost associated to the each point in the voxel.

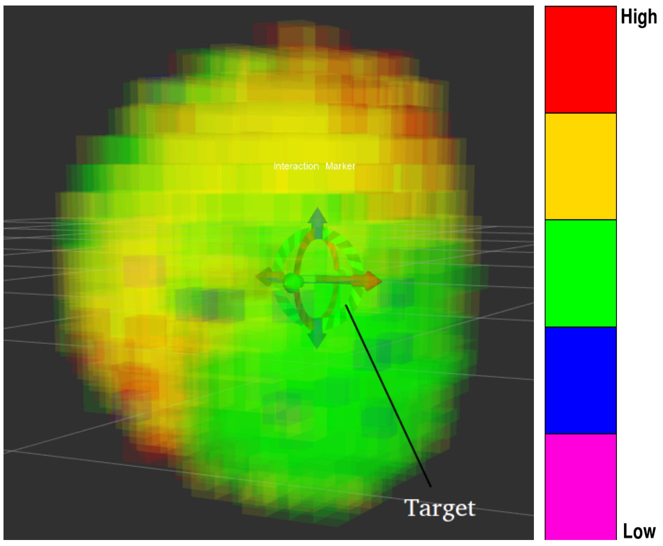


Figure 4-11: Inverse Planability map for arm at resolution 0.1m. Note: The center of image shows the target pose at origin. The colored voxels around it indicate the minimum path length for reaching the target when the arm is placed in the voxel. The lengths have been scaled to lie between 0 rad and 100 rad for visualization. Colors range from Red = High minimum path length (>80) to Magenta = Low minimum path length (<20) for reaching pose at origin.

Algorithm 4 Generating Inverse Planability Map

```
1: create $V$ voxels
2: read planability map stored in memory
3: for each planable pose $p_j$ in planability map do
4:     compute origin to end-effector transform $T_{origin}^{EE}$
5:     invert transform to compute $T_{EE}^{origin}$
6:     store translation of inverse transform in a point cloud $pcl$
7:     store $<pcl_j, c_j>$ in a multimap data structure
8: end for
9: for each voxel $v_i$ in $V$ with voxel center $s_i$ do
10:     use nearest neighbor within voxel search to identify number of $pcl$ points within $v_i$
11:     compute inverse planability cost $C_i^{inverse}$
12:     store $<s_i, C_i^{inverse}>$ in hdf5
13: end for
```

Higher cost of a voxel implies the minimum amount of joint motion required to reach the origin is high when the robotic arm is placed at that location. It is important to note that the real path length depends upon the scene as well as the planner used. However, the minimum cost is being used as an indicator to select a suitable location. Figure 4-11 shows the inverse planability map computed for the RCS arm at resolution 0.1m. It can be seen that locations closer to the target pose are better as compared to those that are far away or at the top. It is observed that the difference between quality of various voxels of the inverse planability map (standard deviation 11.81 rad) is much less than that of inverse reachability map (standard deviation 46.52 rad).

4-3 Online Computation

Until the previous section, the processes involved in offline computation for determination of arm location have been discussed. In this section, focus is on how those inverse reachability and planability maps are utilized for generating the commutation configuration online.

4-3-1 Fusion of Maps

The reachability and planability maps in themselves reflect only a certain quality of manipulation feasible with the given robot. The reachability map, and therefore the inverse reachability map do not express the quality of path that can be planned between the start configuration and the target pose. Similarly, the planability map and therefore the inverse planability map do not reflect the probability of being able to reach a target pose successfully. Hence DeCOWA combines the IRM and IPM into an Inverse Fusion Map (IFM).

In order to combine the reachability and the planability maps, the cost of reaching a particular region can be seen as a penalty to its reachability. Accordingly the quality of the voxel would be determined by the ratio of reachability R_i to the cost of reaching C_i. When the chances of reaching the voxel are higher, the quality of the voxel is also higher. But for a greater cost of reaching the voxel, the quality is lower. This way of fusion is chosen as it does not need weight coefficients to be used. Mathematically,

$$Q_i^{inverse} = \frac{R_i^{inverse}}{C_i^{inverse}}.$$

This fusion is done online rather than precomputed to maintain modularity of the algorithm. Depending upon the specific requirement, DeCOWA can use the IRM, the IPM, or the IFM for robot placement. Furthermore, the criteria of short motion can be replaced with inverse planability maps reflecting other criteria such as power consumption, mechanical energy cost and so on. Having generated the inverse fusion map, the next step is to use it for computing the arm location online. This procedure is detailed in the next two subsections.

4-3-2 Transforming and Slicing the Inverse Metric Map

The inverse reachability map, the inverse planability map, and the inverse fusion map indicate the best locations in SE(3) to place the manipulator in order to reach the pose at origin. To

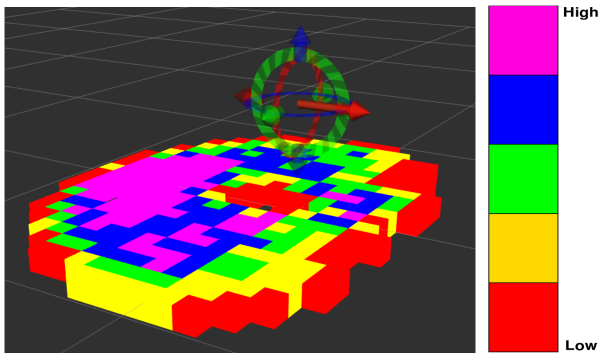


Figure 4-12: Sliced and transformed inverse reachability map for reaching the target pose. The target pose is indicated with the red arrow. The colored voxels below it indicate the number of ways of reaching the target when the arm is placed in the voxel. Colors range from Red = Low Success Likelihood (< 20%) to Magenta = High Success Likelihood (> 80%) in reaching target.

utilize these maps in finding the best arm location for a target other than origin, all the voxels in the map need to be transformed accordingly.

For each voxel v_i defined by the voxel center s_i, the new coordinates of centers must be calculated. If the transform from origin to the target pose is given by T^{target}_{origin}, then

$$s_i' = T^{target}_{origin} \cdot s_i$$

where s_i' is the new center of the transformed voxel v_i.

Furthermore, since the robotic arm is rigidly placed on the robot at a certain height, this constraint must be applied in order to filter out a subset of possible commutation configurations. Consider a surface S defined by the arm's height belonging to $\mathrm{SE}(2) \cong \mathbb{R}^2 \times \mathbb{S}^1$ which is a subspace of SE(3). This plane goes through the arm's height and is parallel to the xy plane. Therefore,

$$S = \{ \begin{bmatrix} R_z & 0 & p_{x,y} \\ 0 & 1 & h_{manipulator} \\ 0 & 0 & 1 \end{bmatrix} \in \mathrm{SE}(3) \mid p_{x,y} \in \mathbb{R}^2, R_z \in \mathbb{S}^1 \}$$

where R_z is the rotation of the plane about the $z - axis$, $p_{x,y}$ are the points on the plane at arm's height $h_{manipultor}$. To filter the voxels based on this criteria, the voxels whose centers lie within $H_{\text{manipulator}} \pm r$ are retained. Here, r denotes the resolution of the inverse fusion map.

Such a sliced inverse reachability map is shown in Figure 4-12. In the next step, this map is filtered to remove the possible arm locations that occupy a non-available cell in the 2D costmap. This is illustrated in Figure 4-13. This leads to a final subset of locations where the robotic arm can be placed. The next subsection describes the iterative search method to compute this placement.

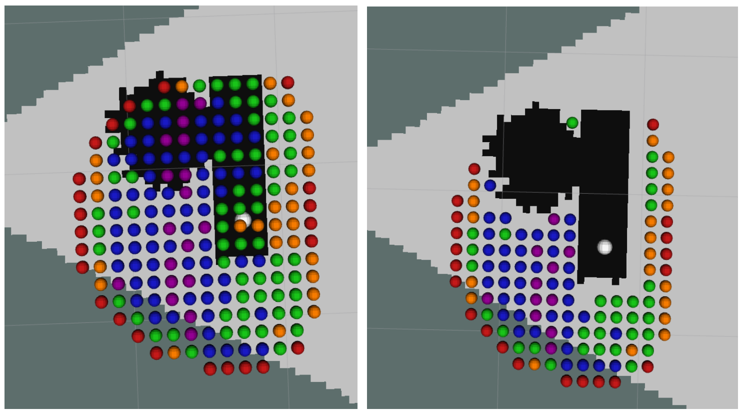


Figure 4-13: Top view of Inverse Reachability map filtered based on 2D cost map.
Left: Sliced IRM; Right: Filtered Inverse Reachability Map; White ball is the Goal as seen from the top. Th 2D cost map is created on the basis of the 3D octomap. The light gray region shows the visibility range of RGB-D camera. Black region is the projection of 3D obstacle on ground. Dark gray region is the area not covered by the RGB-D camera. Filtering removes voxels intersecting with the obstacle.

4-3-3 Iterative search and Optimizing Platform Orientation

Using the inverse metric maps, one can search through possible set of arm locations as illustrated in Figure 4-12. The quality associated to the voxels can be taken as probabilistic measure of finding a successful short motion to the target [27]. Therefore, an iterative search in the order of decreasing quality values is performed until a collision free inverse kinematic solution to the target is found. For the collision checking, the inflated model of the robot URDF is utilized as discussed in section 3-2-1.

The presence of obstacles imply not all locations will produce collision free inverse kinematic solutions. For those locations where the arm can reach the target pose, orientation of the platform is constrained in a circle around it (see Figure 4-14 for illustration).

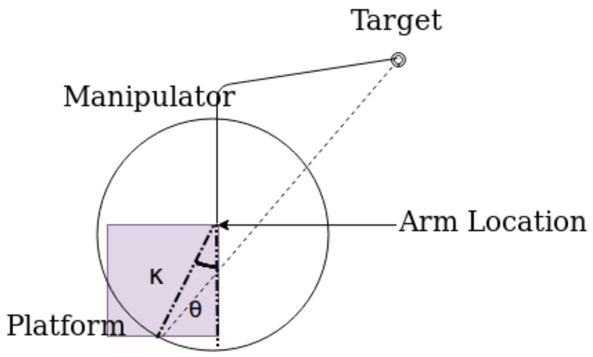


Figure 4-14: Illustrating multiple platform orientations for the same arm location

In order to select the robot orientation, the following function is minimized,

$$f(\theta) = \begin{cases} sqrt(x(\theta) - x_0)^2 + (y(\theta) - y_0)^2, & \text{robot} \notin (\mathcal{C}^{obst} \cup \mathcal{S}) \\ \infty, & \text{otherwise} \end{cases}$$

where $x(\theta)$ represents the x coordinate of the platform in the *odom* frame, and $y(\theta)$ represent the y coordinate. (x_0, y_0) are the coordinates of present location of the robot. Since the robot is constrained in a circle, these values are a function of θ.

Hence the optimization objective is to,

$$\underset{\theta}{\text{minimize}} f(\theta)$$

subject to,

$$\sqrt{(x_{arm} - x(\theta))^2 + (y_{arm} - y(\theta))^2} = \kappa$$

where κ is the fixed distance between the platform's and arm's coordinates. Nelder Mead [35] in the NLOpt library [89] is used to perform the above optimization. Since the domain of minimizing the platform orientation is dependent upon the arm location, the output solution is not necessarily the closest available commutation configuration. However, this approach ensures that the commutation configuration satisfies the criteria for manipulation as well for navigation with greater priority placed on the characteristics of the robotic arm.

If the optimization is unable to find a solution, the iterative search of the arm location continues. In the next section, the complete robot placement algorithm is presented.

4-4 Robot Placement Algorithm

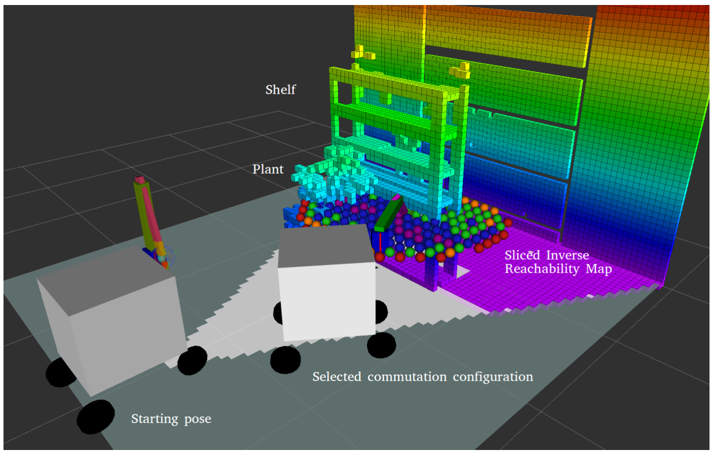

Shelf
Plant
Sliced Inverse Reachability Map
Selected commutation configuration
Starting pose

Figure 4-15: Commutation Configuration found using the proposed Robot Placement Algorithm. The figure shows a scene as visualized through a RGB-D camera. The light grey region on the ground is the visibility range of the camera. The obstacles i.e. the plant and the shelf are seen through an octomap. Slicing and filtering of the inverse reachability map produces a set of locations from which the arm can reach the target with a certain probability. Through iteration in decreasing order of voxel quality, a collision free arm configuration to target is found. The orientation of the platform is determined by minimizing distance to starting pose. This is the commutation configuration as shown above.

Figure 4-16 shows how the offline and the online components of the proposed robot placement algorithm DeCOWA functions along with the respective inputs and outputs of the system.

It is noted that the generation of the inverse metric maps only requires the model of the arm. Therefore, if the arm is placed on another robot in the future, the workspace analysis does not need to be repeated. The planning criteria in this thesis is having fast execution of motion, therefore a short motion plan. In general multiple inverse metric maps may be created offline corresponding to different motion quality criteria. The online selection criteria can be chosen as any one among them. This thesis presents three of the maps that may be used with DeCOWA, namely IRM, IPM, and IFM.

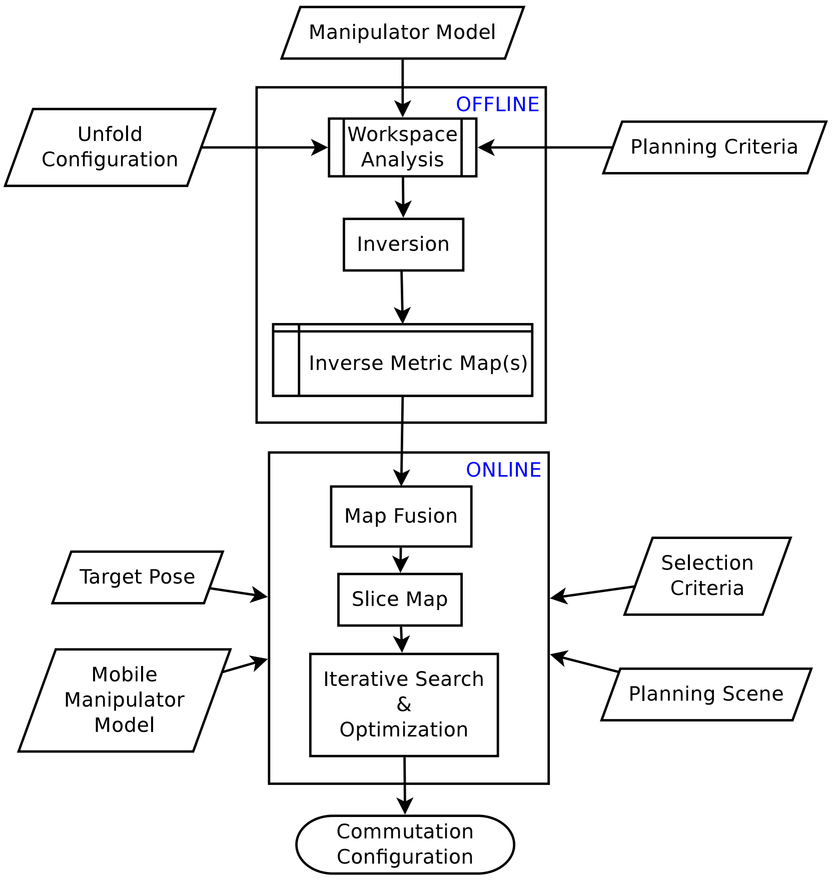


Figure 4-16: Offline and online components of the proposed robot placement algorithm. Offline procedure involves workspace analysis for generation of reachability and planability maps. Inversion of these maps lead to inverse metric maps. Online, these inverse maps are fused, transformed, and sliced. Through an iterative search followed by optimization, the commutation configuration $\mathbf{q_b}^{goal}$ is determined.

The complete approach of DeCOWA in the real world scene is described by Algorithm 5. Figure 4-16 shows a commutation configuration found using this algorithm.

4-5 Summary

In this chapter the robot placement algorithm for mobile manipulators was proposed. The method developed aims at finding a robot placement that maximizes the chance of successful planning while ensuring a short path to the target.

Algorithm 5 Online Generation of Commutation Configuration using DeCOWA

```
Require: Inverse Reachability Map
Require: Inverse Planability Map
Require: Target Pose
Require: 3D octomap and 2D costmap
Require: Robot URDF
 1: read inverse reachability map stored in memory
 2: read inverse planability map stored in memory
 3: for each voxel $v_i$ in inverse planability map do
 4:     fetch $R_i^{inverse}$ corresponding to voxel center $s_i$
 5:     fetch $C_i^{inverse}$ corresponding to voxel center $s_i$
 6:     compute $Q_i^{inverse}$ as ratio of $R_i^{inverse}$ and $C_i^{inverse}$
 7:     store <$s_i$, $Q_i^{inverse}$> in IFM
 8: end for
 9: for each voxel $v_i$ with center $s_i$ in IFM do
10:     if $h_{manipulator} - r < s_{i,z} < h_{manipulator} + r$ then
11:         store <$s_i$, $Q_i^{inverse}$> in filtered IFM
12:     end if
13: end for
14: for each voxel $v_i$ wih center $s_i$ in filtered IFM do
15:     place arm at $s_i$
16:     find IK to target
17:     if collision free IK found then
18:         minimize platform distance to current platform location
19:         if collision free platform orientation found then
20:             return commutation configuration
21:         end if
22:     end if
23: end for
24: return NULL
```

The algorithm segregates the determination of commutation configuration into two parts, namely finding the arm location, and thereafter the corresponding platform orientation. For the selection of arm location workspace analysis is employed. The reachability map indicates the kinematic capabilities of the manipulator. In a similar fashion a planability map is proposed that computes the motion capabilities of the robot. The average minimum joint motion required to reach different regions in workspace of the arm is computed using STOMP.

The reachability and the planability maps are inverted to generate voxels in SE(3) where the arm can reach the pose at origin. It is noted that the actuation error of the mobile base affects the accuracy of reaching the arm goal. Therefore, the voxel size is selected based on the goal tolerance of the local planner, i.e. 0.1m.

Since the inverse reachability map does not account for the motion quality and the inverse planability map does not consider the likelihood of successful planning while ranking the regions, these maps are fused together online. The ratio of reachability of a voxel to its planability cost is used to reflect the quality of a voxel for the inverse fusion map. Any of these three maps may be used along with DeCOWA depending upon the requirements.

Since the height of the arm is fixed, this map is sliced to generate a list of possible arm locations in 2D. Corresponding to each arm location, the quality of the voxels act as frequency distributions from descriptive statistics. By iterating these voxels in decreasing order of their quality, a collision free inverse kinematics solution to the target is found. For this arm location, the orientation of the platform is computed by minimizing the euclidean distance to the current platform location.

The search continues until a pair of collision free arm configuration and platform orientation is found. This is selected as the commutation configuration for mobile manipulation.

The next chapter will evaluate the performance of proposed algorithm DeCOWA for the three maps.

Chapter 5

Evaluation

The previous chapter elaborated the proposed robot placement algorithm DeCOWA satisfying the criteria set in section 1-4. In this chapter, this algorithm is evaluated against the chosen metrics of success-rate, total joint motion (arm path length), and computation time. It is also seen how the algorithm performs in presence of actuation errors. The evaluation is divided into three sections. First, the experimental setup is explained and the hypothesis being tested are discussed. Thereafter the results of different experiments are catalogued. Finally, the implications of these results are discussed noting down the usefulness as well as drawbacks of the proposed approach.

5-1 Design of Experiments

There are primarily two kinds of experiments performed to evaluate the utility of the proposed robot placement algorithm. In the first set of experiments (Experiment A), the performance of DeCOWA (when using IRM, IPM, and IFM) is evaluated against random placement of the robot around the target. Random placement implies the commutation configuration $\mathbf{q_b}^{goal}$ is chosen from a uniform random distribution of poses around the target object within a radius equal to the distance from robot's base to its end effector at maximum extension of the arm. Thereafter, the second set of experiments (Experiment B) analyze the effect of actuation errors on the success of robot placement. All the experiments have been performed for the LEA robot and arm model on a Personal Computer (PC) with Intel core i5, 3.30 GHz and 16 GB of Random Access Memory (RAM). To test the robot placement approaches, the framework described in Chapter 3 is used.

Before describing the experiments and hypothesis in detail, the next section elaborates the experimental setup in V-REP.

5-1-1 Experimental Setup

The domestic environment is highly unconstrained with objects of different shapes, sizes, and textures placed in varied locations. Therefore, a robot can encounter a wide variety of scenes during mobile manipulation. In order to encompass the primary characteristics of these plethora of scenarios, a limited number of representative scenes are evaluated in this thesis. Generally, objects in a domestic environment are picked up from either a table or a shelf. Using these two classes of scenes as the basis, 20 test scenes are considered for experimentation. This approach of evaluating the robot placement algorithm in classes of scenes was also used in [90].

The classes of scenes considered for experimentation encompass the variety of challenges encountered during robot placement; the target pose may be oriented in various ways, it may be in a narrow space, or have obstacles that surround it. The target may be at different heights for which the number of possible commutation configurations is different. In some cases few of the available commutation configurations may be close to singularity. Furthermore, some of the commutation configurations may be unavailable due to obstacles on the ground. Through the evaluation of DeCOWA in the table and shelf classes of scenes, all these variety of problems are evaluated.

Scenes for the Table In the class of scenarios considered for a table, the target has to be reached from the top, i.e vertically. This is generally the case for objects like fruits, and boxes etc. Furthermore, the object may be placed near the center of the table, or towards the edge. Therefore these two combinations of scene configurations are also considered. In one scenario, the table may be cluttered and have obstacles making the motion planning harder, or the table may be relatively empty. Finally, there may be obstacles on the ground that make a few robot placement locations unavailable. Hence the total number of scenes for the table-class is 8 (shown in Figure 5-1):

$$\begin{bmatrix}\text{Object near center}\\ \text{Object near corner}\end{bmatrix} \times \begin{bmatrix}\text{Cluttered with obstacles}\\ \text{No obstacles}\end{bmatrix} \times \begin{bmatrix}\text{Obstacles on ground}\\ \text{No obstacles on ground}\end{bmatrix}$$

Table 5-1: Labels for scenes in the Table class

Combination #	Label	Target Location	Table Clutter	Obstacle on Ground
1	*TCe*	Center	No	No
2	*TCeCl*	Center	Yes	No
3	*TCeOb*	Center	No	Yes
4	*TCeClOb*	Center	Yes	Yes
5	*TCo*	Corner	No	No
6	*TCoCl*	Corner	Yes	No
7	*TCoOb*	Corner	No	Yes
8	*TCoClOb*	Corner	Yes	Yes

Scenes for the Shelf In the class of scenes considered for a shelf, the target has to be reached sideways from the front. This is generally the case while picking up bottles, glasses, books and so on. A shelf is different from a table since the slabs and the frame of the shelf itself act as obstacles. Furthermore the objects on a shelf may be kept at different heights. In the experiment, three categories of heights are considered, i.e. top, middle, and bottom. Also,

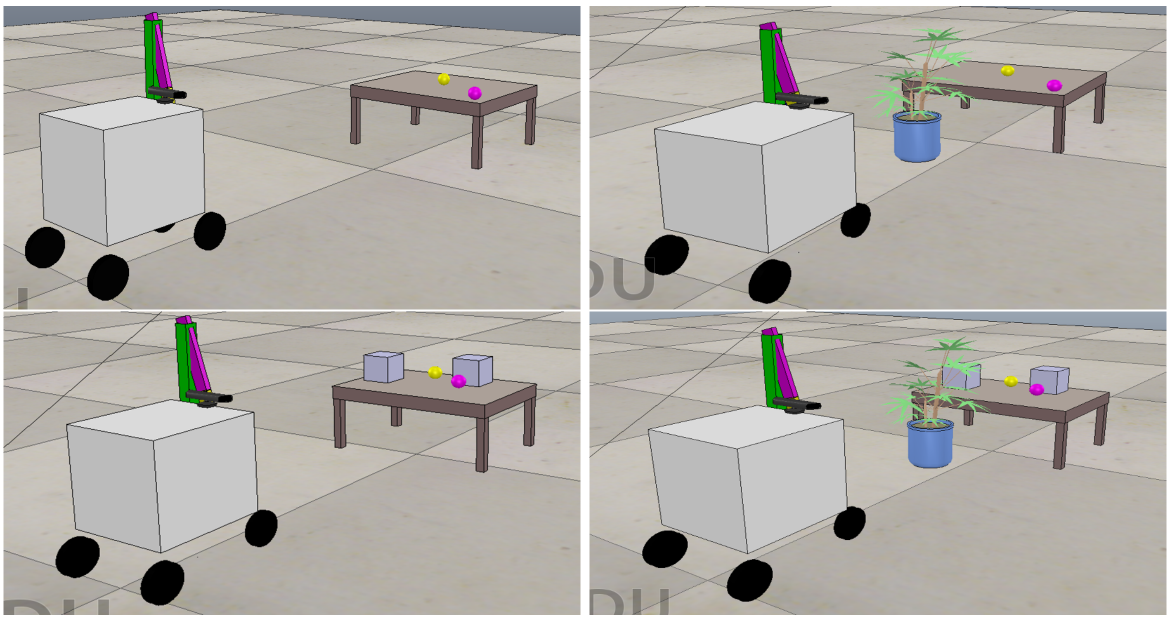


Figure 5-1: Experiment Class - Table. The robot is the starting state at which it visualizes the scene to determine the commutation configuration.
Figure shows 8 scenes = 4 × 2 targets; Yellow = Target in center, Magenta = Target in corner. Scene on Top Left: No obstacles on table, no obstacle on ground; Top Right: No obstacle on table, obstacle on ground; Bottom Left: Obstacles on table, no obstacle on ground; Bottom Right: Obstacles on table, obstacle on ground

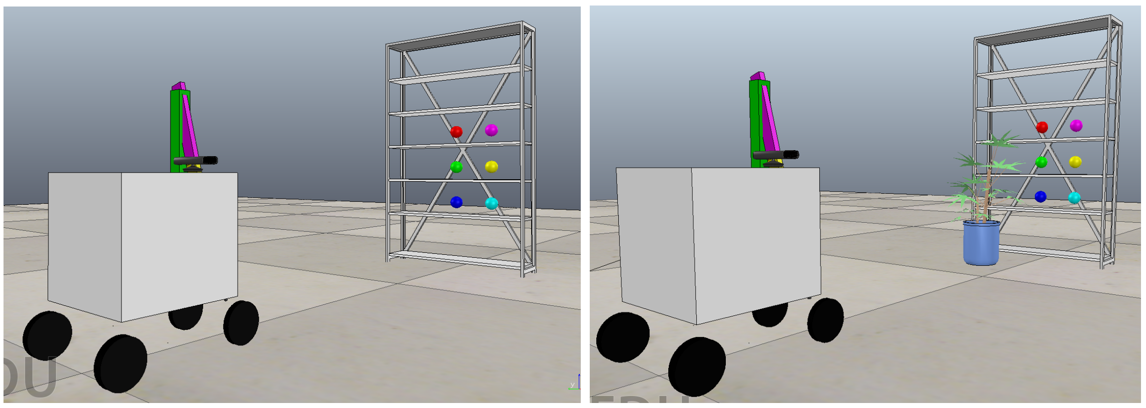


Figure 5-2: Experiment Scene - Shelf. The robot is the starting state at which it visualizes the scene to determine the commutation configuration.
Figure shows 6 targets; Red: Target at top shelf in center, Magenta: Target at top shelf in corner, Green: Target in middle shelf in center, Yellow: Target at middle shelf in corner, Blue: Target at bottom shelf in center, Cyan: Target at bottom shelf in corner.
Scene on Left: No obstacle on ground. Right: With a plant as obstacle on ground

similar to the case of a table, the target may be near the center of the shelf, or near the corner. Finally, there may be objects on the ground that limit the possible locations from which the target is accessible. Therefore, the total number of sub scenes for the shelf class is 12 (shown in Figure 5-2):

$$\begin{bmatrix} \text{Object at top} \\ \text{Object in middle} \\ \text{Object at bottom} \end{bmatrix} \times \begin{bmatrix} \text{Object at the center} \\ \text{Object at the corner} \end{bmatrix} \times \begin{bmatrix} \text{Obstacles on ground} \\ \text{No obstacles on ground} \end{bmatrix}$$

Table 5-2: Labels for scenes in the Shelf class

Combination #	**Label**	**Target Location**	**Target Height**	**Obstacle on Ground**
1	$SCeT$	Center	Top	No
2	$SCeM$	Center	Middle	No
3	$SCeB$	Center	Bottom	No
4	$SCeTY$	Center	Top	Yes
5	$SCeMY$	Center	Middle	Yes
6	$SCeBY$	Center	Bottom	Yes
7	$SCoT$	Corner	Top	No
8	$SCoM$	Corner	Middle	No
9	$SCoB$	Corner	Bottom	No
10	$SCoTY$	Corner	Top	No
11	$SCoMY$	Corner	Middle	No
12	$SCoBY$	Corner	Bottom	No

It must be noted that the robot is placed close to the shelf and the table for experimentation to accurately visualize the scene octomap, and avoid the effects of sensing inaccuracies. Furthermore, motion planning to the commutation configuration (locomotion) is not part of the experimentation.

Also, while the "reach from top" approach is considered only for the table and "reach from side" approach is considered for the shelf, together they represent the different ways in which objects may be reached. The limitation of this setup is that it does not consider the cases where the targets are approached at an oblique angle.

To summarize the experimental setups and make it easier for further referencing, Table 5-1, and Table 5-2 labels the scenes and target combinations accordinTo summarize the experimental setups and make it easier for further referencing, Table 5-1, and Table 5-2 labels the scenes and target combinations according to their characteristics. g to their characteristics.

Following the description of evaluation setup, the next two sub sections describe the experiments conducted in detail.

5-1-2 Experiment A: Performance Measurement

The first experiment aims to measure the performance of DeCOWA with IRM, IPM, and IFM in different scenes. As explained previously, performance of the proposed algorithm is compared against random placement of the robot to reach the target pose. Random placement is chosen for comparison because no existing algorithm considers all the desired criteria. Therefore, random placement is an approach unbiased towards any particular characteristic. This method is explained hereafter.

Random Placement: This approach uses a uniform random distribution $\mathcal{U}$ of poses to select a placement for the robot within a radius L around the target $\mathbf{p}^* = [p_x^*, p_y^*, p_z^*, p_\theta^*, p_\psi^*, p_\phi^*]^T$. The radius of search equals the distance from robot's base to its end effector at maximum extension of the arm. Accordingly, for LEA URDF model, the radius of search is $0.85m+0.6m = 1.45m$. The robot is repeatedly placed at one of these poses until a collision free IK solution to the target is found. The search is iterated until a time limit of 5 seconds as a marketable robot should function fast enough for users. This method is illustrated in Algorithm 6.

Algorithm 6 Online Generation of Commutation Configuration using Random Placement

```
Require: Target Pose
Require: 3D octomap and 2D costmap
Require: Robot URDF
 1: while run time < 5s do
 2:     generate $x$ coordinate of pose from $\mathcal{U}_x \in [p_x^* - L, p_x^* + L]$
 3:     generate $y$ coordinate of pose from $\mathcal{U}_y \in [p_y^* - L, p_y^* + L]$
 4:     generate yaw of pose from $\mathcal{U}_\theta \in [0, 2\pi]$
 5:     if $\sqrt{(x - p_x^*)^2 + (y - p_y^*)^2} < L$ then
 6:         compute IK to target
 7:         if IK solution to target exists then
 8:             check for collision with octomap
 9:             if robot is collision free then
10:                 return commutation configuration
11:             end if
12:         end if
13:     end if
14: end while
15: return NULL
```

The robot placement approach DeCOWA presented in the previous chapter might use any of the inverse metric maps for the iterative search. Therefore the following methods are compared in experiment A:

1. RoP_{IRM}: DeCOWA Robot Placement using Inverse Reachability Map (IRM) and Optimization
2. RoP_{IPM}: DeCOWA Robot Placement using Inverse Planability Map (IPM) and Optimization
3. RoP_{IFM}: DeCOWA Robot Placement using Inverse Fusion Map (IFM) and Optimization
4. RaP: Random Placement

The performance of the aforementioned robot placement methods is measured by the following metrics:

- **Success rate of motion planning to target pose** (SR_{MP}): This measures the percent of times a motion planner is able to plan a path to the target pose when the robot is placed at the commutation configuration as given in Eq. (5-1).

$$SR_{MP} = \frac{\text{No. of successful plans}}{\text{No. of trials}} \times 100\% \tag{5-1}$$

  It is assumed that there are no constraints on the end effector for the motion. For the purpose of motion planning, MoveIt!'s default time limit of 5 seconds is imposed on the planner. If in a particular iteration the robot is not able to find a commutation configuration, the planning is also considered a failure. It must be noted that different planners are more successful in different kinds of scenes. For this thesis, the default planner with MoveIt!, RRTConnect is used for experimentation. Note that this planner is not the most suitable for application in domestic environments due to randomized paths. However, the aim is to measure the efficacy of the proposed robot placement approach which does not depend on the planner used during runtime. This measure was also used by [11].

- **Path length to target** (PL): In case of successful planning, the total joint motion as measured through Eq. (5-2) is noted.

$$Tosummarizetheexperimentalsetupsandmakeiteasierforfurtherreferencing, Table\ 5-1, andTable\ 5$$
(5-2)

- **Computation time** (τ): Finally, the time it takes to compute the commutation configuration given by Eq. (5-3) is noted.

$$\tau = \frac{\sum_{i=1}^{\text{Successful Trials}} \tau_i}{\text{No. of successful Trials}}. \tag{5-3}$$

  This was measured in literature by [34], and [90].

For this experiment, it is hypothesized that on an average,

- $H_{p,a}$: $SR_{MP}(RoP_{IRM}) \geq SR_{MP}(RaP)$ i.e. the success rate by using DeCOWA with IRM would be higher than that using random placement,
- $H_{p,b}$: $PL(RoP_{IPM}) \leq PL(RaP)$ i.e. the path length by using DeCOWA with IPM would be lower than that using random placement,
- $H_{p,c}$: $SR_{MP}(RoP_{IFM}) \geq SR_{MP}(RaP)$ i.e. the success rate by using DeCOWA with IFM would be higher than that using random placement,
- $H_{p,d}$: $PL(RoP_{IFM}) \leq PL(RaP)$ i.e. the path length using DeCOWA IFM would be lower than that using random placement.

Finally, the average time taken to find a suitable robot placement through the proposed algorithm will also be noted and compared to the benchmark of 750ms set based on literature [27] [28].

The above hypothesis are based on the understanding that the inverse reachability map based approach increases the chances of successful planning; and the inverse planability map based approach should imply a short path to the target. On combination of these two maps, the inverse fusion map should have a higher success rate as well as a shorter path as compared to random placement. In other words, the DeCOWA produces better robot placements that one would obtain otherwise though chance i.e. random placement.

This experiment is run for each of the 20 scenes and repeated for a 100 times. Therefore, No. of Trials in Eq. (5-1) is 100. In each iteration, the robot placement through DeCOWA as well random placement method is computed. The robot is placed at the chosen location and motion planning to the target is attempted. If the motion planning is successful, the path length is computed using Eq. (5-2). The descriptive statistics of the experiment results are thereby calculated. It must be noted that this experiment is performed without consideration of actuation or sensing errors.

5-1-3 Experiment B: Success Rate in presence of Navigation Uncertainty

In the second set of experiments, the aim is to analyze the effect of error in reaching the commutation configuration on the planning success rate. This effect is only analyzed on DeCOWA with IFM i.e. RoP_{IFM}. In this experiment, a monte-carlo simulation [83] is used to introduce an error in the robot's commutation location. Since the DWA's [75] goal tolerance is 0.1m, it implies that the real location of the robot can be between $\pm 0.1m$ of the generated commutation in x as well as y axis. Furthermore, the goal tolerance in yaw is $0.05rad$. Therefore the robot's orientation can be between ± 0.05 of the desired yaw.

It is assumed that the distribution of these errors is Gaussian with a mean equal to zero. Since 99.7% of values lie within ± 3 times the standard deviations of the mean, the σ_{xy} and σ_{yaw} are calculated as

$$\begin{aligned} 3\sigma_{xy} &= 0.1m \\ \sigma_{xy} &= 0.0\bar{3}m \end{aligned}$$

$$\begin{aligned} 3\sigma_{yaw} &= 0.05\ rad \\ \sigma_{yaw} &= 0.01\bar{6}\ rad \end{aligned}$$

Therefore, after computing the commutation configuration using DeCOWA with inverse fusion map (RoP_{IFM}), an error is introduced in the robot location with $\epsilon_x \leftarrow \mathcal{N}(0, \sigma_{xy})$, $\epsilon_y \leftarrow \mathcal{N}(0, \sigma_{xy})$, and $\epsilon_{yaw} \leftarrow \mathcal{N}(0, \sigma_{yaw})$. The simulation for each of the 20 scenes is run for a 1000 times. The success in finding an IK solution and a path to the target is recorded (1 for success, 0 for failure). This high number of repetitions are necessary for deriving meaningful results from the monte-carlo experimentation. The success rate in presence of actuation error is then compared to that computed without the introduction of error in experiment A.

5-2 Results and Discussion

In this section the results of the experiments are described. All results are analyzed against the hypothesis set in the previous sections.

5-2-1 Experiment A: Performance

Success Rate : The results of the performance measuring experiments are presented in seven graphs. The first two graphs, shown in Figure 5-3 and Figure 5-4 examine the success rate for the shelf and the table class of scenes (refer Table 5-1, and **??**). It is seen that the success rate of robot placement using the inverse reachability map is higher than that of random placement in all the scenes, except for TCeCl where they are equal, and SCeB where random placement slightly outperforms DeCOWA with IRM. This can be ignored as experimental artefact because all the robot placement methods result in a success rate very close to 100% for SCeB. Based on these graphs, hypothesis $H_{p,a}$ is found to be true for the table class and the shelf class.

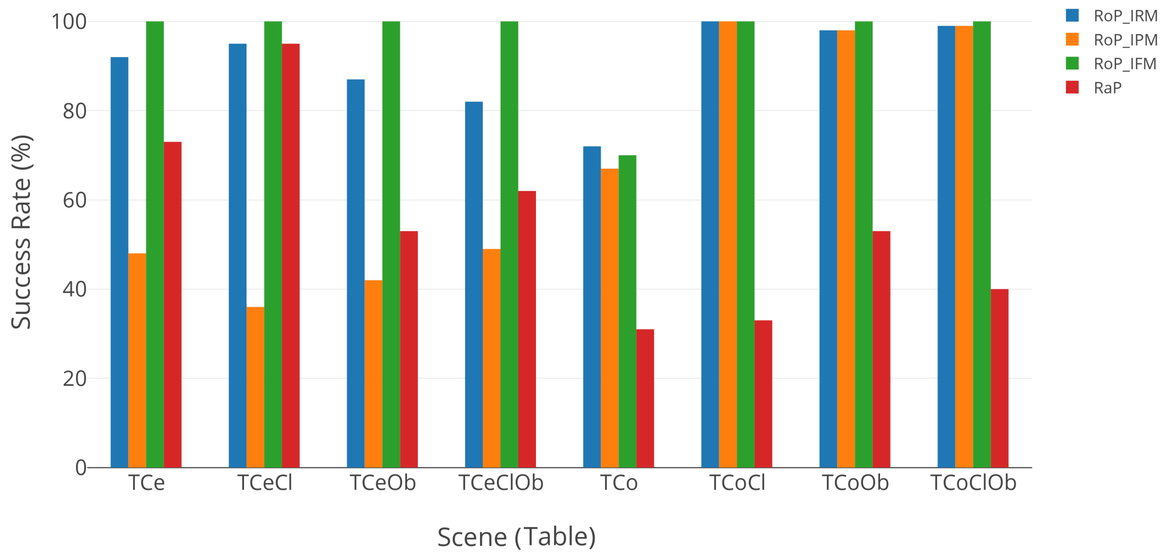


Figure 5-3: Success rate of planning to target in the table class of scenes (refer Table 5-1). The success rate of RoP_IRM and RoP_IFM is greater than that of RaP for all scenes

Similarly, the success rate while using DeCOWA with IFM is found to be better than or equal to that of random placement in all the cases. Therefore, hypothesis $H_{p,c}$ is also true.

It is interesting to observe that the success rate when using DeCOWA with IPM is in many cases lesser than that of random placement (TCe,TCeCl, TCeOb, TCeClOb, SCeMY, SCeBY). This affirms the intuition that using the inverse planability map alone is not sufficient to guarantee a high success rate as well. Accordingly, IRM and IPM have rightfully been fused while using DeCOWA to satisfy all the requirements set in section 1-4.

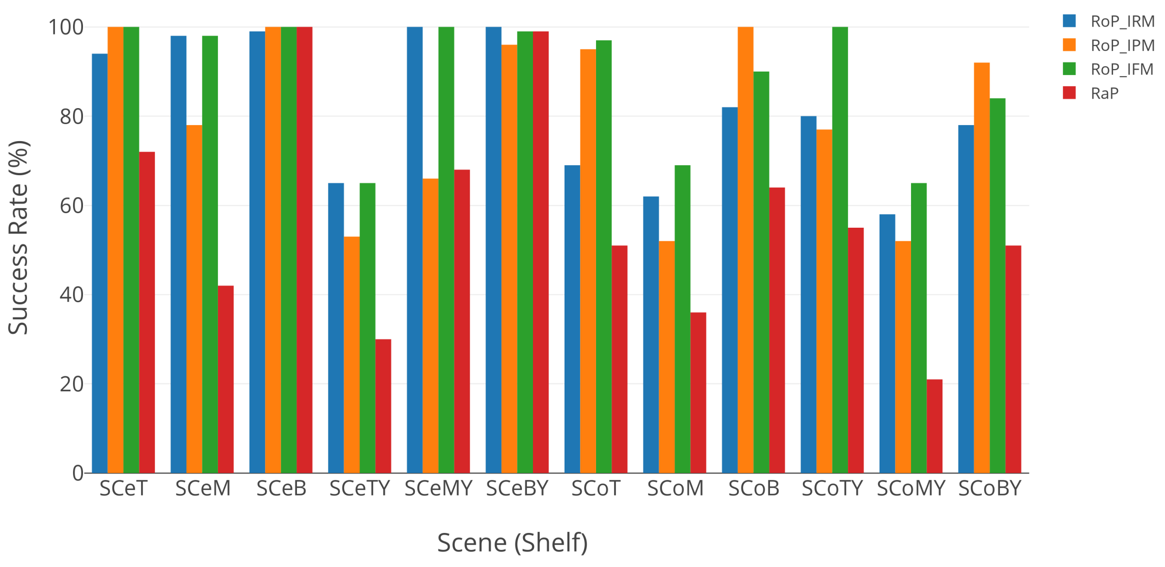


Figure 5-4: Success rate of planning in the shelf class of scenes (refer Table 5-2). The success rate of RoP_IRM and RoP_IFM is greater than that of RaP for all scenes, except for SCeB

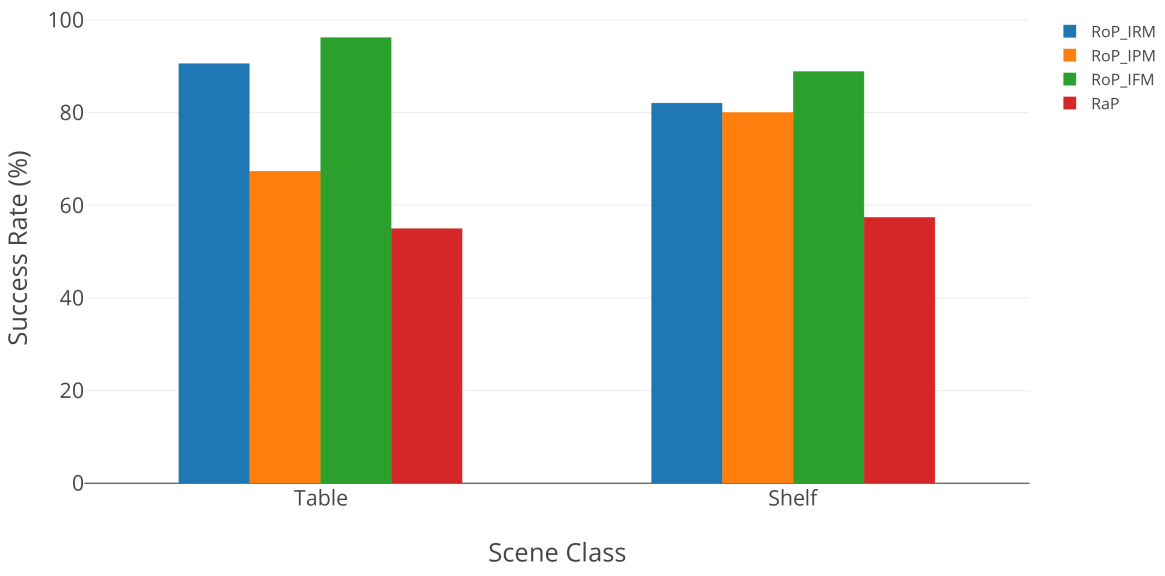


Figure 5-5: Average success rate of planning in the shelf and table class. The average success rate obtained using RoP_IRM and RoP_IFM is greater than that obtained using RaP

The average success rates for the different scenes in the two classes are summarized in Figure 5-5. Proposed algorithm using IFM results in a success rate that is respectively 75% and 56% greater than using random placement for the table and shelf class. It is also seen that the success rate as obtained using IFM is on an average greater than that obtained using IRM. This could be attributed to the fact that it is easier to compute motion within the time threshold when there are more ways of reaching the target through a shorter distance.

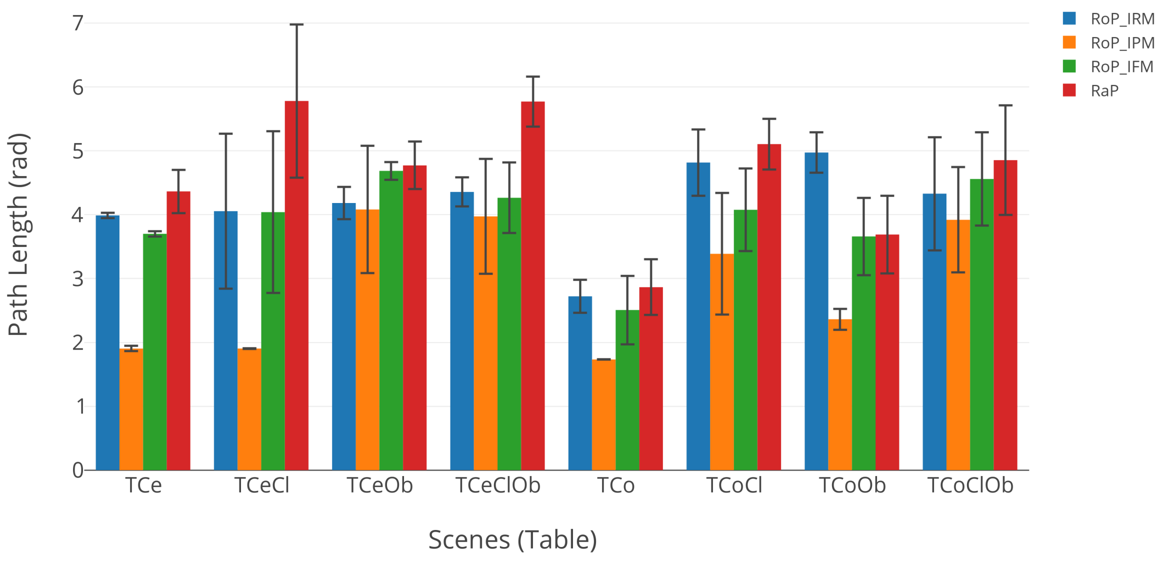


Figure 5-6: Distribution of path lengths for scenes in the Table class(refer Table 5-1). The bars show the average path length to target, and the lines show one standard deviation of its variation. The path length obtained using RoP_IPM, and RoP_IFM is lower than that obtained using RaP for all scenes.

Path Length : Thereafter, Figure 5-6 and Figure 5-7 compare the path lengths for the successful plans in the two classes of scenes. It is observed that the average path length to target is lower when using DeCOWA with IPM or IFM as compared to random placement. An exception to this is observed for scene SCoM where the average path length from IFM is the same as that from random placement. However, the standard deviation of path length is higher when using random placement. This makes DeCOWA more suitable than random placement for generating commutation configurations resulting in smaller path length. Hence, the consistency of results from DeCOWA is higher and preferable for use in domestic environments. Therefore, hypothesis $H_{p,b}$ and $H_{p,d}$ are concluded to be true.

It is further observed, when using DeCOWA with IFM, the path length generally lies between that when using IPM and random placement. This implies that using IFM compensates for the inability of IPM to indicate success rate at the expense of a slightly increased path length. This observation is substantiated from Figure 5-8 summarizing the results for the two scenes.

It concluded that using DeCOWA with IFM results in a path length that is 15%, and 12% shorter than random placement for the table and shelf scenes respectively. It must be noted

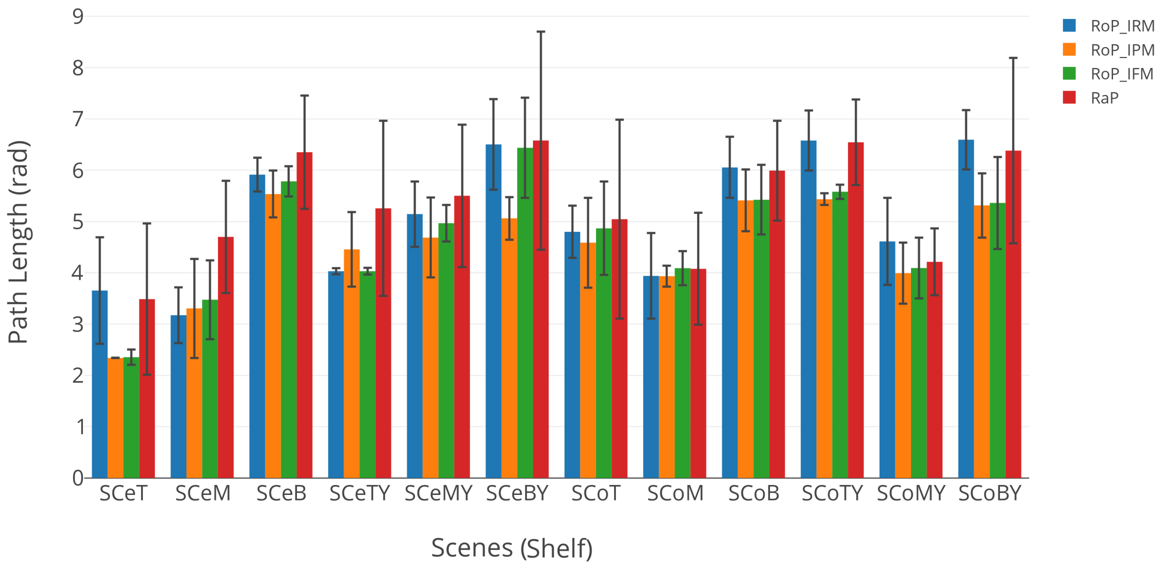


Figure 5-7: Distribution of path lengths for scenes in the Shelf class (refer Table 5-2). The bars show the average path length to target, and the lines show one standard deviation of its variation. The path length obtained using RoP_IPM, and RoP_IFM is lower than that obtained using RaP for all scenes.

that these values are applicable for RRTConnect only. Repeating the experiment with another planner should result in different percentage of reduction in path length.

Computation Time : Finally, the computation time for robot placement in different scenes is illustrated in Table 5-3. The time required to compute the commutation configuration for the table class is much smaller than that for the shelf class. This is because of the higher complexity of obstacles in the shelf scenario. It must be noted that while the average time required for robot placement using IFM lies between that required for IRM and IPM, the time for loading the maps is not presented here. Map loading time is higher for IFM as compared to IRM or IPM as it needs to load both the maps for fusion. It is noticed that the average computation time of DeCOWA is 72% lower than the set benchmark of 750ms for the scenes considered. This can be attributed to lesser number of iterations being required when searching 3D voxels as compared to 6D voxels generally used.

5-2-2 Experiment B: Robustness

The results for the robustness experiments are presented in Figure 5-9 and Figure 5-10. It is observed that the presence of error in reaching the desired commutation configuration leads to a reduction in the success rate for the scenes as expected. From Figure 5-11, it is seen that this reduction is higher for the shelf class (34%) than the table class (27%). This is due to the shelf being more complex structure and having a higher number of obstacles than the table. Therefore, errors may imply a lack of collision free IK solution. Even with this reduction, it

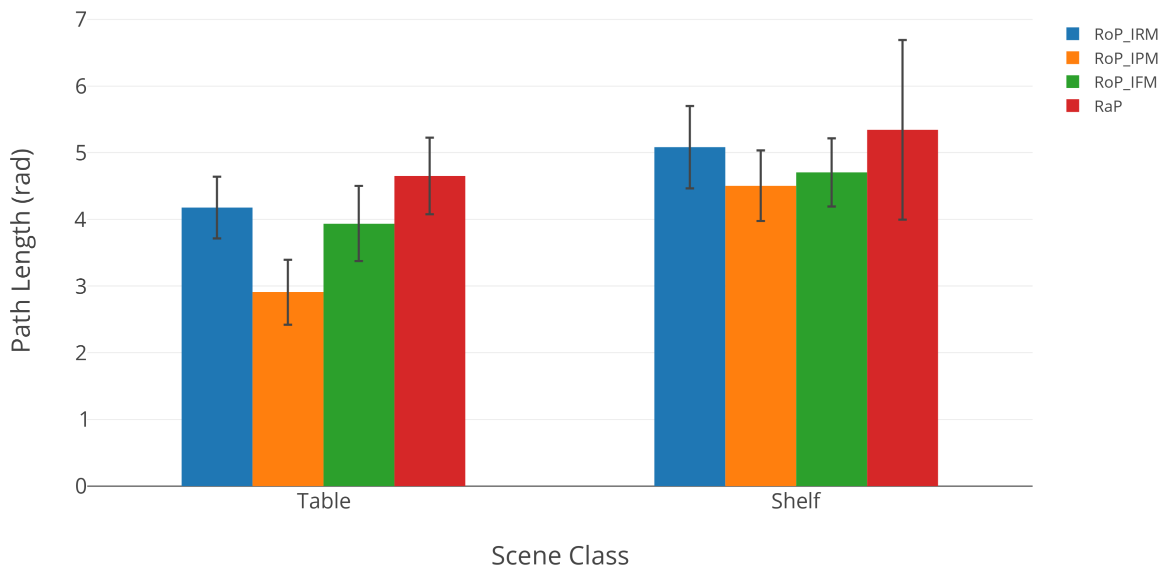


Figure 5-8: Average of average path length for scenes in the shelf and table class. The bars show the average of average path length to target, and the lines show one standard deviation of its variation. The path lengths obtained using RoP_IPM and RoP_IFM is lower than that obtained using RaP. Moreover the path length obtained using the RoP_IFM lies between that found using RoP_IPM and RaP

Table 5-3: Average time (in ms) for generating commutation configuration in the Table and Shelf class using different inverse metric maps. The average time for computing the commutation configuration using IRM is fastest among the three maps. Finding the commutation configuration for the table is faster than that for the shelf. The average time for finding the robot placement is 210.58 ms.

DeCOWA Map	**Scene Class**		
	Table	*Shelf*	***Avg Time (Map)***
IRM	34.83	291.24	*163.04*
IPM	38.60	504.43	*271.52*
IFM	23.74	370.63	*197.18*
Avg Time (Class)	*32.39*	*388.76*	***210.58***

is observed for both the classes that the success rate in presence of actuation uncertainty is still higher than that obtained from random placement without error. As compared to the random placement with error, the success rate of DeCOWA with error is 60% and 49% higher for the table and shelf classes respectively.

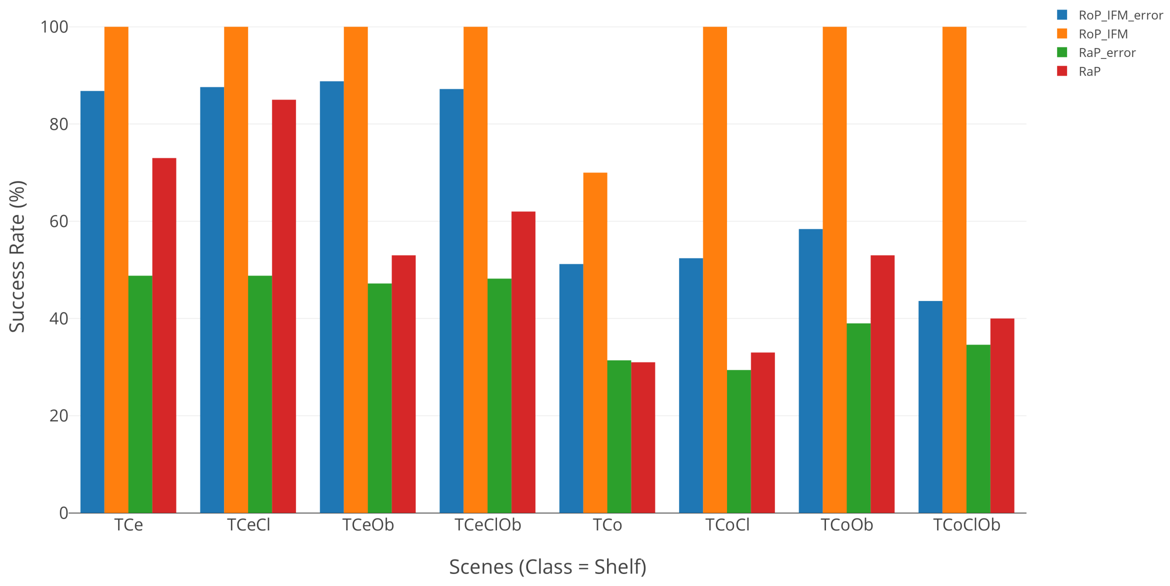


Figure 5-9: Average success rate in planning to targets on Shelf in presence of actuation uncertainty (RoP_IFM_error) is compared against the RoP_IFM, RaP, and RaP_error. The success rate of RoP_IFM_error is higher than success rate of RaP_error as well as RaP.

In conclusion, it is noted that DeCOWA is able to satisfy the robustness requirements set in section 1-4. In the next section, some more advantages of the proposed algorithm are discussed.

5-2-3 Advantages

Apart from including path length in deciding robot placement, accounting for actuation errors, and being computationally fast, DeCOWA provides few more advantages over the existing robot placement methods in literature.

- **Quality of motion for robot placement**: The proposed approach is the first one to account for the quality of motion feasible while determining the robot placement. While the present thesis focuses on robot placement considering short path for the arm, more metrics such as power consumption, mechanical energy required etc. can be used as well.
- **Online selection of metrics**: Unlike existing approaches where one single metric is precomputed and used for robot placement, this thesis presents an approach where a single metric such as reachability, or planability, or a combination of the two can be used.

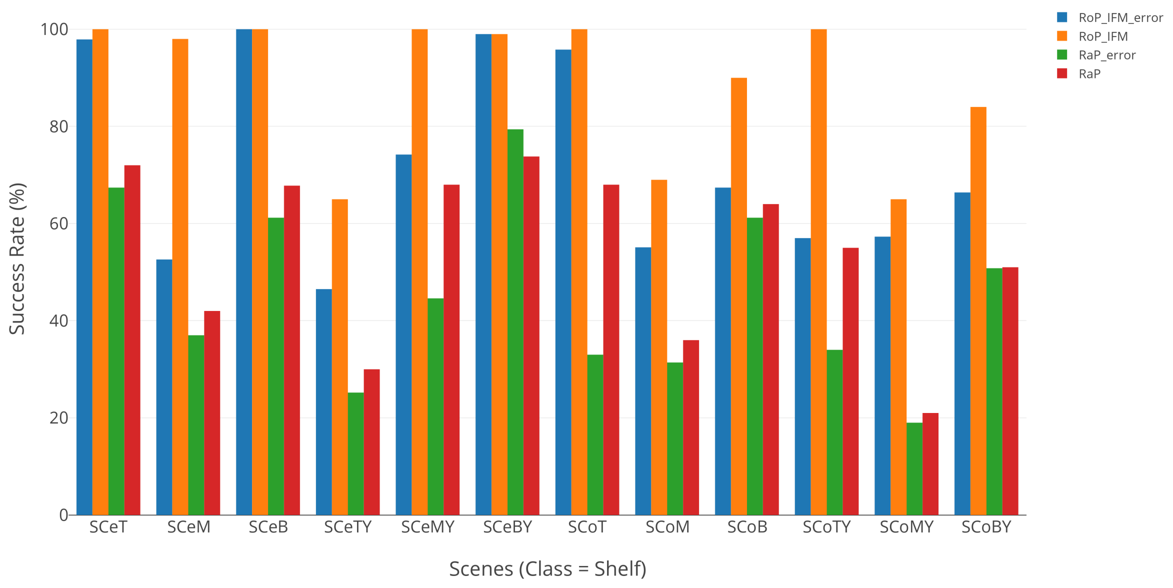


Figure 5-10: Average success rate in planning to targets on Shelf in presence of actuation uncertainty (RoP_IFM_error) is compared against the RoP_IFM, RaP, and RaP_error. The RoP_IFM_error is higher than RaP_error as well as RaP.

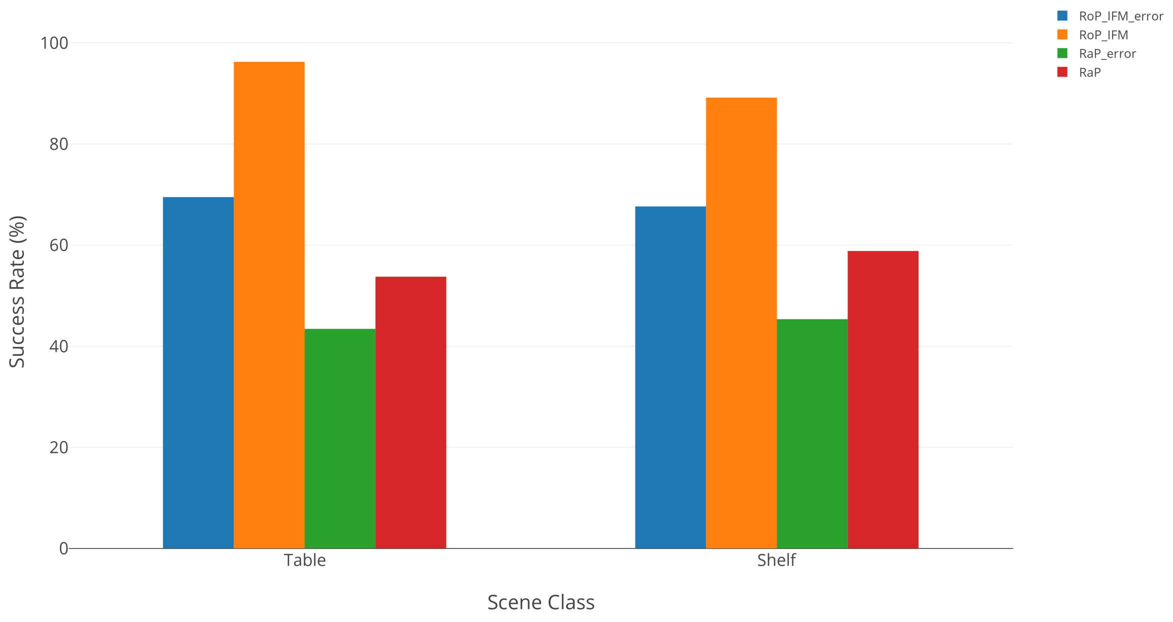


Figure 5-11: Average success rate in planning to targets on Table and Shelf in presence of actuation uncertainty. Introduction of actuation error reduces the success rate, yet performance remains better than random placement

Therefore, depending upon the context of the scene and task, the relevant combination of metrics may be selected for robot placement.

- **Shareability of inverse metric maps**: Unlike existing approaches that compute reachability maps for the whole robot, DeCOWA segregates the selection of arm location from determination of the platform orientation. Therefore, the metric maps of reachability and planability only need to be computed for a robotic arm once and can be utilized across mobile manipulators that use it. Therefore, a common database of reachability and planability maps for common manipulators by ABB [91] and KUKA [92] may be created for use.
- **Robot placement in dynamic scenes**: With the present algorithm, the robot placement is computed in a static scene. However, since the time to compute the commutation configuration is small, the scene could be continuously monitored for changes and DeCOWA could be re-utilized if any changes are observed.

Having discussed the advantages of using DeCOWA, the next section details the drawbacks of the proposed approach.

5-2-4 Limitations and Drawbacks

The proposed algorithm utilizes the inverse reachability and planability maps as indicators for success rate and path length respectively. Since these indicators were computed using inverse kinematics and motion planning in free space, it is expected that the arm location obtained using the proposed approach would be different from the optimal arm location in presence of obstacles. In order to find the optimal location in terms of reachability or planability, an exhaustive search can be performed which is time consuming.

A set of experiments were performed to benchmark the best arm location obtained using proposed algorithm against the real optimal location for each scene.

For each scene, a filtered and sliced inverse reachability map as shown in Figure 4-13 is computed. Thereafter the robotic arm is placed at each of those available locations. If an inverse kinematic solution to the target is available, the platform orientation is computed as explained in section 4-3-3. For the arm locations that generate a successful commutation configuration, the motion to the target is planned (repeated for 50 times). For each successful plan, the path length is also noted. Accordingly, the real success rate, and path length to target is computed for all the possible locations of the arm using Eq. (5-1) and Eq. (5-2).

Using this data the following measures are computed for each scene:

- max (SR_{MP}): maximum success rate among filtered and sliced voxels
- min (PL): minimum path length among filtered and sliced voxels
- avg (SR_{MP}): average success rate for filtered and sliced voxels
- avg (PL): average path length for filtered and sliced voxels

Thereafter, these measures are compared against the path length obtained from IPM, and success rate obtained using IRM with the proposed algorithm.

Figure 5-12 shows that the success rate obtained using IRM as well as IFM is lesser than the maximum possible success rate. Specifically, the success rate at the commutation configurations generated by IFM are 4.75% and 12.2% less than the maximum for the table and shelf class respectively. It is interesting to note that the success rate when using IPM is also higher than the average value among the filtered voxels. This is due to the fact that placing the arm at many of the voxels does not result in a collision free IK solution.

Similarly, Figure 5-13 shows that the path length obtained using IFM lies between the average and the minimum possible values. The distance from the minimum value depends upon the planner used.

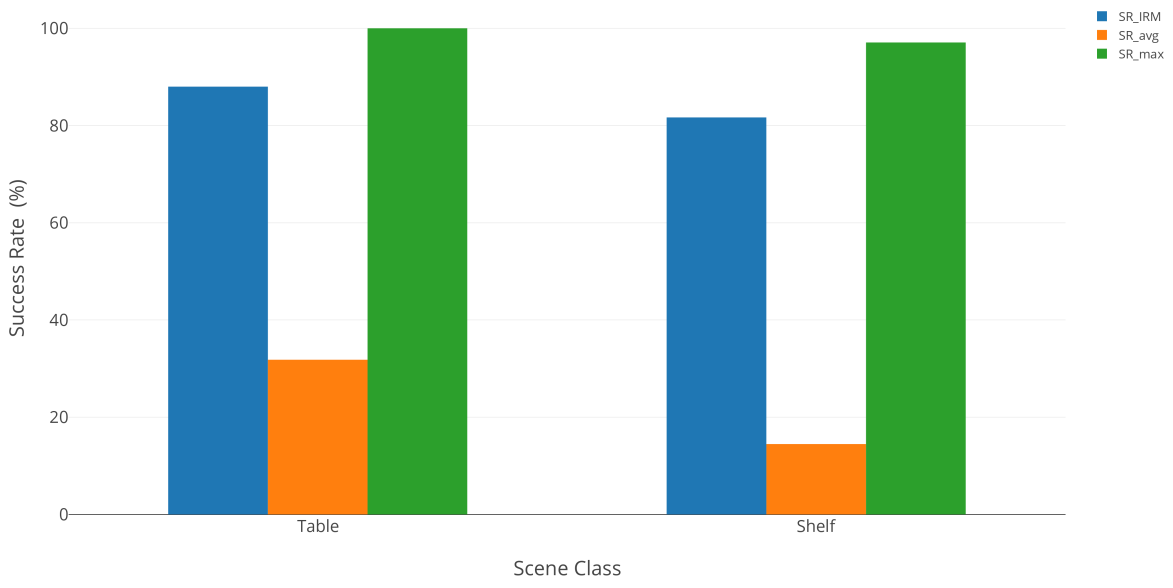


Figure 5-12: Benchmarking the success rate indicator, reachability for the table and shelf class. The success rate obtained by using IRM lies between the maximum and average success rates of all possible arm locations in the filtered and sliced IRM

As can be seen, the proposed approach suffers from some drawbacks.

- **Sub-Optimal within time constraints**: The indicators used for the computation of the commutation configuration are not optimal in terms of reachability and planability. This is due to two reasons. One, the indicators have been computed in free space and the quality values of the inverse reachability and planability map is not updated based on the octomap of obstacles in the scene. The second reason is due to the way planability measure has been defined. The planability compares the minimum amount of joint motion required to reach a target. While this is in accordance with other heuristics that use minimum path length to expand search nodes, this is not a perfect equivalent of the actual path length in presence of obstacles. This limitation is true for existing robot placement algorithms as well for the same reasons.

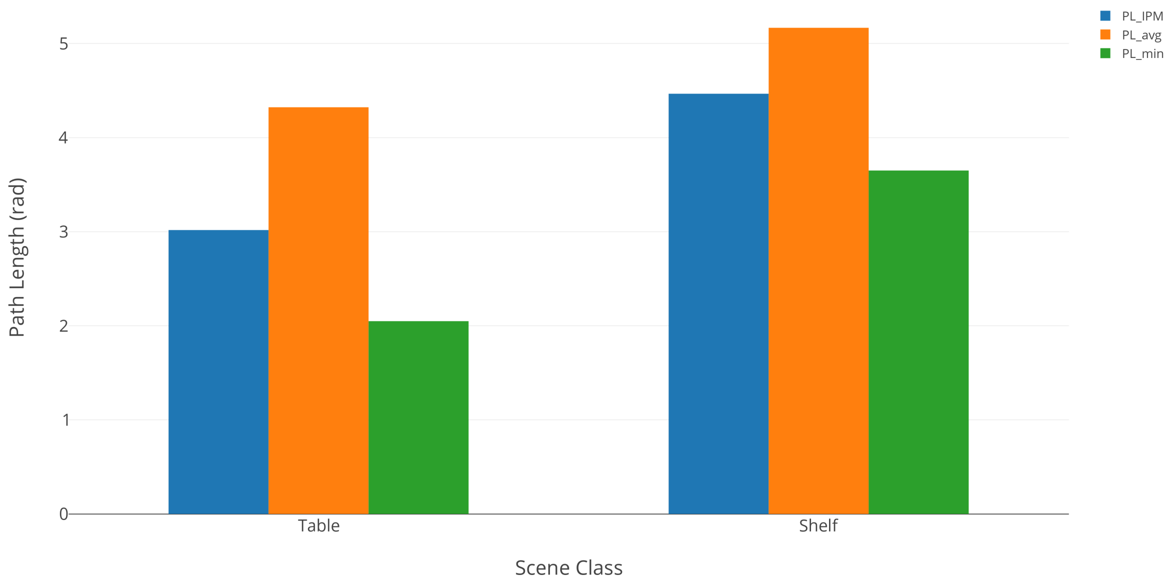


Figure 5-13: Benchmarking the path length indicator, planability for the table and shelf class. The path length obtained by using IPM lies between the minimum and average path lengths of all possible arm locations in the filtered and sliced IPM

- **Camera angle of view is not considered**: The camera viewing angle is limited. Therefore, the robot is incapable of viewing the target at the commutation configuration at all orientations. This constraint has not been accounted for in DeCOWA.
- **Applicability to fixed arm height**: The proposed algorithm is only applicable to mobile manipulators where the arm's height is fixed. In case the arm height can be altered, there can be a series of planes where the arm can be located before determining the appropriate platform orientation.
- **Re-computation for different resolutions**: While the inverse metric maps need not be re computed when the same arm is used along with a different mobile platform, the map must be recomputed on using a different resolution. This resolution is also dependent upon the tolerance value of the local planner for the platform as explained previously. When a different local planner is used, the inverse metric maps of a different resolution may need to be recomputed.

5-3 Summary

This chapter evaluated the performance of the proposed robot placement algorithm DeCOWA based on success rate, path length and robustness to actuation errors. It was found that the proposed algorithm performs better than random placement in terms of success rate by 66%. Furthermore, even in presence of actuation errors, the success rate remained higher than that obtained by random placement. The path length to target obtained by using the proposed algorithm was found to be 13.5% shorter than that from random placement. Also, the proposed algorithm was found to be within the set benchmark of 750ms. Even in presence of actuation errors, it was observed that DeCOWA is able to generate commutation configurations that give a 55.45% higher planning success rate than random placement. The chapter concluded with noting down limitations to the proposed approach. It was found that the locations generated by the algorithm within the desired time are not optimal and can be improved in the future by updating the inverse metric maps on the basis of obstacles present in the scene.

Chapter 6

Conclusion and Future Research

Research into Mobile Manipulation is at the forefront of robotics development. With an increasing elderly population in Europe, mobile manipulators have been proposed as a viable solution for personalized geriatric care. Robot Care Systems (RCS) is one of the companies developing a mobile manipulator for domestic use by the elderly. They have designed a prototype with a 6 degrees of freedom (dof) robotic arm mounted on top of a mobile robotic stroller named Lean Empowering Assistant (LEA). In order to utilizes the autonomous navigation and manipulation capabilities of LEA, an algorithm for smart robot placement was sought to be developed. A suitable robot placement algorithm generates a position and orientation for the robot at which it is able to successfully reach a target pose in an unconstrained environment. The approaches to robot placement available in literature focused predominantly on finding a location with high manipulability, and did not consider the quality of motion to the target or the sources of uncertainties in an unstructured setting. In this thesis, a novel robot placement algorithm **De**termining **C**ommutation-configuration using **O**ptimization and **W**orkspace **A**nalysis (DeCOWA) for domestic mobile manipulation was developed that presents a high chance of successful planning to the target through a short path, while accounting for actuation and sensing errors typical of the real world.

The design of such a robot placement algorithm depends upon the approach to mobile manipulation. Unlike sequential mobile manipulation, full body approaches require that motion planning is necessarily considered in combination with robot placement. These methods were compared against the metrics important in a domestic environment. For this thesis, sequential mobile manipulation was deemed suitable because it is safer, computationally efficient, and allows for the errors in locomotion to be corrected in the planning stage of manipulation. Furthermore, with variety of planners available for motion planning of the arm and the mobile base, sequential mobile manipulation can be used to generate smooth and consistent trajectories that appear comfortable to users. Accordingly, a modular framework for sequential mobile manipulation was developed using Robot Operating System (ROS). With sequential mobile manipulation, the output of the robot placement approach is the commutation configuration at which the robot switches from locomotion to manipulation. The commutation configuration serves as a goal for the navigation module of the mobile manipulator. With the designed framework, the robot placement algorithm could be reused to compute another solution if a path to the generated commutation configuration is not found.

Once the robot navigates to the chosen location, the ability to successfully reach the target depends upon the kinematic capabilities of the manipulator. Therefore, DeCOWA uses the reachability and planability maps for the manipulator created offline. These maps quantify the reaching capabilities of the manipulator, and measure the minimum path length to different regions in its workspace. By inverting the reachability and planability maps, the possible set of arm locations in SE(3) from which a target at origin can be reached is generated. Since the actuation error of the mobile platform affects the accuracy of reaching any arm location, the voxels of the maps are sized based upon the goal tolerance of the navigation local planner at 0.1m. During online use, the inverse reachability (IRM) and planability maps (IPM) are fused together by taking the ratio of reachability of a voxel to its planability cost generating the inverse fusion map (IFM). This allows to rank the different voxels in space based on the number of ways the arm can plan a short path to the origin when placed at that location. Given an end effector target to be reached, the IFM is transformed to the target pose. Thereafter, the map is sliced to the plane at arm's height, parallel to the ground, and the 2D cost map of the scene is used to discard those voxels that fall in the obstacle region. Through an iterative search, the manipulator is placed at each of the remaining voxels in the decreasing order of their quality. For the voxels at which a collision free inverse kinematic solution to the target pose is available, the orientation of the mobile platform is determined by minimizing the distance to the current location of the robot through Nelder Mead. Throughout the process, collision checking is performed using Flexible Collision Library (FCL) with an inflated URDF model of the robot in MoveIt!, thereby allowing to account for errors in sensing. The iterations terminate as soon as a collision free commutation configuration is determined or when all the filtered voxels have been examined. The algorithm is noted to be modular, since several planability maps can be created to represent different criteria if required.

The developed mobile manipulation framework is utilized to test the proposed algorithm on LEA for its performance and robustness to errors. Twenty scenes mimicking real world scenarios that encompass the variety of challenges to robot placement are considered for experimentation in V-REP. The first set of experiments compares the functioning of the proposed algorithm against random placement of robot near the target. It is found that the planning success rate when placing the robot at the pose generated through DeCOWA is 66% higher than that using Random Placement. Furthermore, the path length to target when using DeCOWA is 13.5% shorter than that from random placement when using RRTConnect. Even in presence of actuation errors, the success rate in reaching the target pose with robot placement through DeCOWA is 55.45% higher than that using random placement. This clearly demonstrates the efficacy of DeCOWA against desired characteristics. Furthermore, the average time required to compute the commutation configuration through DeCOWA is 210.58ms that is within the set benchmark of 750ms.

In summary, DeCOWA is shown to work for a variety of scenes that encompass various challenges typical of a real world setting. Along with the success of the algorithm, a few limitations are noted. The primary limitation is sub optimality of the generated commutation configuration for success rate or path length when compared against exhaustive search. This is acceptable with the fast performance of the approach and is typical of other robot placement algorithms in literature as well. In conclusion, the proposed algorithm is found to be modular and suitable for use in domestic environments while satisfying the objectives set for the thesis.

6-1 Future Work

The thesis presented a novel algorithm for robot placement considering the characteristics of a domestic environment. The following are some of the possible improvements or future research directions:

- **Account for obstacles**: As was noted, the sub-optimality of the algorithm is because the inverse reachability map is not updated based upon the obstacles present in the scene. This can be considered by precomputing additional information. Presently, only the reachability of a voxel is determined based on the number of poses the robot can reach in it. Correspondingly, the configurations of the manipulator at reachable poses can be noted. This allows to associate all the voxels that the configuration passes through in order to reach the pose with reachability of the voxel. In order to update the reachability map, all voxels in collision with the octomap must be removed. Accordingly, all the configurations that pass through these colliding voxels must also be removed, thereby modifying the number of poses reachable and dynamically updating the ranking of the voxels. This is illustrated in Figure 6-1. Research on the aforementioned approach is still being conducted.

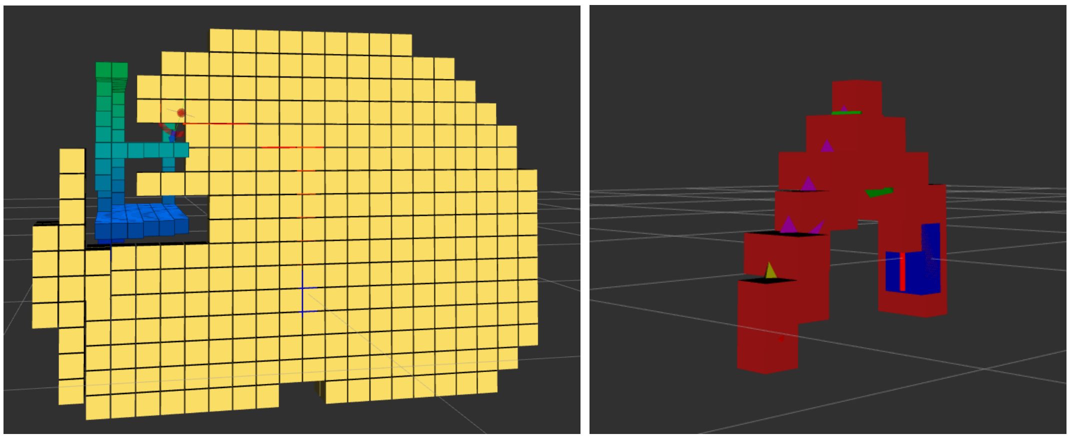


Figure 6-1: Directions to create collision aware reachability maps. Left: A crossection of reachability map is shown. Voxels colliding with the obstacle octomap are removed. Right: The voxels intersecting with a configuration of the arm are shown.
This method is being built to update the reachability or inverse reachability map online based upon the obstacles in the scene.

- **Better measures of quality**: Since the planability map measures the path length of the robot in free workspace and uses it as an indicator to quality of motion, the proposed approach generates commutation configurations that are sub optimal in terms of path length. It would be interesting to explore other indicators for a short path length, and to find ways of improving it in the presence of obstacles.

- **Multiple metrics of quality**: The presented approach only considers the quality of motion for robot placement in terms of the path length. As explained earlier, this can be extended to other measures such as power consumption by creating similar planability maps. Therefore approach presented in the thesis can also be extended to account for multiple quality metrics at the same time. The updated format of fusing multiple inverse metric maps could be a weighted sum of the planability costs as shown.

$$Q_i^{inverse} = \frac{R_i^{inverse}}{w_1 \cdot C_{1,i}^{inverse} + w_2 \cdot C_{2,i}^{inverse} + w_3 \cdot C_{3,i}^{inverse} + \dots}.$$

- **Multiple target poses**: The approach presented only finds suitable robot placement to reach a single target pose. This can easily be extended to reaching multiple targets. This will be especially useful for non holonomic mobile manipulators as it is hard for them to maneuver small distances. One of the ways this could be achieved is through considering intersections of the filtered and sliced inverse metric maps. As shown in Figure 6-2 only the voxels in the intersection can allow the arm to reach all the target poses. The $R_i^{inverse}$ and $C_i^{inverse}$ of the maps could be updated using

$$R_{i,intersection}^{inverse} = R_{i,p_1}^{inverse} \times R_{i,p_2}^{inverse} \times R_{i,p_3}^{inverse} \times \dots$$
$$C_{i,intersection}^{inverse} = C_{i,p_1}^{inverse} + C_{i,p_2}^{inverse} + C_{i,p_3}^{inverse} + \dots$$

where $R_{i,intersection}^{inverse}$ and $C_{i,intersection}^{inverse}$ represent the reachability measure, and planability cost of the voxels common to maps around target p_1, p_2, p_3 and so on.

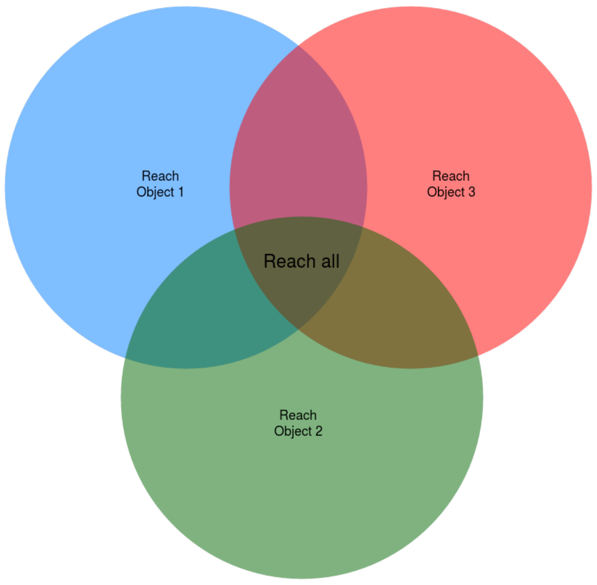


Figure 6-2: Intersection of multiple inverse metric maps. The figures shows the set of commutation configurations from which object 1, 2, and 3 can be reached. The intersection of those include voxels from which the arm can reach all the voxels.

- **Robot placement for non-wheeled mobile manipulators**: It would be interesting to extend DeCOWA to application in humanoids and flying mobile manipulators with some changes (for example, the humanoid can bend its legs and hence change the height of the arm from ground. This angle of bending must also be determined).

Appendix A

LEA and arm specifications

For LEA,

- Main body:
  - Height: 550 mm (minimum)
  - Breadth: 550 mm
  - Width: 600 mm
- Wheel
  - Radius: 100 mm
  - Width: 40 mm
- Total Weight: Approximately 29.0 kg

For the arm,

- Joints: Serial six axis kinematic
- Payload: 1.0 kg at half reach, and 750 g when fully extended
- Reach: 850 mm
- Weight: Approximately 6.0 kg

Appendix B

Parabolic Path with PID control for Navigation

When using the parabolic path for navigation as explained in section 3-2-3 the equation of the parabola is given by

$$y' = a(x' - h)^2 + k.$$

Here the (x', y') represents the coordinates in the reference frame placed at the goal. The parabola is symmetrical about the y' axis whose vertex is at (h, k). In this way, the robot will be facing the origin of the reference frame when it reaches the goal.

The trajectory to be followed by the robot is given by parabola's time derivative

$$\dot{y}' = 2a(x' - h)\dot{x}'.$$

When the robot reaches the goal, k is zero. Therefore,

$$y' = a(x' - h)^2. \tag{B-1}$$

Accordingly the coefficient of the parabola is given by

$$a = \frac{y'_0}{(x'_0 - h)^2} \tag{B-2}$$

where (x'_0, y'_0) is the starting point of the robot in the goal coordinate system.

Let the linear velocity of the robot be v, and its current yaw angle be given by θ'. Then by using the unicycle model,

$$\dot{x}' = vcos(\theta') \tag{B-3}$$
$$\dot{y}' = vsin(\theta') \tag{B-4}$$
$$\dot{\theta}' = \omega \tag{B-5}$$

where ω is the angular velocity input to the robot. If the rate at which the robot moves in the x' direction is fixed to $v_{x'}$. Then,

$$\dot{y}' = 2a(x' - h)v_{x'}.$$

The desired angle at each point of the trajectory (x', y') is given by its tangent

$$\theta'_d = tan^{-1}(\dot{y}'/\dot{x}'). \tag{B-6}$$

Hence, at each point in trajectory we can define the error in angle as

$$e(t) = \theta'_d(t) - \theta'(t). \tag{B-7}$$

A P(I)D controller given by

$$\dot{\theta}'(t) = K_p \times e(t) + K_d \times \dot{e(t)} + K_i \times \int_0^t e(t)dt \tag{B-8}$$

is used for feedback control of the yaw. To set the gains of the PID controller the Ziegler Nichols method [93] is used.

Bibliography

[1] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pp. 258–265, ACM, 2006.

[2] R. Tschakarow, S. M. Grigorescu, and A. Gräser, "Friend-a dependable semiautonomous rehabilitation," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–7, VDE, 2010.

[3] H. Nguyen, C. Anderson, A. Trevor, A. Jain, Z. Xu, and C. C. Kemp, "El-e: An assistive robot that fetches objects from flat surfaces," in *Robotic helpers, int. conf. on human-robot interaction*, 2008.

[4] K. Wada, T. Shibata, T. Asada, and T. Musha, "Robot therapy for prevention of dementia at home," *Journal of Robotics and Mechatronics*, vol. 19, no. 6, p. 691, 2007.

[5] "International federation of robotics." http://www.ifr.org/service-robots/. Accessed: October, 2016.

[6] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2016.

[7] D. C. A. for Statistics, "Highlights of population forecast, 2010-2060."

[8] R. S. W. van der Windt and E. Arnold, "The labour market: nurses, caregivers, and social agents 2008-2012."

[9] R. Kittmann, T. Fröhlich, J. Schäfer, U. Reiser, F. Weißhardt, and A. Haug, "Let me introduce myself: I am care-o-bot 4, a gentleman robot," *Mensch und computer 2015–proceedings*, 2015.

[10] "Robot care systems." www.robotcaresystems.com/en/. Accessed: October, 2016.

[11] D. Berenson, J. Kuffner, and H. Choset, "An optimization approach to planning for mobile manipulation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1187–1192, IEEE, 2008.

[12] V. Pilania and K. Gupta, "A hierarchical and adaptive mobile manipulator planner with base pose uncertainty," *Autonomous Robots*, vol. 39, no. 1, pp. 65–85, 2015.

[13] "Robotnik fetch." http://www.robotnik.eu/manipulators/fetch/. Accessed: March, 2017.

[14] R. Bischoff, U. Huggenberger, and E. Prassler, "Kuka youbot-a mobile manipulator for research and education," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.

[15] W. Garage, "Overview of the pr2 robot," 2009.

[16] C. Borst, T. Wimböck, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. R. Giordano, R. Konietschke, W. Sepp, S. Fuchs, *et al.*, "Rollin'justin-mobile platform with variable base.," in *ICRA*, pp. 1597–1598, 2009.

[17] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 20–29, 2007.

[18] "Rollin' justin website." http://www.dlr.de/rmc/rm/en/desktopdefault.aspx/tabid-11427/#gallery/29202. Accessed: March, 2017.

[19] A. Hornung, M. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 423–429, IEEE, 2012.

[20] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, "Ikeabot: An autonomous multi-robot coordinated furniture assembly system," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 855–862, IEEE, 2013.

[21] P. Stone, "What's hot at robocup.," in *AAAI*, pp. 4346–4348, 2016.

[22] "Robocup@home rulebook." https://latexonline.cc/compile?git=git%3A%2F%2Fgithub.com%2FRoboCupAtHome%2FRuleBook.git&target=Rulebook.tex. Accessed: May, 2017.

[23] M. Ciocarlie, K. Hsiao, A. Leeper, and D. Gossow, "Mobile manipulation through an assistive home robot," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5313–5320, IEEE, 2012.

[24] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3229–3236, IEEE, 2007.

[25] "Darpa robotics challenge (drc)." https://www.darpa.mil/program/darpa-robotics-challenge. Accessed: October, 2017.

[26] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM transactions on graphics (TOG)*, vol. 23, pp. 309–314, ACM, 2004.

[27] N. Vahrenkamp, T. Asfour, and R. Dillmann, “Robot placement based on reachability inversion,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1970–1975, IEEE, 2013.

[28] P. Abolghasemi, R. Rahmatizadeh, A. Behal, and L. Boloni, “A real-time technique for positioning a wheelchair-mounted robotic arm for household manipulation tasks,” in *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[29] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of kinect depth data for indoor mapping applications,” *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.

[30] “Move base package.” Accessed Feb 2017.

[31] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, 2009.

[32] W. Garage, “Universal robot description format (urdf),” *Http://Www. ros. org/urdf/, 2009*, 2009.

[33] P. Abolghasemi, R. Rahmatizadeh, A. Behal, and L. Bölöni, “Real-time placement of a wheelchair-mounted robotic arm,” in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*, pp. 1032–1037, IEEE, 2016.

[34] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann, “Manipulability analysis,” in *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pp. 568–573, IEEE, 2012.

[35] D. M. Olsson and L. S. Nelson, “The nelder-mead simplex procedure for function minimization,” *Technometrics*, vol. 17, no. 1, pp. 45–51, 1975.

[36] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, “Human-aware robot navigation: A survey,” *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[37] E. T. Hall, “The hidden dimension,” 1966.

[38] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, “A human aware mobile robot motion planner,” *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[39] E. A. Sisbot, A. Clodic, R. Alami, and M. Ransan, “Supervision and motion planning for a mobile manipulator interacting with humans,” in *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pp. 327–334, IEEE, 2008.

[40] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3d mapping framework based on octrees,” *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[41] R. W. Brockett *et al.*, “Asymptotic stability and feedback stabilization,” *Differential geometric control theory*, vol. 27, no. 1, pp. 181–191, 1983.

[42] B. Cohen, I. A. Şucan, and S. Chitta, “A generic infrastructure for benchmarking motion planners,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 589–595, IEEE, 2012.

[43] L. Sciavicco and B. Siciliano, *Modeling and control of robot manipulators*, vol. 8. McGraw-Hill New York, 1996.

[44] M. Pivtoraiko and A. Kelly, “Generating near minimal spanning control sets for constrained motion planning in discrete state spaces,” in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3231–3237, IEEE, 2005.

[45] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “Stomp: Stochastic trajectory optimization for motion planning,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4569–4574, IEEE, 2011.

[46] I. A. Sucan, M. Moll, and L. E. Kavraki, “The open motion planning library,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[47] D. Chen, Z. Liu, and G. von Wichert, “Uncertainty-aware arm-base coordinated grasping strategies for mobile manipulation,” *Journal of Intelligent & Robotic Systems*, vol. 80, p. 205, 2015.

[48] B. Du, J. Zhao, and C. Song, “Optimal base placement and motion planning for mobile manipulators,” in *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1227–1234, American Society of Mechanical Engineers, 2012.

[49] D. Hershberger, D. Gossow, and J. Faust, “Rviz,” 2011.

[50] J. J. Kuffner and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in *Robotics and Automation, 2000. Proceedings. ICRA’00. IEEE International Conference on*, vol. 2, pp. 995–1001, IEEE, 2000.

[51] S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev, “Multi-heuristic a*,” *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 224–243, 2016.

[52] L. Sentis and O. Khatib, “A whole-body control framework for humanoids operating in human environments,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2641–2648, IEEE, 2006.

[53] K. Gochev, A. Safonova, and M. Likhachev, “Planning with adaptive dimensionality for mobile manipulation,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 2944–2951, IEEE, 2012.

[54] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[55] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[56] Y. Yang and O. Brock, “Elastic roadmaps - motion generation for autonomous mobile manipulation,” *Autonomous Robots*, vol. 28, no. 1, pp. 113–130, 2010.

[57] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[58] "Costmap 2d package." Accessed Feb 2017.

[59] B. Bayle, J.-Y. Fourquet, and M. Renaud, "Manipulability of wheeled mobile manipulators: Application to motion generation," *The International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 565–581, 2003.

[60] B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2902–2908, IEEE, 2010.

[61] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, "Positioning mobile manipulators to perform constrained linear trajectories," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 2578–2584, IEEE, 2008.

[62] F. Zacharias, W. Sepp, C. Borst, and G. Hirzinger, "Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pp. 55–61, IEEE, 2009.

[63] P. Jauer, I. Kuhlemann, F. Ernst, and A. Schweikard, "Gpu-based real-time 3d workspace generation of arbitrary serial manipulators," in *Control, Automation and Robotics (ICCAR), 2016 2nd International Conference on*, pp. 56–61, IEEE, 2016.

[64] O. Porges, T. Stouraitis, C. Borst, and M. A. Roa, "Reachability and capability analysis for manipulation tasks," in *ROBOT2013: First Iberian Robotics Conference*, pp. 703–718, Springer, 2014.

[65] T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "Armar-iii: An integrated humanoid platform for sensory-motor control," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pp. 169–175, IEEE, 2006.

[66] F. Stulp, A. Fedrizzi, L. Mösenlechner, and M. Beetz, "Learning and reasoning with action-related places for robust mobile manipulation," *Journal of Artificial Intelligence Research*, vol. 43, pp. 1–42, 2012.

[67] J. Yang, P. Dymond, and M. Jenkin, "Reaching analysis of wheelchair users using motion planning methods," *Impact Analysis of Solutions for Chronic Disease Prevention and Management*, pp. 234–237, 2012.

[68] A. Vatsyayan, "Vision assisted motion planning of robotic arm for service robots," *TU Delft Repository.*

[69] "Octomap server package." Accessed Feb 2017.

[70] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1321–1326, IEEE, 2013.

[71] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[72] S. Thrun and J. J. Leonard, "Simultaneous localization and mapping," in *Springer handbook of robotics*, pp. 871–889, Springer, 2008.

[73] L. Almeida, P. Pedreiras, and J. A. G. Fonseca, "The ftt-can protocol: Why and how," *IEEE transactions on industrial electronics*, vol. 49, no. 6, pp. 1189–1201, 2002.

[74] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3859–3866, IEEE, 2012.

[75] 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.

[76]

[77] C. Robotics, "Rosinterface."

[78] C. Robotics, "Rosplugin."

[79] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2149–2154, IEEE, 2004.

[80] L. Nogueira, "Comparative analysis between gazebo and v-rep robotic simulators," *Seminario Interno de Cognicao Artificial-SICA 2014*, p. 5, 2014.

[81] "Reuleaux package for reachability analysis." Accessed Feb 2017.

[82] H. Samet, "An overview of quadtrees, octrees, and related hierarchical data structures," *NATO ASI Series*, vol. 40, pp. 51–68, 1988.

[83] J. Hammersley, *Monte carlo methods.* Springer Science & Business Media, 2013.

[84] E. B. Saff and A. B. Kuijlaars, "Distributing many points on a sphere," *The mathematical intelligencer*, vol. 19, no. 1, pp. 5–11, 1997.

[85] R. Diankov, "Ikfast: The robot kinematics compiler," 2010.

[86] M. Folk, A. Cheng, and K. Yates, "Hdf5: A file format and i/o library for high performance computing applications," in *Proceedings of Supercomputing*, vol. 99, pp. 5–33, 1999.

[87] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.

[88] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration.," *VISAPP (1)*, vol. 2, no. 331-340, p. 2, 2009.

[89] S. Johnson, "The nlopt nonlinear-optimization package [software]," 2014.

[90] Y. Yang, V. Ivan, Z. Li, M. Fallon, and S. Vijayakumar, “idrm: Humanoid motion planning with realtime end-pose selection in complex environments,” in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pp. 271–278, IEEE, 2016.

[91] “Abb.” `http://new.abb.com/`. Accessed: March, 2017.

[92] “Kuka.” `https://www.kuka.com/en-de/`. Accessed: January, 2017.

[93] J. G. Ziegler and N. B. Nichols, “Optimum settings for automatic controllers,” *trans. ASME*, vol. 64, no. 11, 1942.

Glossary

List of Acronyms

API	Application Programming Interface
CAN	Controller Area Network
DARPA	Defense Advanced Research Projects Agency
dof	degrees of freedom
ERS	Ease of Reachability Score
DeCOWA	**De**termining **C**ommutation-configuration using **O**ptimization and **W**orkspace **A**nalysis
DWA	Dynamic Window Approach
ERM	Elastic Roadmap
FCL	Flexible Collision Library
FK	Forward Kinematics
FRIEND	Functional Robot arm with user-frIENdly interface for Disabled people
GPU	Graphics Processing Unit
GUI	Graphical User Interface
IFR	International Federation of Robotics
IK	Inverse Kinematics
IFM	Inverse Fusion Map
IMU	Inertial Measurement Units
IP	Internet Protocol

IPM	Inverse Planability Map
IRM	Inverse Reachability Map
LEA	Lean Empowering Assistant
NN	Nearest Neighbor
OMPL	Open Motion Planning Library
PC	Personal Computer
PCL	Point Cloud Library
PID	Proportional-Integral-Derivative
RAM	Random Access Memory
RCS	Robot Care Systems
RGB-D	Red Green Blue - Depth
ROS	Robot Operating System
RRT	Rapidly Exploring Random Trees
SBPL	Search Based Planning Library
SLAM	Simultaneous Localization and Mapping
SE	Special Euclidean
STOMP	Stochastic Trajectory Optimization for Motion Planning
TCP	Transmission Control Protocol
TLD	Tracking-Learning-Detection
URDF	Universal Robot Description Format
V-REP	Virtual Robot Experimentation Platform