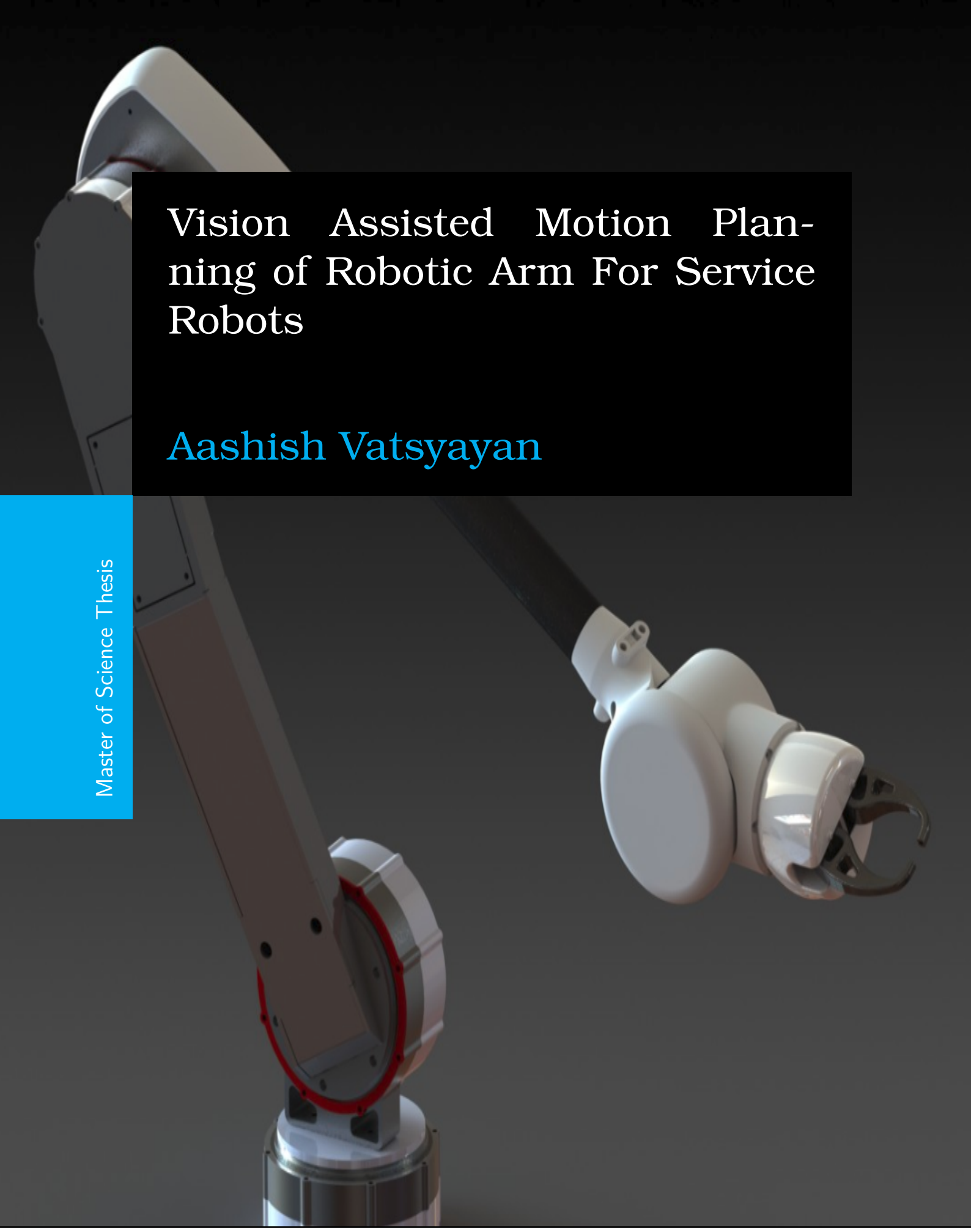

Vision Assisted Motion Planning of Robotic Arm For Service Robots

Aashish Vatsyayan

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Vision Assisted Motion Planning of Robotic Arm For Service Robots

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Aashish Vatsyayan

August 15, 2016

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

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

Abstract

In this thesis a vision assisted system is developed for manipulation of a robotic arm which is to be used in unconstrained environments with service robots. The vision module comprises of segmentation and object tracking that allows the user to select the object they want to grasp. It is shown that GrabCut segmentation improves the efficiency of the Tracking-Learning-Detection (TLD) tracker. The Moveit! platform for solving motion planning problems is used in this thesis. Apart from the default Open Motion Planning Library (OMPL) in Moveit!, Stochastic Trajectory Optimization for Motion Planing (STOMP) and Search-Based Planning Library (SBPL) have also been explored to solve motion planning problems. Inverse kinematics based genetic search is used for generation of waypoints in challenging manipulation tasks and has been incorporated into the Moveit! framework. The waypoints generated through genetic search have shown to be valid. Additionally an analysis is done to evaluate the performance of different motion planning libraries to find implementable solutions in cases of varying relative position to arm and clearances to nearby obstacles. It is shown that certain motion planning libraries have superior performance for varying clearance and position of goal. A contextual awareness module is developed that determines the best planning algorithm for the clearance from obstacles and relative position of the target pose to the arm. A flexible framework is created that incorporates the vision module, genetic search, contextual awareness and allows for switching between the three motion planning libraries. The system is also tested on the robotic arm at Robot Care Systems.

Table of Contents

List of Figures

List of Tables

Preface

All the work presented in this thesis was conducted at Robot Care Systems, the sources of information are cited and all work presented is original.

Acknowledgements

I would like to thank my supervisor Prof. Dr. Robert Babuska for his assistance during the writing of this thesis. I would also like to thank Aswin Chandarr for his insight and guidance in my research. Finally, I would like to thank Robot Care System for providing me with the incredible opportunity to do research and test my findings.

Delft, University of Technology — Aashish Vatsyayan
August 15, 2016

"A poem is no different than a painting, nor is it different from a machine.
In their essence they are all ideas, skill, structure, patience, hard work and dreams."

— *Anonymous*

Chapter 1

Introduction

Robots are becoming a part of our everyday lives. From the room cleaning Roomba [1] to PARO [2] the robotic seal that is used for comforting people, robots have integrated themselves into our lives. Service robots have found more implementation in recent years due to development of faster systems that are more intuitive to use and can be operated by anyone.

In this thesis robotic manipulation for service robots is addressed by considering the requirements of the users, the challenges of an unconstrained environment. A system is developed with modules to increase the success of motion planning tasks.

Robot Care Systems has developed a robotic stroller Lean Elderly Assistant (LEA) aimed to help elderly users walk and exercise. The users often have difficulty in reaching for objects on the floor or on a high shelf. The manipulator mounted on LEA can be of great assistance in these situations. LEA can also navigate from one room to another, combined with the ability to grasp objects it can be used to retrieve objects in the event that the user is unable to do so themselves. The objective of LEA is to assist the elderly in carrying out day to day tasks with ease and save time on the involvement of care takers for the elderly. A robotic arm would empower the user to pick and place objects, combined with the navigational capabilities of LEA it also opens the possibility of remote grasping.

For safe and successful grasping of objects it is important that the motion planning is accurate, avoids obstacles, has smooth and power economic movements. Aspects of vision and motion planning are examined to develop a system that incorporates all the aforementioned attributes. In the first chapter state of the art service robots are examined, the use cases for users interacting with the manipulator are created and the aspects of vision and motion planning required for the thesis are reviewed. Following chapters provide an insight into the framework of the system developed in the thesis followed by vision and motion planning in detail.

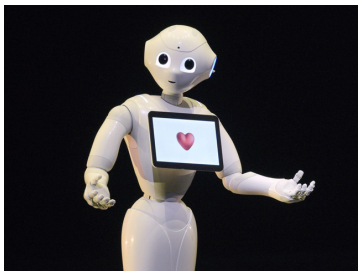

Figure 1-1: Pepper

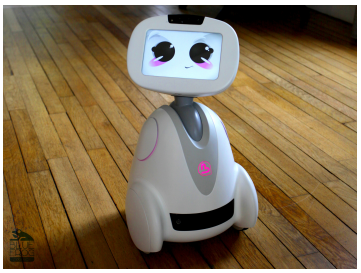

Figure 1-2: Buddy

Figure 1-3: Paro

1-1 Service Robotics

Industrial robotics primarily focuses on precision, speed and efficiency whereas service robotics is more concerned with safe robot-human interaction and navigation with the uncontrolled environment.

As per the International Federation of Robotics [3] :

- A service robot performs useful tasks for humans excluding industrial automation application.
- A personal service robot or a service robot for personal use is a service robot used for health care tasks, usually by untrained people. Examples are domestic servant robot, automated wheelchair, personal mobility assist robot, and pet exercising robot.
- A professional service robot or a service robot for professional use is a service robot used for a commercial task, usually operated by a properly trained operator. Examples are cleaning robot for public places, delivery robot in offices or hospitals, fire-fighting robot, rehabilitation robot and surgery robot in hospitals. In this context an operator is a person designated to start, monitor and stop the intended operation of a robot or a robot system.

The service industry of robots aims at aiding and assisting people to do everyday tasks with ease. Robots such as Pepper [4] and Buddy [5] have revolutionized the role of robots in human lives. Capable of voice recognition the robots are aimed at making lives more comfortable for the users. It can read emotions of humans and act accordingly. Buddy is developed as a companion robot capable of home security, control of smart devices in a house, social interaction such as telepresence, fall detection and unusual activity detection.

In [6] a wheelchair-based rehabilitation robotic system for the disabled is described. The developed system is titled KAIST Rehabilitation Engineering Service System II (KARES II) and has two main components: robotic arm and user interfaces. To compensate for different levels of disability they made interfaces through movement of eyes, shoulders as an input for the system. University of Bremen have developed care providing robot FRIEND [7] which has a wheelchair and a robotic arm capable of picking and placing objects for the user.

Areas of research such as cognition, image processing, artificial intelligence, motion planning, stable and robust control are all tied into completing tasks in service robotics. There has been focus on developing algorithms that achieve higher efficiency in setting where robots are expected to be used such as indoors.

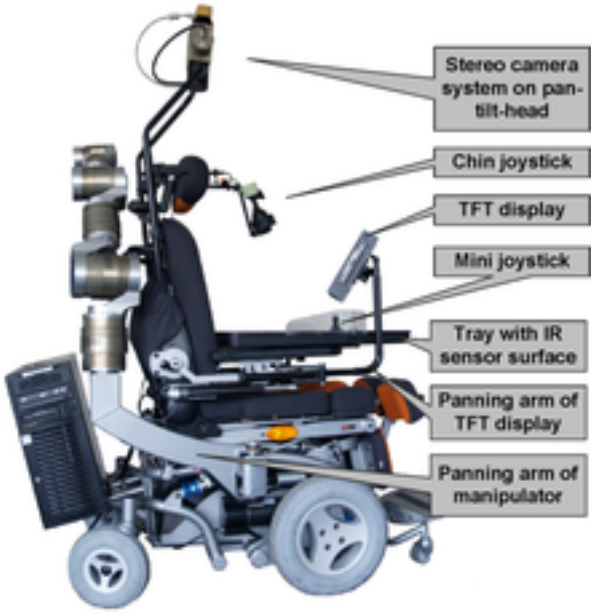

Figure 1-4: FRIEND

1-2 Recent Developments

There has been significant progress in the area of service robotics. Willow Garage, a former software company made several developments in motion planning, obstacle avoidance and their integration onto a single platform. Rice university has developed some cutting edge motion planning algorithms such as the KPIECE [8]. In [9] an architecture is presented that integrates creation of 3D maps for collision detection and real time motion planning for the PR2 robot. The PR2 robot can be seen in Figure: 1-5

Figure 1-5: The PR2 robot

In [10] pick and place operations on the PR2 with 91 % accuracy in 6.7 seconds are presented. Such a pick and place application can be of significant interest in service robotics. In [11] presented perception and planning in cluttered and unstructured environments using a combination of 2D and 3D visual processing, tactile and proprioceptive sensor data, fast motion

planning and reactive control. This application was again on the PR2 robot.

1-3 Use cases

A service robot such as LEA (shown in Figure: 1-6) has to constantly interact with elderly users. In order to develop a system for manipulation it is first important to understand the needs and requirements of these users. For a service robot it is important to identify intended users and the desired behaviour towards them.

A short use case analysis is done recognizing the primary, secondary and tertiary users for the robotic arm and observing their requirements. The roles are described in priority of use. The things that the user has to be able to do is mentioned as available choices. The needs or expectations of the user from the system are discussed in 'Needs' and the safety precautions are discussed as precautions.

Figure 1-6: LEA

Primary user The elderly person using the stroller is the primary user.

Available Choices:

- Manual end effector control.
- Select object to pick up.
- Pick and place object from one location to another.
- Emergency freeze in case of malfunction.

Needs:

- A simple and intuitive interface for the user.
- Notification when there is an error.
- Sophisticated vision system for separating selected object from background.
- Motion planning to generate trajectories for grasping actions.

- Good controller design for motors to ensure smooth motion.
- Notification when an object is out of range or a place is unsuitable to place an object.

In order to meet the requirements of the primary user the system needs to have three crucial components : computer vision, motion planning and controller design. All other functionalities can be built on top of these main concepts. These concepts will be looked in detail in the following chapters.

Precautions:

- Movement must be constrained, collisions must be avoided.
- Verification from user if the correct object is isolated in the video from the user selection.
- Check if the object weighs less than the maximum weight the arm can lift and notify user if it isn't.

Secondary user The care giver or family member that is responsible for the care of the elderly user is the secondary user.

Available Choices:

- Handing and receiving objects to and from the arm

Needs:

- Avoiding collision and not overshooting when handing objects.
- Detection of hands while handing object over.

Precautions:

- Checking if the object has been successfully handed or picked

The use cases specify the requirements from the robotic arm and were considered while developing the system and selecting the appropriate algorithms. The main requirements that emerged were: safety, ease of use, obstacle avoidance and accuracy. This thesis aims at targeting the vision and motion plannings aspects of the points determined.

Now that the use cases for users have been identified the tools required for development of a system that can implement the requirements will be examined. First the vision aspect of the system will be examined and then the motion planning module will be looked into.

1-4 Vision

This section explores the tools needed for interacting with the user, keeping a track of a selected object and how it can be improved in a cluttered environment. The vision system can be used for keeping a track on the movement of the desired object. Vision can also be used to compensate for poor joint encoders to estimate the position of the end effector.

A house is a highly unconstrained environment, things may be moved intentionally or by accident. The thesis aims at developing a system that can accommodate this uncertainty. The first task however is to allow the user to select the object they want to grasp. In the literature review several tracking algorithms were examined and TLD tracker [12] was selected due to its ability to handle change in poses, user interaction and its 3 layered structure that allows for tracking, learning and detection.

TLD tracker allows users to generate the bounding boxes for objects they want to grasp. This allows for user interaction but at the same time allows for accumulation of noise that will be tracked due to addition of extra areas in the bounding box apart from the object they want to grasp. The user may not always be able to accurately select the object they want to grasp in a cluttered environment. This can be rectified by using a segmentation algorithm in conjugation with the tracking. Through research conducted in the literature review GrabCut [13] was selected due to its superior performance in the presence of background clutter. Figure: 1-7 from [13] shows the performance of GrabCut in comparison with Magic Wand (Adobe Systems Incorp. 2002), Intelligent Scissors [14], Bayes matting [15], Knockout 2 (Corel Corporation 2002) and Graph cut [16].

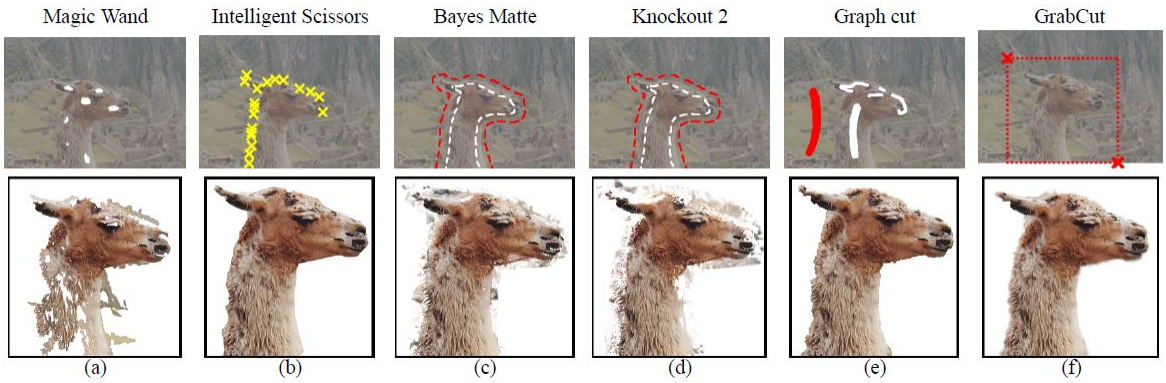

Figure 1-7: GrabCut algorithm in comparison with other segmentation algorithms

The superior performance in comparison with other state of the art segmentation algorithms can be seen. The integration of segmentation and tracking and its implementation are explored in Chapter : 3. The tracked coordinates of the target object can be used to estimate motion of the object. In the event that the object moves the trajectory execution is paused. In the event there is no movement of the target object the 3D coordinates of the goal are passed to the state machine (discussed in Chapter: 2) to generate a motion plan.

Table 1-1: Comparison of motion planning planning libraries

Planning Library	Advantages	Disadvantages
OMPL	-Fast, versatile -Sampling based algorithms are effective for obstacle avoidance -Tried and tested	-Random trajectories are possible -Poor orientation constraint handling
SBPL	-Consistent movements -Objective function minimized -Uses motion primitives	-Poor obstacle avoidance and constraint handling -Higher planning time
STOMP	-Stochastic trajectory optimization -Good constraint handling -Local planner -Smooth trajectories	-High planning time -Difficult to solve global path planning problems in realistic time

1-5 Motion Planning

This section explores state of the art motion planning methodologies that may be utilized in the thesis. The motion planning research conducted in the literature review presented strengths and weakness of some motion planning libraries. This section evaluates 3 motion planning methodologies and identify their strengths and weaknesses at the same time reflecting their applicability in the thesis considering the requirements discussed earlier.

The libraries considered are: Open Motion Planning Library (OMPL) [17], Search-Based Planning Library (SBPL) [18] and Stochastic Trajectory Optimization for Motion Planing (STOMP) [19] .

A short introduction into the underling principles of these motion planning libraries are presented in this section. A detailed insight can be found in the appendix.

OMPL: Has sampling, combinatorial, graph based, optimization and control based motion planning algorithms. It is one of the largest motion planning libraries.

SBPL: Relies on generating motion primitives, search algorithms and a graph based discritization of the environment to solve a motion planning problem.

STOMP: Uses stochastic trajectory optimization technique. Generates a simple trajectory between 2 points and then several noisy ones in order to optimize a cost function.

Each of these motion planning libraries have their strengths and weaknesses, particularly in the context of the thesis. Some strengths and weakness for the motion planning libraries are discussed in Table: 1-1

For the thesis each motion planning library will be explored and the strength of each will be used to boost the planning success rate. This is examined in Chapter: 4. It is not possible to say which motion planning library can solve the motion planning tasks in uncontrolled, cluttered environments. This thesis aims to apply the strengths of each planner in an attempt to boost the success rate.

1-6 Discussion

In this chapter the motivation for the thesis was examined, use cases for the indented users was discussed. The aspects of vision and motion planning were evaluated. TLD tracker and GrabCut segmentation was decided upon. For motion planning the benefits and drawbacks of each motion planning library were discussed. Since the robotic arm and the simulation have the same kinematics most of the experiments performed in the thesis are in simulation. Once proven to be safe and stable, the results from simulations are tested on the robotic arm.

The remainder of the thesis is structured as follows: Chapter: 2 looks at the overall framework for the system designed in the thesis and explores each building block. Chapter: 3 shows the implementation of tracking and segmentation algorithm and finally Chapter: 4 examines the implementation of each motion planning algorithm in the system, measures to improve the success rate are discussed and evaluated.

Chapter 2

Framework

In the previous chapter the motivation of the thesis was presented along with the choices for the vision and motion planning modules. Robotic manipulation in the service industry offers many challenges, in order to prepare a safe and modular solution having a good framework is important. The overall transfer of information is presented and each building block is examined. The framework also takes into account real world factors such as noise and unpredictable environmental changes. There are certain key factors kept in mind while designing the architecture of the thesis:

- Modular: All modules work independently and can be replaced, changed if needed.
- Adaptable: If there are changes in modules, the system must still work.
- Safe: There must be fail safes built in the framework in case of emergencies.

The entire system was developed in ROS [20] on Linux operating system. All the code is written in C++. For motion planning Moveit! [21] platform was used and for vision ROS package for TLD was used and modified to meet the requirements of thesis. Before testing the motion planning on the actual robot the system was developed and tested in simulation. V-rep [22] was used as a replacement for the real world robot. The following sections will describe the developed framework. This chapter focuses on the overview of the system developed in the course of the thesis followed by an explanation of the building blocks involved.

2-1 Overview

Figure: 2-1 shows the overall framework of the system developed in this thesis. The information received from the 3D camera consists of RGB (Red Green Blue) data and Pointcloud messages. Pointcloud data contains the depth information of points in an image. This information is then sent to the TLD tracker and displayed to the user.

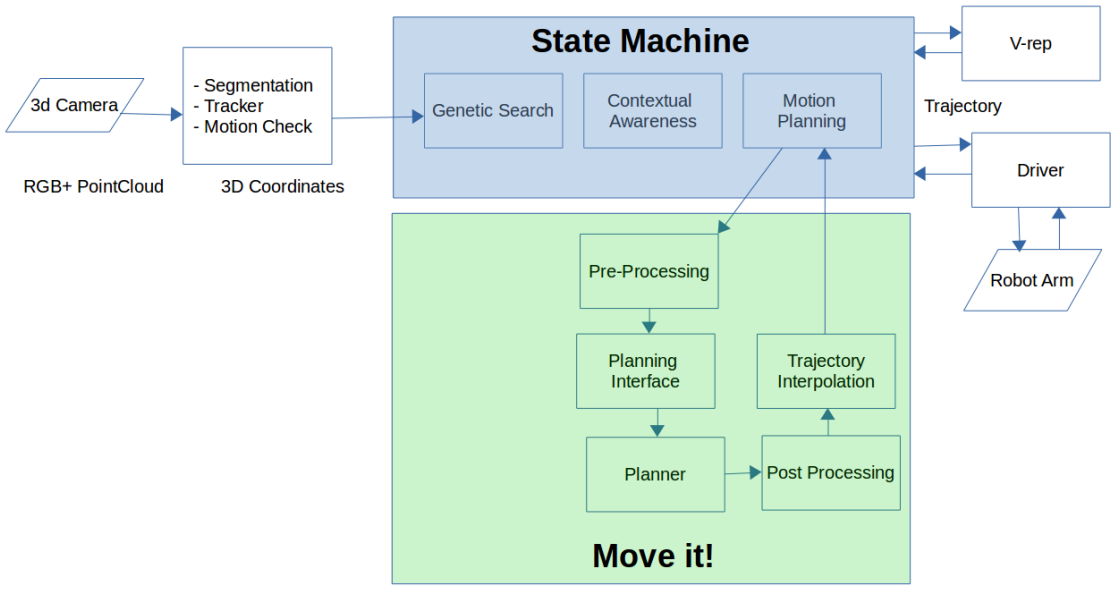

Figure 2-1: Overall framework

Here, the user selects the object they want to grasp. Before the coordinates are sent to the state machine three crucial functions are performed. First is the tracking initialization, second is the segmentation of selected object from the background and continuation of tracking. The final function performed is the motion check. This module examines the position of the selected bounding box and if the position of the object changes more than a certain threshold it is considered as moved. If the object is moved a command is sent to the state machine that stops the execution of trajectory. This module adds a dynamic element to the developed system.

The state machine is where all the essential decision making is done. Here all the information from different sources is processed and request for motion planning is made. The decision making is done via a state machine architecture. From this block both simulation and robot arm hardware can be accessed. The main output of this block is the generated trajectory for the motion planning problem. If the simulation is being run then V-rep environment is launched which mimics the real world settings. This is useful for testing algorithms and study their execution before testing it on the real robot arm.

In order to operate the robotic arm the driver sends the joint positions to the motors on the manipulator. The local Proportional Integral Differential (PID) controllers ensure the execution of joint position commands sent by the driver. The communication takes places through a Controller Area Network (CAN) bus. The CAN messages are explained in this chapter. Execution of trajectories on the robotic arm is demonstrated in Chapter: 4.

2-2 Building Blocks

An insight into the building blocks of the framework described is explained.

ROS

This section explains the concepts that overlap for the framework used in simulation and the actual robot. There are many ways to communicate with different packages in ROS. This can be done in the form of services, messages, publishers/subscribers, parameter server etc. More information about ROS framework can be found in [23]. In ROS the communication between different packages takes place in the form of messages that are published on topics. In order to transmit a message a publisher has to be initialized that broadcasts a message of a specific data type on a topic.

Figure : 2-2, shows the mechanism of ROS publishers and subscribers for communication between two nodes. In the thesis communication takes place via publisher-subscriber from the vision module to the state machine and then to V-rep or robotic arm driver subsequently. Joint states from the robot driver or V-rep are also communicated to the state machine via publisher-subscriber.

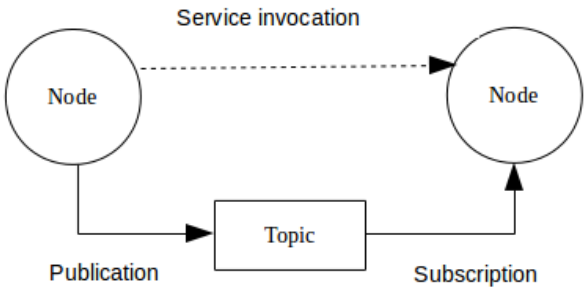

Figure 2-2: ROS communication

ROS was developed primarily for robotics (as the name suggest). The node based communication allows for different nodes to be called when they are needed. To bring this into perspective, a motion planning node will begin to process information received from the vision node when the data transmitted.

In order to initialize a package and launch a desired node the `roslaunch` file is used. It is a tool for launching ROS nodes and setting ROS parameters. More information for the `roslaunch` package can be found in [24]. This chapter deals only with the way the information is passed from one package/node to another. The details of what happens inside the vision and motion planning modules are presented in chapters 3 and 4 respectively. The video feed is provided from the 3D camera that is initialized by the `openni2` package [25], which is launched on start up. The `openni2` package essentially sets the calibration parameters for the camera and makes RGB and depth information available. The depth information is present in the form of ordered arrays. In order to extract the x, y and z information of a point in the video it has to be isolated using the x and y coordinates.

Once the RGB and depth data stream is initialized by the `openni2` package `tld_gui` package is called. This package starts a Graphical User Interface (GUI) which has the option to select the points that define the required object. The selected points are published on a topic which has a subscriber in `tld_tracker` node. The PointCloud2 information is also taken by the `tld_tracker` package via a subscriber. Using the point coordinates and the depth information a 3D point that represents the desired object is generated, however this information cannot be used as is. The generated 3D point is with respect to the camera frame of reference, whereas for motion planning it should be from the frame of reference of the base link of the robot. In order to transform the coordinates from one frame of reference to another the ROS transformation package [26] is used.

The package takes the joint position of each joint, the robot description and generates information about the position of each link and joint when there is motion. If the camera is placed statically then the transform will be constant, however if the camera is placed on a moving link of the robot arm then the dynamic transforms are generated. For this to happen the `move_group` node has to be active. This node in turn is activated when the `user_client` node is activated. The `tld_tracker` node waits for this information to be published on the `tf` topic. Once a transform is present between the camera and the base link, coordinates are published on a topic which the `user_client` node subscribes to.

Here the `user_client` node has to make the decision to send the 3D coordinates to begin motion planning. The communication between `user_client` and `user_server` is done through an action server [27]. An action server allows for a request from the client to the server to perform a task in a way that there is feedback and a possibility to stop the task in case there is a change in goal. An overview of this can be seen in Figure: 2-3

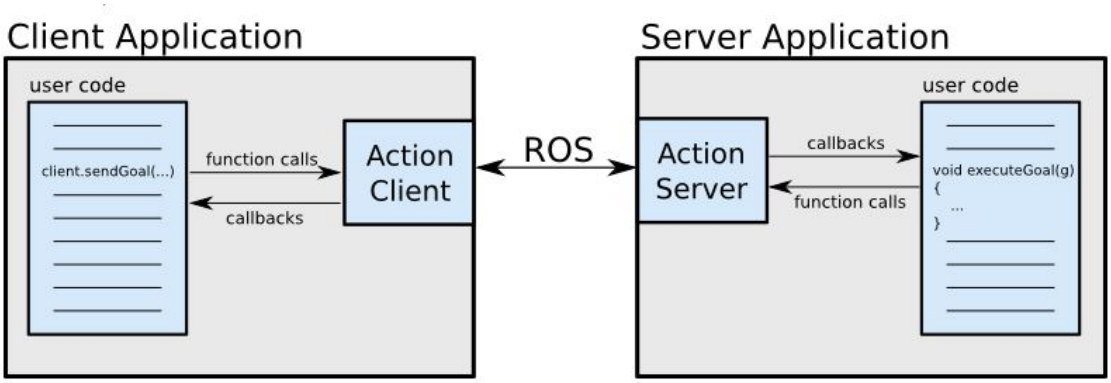

Figure 2-3: Client-Server interaction

The main aspects of client-server communication in ROS are:

- **Goal**: This is a user defined message that containing the x,y and z coordinated for motion planning.
- **Feedback**: The incremental progress in the server side, this is user defined as well. For this thesis it can be the status of solving the motion planning problem.
- **Result**: This feedback is sent exactly once in contrast to the Feedback for other processes. This might contain the final pose of the robot after execution motion planning.

In the thesis the `user_server` is the state machine as shown in Figure: 2-1. It is interfaced using a client-server architecture where the `user_client` receives the coordinates from the vision module. In the `user_client` motion detection is performed. Once there is no motion detected the coordinates are sent to `user_server`. Hence the module that performs tracking, segmentation and motion detection as shown in Figure: 2-1 consists of a package publishing the coordinates which is received by the `user_client` where motion detection is performed. This structure has been adopted keeping in mind that finally the user should be able to see the video on a tablet constantly for selection of object and once they are satisfied with the selection start the motion planning task.

The action-server communication adds to the modularity of the framework used in the thesis. The motion planning request to a certain point in 3D space is received in the `user_client` node which is the client in the action server and the request is sent to `user_server` node, which is the server in the form of a goal. Once the server is initialized the motion planning begins. Inside the `user_server` node the motion planning request is analyzed and an appropriate motion planning library is selected for initializing the `move_group` node which is the main node for communication with Moveit!. Inside the `user_server` a state machine architecture is used, which streamlines the motion planning process and gives feedback to the `user_action` node which in turn can relay messages to the user regarding the progress.

State Machine

The supervisory state machine is responsible for interfacing with all the sources of data and processing a suitable action. Figure: 2-4 shows the flowchart of the states

- Unfold: Initially docked in a folded position the arm postures itself in order to prepare for executing motion planning trajectories. This can be done using pre-determined joint state commands, motion planning is not required for this state. independently and can be replaced, changed if needed.
- Pre-Grasp Pose: Here the coordinates for motion planning are received and if possible a motion plan to the pre-grasp pose is generated. The target coordinates are first passed through genetic search that computes closest reachable point to the target. This coordinate is then examined via the contextual awareness module which examines the clearance of target, its position relative to the robot and produces the most suitable motion planner for the task. With the updated target coordinates and planner, a motion plan is requested. In the event that the target pose is not reachable the closest point generated by the genetic search is fixed as a way point. Once this is reached a local planner or conventional controller can be used to navigate through confided spaces. In the event that the coordinate generated by genetic search is within a threshold of the target pose a direct motion planning is attempted to it.
- Grasp: State to navigate arm to grasp the target object. If a successful grasp is achieved the next state is Place, else a new pre-grasp pose is attempted.
- Place: In the event of a successful grasp the target to place the object is examined and a valid point is selected to place the object.
- Fold: As indicative by its name, once the motion planning request is completed the arm is folded in the absence of a new motion planning command.

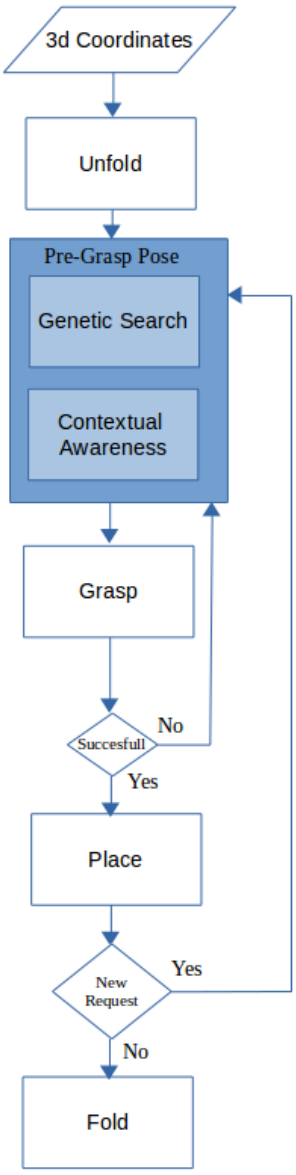

Figure 2-4: State machine overview

The thesis is focused on pre-grasping pose generation and trajectory execution for the robotic arm. Grasping pose generation for unknown shaped objects is not addressed in this thesis as it is a separate field of research from motion planning and it depends on the choice of gripper which has not been decided yet for the robotic manipulator RCS. The genetic search and contextual awareness modules are explained in detail in Motion Planning chapter. In essence the state machine supervises the motion planning from one state to another. The proposed system does not respond to dynamic obstacles. The environment is analyzed only in the beginning of the motion planning task and ignored during the execution of the trajectory. In order to make the system respond to dynamic obstacles the octomap has to be constantly monitored, however this will be addressed in future research.

Simulation

Once the building blocks are in place the algorithm and the system can be tested in simulation before being used on the actual setup. This established a good testbed to vary parameters and see their performance without damaging the robot. The simulation software has to be ROS compatible and it should be possible to import the robot description into its environment, furthermore it must be possible to add obstacles, daily household items and sensors in the simulation environment. A software is ROS compatible when communication in the form of publishers and subscribers can be established. V-rep [28] satisfies all these requirements as does Gazebo [29]. Due to ease of adding custom shaped objects in V-rep [30], it is selected for the thesis. ROS compatible messages can be broadcasted from inside of V-rep environment. After importing the robot description in the from of a Universal Robot Description Format (URDF) file a non-threaded child script is added to the robot. This child script is called recursively and is divided into different parts. Figure: 2-5 shows the robotic arm simulation in V-rep environment.

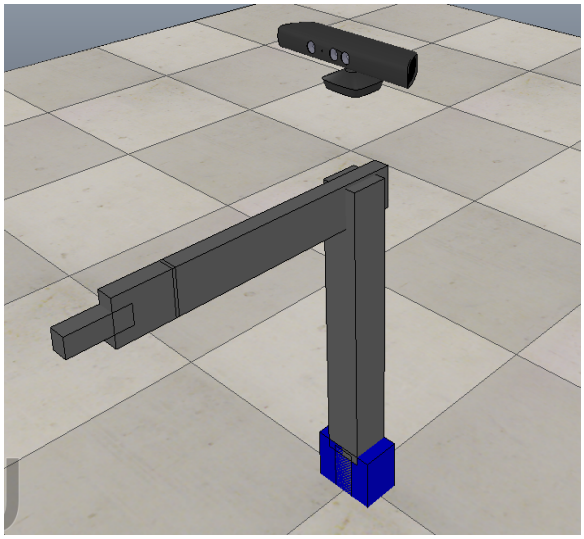

Figure 2-5: Robot arm in simulation

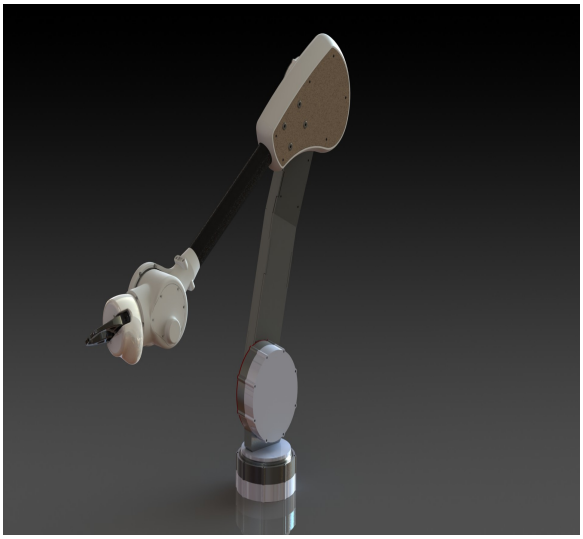

Figure 2-6: Robotic arm at RCS

Inside the section that is called recursively, the publishers and subscribers have to be initialized. Here the Joint state information is published on the `Joint_State` topic and position commands are received for each joint in the form of `Joint_State` messages. A 3D camera is present in the V-rep library and can be used to publish the `PointCloud2` depth messages that are compatible with ROS. For working in simulation the joint states are taken from the simulation environment. For the actual robot however a Joint state publisher is created.

Robotic Arm

For interfacing with the hardware a few changes need to be made from the simulated environment framework. In order to communicate with the robotic arm a driver has to be developed. CAN protocol is used to interface with the joint controllers in the arm. The hardware used at RCS is a custom arm developed in accordance with the needs of the robot. The manipulator

has 6 Degrees Of Freedom (DOF). The hardware was developed with Common Place Robotics [31] and the specifications for the arm can be found in [32]. Since the experiments for this thesis were performed with prototypes of the manipulator there were certain challenges that resulted in streamlining the development of future design of the arm. Figure 2-6 shows the robotic arm being used at RCS. The arm was designed to make it usable with service robots. The specifications of use are:

- Kinematics: Serial six axis standard kinematic
- Payload: 1.0 kg at half reach, 750 g fully extended
- Reach: 850 mm
- Weight: approx. 6.0 kg
- Communication: Internal CAN, external via Ethernet
- Supply Voltage: 24V

CAN protocol

This section details the CAN protocol that is used to communicate with the boards on the robotic arm. Each joint has a controller board that listens to a specific CAN ID. Figure: 2-7 shows the PCAN-USB adapter used to interface between the robot and a CPU.

Figure 2-7: PCAN-USB PEAK adapter

The exact CAN protocol can be found in the appendix.

2-3 Discussion

This chapter explored the framework and its individual modules used in the thesis. ROS, V-rep, the state machine, robotic arm hardware and the CAN protocol were examined. The proposed system for the thesis has several modules that are intended to improve the performance of the system. A drawback of the proposed system is that it cannot respond to dynamic obstacles. This can be addressed in future research. The next chapter explains the components of vision system incorporated in the system and the impact on performance.

Chapter 3

Vision

This chapter describes the components of the vision module used in the thesis. In order to implement efficient grasping and obstacle avoidance, awareness of the environment is necessary. This chapter will cover the implementation of each component in the vision module. First the TLD and Grabcut algorithms are implemented following which their integration will be implemented. It is demonstrated that by using the combination of segmentation and tracking algorithms the overall efficiency of the vision tracking system improves.

3-1 Segmentation

An average house will have several objects that may be stacked on top of each other or placed in the presence of other objects. Even if a cup is placed on a table for the purposes of manipulation it is important to know the exact position of the cup in a 3D space. In order to separate the cup from its background image segmentation algorithms will be used. An implementation of GrabCut algorithm is present in the Point Cloud Library (PCL) [33]. PCL offers a variety of image segmentation algorithms that can be used and tuned according to the application. An off-the-shelf version of the algorithm is used in the thesis, however it remains possible to fine tune the parameters in order to achieve sharper segmentation. Segmentation is performed on a static image to separate the foreground from the background. The aim of performing segmentation is isolate the desired object to be grasped from surrounding clutter. From the point of view of the user, which will be either the elderly or people suffering from muscular impairments, the amount of user involvement for selecting the object to be grasped has to be minimal. In GrabCut the user has to select 2 diagonally opposite points with the desired object encapsulated in the bounding box created by the 2 points. This allows for easy interaction between the user and the system which is an important consideration behind the system design. In the system developed for this thesis image segmentation needs to be performed only when an object is being selected for grasping.

For the system developed in this thesis the aim is to make the selection of objects easy for the user. Since it is difficult for the user to pinpoint exactly the object they want, the segmentation

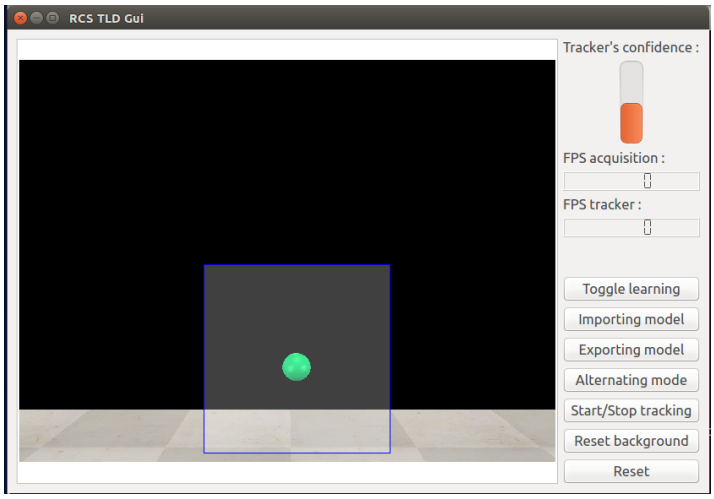

Figure 3-1: Bounding box selected in interface

algorithm is indented to make it simpler. From the perspective of motion planning it is desired to have an exact target for generating a trajectory.

In order to determine the target coordinates for motion planning the x,y and z coordinates are required. From the RGB image it is possible to find the x and y coordinates, a simple transformation from the camera frame of reference to the base frame of reference of the manipulator. In order to find the depth information the center of the bounding box is taken. This is achieved by using the x and y coordinates of the center of the bounding box to extract the depth information from the point cloud data. Figure: 3-1 shows the interface with of the tracking algorithm window. The user can select a bounding box around the object they want to grasp. Figure: 3-1 shows a sphere being selected by the user, on the right side the tracker confidence can be seen along with available options. It can be seen that the tracker confidence is not very high and that a lot of ambient area is selected to be tracked by the user, this can cause accumulation of errors and mar the performance of the tracker.

Figure 3-2 shows the segmentation algorithm working in the presence of a textured background object. The sphere in front is separated from its background object, this can make the difference between using the tracking algorithm to identify the object to be grasped by the manipulator. If the background is not segmented from the foreground the depth information received from the depth camera may be misleading.

Figure: 3-1 shows the reduced bounding box after segmentation is performed. It can be seen that the bounding box is the smallest possible for contour of the selected object. The tracker confidence is higher in comparison with Figure: 3-3. Segmentation can be difficult in the presence of background objects, Figure : 3-2 shows the performance of GrabCut algorithm for such a case. The performance of GrabCut is satisfactory and it improves the tracker confidence as well, next it's impact on tracking is investigated.

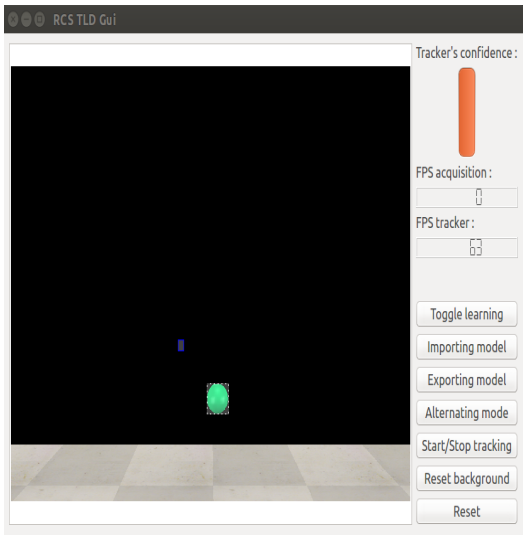

Figure 3-2: Minimized bounding box after segmentation

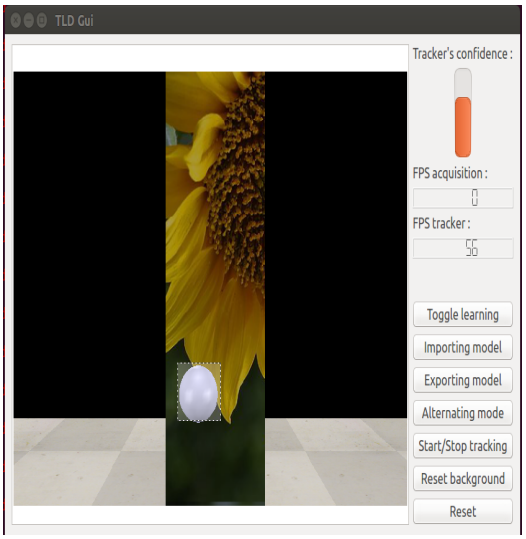

Figure 3-3: Segmentation in presence of background texture

3-2 Tracking

Keeping a track on the desired target object in the manipulation task is important due to the dynamic nature of the environment. It is possible that the object is occluded or moved during the manipulation action. Keeping a track on the position of the desired object adds to the dynamic nature of the robotic system as it allows for replanning or canceling the manipulation motion.

After a rigorous literature study of available state of the art tracking algorithms Tracking-Learning-Detection (TLD) [12] algorithm was selected. The basic algorithm can be used off-the-shelf. For the thesis, the parameters are not changed.

3-2-1 Testing

For testing the implementation and efficiency of the tracking algorithm a simulated environment was created in V-rep. Most of the simulation experiments are conducted in V-rep [22] , however it is explained in detail in later sections. The Robotic Operating System (ROS) implementation of TLD can be found in [34]. TLD can be used off-the-shelf, however by including segmentation as an additional step the performance may be improved.

3-2-2 Results

In Figure: 3-4 the layout of the experiment can be seen. The green ball is moved slowly to the location of the red ball, the obstacle is to challenge the tracking algorithm. It can be seen that by including the segmentation module in the tracking algorithm the performance can be improved.

Figure: 3-5 shows that without segmentation the tracker fails as the object changes its position. This can be attributed to the bounding box drawn around the green ball initially. The

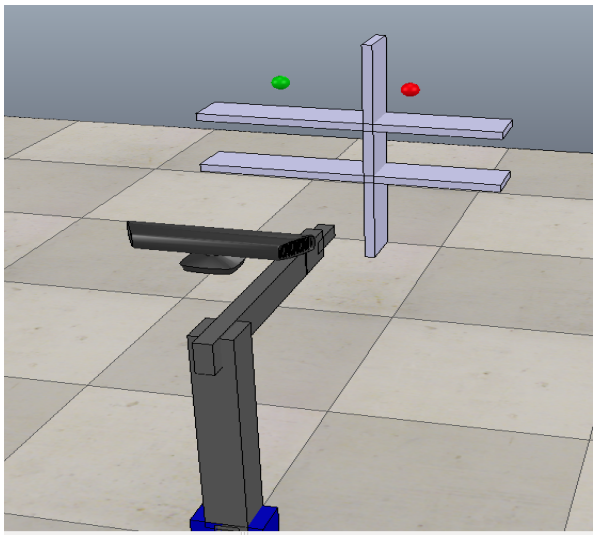

Figure 3-4: Experiment for testing impact of segmentation

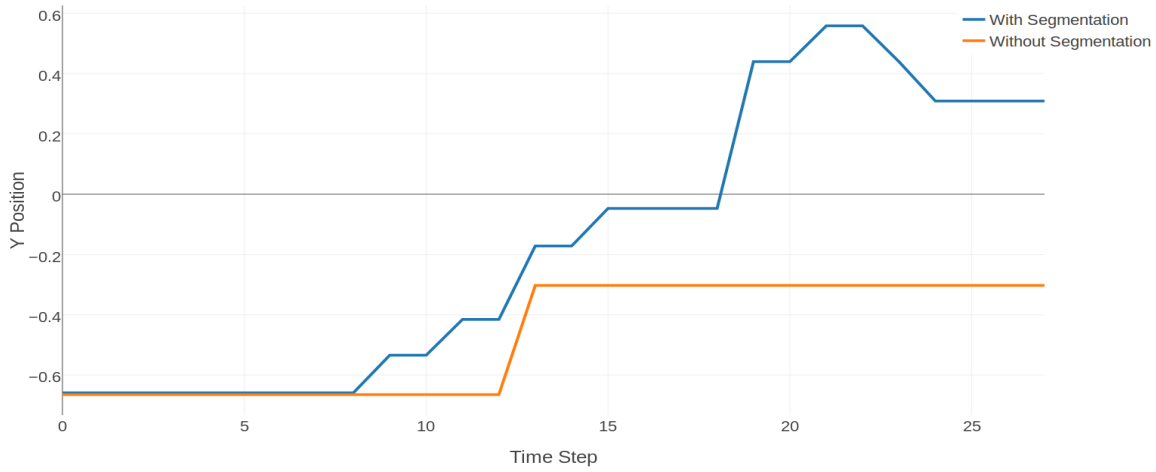

Figure 3-5: Tracker results

bounding box contains some background information, which in a cluttered environment will make it difficult to separate the desired object from the background. During the motion of the ball from its initial position to the red ball the tracker is lost, it is eventually regained once the ball is moved further away from the obstacle after 12 seconds. It can be seen that the tracker fails to follow the object to its final position. Figure: 3-6 shows the tracker is lost when near the obstacle.

Figure: 3-5 also shows the impact of segmentation in the tracking algorithm. The tracker follows the change in position of the object to the final position. From time step 15 to 20 the performance is low due to the presence of the obstacle in the background, however the tracker locates the object again and follows it to the goal position. It can be seen in Figure: 3-7 the bounding box is present even near the obstacle.

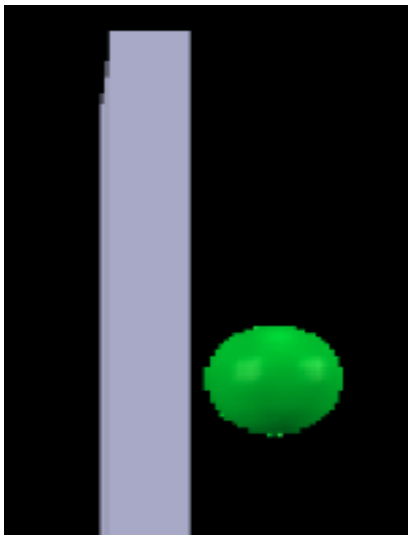

Figure 3-6: Tracker lost without segmentation

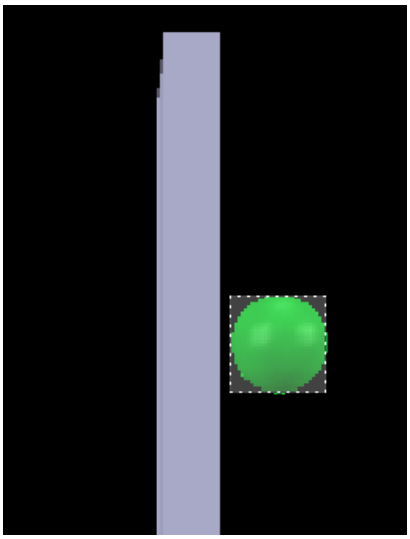

Figure 3-7: Tracker present with segmentation

3-2-3 Additional Changes

In order to facilitate the user to determine where the object is to be placed a bounding box for the desired location of placement can also be selected by the user. Although it gives the 3D coordinates of the bounding box for that moment, validity of the point can be checked in future work. In the images shown in this chapter the place bounding box is the small blue box, it is not used in motion planning experiments for those cases.

3-3 Discussion

In this chapter GrabCut segmentation algorithm was inspected and its impact on tracking was evaluated. It was observed that addition of segmentation improves the tracking performance. Tracking tends to fail in cluttered spaces and there is heavy reliance on the user to draw the bounding box accurately. In order to make the system more usable and improve the performance an additional segmentation function was added which proved to improve the tracking. Chapter: 4 looks into the motion planning module, as shown in the overview (Figure: 2-1) the coordinates selected from the vision system are passed to the state machine. The next chapter will look into the modules present in the state machine and how they impact the over all success of motion planning task.

Chapter 4

Motion Planning

In the previous chapter the components of the vision module were examined and the results were documented. In this chapter the motion planning module will be examined. It was seen that segmentation assisted tracking has advantages. Motion planning constitutes of formulating a path to the intended target pose while following predetermined constraints and avoiding obstacles in the environment. The robotic system is intended to be used in a domestic setting which is highly unconstrained. As mentioned in the use cases in chapter 1, the user should be able to pick and place objects placed in the environment. To achieve this a collision free path needs to be generated that follows the constraints imposed upon the path trajectory. The constraints can be simple requirements such as not spilling the contents of the object being manipulated, however in the context of motion planning this requires generation of a trajectory with an orientation on the end effector. Furthermore, the planned trajectory must be smooth and not erratic. These additional considerations along with the dynamic environment make it a challenging task to find a trajectory. This chapter looks into creating a simulated environment that recreates the dynamic nature of the environment and acts as a testbed for motion planning algorithms. In order to determine the usability of motion planning algorithms benchmarking experiments are performed with different setups that resemble the conditions in which the robotic system is intended to be used. Different motion planning libraries were explored that contain several algorithms that solve the motion planning problem. Combination of motion planning algorithms from different libraries are also tested to extract the benefits of different methodologies.

4-1 Representation of The Robot

In order to recreate the dynamic nature of a user's home V-rep software is used. There are several options for simulation software such as Gazebo [29], however according to [35] , there are certain inherent advantages of using V-rep such as compatibility with ROS, ease of adding environment objects such as glass, cabinet, sensors etc. A bare-bone version of the robotic arm is created by creating a simple Universal Robot Description Format (URDF) file. This

allows to import the model into V-rep environment. The model is built in accordance with the dimensions of the real hardware, the exact contours are replaced by cuboids. The cuboids are slightly bigger than the actual dimensions of the hardware, this is useful in having a higher clearance with the obstacles during motion planning for the actual arm. For experiments this version of the robotic arm is sufficient as it mimics the kinematics of the actual hardware and also provides greater clearance from obstacles.

4-2 Motion Planning using Moveit!

Moveit! [21] is an open source platform designed for development of solutions for motion planning problems. The architecture of Moveit! can be seen in Figure:4-1

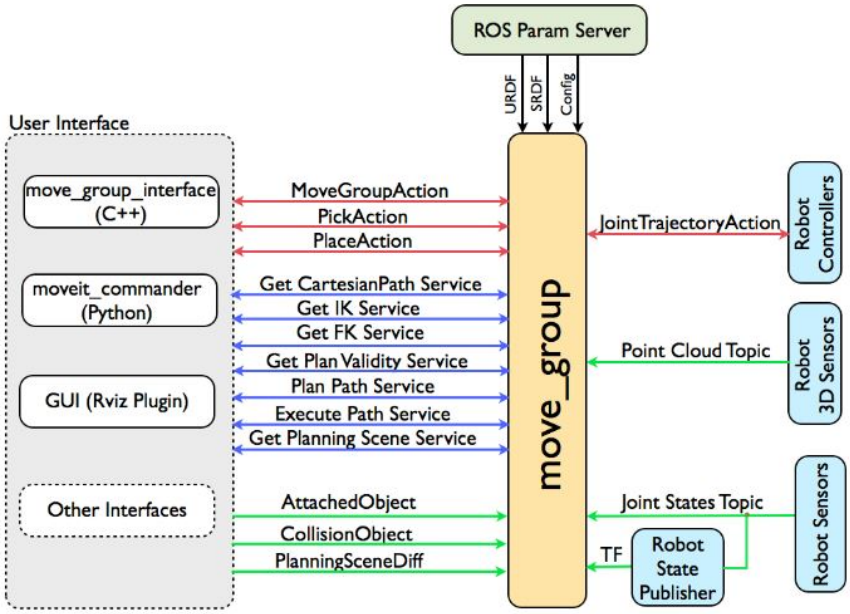

Figure 4-1: Move group

The `move_group` node is the primary node that acts as a base for retrieval and transmission of data to axillary nodes. The user can execute actions or services by interacting with the `move_group` node. This node can be interfaced by the user interfacing via C++, python or GUI.

The robot description is stored in the URDF file format. It contains information about the links and joints. The Semantic Robot Description Format (SRDF) file describes which joints are part of the planning group, it specifies the end effector link and contains information about the kinematic chain formed by the links. All this information is uploaded into ROS parameter server which `move_group` node accesses.

In order to communicate with the robot for motion planning the following information is needed:

- Joint State Information: The current position of each joint. In case of more than one planning group multiple joint state publishers can be used.
- Transform Information: This allows for updating the position of each link using the joint state information and the robot description in URDF.
- Planning Scene: This prepares the scene for motion planning by adding information from the sensors. Information about the environment and the robot is updated here.

Moveit! communicates with motion planners using a plugin interface. This allows for different motion planning libraries to be uploaded and used. While generating the trajectory collision and constraints are checked. Moveit! allows for setting kinematic constraints:

- Position constraints : Restrict the position of a link to lie within a region of space.
- Orientation constraints : Restrict the orientation of a link to lie within specified roll, pitch or yaw limits.
- Visibility constraints : Restrict a point on a link to lie within the visibility cone for a particular sensor.
- Joint constraints : Restrict a joint to lie between two values.
- User-specified constraints : Custom constraints with a user-defined callback.

The trajectory that is generated by the `move_group` node utilizes the joint limits mentioned in the URDF file. From the motion planning request to the response planning request adaptors are utilized. The adaptors are used for pre and post processing of the request and response respectively. The default motion planning adaptors are as follows:

- FixStartStateBounds: This sets the joint position to be in the allowable bounds in case they are out of bounds in the starting state. This is useful in case the robot joints are slightly out of their allowed range which can happen with robotic setups.
- FixWorkSpaceBounds: A default workspace of 10m X 10m X 10m cube is set in case the workspace is not mentioned explicitly.
- FixStartStateCollision: In case the links are in starting state are in collision this adapter changes the positions of the links with a pre mentioned range such that there is no collision.
- FixStartStatePathConstraints: In the event that the start pose is in violation of the constraints imposed on the path this adapter attempts to find a path from the start state to one that obeys the constraints.
- AddTimeParameterization: Velocity and acceleration constraints are added to generated trajectories.

The planning scene monitor takes information from the joint states, sensor information and the inputs from the planning scene. Shown in Figure: 4-3

For inverse kinematics calculations different plugins can be used. For collision checking Flexible Collision Library (FCL) [36] is used. Due to the plugin based architecture it is possible to change the default plugins.

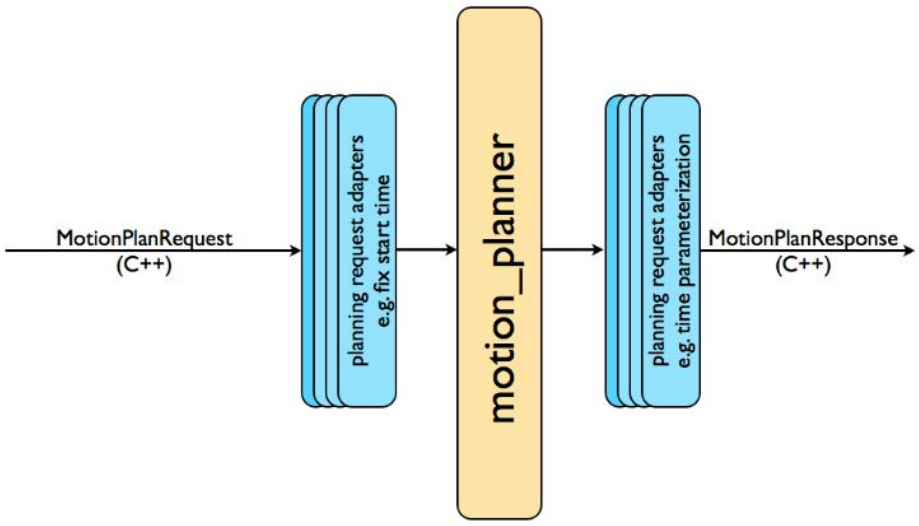

Figure 4-2: Planning pipeline

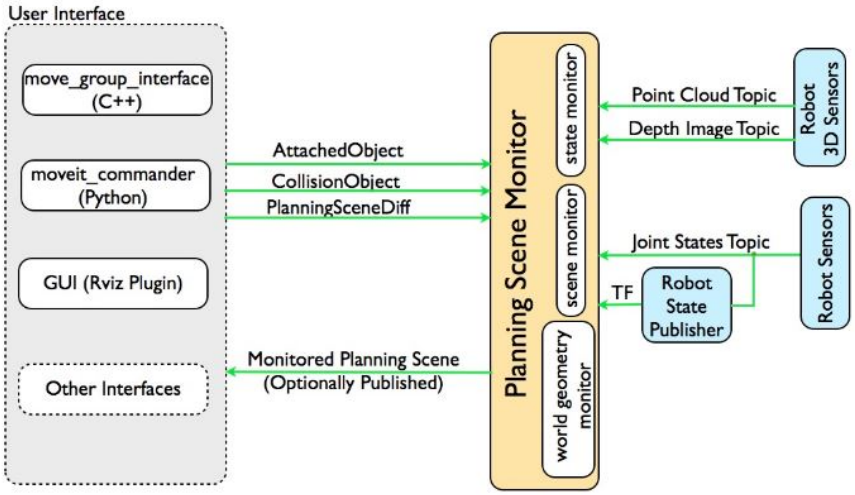

Figure 4-3: Planning scene monitor

For reducing the computational load for internal computations the depth information is presented as octomap [37] instead of pointcloud data. Figure: 4-4 shows the scene and Figure: 4-5 shows the octomap representation that is used for internal computations. The stick figure representation shows the transforms published between each link.

In Figure: 4-5 the octomap representation can be seen. The individual block is broken into eight parts or leaves. If neighbouring points are occupied then the voxel is occupied. The resolution and range of the octomap can be modified according to the application. For the thesis the range does not have to be large, the resolution can be adjusted.

Different motion planning libraries perform obstacle avoidance differently. Inside OMPL there are many different motion planning algorithms that perform obstacle avoidance differently.

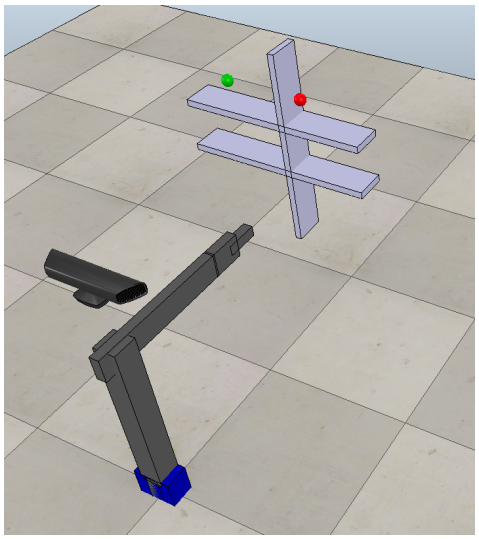

Figure 4-4: Representation in V-rep

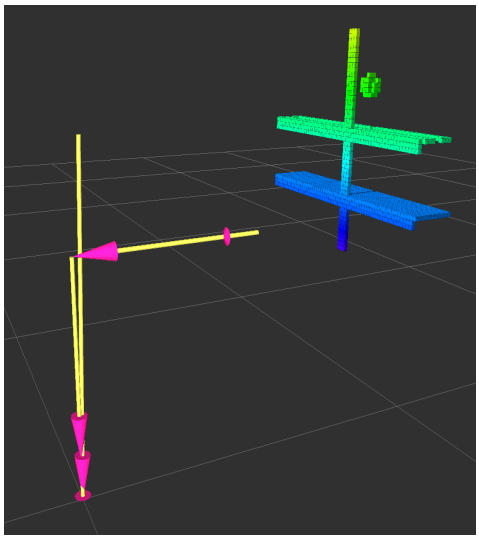

Figure 4-5: Octomap representation

Sampling based algorithms do not generate points on obstacles, combinatorial motion planners generate a road map, search based motion planners discretize the environment and use search based algorithms to find a collision free solution for the motion planning problem.

4-2-1 Trajectory Interpolation

The trajectories generated through Moveit! sometimes cannot be implemented directly on the actual robot due to trajectory points being far apart. It is also desirable to have the ability to manipulate the points in the trajectory for desirable smoothness in motion. An additional trajectory interpolation adaptor [38] has been added into the Moveit! framework for this thesis. The generated trajectories are re-sampled by adding a uniform time step between trajectories. A higher order spline smoother is used in joint space to re sample the points. For the thesis the sample duration is kept at 0.01. This low value increases the trajectory significantly thereby making the motion smoother. The controller on the robotic arm throws an error if the set target is beyond a certain threshold as the current controller attempts to move to the far away target in a quick motion. This is unsafe for the user and the environment and hence the trajectory interpolation is needed for interfacing with the hardware. Improving the controllers on the robotic arm remains to be an area of future research and is not handled in this thesis.

4-3 Motion Planning Libraries

A motion planning library contains the motion planning algorithm and means to interface it with ROS. Moveit! has a modular framework that allows it to interface with many motion planning libraries. In this thesis three motion planning libraries are explored and implemented for the system. The planners are tuned to a satisfactory level by exploring the parameters

empirically. The aim of the thesis is not to optimize the motion planning algorithms but to implement the libraries in a manner that they can be used for motion planning requests. Their performance is tested on different benchmarking cases to decide in which setting a motion planning library would be best to use.

In order to facilitate the motion planning libraries to be usable with Moveit! they have some files in common that are explained below:

- YAML Ain't Markup Language (YAML) file: The configuration parameters for the planning algorithm are stored here. These parameters are uploaded in the ROS parameter server and are accessed by Moveit!.
- Launch file: Contains a list of planning adaptors to use for the motion planning library. Parameters are uploaded into ROS parameter server.

4-3-1 OMPL

OMPL is the default motion planning library that Moveit! uses. There are several sampling based, combinatorial based motion planning algorithms in OMPL. An entire list of motion planners in OMPL is present in [39]

These algorithms are a mixture of sampling based, combinatorial, optimization and control algorithms. For the thesis however the aim will not be to obtain optimum tuning, only till the point that they can operate satisfactorily. The parameters that can be used to vary the performance of the motion planning algorithms are present in the YAML file. The YAML files are generated by default by Moveit! when the Moveit! setup assistant [40] is used. For OMPL motion planning libraries the planners come with a predefined set of configuration parameters that are set on initialization unless the default parameters are changed. These parameter values give satisfactory performance so that the planners can be used off the shelf. Although the values can be optimized either by using grid based or model based optimization it is not in the scope of this thesis. The parameters are however explored.

The main parameters of the OMPL motion planning algorithms are:

- **Range**: The length of motion that can be added in the motion tree. The length can impact the planning time for the algorithm.
- **Goal Bias**: If the algorithm knows the goal state with some probability it can select the goal state during sampling with greater ease. The value ranges from 0 to 1 and 0.05 is the default value selected by the developers at OMPL after rigorous experimentation.

The planners being considered are:

- Probabilistic Roadmap Method (PRM) (Probabilistic Roadmap Method): A multi query motion planning algorithm. This is a combinatorial planner that generates the map once that can be used for multiple queries. A road map is created that reflects the connectivity of the state space. The parameter that can be varied is `max_nearest_neighbors`. The default value is 10.

- Expansive Space Trees (EST) (Expansive Space Trees): A tree based motion planning algorithm that stresses on exploring the less explored parts of space. The parameter that can be varied are `range` and `goal_bias`. The `goal_bias` is set to 0.05 and the range is set by OMPL on initialization.
- Single-query Bi-directional Lazy collision checking planner (SBL) (Single-query Bi-directional Lazy collision checking planner): It is a bi-directional planner that is the tree starts from the starting and goal position, the lazy collision checking implies that validity of states is not checked in the algorithm, it has to be done separately. Once invalid path is found, it is removed. The `range` is set during initialization.
- Kinematic Planning by Interior-Exterior Cell Exploration (KPIECE) (Kinematic Planning by Interior-Exterior Cell Exploration): A tree based planner in which grids are generated at multiple levels to discritize the state space. The Bi-directional KPIECE (BKPIECE) uses two trees instead of one and Lazy Bi-directional KPIECE (LBKPIECE) uses a bidirectional lazy collision checking. The parameter that can be tuned is the range and goal bias. The default value of the goal bias is 0.05.
- Rapidly-exploring Random Trees (RRT) (Rapidly-exploring Random Trees): Tree based motion planning algorithm that uses random sampling to construct a solution. RRT Connect (RRTConnect) is a bidirectional variation of RRT. The parameter that can be varied is the range.

Sampling based motion planning algorithms can find a solution in a short planning time. One of the drawbacks of sampling based algorithms is that the generated trajectory can have random motions. There could be unexpected motions of joints which is not preferred in a domestic setting. In order to account for randomness of sampling based motion planning algorithms two more motion planning libraries are explored.

Constraint Handling

In many tasks it may be expected from the manipulator that it keeps the pose of the end effector constant. Picking up a glass of water from a table is a classic example for this kind of requirement. In such cases an orientation constraint is imposed onto the end-effector of the manipulator. The orientation constraints limit the sampleable region for motion planning, hence the planner tries to find a path that respects the orientation constraints imposed on it within certain bounds. The constraints are imposed using quaternions. Experiments were conducted simulating real world settings where such a constraint would be used. There were two tests conducted one with an obstacle between two poses and one without, in each case the end effector must maintain its initial pose. The tests were run three times for each case.

The experiment is shown in Figure: 4-6. The two spheres represent the two poses the end effector must reach from an initial position shown in the figure. The motion planning can hence be broken down into two tasks: first to reach the ball on the left and then plan a path to avoid the obstacle in the middle and reach the second ball while maintaining the orientation constraint. For the experiment without obstacles the results can be seen in Figure: 4-8. It can be seen that most OMPL planners suffer from poor performance in constraint handling. PRM, a combinatorial planner gave best performance in comparison with the other planners. In Figure: 4-8 and 4-6 0 is success and 1 is failure in motion planning.

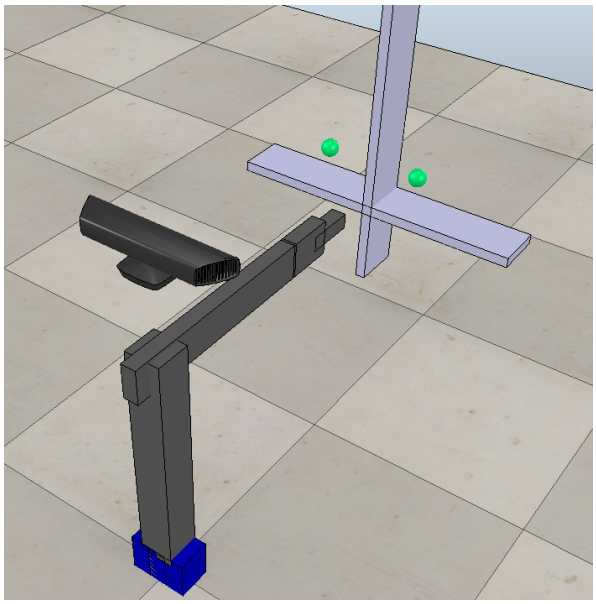

Figure 4-6: Orientation constraint with obstacle

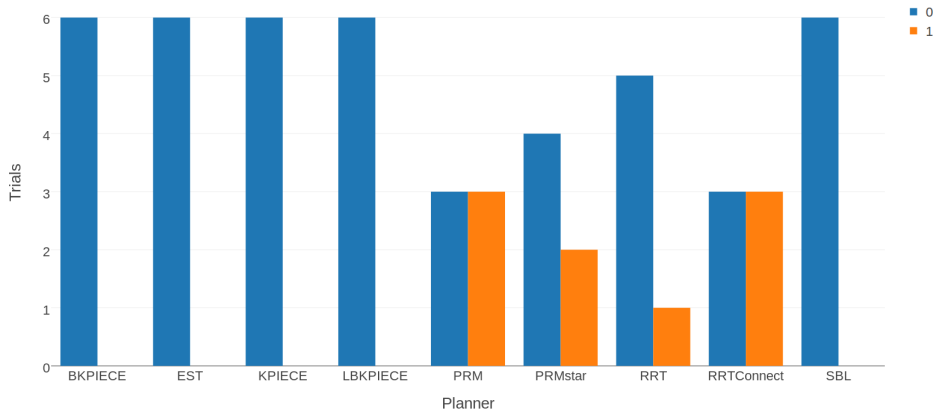

Figure 4-7: Orientation constraint with obstacle

Though some planners were successful in one of the motion planning tasks, it failed in the other. Most planners were able to reach only the first ball, after spending extensive amounts of time. In reality none of the planners were able to complete the motion planning tasks while maintaining the orientation constraints. This could be due to the planner configurations that were kept at default for the OMPL planners, but the aim of the thesis is not to tune each planner but to evaluate them with satisfactory operating parameters.

In the case of planning with an obstacle in between the two poses all planners failed to complete the motion planning task. The results shown in 4-7 show that all planners suffer to perform with orientation constraints.

It can be concluded that OMPL planners do not perform to a satisfactory level in the presence of orientation constraints for a setting that will be common in a domestic setting. This also

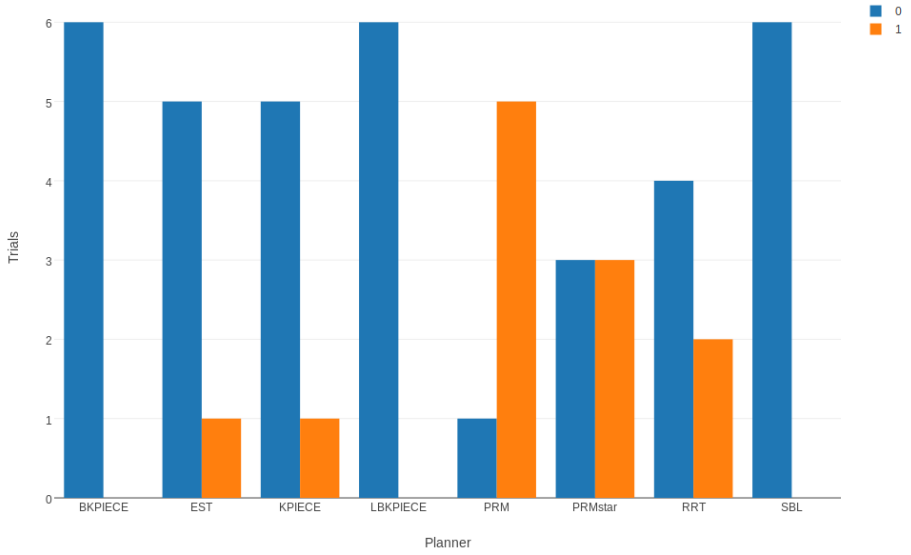

Figure 4-8: Orientation constraint without obstacle

shows the need for exploring other motion planning libraries which are done in the remainder of this chapter.

4-3-2 STOMP

The parameters of STOMP that can be altered are presented in the points below along with the analysis of the parameters and their impact on the motion planning. The aim here is to find a suitable value for parameters that allow STOMP to be used for regular motion planning tasks. Table: 4-1 shows the parameters for STOMP planning library.

- Trajectory duration: The maximum size of the trajectory that is allowed. Setting a high value for this can allow for a trajectory that avoids obstacles but also allows for a longer path in case there is no obstacle.
- Number of roll outs per iteration: STOMP finds a suitable trajectory by considering several noisy ones, this number sets the number of candidate noisy trajectories that will be considered. A higher value will offer a possibility of finding a better solution but at the cost of a higher planning time.

  Tuning: By setting a low value for this parameter solving the optimization problem becomes difficult, however setting it high will lead to a large planning time. It is preferable that the planning time is less than one second and the planner is capable of solving tougher motion planning problems as well. A simple motion planning experiment is conducted where the arm is expected to reach a target without the presence of obstacles (0.17,0.65,0.52)

Table 4-1: STOMP parameters

Parameter	Value	Description
Number of time steps	50	The greater this value is the more joint configurations will be considered per trajectory.
Max iterations	100	Setting a higher value for this parameter might allow for STOMP to find a solution around obstacles and in narrow environments
Number of roll outs per iteration	100	The Number of candidate noisy trajectories that will be considered
Noise coefficients	[0.1,0.1,0.1, 0.1,0.1,0.1]	The values are used to randomize the joint values during generation of candidate trajectories (noisy). These trajectories are fed into the optimization task.

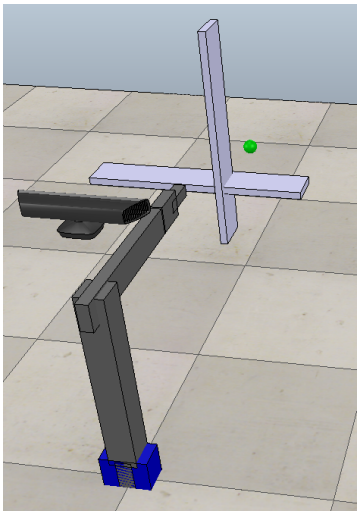

Figure 4-9: Experiment for varying stdev and decay

The spread for the planning time required over 20, 50 and 100 iterations for planning with STOMP for different values of roll outs per iteration are shown in Figure: 4-10, Figure: 4-11 and Figure: 4-12 respectively. It can be seen that increasing the value alters planning time proportionally. Since planning time under one second is preferable, the value for this parameter is selected as 100.

- Noise coefficients: This consists of 2 parameters : `stddev` and `decay`. The values for these parameters have to be specified for each joint. The values are used to randomize the joint values during generation of candidate trajectories (noisy). These trajectories are fed into the optimization task. The joints move rapidly if a higher value is set, however it might also cause random motions or longer trajectories than necessary.

  Tuning: These are important parameters for tuning of the STOMP planner. They specify the standard deviation and decay for the noisy trajectories that are generated. Since the values are set for individual joint finding an optimum value becomes very

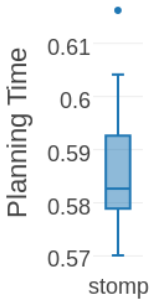

Figure 4-10: Rollout iteration=20

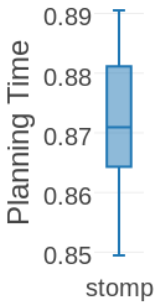

Figure 4-11: Rollout iteration=50

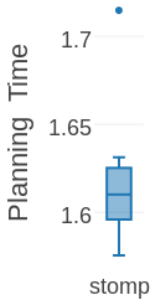

Figure 4-12: Rollout iteration=100

Figure 4-13: stddev, decay=0.1

Figure 4-14: stddev, decay=1

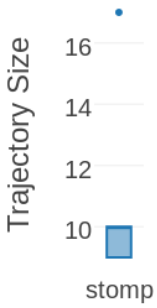

Figure 4-15: stddev, decay=2

problem specific. Since the robotic arm is for use in uncontrolled domestic environments tuning becomes very difficult. Setting a large value makes the movement to be large and not optimum since we are taking the first collision free trajectory that is generated.

For this thesis STOMP is considered as a local planner which will not require to do global obstacle avoidance hence it is expected to be used in areas where following constraints will be more important that planning around objects. In these cases a smaller trajectory is preferred. Joint 2,3 and 4 are the most crucial in determining the length of the trajectory in most motion planning problems for the 6 Degrees Of Freedom (DOF) arm hence these joint values will be altered to examine the changes.

Figure: 4-13, 4-14 and 4-15 show the results for planning 20 times for stdev and decay values of 0.1, 1 and 2 for joint 3, 4 and 5 respectively for the planning scene shown in

Table 4-2: Parameters for SBPL

Parameter	Value	Description
Initial epsilon	10	This value determines how good the first found trajectory should be in comparison to the optimum value. Setting a high value of this parameter should reduce the planning time but at the cost of the quality of the solution.
Final epsilon	5	The extent of optimality that is required for motion planning
Decrement epsilon	0.01	The value determines how much better the new solution should be as compared to the previous one
W_{cell}, W_{action}, W_{smooth}	[10,1,1]	The weights for the coefficients for the cost in SBPL. Low weights result in lower planning time.

Figure: 4-9.The green ball shows the target and the cuboids represent obstacles.

Working with constraints

In this section orientation constraint was imposed on STOMP planner to keep the end effector position same as the start pose. The same motion planning challenge as shown in 4-6 was used. In contrast to OMPL, STOMP was successful 100 % of the times for both scenes, with and without an obstacle in between. This shows favourable performance in comparison to OMPL for cases where orientation constraints have to be handled. This superior performance can be attributed to the cost function that takes the constraints in computations. STOMP can be used as a local planner for cases where constraint handling is important.

4-3-3 SBPL

The parameters present in the ROS package for SBPL can be seen in Table: 4-2. An insight into their tuning is provided below.

- Epsilon: The measure of how good the current solution is in comparison to the optimum solution. A high epsilon value indicates that a suboptimal solution will be found but in less time. A smaller value in turn will make the planner to find a more optimal solution at the expense of planning time. In the YAML file the initial and final values of the epsilon can be defined along with the rate of decrement of epsilon.
- Initial epsilon: The measure of quality of solution. This value determines how good the first found trajectory should be in comparison to the optimum value. Setting a high value of this parameter should reduce the planning time but at the cost of the quality of the solution.

  Tuning: A high value would reduce the planning time, since the found trajectory would be a valid one it may not be necessary to search an further. The main advantage of SBPL

over other planning libraries is the repeatability of motion but the motion must not be very erratic, hence a very high value of this parameter cannot be selected. Through experiments an initial value of 10 was found to be satisfactory.

- Final epsilon: This parameter determines the extent of optimality that is required for the motion planning. The planner will not look any further than this value of quality of solution.

  Tuning: The value can be set low, however by setting it low the planning algorithm will spend more time trying to find a better solution. It is possible that a better solution may be found but at the expense of planning time.

- Decrement epsilon: It determines the rate at which a better solution is searched for. This is the value that is reduced from the epsilon value used in previous iteration that found a successful solution.

  Tuning: By setting a small value the planning time increases as the planer iterates several times, slowly trying to find a better solution. Setting a larger value in turn will reduce the planning time but taking bigger jumps leaves the possibility of not finding the best solution possible in the range of epsilon set. The value being used is 0.01

Constraint Handling

The same motion planning task was tested with Anytime Repairing A* (ARA*) of SBPL, however it was not successful in any part of the motion planning tasks shown in Figure: 4-6 . This can be attributed due to the limitations of the planner or due to the planner configurations.

4-4 Experimental Setup

In the previous section each planning library was examined and subjected to experiments in order to evaluate the performance under simple orientation constraints. For the thesis a system was developed that can switch between motion planning libraries after completing a motion planning task. The section focused on tuning the planner such that they can perform manipulation in everyday setups. Finding optimal parameters for a highly unregulated environment such as a house is a daunting task and not addressed in this thesis. From the experiments performed it can be concluded that STOMP performs best for handling orientation constraints hence STOMP can be employed for the more clustered regions in the motion planning task. The contextual awareness section examines the performance of the planners without constraints but in areas of varying clearances and positions.

The tuned planners are tested on the robotic arm to test their performance. Transitioning from simulation to real hardware required some changes. Figure: 4-16 shows the new experimental setup. The individual components are:

- A (Teaching Pendent) : An interface that allows manual movement of the arm, it also acts as an emergency stop.
- B (Robotic Arm) : The 6 DOF robotic arm, it is clamped to the table for stability.

Figure 4-16: Experimental Setup

- C (3D Camera) : The 3D camera attached on to a mount.
- D (Cup): The object that is to be grasped.

It can be noticed that the camera is placed on the table and not above the robotic arm as shown in Figure: 2-5. The change in this position was reflected in the URDF as well so it can be taken into account for motion planning. STOMP algorithm was selected for solving the motion planning problem. The setup incorporates all the components discussed in previous chapters. The vision system is used to identify the position of the cup, the selected bounding box is segmented and the 3D coordinates are passed to the motion planning module. On receiving the coordinates the STOMP algorithm finds a solution and the trajectory is sent to the driver for interfacing with the hardware. CAN protocol is used to send the joint positions and receives the joint states. There are several sources of noise that impact the efficiency such as noise for 3D camera due to ambient light. The exact origin of the coordinate system for the robotic arm is selected approximately by testing, an offset of three centimetre is present.

Figure: 4-17,4-18,4-19,4-20,4-21,4-22 show the motion planning conducted using STOMP algorithm for the robotic arm. In Figure: 4-17 the cup is selected using the vision system as the target, in Figure: 4-18 the motion plan is computed and trajectory execution reaches the Pre-Grasp pose for the cup. Figure: 4-19 shows the arm reaching towards the final pose by following a simple path. Figure: 4-20 shows the Pre-Grasp pose for a case with an obstacle present. The initial pose is not very different from the case without an obstacle, however Figure: 4-21 shows the change in the trajectory to avoid

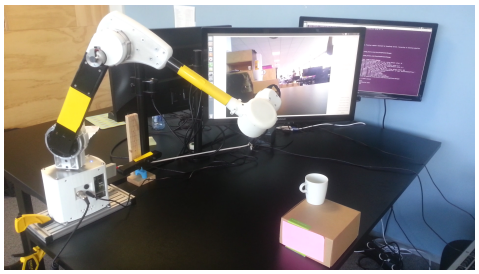

Figure 4-17: No obstacle

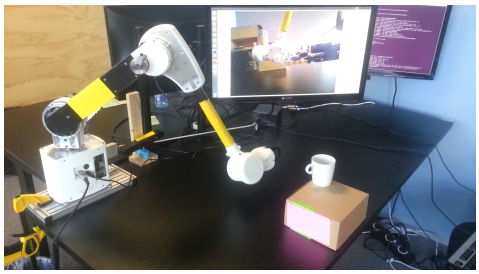

Figure 4-18: Pre-Grasp pose

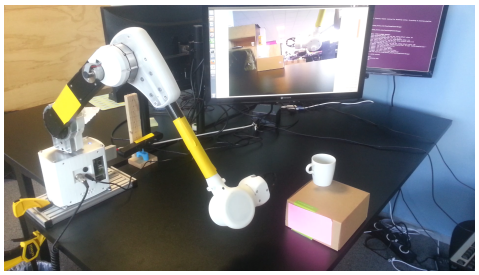

Figure 4-19: Final pose

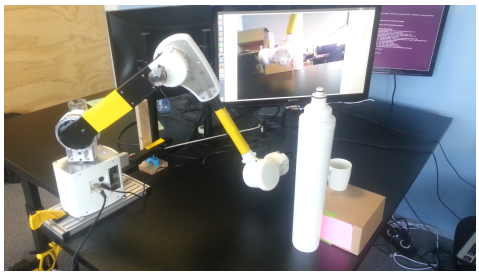

Figure 4-20: One obstacle

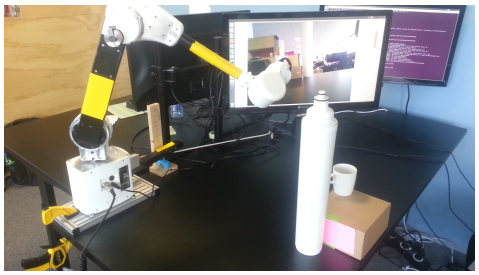

Figure 4-21: Avoiding obstacle

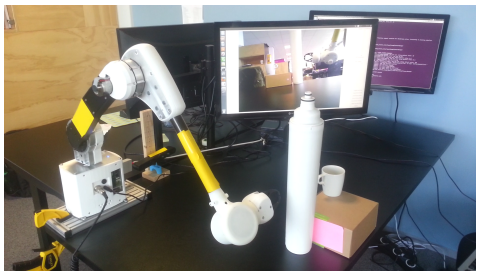

Figure 4-22: Final pose

the obstacle. Figure: 4-22 finally shows the final pose is successfully reached, hence proving the capability of the system to plan around obstacles.

Despite the sources of errors for the system the performance is acceptable. The error in position the end effector reaches is within 1-2 cm. This range of error is acceptable, the setup shown in Figure: 4-16 is to test the working of the entire system, fixing the position of the camera and arm will improve the efficiency. To minimize the effects of ambient light for the 3D camera can be achieved used filters, this is however beyond the scope of the thesis and remains to be a future area of research. Even when the position of the camera is changed the vision system is able to track its motion and STOMP is able to plan a path towards it. The next section explores the genetic search algorithm implemented for the thesis.

4-5 Genetic Search

Inverse kinematic problems can be solved using genetic search as shown in [41]. The algorithm mentioned in [41] converts the inverse kinematics problem into a minimization problem and then employs genetic algorithm to find all the global minimums of the

Table 4-3: The IK Algorithm

Step	Description
1	Randomly Initialize the Customer Population Randomly Initialize the Businessman Population Initialize d min with d min start
2	Customers Raw Fitness Value Calculation Businessmen Raw Fitness Value Calculation Assignment of Customers to the closest Businessman Customers Shared Fitness Value Calculation
3	Forming the customers parent pool by Tournament Selection Adding the fittest businessmen to the pool
4	Customer Crossover
5	Businessmen Imprint
6	Updating d min
7	If the termination criterion is not reached return step 2

problem. The algorithm essentially attempts to find the joint angles that produce the least position and orientation error for the end effector from the target values. In genetic algorithms the fitness of an agent decides its potential for crossover with other agents to form the next generation. For the case of manipulation this fitness is the measure of error between the position of orientation of the end effector from its target value. Essentially only the solutions with least error are selected for crossover with the next generation.

4-5-1 Algorithm

The algorithm used can be seen in Table: 4-3. From: [42] the definitions of customer and businessmen population can be taken as:

- Customer Population: The common population of the solution candidates, searching for areas with a high fitness value through selection and recombination.
- Businessman Population: A population of solution candidates as well, though the businessmen interact with the customer population to find those locations which yields them the highest payoff. Their fitness function enables them to place niches at highly fit regions of the search space

Where:

- Initialization: After the customer and businessman population are randomly initialized the d_{min} is calculated as a function of degree of freedom (6 for this case).
- Fitness Value Calculation: The fitness depends on the error between the orientation and position of the end effector in comparison to the target pose. For the position error Eucledean norm is used and for the orientation error Euler angles are used.

The fitness function is the sum of the errors with a weight for each term. The distance error:

$$E_p = ||P_{desired} - P_{ind}|| \tag{4-1}$$

Orientation error:

$$E_o = min(||O_d - \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{bmatrix}||, ||O_d - \begin{bmatrix} \alpha_i + \pi \\ -\beta_i \\ \gamma_i - \pi \end{bmatrix}||) \tag{4-2}$$

Where:

$$O_d = \begin{bmatrix} \alpha_d \\ \beta_d \\ \gamma_d \end{bmatrix} \tag{4-3}$$

where O_d is the desired orientation in Euler angles and α_i ,β_i ,γ_i are the Euler angles of the individual. The fitness function is hence:

$$F.F = w_p F_p + w_o E_o \tag{4-4}$$

where w_p and w_o are weights.

- Selection: A fixed value of individuals are selected at random from the customer pool, the customer with the least error is transferred to the parent pool. Likewise the businessman with the best fitness are selected and added to the parent pool.
- Customer Crossover: Generations of customers produced from the parent pool.
- Businessman imprint: The businessman population is individually compared with random individuals in the parent pool. If the contender is an improvement over the businessman and is d_{min} away from other businessmen it will replace the original.
- Updating d_{min}: The value is reduced after every successful iteration. The lowest value depends on the number of maximum iterations.

In order to determine multiple solutions using inverse kinematics for a motion planning request a genetic algorithm is used. This is desirable when an exact solution is not possible for a motion planning problem due to obstacles or confined spaces. The genetic search algorithm will present multiple solutions that are closest to the desired target. The businessmen population is a population of solution candidates as well, though the businessmen interact with the customer population to find those locations which yields them the highest payoff. Their fitness function enables them to place niches at highly fit regions of the search space

From the solution generated by the genetic search algorithm it is possible to switch to a local planner such as STOMP or even a conventional local controller that can navigate towards the target. This opens up a lot of possibilities for solving the motion planning problem in complex cases.

This algorithm can also be used to check if the target point is within the workspace of the robotic manipulator.

The algorithm has been added into the Moveit! framework for this thesis and hence its parameters can be adjusted from the Application programming interface (API). The parameters to set in genetic search is the distance function. This function determines if

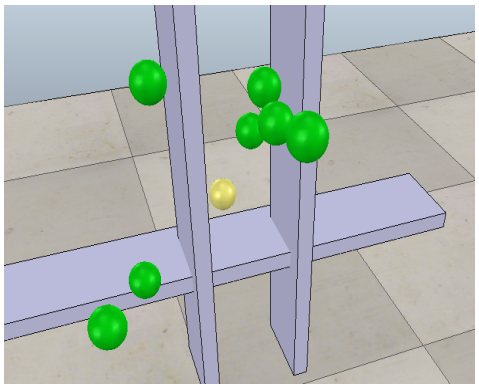

Figure 4-23: Genetic Search: Scene 1

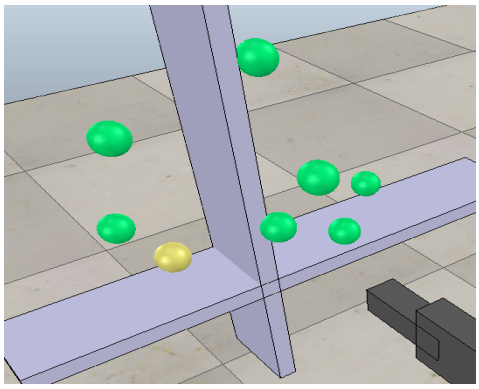

Figure 4-24: Genetic Search: Scene 2

an existing state is valid or not. Currently the distance function uses a threshold value to determine if the found solution is satisfactorily close to the target pose or not.

The aim of integrating a genetic search into the motion planning task is to have the option of switching planners. Motivation for this was taken from [43]. Whereas only sampling based motion planners are selected in [43] it is possible to have combination of search based, STOMP or sampling based algorithms. In the thesis a combination of motion planners are also explored. The task then becomes to find an appropriate global planner and a local planner. The benchmarking results of contextual awareness can be used to determine the best global planner. For the local planner STOMP is a strong candidate. In [43] CHOMP is used which is also a trajectory optimization planner but suffers from the limitation of using gradients which leaves it open to local minima. The following section evaluates the impact of using a genetic search algorithm in the motion planning task.

4-5-2 Experimental Results

This section explores scenarios where the target pose has obstacles in the path. In these situations a direct motion planning approach may fail due to absence of valid sampleable states if using an OMPL motion planner. For a local planner STOMP is used. In future the local planner can also be replaced for a conventional controller that performs tracking or grasping. Figure: 4-23 shows a scene that was created for testing the genetic search algorithm. The yellow ball represents the desired goal state, this state was not achievable due to low clearance, the goal here is to find coordinates that are within a certain threshold of the final state that is reachable. The green balls represent the states that were generated by the genetic search algorithm.

For the results shown in Table: 4-4 and 4-5 the values are in metres and the origin is at the base of the manipulator. The target coordinates for scene 1 are : [0, 0.839, 0.538] and for scene 2 are: [-0.1, 0.7, 0.55]. All the coordinates are in metres, the origin is the base of the manipulator.

From the generated points by the genetic search algorithm there was at least one point that was reachable. For scene 1 it was: [0.0151427, 0.685677, 0.688521] and for scene 2

Table 4-4: Genetic Search for Scene 1, all values are in metres

x	y	z
-0.14257	0.701468	0.535102
-0.0935237	0.735329	0.718481
-0.00165319	0.839955	0.538005
-0.124452	0.724404	0.535856
0.0244106	0.745429	0.693847
0.0151427	0.685677	0.688521
0.00914489	0.598753	0.718485

Table 4-5: Genetic Search for Scene 2, all values are in metres

x	y	z
-0.0932508	0.787615	0.552605
-0.0276968	0.569799	0.663068
-0.144627	0.69165	0.694951
-0.0363922	0.63761	0.569216
-0.0261918	0.644098	0.760538
-0.151676	0.745983	0.559448
0.0528164	0.66238	0.508246

it was:[-0.0276968, 0.569799, 0.663068]. These points serve as waypoints for completing the motion planning task. Though it can be seen that not all points are ideal for a mid way point for motion planning but this can be attributed to the distance function that is being used in the algorithm. The distance function evaluates if state generated by the genetic search algorithm is valid or not. Currently the distance function measures the distance of the generated state from the goal state and declares it as valid if it is within a certain threshold of the final state. This explains why some points are behind the goal state, however a distance function can be selected that takes into account the relative position of the robot with the goal state. The aim of this section was to first implement a genetic search algorithm that uses Inverse Kinematics (IK) of the robot to generate states for a scene where the goal state is very difficult to reach and then prove that at least one of the states is reachable using the current motion planners. For both cases there were at least one point was reachable via a global motion planner, though the success rate can be improved by using an advanced distance function.

4-6 Contextual Awareness

OMPL, STOMP and SBPL provide general solutions for any motion planning task. However, the solutions may not be suitable to implement on the robotic arm. Sampling based algorithms can generate trajectories that follow random paths, alternatively SBPL and STOMP may take too long to find a solution. It is important to evaluate the performance in cases that may be presented in homes offline so a quick decision can be made in real time. It is a daunting task to predict all the situations that may emerge in an unconstrained environment, however certain markers can be used to classify motion planning problems. In this thesis clearance to obstacles and relative position of the target pose from the base of the robotic arm are considered as the markers. These markers can be extracted from the RGB and depth information which is received from the 3D camera.

First, offline simulations were performed for different clearance in x and z direction of the target pose and position compared to the robotic arm. Fuzzy sets were created to classify the positions. The range of sets for clearance are shown in Table: 4-6. It is important to determine evaluation criteria that makes one algorithm performance more preferable over another. Success rate, trajectory length and planning time were

Table 4-6: Sets for tests, all values are in metres

Clearance	Low	Medium	High
X	0.1	0.3	>0.3
Z	0.1	0.3	>0.3

taken as the performance indicators. The arguments for the choice of these performance indicators can be summarised as:

- A high success rate is an obvious sign that a planner is suitable for a motion planning task, however it may be unsuitable if it follows a random trajectory or has high planning time.
- the arm is to be mounted on a service robot the trajectories must use as little power as possible, a smaller trajectory will achieve this.
- Repeatability of motion for a task makes the movement easy to predict and may make the user more comfortable to use the system. This can be seen in the variation of trajectory length for several executions of the same task.
- A smaller planning time ensures that the user does not have to wait for a long time before the arm starts to move.

There are three clearance sets for x and z direction. The position relative to the arm can be in classified as: front , left or right and the height can be either in front of the end effector (0.5 m above ground) or above it (0.5-0.8 m above ground). The decision to make these sets was motivated by the area viewable by the camera and expected areas of use. The camera cannot see the floor in its current position in simulation, also the aim here is to prove that it is possible to identify algorithms that may perform better in situations identified by the markers. The user is expected to reach for things in shelves, tables, counter tops etc. All those scenarios are covered under combinations of clearance and relative position.

Through experiments it was concluded that the results for the target pose being to the left or the right were not very different. This can be attributed to the fact that motion planning primarily depends on the clearance to the target pose in the absence of any constraints. Hence the results are presented only for cases where the target pose is to the center or the right. There are other permutations that are not considered in the experiments since the aim is not to consolidate the best planner for every situation, which would be a daunting task due to the unregulated nature of the environment. The aim is to establish that some planners perform better than others on the basis of planning time, success rate and trajectory size.

Benchmarking experiments were run for all permutations of the environmental markers discussed above and each experiment was executed 50 times. The goal here is to examine the variation in performance that may be seen overtime. In future research the impact of changing the position of the target depth will be explored.

4-6-1 Evaluation of results

In this section the results of the experiments are documented and the best planner for each experiment is selected. The criteria of selecting the planner depends on the algorithm having high success rate, small trajectory size and low planning time. In cases where there is confusion between planners the one with the higher success rate will be selected as the aim of this module is to increase the success rate for motion planning problems. Table: 4-7 shows the permutations of experiments that were conducted in V-rep environment for different x and y clearance and position. The results are then shown in the section after that. Finally the conclusion of the experiments are discussed. For the success rate graphs 1 is a success and 0 is failure in motion planning.

From experiments a few planners emerge with better results as compared to others. Few of the representative cases for the contextual awareness are presented in this section

- For case 12 we can see from Figures: 4-25, 4-27 and 4-26 RRTConnect has the highest success rate, furthermore its trajectory size is comparable to ones found by other algorithms. It is interesting to notice that STOMP found the smallest trajectory but at the expense of a high planning time. For this case RRTConnect is selected.
- For case 16 we can see from Figures: 4-28, 4-30 and 4-29 there is ambiguity for selection of planning algorithm. PRMStar has the highest rate of success but its planning time is high. After PRMStar RRTConnect and SBPL have the highest success rate. RRTConnect has a small planning time but the planned trajectory has a high variation, this is indicative of the randomness of sampling based algorithms. SBPL however has a small and consistent trajectory. For this case SBPL is selected.
- For case 18 we can see from Figures: 4-31, 4-33 and 4-32 SBPL is preferred again due to its smaller trajectory size and the low planning time. In this case STOMP can be selected as well since planning time is under a second. Sampling based algorithms perform well as well with only a few outliers. SBPL is selected for this case.
- For case 27 we can see from Figures: 4-34, 4-36 and 4-35 All the motion planing algorithms can be selected for this case as they all meet the requirements except for STOMP and KPIECE.
- For case 32 we can see from Figures: 4-37, 4-39 and 4-38 STOMP is the only algorithm that could solve the motion planning problem. Hence STOMP is selected for this case.

Through the benchmarking experiments performed it is evident that some algorithms perform better than others for varying clearance and position. For cases 9 and 10 that only differ in the position of the target pose different planners perform better, same can be seen for cases 11-12, 21-22 and 31-32. When cases 33 to 36 are examined it can be seen that in cases of high clearances RRTConnect performed best. Hence for cases where there is high clearance RRTConnect can be the default planner. For certain cases where the target is top-right with high x clearance and medium z clearance. SBPL emerged as the planner that performed best for cases of varying clearance. It does not imply that SBPL was the only planner able to solve the motion planning challenge but

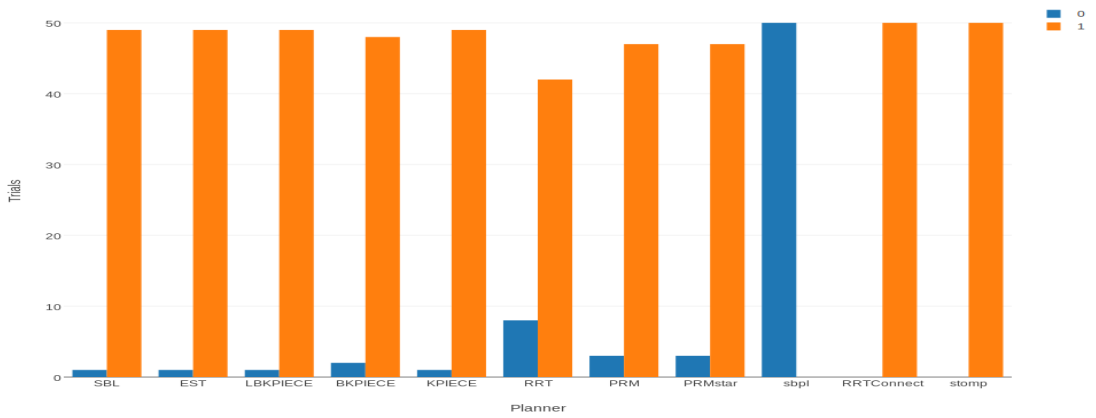

Figure 4-25: Success for 50 trials for case 12

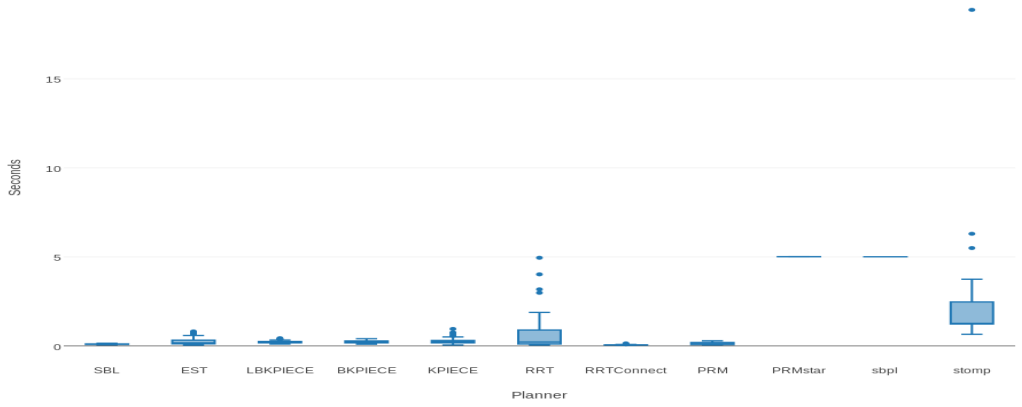

Figure 4-26: Planning Time for case 12

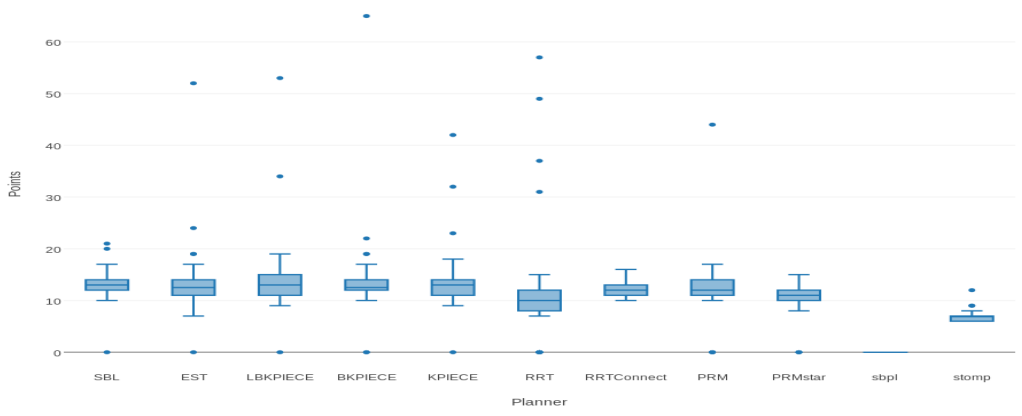

Figure 4-27: Number of points in trajectory for case 12

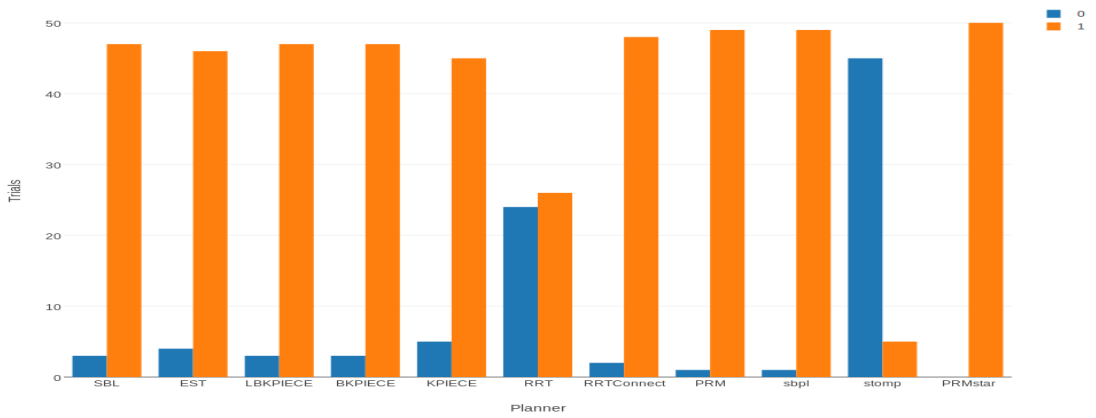

Figure 4-28: Success for 50 trials for case 16

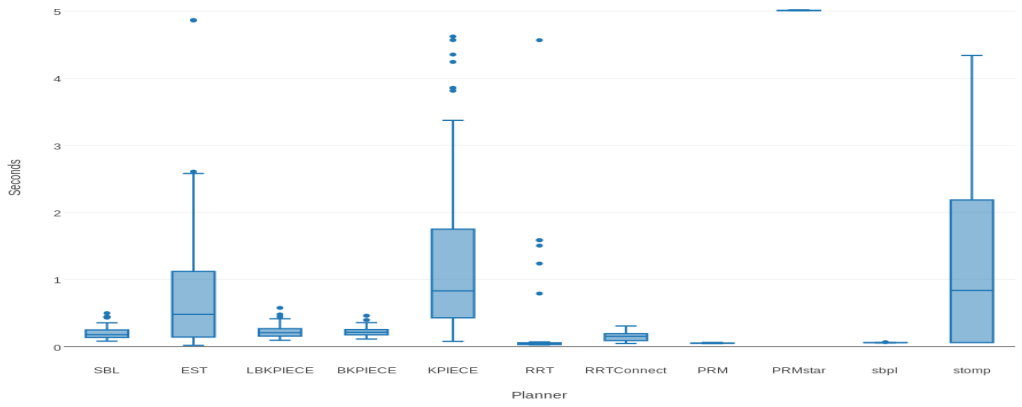

Figure 4-29: Planning time for case 16

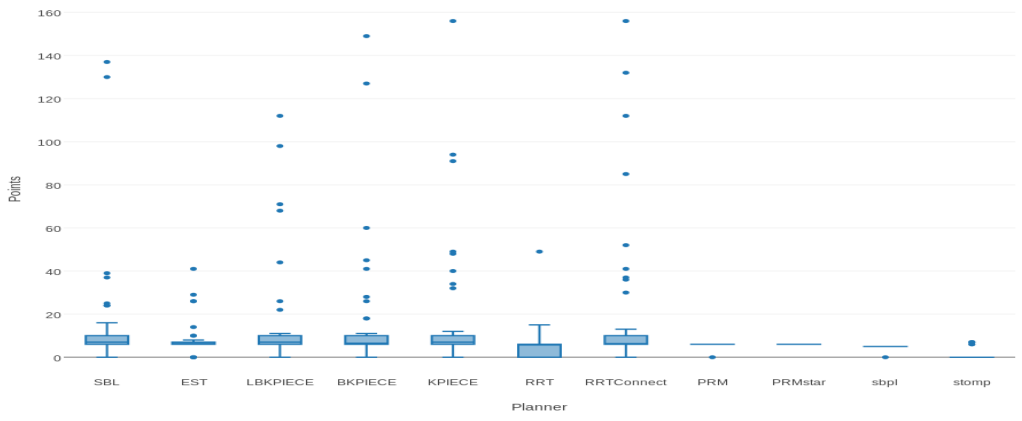

Figure 4-30: Number of points in trajectory for case 16

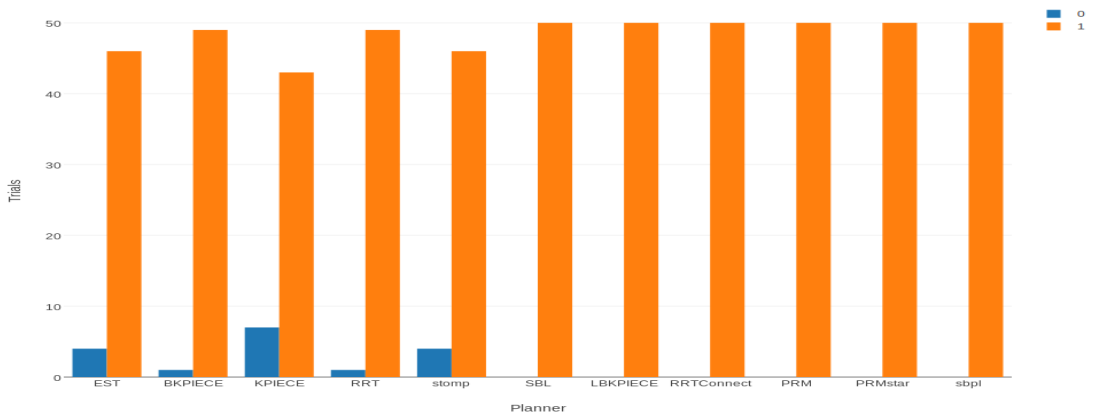

Figure 4-31: Success for 50 trials for case 18

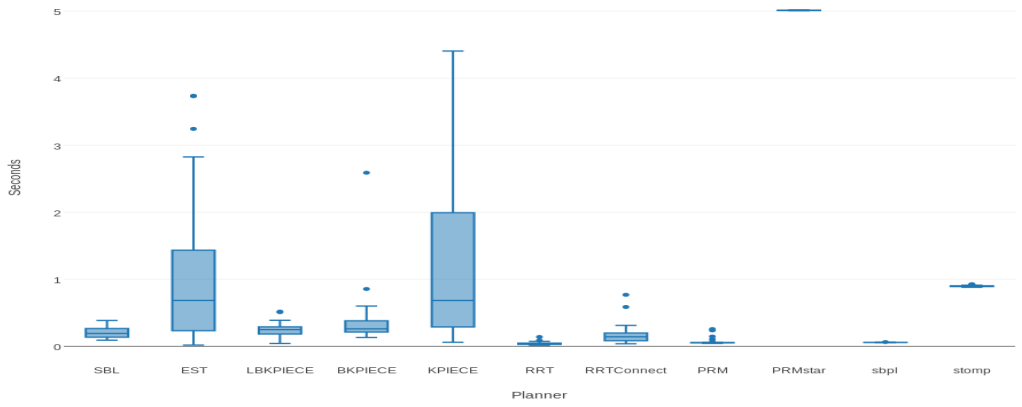

Figure 4-32: Planning time for case 18

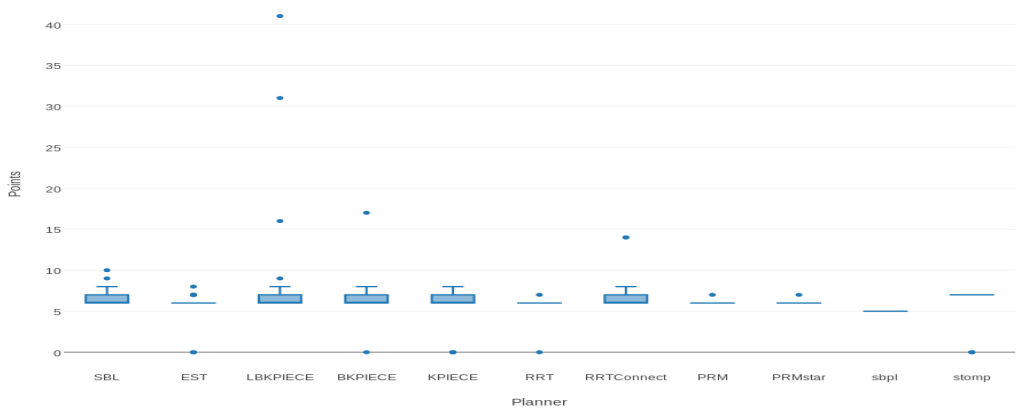

Figure 4-33: Number of points in trajectory for case 18

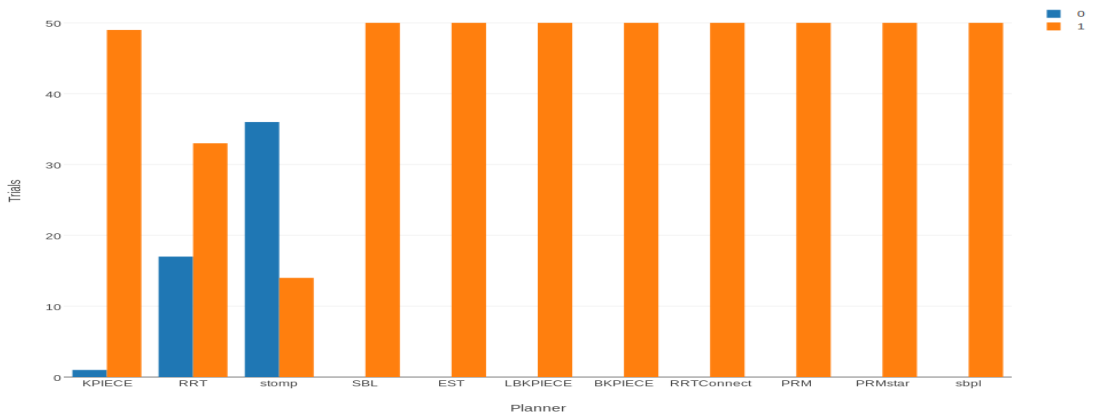

Figure 4-34: Success for 50 trials for case 27

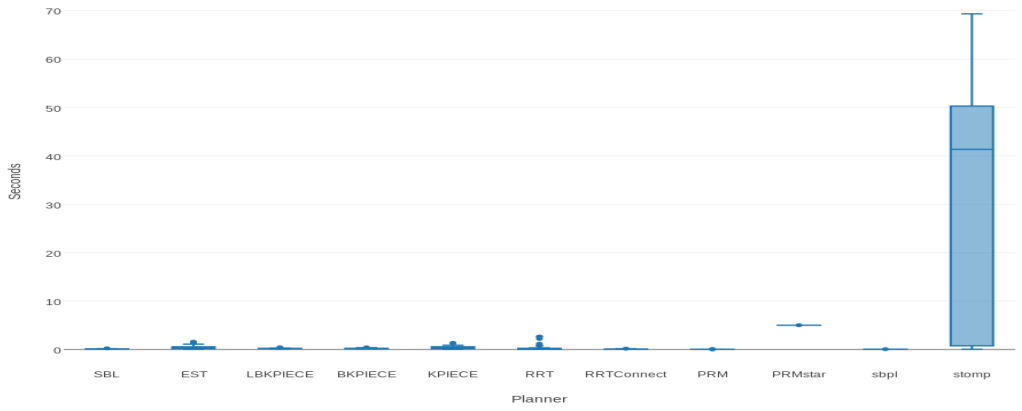

Figure 4-35: Planning time for case 27

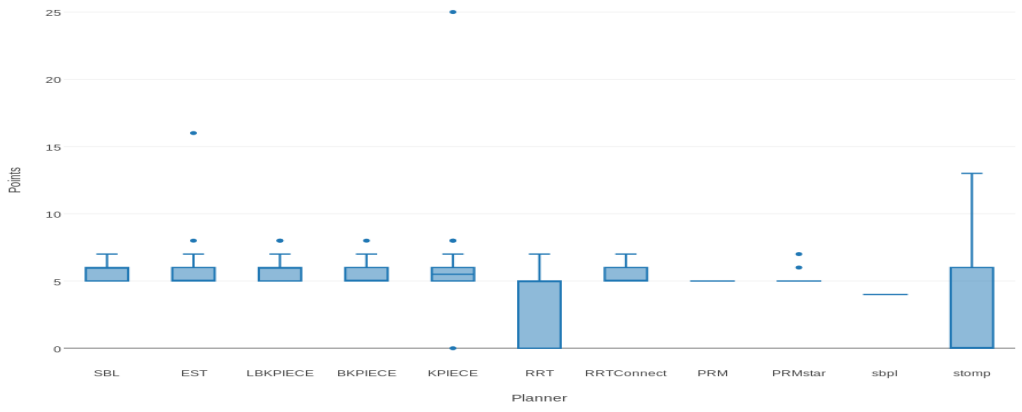

Figure 4-36: Number of points in trajectory for case 27

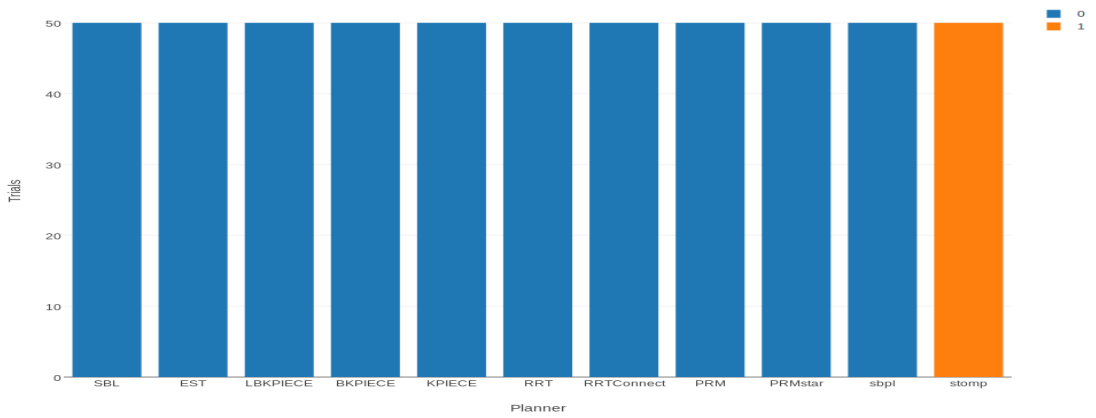

Figure 4-37: Success for 50 trials for case 32

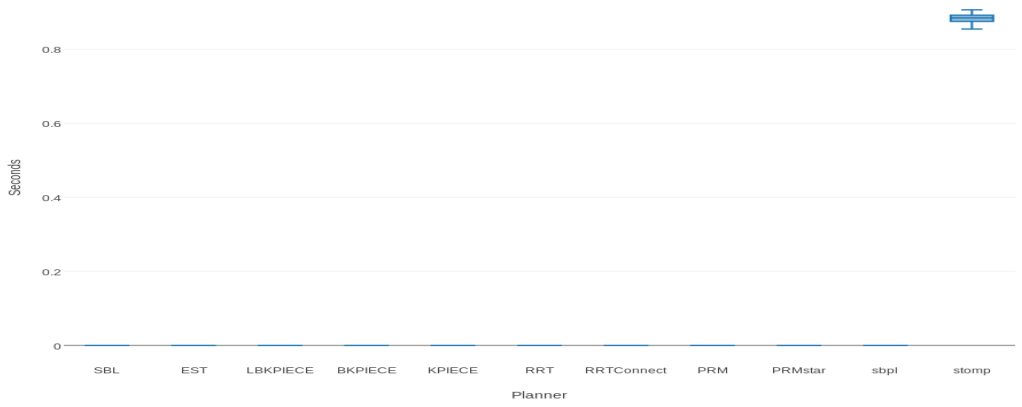

Figure 4-38: Planning Time for case 32

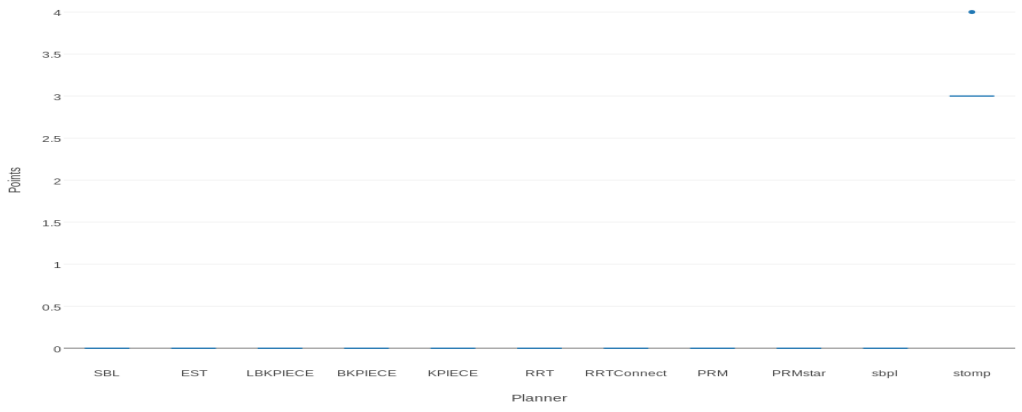

Figure 4-39: Number of points in trajectory for case 32

Table 4-7: Contextual Awareness Experiments

X clear	Y clear	where	height	Case	Planner Selected
low	low	center	medium	Case 1	RRTConnect
low	low	center	high	Case 2	SBPL
low	low	right	medium	Case 3	RRTConnect
low	low	right	high	Case 4	SBPL
low	medium	center	medium	Case 5	SBPL
low	medium	center	high	Case 6	SBPL
low	medium	right	medium	Case 7	SBPL
low	medium	right	high	Case 8	SBPL
low	high	center	medium	Case 9	RRTConnect
low	high	center	high	Case 10	SBPL
low	high	right	medium	Case 11	SBPL
low	high	right	high	Case 12	RRTConnect
medium	low	center	medium	Case 13	SBPL
medium	low	center	high	Case 14	SBPL
medium	low	right	medium	Case 15	SBPL
medium	low	right	high	Case 16	SBPL
medium	medium	center	medium	Case 17	SBPL
medium	medium	center	high	Case 18	SBPL
medium	medium	right	medium	Case 19	SBPL
medium	medium	right	high	case 20	SBPL
medium	high	center	medium	Case 21	RRTConnect
medium	high	center	high	Case 22	SBPL
medium	high	right	medium	Case 23	SBPL
medium	high	right	high	Case 24	RRTConnect
high	low	center	medium	Case 25	SBPL
high	low	center	high	Case 26	SBPL
high	low	right	medium	Case 27	RRTConnect
high	low	right	high	Case 28	RRTConnect
high	medium	center	medium	Case 29	SBPL
high	medium	center	high	Case 30	SBPL
high	medium	right	medium	Case 31	SBPL
high	medium	right	high	Case 32	Stomp
high	high	center	medium	Case 33	RRTConnect
high	high	center	high	Case 34	RRTConnect
high	high	right	medium	Case 35	RRTConnect
high	high	right	high	Case 36	RRTConnect

that it had low trajectory variations over 50 trials along with a planning time that was under a second. Through the series of experiments suitable planners are determined for situations that mimic situations that can be potentially faced in real world situations such as cabinets, shelves, tables etc. In the state machine the appropriate planner can be selected for a suitable situation.

Implementation in ROS

In order to extract the clearance of target pose the data from the 3D camera is used. In Moveit! the 3D information is processed in the form of octomaps [37]. A leaf iterator can be used to search for the closest obstacle in the surrounding voxels of the target pose. The octomap representation is less computationally expensive than using pointclouds since nearby points are clustered together to form a leaf of the octomap. Once the clearance is extracted from the 3D information the table of results can be iterated through to find the algorithm that has performed best in simulation. By having a fuzzy selection process to determine the most suitable algorithm for a planning problem the motion planning task can be accelerated.

4-7 Discussion

This chapter explored motion planning libraries, IK based genetic search and contextual awareness modules. The main motivation for exploration of these modules was to create a system that can take advantage of offline learning and can adapt to difficult motion planning tasks. Genetic search enables the system to reach a waypoint towards the target pose, a local planner can be used to complete the task. Though in this thesis a local controller was not used to complete the task it was proven that at least one point among the several points generated was a valid waypoint. It is important to tune the parameters of the genetic search algorithm, however it was not within the scope of this thesis. A simple threshold value was used to determine state validity but this can be replaced with a more intuitive function that takes the position and orientation of the end-effector into considerations. This remains to be a future research topic. In contextual awareness module the offline learning is utilized to select the appropriate motion planning algorithm according to the situation. The results however depend on parameter tuning for the algorithms in the planning libraries to a great extent. The motion planning libraries are tuned to a satisfactory value as explained in this chapter, however by changing the parameter values the performance will change. Keeping in accord with the aim and scope of the thesis the planners were tuned to a satisfactory level that indicates their ability to find solutions. The main aim of the contextual awareness module was to use markers to predict motion planning tasks in unregulated environments, make performance evaluators to rate the performance of offline learning results for planners with a fixed tuning. Furthermore the framework created to analyse depth information and determine the suitable planner is adaptable to changes in results of offline learning. The overall aim of this chapter was to develop and use tools that improve the success rate of motion planning tasks in unregulated environments by creating a flexible and modular framework.

Chapter 5

Conclusion and Future Research

In this thesis a system was developed for motion planning of a robotic arm in a domestic unconstrained setting. The arm is to be mounted on top of LEA, a robotic stroller developed at RCS for elderly users. In order to understand the requirements of the system use cases were developed that analyzed the potential user roles, their needs and precautions that need to be taken. The two main aspects of the developed system are vision and motion planning. The developed module allows the user to select the object they want by creating a bounding box and then the system refines the selection to extract as accurate coordinates for manipulation as possible. The module also checks if the position of the target object is changed and stops the execution of the motion planning task. Though grasping was not addressed in this thesis the vision system also allows to select where the object has to be placed if a successful grasp is performed.

The motion planning module consists of genetic search, contextual awareness and motion planning blocks. The genetic search uses IK of the arm to generate points that are close to the target pose for cases direct manipulation to the target pose is restricted. It was proven that at least a point can be generated that can act as a way point between the end-effector and the target pose for two scenes. The contextual awareness module examines the depth information received from the sensor and evaluates the distance of the closest obstacle to the target pose and the relative position of the target from the end effector. Sets are created that capture the permutations of the position and clearance in x and z direction. Through a series of benchmarking experiments algorithms from three motion planning libraries are evaluated and the most suitable for a certain case is noted. This allows for quick selection of planning algorithms in challenging cases. The modules of genetic search and contextual awareness were developed to assist in solving motion planning tasks The actual motion planning takes place using Moveit! software. The entire system was tested in V-rep simulation environment and then on the robotic arm. Before the system could be tested on the robotic arm an additional trajectory interpolation adaptor was added to Moveit! that uniformly adds points between generated trajectory points according to a user defined resolution by utilizing higher order spline based smoother. The motion planning task was tested on the robotic arm successfully. A driver was developed that communicated to the motor controllers present

in the robotic arm and read the joint positions from the encoders. The driver converted the trajectory points into CAN messages that are sent to the motor controllers.

The developed system does not take into account dynamic obstacles, since the depth map is only considered in the beginning of the motion planning task, this can be accounted for by constant examination of the depth map during trajectory execution. To test if a trajectory is executed properly a supervisory trajectory manager also needs to be developed. The supervisor would take into account collision with objects during trajectory execution and lags between expected and current joint states. The algorithms used in this thesis have been tuned to a satisfactory but not optimum level, this can be an avenue of research pursued in the future. The joint controllers on the robotic arm operate on simple PID controllers but they can be changed to accommodate load variations and incorporate gravity compensation. Furthermore, genetic search and the contextual awareness modules have been developed and tested as stand alone elements. To integrate them successfully on the real system will require work. A service robot must be able to make energy efficient, intelligent decisions that do not startle the user. To achieve this different motion planning libraries were examined. Haptic sensors can be used to identify contact with the environment since a camera cannot always scan the entire environment. The arm can cease trajectory execution in the event it comes in contact with the environment. For grasping tasks the haptic sensors can confirm contact with the object and can confirm that it remains grasped during the trajectory execution.

Appendix A

Appendix

A-1 OMPL

An insight into motion planning libraries Ompl, SBPL and STOMP is provided in this chapter. Within OMPl motion planning algorithms can be classified into 2 main parts:

- Combinatorial planning: The free configuration space is characterized explicitly using the connectivity of the free points in the configuration space into a graph. A solution is found using search algorithms.
- Sampling-based planning: Incrementally search the configuration space by incorporating collision detection to find a solution.

There are also optimization based planners within OMPL which allow for the user to define optimization criteria and define a custom cost function for finding the optimal paths. Traditionally the evaluation criteria is the path length, smaller the path better the algorithms. This however may not be applicable in the application which is being focused on in the thesis. A safer and more constrained path may be preferred over a shorter one.

The Control-based planners account for differential constrains. There are constraints on the velocity of the robot as it moves between points. Since controlling the velocity between trajectory points is not of the prime importance and can be achieved using the motor drivers, it is not pursued deeper in the thesis.

Combinatorial planning

Discretization the continuous space for path planning is useful for analysis of free spaces and possible paths can be taken by the robot. The Cartesian space is defined by connecting free points in a graph and then using search algorithms to find the solution. [44] presents a good insight into combinatorial planning algorithms. A few of the prominent methods are presented in this section.

Combinatorial planning algorithms produce a **road map** which is a graph in the free Cartesian space. In the graph each vertex is a configuration in the free Cartesian space and each edge is a path without collision through the obstacle free Cartesian space.

The focus is more on representation of the free space and obstacles around the robot. The search algorithms mentioned in [44] are used for finding the final path between different configurations.

Combinatorial planning algorithms are known as multi-query planners. Once the roadmap is built it can be used for multiple starting and finish states. The inherent problem is the need to map the entire Cartesian space in the beginning of each motion planning query in case of dynamic obstacles. This does however provide some interest for the thesis in areas where there are no changes in the environment and hence some methods are looked into.

Visibility Graphs One of the earlier path planning algorithms. A path is constructed connecting the initial and final state. Each vertex is a point in the observed Cartesian space and each edge represents a connection between two points. A polygon line is formed through series of interconnection between points. The shortest-path road map[45] finds the shortest path. The first application was in [46]. Figure A-1 displays the shortest polygon line from the initial point q_I to the final point q_G through the black polygons which are the obstacles.

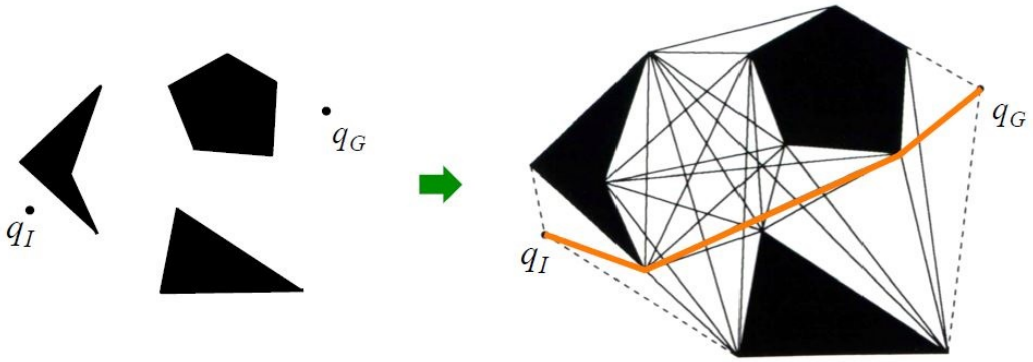

Figure A-1: Visibility graph

Voronoi Diagram As opposed to finding the shortest path between the goal and the initial point Voronoi diagram based algorithms attempt to navigate in the areas of maximum clearance of obstacles. This can be useful since the exact location of a robot as observed by the algorithm may be influenced by noise from the sensors. The visibility graph method would even suggest a path that just grazes and obstacle but in real world applications that can be risky. This is also known as the maximum clearance roadmap. As mentioned in [44]: Each point along a roadmap edge is equidistant from 2 points on the boundary of the observed Cartesian space. Each roadmap vertex corresponds to the intersection of 2 or more roadmap edges and is therefore equidistant from three or more points along the boundary of the observed Cartesian space.

Though is presents a better path planning approach in terms of dealing with uncertainties in the location of the robot and the obstacles there is a very large possibility of settling for suboptimal paths.

Exact Cell Decomposition The free Cartesian space is decomposed into cells that do not overlap with each other. A connectivity graph is constructed between adjacent cells and then a search algorithm is used to find the solution. A well known implementation of this method involves dividing the free Cartesian space into trapezoids and use their centers for connecting adjacent cells. The trapezoids are made with vertical lines through the space and shooting rays upwards and downwards from each polygon vertex. In figure A-2 the main steps of the algorithm can bee seen.

In (a) a Cartesian space is constructed with the obstacle region isolated, in step (b) vertical side segments by shooting rays upwards and downwards from each polygon vertex. In step (c) the interior of each trapezoid is considered as a point on the graph and possible paths are constructed. Finally, in (d) the optimum path is chosen.

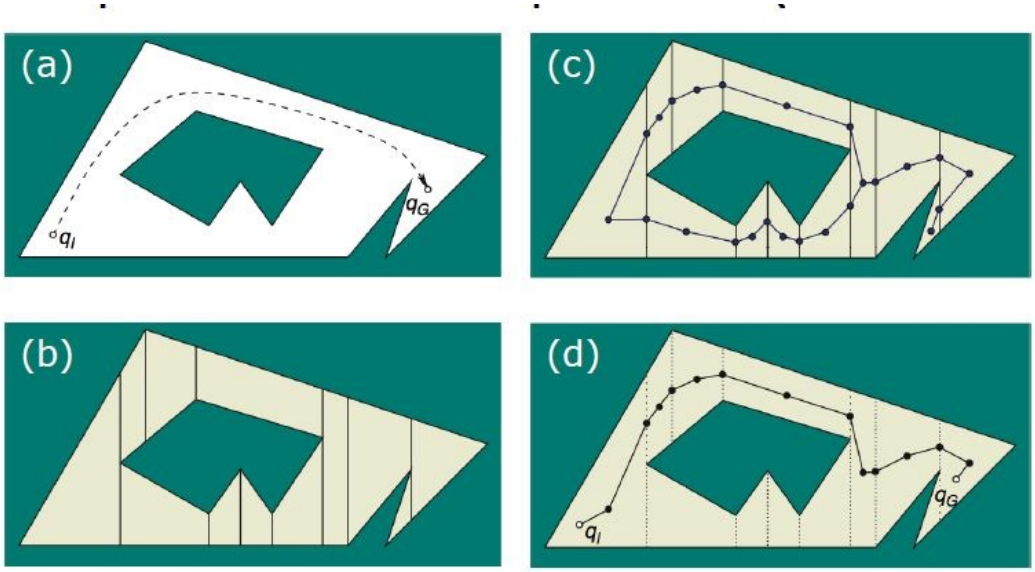

Figure A-2: Exact cell decomposition

This method can be computationally expensive and the graph has to be built every time an obstacle changes its location, also due the variable nature of the trapezoid shapes it can be hard to compute the points. In approximate cell decomposition the same size cells are used instead of a variable cell size. The approximate cell decomposition is numerically more stable and involves simpler calculations and is easier to implement.

Sampling based planning

Sampling based algorithms use collision detection in an iterative manner to find a solution to the path planning problem. Whereas combinatorial algorithms depend on

separating the free and occupied Cartesian space, sampling based algorithms have no need for such separation.

Sampling based algorithms have been becoming increasingly popular in robotic motion planning tasks. A through insight can be gained into sampling based algorithms in [44]. In this section an insight into sampling based algorithms is presented and the merits and demerits are discussed with pertinence to the thesis. A vertex is a configuration point and an edge is a path. It is desirable that the found interconnected edges are free of collision with obstacles. In [44] metrics to determine distance between manifolds.

Discretization of the configuration space is done using grids. However the configuration space is a topographical graph (γ) or a manifold and the vertices (configuration points) can be found using many algorithms such as K-nearest neighbours (Knn). The collision points with obstacles are removed from the topographical graph. The paths are found using search algorithms mentioned in [44].

Sampling based algorithms are probabilistically complete which means that if a solution exists then the probability of finding it converges to 1. The downside of this is that the algorithm may run forever in case it is unable to find a solution. For practical applications where a certain time is given to find the solution the algorithm may be terminated after a certain amount of time in case no solution is found and alternate approach may be tried. This mitigates the inherent flaw in the algorithm making it usable. In case of narrow passages sampling based algorithms have demonstrated low performance, however they are still widely used in robotics for motion planning tasks.

Sampling based algorithms construct a roadmap without considering the q_I and q_G. The generated road map is then used for any new points between which motion needs to be planned.

Probabilistic Road Maps (PRM) The philosophy of this method is to sample the points in the Cartesian space and use a local planner [47] to connect the points and construct a graph. The local planner checks for collisions in line of sight. Nearby points are found using k-nearest neighbors or similar algorithms. The map is built till it is dense enough to find a solution.

As it can be seen in Figure : A-3 that the free Cartesian space is approximated by randomly selecting points from the cartesian space and separating them in the obstructed space (C_{obs}) and free space C_{free}. The local planner connects the vertices and makes a roadmap by checking if the transition from current configuration to next configuration suffers from collision or not.

After construction of the roadmap the initial and final points for the motion planning are given ($q_I and q_G$). This is known as the query phase. The generated roadmap can be used for different queries. Using probabilistic roadmaps for motion planning removes the need for construction of a Cartesian space. They can also be used for higher dimensional problems and are probabilistically complete. PRM does not perform well for narrow passages. The solutions that are found are not optimal or complete.

OMPL has a few implementations of PRM algorithms. In their implementation one thread constructs a roadmap while the other checks for the existence of a path in the roadmap.

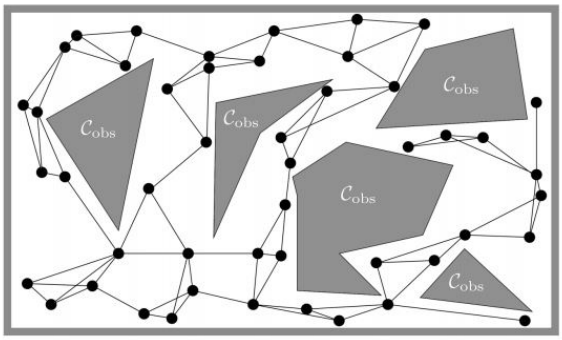

Figure A-3: Probabilistic road map

LazyPRM [48] is an adaption of PRM where the initial learning time taken is avoided or reduced by using a lazy collision checking strategy. The roadmap is evaluated repeatedly for the shortest path between the initial and the goal node and the nodes along the path are checked for collisions. In case there is a collision the node is removed from the map and a new shortest path is searched. In PRM* [49] the number of neighbors to connect is computed based on the coverage of the space instead of a fixed value. A lazy version of PRM* is also present which uses a lazy state validity checking.

Rapidly Exploring Random Trees (RRT) Rapidly exploring random trees comes under the family of rapidly exploring dense trees. In the case of RRT [50] the search method is random. The main aim is to aggressively generate a dense covering of the space to find a solution. The term 'Lazy' implies that collision detections are only done when absolutely necessary thereby reducing the computational requirements. The combination of its three traits makes it favorable as it is faster and can be applied to difficult motion planning applications.

The search starts by sampling from a bounded region centered around q_o which is the starting state . Tree branches are spread in a random manner exploring the topographical graph. Then the closest vertex is found in the topographical graph using a distance function. As mentioned earlier this has to be a predefined metric on the Cartesian space. The closest configuration point is then found using algorithms such as knn. The closest vertex can also be found by checking intermediate points at regular intervals and selecting one. This point is now q_{near}. From q_{near} a random point is connected using a local planner [47]. If there are no collisions then a new edge is added, else a new random point is found in the observed Cartesian space and the algorithm is restarted from that point.

A deeper insight into RRT algorithm can be found in [44]. RRT is popular in motion planning applications for robotics as it mostly finds a solution and there is no need to explicitly define the Cartesian space.

Path planning is performed using the RRT by selecting the new random state q_{rand} as the final state q_{goal} after a certain number of iterations. If the map is dense enough and a solution is found then the algorithm is stopped. A depiction of the RRT algorithm can be seen in Figure: A-4

It may seem like a good idea to select the final state as the new random state in every iteration in the hope that a simple solution exists, however this would restrict the exploration of the algorithm and most of the exploration would only take place in the observed Cartesian space.

For problems with a larger Cartesian space it is also possible to start the algorithm from the initial and final side with the aim that they intersect and a solution is found. This is known as bidirectional search and is demonstrated in Figure :A-5

Figure A-4: RRT

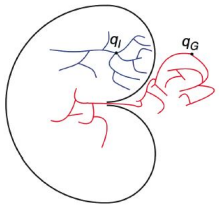

Figure A-5: Bi-directional RRT

Advantages

- Easy implementation
- No need to explicitly define Cartesian space
- Randomized search allows for good exploration of the space

Disadvantages

- The distances are measured by a predefined metric and is hence sensitive to that choice
- May take a long time to converge, in some cases may not converge at all

Stemming from the basic idea of RRT there have been variation in recent years. RRT-connect (RRTC) [51] uses the bidirectional search using simple greedy heuristics. The algorithm was initially used for solving the path planning problem for a 7 DOF arm but since has been extended for path planning problems. RRT-blossom [52] uses a flood-fill mechanism to avoid being stuck in local minima. This is targeted for high constrained environments which can be useful in case there are many obstacles surrounding the robot or the configuration space is narrow, this is useful since RRT algorithms do not perform well in narrow spaces. To optimize the performance of RRT in modern day computers PRRT [53] (Parallel RRT) algorithm is developed. EST [54] presents a tree based planner that incrementally updates the tree while retaining information from previous steps and biasing with a greedy exploration strategy. The comparative study in [54] shows that EST performs better than RRT for re planning in an unknown workspace with a non-holonomic platform. Lazy RRT [55] constructs RRTs on tangent space approximations and constraint manifold and performs lazy projections to the manifold when the deviation exceeds a certain threshold.RRT* algorithm[56] ensures the convergence to an optimal solution which was a draw back in RRT, however RRT* can prove to be slow. This problem was addressed in [57] using artificial potential fields.

Expansive Space Trees (EST) EST is a tree based search algorithm where the less explored areas in the space by using a grid imposed on the projection of the state space. In place of exploring all the areas in the state space, the areas of interest are explored and other areas are ignored. A space is considered expansive if every point in the free Cartesian space sees a small fraction of the free space and that every point must have a large lookout set. The lookout function is defined in Equation: A-1

$$LOOKOUT_{\beta}(S) = \{q \in S | \mu(\vartheta(q) \backslash S) \geq \beta \mu(\Gamma^{'} \backslash S)\} \tag{A-1}$$

where Γ is the set of all free configurations, S is a subset of Γ, $\mu(\vartheta(q))$ is the set of all configurations seen by a free configuration q, $\Gamma^{'}$ is the set of connected components in free space and $\mu(S)$ denotes the volume of S. β is a constant in (0,1].

Let ϵ ,α ,β be constants in (0,1]. The free space Γ is expansive if each of its connected components satisfies the following 2 conditions:

$$\text{For every point } p \in \Gamma^{'}, \mu(\vartheta(p)) \geq \epsilon \mu(\Gamma).$$
$$\text{For any connected subset S} \subseteq \Gamma^{'}, \mu(LOOKOUT_{\beta}(S)) \geq \alpha(S).$$

The basic version of the algorithms starts randomly sampling points but retains only the nodes that are path-connected to either the starting or the goal position. This way 2 trees start growing and when there is an intersection in the visibility regions a solution is found. The algorithm iteratively performs 2 steps: expansion and connection. A more detailed insight can be found in [58]. One of the greatest advantages of EST is that a solution can be found even if there is a narrow opening, the major problem being addressed can be seen in Figure: A-7 where S_1 and S_2 are sets of free space on either side of the narrow passage. The expansive exploration of the space is the key in solving the narrow passage problem.

OMPL has Single-query Bi-directional Lazy collision checking planner (SBL)[59] which follows the same expansion strategy. SBL is a single-query bi-directional PRM with lazy collision checking.

According to the OMPL description EST is not as sensitive to having a good distance measure, however this has been addressed by using a low-dimensional projection to evaluate the exploration of the state space.

KPIECE KPIECE [8] constructs a tree in the state space where each branch represents a motion. Each motion is attributed with a robot state, a control input and time for completion. KPIECE has proven to perform better than most algorithms for motion planning tasks. A multi layered grid based discretization is used to estimate the coverage of the state space. KPIECE uses physical simulations and the multi core capabilities of processors to boost the speed. There is no dependence on distance metrics or state space sampling which improves the speed of the algorithm. The motion planning problem is addressed in the form of a tuple S=(Q,U,I,F,f) where Q is the state space, U is the state space, I is the set of initial states and F is the set of final states. The dynamics are represented by a forward movement scheme f: $Q \times U \longrightarrow T_g Q$. The tree of motions μ= (S,U,t) are created iteratively . The unexplored areas of the state space are searched

through a discretization process. A multilayered grid is constructed starting from coarse and large cells to smaller and refined cells. Wherever the motion passes through, the cells are refined till the point that the motion is passing through only one cell. This helps in recognizing the large and empty parts of the grid. Figure: A-6 shows the discretization process, the higher layer have coarse cells and as the motion passes from the top to the bottom layer the cells get smaller. The cells in lite blue are the ones through which the motion passes.

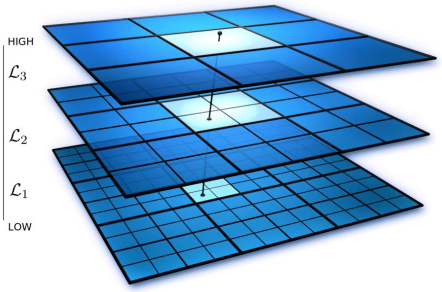

Figure A-6: KPIECE discretization

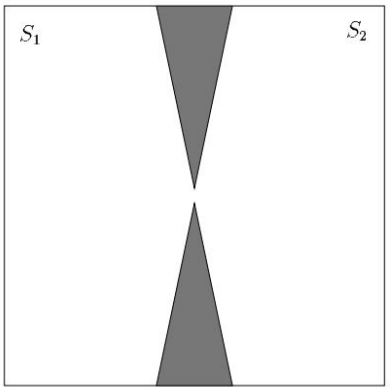

Figure A-7: Narrow passage problem

OMPL has 2 implementations of KPIECE. The first is Bi-directional KPIECE (BKPIECE) which is the regular implementation of KPIECE but with single level of discretization, only one grid is used. The second one is a lazy version of the Bi-directional KPIECE.

Path-Directed Subdivision Tree exploration algorithm (PDST) As explained in the OMPL library PDST [60] is a tree-based motion planner that instead of sampling points randomly it samples points randomly along a deterministically selected path segment. The less explored areas are detected using binary space partitioning of the projected state space. Large cells with fewer path segments are preferred, this translates to exploring empty areas in the space.

Search Based Motion Planning Sampling based motion planners are time efficient but at the expense of an optimal solution. If a sampling based motion planner is run several times it is very likely that its motion may be different every time. This may be a problem for certain applications where repeatability is more of a concern than time of motion planning.

Search based motion planners try and find the solution with minimal cost and guarantees solution sub-optimality. Motion primitives are predetermined motions a robot can make. Search based motion planning algorithms construct motion primitive based graph and search it for low cost solutions. The main aim is to look at the constructed graph for the existing state of the robot and find a solution in the graph that connects it to the final pose, position. The main steps for search based motion planning can be broken into:

- Construction of graph of motion primitives.
- Use a cost function to assign edge costs in graph.
- Use heuristic function to guide the graph search

A-2 STOMP

STOMP stands for Stochastic Trajectory Optimization for Motion Planning. It relies on generating noisy trajectories to find a valid path around an initially feasible/infeasible trajectory. The goal is to find a final valid trajectory with a lower cost. The cost function is defined by a combination of obstacle avoidance, constraint handling and trajectory smoothness. The goal is to minimize this cost function at every iteration. Constraint handling, torque control and energy minimization to be efficient in terms of battery consumption are the intent ed benefits of using a STOMP approach for motion planning. Since the robotic arm is intended to be used on a mobile robot in this thesis STOMP offers some strong points.

This planning library uses the STOMP algorithm for finding the trajectories. This is a local planner that can perform better under constraints as compared to OMPL. The drawback being that the planning time is high and it cannot solve complex motion planning tasks in reasonable time.

noisy trajectories are simulated to estimate cost and are then used to update candidate solution. optimization of motion torques to perform a movement. For stomp algorithm to work it needs the starting and goal points of a trajectory. In terms of motion planning it would be the starting pose of the end effector and the target pose, both of which are available in the beginning of the motion planning task. STOMP attempts to optimize the trajectory in each iteration by considering state-dependant costs. The general optimization problem addressed is:

$$\underset{\tilde{\Theta}}{min}\, \mathbb{E}\Big[\sum_{i=1}^{N} q(\tilde{\Theta_i}) + \frac{1}{2}\tilde{\Theta^T}\mathbf{R}\tilde{\Theta}\Big] \tag{A-2}$$

where:

- $\tilde{\Theta}$: A noisy parameter vector $\mathcal{N}(\Theta\ \sum)$ with mean Θ and variance $\sum$.
- q($\tilde{\Theta_i}$): A state dependent cost function. Torques, obstacle cost and constraints can be included.
- $\mathbf{R}$: Positive semi-definite matrix, this represents the *control* costs.

trajectory, exploration

The STOMP algorithm as mentioned in [19] is presented below:

- Given:
  - Start and goal positions x_0 and x_N
  - An initial 1-D discretized trajectory vector Θ
  - A state dependent cost function q(Θ_i)

- Precompute:
  - $\mathbf{A}$ = Finite difference matirx
  - $R^{-1} = (A^T\ \mathbf{A}\ )^{-1}$
  - $\mathbf{M} = R^{-1}$, with each column scaled such that the maximum element is $\frac{1}{N}$
  - Create K noisy trajectories, $\Theta_1 \cdots \Theta_K$ with parameters $\Theta + \epsilon_k$, where $\epsilon_k = \mathcal{N}(0,\ R^{-1})$
  - For k = $1 \cdots K$, compute:
    - $\mathrm{S}(\tilde{\Theta_{k,i}}\ ) = \mathrm{q}\ (\tilde{\Theta_{k,i}})$
    - $\mathrm{P}\ (\tilde{\Theta_{k,i}}) = \frac{e^{-\frac{1}{\lambda}S(\tilde{\Theta_{k,i}})}}{\sum_{l=1}^{K}\left[e^{-\frac{1}{\lambda}S(\tilde{\Theta_{l,1}})}\right]}$

    For i = $1 \cdots$ (N-1), compute: $[\ \delta\tilde{\Theta}\ ]_i = \sum_{k=1}^{K} \mathrm{P}\ (\tilde{\Theta_{k,i}})[\epsilon_k\ ]_i$
  - Compute $\delta\Theta = \mathrm{M}\ \delta\tilde{\Theta}$
  - Update $\Theta \leftarrow \Theta + \delta\Theta$
  - Compute trajectory cost $\mathrm{Q}(\Theta) = \sum_{i=1}^{N} \mathrm{q}(\theta_i) + \frac{1}{2}\ \Theta^T\ \mathbf{R}\ \Theta$

A is a difference matrix which when multiplied by position vector Θ, produces acceleration $\ddot{\Theta}$. We get:

$$\ddot{\Theta} = A\Theta \tag{A-3}$$

$$\ddot{\Theta^T}\ddot{\Theta} = \Theta^T(A^T A)\Theta \tag{A-4}$$

Hence when R = A^T A it ensures that Θ^T R Θ represents the sum of squared accelerations along the trajectory.

Motion planning for robotic arm using STOMP involves sampling and update steps for 6 dimensions individually for each iteration. Trajectories are considered in joint states and not Cartesian space, this reflects when using the planner in Moveit! API as well. The cost function mentioned in Equation: A-2 is for a general case, the cost function for the motion planning problem is as presented in Equation: A-5.

$$q(\Theta) = \sum_{t=0}^{T} q_0(\Theta_t) + q_c(\Theta_t) + q_t(\Theta_t) \tag{A-5}$$

Where q_o,q_c and q_t represent the obstacle, torque and constraints cost.

Obstacle Cost:

A binary voxel representation is taken from the 3D date received from the camera or laser scanner. The distance to the border of the closest obstacle is calculated using Euclidean Distance Transform (EDT). The EDT value is negative inside the obstacle, 0 at the boundary and positive outside the obstacle. Hence information about the penetration depth, contact and proximity are extracted from the 3D voxel map. In Moveit! an octomap representation is used to depict the occupied voxels, this is faster than a point cloud representation since instead of exact points , regions are depicted as occupied or free. In STOMP algorithm the robot body (β) is represented as a set of overlapping spheres. In order to avoid obstacles all points on the sphere have be a

distance ϵ away from the closest obstacle. If the sphere has a raidus r then the constraint becomes $\epsilon + r$. The obstacle cost function is shown in A-6

$$q_o(\Theta_t) = \sum_{b \in \beta} max(\epsilon + r_b - d(x_b), 0)||\dot{x_b}|| \tag{A-6}$$

Where:

- r_b: Radius of sphere b, b is a sphere and b $\in \beta$.
- t : Time computed from the kinematics model where the representation of the body is b.
- $||\dot{x_b}||$: Magnitude of the workspace velocity of the sphere.

Torque Cost: The torque of motors are represented as a function of their joint states and their derivatives:

$$\tau_t = f(X_t, \dot{X}_t, \ddot{X}_t) \tag{A-7}$$

where X_t represents the joint state and $\dot{X}_t$, $\ddot{X}_t$ represent the velocity and acceleration respectively. The torque cost is represented by adding the torques.

$$q_t(\Theta_t) = \sum_{t=0}^{T} |\tau_t| dt \tag{A-8}$$

Constraint Cost: The magnitude of violations of the constraints is used to optimize the end effector position or orientation. The summation of the violations acts as the constraint cost, as shown in Equation:A-6

$$q_c(\Theta_t) = \sum_{c \in C} v_c|(\Theta_t)|, \tag{A-9}$$

A-3 SBPL

SBPL has implementations of search based planners. Anytime A* with Provable Bounds on Sub-Optimality (ARA*) [61] is a heuristic search based motion planning algorithm that finds a suboptimal solution initially using a loose bound and then refines the bound. This is an improvement from traditional search based planners that initially find a solution and then continue to improve it till time runs out.

Search based planning library

Search based algorithms such as ARA*, AD*, R*, D* Lite, etc together form the SBPL. In contrast to sampling based motion planning algorithms that operate in continuous space search based algorithms work in discrete space. Graph search methods are used in all SBPL algorithms hence it is important to first convert the continuous space into discrete space and then find the best solution. Sampling based algorithms are oriented towards finding a feasible solution instead of minimizing a cost function.As mentioned in [62] Search based motion planning algorithms aim at finding solutions with minimal cost

which provides grantee on suboptimality of solution whereas sampling based algorithms do not provide grantees on suboptimality and provide completeness only in the limits of samples. Since a cost function is being minimized consistency of motion can be expected from this motion planner.

The most apparent difference between the two methods are speed and optimality: graph-search methods can guarantee how "efficient" a solution is (defined more later), but, in general, does so at the expense of computation time. Sampling based planners run fast, but can result in unusual looking and possibly erratic. When using in a domestic setting it is important that the user is comfortable with the motions of a robot, if the motions seem random to the user who would compare the motion of the robotic arm with the approach they would have taken then they might think something has gone wrong. Search based motion planners can be used for a more repeatable solution that is not erratic', this is explored in the benchmarking section.

The first step towards using search based motion planners is to create a representation of the state space. This is done by discretizing all possible states for each joint of the robotic arm. A path would be generated for the motion of the robot through adjacent states in the state space. State validity checking is done by checking if a state is in collision with an object or itself. Additional constraints such as joint constraints can be imposed on the planner in which case apart from checking collision with the environment the joint constraints would also be evaluated for state validity checking. The possible states are used to create a graph where each cell has an identification number, search based planners are then used to navigate through the graph in order to find the solution.

The main aspects of planning with search based algorithms as per [62] are:

- Graph Construction: A graph is represented as $G = (S, E)$ where S denotes the set of states in the graph and E is the set of transitions between the states.The states S is the set of possible joint configurations. The state cab be represented as a n+1 tuple (Θ_0, Θ_1, Θ_2, $\cdots$, Θ_m, m) for a manipulator with n joints. Where m represents the index of the motion primitive used to reach the state s. A motion primitive is a vector of joint velocity, they are defined for all the joints in the manipulator.
- Cost Function : This is the function that is to be minimized to find the smallest path length, maximum path smoothness and maximise the distance between the manipulator and an obstacle. The cost between 2 states s and $s^{'}$ is shown in Equation: A-10

$$c(s, s^{'}) = c_{cell}(s, s^{'}) + w_{action} * c_{action}(s, s^{'}) \tag{A-10}$$

  where:

  * w_{smooth} and w_{action} are weights. Setting a higher value to w_{smooth} increases the smoothness of the trajectory but at the expense of optimality of solution.
  * $c_{cell}(s^{'})$: This cost is calculated by measuring the shortest distance between manipulator state $s^{'}$ and the closest obstacle.
  * c_{action} : The fixed cost applied to each motion primitive. This can be set according to the power consumption, size of links and preference of joints that are to be moved.
  * c_{smooth}(s,$s^{'}$): The cost imposed on the trajectory to reduce random motions.

– Heuristics: Heuristics helps the search to move in a favourable direction. The heuristic function must be admissible and consistent. The function can be described as in Equation:

$$h(s) \leq c(s, s^{'}) + h(s^{'}) \tag{A-11}$$

where:

* s is the initial state and $s^{'}$ is the final state
* $c(s,s^{'})$ is the action that comes from connecting s and $s^{'}$

The heuristic function should accommodate for the desired position and orientation of the end effector. Hence the heuristic function is defined as:

$$h(s) = h_{xyz}(s) + w * h_{rpy}(s)$$

Where:

* h_{xzy}: The function is calculated by computing the cost to goal using 3D DijkstraÅŻ search for the voxels with the coordinates.
* h_{rpy}: The function computes a cost based on the difference in the orientation of the end-effector between the current and final state. This is equal to the angle of rotation about a fixed axis specified by the axis-angle representation of the rotation between the end effector orientation of state s and s_{goal}

– Search: ARA* search algorithm is preferred for searching through the graph for a solution due to its size. This provides a bound (ϵ) on the sub optimality of the solution at any point. The value of epsilon decays after every successful solution is found. If the resolution of this variable is very small it can increase planning time.

By changing the epsilon value for the search algorithm it is possible to change the optimality of the found solution. In cases where a quick reaction is required a suboptimal solution can be found.

Search based motion planning algorithms employ a graph-based search that guarantees the efficiency of a solution

A-4 CAN Protocol

The exact CAN protocol used for the manipulator at RCS is presented in this section. The joint values are in hexadecimal.

Protocol: Command Velocity pos0 pos1 pos2 pos3 timeStamp digitalout

Where commands are CAN messages to connect, reset, enable the joints. Position is a 32 bit signed long position value. Here Pos3 is the least important byte, pos0 the most important byte. TimeStamp is an arbitrary number, the module will copy this code in the answer. As an example:

0x20 - 0x14 0x04 0x00 0x00 0x83 0xF1 0x51 0x02

Here the command sets joint 0x20 to position 0x000083F1. The second digital output is set to high. The time stamp of the message is 0x51.

The message ID used in the response is the message ID+1. If the message ID of a command is 10, the response will have message ID 11. For example:

Protocol: ErrorCode pos0 pos1 pos2 pos3 timeStamp na na

This answer means that joint 0x20 is not active 'motor not enabled' and the current position is 33777.

Bibliography

[1] "Roomba." http://www.irobot.com/For-the-Home/Vacuum-Cleaning/Roomba.aspx.

[2] "Paro therapeutic robot." http://www.parorobots.com/.

[3] "International federation of robotics." http://www.ifr.org/service-robots/.

[4] A. S. B. Group, "Pepper robot." https://www.aldebaran.com/en.

[5] B. F. Robotics, "Buddy." http://www.bluefrogrobotics.com/en/home/.

[6] Z. Bien, D. Kim, M.-J. Chung, D.-S. Kwon, and P.-H. Chang, "Development of a wheelchair-based rehabilitation robotic system (kares ii) with various human-robot interaction interfaces for the disabled," in *Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on*, vol. 2, pp. 902–907, IEEE, 2003.

[7] U. of Bremen, "Friend." http://www.iat.uni-bremen.de/sixcms/detail.php?id=1078.

[8] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII*, pp. 449–464, Springer, 2009.

[9] R. B. Rusu, I. A. Şucan, B. Gerkey, S. Chitta, M. Beetz, and L. E. Kavraki, "Real-time perception-guided motion planning for a personal robot," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 4245–4252, IEEE, 2009.

[10] A. Cowley, B. Cohen, W. Marshall, C. J. Taylor, and M. Likhachev, "Perception and motion planning for pick-and-place of dynamic objects," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 816–823, IEEE, 2013.

[11] S. Chitta, E. G. Jones, M. Ciocarlie, and K. Hsiao, "Perception, planning, and execution for mobile manipulation in unstructured environments," *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, vol. 19, no. 2, pp. 58–71, 2012.

[12] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1409–1422, 2012.

[13] C. Rother, V. Kolmogorov, and A. Blake, ""grabcutâĂİ – interactive foreground extraction using iterated graph cuts," *ACM TRANS. GRAPH*, pp. 309–314, 2004.

[14] E. N. Mortensen and W. A. Barrett, "Interactive segmentation with intelligent scissors," in *Graphical Models and Image Processing*, pp. 349–384, 1998.

[15] M. A. Ruzon and C. Tomasi, "Alpha estimation in natural images," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 1, pp. 18–25, IEEE, 2000.

[16] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, pp. 105–112, IEEE, 2001.

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

[18] "Search-based planning library." http://sbpl.net/.

[19] 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.

[20] "Ros." http://www.ros.org/.

[21] "Moveit!." http://moveit.ros.org/.

[22] "V-rep." http://www.coppeliarobotics.com/.

[23] "Ros concepts." http://wiki.ros.org/ROS/Concepts.

[24] "Roslaunch ros." http://wiki.ros.org/roslaunch.

[25] "Camera driver." http://wiki.ros.org/openni2_launch.

[26] "Transformation package ros." http://wiki.ros.org/tfl.

[27] "Ros action server." http://wiki.ros.org/actionlibl.

[28] "V-rep." http://www.coppeliarobotics.com/.

[29] "Gazebo." http://gazebosim.org/.

[30] L. Nogueira, "Comparative analysis between gazebo and v-rep robotic simulators,"

[31] "Common place robotics." http://www.cpr-robots.com/.

[32] "Manipulator specifications for robotic arm." http://www.cpr-robots.com/products/sra.html/.

[33] "Point cloud library." http://pointclouds.org/.

[34] "Open tld." https://github.com/Ronan0912/ros_opentld/.

[35] L. Nogueira, "Comparative analysis between gazebo and v-rep robotic simulators,"

[36] 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.

[37] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013. Software available at http://octomap.github.com.

[38] "Trajectory interpolation adaptor." http://wiki.ros.org/industrial_trajectory_filters/Tutorials/filters_inside_moveit/.

[39] "Ompl planners." http://ompl.kavrakilab.org/planners.html/.

[40] "Moveit! setup assistant." http://docs.ros.org/hydro/api/moveit_setup_assistant/html/doc/tutorial.html/.

[41] S. Tabandeh, C. M. Clark, and W. Melek, "A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering," *Computer Science and Software Engineering*, p. 63, 2006.

[42] M. Neef, D. Thierens, and H. Arciszewski, *A case study of a multiobjective elitist recombinative genetic algorithm with coevolutionary sharing*. Utrecht: Universiteit van Utrecht, 1999.

[43] I. A. Şucan, M. Kalakrishnan, and S. Chitta, "Combining planning techniques for manipulation using realtime perception," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2895–2901, IEEE, 2010.

[44] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[45] S. K. Ghosh and D. M. Mount, "An output-sensitive algorithm for computing visibility graphs," *SIAM Journal on Computing*, vol. 20, no. 5, pp. 888–910, 1991.

[46] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," tech. rep., DTIC Document, 1969.

[47] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," *Robotics and Automation, IEEE Transactions on*, vol. 16, no. 4, pp. 442–447, 2000.

[48] R. Bohlin and L. E. Kavraki, "Path planning using lazy prm," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, pp. 521–528, IEEE, 2000.

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

[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] 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.

[52] M. Kalisiak and M. van de Panne, "Rrt-blossom: Rrt with a local flood-fill behavior.," in *ICRA*, pp. 1237–1242, 2006.

[53] J. Ichnowski and R. Alterovitz, "Parallel sampling-based motion planning with superlinear speedup," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1206–1212, IEEE, 2012.

[54] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[55] T. T. Um, B. Kim, C. Suh, and F. C. Park, "Tangent space rrt with lazy projection: An efficient planning algorithm for constrained motions," in *Advances in Robot Kinematics: Motion in Man and Machine*, pp. 251–260, Springer, 2010.

[56] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[57] A. H. Qureshi, K. F. Iqbal, S. M. Qamar, F. Islam, Y. Ayaz, and N. Muhammad, "Potential guided directional-rrt* for accelerated motion planning in cluttered environments," in *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on*, pp. 519–524, IEEE, 2013.

[58] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, pp. 2719–2726, IEEE, 1997.

[59] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Robotics Research*, pp. 403–417, Springer, 2003.

[60] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes.," in *Robotics: Science and Systems*, pp. 233–240, 2005.

[61] M. Likhachev, G. J. Gordon, and S. Thrun, "Ara*: Anytime a* with provable bounds on sub-optimality," in *Advances in Neural Information Processing Systems*, p. None, 2003.

[62] 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.

Glossary

List of Acronyms

PCL	Point Cloud Library
TLD	Tracking-Learning-Detection
ROS	Robotic Operating System
URDF	Universal Robot Description Format
FCL	Flexible Collision Library
CAN	Controller Area Network
OMPL	Open Motion Planning Library
SBPL	Search-Based Planning Library
STOMP	Stochastic Trajectory Optimization for Motion Planing
SRDF	Semantic Robot Description Format
IK	Inverse Kinematics
LEA	Lean Elderly Assistant
DOF	Degrees Of Freedom
ARA*	Anytime Repairing A*
GUI	Graphical User Interface
Knn	K-nearest neighbours
PID	Proportional Integral Differential
YAML	YAML Ain't Markup Language
PRM	Probabilistic Roadmap Method
EST	Expansive Space Trees
SBL	Single-query Bi-directional Lazy collision checking planner
KPIECE	Kinematic Planning by Interior-Exterior Cell Exploration
RRT	Rapidly-exploring Random Trees
API	Application programming interface