University of Tartu

Faculty of Science and Technology

Institute of Technology

Muhammad Usman

Development of an Optimization-Based Motion Planner and Its ROS Interface for a Non-Holonomic Mobile Manipulator

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisors:

Arun Kumar Singh
Karl Kruusamäe

Tartu 2020

Abstract

Development of an Optimization-Based Motion Planner and Its ROS Interface for a Non-Holonomic Mobile Manipulator

The application of mobile manipulators is expanding to different fields such as space, underwater, construction, service and, health-care, as such robotic systems provide the ability to move around the environment and manipulate objects. The thesis presents an optimization-based motion planning algorithm for a manipulator mounted on a non-holonomic mobile base. In particular, this work deals with the sub-class of problems called task-constrained trajectory optimization, where the end-effector position and orientation are given by the user as input, and the algorithm computes the necessary joint motions of the manipulator and the mobile base. This class of problems is especially important for applications like 3D printing and robotic painting, where the mobile base and the manipulator needs to be moved simultaneously. The proposed algorithm computes smooth (as defined by higher-order differentiability) motions for both manipulator and mobile base also, it provides hyper-parameters that can be tuned to trade-off different aspects of the motions. The proposed trajectory optimization is implemented on hardware consisting of a UR5e arm mounted on the top of the MiR100 mobile base. To achieve a rigorous implementation, the thesis also develops a custom Robot Operating System (ROS) interface for the aforementioned hardware.

Keywords: Mobile manipulator, motion mlanning, optimization, ROS

CERCS: T125 Automation, robotics, control engineering

Abstract in Estonian

Optimeerimisele Baseeruva Liikumisplaneerija Arendamine ja SelleROSi Liides Mitteholonoomse Mobiilse Manipulaatori Jaoks

Mobiilsete manipulaatorite rakendamine laieneb erinevatesse aladessenagu vabas ruumis, vee all, ehitus, teenindus ja tervishoiu, kuna sellisedseadmed pakkuvad võimalust liikuda keskonnas ning manipuleeridaesemeid. Väitekirjas on esitatud optimeerimisele baseeruv liikumisekavandav algoritm manipulaatori jaoks, mis asub mitteholonoomselalusel. Täpsemalt tegeletakse töö raames probleemide alamhulkaganimega ülesande-piiratud trajektoori optimiseerimine, kus lõpp-mõjuriasukohta ja suunda on antud kasutaja poolt sisendina, ja algoritmarvutab manipulaatori ja aluse vajalikuid ühiseid liikumisi. Sellise tüübiprobleemid on eriti tähtsad rakenduste jaoks nagu 3D printimine jarobootne värvimine, kus mobiilse aluse ja manipulaatori tuleb liigutadasamaaegselt. Esitatud algoritm arvutab siledaid (s.t. kõrgejärguliseltdiferentseeruvaid) liikumisi nii manipulaatori kui aluse jaoks, ja samal ajalpakub hüperparameetreid, mille abil on võimalik sättida liikumiseerinevaid ilmeid. Pakutud teekonna optimeerimine on teostatudriistvaras, mis koosneb mobiilsel MiR100 alusel asetatud UR5e kodarast.Saavutamaks ranget teostust on väitekirjas arendatud tavapäranerobootse operatsioonisüsteemi (ROS) liides ülalmainitud riistvara jaoks.

Võtmesõnad: Mobiilne manipulaator, liikumise kavadamine, optimeerimine, ROS

CERCS: T125 Automatiseerimine, robootika, juhtimistehnika

Contents

List of Figures

List of Tables

1 Introduction

Robots are intelligent machines designed to assist human beings, and their scope of application spans across diverse fields such as bio-mechanics, entertainment, surgery, medical and healthcare, military, etc. [1].
A mobile manipulator is a robotic system that consists of a manipulator mounted on the top of a mobile platform. A mobile manipulator leverages the capability of the mobile platform to move in an environment, and thus has the ability to manipulate objects in an extensive workspace compared to a fixed base manipulator. The potential applications of mobile manipulators are in healthcare [2], construction [3], nuclear reactor maintenance [4], planetary exploration [5] among others. For many applications, a mobile manipulator can be considered as two separate systems, and their individual motions can be coordinated in sequence. For example, the mobile platform moves to a specified position while maintaining the manipulator stationary, and then keeping the mobile platform stationary, the manipulator is used to grasp objects. However, certain applications such as robotic 3D (three dimensional) printing [6], requires the mobile base and the manipulator to be moved simultaneously to ensure that the end-effector follows a given trajectory [7]. Subsequently, this requires synchronous coordination of the mobile base and the manipulator.
The problem of computing the required coordination comes within the ambit of motion planning, a computational technique to generate a sequence of valid configurations connecting the robot's initial pose to the goal pose. Motion planning of mobile manipulators is challenging due to the complex non-linear nature of the kinematics. In the context of this thesis, a specific case of mobile base with the so-called non-holonomic behavior is considered. Intuitively, this refers to the condition that the mobile base can only move forward/backward along its heading, similar to a car. This in turn, results in a highly non-linear coupling between the mobile base and the manipulator. The complexity of motion planning is also increased due to the presence of kinematics redundancy, i.e the mobile base and the manipulator together posses more degree of freedom than that required to perform a given manipulation task. Thus, the motion planning algorithm should be sophisticated enough to extract the best possible solutions among several possibilities.
Mobile manipulation also presents key challenges at the hardware level. Several existing robots are controlled through a universal application programming interface (API) called Robot Operating System (ROS) that provides extensive software libraries for controlling manipulators. However, support for mobile manipulators is limited, and thus require to perform the software and hardware integration from scratch.

1.1 Objectives and Contribution

The thesis presents a unified approach for addressing the above challenges and consists of the following objectives.

- To develop a motion planner that can take into account the various task-level objectives (such as goal reaching) of mobile manipulation while satisfying constraints on motion of the manipulator and the mobile base.
- To develop a ROS interface for controlling the in-house mobile manipulation platform consisting of a MiR100 industrial mobile base and a UR5e manipulator.

The thesis presents the following contribution in light of the above mentioned objectives.

- It presents an optimization-based motion planning algorithm with the following useful features.
  - The proposed trajectory optimization approach produces smooth manipulator and mobile base joint trajectories as measured by the acceleration profile. This is achieved by parameterizing the motions in terms of time dependent polynomials and formulating the costs and constraints in terms of the coefficients of the polynomial.
  - The proposed trajectory optimization approach maintains cyclicity which means that the closed cyclic trajectories for the end-effector lead to closed cyclic trajectories for the mobile base and the manipulator. This, in turn, proves useful when multiple repetitions of a given cyclic trajectory need to be performed. The cyclicity feature essentially ensures that it is possible to replay the manipulator and base motions to achieve this.
- The thesis also develops a ROS based software set-up for executing the computed motions plans on an actual mobile manipulator hardware. This involves integrating the kinematics and control description of the combined mobile base and manipulator system in the ROS architecture and developing the interface for executing velocity and joint commands on the robot.
- Finally, the thesis also presents extensive simulation and experimental results to validate the efficacy of the proposed algorithms and software set-up.

1.1.1 Functional Requirements

- Generate smooth trajectories for mobile manipulators to follow a given cyclic trajectory.
- Generate the corresponding velocity commands for the mobile base and joint motion for the manipulator.
- Track the end-effector position using ar-markers.
- Compare joint motions with KDL (Kinematics and Dynamics Library) method.

1.1.2 Software Requirements

- Ubuntu 18.04 for running ROS Melodic Distribution
- Python, including the Scipy, Sympy, Numpy, and Auograd libraries, required for developing the trajectory optimization algorithm.
- C++ programming language to create ROS nodes, needed to send commands to the mobile manipulator.
- Use KDL as a kinematics solver to compare results with the proposed optimization method.

1.1.3 Hardware Requirements

- ROS-enabled mobile manipulator platform.

1.2 Organization of Thesis

The content of this thesis is organized into 6 chapters. current chapter presents the introduction, motivation, contributions and objectives of the thesis. The concepts of mobile manipulator, motion planning and ROS are given in chapter 2. Chapter 3 describes the trajectory optimization approach and how to compute the motions for the mobile manipulators. The implementation of the proposed method both in simulation and real robot is presented in chapter 4. The results of the weight tuning and convergence validation, comparisons of joint motion with KDL and end-effector trajectory tracking analysis are presented in chapter 5. Conclusions and future work are presented in chapter 6.

2 Literature Review

2.1 Stationary Robots

Conventional manipulator robots consist of several metal segments which are connected together by joints. Traditionally, these kinds of manipulators are bolted on the fixed bases and are designed to assist humans in performing dirty, dangerous, repetitive and tedious tasks such as welding [8], painting [9], material handling [10] and industrial manufacturing [11]. Despite the great advantages, these manipulators only can offer limited operational workspace and reachability.

2.2 Mobile Manipulator

A mobile manipulator consists of a robotic manipulator mounted on top of the mobile base that combines the dexterous manipulation offered by the manipulator and mobility provided by the mobile base. As compared to stationary manipulators, mobile manipulators have a significantly large and extensive workspace that allows them to perform a wide array of tasks that need locomotion and manipulation abilities [12].

2.3 Application of Mobile Manipulator

The application of mobile manipulator robots are expanding to space exploration [5], home-care [13], health-care [2], search and rescue [14], nuclear reactor maintenance [4], military [15], construction [3], etc and they are not restricted to only industrial environments. In [16] the author categorizes different applications of mobile manipulators into four major domains: professional/service (home and health-care), space exploration, military, and industry (Figure 2.1).
Recently, the environment for the applications of mobile manipulators has turned from factory environments to human environments, due to the fact that they are well suited for human tasks, therefore, such robots are currently present in homes (assisting elderly and/or disabled peoples), offices , hospitals , etc [17]. For instance, a mobile manipulator with ability to manipulate the environment is widely used for domestic applications such as MOVIAD [18], EL-E [19], Care-o-bot II [13] and PR2 [20], which were developed in research institutes to provide mobile manipulation assistance in the home environment for elderly and disabled persons.

Figure 2.1: *Major applications of mobile manipulator: professional/service (home and health-care), space exploration, military, and industry [16].*

In the past few years, mobile manipulators have gained significant interest and their applications extended towards different environments such as underwater and aerial environments [21]. Autonomous underwater vehicles (AUVs) equipped with robotic manipulators (known as autonomous underwater vehicle manipulator) are used to perform many underwater intervention tasks such as oil exploration, installation of underwater telecommunication cables, marine search and rescue and military services [22]. Figure 2.2 shows the TRIDENT project underwater intervention demonstration where Girina 500 I-AUV recovering marine black-box with a 7-DOF manipulator autonomously [23].

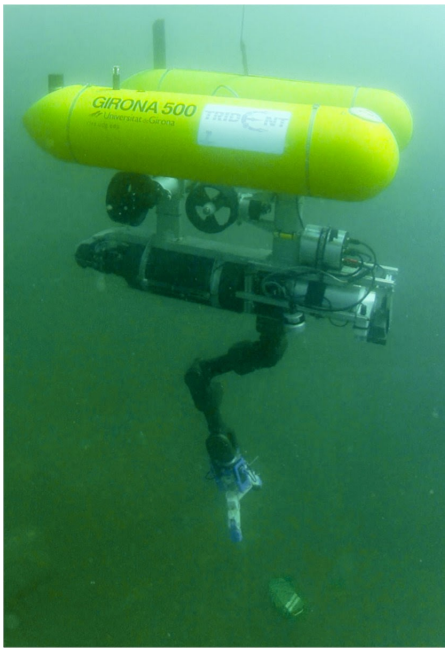


Figure 2.2: *TRIDENT project: underwater intervention demonstration [23].*

Furthermore, aerial manipulation systems are composed of unmanned aerial vehicles (UAVs) and robotic arms which combine the mobility and agility of UAVs with manipulation dexterities of manipulators, used for manipulation tasks such as simple collection, transporting, assembly and disassembly of mechanical parts and high-voltage transmission lines inspection and repairs, etc [24]. Suarez et al. [25] presented the aerial manipulation consisting of a hexarotor platform and dual-arm manipulators. Figure 2.3 shows an aerial manipulator system which is developed in the AEROARMS project for object grasping and release operations.

Figure 2.3: *AEROARMS project: An aerial manipulator system consisting of a hexarotor platform and dual-arm system [25] .*

2.4 Motion Planning

Motion planning is a technique to generate a sequence of valid configurations that moves the robot gradually from the initial pose to the goal pose. One of the fundamental robotics task is to plan collision-free motions for robots to move in an environment with obstacles and without colliding with the robot links itself [1]. There are two general classes of motion planning algorithms which are commonly used to solve motion planning problems for mobile manipulator robots, called sampling-based motion planning and optimization-based motion planning [7].

2.4.1 Sampling-Based Motion Planning

The sampling-based motion planning is an algorithms which iteratively construct a graph by randomly sampling the configuration space (space of possible positions that the robot may reach). These algorithms are based on a collision checking module, that provides information about possible trajectories and connects a set of points sampled from the obstacle-free space

in order to build a graph of feasible trajectories [26]. Sampling-based algorithm is effective in high-dimensional spaces because it does not use an explicit representation of the environment and the obstacles, thus it decreases the search cost and runtime for most problems [26]. However, the main issue is that it is not able to find an optimal solution [27].
Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmap (PRM) are two different and common approaches to implement sampling-based algorithms [1]. In [28], Oriolo et al. implemented single-query probabilistic planners to generate collision-free motions for a non-holonomic mobile manipulator moving along a given end-effector path. In [29], Burget et al. present a planning framework for generating asymptotically optimal paths for mobile manipulators subject to task constraints using a bidirectional RRT in three scenarios: traverse an environment equipped with obstacles, transportation of container of liquids in narrow spaces and pulling a cart. The framework is able to generate solutions for the individual tasks taking into account the end-effector orientation constraints in each case. In [30], Jaillet et al. mention that having the kinematics constraints, the probability of obtaining a valid configuration by sampling in the joint space(all of the places that each joint of a robotic arm link can reach) is null, which complicates the sampling-based path planners. Using a combination of Atlas (define diagrams that locally configure a manifold and coordinate the diagrams to form an atlas) and RRT algorithms, it was possible to grow branches on the configuration space, controlling the directions of expansion with RRT, and thus find solution paths.

2.4.2 Optimization-Based Motion Planning

The motion planning of a robot can be formulated as a trajectory optimization problem. Trajectory optimization is the process of designing a trajectory that minimizes (or maximizes) an objective function such as velocity, acceleration and smoothness, while satisfying a set of constraints, for example, manipulator's joint limits and end-effector pose [31]. The common trajectory optimization planner for robotics manipulation applications are CHOMP: Covariant Hamiltonian Optimization for Motion Planning [32], STOMP: Stochastic Trajectory Optimization for Motion Planning [33] and TrajOpt [34]. In case of mobile manipulator, these algorithms model the end-effector path constrains as a non-linear equality constraint and provide the optimal solution.
An optimization-based motion planning algorithm was proposed in [35], in which the implemented coordination motion planning generates a trajectory for the mobile base and manipulator simultaneously with collision avoidance, joints limitation and other constrains. The algorithm consists of two nested loops, one for a penalty factor for the inequality constraints, and the second loop is used to find a collision-free trajectory solving the equality constrains iteratively.
In [36], Bersenon et al. present an optimization approach for grasping and path planning for mobile manipulators performing pick-and-place tasks. The algorithm has two phases: optimization and planning. In the optimization phase, a co-evolutionary algorithm was used to find the optimal robot configurations and grasp for the object in its initial and goal pose, and in the planning phase, the path between the two robot configuration was computed by using a bidirectional RRTs. A Sequential Convex Optimization method was implemented by Schulman et al. [37] to find collision free trajectories for a 7-DOF robotic arm and an

18-DOF full body humanoid robot in a walking motion task. The optimization algorithm repeatedly constructs convex sub-problems and turn the infeasible constraints into penalties, that are multiplied with a coefficient to ensure that the constraint violation is driven to zero.

2.4.3 Challenges in Mobile Manipulation

The key objective of motion planning is to compute the inverse kinematics, which is redundant as it has more degrees of freedom than the necessary to perform a specific task. One of the most critical issues is the coordination between the mobile platform and the n-degree-freedom robotic arm [38].
Mobile manipulators need to perform integrated control of the two components to achieve efficient motion control. Each hardware has their own control system, and integrating these to produce the desired outcome can be challenging. In applications like pick an object, it may be sufficient to move the mobile platform to a desired location, and then perform some manipulation, however, tasks like robotic 3D printing while moving [6] require a closer coupling of these control systems [39].
In [40], Gao et. al. state that the main challenges with mobile manipulators are related to the motion planning and coordinated control due to the redundancy resolution problem. In [21], Khatib mention that mobile manipulators have limited abilities for manipulation and interaction with humans, which depend largely on the full integration of mobility, manipulation, and interaction. Therefore, control strategies are needed to develop vehicle/arm coordination and compliant motion tasks to address kinematics redundancy.

2.5 Robot Operating System (ROS)

The Robot Operating System (ROS) [41] is an open-source software framework for robotic systems and considered as generic middleware for developing robotic applications. It provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [42]. ROS supports multilingual programming platforms such as Python, C++, among other.
The main objectives of ROS are to increase the reuse of the code and to support as many different robots as possible by setting a universal standard in robotics research and development and it is largely supported by the robotics (commercial and academic) community [43]. It includes a variety of tools, libraries and software packages for robot description, pose estimation, communication, motion planning, grasping, navigation, simulation, perception, mapping, localization, etc [44].

2.5.1 ROS Communication

A complex robotic system with a mobile base, robotics arm and gripper, etc. requires several calculations and execution of many different programs simultaneously. ROS solves this problem by allowing all the functions of the robot to be divided into a number of small pieces that communicate with each other via messages. These small pieces are called nodes and they function as a separate process to perform the overall computation of the robotic

system concurrently [45]. For instance, a node is responsible to perform given tasks, including hardware initialization and control, motion planning, localization, processing sensor data and visualization of the system, etc. [46]
The ROS Master, which is the core of the ROS system, manages communication between nodes by providing unique name and registration to nodes. Without the Master, nodes would not be able to find each other, exchange messages/invoke services [46]. Further, nodes can communicate with each other in three different ways by publishing or subscribing to topics (e.g reading or broadcasting sensor information), query/response via services and accepting/rejecting an action [47].

2.5.2 ROS Topics

ROS topics are named buses in which nodes communicate with each other by exchanging messages [48]. ROS topics provide a one-way communication channel between publisher nodes and subscriber nodes. Nodes publish data or information that they want to share and they subscribe to information that is useful for them. A single node may publish and/or subscribe to many topics, and a single topic may have multiple publishers and subscribers at same time [48]. Logically, the topic is typed message bus where each bus has a name and anyone connected to the bus can receive and send messages as far as they are the right type [46]. Figure 2.4 shows the communication model between ROS nodes using ROS topic.

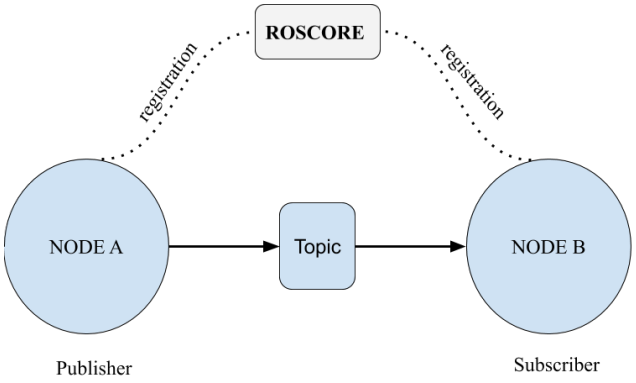


Figure 2.4: *Communication between ROS nodes using ROS topic.*

2.5.3 ROS Services

A ROS service is a query and response type of communication. A service client node sends a query message to the service server node, and the server sends back a response message to the client. ROS service provides a one-time communication between server and client, so once a request is sent to the server, the client must wait until the server responds [46]. ROS services are suitable for remote procedure calls e.g. for querying the state of a node or doing a quick calculation such as inverse kinematics [47]. Figure 2.5 shows service query and response communication between server and client.

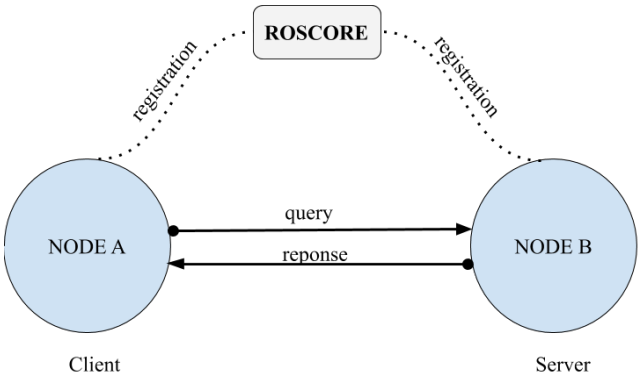


Figure 2.5: *Communication between server and client.*

2.5.4 ROS Actions

Like ROS services, ROS actions are also a type of a query and response communication between action server and action client. The action server can provide frequent feedback to monitor the progress before the requested goal has been achieved [47]. ROS *actionlib* [49] package provides tools to create action servers and action clients interface by defining three types of messages: goal, feedback and result with which they communicate with each other.
GOAL: Action client sends goals to action server.
FEEDBACK: Action server sends progress feedback to action client while performing long-term goals.
RESULT: Action server sends a result message to action client as soon as the task is completed.
Figure 2.6 depicts action query and response communication between action server and client.

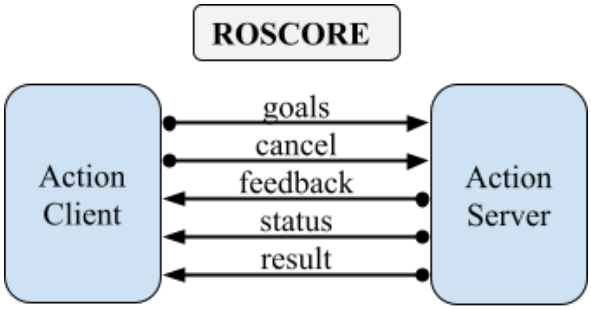


Figure 2.6: *Communication between action server and action client.*

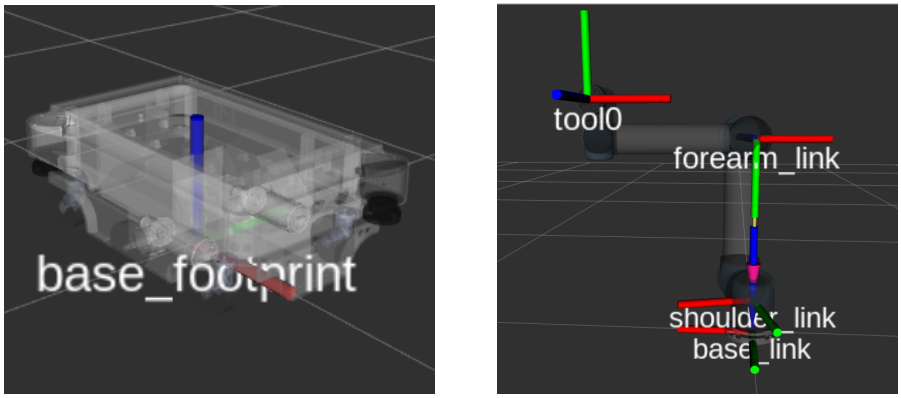
2.5.5 Robot Description Modeling

ROS describes the model of robot with Unified Robot Description Format (URDF) [50] file which is an XML template used to specify the kinematics and dynamics, robot visualization and the collision model of a robot. URDF files describes all the elements of the robot, such as number of joints, joints limits, links length, sensors, etc, furthermore, the links of a robot

can be defined by any geometric shape with the help of mesh file. To visualize the robot in a 3D world, ROS *rviz* [51] is a graphical tool that allows to render a 3D model of robots using URDF file information.

2.5.6 Coordinate System and Transforms (tf)

Coordinate frame system describes the position and orientation of a robot in an environment, coordinate system in ROS are right-handed and in 3D. According to ROS usual convention, the base frame attached to mobile robot always comes up with *X-axis* towards the front, *Y-axis* to left and *Z-axis* to upwards direction. Figure 2.7(a) shows the coordinate frames of a mobile robot that follows this convection.
ROS *tf* library provides a standard way to keep track of coordinate frames over the time and transform data within the system by attaching a unique coordinate frame to each element (link) of the robot as defined in the URDF model [52]. Furthermore, *tf* maintains the relationship between multiple coordinate frames in a tree structure and it also allows to transform points, vectors, pose, etc., between any two coordinate frames at any requested time [53].
The *tf* library has two modules, broadcaster and listener. The broadcaster designed to update data regarding coordinate frames of the robot over time to the rest of the system, and the listener collects the received data and stores the values coordinate frames that are published to the system, and may query for any particular transforms between coordinate frames on the system [52]. The *tf* provides essential information regarding robot position and orientation in the environment and helps to track exact location of the robot over time. Figure 2.7 shows an example of the *rviz* visualization of a mobile robot and a robotic arm with *tf* coordinate frames.



(a) *Coordinate frame of the mobile robot that has X-axis towards the front, Y-axis to left and Z-axis to upwards direction*

(b) *Coordinate frames of the robotic arm from base (base_link) to end-efftor(tool0)*

Figure 2.7: *Visualization of a mobile robot and a robotic arm with tf coordinate frames. The red, green and blue (RGB) cylinders represent the X-axis, Y-axis and Z-axis respectively.*

2.6 Manipulator Kinematics

The kinematics of the manipulator robot describes the mathematical relationship between the movement of connected links, position and orientation in space, without considering the cause of motion [54]. **Forward Kinematics** takes joint variables values as input and computes the end-effector position and orientation, whereas the **Inverse Kinematics** takes the pose of the end-effector and determines the joint variables values. The DH (Denavit-Hartenberg) convention [55] is a common way to define reference frames for robotics applications, and express the relationship between links through homogeneous transformation matrix that can be used to solve forward and inverse kinematics. Figure 2.8 shows the coordinate frames of a 6-revolute joints manipulator robot based on DH conventions.

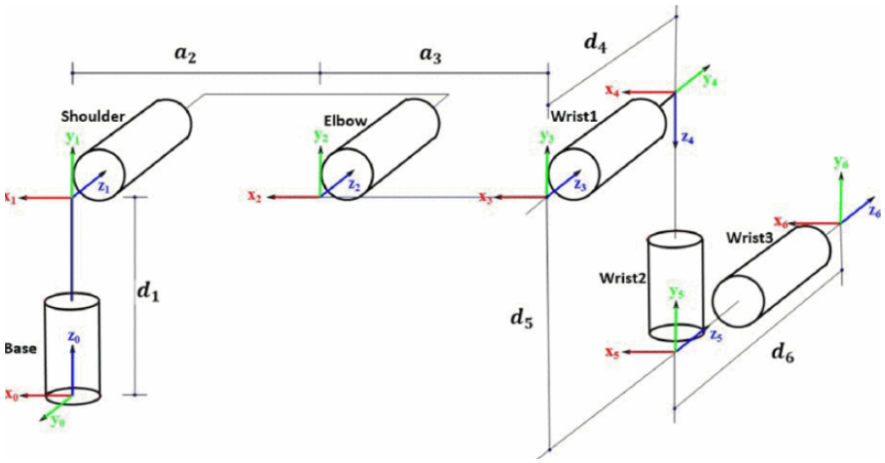


Figure 2.8: *Coordinate Frames of a 6-revolute manipulator robot based on DH conventions [56]*

A transformation from frame *i-1* to frame i can be computed using homogeneous transformation matrix, denoted as T_i^{i-1}

$$T_i^{i-1} = \begin{bmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & -cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

Where:
R represents the rotation matrix (3x3) and T represents a (3x1) translation vector. α_i, a_i, θ_i and d_i are DH parameters, which are generally named as link twist, link length, joint angle and joint offset respectively. To calculate the forward kinematics of a manipulator robot, transformation matrices from base frame to end-effector frame are multiplied.

$$T_n^0 = T_1^0 T_2^1 T_3^2 .... T_n^{n-1}$$

2.7 Mobile Robot Kinematics

Contrary to manipulators, mobile robots can move freely with respect to their environment. The kinematics of mobile robot describe the behavior of the mobile base where each individual wheel contributes to the robot's motion and, at the same time, the kinematics of the wheeled mobile platform is subjected to velocity constraint. The model contemplates how the wheels are tied together based on robot chassis geometry [57]. The Figure 2.9 represents the kinematics model of non-holonomic mobile robot. The mobile robot has the local frame $\{l\}$ which is located in the middle of the base, (x_l, y_l) coordinates represents the position of the mobile robot and ϕ is the heading angle with respect to the global frame $\{g\}$. To drive the mobile robot, it is necessary to define linear velocities $(\dot{x}, \dot{y})$ and angular velocity $\dot{\phi}$ [58]. The velocities are decomposed into corresponding speed for each wheel.

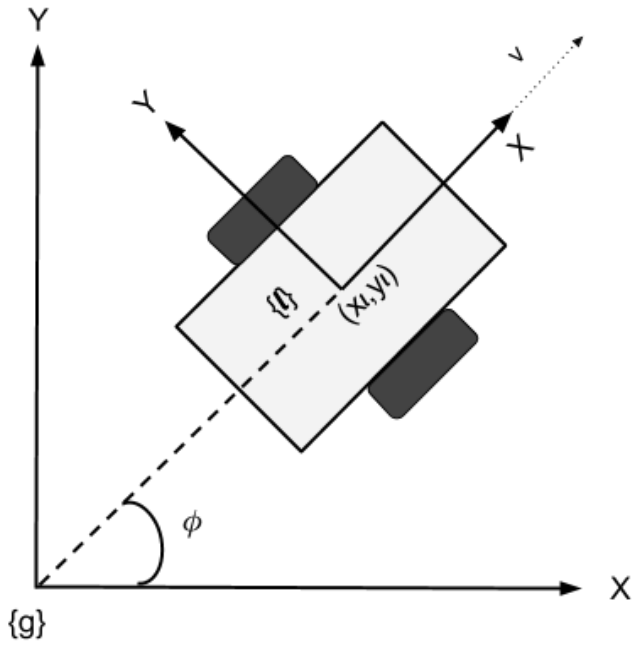


Figure 2.9: *Kinematics model of a non-holonomic mobile robot*

3 Methodology

In this chapter, a trajectory optimization approach for motion planning of a manipulator mounted on non-holonomic base (e.g a differential drive robot) is presented. The objective of the proposed method is to compute the corresponding smooth joint space motions for the given end-effector position and orientation trajectory.

3.1 End-Effector Pose Constrained Trajectory Optimization

The problem considered in this section can be formally described as follows.

> Problem Definition (**P1**): Given an end-effector pose (position + orientation) trajectory in the global frame, compute joint motions of the manipulator and the mobile base, based on user defined optimality criteria.

3.1.1 Symbols and Notations

Italic letters represent scalars and boldfaced lower case letters represent vectors while upper case variants will represent matrices. Table 3.1 summarizes the important symbols used in the thesis. Some symbols are also defined at their first place of use. The time dependence of the vectors, matrices and other variables are shown with a left subscript t and a left superscript of 0, l, g and e to denote whether a vector/matrix is defined in the manipulator base, mobile base, global or end-effector reference frame respectively. With a slight abuse of notation, the position vectors of the end-effector is also represented with subscript e. For notational simplicity and where it is obvious, the subscripts defining the reference frame is removed, e.g $\phi_t^b, \mathbf{q}$.

3.1.2 Trajectory Optimization

Considering the symbols and notations described in Table 3.1, the problem (P1) can be formulated as the following trajectory optimization.

Table 3.1: Important Symbols

$\{0\}, \{l\}, \{g\}, \{e\}$	Manipulator base, mobile base, global reference and end-effector reference frame respectively.
${}^{l}\mathbf{x}_o = ({}^{l}x_o, {}^{l}y_o, {}^{l}z_o)$	Position of the origin of $\{0\}$ with respect to the origin of $\{l\}$ expressed in the reference frame of the later.
${}^{g}\mathbf{x}_t^b = ({}^{g}x_t^b, {}^{g}y_t^b, 0)$	Position vector to origin of the mobile base frame measured and expressed in global frame
${}^{l}\mathbf{x}_t^e = ({}^{l}x_t^e, {}^{l}y_t^e, {}^{l}z_t^e)$	Position vector to end-effector measured in manipulator frame $\{0\}$ and expressed in mobile base frame $\{l\}$
${}^{g}\mathbf{x}_t^e$	Position vector to end-effector measured and expressed in global frame
ϕ_b	Heading angle of mobile base measured in global frame
$\mathbf{q}_t = (q_t^1, q_t^2, \ldots, q_t^n)$	Vector of joint angles of the manipulator.
${}^{g}\mathbf{x}_t^d = ({}^{g}x_t^d, {}^{g}y_t^d, {}^{g}z_t^d)$	Desired end-effector position measured and expressed in global frame
${}^{g}\mathbf{k}_t^e, {}^{g}\boldsymbol{\theta}_t^e = ({}^{g}x_t^d, {}^{g}y_t^d, {}^{g}z_t^d)$	Axis and angle respectively for representing end-effector orientation in global frame

$$\arg \min_{\mathbf{q}, x_b, y_b, \phi_b} w_1 \sum_t \overbrace{\|\ddot{\mathbf{q}}_t\|_2^2}^{\textbf{smoothness cost}} + w_2 \sum_t \overbrace{\|\dot{\mathbf{x}}_t^b\|_2^2 + \|\dot{\mathbf{y}}_t^b\|_2^2}^{\textbf{minimize base motion}}$$
$$+ w_3 \Big(\sum_t \overbrace{f_{pos}(\mathbf{q}_t, {}^{g}\mathbf{x}_t^b, \phi_t^b)}^{\text{position cost}} + \sum_t \overbrace{f_{orient}(\mathbf{q}_t, \phi_t^b)}^{\text{orientation cost}} + \overbrace{\mathbf{f}_m(\mathbf{q}_{t_f}) + \mathbf{f}_b({}^{g}x_{t_f}, {}^{g}y_{t_f}, \phi_{t_f}^b)}^{\text{terminal cost}} \Big) \tag{3.1}$$

$$\mathbf{q}_{min} \leq \mathbf{q}_t \leq \mathbf{q}_{max} \tag{3.2}$$

$${}^{g}\dot{x}_t^b \sin\phi_t^b - {}^{g}\dot{y}_t^b \cos\phi_t^b = 0 \tag{3.3}$$

The cost function in (3.1) has several components. The first term ensures smoothness in the manipulator joint trajectory $\mathbf{q}_t$ by minimizing the norm of the accelerations at each time instant. Such notion of trajectory smoothness in terms of the norm of the acceleration vector has been extensively used in the robotics literature. For example in [33], [59] and the references therein.

The second term in (3.1) minimizes the base motion by minimizing the norm of the linear velocity of the base at each time instant. This term is motivated by the intuition that mobile base are generally heavy and thus its motion requires more energy than the manipulator. Therefore, it is beneficial to utilize the manipulator motion more than the base motion to track the given end-effector pose trajectory.

The third term in (3.1) is the position cost. Its role is to ensure that the end-effector ${}^{g}\mathbf{x}_t^e$ position in the global frame coincides with the desired trajectory ${}^{g}\mathbf{x}_t^d$ at each time instant. Note that the desired trajectory is assumed to be given. The closer ${}^{g}\mathbf{x}_t^e$ is to ${}^{g}\mathbf{x}_t^d$, the smaller the value of f_{pos}. It is also worth pointing out that f_{pos} represents a cost, it is always non-negative, i.e its smallest value can be zero and it happens when ${}^{g}\mathbf{x}_t^e$ is to ${}^{g}\mathbf{x}_t^d$ are exactly the same. Later in this section, the exact algebraic form for f_{pos} is presented.

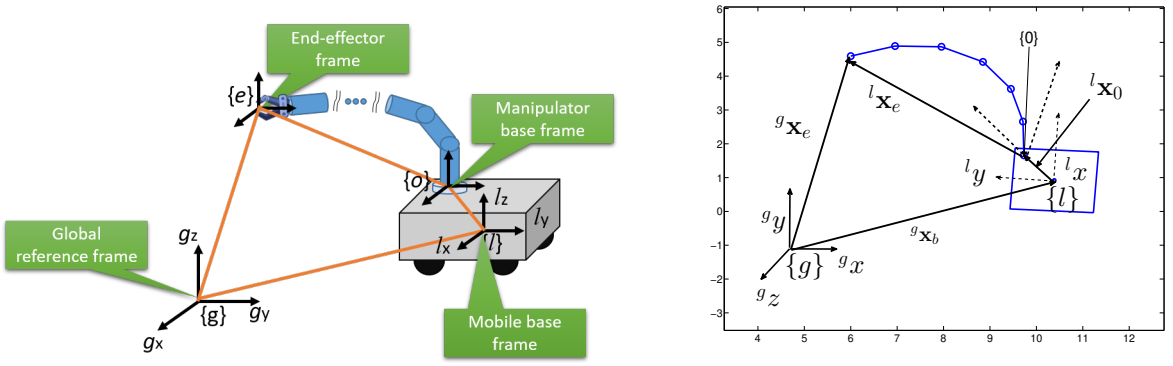
The fourth term in (3.1) is the orientation cost which ensures that the end-effector has the specified orientation at each time instant. The form of f_{orient} is similar to f_{pos} and is derived in the next section. The final two terms in (3.1) models the terminal cost, i.e, the cost on the final manipulator joint values (f_m) and, position and orientation of the mobile base (f_b). The weights w_i allows to achieve trade-off between different cost components. For example, by choosing w_3 higher than w_1, w_2 the position, orientation and terminal costs can be reduced by minimizing smoothness.
There are two sets of constraints in the formulated trajectory optimization. The constraints (3.2) are the bounds on the manipulator joint values. The equality constraints in (3.3) are called the non-holonomic constraints. These are specific to mobile robots with differential drive mechanism. It states that the mobile base motion in the x and y are coupled with each other through the heading angle. To put it more simply, the mobile base can only move forward or backward in the direction of the instantaneous heading of the mobile base. In the following subsections, the algebraic form for the different components of the cost function is derived.

3.1.3 Algebraic Form for the Costs

Position Cost f_{pos}: For deriving the position cost f_{pos}, the forward kinematics of the combined system of the mobile base and the manipulator need to be derived firstly. From Figure 3.1, the end-effector position in the global frame ${}^{g}\mathbf{x}_t^e$ can be represented in the following manner [7].

$$ {}^{g}\mathbf{x}_t^e = {}^{g}\mathbf{x}_t^b + {}^{g}_{l}\mathbf{R}(\phi_t^b)({}^{l}\mathbf{x}_0 + \overbrace{{}^{l}_{0}\mathbf{R}\,{}^{0}\mathbf{x}_t^e}^{{}^{l}\mathbf{x}_t^e}) \tag{3.4} $$

Where, the rotation matrix ${}^{l}_{0}\mathbf{R}$ depends on how the manipulator is connected to the mobile base and is constant. The term ${}^{g}_{l}\mathbf{R}$ represents the rotation matrix between the local mobile base and the global frame and is a function of ϕ_t^b. The vector ${}^{0}\mathbf{x}_e$ represents the end-effector position in the manipulator local reference frame.



(a) (b)

Figure 3.1: Relevant vectors for computing the end-effector position in the global frame

Thus, the position cost takes the form

$$f_{pos}(\mathbf{q}_t, {}^g\mathbf{x}_t^b, \phi_t^b) = \|{}^g\mathbf{x}_t^b + {}_l^g\mathbf{R}(\phi_t^b)({}^l\mathbf{x}_0 + \overbrace{{}_0^l\mathbf{R}\,{}^0\mathbf{x}_t^e}^{{}^l\mathbf{x}_e}) - {}^g\mathbf{x}_t^d\|_2^2. \tag{3.5}$$

The desired trajectory ${}^g\mathbf{x}_t^d$ has to be given. Also, note that the ${}^0\mathbf{x}_t^e$ is obtained by the forward kinematics of only the manipulator and is a function of the joint angles $\mathbf{q}_t$.

Orientation Cost: The final rotation matrix describing the orientation of the end-effector in the global reference frame is given by

$${}^g\mathbf{R}_e = {}_l^g\mathbf{R}(\phi_t^b){}_0^l\mathbf{R}\,{}^0\mathbf{R}_e(\mathbf{q}_t) \tag{3.6}$$

From the final rotation matrix, the axis-angle representation of the end-effector can be computed. The axis vector ${}^g\mathbf{k}_t^e$ can be extracted from ${}^g\mathbf{R}_e$ in the following manner.

$${}^g\mathbf{k}_t^e = \begin{bmatrix} {}^g\mathbf{R}_e(3,2) - {}^g\mathbf{R}_e(2,3) \\ {}^g\mathbf{R}_e(1,3) - {}^g\mathbf{R}_e(3,1) \\ {}^g\mathbf{R}_e(2,1) - {}^g\mathbf{R}_e(1,2) \end{bmatrix} \tag{3.7}$$

where, ${}^g\mathbf{R}_e(i,j)$ represents the (i,j) element of the matrix. The right hand side of (3.7) can be substituted in (3.1) to get a first half of the orientation cost. Please note that the axis is a function of $\mathbf{q}_t, \phi_t^b$. The angle associated with the axis ${}^g\mathbf{k}_t^e$ is given by the following

$$\cos({}^g\theta_t^e) = \frac{Tr({}^g\mathbf{R}_e) - 1}{2} \tag{3.8}$$

where $Tr(.)$ represents the trace operator. For computing the orientation cost, assuming that the desired end-effector orientation is also described in terms of some desired axis ${}^g\mathbf{k}_t^d$ and desired angle ${}^g\theta_t^d$. Thus, the orientation cost can be formulated in the following form

$$f_{orient}(\mathbf{q}_t, \phi_t^b) = \|{}^g\mathbf{k}_t^e - {}^g\mathbf{k}_t^d\|_2^2 + \|\cos({}^g\theta_t^e) - \cos({}^g\theta_t^d)\|_2^2 \tag{3.9}$$

Terminal Cost: The terminal cost essentially ensures that the values of $\mathbf{q}_t, x_t^b, y_t^b, \phi_t^b$ at final time instant are close to some specified values.

$$f_m(\mathbf{q}_t) = \left\| \begin{bmatrix} \mathbf{q}_{t_f} - \mathbf{q}_f \\ \dot{\mathbf{q}}_{t_f} - \dot{\mathbf{q}}_f \\ \ddot{\mathbf{q}}_{t_f} - \ddot{\mathbf{q}}_f \end{bmatrix} \right\|_2^2 \tag{3.10}$$

where $(\mathbf{q}_f, \dot{\mathbf{q}}_f, \ddot{\mathbf{q}}_f)$ are user specified values. One can write similar expressions for x_t^b, y_t^b, ϕ_t^b to obtain $f_b(.)$ in (3.1) as shown below.

$$f_b(x_t^b, y_t^b, \phi_t^b) = \left\| \begin{bmatrix} x_{t_f}^b - x_f \\ y_{t_f}^b - y_f \\ \dot{x}_{t_f}^b - \dot{x}_f \\ \dot{y}_{t_f}^b - \dot{y}_f \\ \phi_{t_f}^b - \phi_f \\ \dot{\phi}_{t_f}^b - \dot{\phi}_f \end{bmatrix} \right\|_2^2, \tag{3.11}$$

where $(x_f, y_f, \dot{x}_f, \dot{y}_f, \phi_f, \dot{\phi}_f)$ are user specified boundary values.

Special Cyclical Condition: Suppose, the end effector trajectory is cyclic and closed, e.g an ellipse or infinity symbol. Then, it is beneficial to have the trajectories of $\mathbf{q}_t, x_t^b, y_t^b, \phi_t^b$ also cyclic and closed. That is, when the mobile manipulator has come back to its start position after completing the end-effector trajectory, its $\mathbf{q}_t, x_t^b, y_t^b, \phi_t^b$ values should be the same that it started with. This can be ensured by modifying the terminal cost in the following way for the specific case of cyclic and closed end effector trajectory.

$$f_m(\mathbf{q}_t) = \left\| \begin{bmatrix} \mathbf{q}_{t_0} - \mathbf{q}_{t_f} \\ \dot{\mathbf{q}}_{t_0} - \dot{\mathbf{q}}_{t_f} \\ \ddot{\mathbf{q}}_{t_0} - \ddot{\mathbf{q}}_{t_f} \end{bmatrix} \right\|_2^2 \tag{3.12}$$

$$f_b(x_t^b, y_t^b, \phi_t^b) = \left\| \begin{bmatrix} x_{t_0}^b - x_{t_f}^b \\ y_{t_0}^b - y_{t_f}^b \\ \dot{x}_{t_0}^b - \dot{x}_{t_f}^b \\ \dot{y}_{t_0}^b - \dot{y}_{t_f}^b \\ \phi_{t_0}^b - \phi_{t_f}^b \\ \dot{\phi}_{t_0}^b - \dot{\phi}_{t_f}^b \end{bmatrix} \right\|_2^2 \tag{3.13}$$

As shown above, the terminal cost for the cyclicity condition is just a penalty on the difference between the initial and final values of the joint angles of the manipulators, position and orientation of the mobile base.

3.2 Parametrization and Solution Process

To simplify the solution process of optimization (3.1)-(3.3), the variables can be parameterized through smooth polynomials. This reduces the dimensionality of the problem and makes it equal to the number of polynomial coefficients. This also ensures higher order continuity and differentiability in the trajectories.
The parametrization is given by the following

$$\mathbf{q}_t = \mathbf{P}\mathbf{c}_q, {}^g x_t^b = \mathbf{P}\mathbf{c}_x, {}^g y_t^b = \mathbf{P}\mathbf{c}_y, {}^g \phi_t^b = \mathbf{P}\mathbf{c}_\phi \tag{3.14}$$

where $\mathbf{P}$ is a matrix of time dependent polynomial basis functions and $\mathbf{c}_q, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi$ are the coefficients associated with the basis functions. Denoting $\boldsymbol{\xi} = (\mathbf{c}_q, \mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_\phi)$, the trajectory optimization (3.1)-(3.3) can be represented in the following compact form

$$\arg\min_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) \tag{3.15}$$

$$\mathbf{A}\boldsymbol{\xi} \leq \mathbf{b} \tag{3.16}$$

$$\mathbf{g}(\boldsymbol{\xi}) = \mathbf{0} \tag{3.17}$$

Where, $\mathbf{A}$ is a constant matrix formed by diagonally stacking the different P matrices. The vector $\mathbf{b}$ contains the joint limits. The vector valued function $\mathbf{g}$ is obtained by rewriting the the non-holonomic constraint (3.3) at different time instants using the polynomial coefficients. The optimization (3.15)-(3.17) is obtained by substituting (3.14) into (3.1)-(3.3). This form is particularly suitable for describing the solution process next.

3.2.1 Reformulating as an Unconstrained Optimization Problem

Optimization (3.15)-(3.17) represents a difficult constrained non-linear programming problem. One way to simplify its solution process is to reformulate it as the following unconstrained optimization by reformulating the constraints as costs.

$$\min f(\boldsymbol{\xi}) + w_4 \sum \log(1 + \exp(\mathbf{A}\boldsymbol{\xi} - \mathbf{b})^2 + w_5 \|\mathbf{g}(\boldsymbol{\xi})\|_2^2 \tag{3.18}$$

The second term is a cost stemming from the violation of the inequality constraints representing the joint limit. That is, if $\mathbf{A}\boldsymbol{\xi} > \mathbf{b}$, then the second term would be higher and vice versa. This essentially pushes the solution towards region that satisfy the manipulator joint limits. This specific term involving logarithm is a smooth approximation of so called Relu activation function used in Neural Networks. To be precise, any inequality of the form $h \leq 0$ can be reformulated as a cost $\max(0, h)$. However, the max term is non-smooth as shown in Figure 3.2. The cost with logarithm is a smooth approximation of the $\max(0, h)$.
The third term relaxes the equality constraints to a quadratic penalty. It is clear that the residual of $\|\mathbf{g}(\boldsymbol{\xi})\|_2^2$ tends to zero, it would be approaching the required non-holonomic behavior as dictated by the constraints (3.17). The weights $w_4.w_5$ serve the same purpose as the weights w_1, w_2, w_3 in (3.1). They allow to control the residual of cost term they are associated with.
There are many techniques like Gradient Descent, sequential quadratic programming, regularized Gauss Newton, that can be employed to solve (3.18). the implementation of these techniques are relied on the open-source Python Packages like Scipy and Sympy to solve (3.18).

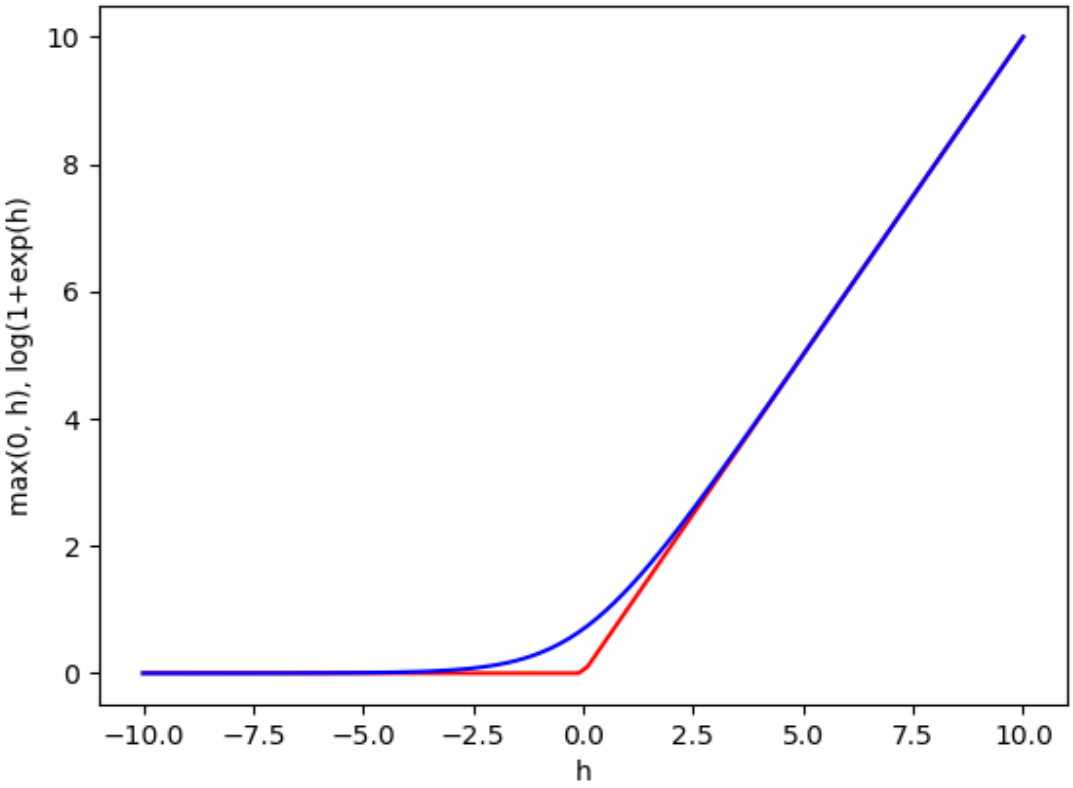


Figure 3.2: For a given function h, the red line shows the plot of $\max(0, f)$ and its smooth approximation given by $\log(1 + \exp(h)$ is shown in blue.

4 Implementation

In this chapter, the implementation of the proposed trajectory optimization approach for the motion planning of a mobile manipulator is demonstrated on both simulation environment and with a real robot by following three different trajectories: elliptical, infinity symbol and triangular.

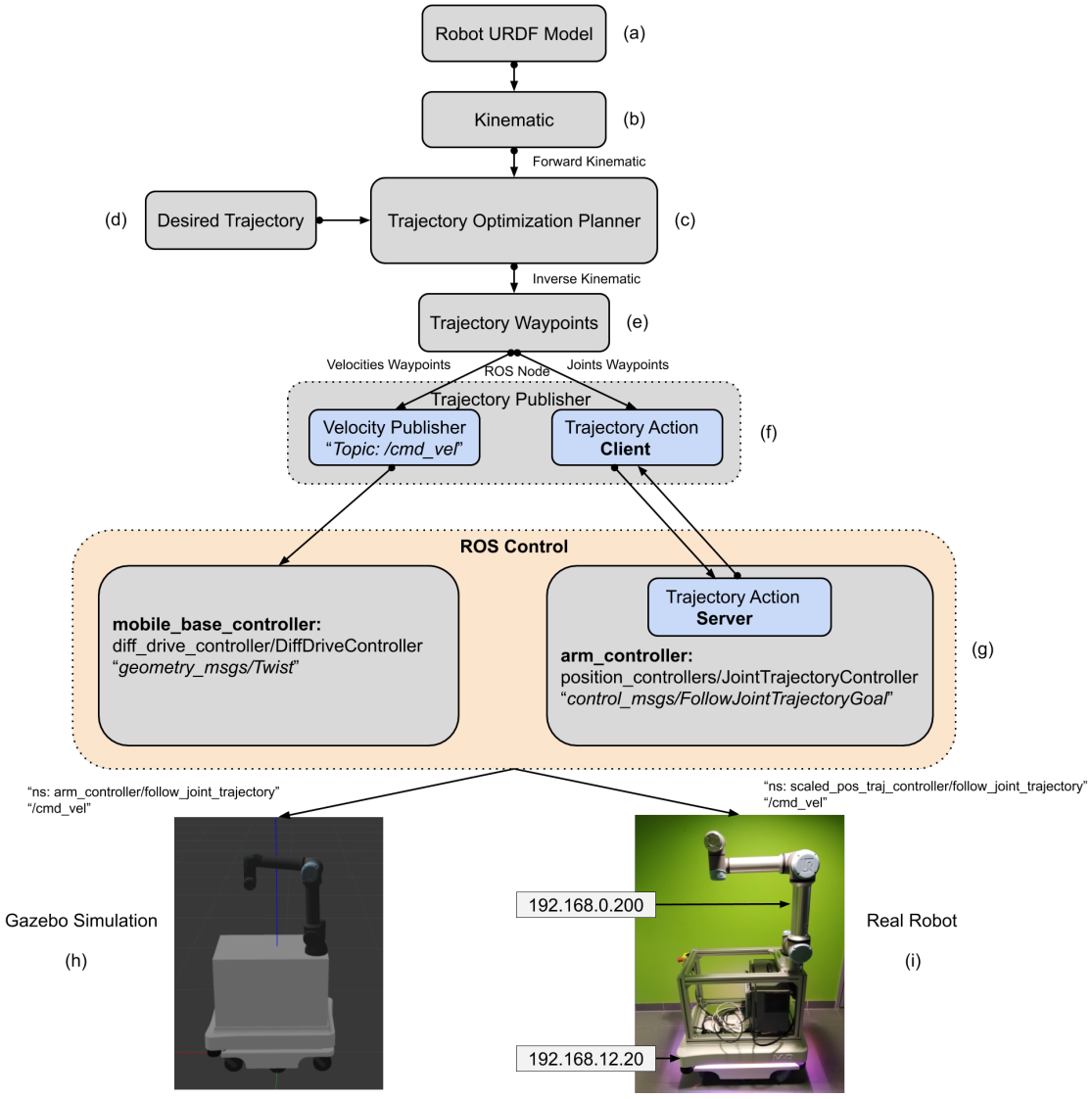


Figure 4.1: *Schematic of motion planning of a mobile manipulator using ROS interface*

Figure 4.1 shows a diagram with the steps about how a mobile manipulator can follow a given trajectory, starting from the description and kinematics of the robot, and by using the propose trajectory optimization algorithms, compute a set of trajectory waypoints at each instance of time, and at the end, send the commands to mobile base and manipulator simultaneously through ROS communication protocol. This implementation can be accessed through this repository [60].

4.1 Mobile Manipulator

The robot used to test and demonstrate the proposed methodology correspond to the following main components,

4.1.1 Mobile platform: MiR100

MiR100 (Figure 4.2) is a non-holonomic mobile platform design and developed by Mobile Industrial Robots [61]. It has the ability to transport up to 100 kg payload with maximum speed of 1.5 m/s in forward-direction and 1.3 m/s in backward-direction. To control and integrate MiR100 with ROS, a ROS community project [62] created by DFKI (the German Research Center for Artificial Intelligence) which provides ROS driver and configuration files.

Figure 4.2: *MiR100 Mobile Platform [61]*

4.1.2 Manipulator: UR5e

UR5e (Figure 4.3) is an adaptable collaborative robotic arm developed by the Universal Robots [63]. It can reach up to 0.85 m with respect to base and lift up to 5 kg payload. The UR5e has six revolute joints with 640 degree rotational limit. It has built-in sensors that detects the external force exerted on arm and immediately activate the emergency stop to protect robot. This feature ensures the safety of the robot while performing repetitive and dangerous tasks. To control and run the UR5e autonomously, ROS-Industrial *universal_robot* meta-package [64] provides drivers and other necessary files to interface with the UR5e.

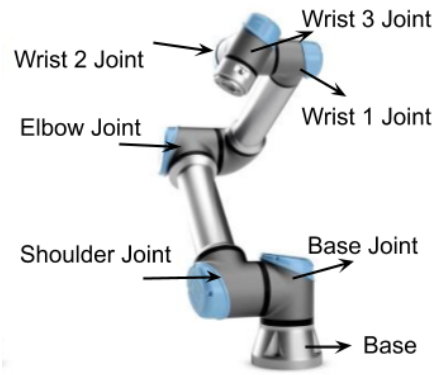


Figure 4.3: *UR5e Robot [63]*

4.2 Kinematics of Mobile Manipulator

The URDF files of both robots were combined in a single file where the UR5e manipulator is attached on top of the MiR100 mobile platform, according to the dimensions of the real robot (Figure 4.1(a)). This integration helps to represent the complete kinematics of the mobile manipulator (Figure 4.1(b)). Figure 4.4 depicts the general relationship between the coordinate frame references of the robot. The global reference frame $\{g\}$ shows the position and orientation of the mobile manipulator robot in Cartesian plane, reference frame $\{l\}$ represents the MiR100 mobile platform and reference frame $\{0\}$ indicates the UR5e manipulator base origin. DH representation are used to calculate the forward kinematics of the manipulator (as described in chapter 2.6). The DH parameters and joint limits of UR5e manipulator are given in the Table 4.1 taken from [65].

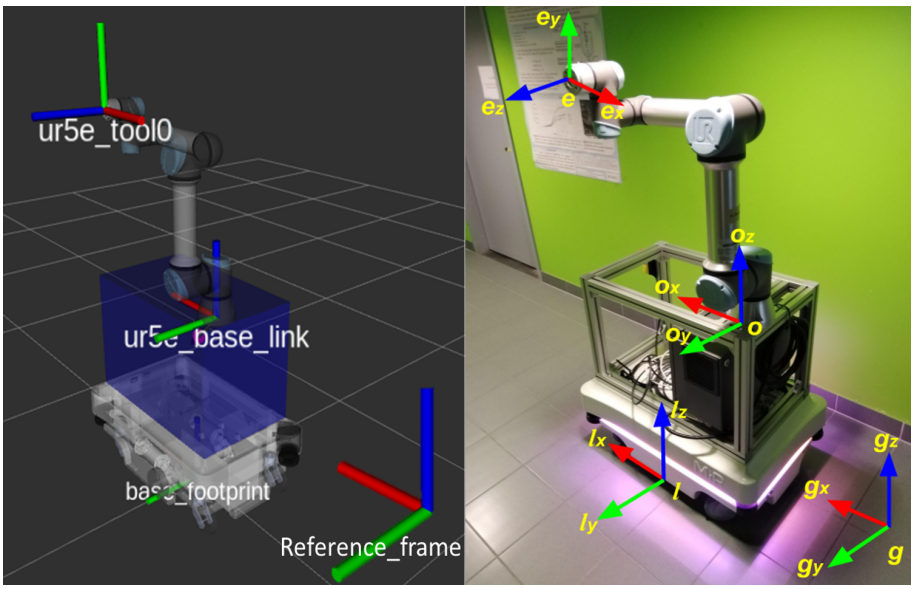


Figure 4.4: *Reference coordinate frames for the mobile manipualtor with URDF visualization in Rviz (left) and the actual hardware (right)*

Table 4.1: DH parameters of UR5e manipulator [65]

Joint	a(m)	d(m)	α(rad)	θ(rad)	θ_{min}(deg)	θ_{max}(deg)
Joint1	0	0.1625	$\pi/2$	q_1	-180	180
Joint2	-0.435	0	0	q_2	-180	180
Joint3	-0.3922	0	0	q_3	-180	180
Joint4	0	0.1333	$\pi/2$	q_4	-180	180
Joint5	0	0.0997	-$\pi/2$	q_5	-180	180
Joint6	0	0.0996	0	q_6	-180	180

Table 4.2 describes the relationship between the position of the manipulator base $\{0\}$ frame with respect to the mobile base $\{l\}$ frame.

Table 4.2: Transformation between the manipulator base frame to the mobile base frame.

${}^{l}\mathbf{x}_0$	Values (m)
${}^{l}x_o$	-0.26235
${}^{l}y_o$	0.1
${}^{l}z_o$	0.842

4.3 Trajectory Optimization Planner

To implement the proposed trajectory optimization approach (Figure 4.1(c)), as discussed in the chapter 3, the SciPy python libraries are used to solve the mathematical expression (3.18) and optimization problem. Figure 4.5 depicts different examples of the desired trajectories with cyclic and closed path, and are considered as input (Figure 4.1(d)) for the trajectory optimization planner which corresponds to the point in space that the end-effector has to follow.

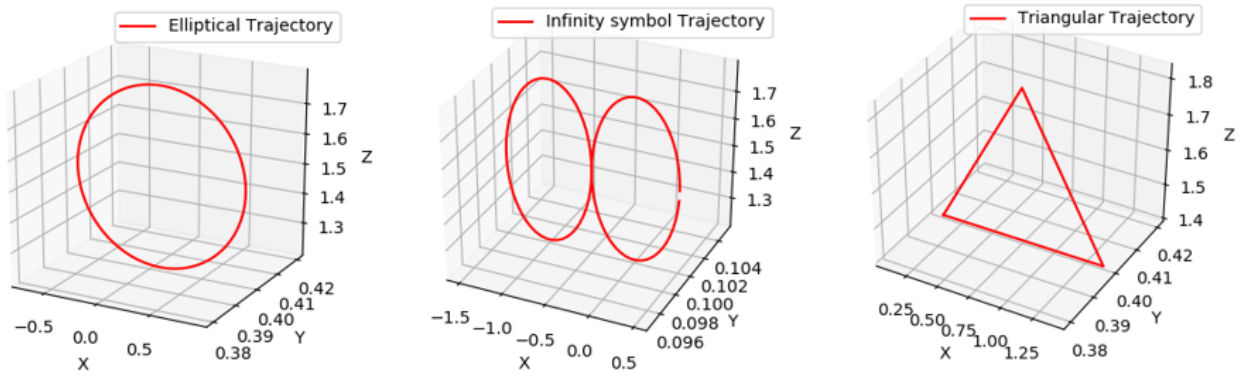


Figure 4.5: *Desired Trajectory*

The trajectory optimization planner generates a sequence of consecutive trajectory waypoints (Figure 4.1(e)) that contain the joints motion ($\mathbf{q}_t$) of the manipulator and the motion $({}^{g}\dot{x}_t^b, {}^{g}\dot{y}_t^b, {}^{g}\dot{\phi}_t^b)$ of the mobile base. In order to follow the given trajectory, 100 trajectory waypoints are generated, which represent the configuration of the robot at each instance of time and are stored in a data file.

4.4 ROS Interface

The trajectory publisher node (Figure 4.1f) is in charge of reading trajectory waypoints and distribute the commands to the corresponding joints of the manipulator and the velocities to mobile base. Each robot has a set of ROS packages among there are ROS controllers that take the commands (Figure 4.1(g)). In order to communicate with the manipulator the controller provides the action server (discussed in chapter 2.5.4), and the trajectory publisher node (Figure 4.1(f)) is an action client able to send trajectory waypoints as goals by using *FollowJointTrajectoryAction*. For the manipulator, the trajectory waypoints are in the form of positions, therefore the controller used is *position_ controllers/JointTrajectoryController*. In the case for the mobile base, the controller subscribes to command velocity topic */cmd_ vel* which has the message type *geometry_ msgs/Twist*. For differential drive wheel robot, ROS provides *diff_ drive_ controller*.

4.5 Simulation

Simulation of robots is essential in the robotic research for rapid and efficient testing of new ideas, concepts and algorithms. The robotics simulation environment can be used to test robot applications without depending on real robot hardware which can save time and resources.[66]. ROS is integrated with Gazebo simulator for robot modeling and simulation. Gazebo [67] is designed to create 3D dynamic multi-robot environments that a robot may encounter in a real world. It has ability to simulate robots, objects and sensors accurately in indoor and outdoor environments. In this thesis, Gazebo simulation environment has been used to test and visualize the motion planning for the mobile manipulator. By spawning the URDF model of the robot in the Gazebo environment, and running the ROS controllers, it is possible to send the base velocities and joint motion to the mobile manipulator to visualize how the end-effector follows the desired trajectory.
The ROS action server on Gazebo uses the namespace "*ns:arm_ controller/follow_joint_ trajectory*" to communicate with the action client which sends the manipulator joint motions, and the mobile base subscribes to the topic */cmd_ vel* to receive base velocities. Figures 4.6, 4.7 and 4.8 show the simulations of the robot following three different trajectories: elliptical, infinity symbol and triangular at different time intervals.

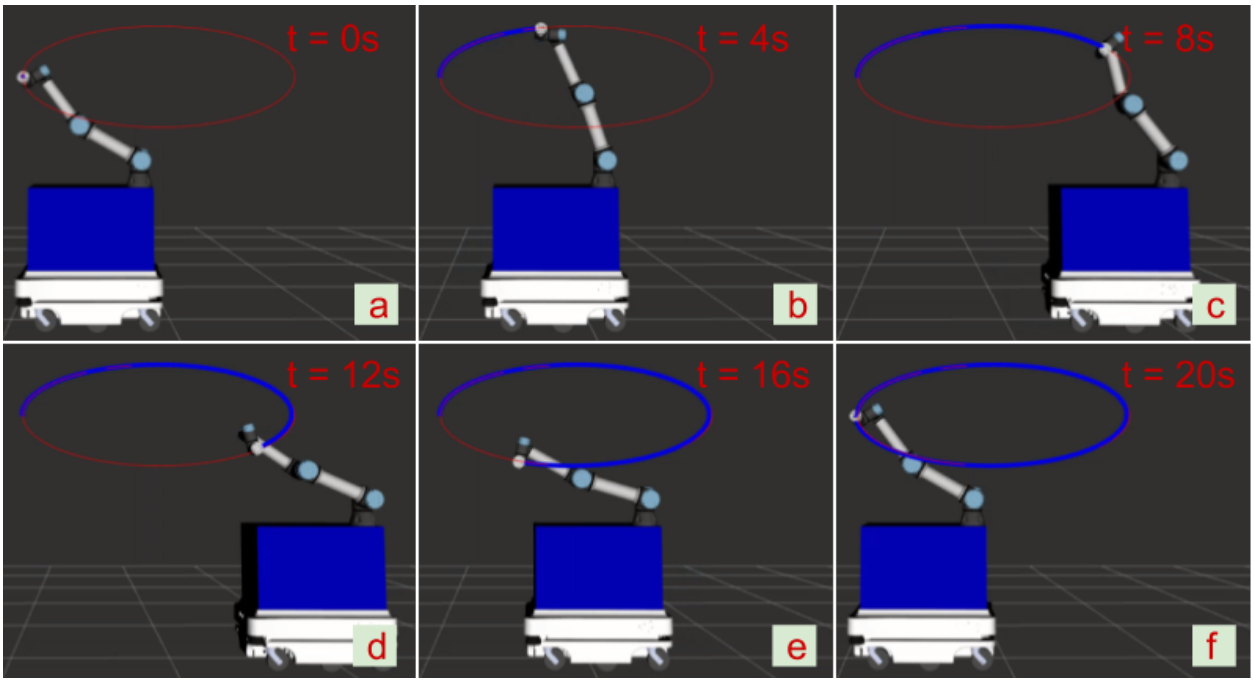


Figure 4.6: *Simulation of the mobile manipulator following the elliptical trajectory at different time intervals. The red marker shows the desired end-effector trajectory and the blue marker shows the traced trajectory.*

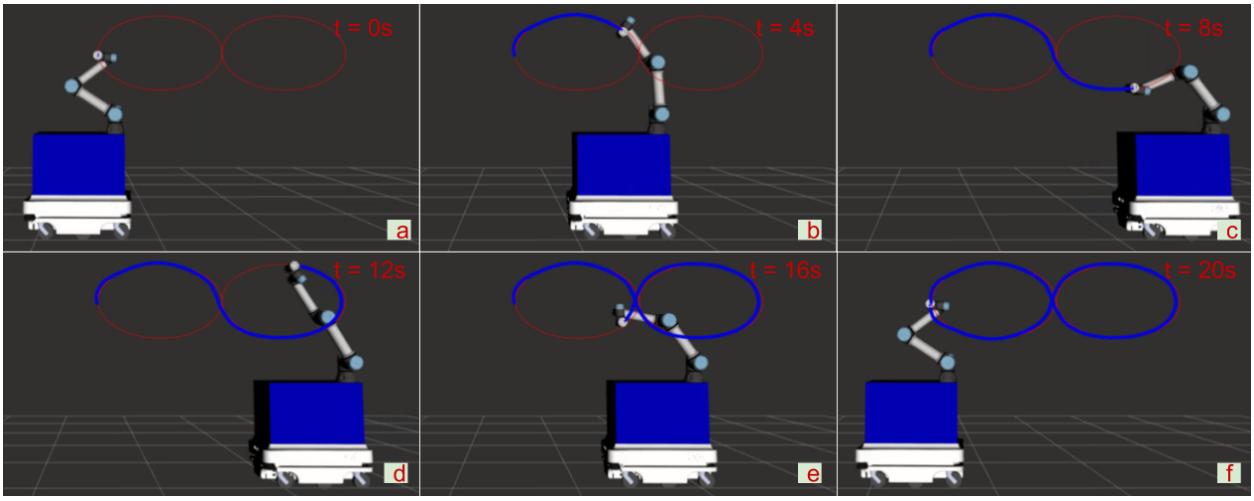


Figure 4.7: *Simulation of the mobile manipulator following the infinity symbol trajectory at different time intervals. The red marker shows the desired end-effector trajectory and the blue marker shows the traced trajectory.*

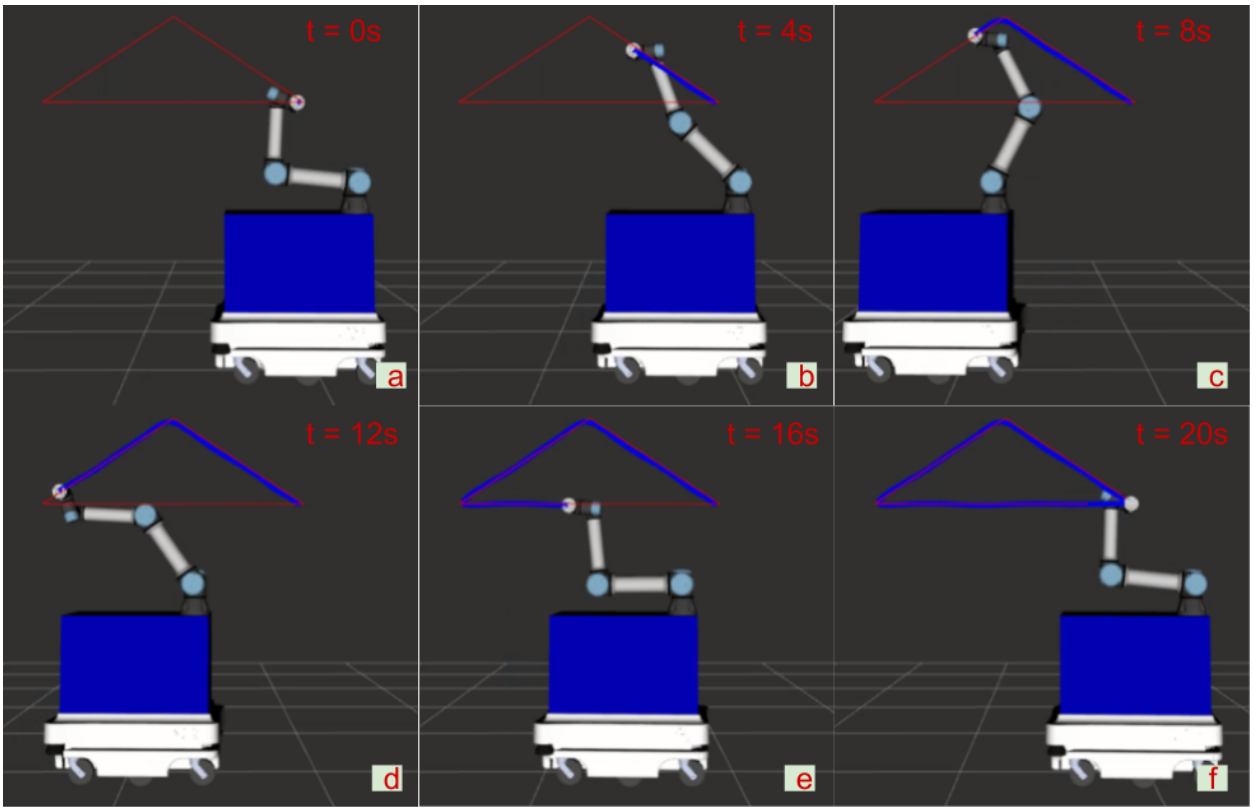


Figure 4.8: *Simulation of the mobile manipulator following the triangular trajectory at different time intervals. The red marker shows the desired end-effector trajectory and the blue marker shows the traced trajectory.*

4.6 Real Robot Hardware

The implementation of the proposed optimization method on real robot hardware is achieved by sending the linear and angular velocities to the mobile base and joints motion to manipulator simultaneously. In order to communicate with the real robot, it is necessary to run the drivers on each robot hardware, which enable the controllers to receive commands. The arm takes the instruction as goals in the ROS action with the namespace "*ns:scaled_pos_traj_controller/follow_joint_trajectory*" and the mobile base subscribes to the topic */cmd_vel*. To track and compare if the end-effector is following the given trajectory, an *ar_marker* [68] was attached to the end-effector of mobile manipulator, and using a camera, the trajectory was recorded. Figures 4.9, 4.10 and 4.11 present the real robot following three different trajectories: elliptical, infinity symbol and triangular at different time intervals.

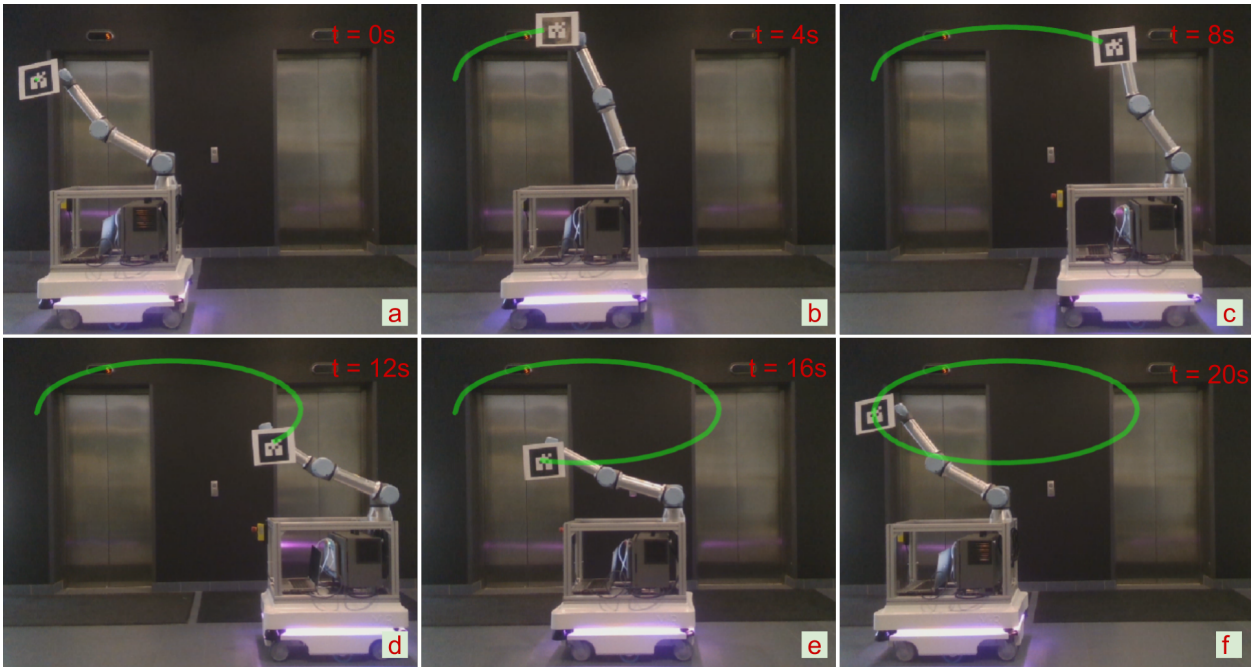


Figure 4.9: *Real robot following the elliptical trajectory at different time intervals. The green marker shows the traced trajectory.*

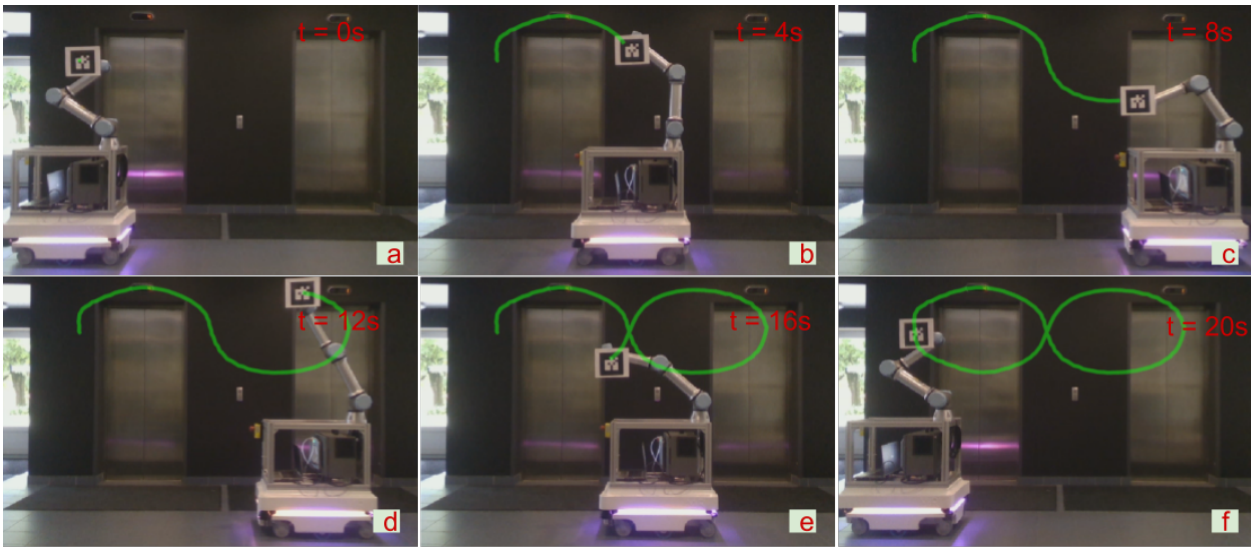


Figure 4.10: *Real robot following the infinity symbol trajectory at different time intervals. The green marker shows the traced trajectory.*

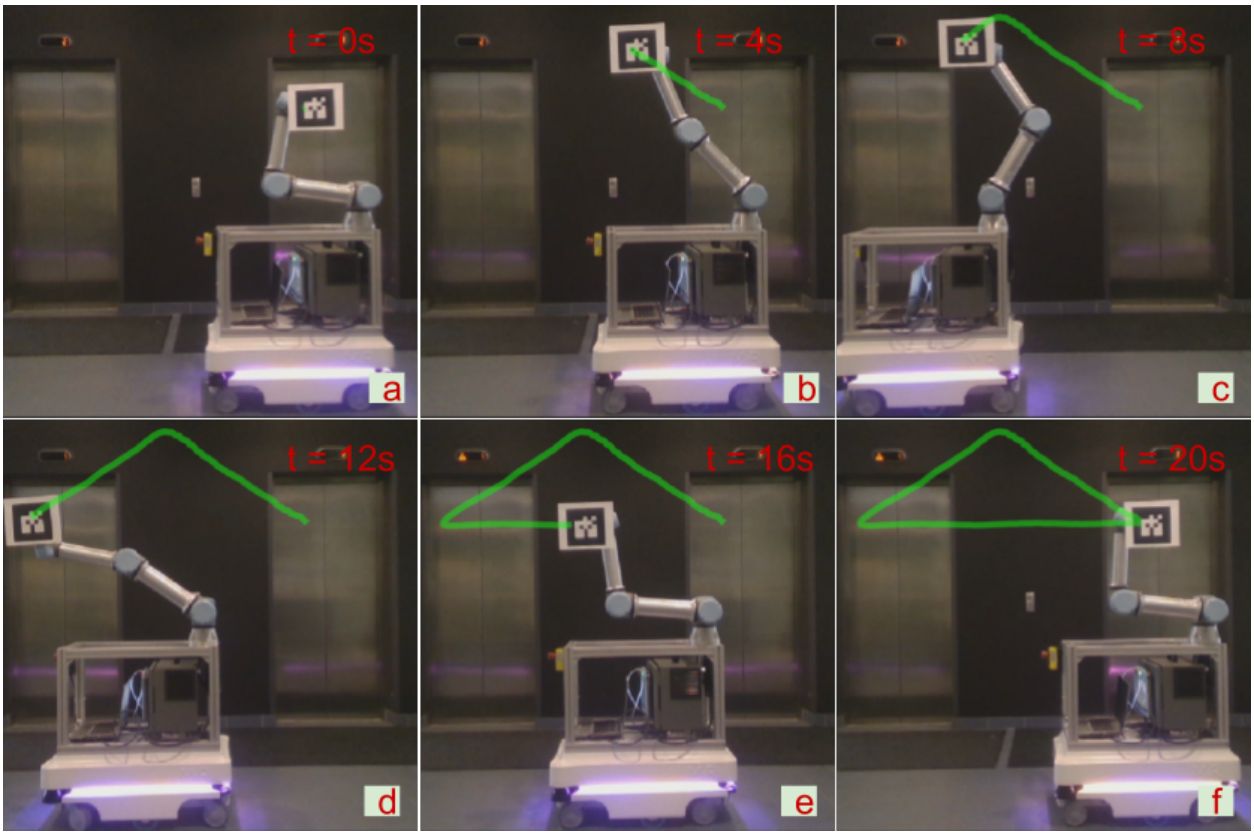


Figure 4.11: *Real robot following the triangular trajectory at different time intervals. The green marker shows the traced trajectory.*

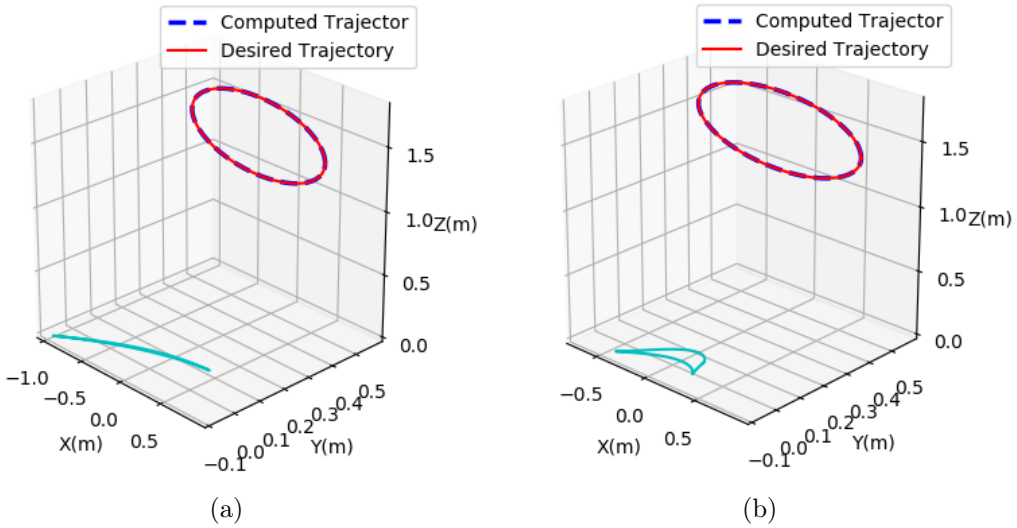
5 Results

The objective of this chapter is three-fold. First to validate the key aspects of the proposed trajectory optimizer such as (i) convergence as measured by the decrease in the cost with iterations, (ii) the ability of the optimizer to generate a diverse class of trajectories and finally (iii) the ability to produce trajectories with cyclicity condition. Second, a comparison with KDL is presented, which is an inverse kinematics library provided within ROS. Finally, the experimental results for the execution of the computed trajectories both in simulation and with real mobile manipulator hardware (consisting of MiR100 mobile base and UR5e arm) are shown to quantify the tracking error.

5.1 Weight Tuning and Convergence Validation

The weights w_1 and w_2 in the cost function (3.1) acts as an hyper-parameter and allows us to tune the behavior of the trajectories obtained with the proposed optimizer. This is a particularly important feature as it allows us to control the individual contribution of mobile base and the manipulator in tracking the given end-effector trajectory. For example, it is often required to minimize the motion of the mobile base as they are generally much heavier than the robot arm and thus require more energy for their motion.
Figure 5.1(a) and Figure 5.1(b) summarize this result wherein the trace of the mobile base trajectory is shown in cyan. As can be seen, as w_2 is increased from 20 to 200, the arc length of the trajectory significantly reduces. This reduction in mobile base motion can be directly correlated with the joint acceleration of manipulator shown in Figure 5.2. The accelerations are higher for $w_2 = 200$ indicating that the manipulator now compensates for reduction in motion of the mobile base.
Smoothness: The joint acceleration plots in Figure 5.2 also clearly shows that the joint accelerations are bounded and change gradually. This in turn is essential for faithfully executing the computed trajectories on the real hardware.
Convergence Validation: Figure 5.3(a) shows the magnitude of the reformulated cost (3.15) with iterations of the optimization solver (SciPy-SLSQP). As can be seen, the reduction in cost is almost monotonic. Furthermore, it is worth noting that at higher w_2, the solver requires more iteration. This is because a high value of w_2 conflicts with other cost terms in (3.15) such as end-effector tracking error and thus the solver needs more iteration to reduce the tracking residuals. Intuitively, this can be understood in the following manner. As w_2 increases, the base motion is limited and thus the optimizer solver needs to search more complex motions for the manipulator to maintain a low end-effector tracking residual.

Cyclicity Validation: Recall that the cyclicity condition requires that for closed and cyclic trajectories of the end-effector, the mobile based and manipulator joint trajectories also remain cyclic and closed. Intuitively, this means that when the mobile manipulator finishes one repetition of the cyclic trajectory, it reaches the same configuration and velocities that it started with. Figure 5.3(b) validates this for the end-effector trajectories shown in Figure 5.1(a) and Figure 5.1(b) by showing the norm of the difference between the start and final configurations and velocities. Low values of the norm implies that the cyclicity condition has been ensured.



(a) (b)

Figure 5.1: *Simulation of an elliptical trajectory for a manipulator mounted on a non-holonomic base for (a) w2 = 20 and (b) w2 = 200. Desired end-effector trajectory is shown in red, the actual trajectory traced in blue dashed line, and the mobile base trajectory is shown in cyan.*

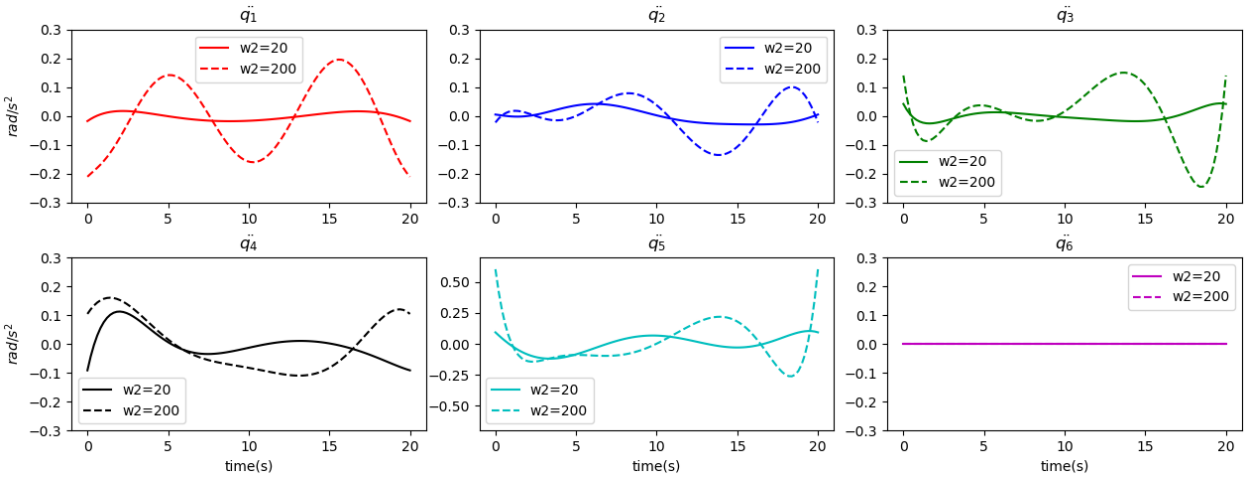
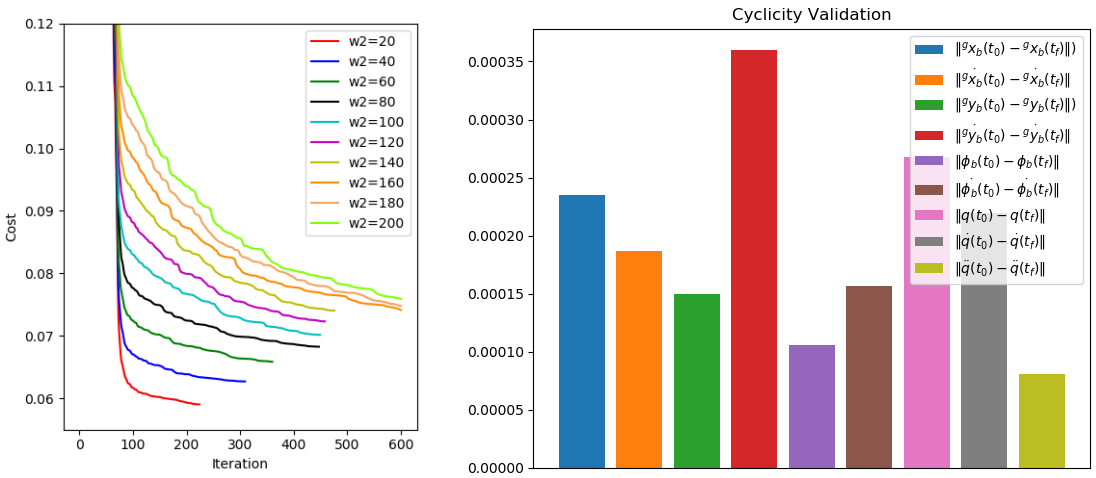


Figure 5.2: *Manipulator joint acceleration ($\ddot{q}$) for w2 = 20 (solid) and w2 = 200(dashed)*



(a) (b)

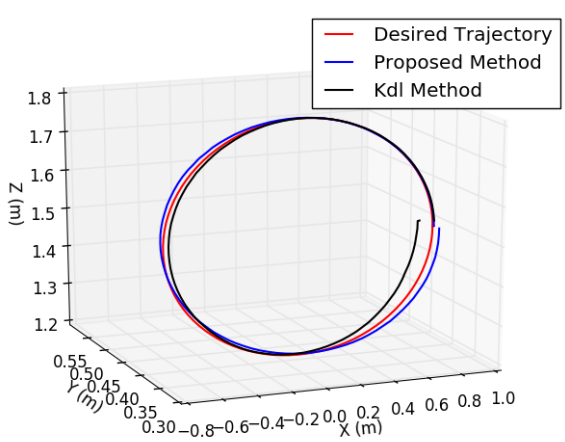
Figure 5.3: *(a) Residual cost for 10 different values of w2. (b) Cyclicity validation: Average residual between initial and final configurations, velocities, and accelerations using 10 different trajectories*

5.2 Joint Motions Comparison with KDL

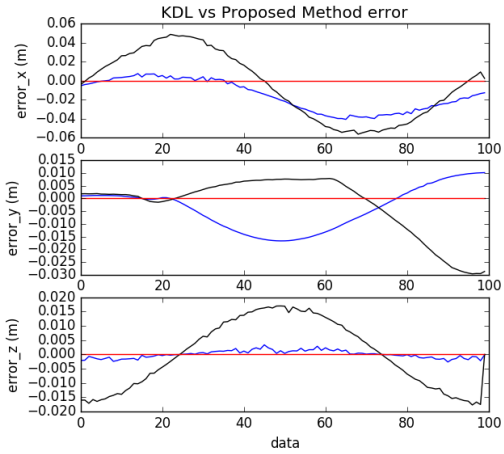
This section presents the comparison of the proposed optimization-based approach with the inverse kinematics solver called KDL provided within ROS. Similar to the proposed algorithm, KDL can generate manipulator joint motions such that the end-effector follows the specified trajectory. However, in its current form, KDL is only applicable for the fixed based manipulator. Thus for comparison purposes, an example where the mobile base motion is fixed to a known trajectory, and only compute the joint motions for the manipulator is considered. In effect, the mobile base motion can be subtracted from the desired end-effector trajectory to obtain the part which requires to be followed by only the manipulator.
The comparison is performed in terms of the following metrics: The first metric is the trajectory tracking error. The second metric is the smoothness defined in terms of norm of the joint acceleration. Lower the norm, the smoother the joint trajectory. The final metric is the manipulability index defined in the following manner.

$$\textbf{Manipulability} = \sqrt{det(\mathbf{J}(\mathbf{q})\mathbf{J}(\mathbf{q})^T)}, \tag{5.1}$$

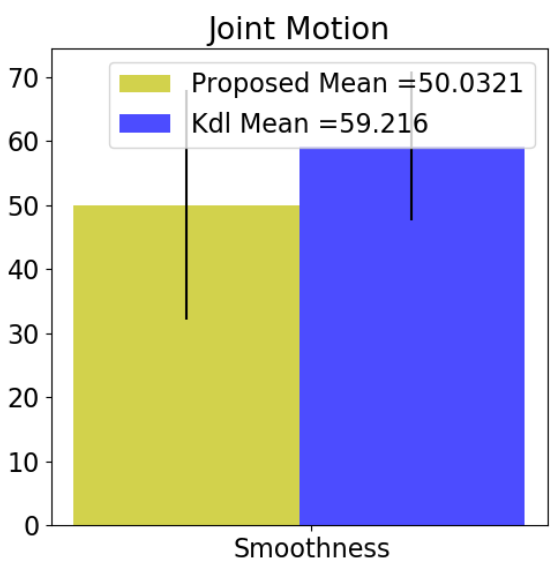
$\mathbf{J}(\mathbf{q})$ is the Jacobian of the manipulator and $det(.)$ refers to the determinant of a matrix. Higher the manipulability index, the farther the manipulator is from the singular configuration. The results are summarized in Figure 5.4(a)-5.4(d). As shown in Figure 5.4(a) and 5.4(b), the proposed optimization-based approach follows the given end-effector trajectory more closely. Furthermore as shown in Figure 5.4(c) and Figure 5.4(d), the proposed approach beats the KDL in terms of smoothness and manipulability index as well.



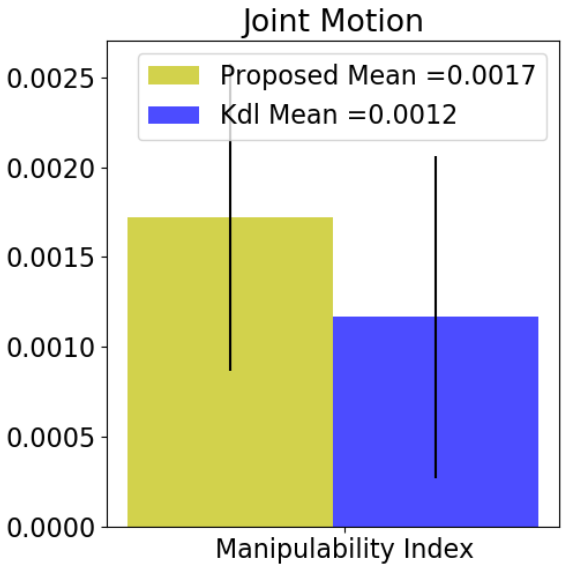
(a) *Elliptical Trajectory*



(b) *Trajectory tracking error*



(c) *Smoothness comparison*



(d) *Manipulability comparison*

Figure 5.4: *Comparison between proposed optimization-based approach and KDL Method*

5.3 End-effector Trajectory Tracking Analysis

In this section the results related to end-effector trajectory tracking for the elliptical, infinity symbol and triangular end-effector trajectory are presented. Figure 5.5 depicts the user specified desired trajectories in red. The simulation tracked values (plotted in blue) were taken by running the simulation and a *tf_listener*, whereas for the real robot the tracked trajectory values (displayed in green) were taken by using a camera and *ar_marker* attached to the end-effector of the mobile manipulator robot. The absolute error is shown in three plots for each axis.

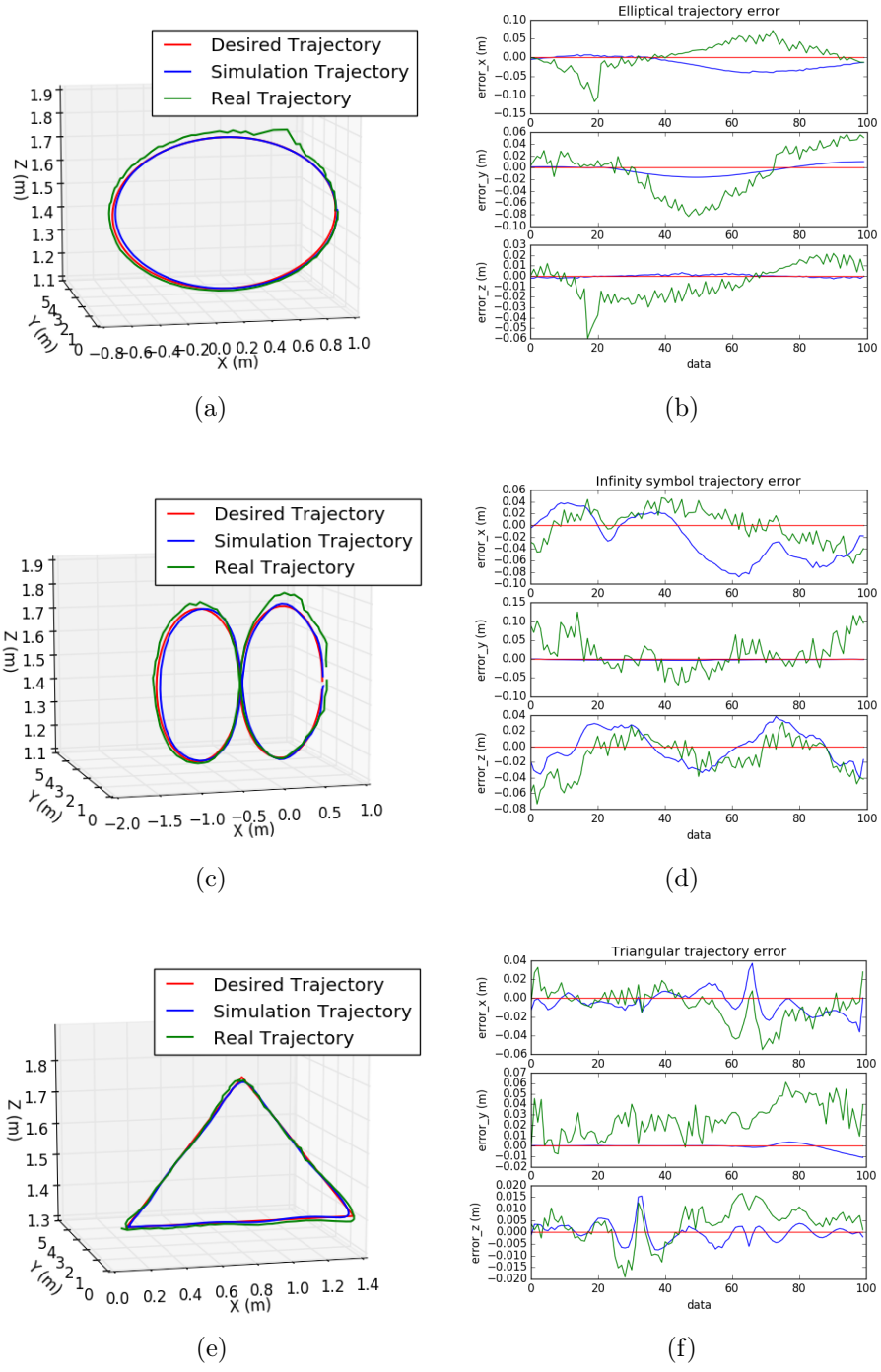


Figure 5.5: *Trajectory tracking with desired trajectory (red), simulation tracked (blue) and real robot tracked (green): 5.5(a) elliptical, 5.5(c) infinity symbol and 5.5(a) triangular trajectory. 5.5(b),5.5(d),5.5(f) shows error in x,y and z axis for each trajectory*

The numerical tracking performance evaluation are listed in the following Tables 5.1,5.2 and 5.3, from which it can be seen that in general the real implementation shows a higher error as compared with the simulation, which may be also related to the fact that the values are

taken from an *ar_marker*, and can be affected by light conditions, size of the marker and the calibration of the camera. However, the robot was able to execute all three trajectories with mean error values of less than 0.035 (m) in all axes. In the elliptical trajectory, the maximum error that occurs at one instance of time with a value of 0.118 (m) the x axis. In the infinity symbol and triangular, the highest error is presented in the y axis, with 0.125 (m) and 0.061 (m) respectively. It can be seen that the highest root mean square error is present in y axes for all three real tracked trajectories, with values of 0.04 (m), 0.046 (m), and 0.03 (m) respectively.

Table 5.1: Tracking performance evaluation - Elliptical Trajectory

	Simulation			Real		
	mean error (m)	max error (m)	RMSE (m)	mean error (m)	max error (m)	RMSE (m)
x	0.018	0.041	0.023	0.027	0.118	0.036
y	0.007	0.017	0.009	0.033	0.084	0.040
z	0.001	0.003	0.001	0.014	0.060	0.018

Table 5.2: Tracking performance evaluation - Infinity Symbol Trajectory

	Simulation			Real		
	mean error (m)	max error (m)	RMSE (m)	mean error (m)	max error (m)	RMSE (m)
x	0.038	0.088	0.046	0.022	0.066	0.027
y	0.002	0.004	0.003	0.034	0.125	0.046
z	0.019	0.041	0.021	0.020	0.073	0.026

Table 5.3: Tracking performance evaluation - Triangular Trajectory

	Simulation			Real		
	mean error (m)	max error (m)	RMSE (m)	mean error (m)	max error (m)	RMSE (m)
x	0.010	0.037	0.013	0.014	0.055	0.019
y	0.001	0.011	0.003	0.026	0.061	0.030
z	0.003	0.015	0.004	0.007	0.019	0.008

6 Conclusions and Future Work

In this thesis, the integration of mobile base and manipulator was formulated as a trajectory optimization problem, in which it was possible to generate a motion planning for a non-holonomic mobile manipulator to follow a given trajectory, by using the kinematics model, end-effector pose constraints and weight tuning.
The forward and inverse kinematics modeling of the manipulator were obtained by using DH method as well as homogeneous transform matrices, and the integration with the mobile base were adjusted by a transformation between the mobile base and the manipulator. The smoothness cost let to minimize the acceleration in the manipulator joints, and the base motion cost allowed to minimize the base velocities. The porposed trajectory optimization method solves the cyclicity bottleneck while trajectories are achieved with any desired degree of differentiability.
Satisfactory results were obtained in the context of trajectory tracking, where the robot successfully perform the three given trajectories: elliptical, infinity symbol and triangular, with small error both in simulation and implementation with real robot. The results showed that the proposed method generates smoother manipulator joint trajectories compared to KDL algorithm.
In this thesis, the coordination between the manipulator and mobile base is expressed in the kinematics model, however, when running a trajectory with the real hardware, the movement of the arm may influence the motion of the platform. Therefore, it would be necessary to include the dynamics analysis of the mobile manipulator, where the optimization method also contemplates the joint torques and contact forces.
Although the proposed trajectory optimization method is able to generate collision free trajectories, it does not contemplate obstacle avoidance. This could be implemented in future work by including the obstacle configuration in the cost function.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Arun Kumar Singh and Karl Kruusamäe, for their encouragement, comprehensive advice, and support during this thesis.

I would also like to thank the Faculty of Science and Technology for providing me the opportunity to extend my knowledge in the field of Robotics and Computer Science.

I want to express my very genuine gratitude to my family, who always provides love and unconditional support.

Finally, I wish to thank my friends and everyone who played a role in my academic accomplishments.

Bibliography

[1] B. Siciliano and O. Khatib. *Springer Handbook of Robotics, 2nd ed.* Springer International Publishing, 2016.

[2] Khaled Goher, Naz Mansouri, and S. Fadlallah. "Assessment of personal care and medical robots from older adults' perspective". In: *Robotics and Biomimetics* 4 (Dec. 2017). DOI: 10.1186/s40638-017-0061-7.

[3] O. Khatib et al. "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96.* Vol. 2. 1996, 546–553 vol.2.

[4] R. Carlton and S. Bartholet. "The evolution of the application of mobile robotics to nuclear facility operations and maintenance". In: *Proceedings. 1987 IEEE International Conference on Robotics and Automation.* Vol. 4. 1987, pp. 720–726.

[5] P. Lehner et al. "Mobile manipulation for planetary exploration". In: *2018 IEEE Aerospace Conference.* 2018, pp. 1–11.

[6] M. E. Tiryaki, X. Zhang, and Q. Pham. "Printing-while-moving: a new paradigm for large-scale robotic 3D Printing". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2019, pp. 2286–2291.

[7] Arun Kumar Singh et al. "Introducing multi-convexity in path constrained trajectory optimization for mobile manipulators". In: *2020 European Control Conference (ECC).* IEEE. 2020, pp. 1178–1185.

[8] *CLOOS Welding Robots.* URL: https://www.cloosrobot.com/de-us/products/qirox/ (visited on 04/12/2020).

[9] *FANUC Painting robots.* URL: https://www.fanuc.eu/se/en/robots/robot-filter-page/paint-series (visited on 04/12/2020).

[10] *Motoman Material Handling Robots.* URL: https://www.motoman.com/en-us/applications/handling (visited on 04/12/2020).

[11] *KUKA Industrial Robots.* URL: https://www.kuka.com/en-de/products/robot-systems/industrial-robots (visited on 04/12/2020).

[12] Y. Yamamoto and Xiaoping Yun. "Unified analysis on mobility and manipulability of mobile manipulators". In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C).* Vol. 2. 1999, 1200–1206 vol.2.

[13] Birgit Graf, Matthias Hans, and Rolf Schraft. "Care-O-bot II—Development of a Next Generation Robotic Home Assistant". In: *Auton. Robots* 16 (Mar. 2004), pp. 193–205. DOI: 10.1023/B:AURO.0000016865.35796.e9.

[14] Yan Guo et al. "Research on Centroid Position for Stairs Climbing Stability of Search and Rescue Robot". In: *International Journal of Advanced Robotic Systems* 7 (Dec. 2010). DOI: 10.5772/10493.

[15] *TALON Military Robot*. URL: https://www.army-technology.com/projects/talon-tracked-military-robot/ (visited on 07/26/2020).

[16] Mads Hvilshøj et al. "Autonomous industrial mobile manipulation (AIMM): Past, present and future". In: *Industrial Robot* 39 (Mar. 2012), pp. 120–135. DOI: 10.1108/01439911211201582.

[17] K. Nagatani et al. "Motion planning for mobile manipulator with keeping manipulability". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 2. 2002, 1663–1668 vol.2.

[18] Paolo Dario et al. "MOVAID: a personal robot in everyday life of disabled and elderly people". In: 1999.

[19] Advait Jain and Charles Kemp. "EL-E: An assistive mobile manipulator that autonomously fetches objects from flat surfaces". In: *Autonomous Robots* 28 (Sept. 2010), pp. 45–64. DOI: 10.1007/s10514-009-9148-5.

[20] Tiffany L. Chen et al. "Robots for Humanity : A Case Study in Assistive Mobile Manipulation". In: 2012.

[21] Oussama Khatib. "Mobile Manipulators: Expanding the Frontiers of Robot Applications". In: *Field and Service Robotics*. Ed. by Alexander Zelinsky. London: Springer London, 1998, pp. 6–11. ISBN: 978-1-4471-1273-0.

[22] Serdar Soylu, B. Buckham, and Ron Podhorodeski. "Redundancy resolution for underwater mobile manipulators". In: *Ocean Engineering* 37 (Feb. 2010), pp. 325–343. DOI: 10.1016/j.oceaneng.2009.09.007.

[23] Pere Ridao et al. "Intervention AUVs: The next challenge". In: *Annual Reviews in Control* 19 (Nov. 2015). DOI: 10.1016/j.arcontrol.2015.09.015.

[24] F. Ruggiero, V. Lippiello, and A. Ollero. "Aerial Manipulation: A Literature Review". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1957–1964.

[25] A. Suarez et al. "Anthropomorphic, compliant and lightweight dual arm system for aerial manipulation". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 992–997.

[26] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761. eprint: https://doi.org/10.1177/0278364911406761. URL: https://doi.org/10.1177/0278364911406761.

[27] Mohamed Elbanhawi and Milan Simic. "Sampling-Based Robot Motion Planning: A Review". In: *IEEE Access* 2 (Feb. 2014), pp. 56–77. DOI: 10.1109/ACCESS.2014.2302442.

[28] G. Oriolo and C. Mongillo. "Motion Planning for Mobile Manipulators along Given End-effector Paths". In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* 2005, pp. 2154–2160.

[29] F. Burget, M. Bennewitz, and W. Burgard. "BI2RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* 2016, pp. 3714–3721.

[30] L. Jaillet and J. M. Porta. "Path Planning Under Kinematic Constraints by Rapidly Exploring Manifolds". In: *IEEE Transactions on Robotics* 29.1 (2013), pp. 105–117.

[31] *Trajectory Planning.* URL: `https://danielpiedrahita.wordpress.com/portfolio/cart-pole-control/` (visited on 07/25/2020).

[32] Nathan Ratliff et al. "CHOMP: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE International Conference on Robotics and Automation.* IEEE. 2009, pp. 489–494.

[33] Mrinal Kalakrishnan et al. "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE international conference on robotics and automation.* IEEE. 2011, pp. 4569–4574.

[34] John Schulman et al. "Motion planning with sequential convex optimization and convex collision checking". In: *The International Journal of Robotics Research* 33 (Aug. 2014), pp. 1251–1270. DOI: `10.1177/0278364914528132`.

[35] Jianfeng Liao et al. "Optimization-based motion planning of mobile manipulator with high degree of kinematic redundancy". In: *International Journal of Intelligent Robotics and Applications* (2019), pp. 1–16.

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

[37] John Schulman et al. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization". In: *Robotics: Science and Systems.* 2013.

[38] P. Dong and X. Zhao. "Static path planning of tracked mobile manipulator and simulation". In: *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC).* 2011, pp. 2266–2269.

[39] Cressel Anderson et al. "Mobile manipulation: A challenge in integration". In: (Apr. 2008). DOI: `10.1117/12.777329`.

[40] C. Gao, M. Zhang, and L. Sun. "Motion Planning And Coordinated Control For Mobile Manipulators". In: *2006 9th International Conference on Control, Automation, Robotics and Vision.* 2006, pp. 1–6.

[41] *ROS Powering the world's robots.* URL: `https://www.ros.org/` (visited on 04/16/2020).

[42] *ROS Introduction.* URL: `http://wiki.ros.org/vn/ROS/Introduction` (visited on 04/16/2020).

[43] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: vol. 3. Jan. 2009.

[44] *ROS Core Components*. URL: https://www.ros.org/core-components/ (visited on 04/16/2020).

[45] Wei Qian et al. “Manipulation Task Simulation using ROS and Gazebo”. In: Dec. 2014. DOI: 10.1109/ROBIO.2014.7090732.

[46] *ROS Concepts*. URL: http://wiki.ros.org/ROS/Concepts (visited on 04/16/2020).

[47] *ROS Communication*. URL: http://wiki.ros.org/ROS/Patterns/Communication (visited on 04/17/2020).

[48] *ROS Topics*. URL: http://wiki.ros.org/Topics (visited on 04/18/2020).

[49] *ROS actionlib*. URL: http://wiki.ros.org/actionlib (visited on 04/19/2020).

[50] *Unified Robot Description Format (URDF)*. URL: http://wiki.ros.org/urdf (visited on 06/16/2020).

[51] *ROS Rviz*. URL: http://wiki.ros.org/rviz (visited on 06/16/2020).

[52] T. Foote. “tf: The transform library”. In: *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. 2013, pp. 1–6.

[53] *ROS tf*. URL: http://wiki.ros.org/tf (visited on 06/17/2020).

[54] J. Xiao, W. Han, and A. Wang. “Simulation research of a six degrees of freedom manipulator kinematics based On MATLAB toolbox”. In: *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*. 2017, pp. 376–380.

[55] Jacques Denavit. “A kinematic notation for lower-pair mechanisms based on matrices.” In: 1955.

[56] Y. Xiao et al. “A Manipulator Design Optimization Based on Constrained Multi-objective Evolutionary Algorithms”. In: *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*. 2016, pp. 199–205.

[57] *Mobile Robot Kinematics*. URL: http://www.cs.cmu.edu/~rasc/Download/AMRobots3.pdf (visited on 07/25/2020).

[58] Soonshin Han, Byoungsuk Choi, and Jangmyung Lee. “A precise curved motion planning for a differential driving mobile robot”. In: *Mechatronics* 18 (Nov. 2008). DOI: 10.1016/j.mechatronics.2008.04.001.

[59] Marc Toussaint. “A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, optimal control, and probabilistic inference”. In: *Geometric and numerical foundations of movements*. Springer, 2017, pp. 361–392.

[60] *Skywalker Github*. URL: https://github.com/ut-ims-robotics/skywalker.git (visited on 07/28/2020).

[61] *Mobile platform: MiR100*. URL: https://www.mobile-industrial-robots.com/en/solutions/robots/mir100/ (visited on 06/12/2020).

[62] *MiR100: Github Repository*. URL: https://github.com/dfki-ric/mir_robot (visited on 06/12/2020).

[63] *UR5e: Universal Robot*. URL: https://www.universal-robots.com/products/ur5-robot/ (visited on 06/12/2020).

[64] *ROS universal_ robot GitHub repository*. URL: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver (visited on 06/12/2020).

[65] *UR5e: DH Parameters*. URL: https://www.universal-robots.com/articles/ur/parameters-for-calculations-of-kinematics-and-dynamics/ (visited on 07/29/2020).

[66] P. Castillo-Pizarro, T. V. Arredondo, and M. Torres-Torriti. "Introductory Survey to Open-Source Mobile Robot Simulation Software". In: *2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*. 2010, pp. 150–155.

[67] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3.

[68] *AR Tracker*. URL: http://wiki.ros.org/ar_track_alvar (visited on 07/25/2020).

Non-exclusive licence to reproduce thesis and make thesis public

I, Muhammad Usman

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   "Development of an Optimization-Based Motion Planner and Its ROS Interface for a Non-Holonomic Mobile Manipulator"

   supervised by Arun Kumar Singh, Karl Kruusamäe,

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Muhammad Usman
15-08-2020