BEN-GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Planning Reach-to-Grasp Motion for a Robotic Arm using Rapid-exploring Random Trees

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE M.Sc DEGREE

By: Roi Reshef

JANUARY 2015

BEN-GURION UNIVERSITY OF THE NEGEV
FACULTY OF ENGINEERING SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Planning Reach-to-Grasp Motion for a Robotic Arm using Rapid-exploring Random Trees

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE M.Sc DEGREE

By: Roi Reshef

Supervised by: Sigal Berman

Author:………………………………….. Date:………………….

Supervisor:………………………………… Date:…………………..

Date:…………… Chairman of Graduate Studies Committee:……………………..

JANUARY 2015

ABSTRACT

The following research presents grasp-regions rapid-exploring random trees (GR-RRT) algorithm, an innovative algorithm for automatic planning of reach-to-grasp motion. This algorithm includes two phases: a-priori computation of grasp regions and path planning. The grasp regions are regions of grasp poses (position and orientation) that are expected to lead to successful grasps. The method is based on the minimum-volume bounding box (MVBB) clustering algorithm, the Gaussian mixture model (GMM) and the use of graspability maps. Path planning is based on the rapid-exploring random trees (RRT) algorithm augmented with the use of the GMM for intermittently generating additional goal configuration. The method maintains probabilistic completeness while ensuring collision-free path are planned towards poses that afford successful grasps. Paths planned by the RRT algorithm are typically tortuous. The triple smoothing heuristic was developed within this research, and integrated with the path planning algorithm.

The performance of the algorithm was evaluated and compared using simulations to the Inverse kinematics bi-directional RRT (IKBiRRT) algorithm. The IKBiRRT facilitates planning towards manually defined pose regions, in which the poses are assumed to be uniformly distributed. Two scenarios (apple-harvesting and home environment) were tested. In the apple-harvesting scenario the apples require a single grasp-pose region, while in the home environment scenario the objects (mug and frying pan) have a more complex shape and require multiple grasp-pose regions. Evaluation is based on a comprehensive analysis of both efficiency and quality. The algorithm was also implemented in hardware and tested in the home environment scenario.

Both algorithms successfully found collision free paths in all tested cases. The GR-RRT algorithm successfully planned paths to successful grasp poses in 82% of the trials. This was a very large increase with respect to the IKBiRRT algorithm in grasp-success rate with only a small increase in average planning time and path length. In the hardware implementation most paths were successfully executed. In one case a collision occurred due to unmodeled wires. Thus the advantages of the GR-RRT algorithm for reach-to-grasp path planning were validated. For physical implementation path planning should be augmented with collision avoidance during execution.

Index Terms – robotics, grasping, path planning, grasp planning, rapid-exploring random trees, minimum volume bounding box, Gaussian mixture model, path smoothing.

ACKNOWLEDGEMENTS

Foremost, I would like to express my gratitude to my supervisor, Dr. Sigal Berman, for continuously supporting this work with fun brainstorming and great ideas, for her motivational, enthusiastic and professional guidance throughout the last two years. Furthermore, I would like to thank Danny Eizicovits for fruitful collaboration and for his patience. Finally, I wish to thank both Dr. Yisrael Parmet and Michael Bendersky for assisting whenever needed and providing a solid statistical consulting and support.

This research was partially supported by European Commission in the 7th Framework Programme (CROPS GA no 246252).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

BIC	Bayesian Information Criterion
B^w	Matrix of bounds for a workspace goal region
CD	Collision Detection
C_{free}	Free sub-space of C_{space}
C_{obs}	Obstacle collision sub-space of C_{space}
C_{space}	Robot-arm's configuration space
FC, FCA	Force closure, Force closure angle
FK	Forward Kinematics
$G(V,E)$	Graph consisting a lists of vertices (V) and edges (E)
GMM	Gaussian Mixture Model (statistical model)
IK	Inverse Kinematics
JAIC	Joint Angles Index of Curvature (City Block distance)
MVBB	Minimum Volume Bounding Box
NDbc	Normalized Distance (City Block)
NDe	Normalized Distance (Euclidean)
PIC	Path Index of Curvature (Euclidean distance)
PRM	Probabilistic Road Map
q or Configuration	Set of the robot-arm's joint variables (configuration)
q_i or $q(i)$	The configuration carrying the index i
$q[i]$	The ith joint's variable
q_{init}	Initial configuration of the robot-arm
q_{goal}	Goal configuration of the robot-arm
RRT	Rapid-exploring Random Tree
S	Swath
SD	Stability Distance
T_i^j	Homogenous transformation of poses from the ith coordinate-frame to the jth coordinate-frame
W or Workspace	2D/3D space where the robot-arm lies
WGR, w	Workspace Goal Region
ϵ	Step size constant
τ_i	ith RRT Tree data structure
θ, θ^*	Volume minimization gain, its threshold (MVBB)

CHAPTER 1: INTRODUCTION

1.1 Project Background

Robotic applications handle complex and tedious tasks that have previously been done by humans. Today mobile robotic manipulators have become common in industry and in robotics research laboratories. For such systems autonomous object manipulation still remains one of the greatest challenges. In industrial robotics, robotic manipulation is often reduced to very structured tasks, making it is possible to plan the robot's entire motion beforehand. When the motion cannot be completely defined prior to execution, efficient planning algorithms are required for finding collision-free paths.

Robotic technology facilitates integration of robots in many everyday tasks. On-going research and development broadens these capabilities increasing robotic systems autonomy and robustness. Autonomous robotic systems with advanced object manipulation skills will have a high impact on many areas of life. Accordingly the demand for such capabilities is high. For example, service robotic applications are becoming extremely popular. There are many types of service robots, such as entertainment robots (Takahashi and Mitsukura 2012), rehabilitation robots (Prenzel, Feuser and Graser 2005), food serving robots (Jyh-Hwa and Su 2008)(Figure 1) and more. Advanced object manipulation capabilities are imperative for such systems.

Figure 1 - A food serving robot at Hajime Robot restaurant[1]

Agriculture is one of the fields in which there is a high demand for advanced manipulation capabilities. Due to constant growth in world population, commercial agriculture is in demand to raise production capacities. Due to its inherent difficulties agriculture has been relatively slow to adapt to robotic technology. Yet improving productivity in agriculture through the use

[1] Hajime Robot restaurant, Thailand. (http://hajimerobot.com/)

of robotics is diligently researched (Bulanon, et al. 2002) (Van Henten, et al. 2003). In selective fruit harvesting (Figure 2), a robot must harvest the fruit without damaging either the fruit the plant. This must be achieved albeit the lack of prior knowledge regarding the location of the fruit of the obstacles about it. Such robotic systems must include sensors for identification of obstacles and fruit, and be capable of planning a path towards poses in space from which the fruit can be harvested, during run-time.

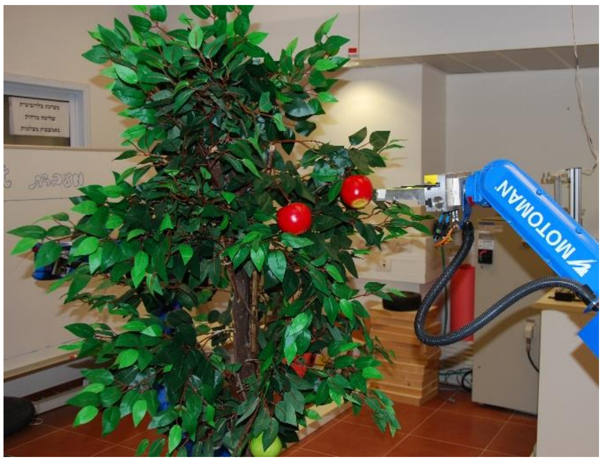

Figure 2 – Robotic arm picking an apple at the Tele-robotics lab at Ben Gurion University

Reach-to-grasp path planning is of high significance as grasping is the starting point of any manipulation task (Saut and Sidobre 2012). However, it is also considered among the most difficult tasks for robotic applications (Latash and Lestienne 2006). Grasp planning entails finding where to place the end-effector's fingers on the object in order to manipulate it. Reach-to-grasp planning includes planning the motion of the manipulator towards the pose that facilitates performing the required grasp.

This research deals with reach-to-grasp planning in unknown environments, where a model of the object to-be grasped is known *a priori*. Such environments where partial information regarding the environment is known are often referred to as *semi-structured* environments (Kim and Chung 2013).

1.2 Research Scope and Objectives

In the current thesis, path planning and grasp generation are combined to form a general reach-to-grasp planning framework Grasp-Regions Rapid Exploring Random Trees (GR-RRT) for robotic applications. Such a method should be efficient, i.e., it should terminate in a short time appropriate for real applications; high quality, i.e., it should provide paths that are as short as possible leading to a robust grasp of the object; and complete, i.e. it should guarantee finding a feasible solution if such exists.

This research targeted selective apple harvesting required for implementation within the Crops project, which targeted developing a robotic platform for selective harvesting of apples, peppers, and grapes. The scope of this thesis was further extended to deal with more general manipulation tasks, thus we tested algorithm applicability to general household objects (a mug and a frying pan). Two experiments were designed and executed in simulation to demonstrate the advantages and the algorithms developed in the current thesis. Finally the applicability of the planned paths was verifying in hardware.

As one of the requirements from the solution is efficiency, the research included a study of data structures and modification of planning techniques to achieve an efficient implementation. The developed algorithms include several parameters that must be defined by the user and can be optimized. The tuning of those parameters is beyond of the scope of the current thesis.

1.3 Research Contributions

The main contribution of this thesis is GR-RRT, a novel algorithm for reach-to-grasp planning that exploits graspability maps and randomized planning. The algorithm is suitable for complex objects for which finding suitable grasp poses is not trivial. Two additional contributions include:

- An improved path smoothing heuristic suitable for post processing paths (found by the RRT path planning algorithm) was developed. The method is especially suitable for complex environments.
- A thorough measure set for evaluation and comparison of different reach-to-grasp planning algorithms was formulated and used throughout the research. The measures quantify both performance efficiency and attained quality.

1.4 Thesis Structure

This thesis is organized as follows: Chapter 2 contains a literature review presenting the required mathematical background regarding robotic manipulators, path planning algorithms, path smoothing algorithms, grasp planning, and statistical inference and clustering techniques. Chapter 3 Presents the GR-RRT algorithm. Chapter 4 describes the main experiment conducted to evaluate the GR-RRT algorithm within the home environment and presents the result analysis. Chapter 5 presents the hardware implementation. Chapter 6 summarizes the work and presents the research conclusions. The development and testing of the core RRT planning engine and the path smoothing heuristic are detailed in Appendix I. An experiment testing GR-RRT for apple harvesting is detailed in Appendix II.

CHAPTER 2: BACKGROUND

2.1 Overview

This chapter reviews the fundamentals of robotics path planning, grasp planning, and the different concepts and methods that were used throughout this project. The issues discussed in this chapter are arranged as follows: Section 2.2 provides a background on robotics in general and in the domain of fruit harvesting. In Section 2.3 we formulate the problem of path planning. Following that, in Section 2.4 we describe various theoretical techniques for path planning such as polygonal-obstacle-region roadmaps, cell decomposition and potential fields. The concept of sampling-based motion planning is explained and the Rapid-exploring Random Trees (RRT) algorithm is discussed in detail. Section 2.5 includes a short introduction to path smoothing which is an important post-processing step for sampling-based planners. In Section 2.6 we review some approaches for reach-to-grasp planning, describe the concept of graspability maps, and introduce the workspace goal regions planning concept. Sections 2.7 and 2.8 present the fundamentals of statistical inference and clustering methods used in this work, e.g. Gaussian mixture models (GMM) and minimum volume bounding boxes (MVBB). Finally, the KD-Tree data-structure is explained in Section 2.9.

2.2 Robotics

Robotics deals with the design, construction, applications and operation of robots. It is a field of modern technology that extends traditional engineering boundaries. One definition for a robot is a “mechanical, computer controlled, device equipped with actuators and sensors” (Latombe 1991). Thus, understanding the complexity of robots and their applications requires knowledge of electrical engineering, mechanical engineering, systems and industrial engineering, computer science, economics, and mathematics (Spong, Hutchinson and Vidyasagar 2005). Diverse knowledge of these fields of study is being used to design and manufacture automated machines that replace humans, in performing hazardous or exigent tasks, such as manufacturing, helping the elderly, housekeeping, and more.

2.2.1 Robotics in Agriculture

The complexity of successful harvesting a crop greatly depends on the type of crop involved. For field crops, like corn and wheat, there are relatively few challenges. A single farmer can harvest a wide area quickly by riding over it in a combine, using GPS technology for steering. Several farm equipment makers have been developing technology that aims to allow tractors to operate without a farmer behind the wheel. For other crops, the challenge is typically harder. Workers often must pick the fruits by hand gently in order to avoid bruising them. Thus, the amount of personnel involved in harvesting such crops is dramatically higher. The increasing cost of labor, the shortage of skilled staff, and the shortage in seasonal workers compel the industry to find methods for reducing direct costs while increasing the productivity of already existing labor resources. One solution to these issues is robotic technology. Robotic systems have started to penetrate the agricultural industry. For example, in the middle of the 90s, the Douglas Bomford Trust and the Biotechnology and Biological Sciences Research Council (BBSRC) funded a project for the design of an autonomous vehicle for selective agrochemical operations (Hague, Marchant and Tillett 1997). In 2009, the Harvest Automation Company[2] offered a robot designed to move potted plants on nurseries. In 2010, researches of the Japanese Agricultural Machinery's Bio-oriented Technology Research Advancement Institution created an award-winning strawberry picking robot that can also sense their ripeness (Hicks 2012).

Due to the complexity of building safe and reliable robots for agricultural work, the agricultural domain has been relatively slow to adapt to robotic technology. The design of robotic systems for agriculture is harder than it is for traditional manufacturing, due to the unstructured environment and the sensitivity of objects in it. Harvesting robots need to operate in unfamiliar environments that are often cluttered with obstacles. They must guarantee a successful and selective harvest of specific fruits, where the fruit may be hidden behind obstacles and/or hard to reach. The systems must guarantee no damage is done to the fruit or to its plant. Finally, reachability is typically limited. These limitations place high flexibility, perception, and computation demands on the robotic systems and the task of developing economic robotic harvesters is still unresolved.

2.3 The Path Planning Problem

The path planning problem has relevance in areas such as robotics, computer graphics, simulations, geographic information systems (GIS), and more. It is considered among the difficult problems in robotics, where: "The description of this problem is deceptively simple, yet the path planning problem is among the most difficult problems in computer science" (Latombe 1991).

The basic path planning problem can be formulated as follows: Given a solid object in two or three dimensional space (2D or 3D) of known size and shape, its initial and target position and orientation, and a set of obstacles whose shapes, positions, and orientations in space are fully described, the task is to find a continuous path for the object from the initial position to the target position while avoiding collisions with obstacles along the way (Lumelsky and Stepanov 1987). Algorithm evaluation, in most applications, is done according to the quality of its returned solution(s). For example, a planning algorithm can compute a solution according to a cost function, e.g., returning a solution that minimizes the path's length, the time required to execute it, or the minimal distance to obstacles.

Planning collision-free paths is known to be a P-Space hard problem (Reif 1979), even in its simplest case. The original approach developed for solving path planning problems was the *complete planning* approach, which guaranteed finding a solution whenever one exists, or correctly reporting failure otherwise (LaValle, Planning Algorithms 2006). Complete path planning algorithms tend to suffer from poor performance when run-time planning is required, due to the vast computations they involve, and are vulnerable to high-dimensional search spaces. For run-time applications, a different approach towards planning has been developed, known as *Sampling-Based motion planning* (LaValle, Planning Algorithms 2006). These

[2] http://en.wikipedia.org/wiki/Harvest_Automation

algorithms trade completeness for a *probabilistic completeness*, i.e., the probability that the planner fails to find a path, if one exists, asymptotically approaches zero with the number of iterations growing to infinity. As a result, they impose lower computational complexity.

2.3.1 Path Planning for Robotic Arms

One of the ultimate goals in robotics is to create autonomous robots (Latombe 1991). Such robots should accept high-level descriptions of tasks and execute them without further human intervention. Developing the technologies necessary for such robots raises many important problems. Among them, and the central theme of this thesis, is path planning, which is also known as *motion planning* in the context of robotics. It can be simply defined as "how can a robot decide what motion to perform in order to achieve a desired manipulation of physical objects" (Latombe 1991).

In order to dive deeper into the world of robotic path-planning algorithms, several common concepts and definitions must be introduced. First, we define the *Workspace* (or *world*), marked as W, which is the space explicitly describing the geometry of the robot, the obstacles, and the object(s) of interest. Such spaces are either 2D, in which $W = R^2$ or 3D, in which $W = R^3$ (LaValle, Planning Algorithms 2006).

The basic goal of robotic path planning is to plan a collision-free path for a robot, connecting an initial pose (position and orientation) of the arm to a goal pose. To this end, a complete specification of the robot's full geometry must be provided. The primary way of specifying the state of a robotic arm (a *configuration*) is based on its joint-variables. The *Configuration Space*, often marked as C_{space} (or just C), represents the set of all possible robot configurations, given its kinematics ($q \in C$). The advantages of the configuration space representation are that a state of a robot which is a complex geometric shape, is mapped to a single point in C (which has a number of dimensions equal to the number of degrees of freedom of the robotic system), and that it makes the motion constraints of the robot more explicit (Latombe 1991) (Bruno and Oussama 2008).

In order to deal with obstacles in path planning, we define a subset of C containing configurations that represent collisions with obstacles as $C_{obs} \subset C$. Accordingly, the set of configurations that avoid collisions with obstacles is $C_{free} = C \setminus C_{obs}$, and is called the *free space* (Bruno and Oussama 2008). Given an initial configuration q_{init} and a goal configuration q_{goal}, the basic motion planning problem is defined as generating a free path between the two configurations, if they belong to the same connected component of C_{free}, and to report failure otherwise. Note that as we are planning for a rigid-body robot arm, all of its components must not collide with obstacles.

The classical approach to planning collision-free paths requires mapping the obstacles from the workspace into the configuration space in order to take them into account in the planning. The problem of transforming a presentation of the tool or the end-effector pose from W (workspace) into C (configuration space) is called *Forward Kinematics* (or FK), while the reverse problem is called *Inverse Kinematics* (or IK).

2.4 Path Planning Algorithms

In this section we present a survey of path planning algorithm. The workspace is continuous, making the set of possible configurations for the arm theoretically infinite. To overcome computation issues, complete motion planning approaches are based on a discretization of the original model.

2.4.1 Complete Motion Planning

In complete motion planning algorithms, a discrete representation is built based on the input model, in a way that the original problem is **exactly** represented. Virtually any of the algorithms in this subsection involve a construction of a *roadmap* in order to solve the path planning query. The idea behind the roadmap approach consists of capturing the connectivity of C_{free} in a network of one-dimensional curves.

Let $G(V,E)$ be a topological graph, i.e., every vertex $v \in V$ in G corresponds to a point in a subset $X \in \mathbb{R}^n$ (representing a part of the workspace), every edge $e \in E$ corresponds to a continuous function $\tau : [0,1] \to X$, the image of τ connects the points in X that correspond to the endpoints (vertices) of the edge, and the images of different edge functions are not allowed to intersect, except at vertices. Furthermore, let $S \in C_{free}$ be the *swath*, which is the set of all points in C_{free} reached by G. S can be expressed as:

$$S = \bigcup_{e \in E} e([0,1]) \quad (1)$$

A graph that holds these curves is called a *roadmap* if it satisfies the following conditions (LaValle 2006):

- **Accessibility:** From any $q \in C_{free}$, it is simple and efficient to compute a path $\tau : [0,1] \to C_{free}$ such that $\tau(0) = q$ and $\tau(1) = s \in S$ (in which s may be any point in S). Thus, it is always possible to connect some q_{init} and q_{goal} to some s_1 and s_2, respectively, in S.
- **Connectivity-preserving:** If a path exists such that $\tau : [0,1] \to C_{free}$, $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$, then there is also a path $\tau' : [0,1] \to C_{free}$ such that $\tau'(0) = s_1$ and $\tau'(1) = s_2$ (from the previous condition). Thus, solutions are not missed because G fails to capture the connectivity of C_{free}.

Once a roadmap R has been constructed, the path planning problem is reduced to connecting the initial and goal configurations to points in R and then performing a discrete graph search on R. The constructed path, if it exists, is the concatenation of three sub-paths: a sub-path connecting the initial configuration to the roadmap, a sub-path contained in the roadmap, and a sub-path connecting the roadmap to the goal configuration.

Various methods, which are based on the general idea of roadmaps, have been proposed and developed throughout the years, all compute different types of roadmaps, e.g., *visibility graph*, and *Voronoi diagrams* (LaValle 2006), (Latombe 1991). Each application operates at a different environment and has its own parameters to optimize, assumptions, and targets. Therefore, a different logic is applied while constructing the roadmaps.

The maximum-clearance roadmap and the shortest-path roadmap algorithms are similar in their methodology and both apply to 2D configuration spaces assuming a polygonal obstacle region, although they meet different goals. The ***Maximum-Clearance Roadmap***, also known as *retraction method* (Latombe 1991), is an algorithm that tries to draw a path as far as possible from C_{obs}. The paths resulting from using this algorithm are sometimes preferred by mobile robotics designers, since it is often difficult to measure and control the position of a mobile robot precisely. The retraction method is based on the roadmap called the *Voronoi diagram* of C_{free}. This diagram consists of a finite collection of straight and parabolic curve segments, known as *arcs*. A straight arc is a set of configurations that are closest to a pair of edges or a pair of vertices (Figure 2). A parabolic arc is a set of configurations that are closest to a pair of an edge and a vertex. An example of the 3 types of pairs is presented in Figure 4. Thus, the Voronoi diagram of C_{free} is the roadmap of segments whose minimal distance to the boundary of C_{obs} is achieved with more than one point of those boundaries. Figure 3 shows the roadmap constructed of a C_{free} bounded by a polygonal region.

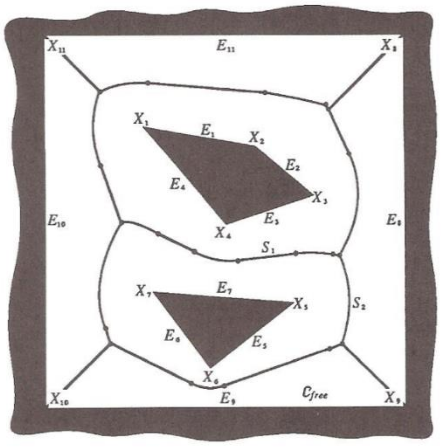

Figure 3 - Voronoi diagram of a polygonal obstacle bounded free space, from (Latombe 1991).

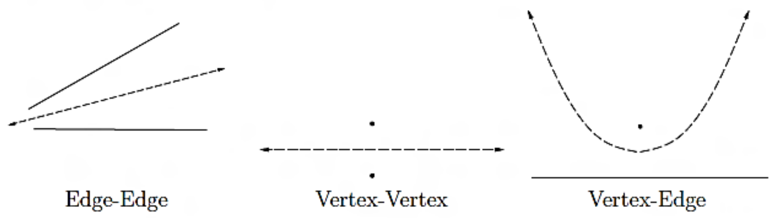

Figure 4 - Different arcs in Voronoi diagrams, from (LaValle 2006).

The ***Shortest-Path* Roadmap**, also known as the *reduced visibility graph method* (Latombe 1991), is one of the earliest path planning methods. The *visibility graph* is the graph G whose nodes are the initial and goal configurations q_{init} and q_{goal}, and all of C_{obs} vertices. The edges of G are all the straight line segments connecting two nodes that do not intersect the interior of the C_{obs} region. Although, in order to find the shortest path the graph G may contain the line segments connecting only the convex corners of the obstacles, thus it is reduced to what known as the *reduced visibility graph* (Latombe 1991). This means that the robot is allowed to touch or to "scrape" the obstacles, but it is not allowed to penetrate them. To actually use the computed paths as solutions to a path planning problem, they need to be slightly adjusted so that they come very close to C_{obs} but do not make contact (LaValle 2006). Figure 5 shows an example of such a graph and the resulting path.

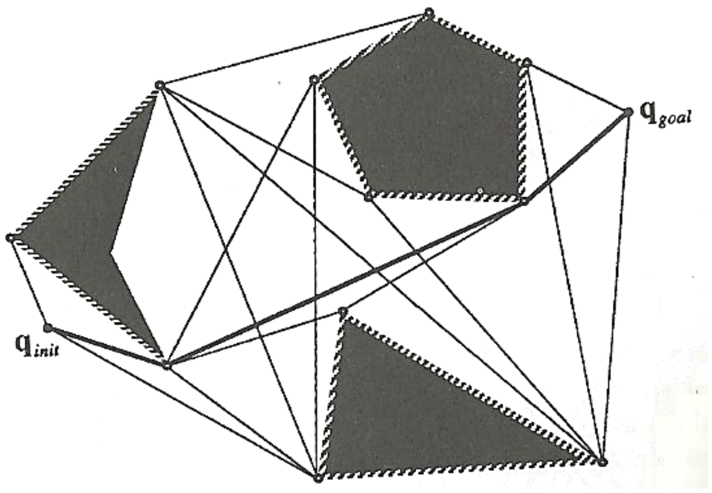

Figure 5 - An example of a reduced visibility graph, from (Latombe 1991).

Cell Decomposition planning algorithms consist of decomposing the robot's free region into smaller, non-overlapping regions. These regions are called *cells*, and their union is exactly C_{free}. Various decomposition approaches such as *vertical cell decomposition* (Chazelle 1987) and *triangulation* (LaValle 2006) based decomposition have been suggested (Figure 6). All cell decomposition algorithms must satisfy three properties (LaValle 2006):

- Computing a path from one point to another inside of a cell must be easy. For example, if every cell is convex, then any pair of points in a cell can be connected by a line segment.
- Adjacency information for the cells can be easily extracted to build the roadmap graph. This roadmap graph is called the connectivity graph, its nodes are the cells extracted from the free space and connected by a link in a way that two nodes are connected if and only if the two corresponding cells are adjacent.
- For a given q_{init} and q_{goal}, it should be efficient to determine which cells contain them.

After the decomposition, the second step of this method is to construct a roadmap (or *connectivity graph*). For each cell C_i we denote an arbitrary sample point q_i such that $q_i \in C_i$. We then join the sample point of every cell C_i to the sample point of every cell $C_i{}'$ adjacent to C_i. Once the roadmap is obtained, it is straightforward to solve the motion planning query.

Let C_0 and C_k denote the cells that contain q_{init} and q_{goal} respectively, the problem can formulated as the search for a path that connects the sample point of C_0 to the

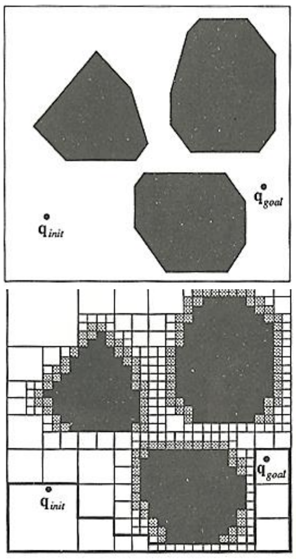

Figure 6 - an obstacle map (upper) and a cell decomposition map consisting of square decomposition, from (Latombe 1991).

sample point of C_k, in the roadmap graph G. Let $q_0, \ldots, q_k$ denote the sample points along the path in G. Then the solution path $\tau : [0,1] \rightarrow C_{free}$ is formed by setting $\tau(0) = q_{init}$ and $\tau(1) = q_{goal}$, and visiting each of the points in the sequence from q_0 to q_k.

The ***Potential Field*** approach (Khatib 1986) doesn't aim to capture the connectivity of the robot's free space into a reduced graph as previous methods do. Inspired by obstacle avoidance techniques, it treats the robot as a particle under the influence of an artificial potential field U. This potential function is typically defined over the free space as the sum of an *attractive potential* stems by the goal configuration, pulling the robot towards it, and a *repulsive potential* stems by obstacles, pushing the robot away from the obstacle region.

In this method planning is performed iteratively. While constructing the path, the potential function induces the artificial force $\vec{F}(q) = -\vec{\nabla}U(q)$, pointing locally to the maximum of U. Once U is defined, a path can be computed by starting from q_{init} and proceeding using gradient descent. This method is highly prone to presence of local minima.

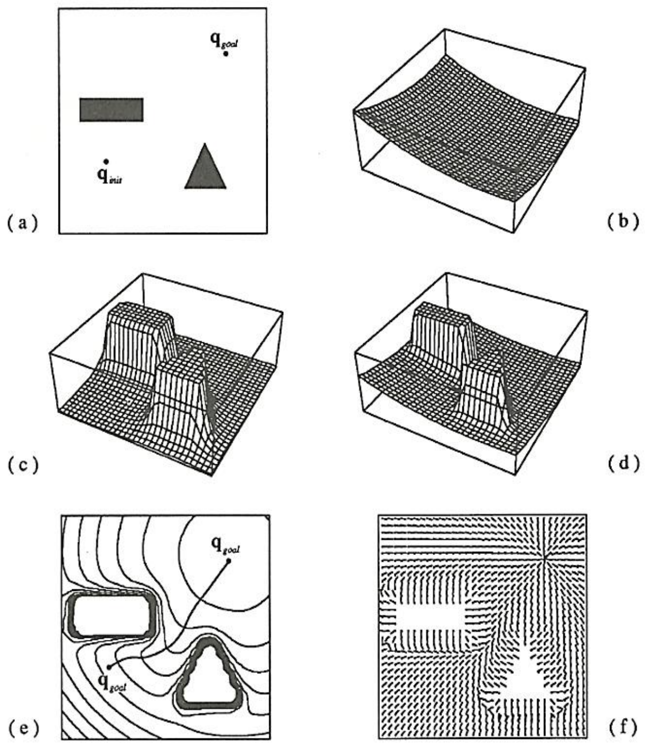

Figure 7 - Potential field example
(a) 2D map of obstacles. (b) The potential field caused by initial and goal points. (c) The potential field caused by obstacle (d) The combined potential field of both obstacle and initial and goal points (e) The resulted path (f) The gradients map. (Latombe 1991)

The reviewed approaches relax the completeness requirement to a weaker requirement, i.e., the ability to return a valid solution, if one exists and the resolution parameter of the algorithm is set to be fine enough (also known as *resolution completeness*). The algorithms provide the best accuracy with respect to their resolution parameter, yet, they all require an explicit representation of the obstacles in the configuration space. This requirement may result in an excessive computation making them often "unsuitable for practical applications" (Karaman and Frazzoli 2011).

2.4.2 Sampling-Based Motion Planning

Sampling-based motion planning refers to algorithms use collision detection modules to sample the configuration space and conduct discrete searches that utilize these samples. In this case, completeness is sacrificed, but it is often replaced with a weaker notion of *probabilistic completeness*. Whilst completeness guarantees are weaker, the efficiency and ease of implementation of these methods have made them suitable for a wide variety of applications, robotics in particular.

The ***probabilistic road-map*** (PRM) is one of the first sampling-based motion planners (Kavraki, et al. 1996). This approach utilizes random sampling of the configuration space to approximate a roadmap of the free configuration space in a computationally efficient way. PRM is divided into two phases: a *learning phase* and a *query phase*. In the learning phase, a PRM is constructed and stored as a roadmap graph (i.e., a graph whose nodes correspond to collision-free configurations and whose edges correspond to feasible paths between these configurations). Constructing a PRM is a conceptually straightforward process. First, a set of random configurations is generated to serve as the nodes in the network. Then, a simple, local path planner is used to generate paths that connect pairs of configurations. In the query phase, q_{init} and q_{goal} are connected to two nodes of the roadmap; the roadmap is then searched for a path joining these two nodes.

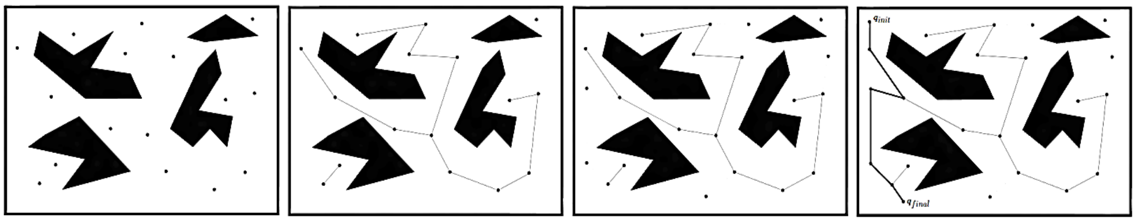

Figure 8 - planning with PRM, (Spong, Hutchinson and Vidyasagar 2005).

One difficulty using roadmap approaches is identifying narrow passages. A uniform random sampling of C_{free} produces in any particular region (within C_{free}) a number of samples that is proportional to its volume (Spong, Hutchinson and Vidyasagar 2005). Thus, using this approach, it is unlikely to place samples in narrow passages of C_{free}, which are often required to reach a solution. Several proposals have been suggested, such as the *bridge test* (Hsu, et al. 2003), which boosts sampling density inside narrow passages it detects by simple tests of local geometry and the *enhancement phase* (Spong, Hutchinson and Vidyasagar 2005), a post-processing phase that involve the connection of major disconnected components constructed by PRM using an additional local-minima-escaping planner.

The ***rapid exploring random trees*** (RRT) algorithm also belongs to the family of the randomized planning algorithms, but it operates in a different fashion. The idea behind RRT is to incrementally construct a search tree that gradually improves the resolution but does not need to explicitly set any resolution parameters (LaValle 2006). The construction is done by randomly sampling the search space and heuristically trying to connect new samples with the existing tree. Some key advantages of RRT are its heavily biased expansion toward unexplored portions of the state space, the fact that the tree constructed always remains a single connected component, and its probabilistically complete guarantee. Since RRT-based algorithms are probabilistically complete, the coverage of the configuration space gets arbitrarily close to any point in C_{free} with increasing numbers of iterations (Harada, Yoshida and Yokoi 2010). RRT is widely used due to its simplicity and efficiency as well as the possibility of involving differential constraints and many degrees of freedom (Harada, Yoshida and Yokoi 2010).

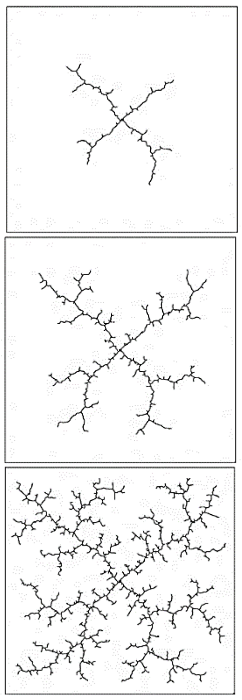

Figure 9 - Iterative construction of an RRT tree, taken from (LaValle 1998)

2.4.3 Rapid-exploring Random Tree Algorithm

RRT randomized data structure was first introduced in order to densely cover and reach as close as possible to every configuration in the search space (LaValle 1998). In contrast to previous approaches, it is not based on an exhausting attempts trying to connect pairs of configurations (or states). An RRT is a topological graph $G(V,E)$. The basic tree expansion algorithm that doesn't account for obstacles is explained as follows (LaValle, Planning Algorithms 2006): initially, a vertex is defined at q_{init}. For each of the iterations in a pre-defined number of iterations, the tree is iteratively grown by randomly sampling a new point in the search space q_{rand} and connecting q_{rand} to its nearest point in the swath $q_{near} \in S$. The connection is made along the shortest possible path. In every iteration q_{rand} becomes a vertex in G. However, if q_n lies in the interior of an edge (rather than on one of its endpoints), then the existing edge is split so that q_n becomes a new vertex and an edge is made from q_n to q_{rand}, as shown in Figure 10. Thus in each iteration, the total number of edges may increase by one or two.

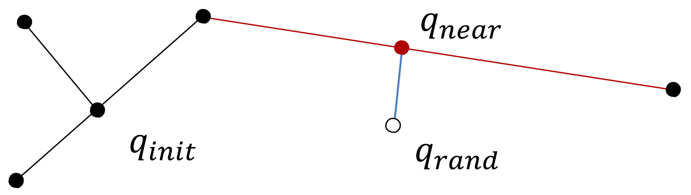

Figure 10 - The expansion method of the RRT-algorithm, taken from (LaValle 2006)

In fact, q_{rand} is not always directly connected to S. In case the distance from q_{rand} to q_{near} is smaller than some fixed incremental distance ε , the actual sampled point is connected. But in case the distance is bigger than ϵ, a linear motion is made from q_{near} toward q_{rand} t with some fixed incremental distance, ϵ (Figure 11).

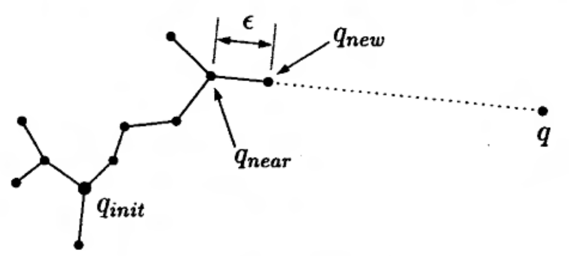

Figure 11 - The incremental expansion of the RRT-algorithm, taken from (Kuffner and LaValle 2000)

An obstacle avoidance module must be integrated in order to account for obstacles. A key advantage of the RRT sampling based algorithm is that it requires no translation of obstacles into the search space (C_{space}). Instead, for every configuration considered in the search space, the collision detection module calculates the corresponding points along the arm in the workspace using FK only.

```
$\boldsymbol{BUILD\_RRT(q_{init})}$
 1. $\tau.init(q_{init})$;
 2. $for\ k = 1\ to\ K\ do$
 3.     $q_{rand} \leftarrow RANDOM\_CONFIG()$;
 4.     $EXTEND(\tau, q_{rand})$
 5. $Return\ \tau$

$\boldsymbol{EXTEND(\tau, q)}$
 1. $q_{near} \leftarrow NEAREST\_NEIGHBOR(q, \tau)$;
 2. $if\ NEW\_CONFIG(q, q_{near}, q_{new})\ then$
   2.1.     $\tau.add\_vertex(q_{new})$ ;
   2.2.     $\tau.add\_edge(q_{near}, q_{new})$ ;
   2.3.     $if\ q_{new} = q\ then$
   2.4.         $Return$ Reached;
   2.5.     $else$
 3.             $Return$ Advanced;
 4. $Return$ Trapped;
```

Figure 12 - Basic RRT Algorithm pseudo-code (Kuffner and LaValle 2000)

Figure 12 presents a pseudo-code of the basic form of the RRT algorithm for path planning. NEW_CONFIG sets q_{new} to be one of the following values:

- The new sampled point (when contained in C_{free} and the distance from q_{near} is smaller than ϵ).
- A new point representing a partial progress toward the sample point (whenever the distance from q_{near} to it is bigger than ϵ).
- A false value (when no progress of size ϵ can be made due to a collision with an obstacle).

There are several ways to use RRTs in a planning algorithm. One approach is to bias q_{rand} so that q_{goal} is frequently chosen according to a constant rate (LaValle 1998), which will be further referred to as the ***goal-biased RRT***.

Another approach is to develop a bidirectional search by growing two trees, one from q_{init} and one from q_{goal} (Kuffner and LaValle 2000), which is termed ***bi-directional RRT*** (also known as RRT-Connect). Using the *connect* heuristic the two trees are biased to connect to each another, creating a continuous path starting from q_{init} and ending at q_{goal}.

The *connect* heuristic involves an additional effort while iterating, to expand each one of the trees. It consists of using the last q_{new} of one tree as a substitute for q_{rand} in extending the other tree. This causes the expansion of each tree to be biased towards the other tree. This effort consumes roughly half of the time, while the other half is spent expanding each tree in the usual way. A pseudo code is presented in Figure 13.

```
$\boldsymbol{RRT\_CONNECT\_PLANNER}(\boldsymbol{q_{init}}, \boldsymbol{q_{goal}})$
 1. $\tau_a.init(q_{init});\ \ \tau_b.init(q_{goal});$
 2. $for\ k\ =\ 1\ to\ K\ do$
   2.1.   $q_{rand} \leftarrow\ RANDOM\_CONFIG();$
   2.2.   $if\ not\ (EXTEND(\tau_a, q_{rand}) = Trapped)\ then$
      2.2.1.   $if\ (CONNECT(\tau_b, q_{new}) = Reached)\ then$
         2.1.1.1.   $Return\ PATH(\tau_a, \tau_b);$
      2.2.2.   $SWAP(\tau_a, \tau_b);$
 3. $Return$ Failure;

$\boldsymbol{CONNECT}(\boldsymbol{\tau}, \boldsymbol{q})$
 1. $repeat$
   1.1.   $S \leftarrow\ EXTEND(\tau, q)$
   2. $until\ not\ (S =\ A$dvanced)
 3. $Return$ S;
```

Figure 13 - bi-directional RRT algorithm pseudo-code

Note that the trees swap after each iteration of the RRT_CONNECT_PLANNER, until the latter returns a successful result. Using this heuristic running times improve often by factor of three or four (Kuffner and LaValle 2000).

2.5 Path Smoothing

Due to their randomized nature, sampling-based motion planning algorithms, and RRT in particular, give up the requirement for path length optimality in order to obtain a feasible solution efficiently. As a result, a key drawback of RRT's results is that they are of low quality, i.e., they tend to be ragged and tortuous and often contain a lot of unnecessary motions beyond those actually required to connect the initial and goal configuration (Carpin and Pillonetto 2005). These are undesired properties for robotic motion planning.

To address this shortcoming, researchers often suggest combining RRT with path smoothing (often called *path shortening*) techniques, for example, using path modifications through nodes shifting and relocating (Waringo and Henrich 2006), omitting unnecessary nodes (Abbadi and Matousek 2012), or using different planning approaches throughout the path construction phase as in (Khanmohammadi, Mahdizadeh and VakilBaghnlisheh 2008). Any change in the path already generated involves checking the feasibility of new configurations against the workspace.

Finding an **optimal** solution using node elimination approach is often not applicable, as an exhaustive search for the subset of nodes (from the N nodes of the original path) whose length is minimal involves checking $O(2^N)$ modified versions of the original path. Each path-feasibility evaluation requires several calls to the collision detection module. To overcome these issues a simple smoothing technique based on an iterative *divide and conquer* concept, similar to a binary search, was proposed (Carpin and Pillonetto 2005). Using this approach, nodes along the path are removed if the direct path between their predecessor and successor is feasible. A pseudo code is presented in Figure 14. This procedure is repeatedly called until the resulting path is fully smoothed, according to the environment limitations.

```
$\boldsymbol{SMOOTH(V, first, last, S)}$
 1. $if\ (first = last)\ then$
    1.1.    $S.pushback(V[first])$
 2. $else\ if\ (first = last - 1)\ then$
    2.1.    $S.puchback(V[first])$
    2.2.    $S.puchback(V[last])$
 3. $else\ if\ \begin{pmatrix} the\ segment\ connecting\ V[first] \\ and\ V[last]\ lies\ entirely\ in\ C_{free} \end{pmatrix} then$
    3.1.    $S.puchback(V[first])$
    3.2.    $S.puchback(V[last])$
 4. $else$
    4.1.    $SMOOTH(V, first, (first + last)/2, S)$
    4.2.    $SMOOTH(V, (first + last)/2, last, S)$
```

Figure 14 - Path smoothing based on divide and conquer approach. V is the input vector of vertices to smooth, S is a placeholder for the output (smoothed path vertices) vector, first and last parameters should be called with the extreme indices of the input vector V

2.6 Reach-to-Grasp Planning

Object manipulation is considered a major component in human daily tasks. It is becoming clear that to allow robots to carry out accurate and intelligent tasks, their augmentation with the ability to handle all sorts of objects autonomously is of great importance. Although mobile manipulators are now common in robotics research labs, manipulation planning for objects of complex shapes is still considered a challenging task, as it requires the control and coordination of both arms and hands (Furui and Nilanjan 2008).

Reach-to-grasp is the task of moving the arm all the way toward a plausible grasp of the object to be manipulated. It is one of the most important sub-tasks required for manipulation, as it is the starting point of any object manipulation task. For robotic arms, reach-to-grasp planning incorporates several subtasks, such as: grasp-planning (i.e. searching for a feasible and plausible grasping pose), where it is essential to guarantee that the grasping configuration synthesized is accessible to the robot, it must not lead to collision (neither self-collision nor against the environment), and it must be stable according to a chosen relevant stability criterion (Saut and Sidobre 2012); solving the IK problem; and finally, finding a collision-free path. Moreover, planning must not just achieve a solution, but also be performed in reasonable time.

2.6.1 Approaches for Reach-to-Grasp Planning

Several approaches have been suggested for reach-to-grasp planning. They mostly differ by the grasp-planning mechanism, i.e., how they approach the grasp generation subtask, and how they integrate grasps into the path planning process.

The classical approach for reach-to-grasp planning is to decouple the problem into two subtasks: grasp-planning and path-planning. The synthesis, evaluation, and specification of a set of desired end-effector goal poses, is often done beforehand. During run-time the planner integrates the set of desired end-effector poses into path planning (Weghe, Ferguson and Srinivasa 2007). The search is conducted in the robot's configuration space, and it is biased toward the desired end-effector goal poses, specified in the workspace.

To formulate sets or regions of desirable workspace end-effector goal poses, different grasp generation and approximation methods exist. In Miller et al (2003), objects are decomposed into their constituting elements, which are then approximated using different shape primitives (spheres, cylinders, cones and boxes). Grasps are generated based on these primitives and evaluated using a force closure (FC) measure. Another shape approximation was suggested using decomposition of objects based on a minimum-volume bounding box (MVBB) algorithm (Huebner, Ruthotto and Kragic 2008).

A different approach suggested by (Vahrenkamp, Asfour and Dillmann 2012), doesn't require any preprocessing of the object, in the sense that the only information it uses is the robot's configurations, and the object's location and spatial structure. The Grasp-RRT algorithm presented in their work is conducted during run-time, where it generates grasping-hypotheses, evaluates and selects feasible grasps, solves the IK problem, and searches for collision-free paths.

Methods that minimize the search during run-time using pre-computed desired goal poses, are prone to grasp evaluation inaccuracies due to their shape-approximation based grasp generation. In addition some require the human to explicitly specify goal regions or shape primitive of the different object elements. Although Grasp-RRT overcomes these issues, it incurs additional run-time computation. In our work, it is assumed that both run-time planning time and grasp success are of high importance.

2.6.2 Graspability Maps

A different approach for grasp generation was suggested (Roa, et al. 2011) based on a evaluation of poses that lead to high-quality grasps. In semi-structured environments, where the information regarding the model (shape and size) of object is available, such information is useful for speeding up the run-time planning.

A robust grasp must bring the object to standstill with respect to the gripper, under all forces and torques expected during its manipulation. Force closure (FC) grasps, which are based on the geometric condition of the closure property, have been extensively studied and used in the context of robotics (Zhu and Ding 2004) (Miao, Wenyu and Xiaoping 2010). In the case of soft contact gripper fingers, force closure of 3D objects can be achieved using two contact points, and can be measured by force closure angle (FCA) (Figure 15) (Eizicovits and Berman 2014):

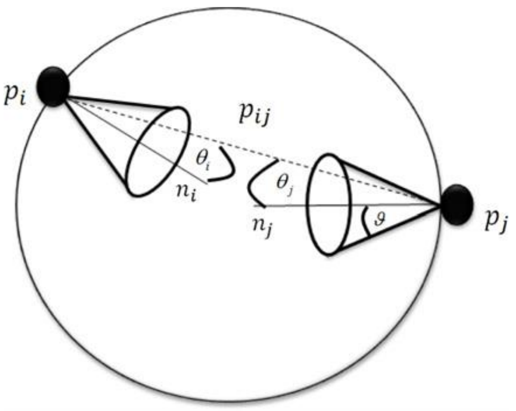

Figure 15 – Force closure, based on friction and the relative direction of contacts.
p_i and p_j are the two contacts, p_{ij} is the line of sight between them, n_i and n_j are the normal vectors and θ_i and θ_j are the contact angles (the angle between the normal and p_{ij}) of p_i and p_j, respectively. ϑ is the friction angle calculated from the static friction coefficient. (Eizicovits and Berman 2014).

Stability distance (SD) is an additional measure for evaluating grasp stability (Karmon, Flash and Edelman 1996). It is calculated as the Euclidean distance between the center of mass and the straight line that connects the two contact points imposed by the gripper fingers (Figure 16), normalized by the fingers width.

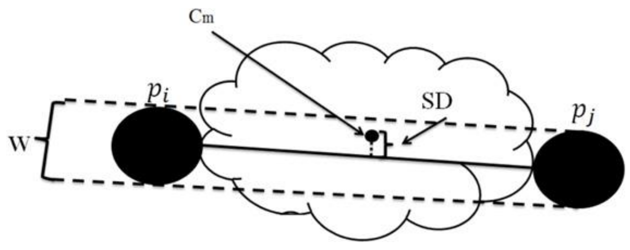

Figure 16 – The SD measure.
p_i and p_j are the two contacts, c_m is the object's center of mass, w is the fingers width and SD is the stability distance.
(Eizicovits and Berman 2014)

The graspability map is a map that represents, for a particular object and gripper, the quality grades of grasp from each pose about the object. For examples the quality grade can be a weighted average of variants of FCA and SD measures. The graspability map can be constructed from a 3D point cloud representation of an object that can be derived from standard depth sensors. The construction is done starting with a scan of the object's surface for contact points, following the calculation of wrist poses required for the grasp (Eizicovits and Berman 2014).

2.6.3 Workspace Goal Regions

Originally, RRT requires a single and pre-defined goal configuration. In cases where the task allows a continuous set of goal poses for the gripper, a matching set of goal configurations exists in the configuration space and should be used for the planning. However, the translation of such set of end-effector poses into the arm's configuration space using IK can be infeasible for run-time applications (Bertram, et al. 2006). In such cases, restricting a randomized planner to a partial finite set of goal configuration by applying IK in advance to planning (Hirano, Kitahama and Yoshizawa 2005) is also undesired, as it revokes the probabilistic completeness guarantee.

Workspace Goal Regions (WGRs) is an alternative concept presented by (Berenson, et al. May, 2009), which allows the specification of continuous regions in the 6D workspace of the end-effector poses as goals for the path-planner. This way, it avoids the translation of regions from the workspace to the configuration space and maintains probabilistic completeness throughout planning.

A single WGR is defined as a 6D volume in workspace, where it specifies continuous ranges in the TCP location and orientation dimensions, which, according to task specifications, express the desirable end-effector goal poses. The assumption is that it is relatively easier and more intuitive to formulate the regions of all desired end-effector goal poses in the workspace (in the object's coordinate-frame) rather than in the arm's configuration space. That way, the planner probabilistically allows all plausible goal configurations and thus preserves probabilistic completeness. The WGR formulation occurs prior to actual path planning, and therefore it doesn't take into account pose feasibility in terms of collisions with the environment.

Let be w a single WGR, it is specified by its boundaries matrix B^w (2) and its reference transform in world coordinates T_w^0, where T_i^j indicates the homogenous transformation of poses from the ith coordinate-frame to the jth coordinate-frame (3)

$$B^w = \begin{bmatrix} x_{min} & x_{max} \\ y_{min} & y_{max} \\ z_{min} & z_{max} \\ \psi_{min} & \psi_{max} \\ \theta_{min} & \theta_{max} \\ \phi_{min} & \phi_{max} \end{bmatrix} \tag{2}$$

$$T_i^j = \begin{bmatrix} R_i^j & t_i^j \\ 0 & 1 \end{bmatrix} \tag{3}$$

The IKBiRRT planner, an extension of RRT, was also presented in (Berenson, et al. May, 2009). Simultaneously to the expansion of RRT tree, the IKBiRRT samples goal poses from within WGRs, projects them onto world-coordinate-frame, validates feasibility against the environment and translates valid goal poses into goal configuration using IK. As a result, the number of roots in the goal tree of the bi-directional RRT is constantly growing, allowing additional feasible goal configurations for the path.

In the presence of multiple WGRs, sampling the WGR to be used, w, is done randomly w.r.t WGRs volumes. Grasp pose parameters (position coordinates and orientation angles) are then sampled uniformly in B^w to form d_{sample}. Using Euler convention d_{sample} is translated into the homogenous transformation matrix T^w_{sample} and then into world coordinate frame T^0_{sample}:

$$T^0_{sample} = T^0_w \cdot T^w_{sample} \cdot T^w_e \quad (4)$$

Figure 17 depicts the coordinate frames and transformation involved. The transformation T^w_e is used in cases where and offset exists between the last arm's joint and the TCP. Once validated for feasibility, T^0_{sample} is translated into the arm's configuration space, using IK, to serve as an additional root in the RRT goal tree.

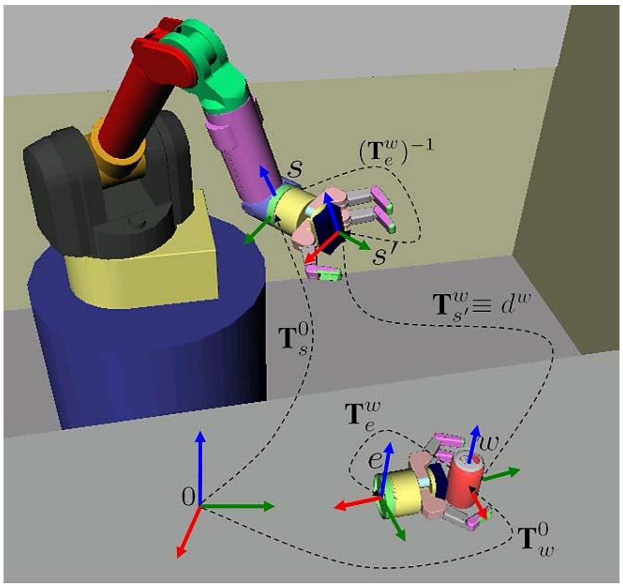

Figure 17 - The transformations and coordinate frames involved in using WGR for planning.

e is the end-effector offset (TCP to last joint), *s* is the coordinate frame of the last joint in a sample configuration. Taken from (Berenson, et al. May, 2009)

As an extension of the bi-directional RRT algorithm, IKBiRRT has proven efficiency and robustness at high dimensional workspaces (Bertram, et al. 2006). Yet, the uniform sampling method assumes all poses within WGRs are of equal quality. In some domains, in particular object grasping, different poses offer different grasp qualities, motivating the use of a more complex sampling scheme.

2.7 Statistical Inference

Statistical inference is the process of drawing conclusions from data that are subject to random variation. Whereas descriptive statistics describe a sample, inferential statistics infer predictions about a population of interest via some form of sampling. Any statistical inference requires some assumptions. A statistical model is a set of assumptions concerning the generation of both the sample data and the population itself (Bartoszyński and Niewiadomska-Bugaj 2008).

In fully parametric statistical inference, the probability distributions describing the data-generation process are assumed to be known and to involve a finite set of unknown parameters. Thus, the aim of such inference is to estimate the value of those parameters with respect to the sample data. Once the distribution parameters are estimated, it can be used to represent the data-generation process. For example, new samples can be drawn from the distribution with the estimated parameters, in order to imitate a behavior of interest.

2.7.1 Multivariate Gaussian Mixture Model

The multivariate Gaussian distribution is a generalization of the one-dimensional Gaussian distribution to higher dimensions (Izenman 2008). A k-dimensional multivariate Gaussian density function is:

$$g(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}} exp\left(\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T\,\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right) \qquad (5)$$

Where $\boldsymbol{\mu}$ is a k-dimensional vector of the expectation values in all dimensions, **x** is a n × k matrix of the observations, and $\boldsymbol{\Sigma}$ is the k × k covariance matrix.

A Gaussian Mixture Model (GMM) is a parametric model with a probability density function represented as a weighted sum of Gaussian component densities. It assumes all observations have been drawn from a mixture of a finite number of multivariate Gaussian distributions with latent parameters according to the following density function:

$$p(\boldsymbol{x}) = \sum_{i=1}^{M} w_i \cdot g(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_\mathbf{i}) \qquad (6)$$

Where **x** is a n × k matrix of the observations, w_i is the weight (prior probability) of the ith component, and $g(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_\mathbf{i})$ is the probability of the observations in x according to the multivariate Gaussian density function of the ith component.

Inferring GMM parameters from the dataset can be done using the Expectation-Maximization (EM) algorithm (Dampster, Laird and Rubin 1977), which trains the model with the goal of maximizing the likelihood of the data.

2.7.2 Inferring the Number of GMM Components

In its basic form, it is essential to provide the EM algorithm with the number of components (Gaussians) in the mixture. When there is imperfect knowledge concerning the number of components, literature suggests using GMM-BIC (Andrews and Lu 2001), an integration of the Bayesian Information Criterion (BIC) penalty criterion into GMM, to infer the optimal number of components based on the likelihood function of the fitted model.

2.8 Cluster Analysis

Cluster analysis is the process of grouping subsets of data into groups (clusters) so that all data within a cluster have high similarity in comparison to one another, but are dissimilar to data in other clusters. Clustering analysis has been studied both as a branch of statistics and as a type of *unsupervised learning* in the field of machine learning, as it does not rely on labeled observations.

Clustering can be achieved by various algorithms that differ in their notion of what constitutes a cluster and how to efficiently learn them. Traditional centroid-based clustering algorithms, among them are the popular k-means (Lloyd 1982) and its variations, are based on distance functions and aim to discover clusters that include groups with small distances among the cluster members, grouped around some points in space.

2.8.1 Minimum Volume Bounding Boxes

Often, a different form of clusters is required, such as box decomposition. In their work, (Geidenstam, et al. 2009) suggests enveloping 3D data points into box shapes by a fit-and-split algorithm that is based on an efficient Minimum Volume Bounding Box (MVBB) algorithm. An example is presented in Figure 18 based on data from Standford's bunny point cloud[3].

The clustering procedure is as follows:

- The data is first being projected onto 2D planes
- Planar split candidates are explored and evaluated for splitting each set of points into two subsets using 2D convex hulls, and the best candidate is chosen, heuristically.
- When the split according to the chosen candidate does not satisfy the termination condition, the dataset is divided into two, and the same procedure goes on for each subset recursively.

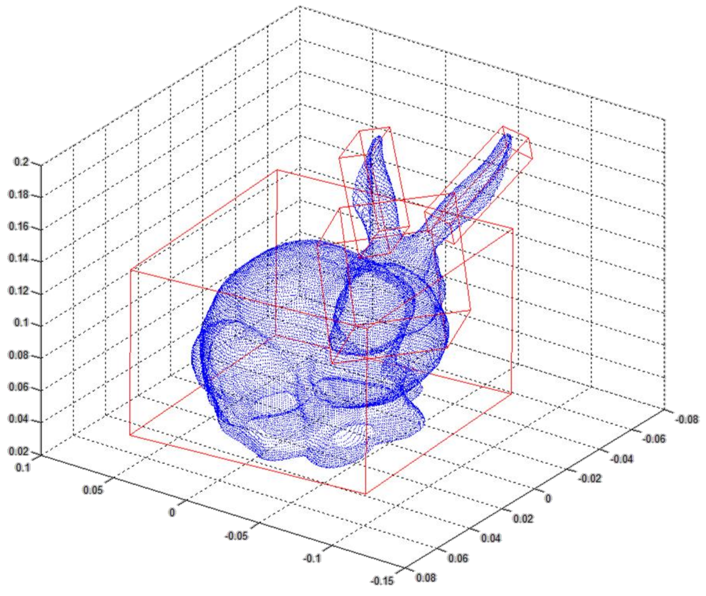

Figure 18 - MVBB decomposition for Stanford's bunny point-cloud

[3] http://www.mrbluesummers.com/3562/downloads/stanford-bunny-model

The termination condition is based on a threshold of the total volume minimization gain from each split according to:

$$\theta^* = \frac{V(C_1) + V(C_2) + V\left(A^{\setminus P}\right)}{V(P) + V(A^{\setminus P})} \tag{7}$$

Where V is a volume function, A is the complete set of boxes in the current hierarchy, P is the current box, and C_1, C_2 are the two child boxes that can be produced by the split, and θ^* is the total volume minimization gained by splitting according to the best split at the current hierarchy (Huebner, Ruthotto and Kragic 2008).

Although clustering with box shape primitives is not appropriate as a clustering technique for any data in general, it fits well in our work, as will be described later.

2.9 KD Tree for Nearest Neighbor Search

The nearest neighbor search is an optimization problem for finding the closest point to a given point from a dataset of candidates, where closeness can be expressed by various types of distance metrics. This problem is common across different fields of application, e.g., pattern recognition, databases, and spell checking. Various approaches have been developed to address this problem, such as locality sensitive hashing (Rajaraman and Ullman 2010) and space partitioning techniques (de Berg, et al. 1997).

KD-Tree is a space partitioning technique and a powerful data structure that is based on a recursive subdivision of a set of points based on alternating axis-aligned hyperplanes (Atramentov and LaValle 2002). This technique was applied to path planning problems in high dimensions with a great success, providing significant improvements in running times.

The data structure construction is done by recursively splitting datasets of points into two, according to a pivot point (often the median in the relevant axis). Each of the two child dataset is divided further, according to next dimension. In every step, the splitting hyperplane is perpendicular to the corresponding axis and passes through the pivot point (Figure 19). When the number of the data points in a dataset falls below a given threshold N_{max}, a “leaf node” associated with this dataset, which stores a list of coordinates for all the points in the dataset. Once the tree construction is done, a nearest neighbor search is done by depth-first search to eliminate non-relevant points in the dataset and minimize the number of candidate points for the actual distance calculation.

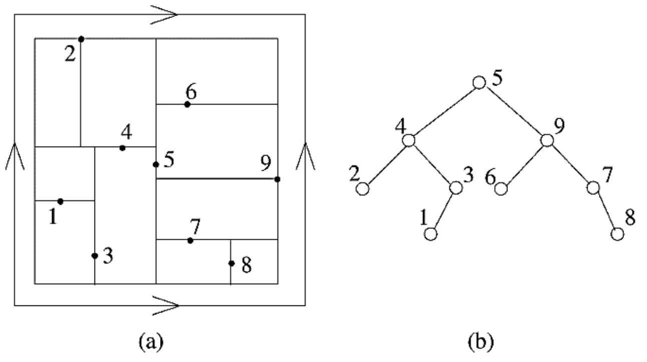

Figure 19 - An example of KD-Tree
(a) How the space is subdivided, (b) the corresponding tree. From (Yershova and LaValle 2007)

CHAPTER 3: REACH-TO-GRASP PLANNING FRAMEWORK

3.1 Overview

The current chapter details the GR-RRT methodology based on graspability map (see Section 2.6.2) and the workspace goal region (see Sub-Section 2.6.3). In our work, we bridge and extend some of the methods reviewed in chapter 2 to achieve the following advantages:

- **Accurate and automatic** representation of graspability knowledge from specific gripper-object simulation.
- **Fast run-time search** that integrates both path planning and grasp generation, with preliminary preprocessing efficiently used by the bi-directional RRT engine.
- **Consistency with RRT's probabilistic completeness** in path planning and grasp generation.

The chapter is organized as follows: Section 3.2 exhibits our novel framework for reach-to-grasp planning, and is generally divided into two sub-sections, where both simple and complex objects are handled. Next, Section 3.3 discusses general modifications to the planner that were required for better and more accurate representation of the continuous 3D workspace, and for better efficiency.

3.2 Grasp-Regions RRT

The Grasp Regions Rapid-exploring Random Trees (GR-RRT) is an extension of the IKBiRRT algorithm (see Sub-Section 2.6.3). As such, it exploits the concept of workspace goal regions to plan paths for robotic arms that are biased towards goal regions in the workspace, which are defined based on the task specifications. GR-RRT incorporates statistical inference and clustering techniques to allow automatic specification, based on a-priori knowledge, of such regions, which are ultimately been used to generate better grasp candidates.

The GR-RRT algorithm incorporates two phases (Figure 20): the grasp region (GR) specification phase, which is done a-priori, and a run-time path planning phase.

In the **GRs specification phase**, the characteristics of the object and gripper are used to form a graspability map. To facilitate the definition of the GR, the wrist poses are translated to tool center point (TCP) poses, similar to the T_e^w transformation from Section 02.6 (Figure 17). According to their corresponding grasp-quality grade, poses with grade lower than a task-specified threshold are filtered out of the database, as the desired behavior is of the successful grasps (i.e. those who hold high grasp-quality grade) only.

For a simple and symmetrical object, it is safe to assume that the dataset of all the remaining successful grasp candidates is gathered in a single cluster, thus a single GR will provide good results (Reshef, Eizicovits and Berman 2014). However, for a more complex object, the spatial behavior of successful grasps tends to change with respect to the different elements that constitute it. Following this, a rotation-translation decoupling is adopted from (Bazin, et al. 2010) so that a 3D box-based clustering procedure is applied to the data in the subspace of TCP location only. Clustering is done using a hierarchical minimum-volume-bounding-box (MVBB) decomposition from (Geidenstam, et al. 2009), originally used for shape approximation of complex objects (Section 2.8).

The result of the last step is an automatic decomposition of regions of successful grasps in the 3D subspace of the TCP locations, relative to the object, using boxes that tightly bound all the successful grasps in the original set. It is straightforward to use these boxes as the boundaries of WGRs in a multiple-regions scheme. However, with using this technique a tradeoff arises: setting a too high value for the threshold θ^* (the total volume minimization gain from each split, used for recursion termination) in the hierarchical MVBB algorithm tends to produce a high number of small boxes and therefore over fit the data, ignoring sub-regions of possibly successful grasps. Setting the value too low tends to produce a low number of bigger boxes that often capture different behaviors of the data ("distant" groups of poses), prone to harming the ability of virtually drawing successful grasps. Finally, θ^* values were selected based on an empirical study and visualization of the results.

To facilitate the full definition of the GRs, a within-cluster 6D spatial distribution is inferred using a Gaussian Mixture Model (GMM), so that later it can be used to generate grasp candidates. The underlying assumption in choosing a mixture of Gaussian components is that there exist a finite number of ultimate grasps, and symmetrical deviations from these grasps (in TCP location or orientation dimensions) will result in equal decrease in grasp-quality grades, with respect to correlations between the dimensions, that are accounted by GMM. To enable an automatic inference procedure, we have implemented GMM-BIC (see Sub-Section 2.7.2), so that the best GMM model is automatically selected in statistical fashion, through penalizing model complexity and accounting for the likelihood of the dataset given the model. The 6D GMM distribution in each cluster is bounded by its bounding-box boundaries, according to the minimum and maximum observed values of each dimension in the subset of grasp points that were assigned to it (a 6D axis-aligned bounding box).

In the **planning phase**, the GR-RRT algorithm exploits both sensory- processed information of the environment (i.e. the existence of objects, their positions and orientations) and the GRs specified in the previous phase. This information allows planning paths that are biased towards successful grasp-poses.

In a single-GR scheme (i.e. for simple and symmetrical objects), GR-RRT uses the GMM model to sample new poses from the GR (compared to the uniform sampling of IKBiRRT) to generate successful grasp candidates. Similarly to IKBiRRT, new grasps are sampled with a constant rate p_{sample} and translated into the configuration space using IK to serve as additional roots for the RRT goal tree, i.e. as candidates for the end of the reach-to-grasp path.

For complex objects multiple GR are required. In such cases, in each iteration of GR-RRT that involves grasp generation, begins with selecting the GR to be used out of the set of available GRs. Selection is made using weighted sampling of GRs, where each GR is weighted proportionally to the sum of the absolute differences between GR-bounds from B^w (2) across all 6 dimensions. This is done rather than according to GR volumes (a multiplication of the absolute differences), since a GR might encompass a difference of 0 in one or more dimensions (Berenson, et al. May, 2009), equating its 6-dimensional volume to zero, so it is totally ignored.

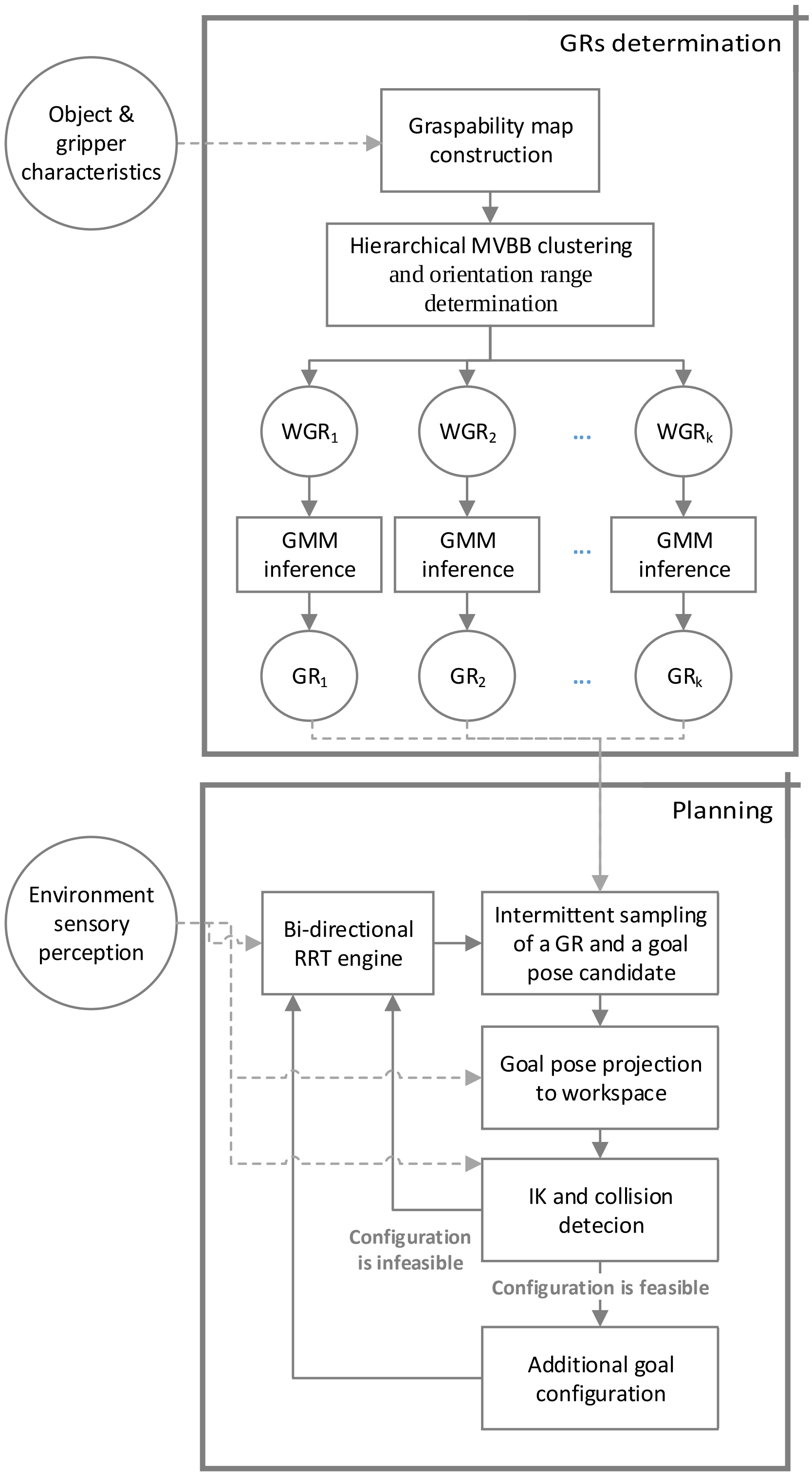
GRs determination
Object & gripper characteristics
Graspability map construction
Hierarchical MVBB clustering and orientation range determination
WGR$_1$
WGR$_2$
...
WGR$_k$
GMM inference
GMM inference
...
GMM inference
GR$_1$
GR$_2$
...
GR$_k$
Planning
Environment sensory perception
Bi-directional RRT engine
Intermittent sampling of a GR and a goal pose candidate
Goal pose projection to workspace
IK and collision detecion
Configuration is infeasible
Configuration is feasible
Additional goal configuration

Figure 20 – Multiple GR framework for reach-to-grasp planning

3.3 Additional Notes

3.3.1 Transitioning to a Continuous 3D Workspace

When accounting for a discrete 2D workspace, collision checking for any point on the robotic arm requires a computation of $O(1)$, i.e., by checking the relevant cell in a binary matrix of obstacle spatial representation in workspace with respect to their discrete coordinate values. It may be argued that a high-resolution 3D discrete representation of the workspace can approximate real environments. However, a high-resolution discrete representation necessitates an extremely big 3D matrix that often cannot be handled by standard computation infrastructure, while still failing to provide the required accuracy for sensitive applications. Therefore, an accurate robotic application must handle a continuous 3D workspace that naturally corresponds to the environment it operates in (Alton and Mitchell 2006).

The transition from 2D to 3D only involves a bigger workspace representation that can ultimately be handled (up to some level of resolution) by sparse-matrix representations (Gilbert, Moler and Schreiber 1991) or by up-scaling the computation infrastructure used for planning. But, the transition to a continuous space, for ensuring high accuracy, dictates the use of a totally different environment representation.

We address this issue by using a point cloud representation (Rusu and Cousins 2011), where every physical object in the environment contributes to the "cloud" a set of points (represented by continuous XYZ coordinate values) that together describe its surface. This is in fact a sparse representation of obstacles in the environment, based on a discretization of their surfaces that generates a subset of the points that constitute them. It can be easily generated using standard depth sensors (see Figure 18), and ultimately be controlled by a resolution parameter.

Consequently, the collision checking module must be adapted. To verify pose feasibility for the robotic arm, it is now required to calculate its Euclidean distance to the point-cloud of obstacles. Similarly to using a discrete workspace, a pose is considered feasible when a discrete set of points along the robotic skeleton lies entirely in the obstacles-free sub-space of the workspace. In a continuous workspace, points along the robotic skeleton are checked for their nearest neighbors in the point-cloud. A minimum distance threshold d_{min} is set, and a pose is considered valid once distances to all nearest neighbors are above this threshold.

The simplest approach to continuous-workspace collision checking involves $O(n_p)$ distance calculations for each point in the skeleton of the robot, where n_p is the number of points in the point-cloud. Needless to say, using this naïve approach makes planning with RRT extremely inefficient. Following that, a KD-Tree data structure (Section 2.9) is used to store all points that belong to the environment point cloud. Using this data structure, the complexity for single nearest neighbor query is $O(\log n_p)$, resulting in an extreme improvement of collision detection runtimes.

3.3.2 Path Smoothing Memory Matrix

In the smoothing technique, as well as in the classical technique, an iterative divide and conquer approach is applied to subsets of a path's set of vertices. Using this approach, same pairs of vertices might be validated more than once for a collision-free path between them. Collision-free validation is the most computation consuming part of this module. This issue induces an excessive use of computational resources, and can be avoided by storing a "memory matrix" holding either a null value for a pair that hasn't been checked yet, 0 for of vertices with no feasible straight path between them, and 1 for a pair of vertices with a feasible collision-free path between them. This modification is implemented further in this work as part of the triple smoothing method.

CHAPTER 4: TESTING GR-RRT

4.1 Overview

We have conducted an experiment to evaluate The GR-RRT planning algorithm. The GR-RRT is compared to the IKBiRRT planner with a multiple WGRs scheme, based on the external shape of the objects (point cloud). A *home environment* scenario, based on real-life problems that can be solved by robotic applications, was simulated. This environment contained two target-objects: a mug and a frying pan, both require a multi-GR sampling scheme. Additionally, *selective apple harvesting* environment was also simulated. The target-objects in this environment were apples, which mandated a simple single-GR sampling scheme. The *selective apple harvesting* experiment is detailed in **Appendix II**. The *home environment scenario* experiment is detailed in the current chapter. Section 4.2 and Section 4.3 introduce the experimental setup and the simulation environment that were set for the experiment, respectively. Next, the metrics and the statistical analyses that were used are detailed in Section 4.4 and Section 4.5. Following that, Section 4.6 brings the experiment results, which are further discussed and concluded in Section 4.7.

4.2 Experimental setup

The simulations were conducted using a PC equipped with an Intel i7-3770K 3.5 GHz processor (CPU) and 32GB installed memory (RAM), running on Windows 8.1 (64-bit). The development and execution of the experiment was done using MATLAB (Version 2011a, Mathworks, USA) and analysis was done using IBM SPSS Statistics (Version 19, IBM, USA).

The MATLAB Parallel computing toolbox was used to reduce the computational time of the preliminary learning procedure. The MATLAB Robotic toolbox was used for a visualization of the robotic arm model, with modifications.

4.3 Simulation Environment

A 3D virtual environment was constructed based on a physical setup. It comprised a virtual model of our six degrees-of-freedom (DOF) manipulator (UP6, MOTOMAN, Japan), a table and several wooden cubes as obstacles, and 2 objects: a mug and a frying pan (Figure 21). Both object are composed of several asymmetrical elements, and therefor constitute complex objects to grasp. A total of 6 different compositions were created using the above elements, assuring diverse reachability of the objects (Figure 22). Point cloud representations of the environment were created, as well as for the two objects used as targets. In both algorithms implemented, a point cloud of the environment was used for collision detection, using KD-tree (see Sub-Section 2.9) based collision detection module.

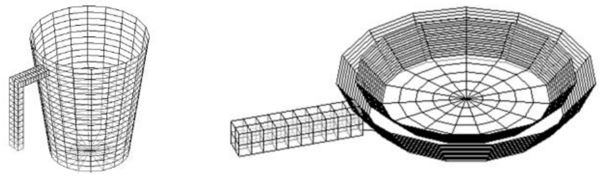
Figure 21 - Models of a mug and a pan used in our simulations as target objects

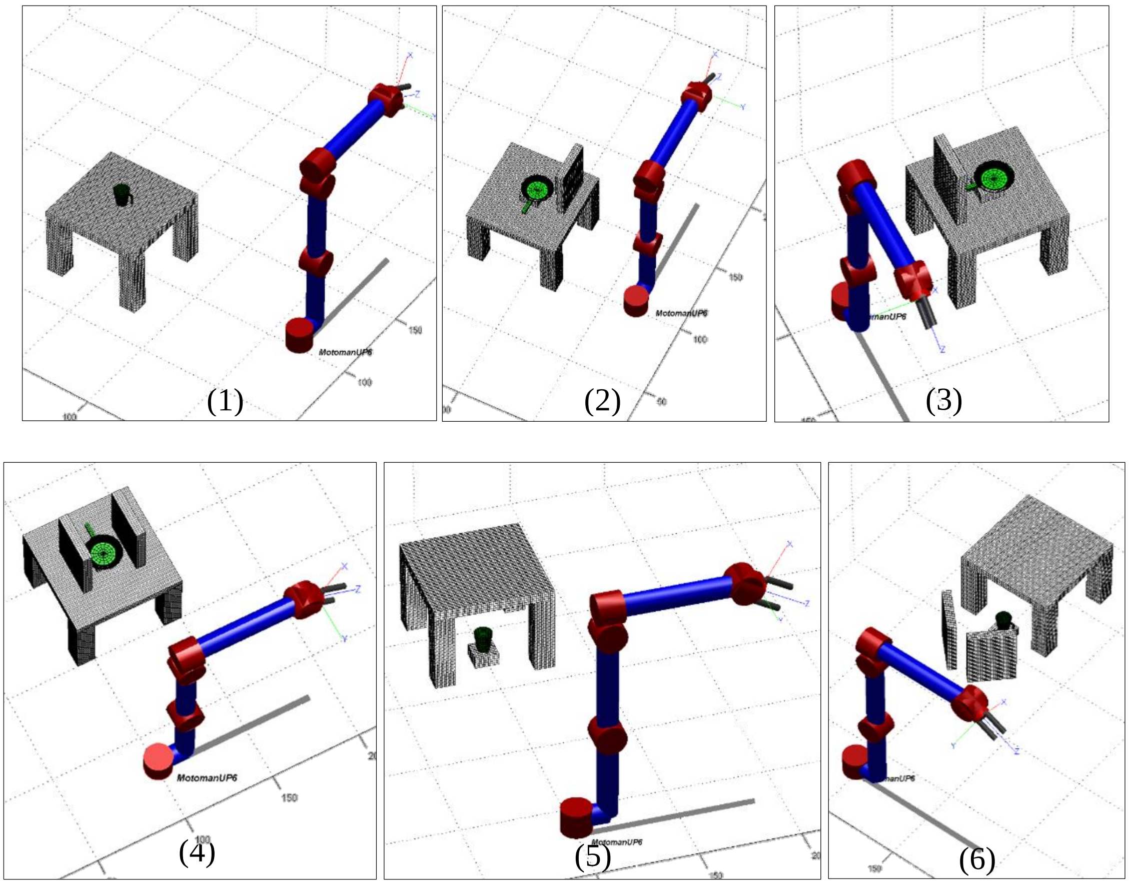

Figure 22 - The six different environments served as problems to be solved by the planner. In each composition, both mug and pan objects were located in the same position and orientation

For both algorithms (IKBiRRT, GR-RRT), a preliminary phase execution was required in order to construct the sampling scheme used for planning. In GR-RRT, the GRs-scheme was generated following the construction of the object's graspability map and based on the data stored in it. In IKBiRRT, the MVBB algorithm was applied to the object's point cloud to formulate WGRs that ultimately bound the object's shape. Both schemes were constructed and once per object.

Planning with both algorithms was executed 100 times per $algorithm \times environment\ composition \times object$ combination. Each resulting path was smoothed by an implementation of a triple path smoothing (see Appendix I).

4.4 Analysis

Algorithm performance was evaluated and compared in terms of computation time, path quality, and final grasp quality. Path quality was quantified by the final path length in the configuration space, using two normalized distance measures: Euclidean (ND_e) and City-Block (ND_{cb}):

$$ND_e = \frac{\sum_{i=2}^{V} \sqrt{\sum_{j=1}^{J} \left(x_{i,j} - x_{i-1,j}\right)^2}}{\sqrt{\sum_{j=1}^{J} \left(x_{1,j} - x_{V,j}\right)^2}} - 1 \qquad (8)$$

$$ND_{cb} = \frac{\sum_{i=2}^{V} \sum_{j=1}^{J} \left|x_{i,j} - x_{i-1,j}\right|}{\sum_{j=1}^{J} \left|x_{1,j} - x_{V,j}\right|} - 1 \qquad (9)$$

Where $x_{i,j}$ (deg.) is the ith vertex's position in the jth dimension of the configuration space, V is the total number of vertices in the path, and J is the total number of joints in the arm.

ND_e and ND_{cb} are based on the Euclidean distance and the city-block distance, respectively. Both measures express the efforts for executing a path, i.e., the total distance of the path, relative to a (not necessarily feasible) lower bound path distance, i.e., the line-of-sight from the initial vertex to the final vertex, which is reflected in the denominator. Additionally, they contain a subtraction of 1 for simpler statistical analysis, which makes them range from 0 (the path is the actual line-of-sight from initial vertex to final vertex) to infinity (the path is highly convoluted, where the value express the additional length with respect to the lower bound distance). The two distance measures (Euclidean, city-block) are used for a thorough analysis of paths, where Euclidean distance better correlates with path length, and city-block distance represents the sum of the total distance travelled in all joints, which implies the effort required in terms of joints actuation.

Grasp quality was compared based on success, i.e., binary representation (if the path ended with a grasp of quality of 0.7 or above, success was determined as one, zero otherwise). The threshold value of 0.7 was used based on preliminary experiments with hardware and physical objects.

4.5 Statistical Analysis

All analyses were conducted with the following predictors: algorithm, object, and their interactions, as the goal was to compare the performance of the two algorithms in terms of different objects. Environment-id was used as a random effect in the following mixed models to remove the influence of the different environment settings.

Computation time was analyzed using a generalized linear mixed model with a gamma link-function. A gamma link function was used as values were non-negative and strongly skewed to the right. Grasp success was analyzed using a mixed logistic regression model.

The fact that both algorithms produced considerable amount of paths that were the line-of-sight to the target pose (both ND_e and ND_{cb} equal 0) required that the analysis of path quality measures will be broken into two. At first, we created the binary variable $IsLOSe$ to compare the rate of line-of-sight paths, so $IsLOSe = 1$ when the Euclidean distance is the same as the line-of-sight (i.e. $ND_e = 0$), and $IsLOSe = 0$ otherwise. We analyzed $IsLOSe$ using a mixed logistic regression model. Following, we removed all line-of-sight paths and analyzed both ND_e and ND_{cb} of the remaining subset of paths using a generalized linear mixed model with a gamma link-function, due to non-negative and right-skewed values.

4.6 Results

The a-priori learning of IKBiRRT's WGRs scheme was based on the mug and pan point-clouds and resulted in 6 and 5 WGRs respectively. 3D TCP location boundaries relative to objects, are visualized in Figure 23.

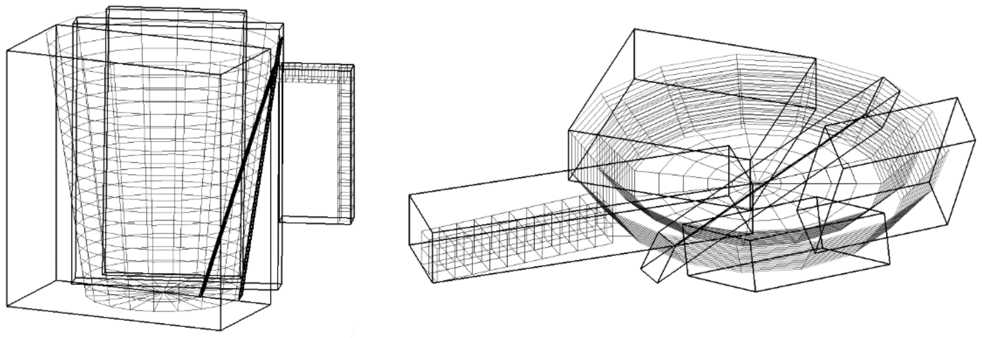

Figure 23 - WGRs location boundaries of mug (left) and pan (right), as used in the IKBiRRT planning algorithm for sampling

Learning of GR-RRT's GRs scheme was based on each object's graspability map (partially visualized in Figure 24. For a full visualization of all graspability maps generated please refer to Appendix II). The sets of successful grasps for mug and pan objects held 6,619 and 24,652 gripper poses (Figure 25), MVBB clustering resulted in 13 and 35 GRs (Figure 26) and GMM inference resulted in numbers of components in the ranges of 6-71 and 2-99, respectively.

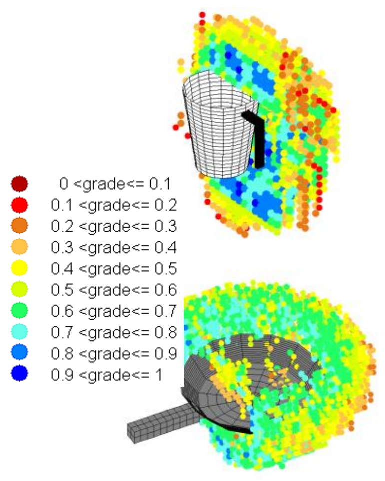

Figure 24 – Partially visualized graspability maps of mug (up) and pan (down)

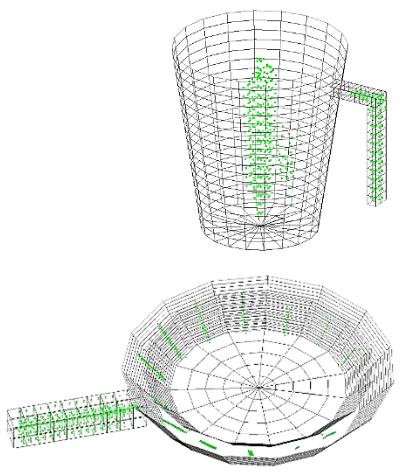

Figure 25 - Extracted TCP positions of successful grasps (green) for mug (upper figure) and pan (lower figure), produced by

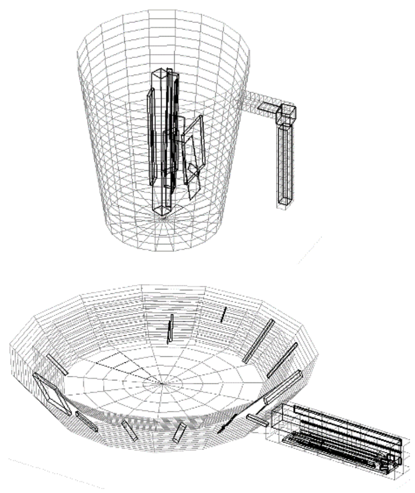

Figure 26 - GRs location boundaries of mug (upper figure) and pan (lower figure), produced by GR-RRT

Both algorithms had a path-planning success of 100% with the applied planning limitation of 20,000 iterations of RRT engine per execution.

For grasp success, the target-object id and interactions between target-object and algorithm were found to be non-significant and thus were removed from the model. The final model included only algorithm as a fixed effect and environment-id as a random effect (Table 1). While results for IKBiRRT showed an average odds ratio of 0.006, which reflects a grasp success rate of ~0.6%, GR-RRT was found to increase odds ratio by 983% on average, providing an average grasp success rate of 85.5% (142 times greater). Final grasps were also visualized for validation by human eye. It can be generally seen (Figure 27) that grasps of high grade are in fact successful grasps of the object that are close to the ones often picked by humans, compared to grasps with low grade.

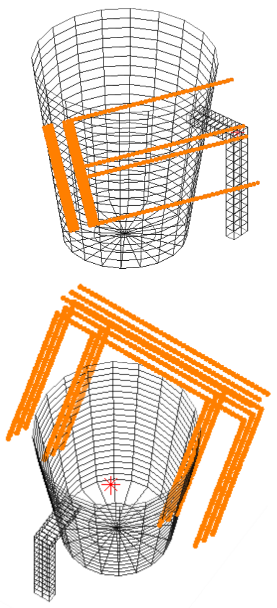
Figure 27 - Example of a bad grasp (up) and a successful grasp (down) of a mug found by GR-RRT.

Table 1 - Logistic regression results for Grasp Success

Grasp Success				
Fixed Effects	**Coef (Standard Error)**	**95% Confidence Interval for Odds ratio**		
		Lower Bound	**Odds Ratio**	**Upper Bound**
Intercept	-5.116 (0.416)***	[0.003]	[0.006]	[0.014]
Algorithm = GR-RRT	6.890 (0.418)***	433.3%	982.7%	2228.7%

*** p<0.001 ** p<0.01 * p<0.05

For **planning time**, the only significant factor was the interaction $\{Object = Pan \times Algorithm = GR - RRT\}$, this means that a significant increase in planning time using GR-RRT was found only in the case of planning towards a grasp of the pan (Table 2). In this case, the average increase in planning time using GR-RRT is of 13.4% compared to the IKBiRRT planning towards a grasp of the mug.

Figure 28 shows a box-plot chart of planning times distribution across all 6 environment compositions.

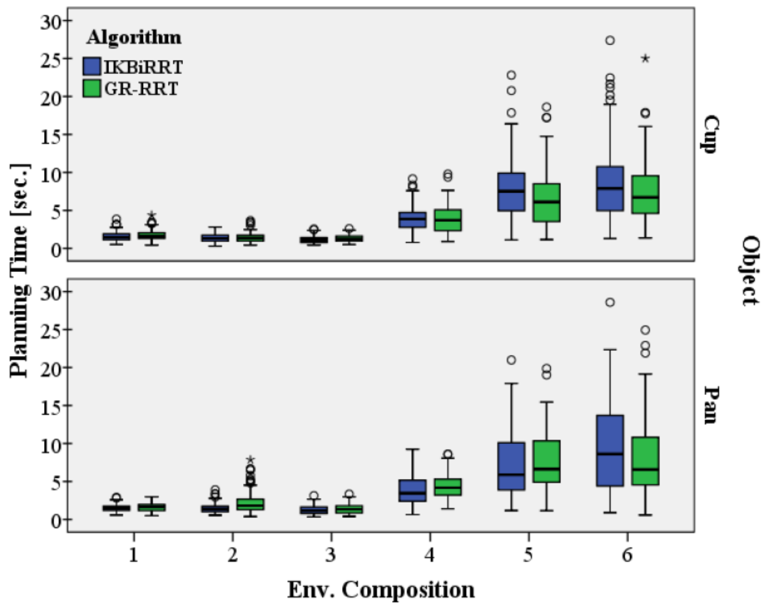

Figure 28 - Algorithm Planning Time boxplot chart by environment-id and target-object

Table 2 - Generalized linear regression results for Planning Time

Planning Time				
Fixed Effects	**Coef. (Standard Error)**	**95% Confidence Interval for Planning-Time Ratio**		
		Lower Bound	**exp(Coef.)**	**Upper Bound**
Intercept [sec.]	0.445 (0.025)***	[1.484]	[1.560]	[1.640]
Algorithm = GR-RRT	-0.012 (0.029)	93.3%	98.9%	104.7%
Object = Pan	0.011 (0.029)	95.4%	101.1%	107%
[Algorithm = GR-RRT] * [Object=Pan]	0.126 (0.041)**	104.6%	113.4%	123.1%

*** p<0.001 ** p<0.01 * p<0.05

For **path quality,** 45.5% of the paths generated had $IsLOSe = 1$, i.e. the path was the actual line-of-sight paths to the target configuration, where all paths in environments #1 (with no obstacles) and #3 (with the handle of the mug/pan towards the cube obstacle) were with $IsLOSe = 1$. In the analysis of $IsLOSe$, the only significant factor was the interaction $\{Object = Pan \times Algorithm = GR - RRT\}$, which means that a significant increase in the odds ratio of line-of-sight paths is obtained for the pan when planning with GR-RRT (Table 3).

Table 3- Mixed logistic regression results for IsLOSe

Line-Of-Sight Rate (IsLOSe)				
Fixed Effects	**Coef. (Standard Error)**	**95% Confidence Interval for Odds ratio**		
		Lower Bound	**exp(Coef.)**	**Upper Bound**
Intercept [O.R.]	2.758 (0.271)***	[9.260]	[15.761]	[26.826]
Algorithm = GR-RRT	-0.144 (0.269)	51.1%	86.6%	146.6%
Object = Pan	-0.326 (0.270)	42.5%	72.2%	122.5%
[Algorithm = GR-RRT] * [Object=Pan]	1.786 (0.390)***	277.8%	596.7%	1281.6%

*** p<0.001 ** p<0.01 * p<0.05

In the analyses of both ND_e and ND_{cb} on the subset of the remaining 54.5% curved paths, none of the predictors were found significant. Average values (for all paths generated) for both ND_e and ND_{cb} were approximately 0.66, means that the average path is 66% longer than the lower bound “line of sight”. The effect of path quality measures is illustrated in Figure 29.

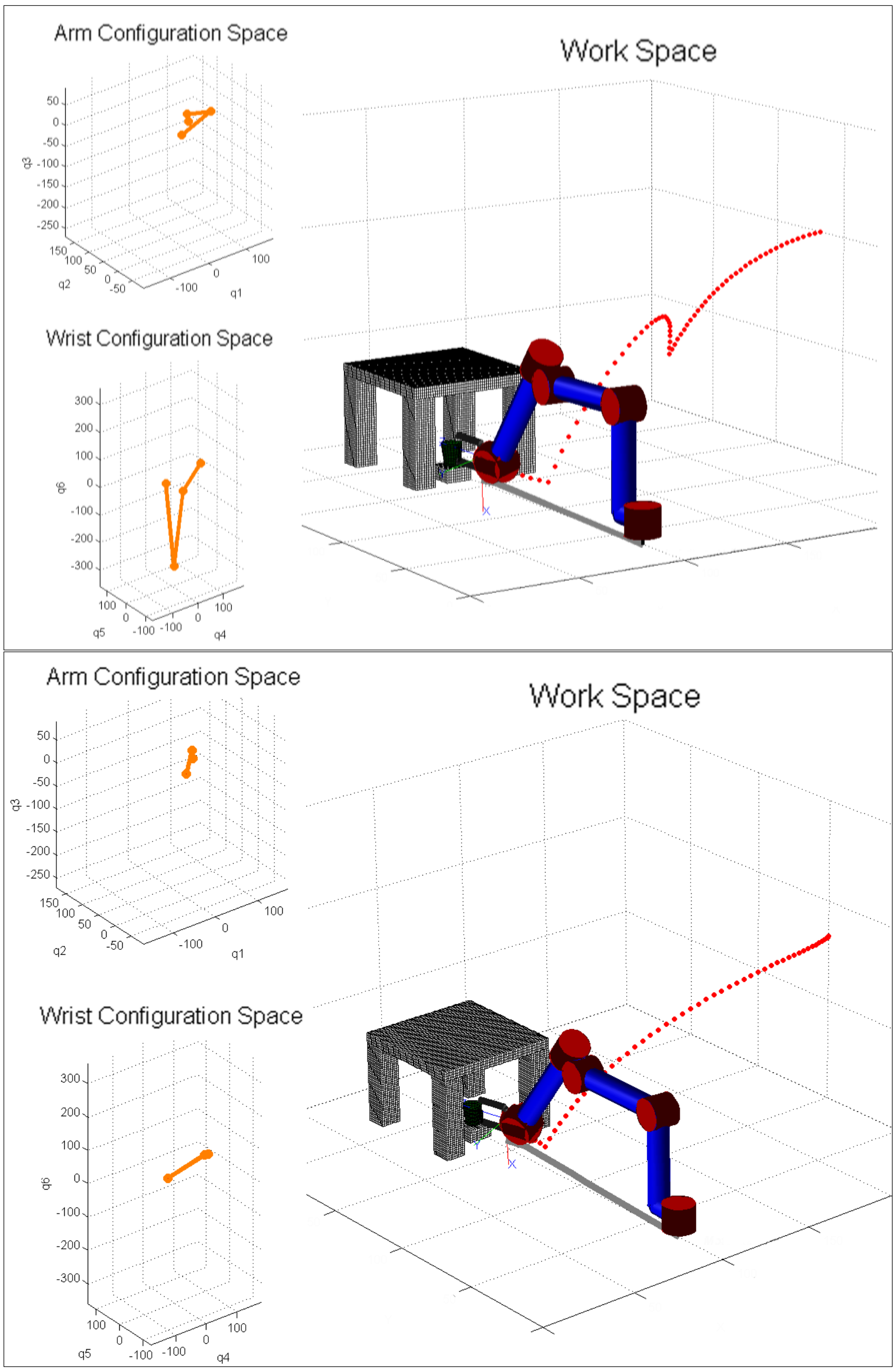

Figure 29 - Illustration of Two solutions found by GR-RRT algorithm for the 5th environment. In each figure, the configuration space is presented on the left (two 3D sub-figures), and the manipulator's movement (in workspace) is illustrated on the right. The upper figure shows a solution of ND_e=2.77, ND_{cb}=2.38. The lower figure shows a solution of ND_e=0.04, ND_{cb}=0.02.

4.7 Discussion

Results suggested that comparing to IKBiRRT GR-RRT achieves a drastic improvement in grasp success without any decrease in path finding success and with only a small increase in computation time and path length for difficult-to-reach targets. According to the results, the drastic improvement in grasp success is achieved with only a small increase in computation time and no increase in path length (in cases when GR-RRT planned toward the pan object, a decrease in path length was spotted).

Both algorithms use the same bi-directional RRT engine, so differences, tem only from the different sampling schemes and the data they are based on. Therefore, the underlying premise of both experiments (and of GR-RRT in general) is that improvement in the quality and accuracy of planning can be gained by:

a. Augmenting the planner with a-priori knowledge regarding grasps.
b. Exploiting the knowledge by integrating it into a randomized planner to bias its random sampling toward the goal (successful grasps).

GR-RRT is augmented with the data stored in a graspability map that is derived a-priori from simulations (and therefore does not affect run-time planning times). It exploits the set of high quality grasps from within the map to infer its distribution in a statistical fashion, so it can be imitated in sampling of gripper goal poses for planning. Assuming the (general shape of the) object to be grasped is known in advance, it has been shown that GR-RRT indeed gains such improvements, and despite the additional overhead, it maintains RRT's short planning times. If the manipulator is required to handle a variety of objects of different sizes and shapes relatively-similar objects may be grouped to reduce preliminary learning times.

The GR-RRT has several parameters that require tuning. In the current work their values were selected empirically. Parameter optimization, can further improve accuracy, planning times, and suitability to exceptional objects and/or scenarios. The parameters for tuning are: parameters deduced by resolution of objects in environment, e.g. RRT's increment size ϵ (Sub-section 2.4.3), collision-detection's threshold value d_{min} (Sub-sections 2.4.3, 3.3.1), both trade accuracy in detecting collision for planning-times; KD-tree parameter N_{max} (Sub-section 2.9) and GR-RRT's rate of grasp-pose sampling p_{sample} (Section 3.2), both affect planning query's latency; and hierarchical MVBB's termination condition threshold θ^* (Sub-section 2.8.1, Section 3.2) that affects grasp generation accuracy.

CHAPTER 5: VALIDATION IN HARDWARE

Hardware validation was conducted based on a 6 DOF manipulator (UP6, MOTOMAN, Japan) in the Tele-robotics lab, the Department of Industrial Engineering and Management, Ben Gurion University of the Negev (Figure 30). For execution of results, we have used MOTOCOM32 drivers to connect our Matlab implementation of GR-RRT to the robot's control-unit.

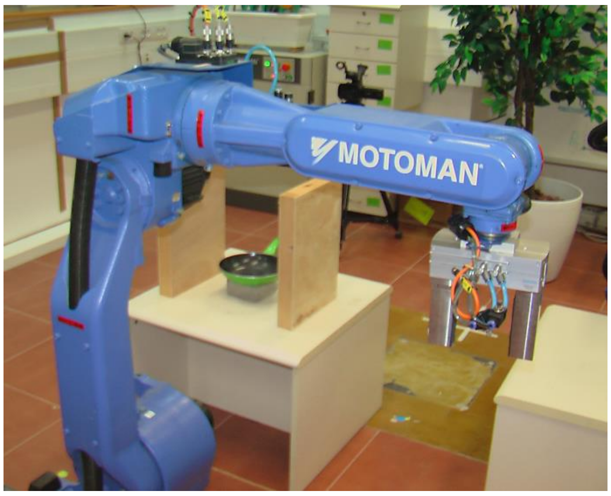

Figure 30 – Motoman UP6 Manipulator used for validation in hardware

The lab environment were arranged to match the simulation settings of environments #3 and #4 (Section 4.2). In accordance with the simulation, two target-objects were used: a frying pan and a mug. The following figures (Figure 31, Figure 32) illustrate two reach-to-grasp paths generated by GR-RRT, where the robot successfully executed a pick-an-place task based on the planned paths.

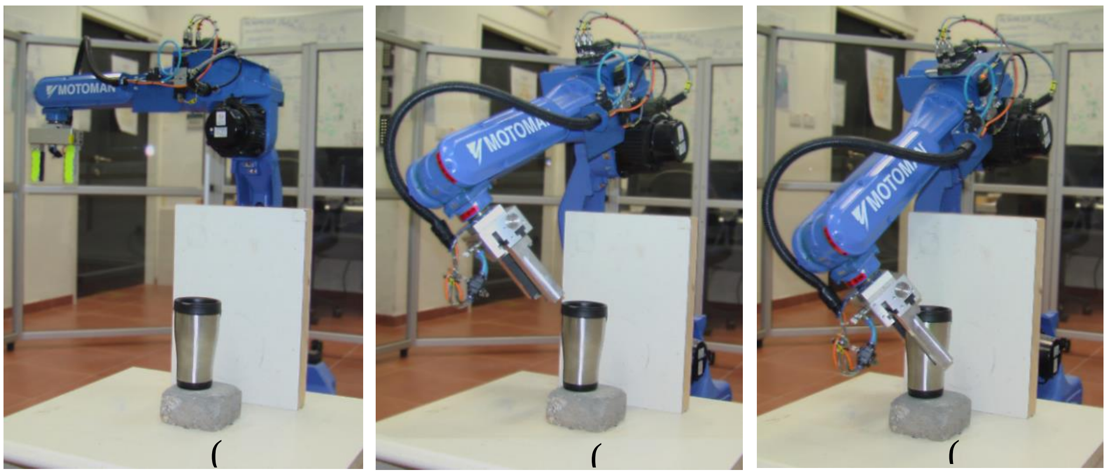

Figure 31 – Illustration of reach-to-grasp of a mug with the UP6 manipulator
(a) initial position, (b) offset pose relative to the final grasp, (c) final grasp.

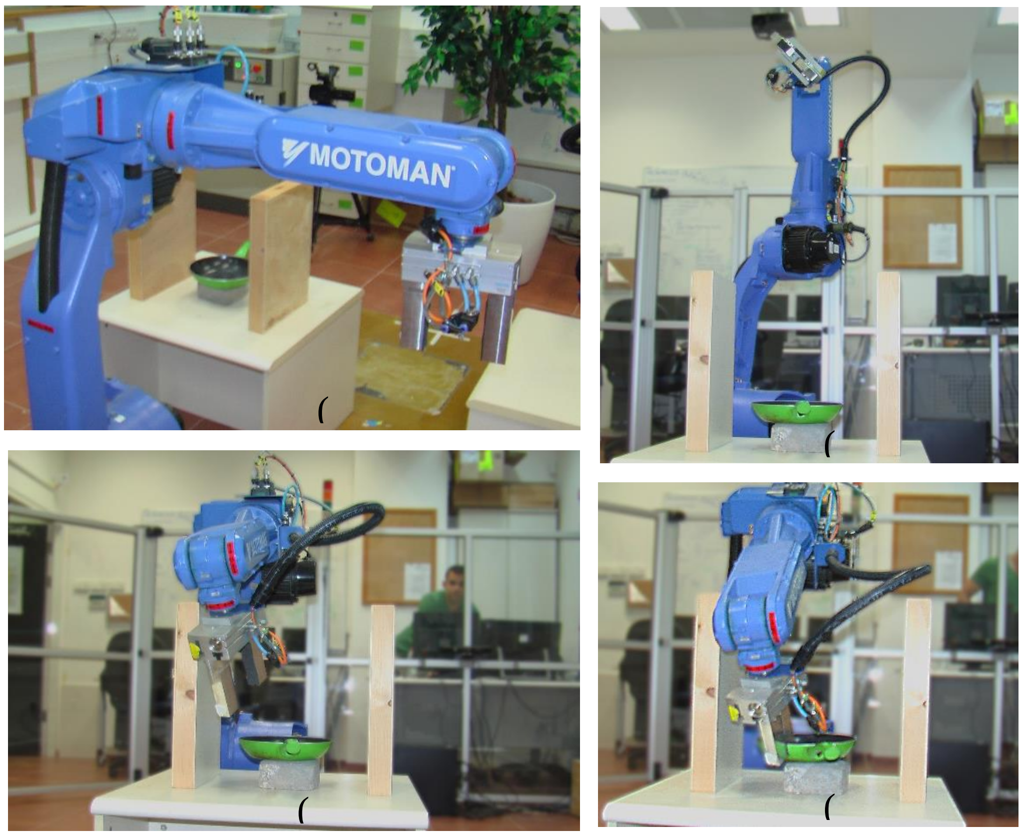

Figure 32 – Illustration of reach-to-grasp of a pan with the UP6 manipulator (a-b) initial position, (c) the offset pose relative to the final grasp, (d) final grasp.

Since the workspace point cloud included the obstacles in the workspace alone (without the object itself), the generated paths often ended with one of the gripper's fingers colliding the object. To tackle this, goal-pose generation of GR-RRT was modified so that poses generated included an offset of twice the length of gripper's fingers in a direction normal to the grasp (Figure 31(b); Figure 32(c)), to allow a final approach motion of the gripper directly towards the original grasp pose (Figure 31(c); Figure 32(d)). Another solution would be to integrate the target-object's point-cloud into planning, with a lower customized threshold $d_{min}^{(object)}$ for collision detection between the gripper and the object to allow final grasps. Although the latter constitutes a better solution in terms of completeness in planning, an additional overhead is expected, following the subtle (restrictive) freedom of motion in the actual grasping motion.

Finally, the actual UP6 arm consisted elements that weren't modeled, so in some paths generated, small elements near the end effector (such as wires) that broke the symmetrical shape of the arm collided with the obstacles (Figure 33). A comprehensive implementation for this specific hardware should take into account all constituting elements of the arm and make sure the collision detection threshold d_{min} (sub-sections 2.4.3, 3.3.1) is set high enough to prevent undesired collisions with obstacles in the environment. Alternatively collision avoidance during motion should be integrated.

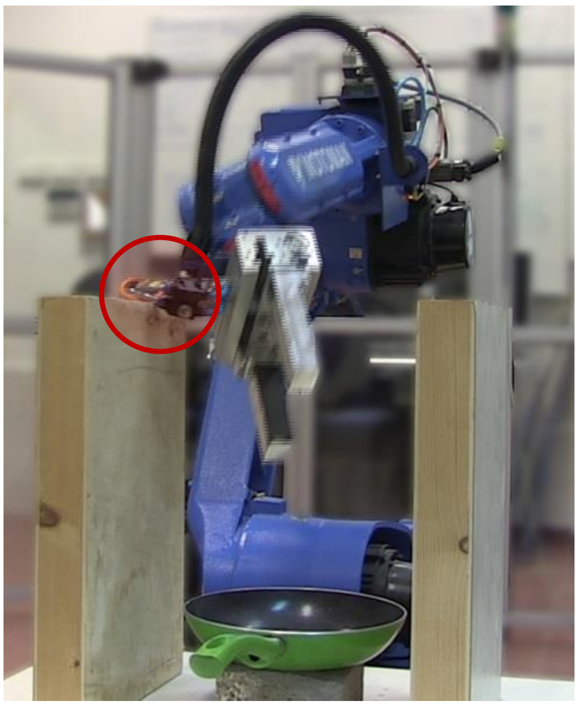
Figure 33 - Wires near the gripper colliding with an obstacle

CHAPTER 6: SUMMARY AND CONCLUSION

This current research developed an algorithm for robotic reach-to-grasp motion. The algorithm was initially developed for apple harvesting and later extended to deal with arbitrary objects. The main contributions of this thesis are:

1. **An improved path smoothing method** that achieves a decrease in planning time over heuristics found in literature. The use of a "memory matrix" for further improvement of efficiency by avoiding repetitive computations was additionally suggested.
2. **A novel framework for reach-to-grasp planning,** GR-RRT was developed. It achieves a dramatic improvement in grasps quality over previous methods. It also maintains probability completeness guarantee of both path planning and grasp generation by exploiting bi-directional RRT and graspability maps characteristics. Its implementation included additional techniques to improve performance, among them a state-of-the-art collision detection technique based on KD-Tree data structure and the developed path smoothing method, to complement planning by the randomized RRT planner.

The developed method included the integration of grasp synthesis into the planning through a preliminary phase dedicated for learning the spatial distribution of the subset of gripper poses that represent successful grasps of objects. To achieve this, statistical inference and machine learning techniques such as GMM and hierarchical MVBB clustering were used. To complement reach-to-grasp planning, the GR-RRT planning algorithm was proposed.

Results of GR-RRT evaluation suggest that GR-RRT provides a drastic improvement in grasp success in all scenarios with respect to the IKBiRRT algorithm, with small increase in computation time and path length in some scenarios. In addition GR-RRT automates the procedure of defining grasp regions. GR-RRT is suitable for both simple and complex objects and is especially suitable for scenarios in which determining grasp poses is not trivial.

Several issues were identified throughout the research and designated for future research:

- **Using all graspability data for inference** including un-successful grasps data, which is currently filtered out before the inference of grasps spatial distribution. Success in doing so can be of great influence on the accuracy of sampled grasps for planning (run-time phase).
- **Object reaching strategy.** Validation of the experiment in hardware showed that the planning algorithm must account for gripper-object collisions, in addition to gripper-environment collisions. We implemented a "final approach" strategy by offsetting the goal pose of the gripper and creating an additional linear motion toward the object to be executed last (Chapter 5), although better approaches may be developed to address this issue.
- **Adjusting GR-RRT for more-DOF manipulators.** Currently, the method is suitable for six (or less) DOF manipulators, as it relies on the fact that IK has a finite number of solutions. It is possible to adjust the method to handle redundant manipulators as well, e.g., with an approach similar to the one demonstrated in

(Bertram, et al. 2006). Degradation in performance is expected (based on single tree expansion).

- **Fine-tuning model parameters** was out of the scope of this research. Such tuning is expected to improve obtained results.
- **Further improvements to RRT's efficiency and planning quality** – several improvements to RRT planning engine were developed in the last years, e.g., parallelizing RRT by utilizing multi-core CPUs (Ichnowski and Alterovitz 2012) or GPUs (Bialkowski, Karaman and Frazzoli 2011), or trading planning time for better results (Salzman and Halperin 2013). The latter may be modified to minimize a cost function that accounts for grasp quality, in addition to path length.

REFERENCES

Abbadi, A., and R. Matousek. 2012. "RRTs Review and Statistical Analysis." *International journal of mathematics and computer in simulation.*

Alton, K., and I. M. Mitchell. 2006. "Optimal path planning under defferent norms in continuous state spaces." *Proceedings 2006 IEEE International Conference on Robotics and Automation.* Orlando, FL: IEEE. 866-872.

Andrews, D. W. K., and B. Lu. 2001. "Andrews and B. Lu, "Consistent model and moment selection procedures for GMM estimation with application to dynam." *Journal of Econometrics* 123-164.

Archambault, P., P. Pigeon, A. G. Feldman, and M. F. Levin. 1999. "Recruitment and sequencing of different degrees of freedom during pointing movements involving the trunk in healthy and hemiparetic individuals." *Experimental Brain Research* 55-67.

Atramentov, A., and S. M. LaValle. 2002. "Efficient Nearest Neighbor Searching for Motion Planning." *International Conference on Robotics & Automation.* Washington, DC: IEEE. 632-637.

Bac, C. W., T. Roorda, R. Reshef, S. Berman, J. Hemming, and E. J. Van Henten. 2014. "Analysis of a motion planning problem for fruit harvesting in a dense obstacle environment." *Submitted for publication.*

Bartoszyński, R., and M Niewiadomska-Bugaj. 2008. *Probability and Statistical Inference.* New Jersey: Wiley.

Bazin, J. C., C. Demonceaux, P. Vassuer, and I. S. Kweon. 2010. "Motion estimation by decoupling rotation and translation in catadioptric vision." *Computer Vision and Image Understanding* 114 (2): 254-273.

Berenson, D., S. Srinivasa, D. Ferguson, A. C. Romea, and J. Kuffner. May, 2009. "Manipulation Planning with Workspace Goal Regions."

Bertram, D., J. Kuffner, R. Dillmann, and T. Asfour. 2006. "An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators." *Proc. IEEE International Conference on Robotics and Automation (ICRA).* 1874-1879.

Bialkowski, J., S. Karaman, and E. Frazzoli. 2011. "Massively parallelizing the RRT and RRT*." *IEEE/RSJ Internation Conference on Intelligent Robots and Systems (IROS).* San Francisco: IEEE. 3513-3518.

Bruno, S., and K. Oussama. 2008. *Handbook of Robotics.* Berlin: Springer.

Bulanon, D. M., T. Kataoka, Y. Ota, and T. Hiroma. 2002. "Automation and Emerging Technologies: A Segmentation Algorithm for the Automatic Recognition of Fuji Apples at Harvest." *Biosystems Engineering* 405-412.

Carpin, S., and G. Pillonetto. 2005. "Motion Planning Using Adaptive Random Walks." *IEEE Transactions on Robotics* 129-136.

Chazelle, B. 1987. "Approximation and decomposition of shapes." *Algorithmic and Geometric Aspects of Robotics* 145-185.

Dampster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society* 1-38.

de Berg, M., M. van Kreveld, M. Overmars, and O. Schwarzkopf. 1997. *Computational Geometry: Algorithms and Applications.* Berlin: Springer.

Eizicovits, D., and S. Berman. 2014. "Efficient sensory-grounded grasp pose quality mapping for gripper design and online grasp planning." *Robotics and Autonomous Systems* 62 (8): 1208–1219.

Furui, W., and S. Nilanjan. 2008. "Supervisory Controller Design for a Robot-Assisted Reach-to-Grasp Rehabilitation task." *30th Annual International IEEE EMBS Conference.* Vancouver: IEEE. 4258-4261.

Geidenstam, S., D. Huebner, D. Banksell, and D. Kragic. 2009. "Learning of 2D Grasping Strategies from Box-Based 3D Object Approximations." *Proceedings of Robotics: Science and Systems.*

Gilbert, J. R., C. Moler, and R. Schreiber. 1991. "Sparse matrices in Matlab: Design and implementation." *SIAM Journal on Matrix Analysis and Applications.*

Hague, T., J. A. Marchant, and N. D. Tillett. 1997. "Autonomous Robot Navigation for Precision Horticulture." *Proc. IEEE International Conference Robotics and Autionmation.* Albuquerque, New Mexico: IEEE. 1880-1885.

Harada, K., E. Yoshida, and K. Yokoi. 2010. *Motion Planning for Humanoid Robots.* London: Springer.

Hicks, J. 2012. *Intelligent Sensing Agriculture Robots To Harvest Crops.* August 06. http://www.forbes.com/sites/jenniferhicks/2012/08/06/intelligent-sensing-agriculture-robots-to-harvest-crops/.

Hirano, Y., K. Kitahama, and S. Yoshizawa. 2005. "Image-based Object Recognition and Dexterous Hand/Arm Motion Planning Using RRTs for Grasping in Cluttered Scene." *IEEE/RSJ International Conference on Intelligent Robots and Systems* 2041-2046.

Hsu, d., T. Jiang, J. Reif, and Z. Sun. 2003. "The bridge test for sampling narrow passages with probabilistic roadmap planners." *IEEE international conference on robotics and automation (ICRA).*

Huebner, K., S. Ruthotto, and D. Kragic. 2008. "Minimum Volume Bounding Box Decomposition for Shape Approximation in Robot Grasping." *IEEE International Conference on Robotics and Automation* 1628-1633.

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

Izenman, A. J. 2008. *Modern Multivariate Statistical Techniques.* New York: Springer.

Jyh-Hwa, T., and K. L. Su. 2008. "The development of the restaurant service mobile robot with a Laser positioning system." *27th Chinese Control Conference.* Kunming: IEEE. 662-666.

Karaman, S., and E. Frazzoli. 2011. "Sampling-based algorithms for optimal motion planning." *The International Journal of Robotics Research (IJRR)* 846-894.

Karmon, I., T. Flash, and S. Edelman. 1996. "Learning to grasp using visual information." *IEEE International Conference on Robotics and Automation.* Minneapolis, MN: IEEE. 2470-2476.

Kavraki, L. E., P. Svestka, J. C. Latombe, and M. H. Overmars. 1996. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *IEEE Transactions on Robotics and Automation* 566-580.

Khanmohammadi, S., A. Mahdizadeh, and M. T. VakilBaghnlisheh. 2008. "A new technique for optimizing and smoothing randomized paths." *IEEE International Joint Conference on Neural Networks* 1704-1708.

Khatib, O. 1986. "Real Time Obstacle Avoidance for Manipulators and Mobile Robots." *International Journal of Robotics Research (IJRR)* 90-98.

Kim, J., and W. Chung. 2013. "Range Sensor-Based Localization of Mobile Robots in Semi-Structured Environments." *10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI).* Jeju: IEEE. 219-221.

Kuffner, J., and S. LaValle. 2000. "RRT-Connect: An Efficient Approach to Single-Query Path Planning." *IEEE international conference on robotics and automation (ICRA).* 995-1001.

Latash, M. L., and F. Lestienne. 2006. *Motor Control and Learning.* New York: Springer US.

Latombe, J. C. 1991. *Robot Motion Planning.* Botons: Kluwer Academic Publishers.

LaValle. 2006. *Planning Algorithms.* Cambridge: Cambridge University Press.

—. 1998. "Rapidly-exploring random tree: A new tool for path planning." *Computer Science Dept., Iowa State University.* Oct. http://janowiec.cs.iastate.edu/papers/rrt.ps.

Lloyd, S. 1982. "Least squares quantization in PCM." *IEEE Transactions on Information Theory* 28: 129-137.

Lumelsky, V. J., and A. Stepanov. 1987. "Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape." *Algorithmmica* 403-430.

Miao, L., Y. Wenyu, and Z. Xiaoping. 2010. "Projection on Convex Set and Its Application in Testing Force Closure Properties of Robotic Grasping." *Third International Conference of Intelligent Robotics and Applications (ICIRA).* Shanghai, China: Springer. 240-251.

Miller, A. T., S. Knoop, H. I. Christensen, and P. K. Allen. 2003. "Automatic Grasp Planning Using Shape Primitives." *Proc. IEEE Int. Conf. Robotics and Automation (ICRA).* 1824-1829.

Prenzel, O., J. Feuser, and A. Graser. 2005. "Rehabilitation Robot in Intelligent Home Environment - Software Architecture and Implementation of a Distributed System." *9th International Conference on Rehabilitation Robotics (ICORR).* IEEE. 530-535.

Rajaraman, A., and J. Ullman. 2010. *Mining of Massive Datasets.* Cambridge: Cambridge University Press.

Reif, J. H. 1979. "Complexity of the mover's problem and generalizations." *Proc. IEEE 20th Annual Symp. Foundations of Computer Science (SFCS 1979).* Washington, DC. 421-427.

Reshef, R., D. Eizicovits, and S. Berman. 2014. "Path Planning of Grasp-Aimed Robotic Tasks using Rapid-exploring Random Trees." *Robotics and associated High-technologies and Equipment for Agriculture and forestry (RHEA).* Madrid: RHEA.

Roa, M. A., K Herkorn, F. Zacharias, C. Borst, and G. Hirzinger. 2011. "Graspability Map: A Tool for Evaluating Grasp Capabilities." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* San Francisco, CA: IEEE. 1768-1774.

Rodriguez, S., T. Xinyu, L. Jyh-Ming, and N. M. Amato. 2006. "An Obstacle-Based Rapidly-Exploring Random Tree." *Proceeding of the 2006 IEEE International Conference on Robotics and Automation.* Orlando, FL: IEEE. 895-900.

Rusu, R.B., and S. Cousins. 2011. "3D is here: Point Cloud Library (PCL)." *IEEE International Conference on Robotics and Automation (ICRA).* Shanghai: IEEE. 1-4.

Salzman, O., and D. Halperin. 2013. "Asymptotically near-optimal RRT for fast, high-quality, motion planning." *arXiv.*

Saut, J. P., and D. Sidobre. 2012. "Efficient models for grasp planning with a multi-fingered hand." *Robotics and Autonomous Systems* 347-357.

Spong, M. W., S. Hutchinson, and M. Vidyasagar. 2005. *Robot Modeling and Control.* New York: John Wiley & Sons, Inc.

Takahashi, K., and Y. Mitsukura. 2012. "An Entertainment Robot Based on Head Pose Estimation and Facial Expression Recognition." *Proceedings of SICE Annual Conference (SICE).* Akita: IEEE. 2057-2061.

Urmson, C., and R. Simmons. 2003. "Approaches for heuristically biasing RRT growth." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE. 1178-1183.

Vahrenkamp, N., T. Asfour, and R. Dillmann. 2012. "Simultaneous Grasp and Motion Planning: Humanoid Robot ARMAR-III." *IEEE Robotics & Automation Magazine* 19 (2): 43-57.

Van Henten, E. J., B. A. J. Van Tuijl, J. Hemming, J. G. Kornet, J. Bontsema, and E. A. Van Os. 2003. "Field Test of an Autonomous Cucumber Picking Robot." *Biosystems Engineering* 305-313.

Waringo, M., and D. Henrich. 2006. "Efficient Smoothing of Piecewise Linear Paths with Minimal Deviation." *IEEE/RSJ International Conference on Intelligent Robots and Systems.* 3867-3872.

Weghe, M. V., D. Ferguson, and S. S. Srinivasa. 2007. "Randomized Path Planning for Redundant Manipulators without Inverse Kinematics." *7th IEEE-RAS International Conference on Humanoid Robots.* Pittsburgh, PA. 477-482.

Yershova, A., and S. M. LaValle. 2007. "Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching." *IEEE Transactions on Robotics* 151-157.

Zhu, X., and H. Ding. 2004. "Planning foce-closure grasps on 3D objects." *Proc. IEEE Int. Conf. Robotics and Automation.* IEEE. 1258-1263.

APPENDIX I – PLANNING CORE AND SMOOTHING METHOD

Overview

It was previously suggested that the performance of the goal-biased RRT and the Bi-directional RRT differ generally for some applications. Thus, this chapter is dedicated to evaluation of the performance of the two RRT variations introduced in Section 2.4, to a modification to the path smoothing algorithm (see Section 2.5), and to a preliminary experiment we conducted to evaluate two RRT planning alternative versions and the two path smoothing methods.

Triple Path Smoothing

A *classical path smoothing* heuristic based on divide-and-conquer technique was presented in the previous chapter. A manual inspection of a variety of smoothed solutions by an implementation of this method (combined with RRT) discovers unpleasing results in terms of final path length, relative to the best solution can be found by a human eye.

In this section we introduce the *triple smoothing*, a modified version of the classical path smoothing, which is based on the same technique but is less-indulgent. The tradeoff between low computation time and promising a proximity to the optimal solution requires checking a subset of proposed solutions out of all the possible solutions, and pick the best out of this subset. We suggest "investing" an additional runtime to check on a wider subset of solutions, in the following way: if the subset of nodes V contains up to k (a predetermined constant value) nodes, check not only the two subsets that result from splitting V into two halves, but also the two subsets that result from shifting the split one node to the right, and also the two subsets that result from shifting the split one node to the left, in V. The pseudo-code for the modified version is presented in Figure 34.

```
$\boldsymbol{TRIPLE\_SMOOTH(V, first, last, S)}$
  1. $if\ (first = last)\ then$
    1.1.    $S.pushback(V[first])$
  2. $else\ if\ (first = last - 1)\ then$
    2.1.    $S.puchback(V[first])$
    2.2.    $S.puchback(V[last])$
  3. $else\ if\ \begin{pmatrix} the\ segment\ connecting\ V[first] \\ and\ V[last]\ lies\ entirely\ in\ C_{free} \end{pmatrix} then$
    3.1.    $S.puchback(V[first])$
    3.2.    $S.puchback(V[last])$
  4. $else$
    4.1.    $TRIPLE\_SMOOTH(V, first, (first + last)/2, S_{Middle})$
    4.2.    $TRIPLE\_SMOOTH(V, (first + last)/2, last, S_{Middle})$
    4.3.    $if\ (|V| < k)\ then$
      4.3.1.    $TRIPLE\_SMOOTH(V, first, (first + last)/2 - \mathbf{1}, \boldsymbol{S_{ShiftLeft}})$
      4.3.2.    $TRIPLE\_SMOOTH(V, (first + last)/2 - \mathbf{1}, last, \boldsymbol{S_{ShiftLeft}})$
      4.3.3.    $TRIPLE\_SMOOTH(V, first, (first + last)/2 + \mathbf{1}, \boldsymbol{S_{ShiftRight}})$
      4.3.4.    $TRIPLE\_SMOOTH(V, (first + last)/2 + \mathbf{1}, last, \boldsymbol{S_{ShiftRight}})$
      4.3.5.    $S \leftarrow the\ subset\ from\ \{\boldsymbol{S_{Middle}}, \boldsymbol{S_{ShiftLeft}}, \boldsymbol{S_{ShiftRight}}\}$
                    $whose\ size\ is\ the\ smallest$
    4.4.    $else$
      4.4.1.    $S \leftarrow \boldsymbol{S_{Middle}}$
```

Figure 34 – Pseudo code for the triple smoothing method

Preliminary Experiment: Planning Engine and Smoothing

We constructed an experiment to evaluate two variations of the RRT algorithm: a goal-biased version and bi-directional expanding version. We have combined those two versions each with two path smoothing algorithms, one at a time: the classical path smoothing method taken from (Carpin and Pillonetto 2005), and our triple path smoothing method from the previous sub-chapter.

Simulation Environment

The test involved designing obstacles maps, and executing each of the algorithms trying to solve them, i.e. provide feasible paths according to maps specifications. The manipulator used was a rigid 2D robotic arm with 3 links and 3 degrees of freedom. Constant initial configuration and goal configuration were chosen for each map, such that a feasible path that connected the two existed.

Maps created consisted of 4 different variations for each level of complexity – Low (Figure 35), Medium (Figure 36), and High (Figure 37), a total of 12 maps. These maps were implemented as 1000x1000 matrices of bits, indicating 0 where a free pixel lies in space and 1 where a pixel is part of an obstacle. Obstacles shapes were chosen to be rectangular, and complexity levels were defined by the density of the obstacles in the map.

Hardware and Software

We have used a PC machine equipped with an Intel i7-3770K 3.5 GHz processor (CPU) and 16GB installed memory (RAM), running on Windows 8 (64-bit). The development and execution of the experiment was done using MATLAB (Version 2011a, Mathworks, USA) and analysis was done using IBM SPSS Statistics (Version 19, IBM, USA).

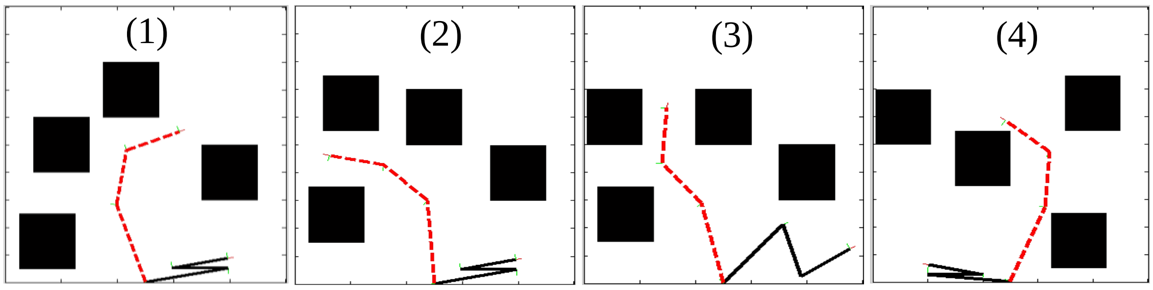

Figure 35 - Low obstacles density maps (1-4).
Solid black lines indicate the robot at its initial configuration,
dashed red lines indicate the robot at its goal configuration.

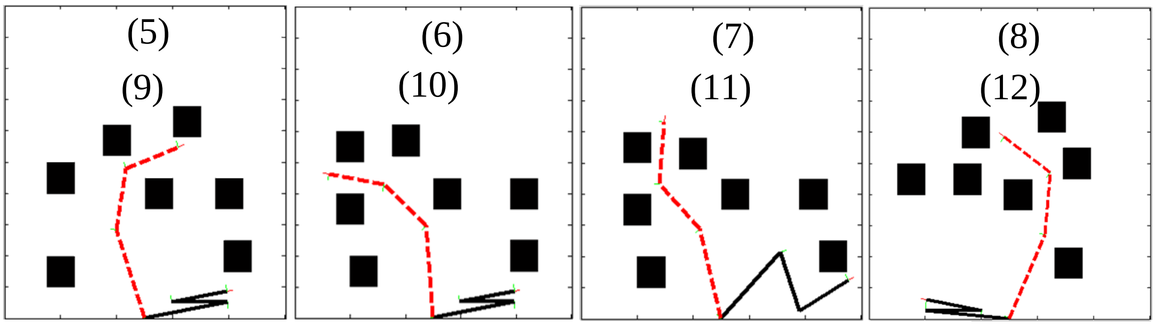

Figure 36 - Medium obstacles density maps (5-8)
Solid black lines indicate the robot at its initial configuration,
dashed red lines indicate the robot at its goal configuration.

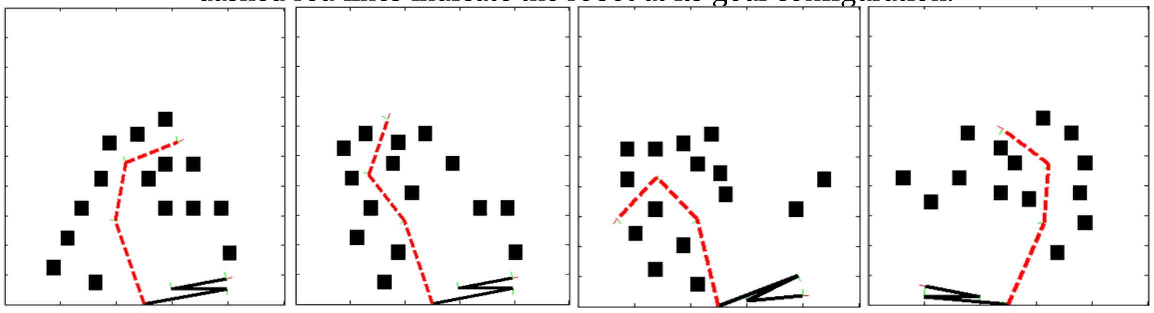

Figure 37 - High obstacles density maps (9-12)
Solid black lines indicate the robot at its initial configuration,
dashed red lines indicate the robot at its goal configuration.

Implementation

The code included a main function which received a map implementation (matrix of bits) as input and called the other modules, e.g. Extend, Connect, Path-Smoothing, and more.

The two versions of the RRT algorithm required setting a resolution parameter ϵ for the step size in each expansion of the tree. The tradeoff here is between low resolution value (higher ϵ), which will result in shorter computation time but can provide an incorrect solution where the middle of a robot link is colliding with an obstacle, and high resolution value (lower ϵ), which will more likely to guarantee a truly collision-free path, but will lead to higher computation time. Preliminary runs were executed in order to determine an appropriate value for this parameter. A value of $\epsilon = 10$ successfully produced collision-free solutions on a 1000x1000 pixel maps, and was high enough to provide short computation time solutions.

The goal-biased version required setting another parameter: the K parameter, defined as the rate of biasing the tree towards the target (so that the tree is biased towards target configuration with the probability of K). By manually exploring several values for K, and in accordance to literature (Urmson and Simmons 2003), we set it to be 10%.

Finally, a total of 4 possible combinations existed (two RRT versions and two path smoothing methods). For each combination, we ran the simulation 10 times per each one of our 12 pre-generated environment maps. This resulted in a total of 480 independent observations.

Analysis

The analysis was based on 7 measures divided into two categories: computation (efficiency) and quality. The computation category included all the measures related to the memory consumption, computation time, and computational burden, and the quality category included measures related to the quality of the solution found, such as its length and the angles the joints must travel to execute it. All measures were averaged per each level of density of the obstacles. We focused on both computational time and path quality because together they both affect the total path execution time, i.e., the time passes from the moment the problem has been full specified until the arm’s motion is completed. The tradeoff between low computation time and the quality of the resulted path shows the complexity of defining the best version of the algorithm and tuning its parameters.

Computation measures are as follows:

- **Success Rate** – The proportion of successful planning runs:

$$SUCCES\_RATE = \frac{\sum_{j=0}^{N} S_{\mathrm{j}}}{N} \qquad (10)$$

Where $S_{\mathrm{j}} = 1$ when the jth run terminated successfully (i.e. with a valid path) and $S_{\mathrm{j}} = 0$ otherwise, N is the total number of trials.

- **Planning Time (PT)**[4] –The average time recorded from beginning of the planning computation of a run until a solution is found:

$$PT = \frac{\sum_{j=0}^{N} t_j}{N} \tag{11}$$

Where t_{j} is the planning time for run j, N is the total number of trials.

- **Nodes**[4,5] – The average number of nodes in the final RRT tree.
- Iterations[4,5] - The average number of iterations performed per run.
- **CDs**[4,5] – The average number of collision detection calls (the most resource-consuming function) per run.

Path quality measures are as follows:

- **Path's Index of Curvature (PIC)**[4,6] – The Index of Curvature is used to estimate path straightness. It is a normalized measure of the line-of-sight of the path by its length, thus ranging in $(0,1]$. PIC is an index of path curvature in the arm's configuration space using Euclidean distance measure. A highly convoluted path in the configuration space provides a low value (ineffective in terms of total path length and/or syncing the joints movements) and the line-of-sight itself provides value of 1. PIC can be formulated as:

$$PIC = \frac{\sum_{j=1}^{J}\left(x_{1,j} - x_{V,j}\right)^2}{\sum_{i=2}^{V}\sqrt{\sum_{j=0}^{J}\left(x_{i-1,j} - x_{i,j}\right)^2}} \tag{12}$$

Where: $x_{i,j}$ (deg.) is the ith vertex's position in the jth dimension of the configuration space, V is the total number of vertices in the path, and J is the total number of joints (dimensions of the configurations space).

- **Joint-Angles Index of Curvature ($JAIC$)**[4,6] – $JAIC$ is an index of path curvature in the arm's configuration space using city-block distance measure. It represents the sum the total length travelled (in degrees) in all of the robot's joints. $JAIC$ can be formulated as:

$$JAIC = \frac{\sum_{j=1}^{J}\left|x_{1,j} - x_{V,j}\right|}{\sum_{i=2}^{V}\sum_{j=0}^{J}\left|x_{i-1,j} - x_{i,j}\right|} \tag{13}$$

Where: $x_{i,j}$ (deg.) is the ith vertex's position in the jth dimension of the configuration space, V is the total number of vertices in the path, and J is the total number of joints (dimensions of the configurations space).

[4] Calculation is based on successful trials only
[5] Based on (Rodriguez, et al. 2006)
[6] Based on (Archambault, et al. 1999)

Statistical Analysis

According to measures from the last section, the analyses goal was to compare the performance of the two RRT variations and the two smoothing methods. We analyzed usability, paths quality and computation time.

Differences in success rates of the RRT versions were analyzed using a t-test for proportions with success rate as the dependent factor and the planning algorithm (e.g. goal-biased RRT and bi-directional RRT) as independent factor, to determine an algorithm's level of usability. An algorithm with a low success rate is not applicable as it is suspected to lack the ability to deal with some of the different problem-environments.

Path quality measures (PIC and $JAIC$) were analyzed using two analyses of variance (ANOVA) models with the quality measure as the dependent factor, map's levels of density (1-3), planning algorithm (e.g. goal-biased RRT and bi-directional RRT) and path smoothing method (e.g. classical smoothing and triple smoothing) as independent factors. Unsuccessful runs were dropped out, as path quality can only be computed for successful runs.

Path planning computation time was analyzed using generalized linear regression (GLM) model with gamma link function due to dependent factor's (planning time) non-negative values and right-tail. Independent factors were map's levels of density (1-3) and planning algorithm (e.g. goal-biased RRT and bi-directional RRT). Unsuccessful runs were ignored in this test.

Path smoothing marginal time was analyzed conducting 2 analyses of covariance (ANCOVA) with planning time as the dependent factor, map's levels of density (1-3) and path smoothing method (e.g. classical smoothing and triple smoothing) as independent factors, collision detection (CD) calls count (RRT only) as a covariant. CD count is a factor of great influence on path planning time of the RRT algorithms. The experiment's results dataset was divided into 2 dataset, one per each RRT algorithm version, and the analyses were conducted on each one of them separately. This split was done so the model could evaluate the covariate's value differently for each RRT algorithm version, based on the difference of using CD calls between the two RRT algorithm versions. Unsuccessful runs were ignored in this test.

Exploratory Analysis of Results

First, we visualized and explored the experiment's results. By analyzing the behavior of the different measures under all the different conditions, we have discovered that map 6 is dissimilar to other maps when it comes to comparing the behavior of the two RRT versions.

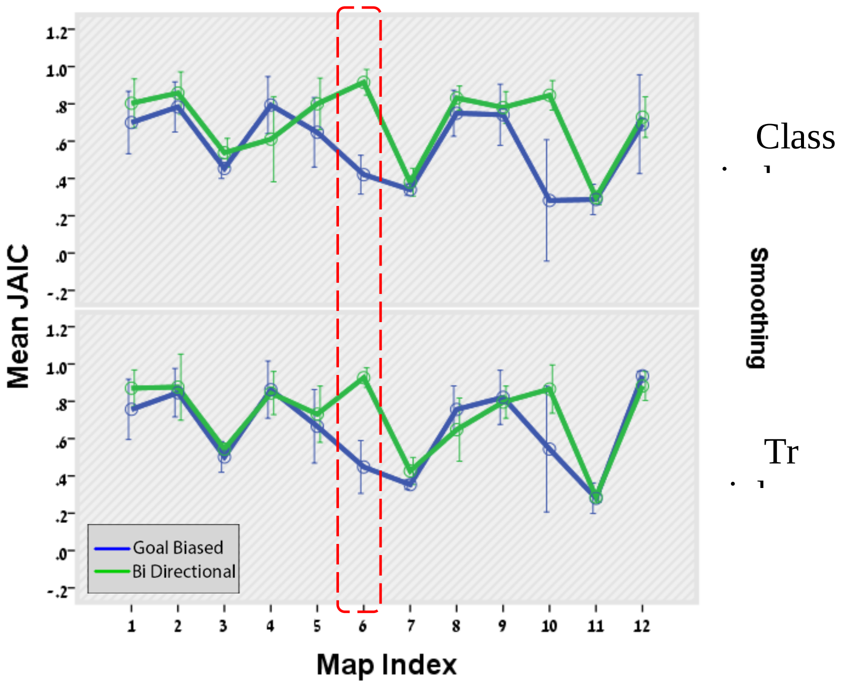

Figure 38 - JAICe results grouped by Smoothing method, RRT Algorithm and Map Index

As it can be seen in Figure 38, while exploring JAIC measure () we found that both goal-biased and bi-directional versions behave similarly in all the levels of obstacle density in the maps except in Map 6. In map 6, the bi-directional RRT produced paths of higher quality (shorter paths in the configuration space).

Following that, we ran another simulation of the two RRT algorithms, with 20 repetitions with map 6 (twice as much as we ran in the primary simulation). Figure 39 shows a visualization of the results in the configuration space where an approximation of infeasible (collision) configurations are visualized in grey **near the goal configuration only**. The red paths are the ones that resulted by running the goal-biased RRT while the blue paths are the ones that resulted by running the bi-directional RRT.

Figure 40 and Figure 41 show a visualization of the results of maps 10 and 12 from the original simulation. While the majority of the results of both algorithms behave more or less the same in the control group (i.e. map 10 and map 12), the results of the two RRT versions in the 6th are generally clustered into two groups by the algorithm version.

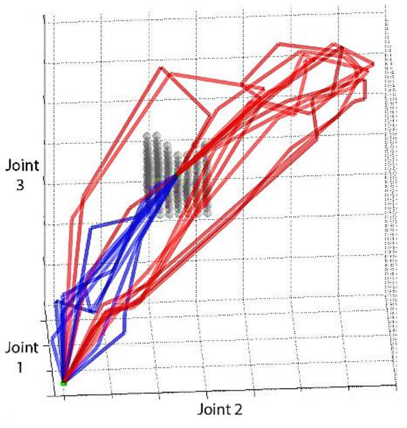

Figure 39 - Visualization of results in the configuration space (map 6)

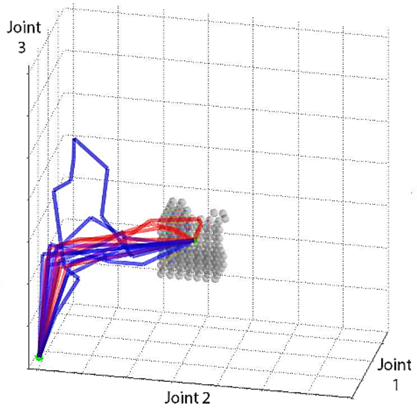

Figure 40 - Visualization of results in the configuration space (map 10)

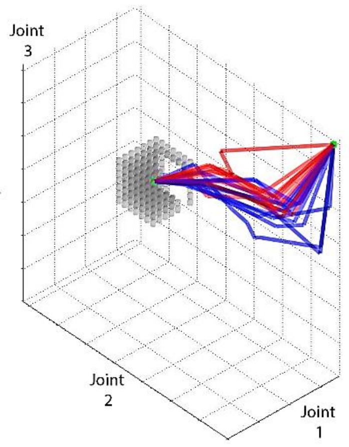

Figure 41 - Visualization of results in the configuration space (map 12)

A feasible explanation to this behavior is discussed in (LaVall 2006) where a "bug trap" obstacle environment is shown. Abstract version (2D) of such environments is presented in Figure 42. Map #6 is suspected to contain a narrow passage in the configuration space that causes this problem to be much easier to solve by a bi-directional tree exploration heuristic than by the goal-biased exploration. This particular environment constitutes an exceptional case which is out of the scope of this research. It therefore was left out of it and was removed from our analysis dataset.

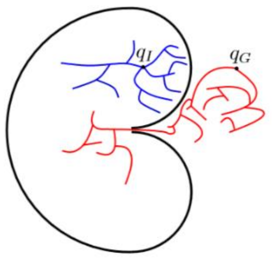

Figure 42 - A "Bug trap" problem solved by a bi-directional RRT algorithm.

Statistical Analyses Results

Success rate analysis shows a significant difference among the two RRT versions (P_{value}<0.01). The bi-directional RRT version produced a higher rate of successful results (100%) than the goal-biased RRT version (93%). Figure 43 and Table 1 present the success rate of the two RRT versions. **Appendix II** shows the results of the t-test for equality of the success rates. We found those results of great significance, as we desire in an algorithm that is able to perfectly cope with all environments and result in a feasible path, assuming such exists.

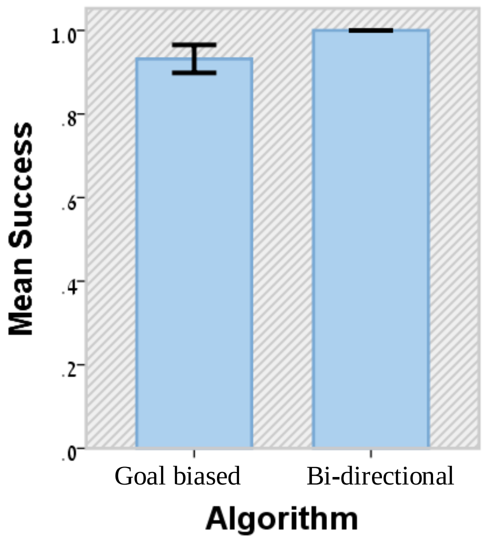

Figure 43 - Average Success rate by algorithm (bar chart). Error bar of 95% confidence level are layered on top of the bars

Table 4 - Success Rate by Algorithm

Group Statistics				
	Algorithm	N	Mean	Std. Deviation
Success Rate [%]	Goal-Biased	220	.93	.253
	Bi-Directional	220	1.00	.000

Path quality was measured using two measures: Path's Index of Curvature (PIC) and Joint-Angles Index of Curvature ($JAIC$), both produced similar results. RRT version and smoothing method did not significantly affect neither one of the path quality measures.

Table 5 - PIC by RRT Version, Smoothing Method

PIC – Descriptive Statistics			
Combination	Level	Mean	Std. Deviation
Goal Biased RRT + Classical Smoothing	1	.610	.145
	2	.511	.180
	3	.631	.214
	Total	.591	.171
Goal Biased RRT + Triple Smoothing	1	.660	.155
	2	.526	.184
	3	.650	.235
	Total	.620	.185
Bi-Directional RRT + Classical Smoothing	1	.636	.147
	2	.602	.227
	3	.591	.201
	Total	.61	.172
Bi-Directional RRT + Triple Smoothing	1	.704	.149
	2	.532	.143
	3	.614	.225
	Total	.624	.176

Table 6 - JAIC by RRT Version, Smoothing Method

JAIC – Descriptive Statistics			
Combination	Level	Mean	Std. Deviation
Goal Biased RRT + Classical Smoothing	1	.682	.158
	2	.578	.213
	3	.716	.277
	Total	.666	.207
Goal Biased RRT + Triple Smoothing	1	.741	.166
	2	.591	.211
	3	.744	.292
	Total	.702	.219
Bi-Directional RRT + Classical Smoothing	1	.702	.152
	2	.670	.252
	3	.663	.248
	Total	.682	.158
Bi-Directional RRT + Triple Smoothing	1	.579	.213
	2	.716	.277
	3	.666	.207
	Total	.742	.166

Table 5 and Table 6 show the performance of the different RRT versions and smoothing methods combinations in terms of PIC and $JAIC$, respectivly. **Appendix I** shows the results of the ANOVA for PIC and $JAIC$.

In the **Path planning computation time** analysis, the bi-directional RRT was found significantly less time consuming than the goal biased RRT, among all levels of obstacles density ($P_{value} < 0.001$). There was also a significant interaction between Algorithm*Level ($P_{value} < 0.001$), this means that the algorithm factor affects the planning time measure differently on different levels of obstacles density. In the lowest level of obstacles density, the mean planning time the goal-biased RRT is about 4.4 times longer/shorter than the mean planning time of the bi-directional RRT. This rate increases with the level of maps complexity, and reaches over 20 in the highest level tested.

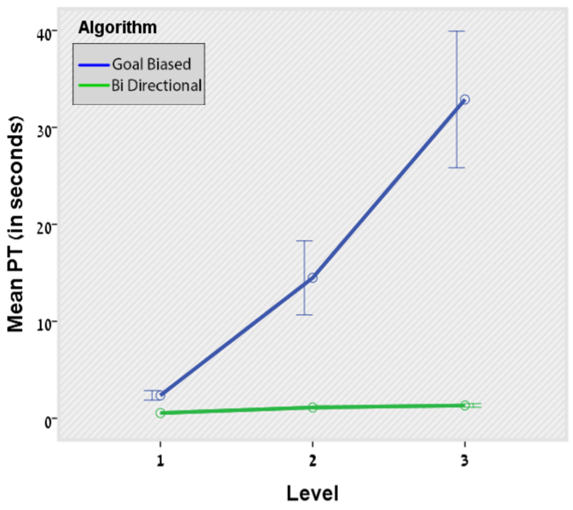

Figure 44 - Average planning time by Algorithm, Level of Complexity

Figure 44 and Table 7 present the effects of the different RRT algorithm versions on planning times. Error bar of 95% confidence level are layered on top of the bars. **Appendix I** presents the results of the GLM for planning time by RRT version.

Table 7 - Average Planning Time by RRT version

Planning Time [sec.]				
Level of complexity	**Algorithm**	**Mean**	**N**	**Std. Deviation**
1 (Low)	**Goal-Biased**	2.359	80	2.221
	Bi-Directional	0.535	80	0.384
2 (Medium)	**Goal-Biased**	14.482	60	14.779
	Bi-Directional	1.116	60	0.644
3 (High)	**Goal-Biased**	32.879	65	28.423
	Bi-Directional	1.317	80	0.830

Analysis of **path smoothing computation time for the goal-biased RRT version** showed no significant difference for path smoothing marginal planning times (PT) between the two path smoothing methods.

Figure 45 and Table 8 present the effects of the different smoothing methods on planning times. Error bar of 95% confidence level are layered on top of the lines. **Appendix I** presents the results of the ANCOVA for planning times by path smoothing methods.

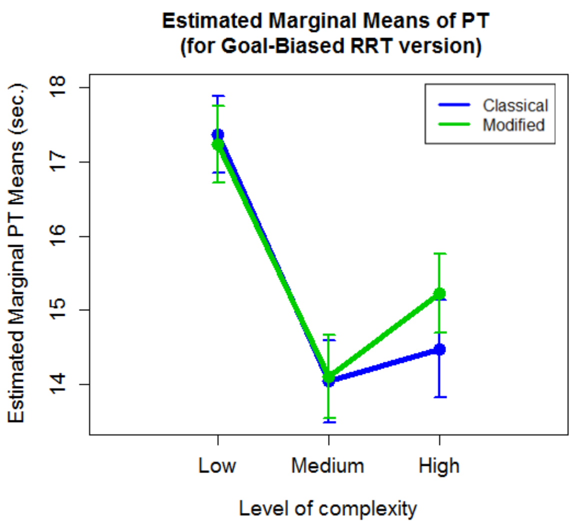

Figure 45 - Estimated Marginal Planning Time of path smoothing for goal-biased RRT solutions, by path smoothing method

Table 8 - Estimated marginal planning time of path smoothing for goal-biased RRT solutions

(Estimated Marginal) Planning Time [sec.]					
Smoothing	**Level**	**Mean**	**Std. Error**	**95% Confidence Interval**	
				Lower Bound	**Upper Bound**
Classical	**1**	17.366	0.521	16.338	18.394
	2	14.032	0.560	12.927	15.136
	3	14.475	0.656	13.181	15.769
Triple	**1**	17.231	0.519	16.206	18.255
	2	14.098	0.560	12.993	15.202
	3	15.224	0.534	14.170	16.277

Analysis of **path smoothing computation time for the bi-directional RRT version** showed a significant difference for Path Smoothing Marginal PT between the two path smoothing methods in all levels ($P_{value} < 0.001$), based on the *Bi-Directional* RRT version results dataset. The interaction $Smoothing * Level$ wasn't significant, therefore we conclude that the difference exists in all levels equally. The effects of the different smoothing methods on planning times are shown in Figure 45 and Table 9. **Appendix I** presents the results of the ANCOVA for planning times by path smoothing methods.

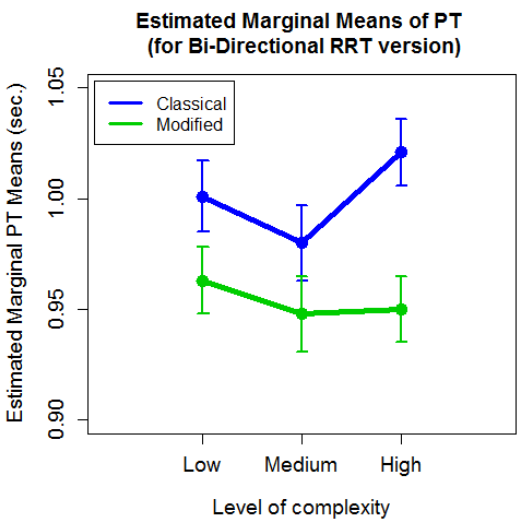

Figure 46 - Estimated Marginal Planning Time of path smoothing for bi-directional RRT solutions

Table 9 - Estimated marginal planning time of path smoothing for bi-directional RRT solutions

(Estimated Marginal) Planning Time [sec.]					
Smoothing	**Level**	**Mean**	**Std. Error**	**95% Confidence Interval**	
				Lower Bound	**Upper Bound**
Classical	**1**	1.001	0.016	0.970	1.031
	2	0.980	0.017	0.946	1.013
	3	1.021	0.015	0.991	1.051
Triple	**1**	0.963	0.015	0.933	0.993
	2	0.948	0.017	0.914	0.982
	3	0.950	0.015	0.920	0.981

Discussion and Conclusions

The purpose of the simulation was to evaluate the alternatives for RRT path planning versions and path smoothing methods. Results show a clear and meaningful distinction between the two RRT versions in both success rate and planning time measures. The bi-directional RRT has a 100% success rate in all levels of obstacles density, as well as with **a reduction in average planning time** ranging from 77% (in the lower level of obstacles density) to 96% (in the higher level of obstacles density). However, path quality did not significantly differ between the two methods.

Literature suggests that the bi-directional version is more robust and capable of handling "bug traps" and similar problems that the single-tree goal-biased is having hard times handling with. We suspected map 6 was such a problem and therefore removed it from the statistical analysis, not to bias result due to a special case. Still, this case brings an additional support for the results of the statistical analysis we conducted, which show that the bi-directional RRT version achieves a superior performance over its opponent.

In the analysis of the two path smoothing methods, results show a high significance for the difference between the two smoothing methods (P_{value}<0.001). Although the triple smoothing method examines a wider set of solutions in each iteration, it requires lower planning time on average when applied to a solution generated by the bi-directional RRT algorithm (its average planning times are 3%-7% lower than the classical smoothing method). We assume that by examining a wider set of solutions in every iteration, the convergence towards the final result happens with less iterations, and is therefore faster. However, the same effect wasn't found in the solutions generated by the goal-biased version. In addition path quality did not differ between the methods.

Given that the bi-directional RRT was previously found to be superior to the goal-biased RRT, the triple smoothing method also turns out to be preferred over the classical path smoothing method, due to the reduction in average planning time when combined with the bi-directional RRT. Although planning time reduction caused by the Triple Smoothing method seems to be negligible when considering total planning time (average of 0.05 seconds), we expect it to increase when executing path smoothing on paths generated in a higher dimension C-Space due to a significant increase in the original number of nodes per solution.

In conclusion, the best alternative to the presented application of triple-linked robot arm is the combination of bi-directional RRT path generating algorithm and the triple smoothing method. The latter is also integrated into a planning algorithm for harvest of peppers (Bac, et al. 2014) in a joint work with researchers from Wageningen University and Research Centre, Holland.

Additional Results

The following tables present additional results of the statistical analysis in the preliminary experiment. Analyzed measures include: success rate (Table 10), path length measures (Table 11, Table 12), planning time (Table 13), and path-smoothing marginal planning time (Table 14,Table 15).

Table 10 - T-test results for Success Rate

Success Rate					
Levene's Test for Equality of Variances		t-test for Equality of Means		95% Confidence Interval of the Difference	
	F	t	Mean Difference (Std. Error)	Lowe r	Upper
Equal variances assumed	74.618 ***	-4.003	-0.068 (0.017)***	-.102	-.035
Equal variances not assumed		-4.003	-0.068 (0.017)***	-.102	-.035

*** p<0.001 ** p<0.01 * p<0.05

Table 11 - ANOVA results for PIC

PIC (Tests of Between-Subjects Effects)				
	Type III Sum of Squares	**Df**	**Mean Square**	**F**
Corrected Model	1.241	11	.113	2.427
Intercept	151.590	1	151.590	3262.772***
Level	0.830	2	.415	8.928***
Algorithm	0.048	1	.048	1.029
Smoothing	0.055	1	.055	1.178
Level * Algorithm	0.092	2	.046	.992
Level * Smoothing	0.132	2	.066	1.424
Algorithm * Smoothing	0.026	1	.026	.561
Level * Algorithm * Smoothing	0.046	2	.023	.496
Error	19.188	413	.046	
Total	177.576	425		
Corrected Total	20.429	424		

*** p<0.001 ** p<0.01 * p<0.05

Table 12 - ANOVA results for JAIC

JAIC (Tests of Between-Subjects Effects)				
	Type III Sum of Squares	**Df**	**Mean Square**	**F**
Corrected Model	1.487	11	0.135	2.209
Intercept	191.426	1	191.426	3127.769***
Level	0.962	2	0.481	7.860***
Algorithm	0.048	1	0.048	0.779
Smoothing	0.115	1	0.115	1.882
Level * Algorithm	0.082	2	0.041	0.671
Level * Smoothing	0.185	2	0.093	1.513
Algorithm * Smoothing	0.021	1	0.021	0.348
Level * Algorithm * Smoothing	0.046	2	0.023	0.377
Error	25.276	413	0.061	
Total	225.170	425		
Corrected Total	26.763	424		

*** p<0.001 ** p<0.01 * p<0.05

Table 13 - GLM for planning time

Total Planning Time			
Fixed Effects	**Coef. (Standard Error)**	**95% Confidence Interval for Planning-Time Ratio**	
		Lower Bound	**Upper Bound**
Intercept [sec.]	0.858 (0.095)***	0.672	1.045
Level = 2	1.815 (0.145)***	1.530	2.100
Level = 3	2.634 (0.142)***	2.356	2.913
Algorithm = Bi-Directional	-1.483 (0.134)***	-1.747	-1.219
[Level = 2] * [Algorithm = GR-RRT]	-1.734 (0.195)***	-2.118	-1.351
[Level = 3] * [Algorithm = GR-RRT]	-1.080 (0.205)***	-1.483	-0.677

*** p<0.001 ** p<0.01 * p<0.05

Table 14 - ANOVA results for Marginal Smoothing Planning Time (Goal-Biased RRT)

Marginal Smoothing Planning Time (Goal-Biased RRT)				
	Type III Sum of Squares	**Df**	**Mean Square**	**F**
Corrected Model	96627.686	6	16104.614	1712.192***
Intercept	1123.362	1	1123.362	119.432***
CDs (covariate)	58918.608	1	58918.608	6264.043***
Smoothing	2.527	1	2.527	0.269
Level	319.492	2	159.746	16.984***
Smoothing * Level	7.010	2	3.505	0.373
Error	1862.357	198	9.406	
Total	148280.092	205		
Corrected Total	98490.043	204		

*** p<0.001 ** p<0.01 * p<0.05

Table 15 - ANOVA results for Marginal Smoothing Planning Time (Bi-Directional RRT)

Marginal Smoothing Planning Time (Bi-Directional RRT)				
	Type III Sum of Squares	**Df**	**Mean Square**	**F**
Corrected Model	114.758	6	19.126	2185.143***
Intercept	0.001	1	0.001	0.061
CDs (covariate)	87.491	1	87.491	9995.596***
Smoothing	0.116	1	0.116	13.211***
Level	0.018	2	0.009	1.004
Smoothing * Level	0.016	2	0.008	0.922
Error	1.864	213	0.009	
Total	327.177	220		
Corrected Total	116.623	219		

*** p<0.001 ** p<0.01 * p<0.05

APPENDIX II - EXPERIMENT: SINGLE-GR SCHEME (SELECTIVE APPLE HARVESTING)

Goals

This experiment was designed to evaluate the performance of GR-RRT framework for planning toward simple objects which require a single GR scheme. A simulated environment that resembles a selective harvest of apples was created and used for the experiment. The performance of GR-RRT was compared to the IKBiRRT algorithm. As GR-RRT uses a prior knowledge of object graspability for planning, IKBiRRT variations are augmented only with the object's structural data, i.e. its shape and size. To facilitate planning toward a grasp of the object while exploiting its structural data alone, two variations of IKBiRRT were used, each incorporated a different automatically-generated WGR:

- $IKBiRRT|_{Bounding}$– Based on a WGR whose XYZ dimensions bounds were defined by the box that bounds the apple's surface (Figure 47).
- $IKBiRRT|_{Bounded}$– Based on a WGR whose XYZ boundaries were defined by the box bounded by the apple's surface (Figure 48).

In both cases the ranges of the orientation angles were $[0,2\pi]$.

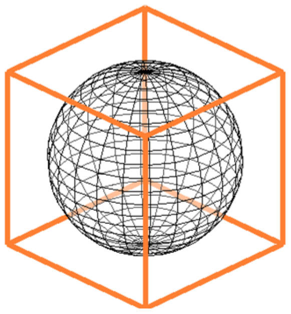
Figure 47 - Minimum volume bounding box for apple object

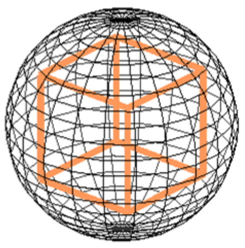
Figure 48 - Maximum volume bounded box inside apple object

Computation Infrastructure

We have used a PC machine equipped with an Intel i7-3770K 3.5 GHz processor (CPU) and 32GB installed memory (RAM), running on Windows 8.1 (64-bit). The development and execution of the experiment was done using MATLAB (Version 2011a, Mathworks, USA) and analysis was done using IBM SPSS Statistics (Version 19, IBM, USA).

Simulation Environment

A virtual environment of size 300x300x200 was created for the experiment. It was populated with four instances of artificial trees of size 88x108x180 each (Figure 49), at fixed positions. The manipulator was modelled as a 6-DOF robotic arm (Figure 51) and was positioned in a fixed initial position. Its Denavit-Hartenberg (DH) convention is detailed in Table 16. The apple target-object was modelled as a sphere with radius of size 4 (Figure 50), and was positioned in six different positions which defined 3 levels of reachability: simple (Figure 52), moderate (Figure 53) and difficult to reach (Figure 54). In each scenario, 100 paths were planned from the initial position towards the apple using each of the three algorithms, and the triple smoothing method (see Appendix I) was applied to each.

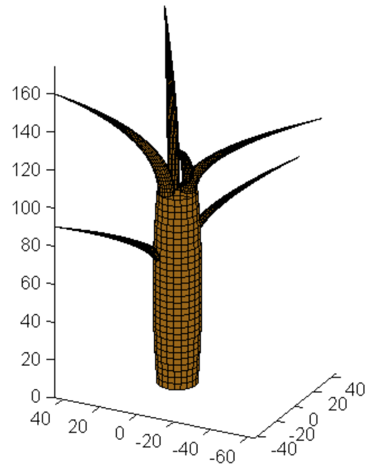

Figure 49 - A tree object used in simulation

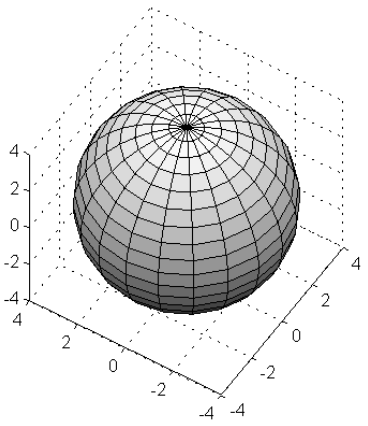

Figure 50 - An apple spherical model used in simulation

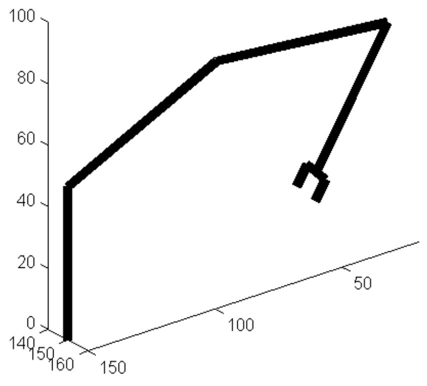

Figure 51 – The 6-DOF manipulator used in simulation.

Table 16 – Denavit-Hartenberg convention of the manipulator used in simulation

Link	a_i	**α**	**d_i**	θ_i
1	0	$-\pi/2$	50	θ_1
2	60	0	0	θ_2
3	60	$-\pi/2$	0	$\theta_3 - \pi/2$
4	0	$\pi/2$	0	θ_4
5	0	$-\pi/2$	0	θ_5
6	0	0	50	θ_6

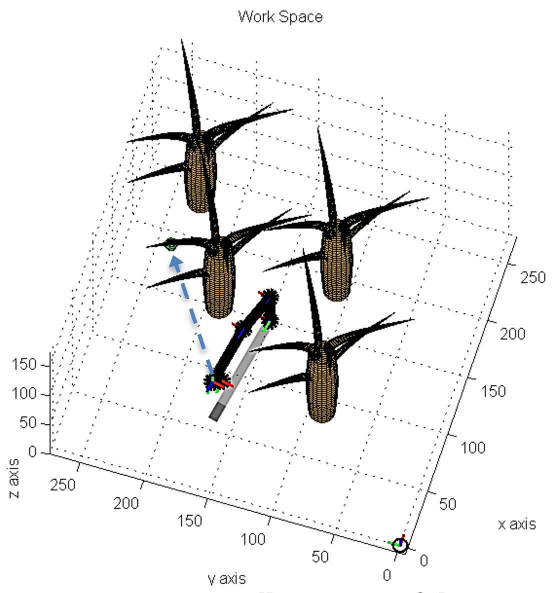

Figure 52 – Illustration of the environment where the apple is simply reachable (location-id #1)

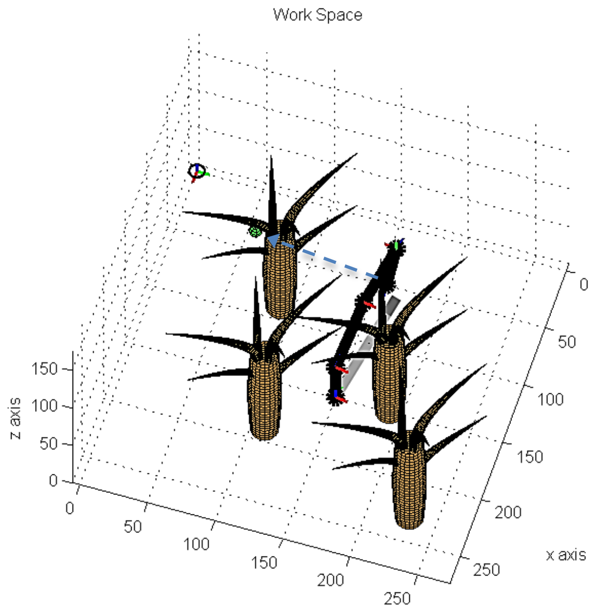

Figure 53 – Illustration of the environment where the apple is moderately reachable (location-id #4)

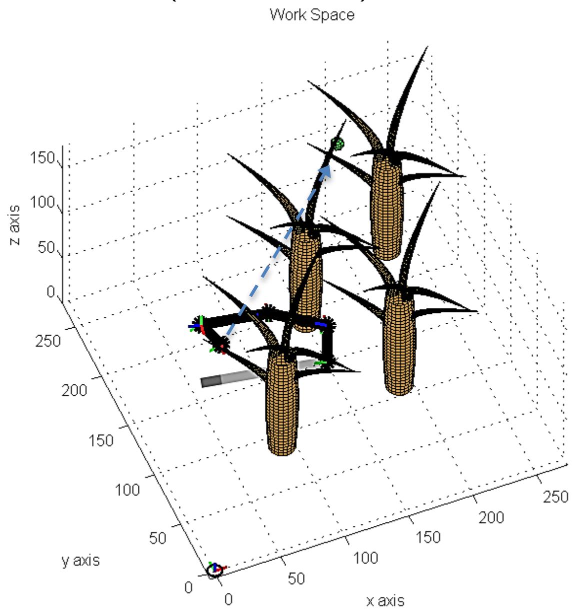

Figure 54 – Illustration of the environment where the apple is hardly reachable (location-id #6)

Analysis

Algorithm performance was compared in terms of **computation time** and **planning quality**. The latter was quantified based on 3 measures: **path success**, i.e. if a path was found before the algorithm terminated then path success was determined as one (zero otherwise), **path length** of the planned path in the configuration space, and **grasp success**, i.e., if the path ended with a grasp-pose for which grasp quality was above 0.7 grasp success was determined as one (zero otherwise).

Statistical Analysis

Path success analysis didn't require any statistical model due to identical results among all variations. **Computation time** and **path length** were analyzed using a generalized linear mixed model with a gamma link-function. A gamma link function was used as the distribution of both measures contained a long right-tail and non-negative values. The initial model included two predictors: reachability level (simple, moderate, difficult to reach), and algorithm (GR-RRT, $IKBiRRT|_{Bounding}$, and $IKBiRRT|_{Bounding}$), and Object-location-id as a random effect. **Grasp success** was analyzed using a logistic regression mixed model with two predictors: reachability level (simple, moderate, difficult to reach), and algorithm (GR-RRT, $IKBiRRT|_{Bounding}$, and $IKBiRRT|_{Bounding}$), and Object-location-id as a random effect.

Results

In the **a-priori learning** of GR-RRT, a graspability map was constructed for the apple object (Figure 55). Based on this map, TCP positions were extracted. Grasps were filtered with a threshold of 0.7 grasp-quality, so the set of successful grasps included a total of 8504 gripper poses (out of 33,240), which were bounded by a single axes-aligned minimum-volume box (Figure 56). GMM inference produced 5 Gaussian components (Figure 57).

All three algorithms had a **path-planning success** of 100% with the applied planning limitation of maximum 50,000 iterations per execution.

For **grasp success**, apple's location-id (random effect), reachability, and the interactions between the predictors, were found non-significant and thus they were removed from the model. The final model included only algorithm (Table 17). Using GR-RRT, grasp success odds are 90 times higher (7.957) than when using $IKBiRRT|_{Bounded}$ and 500 times higher than when using $IKBiRRT|_{Bounding}$.

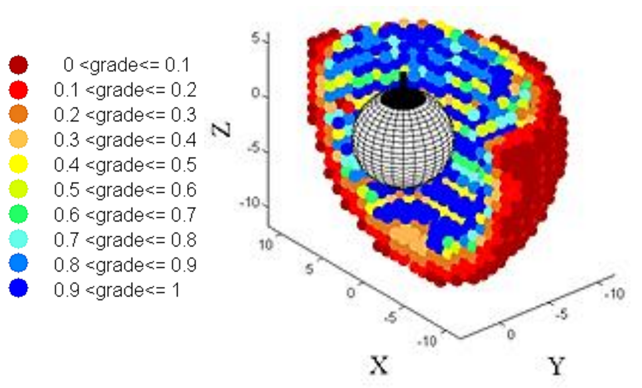

Figure 55 – Visualization of the graspability map produced for the apple object

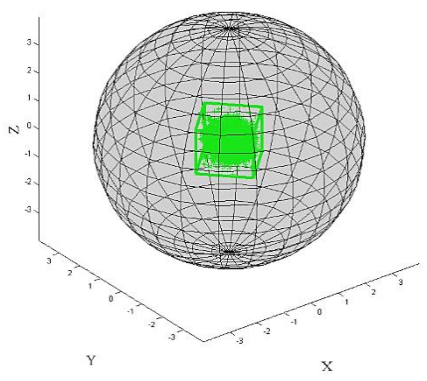

Figure 56 – Extracted TCP positions of successful grasps (green) for apple with GR boundaries

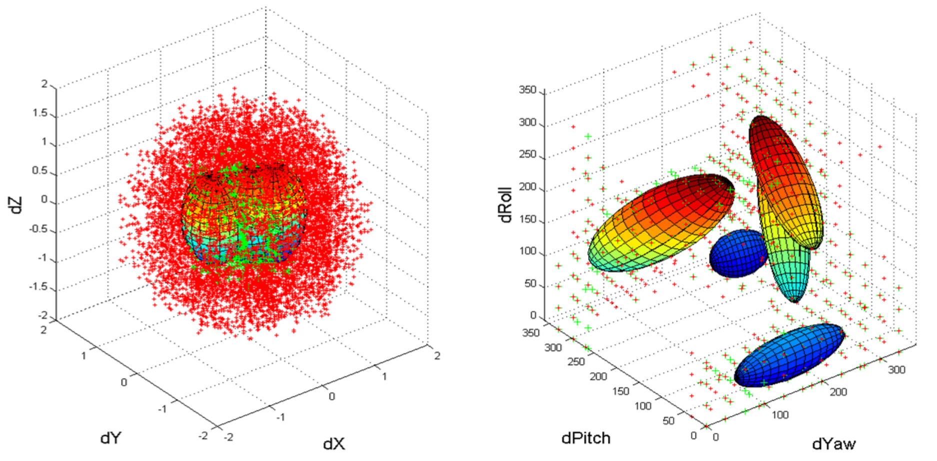

Figure 57 – Visualization of the 6D GMM of an apple object.
The figure is divided into TCP position coordinates (left) and TCP orientation angles (right). Successful grasps are marked in green and bad grasps are marked in red. Ellipsoids represent 95% confidence intervals for the 5 Gaussian components of the GMM.

Table 17 - Logistic regression results for Grasp Success

Grasp Success				
Fixed Effects	**Coef (Standard Error)**	**95% Confidence Interval for Odds ratio**		
		Lower Bound	**Odds Ratio**	**Upper Bound**
Intercept	2.074 (0.130)***	[6.170]	[7.957]	[10.256]
Algorithm= $\boldsymbol{IKBiRRT|_{Bounded}}$	-4.539 (0.200)***	0.7%	1.1%	1.6%
Algorithm= $\boldsymbol{IKBiRRT|_{Bounding}}$	-6.258 (0.360)***	0.1%	0.2%	0.4%

*** p<0.001 ** p<0.01 * p<0.05

For **computation time**, the apple's location-id (random effect), and the interactions were found non-significant and thus were removed from the model. The final model included the reachability level and algorithm. Results indicated that there is no significant difference between GR-RRT and $IKBiRRT|_{Bounded}$. Yet, IKBiRRT$|_{Bounding}$ produced 18% lower than GR-RRT (Table 18).

Table 18 - Generalized linear regression results for Planning Time

Planning Time				
Fixed Effects	**Coef. (Standard Error)**	**95% Confidence Interval for Planning-Time Ratio**		
		Lower Bound	**exp(Coef.)**	**Upper Bound**
Intercept [sec.]	3.264 (0.239)***	[16.36]	[26.154]	[41.84]
Level=1	-2.862 (0.335)***	2.95%	5.72%	11.02%
Level=2	-2.093 (0.335)***	6.38%	12.33%	23.78%
Algorithm= $\boldsymbol{IKBiRRT|_{Bounded}}$	-0.082 (0.058)	82.12%	92.13%	103.25%
Algorithm= $\boldsymbol{IKBiRRT|_{Bounding}}$	-0.202 (0.058)***	72.83%	81.71%	91.58%

*** $p<0.001$ ** $p<0.01$ * $p<0.05$

For **path length**, the apple's location-id (random effect) was found to be non-significant and was thus removed from the model. Algorithm was also found to be non-significant but as interactions were significant it was retained in the model. Results (Table 19) indicated that that the algorithms significantly differ only for difficult-to-reach targets. For this reachability level, on average IKBiRRT$|_{Bounded}$ produces a path that is 20% shorter and IKBiRRT$|_{Bounding}$ produces a path that is 16% shorter than GR-RRT.

Table 19 - Generalized linear regression results for Path Length

Path Length [Δdeg.]				
Fixed Effects	**Coef. (Standard Error)**	**95% Confidence Interval for Planning-Time Ratio**		
		Lower Bound	**exp(Coef.)**	**Upper Bound**
Intercept	4.943 (0.217)***	[91.47]	[140.19]	[214.65]
Level=2	0.737 (0.307)*	114.45%	208.96%	381.90%
Level=3	1.218 (0.307)***	184.96%	338.04%	617.16%
Algorithm= $\boldsymbol{IKBiRRT\vert_{Bounded}}$	0.040 (0.048)	97.74%	104.08%	114.23%
Algorithm= $\boldsymbol{IKBiRRT\vert_{Bounding}}$	0.006 (0.048)	91.67%	100.60%	110.52%
[Level=2] · [Algorithm= $\boldsymbol{IKBiRRT\vert_{Bounded}}$**]**	-0.039 (0.067)	84.28%	96.18%	109.86%
[Level=2] · [Algorithm= $\boldsymbol{IKBiRRT\vert_{Bounding}}$**]**	-0.014 (0.067)	86.42%	98.61%	112.52%
[Level=3] · [Algorithm= $\boldsymbol{IKBiRRT\vert_{Bounded}}$**]**	-0.223 (0.067)**	70.18%	80.01%	91.39%
[Level=3] · [Algorithm= $\boldsymbol{IKBiRRT\vert_{Bounding}}$**]**	-0.179 (0.067)**	73.27%	83.61%	95.50%

*** p<0.001 ** p<0.01 * p<0.05

APPENDIX III – GRASPABILITY MAPS

The following figure (Figure 58) illustrates graspability maps for apple (symmetrical sphere), mug and pan objects, which store data regarding grasps generated in simulation, including their matching grasp quality measure.

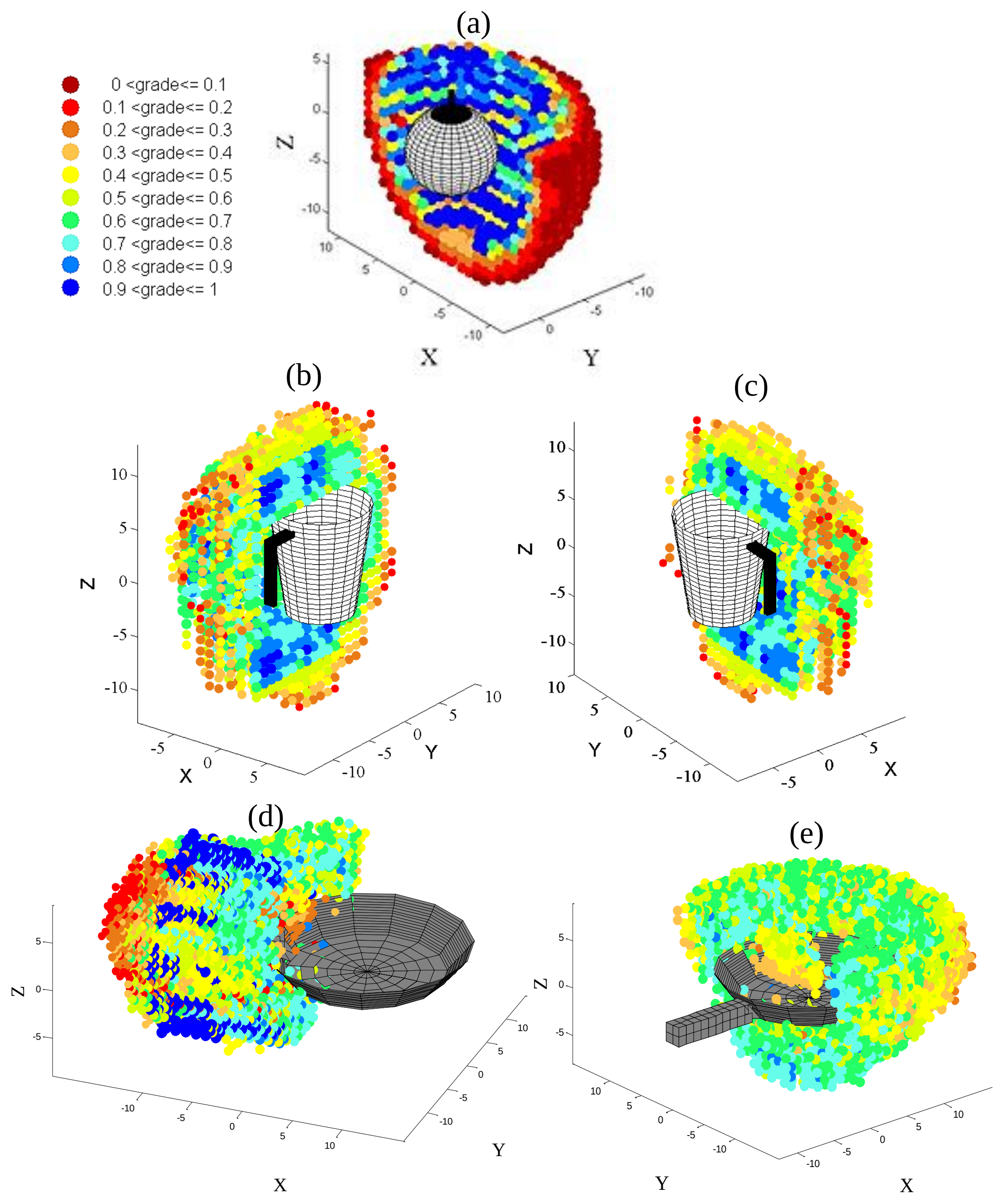

Figure 58 – Graspability maps visualizations of all objects used in this work
(a) apple, (b-c) mug, (d-e) pan

תקציר

מחקר זה מציג את אלגוריתם Grasp-Regions Rapid-exploring Random Trees (GR-RRT), שהינו אלגוריתם חדשני לתכנון אוטומטי של משימות הגעה-עד-אחיזה (reach-to-grasp) עבור יישומים רובוטיים. אלגוריתם זה מורכב משני שלבים: שלב הכנה בו מחשבים את אזורי האחיזה ושלב תכנון מסלול. אזורי האחיזות מכילים מיקומי תפסנית (כיוון ומיקום) הצפויים להוביל לאחיזה מוצלחת. שלב זה מתבסס אישכול בעזרת אלגוריתם תיבות חוסמות בעל נפח מינימאלי (minimum volume bounding boxes) ולמידה סטטיסטית של אזורים בעלי אחיזות איכותיות בעזרת מודל תערובת התפלגויות גאוסיאנית רב ממדיות multivariate Gaussian mixture model (multivariate GMM) ועל השימוש במפות אחיזה. תכנון המסלול נעשה בעזרת גרסה מותאמת של אלגוריתם ה-RRT אשר משתמשת במודל הסטטיסטי מהשלב הקודם בכדי לייצר מיקומים מהם צפויה הצלחה באחיזה ובד בבד מטה את המסלול המתוכנן אל מיקומים אלו, תוך שמירה על תכונת השלמות ההסתברותית (probability completeness) של אלגוריתם ה-RRT. בנוסף, פותחה שיטת ההחלקה המשולשת (triple smoothing) בכדי להשלים את תכנון המסלול ע״י RRT.

הביצועים של האלגוריתם זה נבחנו והושוו לביצועי שיטת IKBiRRT, המבוססת על תכנון בעזרת RRT אל עבר אזורי אחיזה שהוגדרו באופן ידני ובהם ההסתברות לאחיזות המוצלחות מפולגת באופן אחיד. הבחינה התבצעה בעזרת סימולציה של שני תרחישים (קטיף תפוחים וסביבה ביתית). בתרחיש קטיף התפוחים לתפוח יש אזור אחיזה בודד ובתרחיש הסביבה הביתית לחפצים (כוס ומחבת) יש מבנה מורכב יותר והם דורשים מספר אזורי אחיזה. הערכת האלגוריתמים מבוססת על אנליזה מקיפה של התוצאות, ובפרט הערכת יעילות ואיכות. האלגוריתם יושם בעזרת חומרה (זרוע רובוטית).

שני האלגוריתמים שנבחנו מצאו בהצלחה מסלולים בכל המקרים שנבחנו. אלגוריתם GR-RRT תכנון מסלולים אל עבר מיקומים תפסנית המבטיחים אחיזה מוצלחת ב-82% מהמקרים. אחוז זה היה טוב משמעותית ביחס לאחיזות המוצלחות שנמצאו על ידי אלגוריתם IKBiRRT עם רק מעט מאוד עליה בזמן החישוב ובאורך המסלולים שחושבו. ביישום בחמרה רוב המסלולים בוצעו בהצלחה. במקרה אחד הייתה התנגשות בגלל חוטים שלא מודלו. בהתאם יתרונות שיטת GR-RRT עבור תכנון מסלול לצורך אחיזה הודגמו. עבור יישום בחמרה יש להוסיף הימנעות ממכשולים בזמן הריצה לתכנון המסלול.

מילות מפתח – רובוטיקה, אחיזה, תכנון מסלול, תכנון אחיזה, עצי חקירה אקראיים, תיבות חוסמות מינימאליות, מודל תערובת גאוסיאנית, החלקת מסלול

אוניברסיטת בן-גוריון בנגב
הפקולטה למדעי ההנדסה
המחלקה להנדסת תעשייה וניהול

תכנון תנועות הגעה-עד-אחיזה עבור זרוע רובוטית בעזרת RRT

חיבור זה מהווה חלק מהדרישות לקבלת תואר מגיסטר בהנדסה

מאת:	רועי רשף
מנחה/ים:	ד"ר סיגל ברמן

חתימת המחבר........................	תאריך......................
אישור המנחה/ים.........................	תאריך....................
....	תאריך....................
אישור יו"ר ועדת תואר שני מחלקתית........................	תאריך..........................

אוניברסיטת בן-גוריון בנגב
הפקולטה למדעי ההנדסה
המחלקה להנדסת תעשייה וניהול

תכנון תנועות הגעה-עד-אחיזה עבור זרוע רובוטית בעזרת RRT

חיבור זה מהווה חלק מהדרישות לקבלת תואר מגיסטר בהנדסה

מאת: רועי רשף

אלול תשע"ד   ינואר 2015