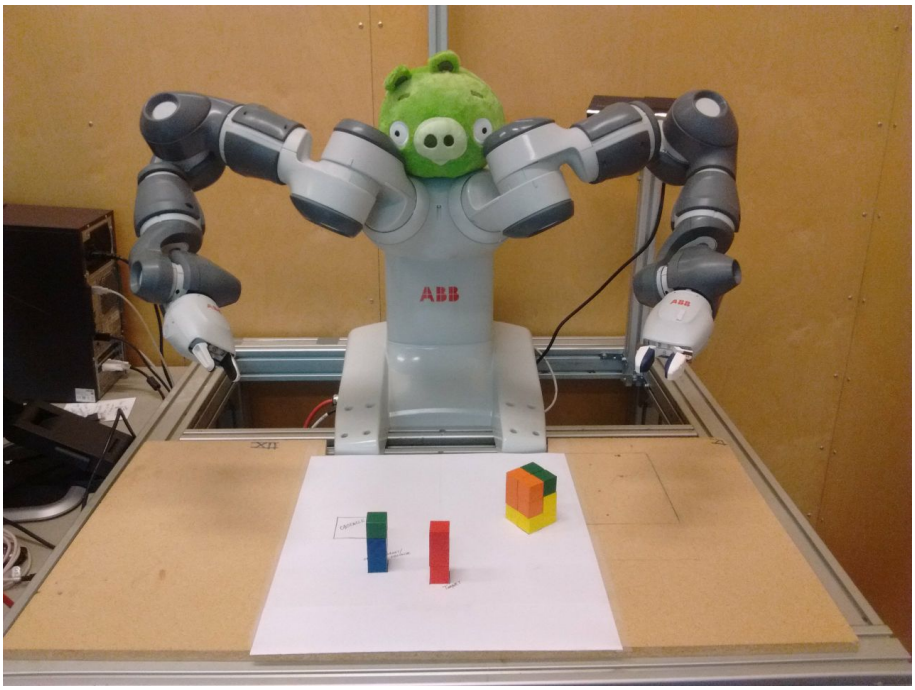
ÖREBRO UNIVERSITY

International Master's Thesis

Automated picking: Extracting constraints for HiQP from perception



Chittaranjan Srinivas Swaminathan
Computer Science

Örebro 2017

Chittaranjan Srinivas Swaminathan

Automated picking: Extracting constraints for HiQP from perception

Supervisors: Todor Stoyanov
Robert Krug

Examiner: Martin Magnusson and Jens Lundell

Contents

Chapter 1
Introduction

1.1 Introduction

In this Master's thesis project we focus on the problem of picking objects autonomously using a collaborative robot. In particular, this project integrates existing software for automated picking and explores the problem of obstacle avoidance using Signed Distance Field representation.

Systems employing sampling based planners [1] suffer from issues of unnatural and sub-optimal trajectories, high planning time in cluttered spaces, etc. In addition, these systems are usually based on traditional sense-plan-act architectures, where it is hard to incorporate real-time sensory feedback. In a control based approach for simultaneous motion generation and execution, the controller *is* the planner and runs with real-time sensory feedback. A manipulation framework based on the task priority framework [2] is quite robust for executing manipulator tasks and for specifying priorities among these tasks. It has been demonstrated to provide good results for picking tasks [3]. The **Hi**erarchical **Q**uadratic **P**rogramming framework is an efficient implementation of the task priority framework by Johansson [4]. This framework allows us to formulate a manipulator goal as a set of inequality constraints.

A particular advantage of a constraint-based approach, such as HiQP, is that a variety of manipulation problems can be modelled as set of constraints. For instance, the task of grasping an object could be modelled as a set of constraints on the pose of the end-effector or gripper. The system by Stoyanov et al. [5], which we will henceforth refer to simply as the *Grasp Planner*, uses sensory information to extract collision free regions around a target object. This region is called the Grasp Envelope. The grasp envelope is then reduced to primitive geometries and task formulations (or constraints) based on those geometries. When the constraints are fed into HiQP, the gripper reaches a pose where it is ready to grasp the target object.

The Grasp Planner only detects collision of the gripper with the environment in a small region around the target object. In other words, although the target pose might be collision-free, the trajectory that HiQP might generate is not guaranteed to be collision-free. This motivates the need for obstacle avoidance in our picking system.

The Signed Distance Field (SDF) representation is conceptually similar to the potential fields approach and has been shown to be well suited for both mapping and planning [6]. The work by Dietrich et al. aims to integrate the

potential fields approach [7] into a task hierarchy [8]. We will adopt a method that is similar to the above mentioned schemes on an abstract level. The main contribution of this project is the integration of a collision avoidance system based on the SDF representation into the HiQP framework. We will also conduct experiments to evaluate the collision avoidance behaviour and assess the joint motions resulting from this framework.

A reactive obstacle avoidance behaviour such as the one we present, can be advantageous for a picking robot. For instance, in a traditional sense-plan-act architecture when an obstacle appears suddenly, replanning might be necessary to successfully avoid the obstacle. This means that a collision might occur when this replanning happens. However, a reactive system such as ours can dynamically avoid the obstacle since it does not involve planning. Consequently, such a system can be used in colaborative picking tasks with humans safely without injury to the colaborating human. Another example could be a social, assistive robot used to assist the elderly or disabled with simple day-to-day tasks, where a manipulator with reactive obstacle avoidance might be an indispensable asset.

Implementation view

In order to provide a perspective for the rest of this text, this section aims to give an implementation view of the entire system. Figure 1.1 shows the different software components in our system and data they exchange. The figure also marks out packages actively developed in this project. The project uses chiefly four software stacks:

1. The HiQP stack [1]
2. The HiQP demos package [2]
3. The grasping stack [3]
4. The Yumi demos stack [4]

We now outline how the complete system works in practice. Firstly, the demonstration node is initialized by the user. The demonstration node then requests the Grasp Planner for constraints. The Grasp Planner then uses the SDF Map computed by the SDF Tracker along with the position given it by the demonstration node to compute constraints for the end effector. These constraints are a combination of primitive geometries and tasks associated with these geometries. For instance a sphere is a primitive geometry. A task associated with that sphere could be: *Stay on the surface of the sphere*. When the end effector satisfies these constraints, it ends up in a pose where it is ready to grasp the target object.

The demonstration node receives these constraints and forwards them to the HiQP controller. The HiQP controller also loads the Obstacle Avoidance Task, which is loaded at runtime as a ROS Plugin. The Obstacle Avoidance plugin also utilizes the SDF Map (which it receives from the SDF Tracker) to avoid any collisions with the environment. The demonstration node continuously tracks the status of these tasks and ends once all the tasks have completed (low enough error).

[1] https://github.com/OrebroUniversity/hiqp
[2] https://github.com/OrebroUniversity/hiqp_demos
[3] https://github.com/OrebroUniversity/grasping_oru
[4] https://github.com/OrebroUniversity/yumi_demos

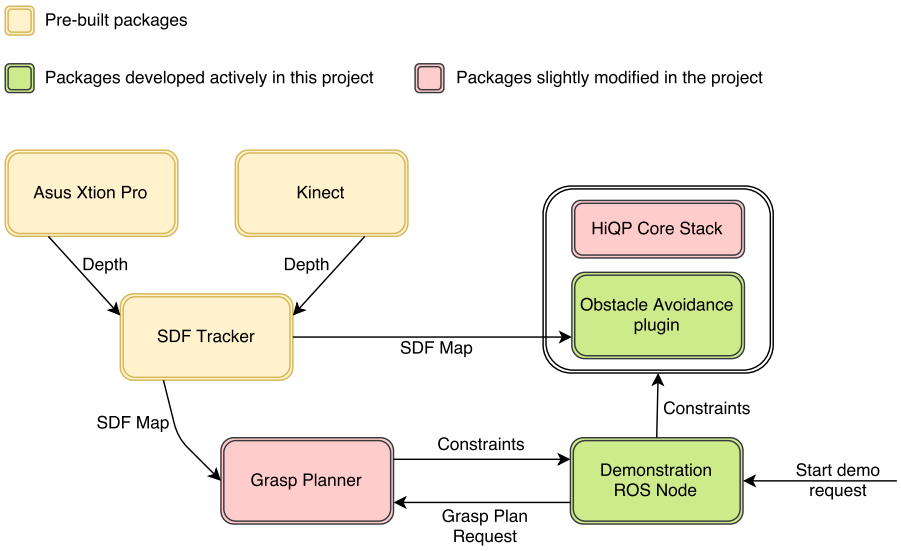


Figure 1.1: Implementation view of the completed system

1.2 Outline

The Chapters in this thesis are organized as follows:

- Section 2.1 describes the HiQP framework.
- Section 2.2 describes the Grasp Planner and how it extracts the Grasp Envelopes from sensory information.
- Chapter 3 deals with the problem of obstacle avoidance using SDF representation.
- Chapter 4 deals with the various experiments performed with our system and the results.
- Chapter 5 offers concluding remarks, summarizes the contributions and suggests subject areas for potential future work.

Chapter 2
Object picking using HiQP

2.1 The HiQP framework

A task is a kinematic goal (move the end-effector to a particular plane) or dynamic goal (apply a certain force on a certain surface) in the context of manipulator control. It is of great advantage if a robot can execute multiple tasks at the same time. This is especially true in the case of redundant manipulators. For instance when a coffee cup is grasped by the manipulator, one task could be to move the end-effector from one point in space to another. At the same time the orientation of the end-effector should be such that the cup is upright.

Furthermore, the task of keeping the cup upright (and hence, not spilling the coffee) might be more important than moving the end-effector. When facing multiple tasks at the same time, it is beneficent to be able to group equally important tasks while prioritizing more critical tasks. The *Hierarchical Quadratic Programming* or *HiQP* for short allows us to do exactly that. In HiQP, every task is formulated as a convex optimization problem.

The HiQP framework is a C++ based implementation of the task priority framework first introduced by Siciliano and Slotine [9] and further generalized by Kanoun et al. [2]. In our work, we wish to exploit the useful properties of the HiQP framework to implement picking tasks and an obstacle avoidance task with the highest priority.

In this section, we look at the mathematical basis of the HiQP framework. The explanations will only expound topics most relevant to our work. The reader is invited to read the work by Johansson [4] for a more thorough study of the framework.

2.1.1 Goals as constraints

A convex optimization problem is one of the form[1],

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && \mathbf{A}_i^T \mathbf{x} \geqslant \mathbf{b}_i,\ i = 1,...,m \end{aligned}$$

[1] Note that there is no loss of generality here, an equality constraint or less-than-or-equal-to constraint can both be written using greater-than-or-equal-to constraints.

if the function $f(\mathbf{x})$ and the inequality constraints are convex. In other words, we wish to values of $\mathbf{x}$ for which $f(\mathbf{x})$ (called the objective) is minimum, subject to the inequality (constraints).

A kinematic or dynamic goal can be modelled an convex optimization problem if the objective function and the constraints are convex. Suppose we have a manipulator whose joint velocities can be controlled and we want to make the end effector reach a particular plane in space. Suppose we have a function $e(\mathbf{q})$ which represents an error. This function is called the Task Function. For instance, it could represent the Euclidean distance between the aforementioned plane and the end effector. The plane is described by the equation

$$\mathbf{n}^T\mathbf{x} - a = 0 \tag{2.1}$$

where vector $\mathbf{n}$ is the normal vector and scalar a is the offset. In this particular case the error described above can written as

$$e(\mathbf{q}) = \mathbf{n}^T\mathbf{p}(\mathbf{q}) - a \tag{2.2}$$

where $\mathbf{p}(\mathbf{q})$ is the position of the end effector, which is a function of the joint angles. The rate of change of the task function is

$$\dot{e}(\mathbf{q}) = \frac{\partial e(\mathbf{q})}{\partial \mathbf{q}}\dot{\mathbf{q}} = \mathbf{n}^T\frac{\partial p(\mathbf{q})}{\partial \mathbf{q}}\dot{\mathbf{q}} \tag{2.3}$$

The quantity $\mathbf{n}^T\frac{\partial p(\mathbf{q})}{\partial \mathbf{q}}$ is called the task jacobian, since it describes the rate of change of the task function with respect to the joint angles and is denoted by $\mathbf{J}_t(\mathbf{q})$ or simply $\mathbf{J}_t$.

For a velocity controlled manipulator the generalized convex optimization equations can be written as

$$\begin{aligned} \underset{\mathbf{w},\dot{\mathbf{q}}}{\text{minimize}} \quad & \frac{1}{2}(\|\mathbf{w}\|^2 + \lambda_r\|\dot{\mathbf{q}}\|^2) \\ \text{subject to} \quad & \mathbf{J}_t\dot{\mathbf{q}} \succcurlyeq \dot{\mathbf{e}}_d + \mathbf{w} \\ \text{where} \quad & \mathbf{w} \text{ is a slack variable.} \\ & \lambda_r \text{ is a regularization parameter.} \end{aligned} \tag{2.4}$$

In words, this formulation says, "Keep the slack variables small, keep the joint motions small while still making sure the rate of change of measured error is a certain value". Here $\dot{\mathbf{e}}_d$ is the desired error dynamics. For instance, this may correspond to exponential decay given by

$$\dot{\mathbf{e}}_d = -\lambda\mathbf{e}$$

In essence, these equations let us formulate a kinematic goal to the manipulator as an inequality. Notice that the objective function is quadratic - it is made up of the square of the magnitude of the joint velocity vector and the square of the magnitude of the slack variables.

2.1.2 Prioritizing Tasks

Suppose solving Equation 2.4 results in

$$\mathbf{w} = \mathbf{w}^*$$

Now suppose we introduce another set of constraints (or in other words, another task) given by

$$\mathbf{J}_t^{(2)}\ \dot{\mathbf{q}} \geqslant \dot{\mathbf{e}}_d^{(2)}$$

These can be solved by the new set of equations

$$\begin{aligned} \underset{\mathbf{v},\dot{\mathbf{q}}}{\text{minimize}}\ & \frac{1}{2}(\|\mathbf{v}\|^2 + \lambda_r\|\dot{\mathbf{q}}\|^2) \\ \text{subject to}\ & \mathbf{J}_t\dot{\mathbf{q}} \geqslant \dot{\mathbf{e}}_d + \mathbf{w}^\star \\ & \mathbf{J}_t^{(2)}\dot{\mathbf{q}} \geqslant \dot{\mathbf{e}}_d^{(2)} + \mathbf{v} \\ \text{where}\ & \mathbf{v} \text{ is a slack variable.} \end{aligned} \tag{2.5}$$

In essence this formulation can be used to priotize tasks. Note that the second task is only executed in the null-space of the first task. As a consequence, the solution to the second task does not interfere with the solution to the first task. In other words, the second task has a lower priority than the first.

2.1.3 Example task: Reaching or avoiding a plane

Let us go back to the example presented in Section 2.1.1. Consider the plane given by equation 2.1

$$\mathbf{n}^T\mathbf{x} - a = 0$$

Now suppose we want to keep the end effector away from the plane. Keeping the end effector away can mean two things: staying either in the positive or in negative half-space associated with the plane. The positive half-space is given by,

$$\chi^+ = \{\mathbf{x} \in \mathbb{R}^3 |\ \mathbf{n}^T\mathbf{x} - a \geq 0\}$$

and the negative half-space is given by,

$$\chi^- = \{\mathbf{x} \in \mathbb{R}^3 |\ \mathbf{n}^T\mathbf{x} - a \leq 0\}$$

For simplicity, we will use *above the plane* to refer to the positive half-space and *below the plane* to refer to the negative half-space.

This task of keeping the end effector above or below the plane can be done by adjusting the inequality in Equation 2.4. The three cases can be summarized as follows:

- Leaving the inequality unchanged in Equation 2.4 corresponds to staying above the plane[2].

$$\mathbf{J}_t\dot{\mathbf{q}} \geqslant \dot{\mathbf{e}}_d$$

- Staying below the plane can be achieved by reversing the inequality sign:

$$\mathbf{J}_t\dot{\mathbf{q}} \leqslant \dot{\mathbf{e}}_d$$

- Reaching the plane can be achieved by combining both the above mentioned inequalities.

[2]For exponential decay error dynamics, when the signed distance is positive, $-\lambda e$ is negative. So zero joint velocity would ensure $\mathbf{J}_t\dot{\mathbf{q}} \geq \dot{e}_d$.

2.1.4 Reaching or avoiding primitive geometries

Tasks such as the one in Section 2.1.3 make use of the distance between the target geometry and the geometry representing the end-effector. Note that the task need not be associated to the end effector at all. In general the framework allows us to attach a primitive geometry to any link on the manipulator.

Also, the geometry attached to the target could be anything as long as the error can be represented mathematically as a distance to the geometry. To summarize, HiQP allows two kinds of tasks between pairs of geometries - *Projection* and *Alignment*. Projection tasks are aimed at reducing the distance between the two associated geometries. Alignment tasks are aimed at orienting two geometries together.

In this work we make use of the following tasks:

- Point on plane projection - keep the point at a certain distance on a specific side of the plane.
- Point on cylinder projection - keep the point at a certain distance either inside or outside of the cylinder.
- Line on line projection - make the lines intersect.
- Line on line alignment - keep the lines parallel to each other.

In the following section we describe the Grasp Planner which will output geometries and tasks of the above form. These tasks when fed into HiQP result in the the target object being picked up.

2.2 Grasp Envelopes

The Grasp Planner forms a crucial part of our picking framework. The grasp planner is described in detail by Stoyanov et. al. in [5] and is described briefly in this section.

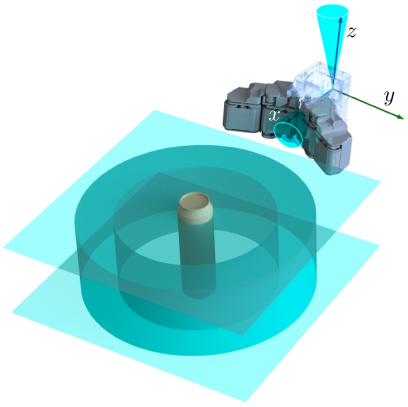
When given the pose and size of the object to be grasped, the Grasp Planner returns a set of primitive geometries (lines, cylinders and/or spheres) and constraints associated with these geometries. It is assumed that the object to be grasped can be fully enclosed in a cylinder (or sphere). And it is the size and pose of this cylinder (or sphere) that is given to the Grasp Planner. Figure 2.1a shows the target object, the enclosing cylinder and the region of grasp poses around the target object. This region of grasp poses is called the *Grasp Envelope*. The grasp planner uses sensory information and information about the gripper to remove invalid poses from this envelope. Figure 2.1b[3] shows one example of a truncated region returned by the grasp planner. The truncated region consists of four planes and two cylinders. When these constraints are fed into HiQP, the end effector is brought into this cylindrical sector.

The grasp planning operation can be divided into main stages:

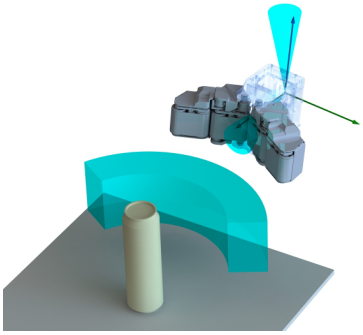
- Pre-computing a collision map.
- Computing valid configurations.
- Posting constraints to HiQP.

We discuss these steps briefly in the following sections.

[3] Figure taken from [5] with permission.



(a) An illustration of the grasp envelope.



(b) An example of the return primitives - they make up the cylindrical sector shown.

Figure 2.1: Grasp Envelopes and collision free regions

2.2.1 Pre-computing a collision map

The region around the grasp primitive (cylinder or sphere) is sampled evenly using three parameters. For a cylinder, the parameters are:

- the distance from the center of the cylinder, l
- the orientation, θ, and
- the height above the base of the grasp primitive, h.

For a sphere, the parameters are:

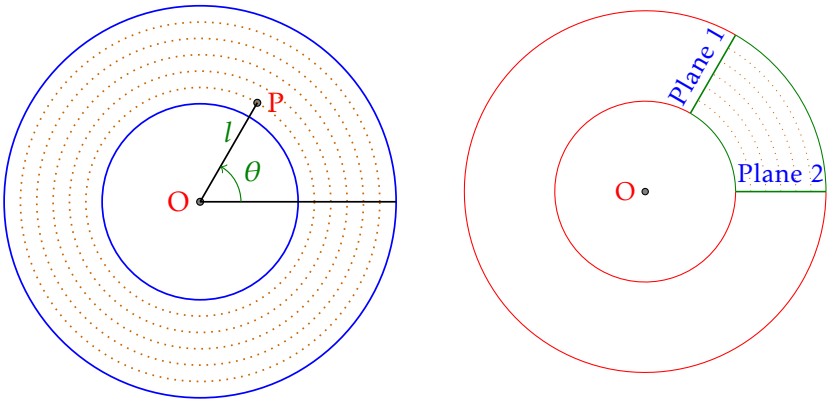
- the distance from the center of the cylinder, l
- the polar angle, θ, and
- the azimuth angle, ϕ.

The top view of such a space for cylindrical grasp primitive is shown in Figure 2.2a. Note that the distance from the center, l, is limitied: configurations where the palm of the gripper would collide with the target object and configurations where the fingers would miss the target object are not considered.

The Grasp Planner constraints the orientation of the gripper to align with the principal components of the target object. In other words, the roll axis of the gripper is constrained to point towards the cylinder axis and the yaw axis is constrained to be parallel to the cylinder axis. These constraints are inspired by how humans grasp objects [10]. Given these constraints each orange dot in Figure 2.2a corrresponds to one pose of the end effector where a grasp might be possible. And for each such configuration of the end effector, a collision map is computed where the cells corresponding to the gripper are marked as occupied and the remaining cells are marked free.

2.2.2 Computing valid configurations

The next step is to validate configurations where the gripper is not in collision with the environment. The environment in our case is represented by a



(a) Top view of the search space (b) Max volume region after search

Figure 2.2: Grasp Planning

Signed Distance Field. In an SDF grid, every voxel contains the distance to the nearest occupied voxel. Note that although we use a TSDF, any representation where occupancy information is available can be used.

Given the collision maps computed in the previous step, we intersect each of these maps with the environment map. This is done by associating corresponding cells in both the maps. If both cells are occupied, there is a potential collision. Three types of collision are possible: 1) palm of the gripper collides with the environment; 2) fingers collide with the environment; 3) fingers collide with the target object. A collision between the gipper palm and the environment results in the corresponding sample being marked as invalid.

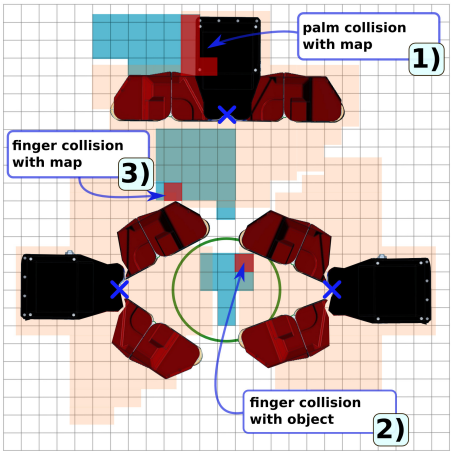


Figure 2.3: An illustration of the types of collisions detected by the planner.

If this collision is between the fingers of the gripper and the environment or target object, then the gripper opening angle is constrained. A collision between the fingers and target object would constraint the minimum opening angle. A collision between the fingers and the environment would constrain the maximum opening angle. If the maximum opening angle is less than the minimum opening angle, the corresponding sample is marked invalid. Figure

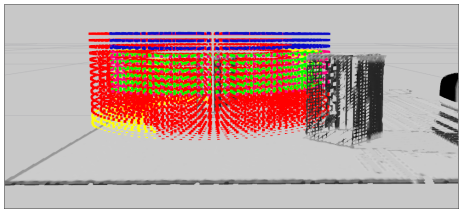
2.3[4] illustrates the three different types of collisions detected by the planner.

2.2.3 Constraints

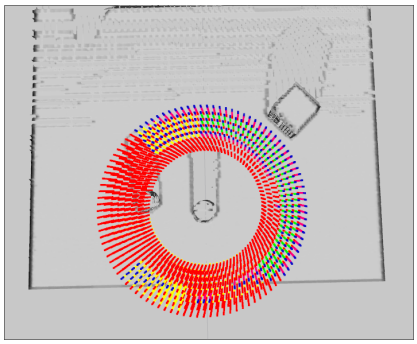
After the above two steps, some configurations have been marked as invalid while others remain valid. At this stage it is necessary to find a continuous region of valid configurations. In particular the region with the maximum volume of all such regions is found. For instance, this looks like Figure 2.2b for a cylindrical grasp primitive. For a detailed explanation of the algorithm the reader is invited to read the work by Stoyanov et. al. [5]

Now that we have the maximum volume region, note that it can be reduced to a few inequalities. In words, the inequalities are equivalent to:

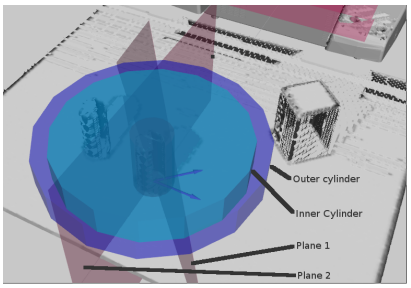
- Stay outside the inner cylinder, and inside the outer cylinder (two point on cylinder projection tasks).
- Stay between planes 1 and 2 (two point on plane projection tasks).
- Stay above the bottom plane and below the top plane (two point on plane projection tasks).
- Keep the end-effector upright (line on line alignement task).
- Make the end-effector face the target object during approach (line on line projection task).



(a) Side view of the search space



(b) Top view of the search space



(c) The resulting constraints

Figure 2.4: Grasp Planner: search space and constraints visualization

As mentioned earlier the last two constraints are inspired by research into how humans pick up objects. Note that these inequalities are exactly similar to what we saw in Section 2.1.3 and subsequently in Section 2.1.4. When fed into hiqp, the gripper ends up in a configuration where it is ready to grasp the

[4]Figure taken from [5] with permission.

target object. This is the first stage in the picking operation. A visualization[5] of the search space and the extracted envelope is shown in Figure 2.4 and Figure 2.5. The color of the displayed samples is described below:

- Red - Palm collides with environment
- Green - Selected valid pose. Is part of the returned envelope.
- Purple - Valid pose. Not part of the returned envelope.
- Yellow - Finger collides with environment or target.
- Blue - Space between fingers is empty in this configuration.

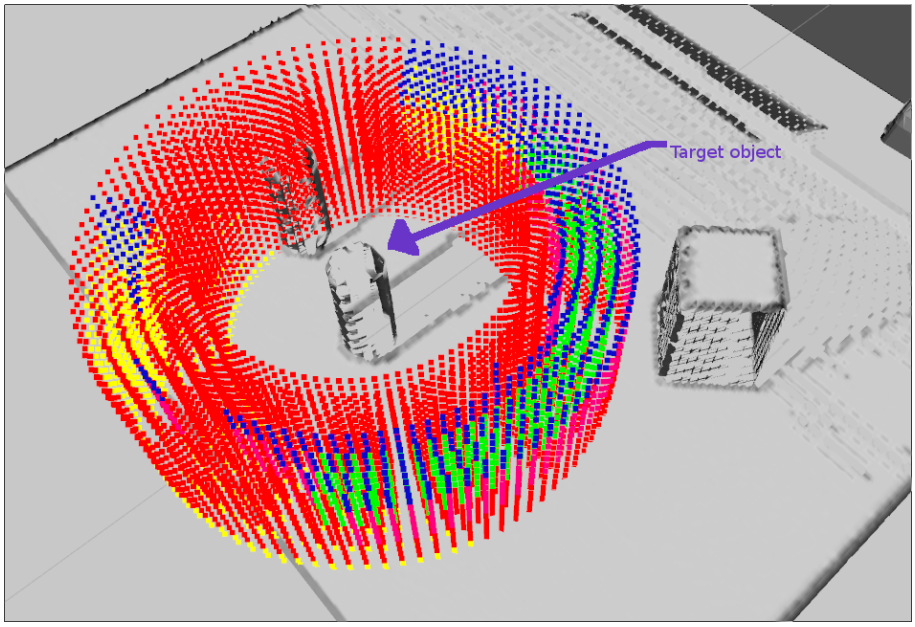


Figure 2.5: Another view of the search space

Once the object is ready to be grasped, it is grasped and extracted. Extraction of the object is also an empirical operation where the object is simply lifted off of its original position. This is done by simply modifying the bottom plane and top plane primitives mentioned before.

2.2.4 When there are obstacles in the way

The grasp planner can deal with obstacles that lie within the sampled region that is used for planning. But consider the situation in Figure 2.5. An object can be seen just outside the sampled region around the target object. When the constraints generated by the grasp planner are fed into the HiQP controller, the manipulator can potentially collide with this object. It is easy to imagine numerous other cases where the manipulator can potentially collide with the environment even in simple picking tasks. For instance, if the object to be picked up is on a shelf, the manipulator's links can potentially collide with the racks in the shelf. It is therefore necessary to have an obstacle avoidance behaviour built into the framework to help make it more robust. The following chapter is dedicated to obstacle avoidance using the HiQP framework and the Signed Distance Transform representation.

[5] This image is a screenshot of rviz when performing experiments on the robot.

Chapter 3
Obstacle avoidance using HiQP

3.1 Generalizing the avoidance task

In Section 2.1.3 we saw how a constraint can be formulated to prevent the end effector from reaching a plane. This idea made use of two quantities: the distance to the plane and the direction in which this distance increases. Sadly, the environment is not made up of planes and in order to generalize this idea to work with any surface, we need a representation that can provide us the distance to the surface and the direction in which this distance increases.

The Signed Distance Field representation provides exactly that. Let us start with the error formulation and differentiate that to obtain the error dynamics. The error is given by,

$$e(\mathbf{q}) = D(\mathbf{p}(\mathbf{q})) \tag{3.1}$$

And the error derivative becomes,

$$\mathbf{J}_t\dot{\mathbf{q}} = \nabla D(\mathbf{p})^T \mathbf{J}_p\dot{\mathbf{q}} \tag{3.2}$$

where $D(\mathbf{p}(\mathbf{q}))$ is the value of the distance field at a point $\mathbf{p}$ and $\nabla D(\mathbf{p})$ is the gradient of the distance field at point $\mathbf{p}$.

An obstacle avoidance behaviour can make use of this basic idea to stay away from obstacles. This chapter focuses on the use of Signed Distance Fields for obstacle avoidance. Signed distance fields have been used to represent 3D volumes in computer graphics [11], in surface construction from sensor data [12], and recently, in optimization based motion planners [13]. Research by Oleynikova et al. [6] illustrate the advantages of using SDF maps as both a representation for online map building and online motion planning.

Some of the advantages of SDF Maps that we would like to exploit are:

- SDF Maps allow fast look-up of closest distance to obstacle surface.
- SDF Gradients provide the direction to the closest obstacle which can be used with our HiQP task formulation to restrict velocity.

The above formulation in Equation 3.1 and 3.2 raises the following question:

How should the manipulator itself be represented? The above equations implicitly assume that there is only one end effector point that needs to be kept away from obstacles. In the most general case, this is not true. The entire manipulator should be kept away from obstacles. One way to do this is to represent the manipulator as a combination of different geometric primitives such as points, spheres, cylinders, etc. Once this is done, we can rewrite the error and task jacobian formulations to work with these primitives and our chosen scene representation format (SDF). This is discussed in Section 3.4.

3.2 Related Work

In the previous section we raised the question: "*How should the manipulator be represented*". Optimization-based motion planners such as CHOMP [13] and STOMP [14] approximate the robot body using a set of spheres. This is inadequate for our work because the velocity of the manipulator in the region between two such spheres would be unrestricted. This means that a collision might occur between the robot link and the environment in a region between the two spheres.

Sugiura et al. use a combination of sphere-swept line segments to approximate the robot for use in self-collision avoidance [15]. Sphere-swept line segments were also used by Kanoun [16] in their work for self collision avoidance. We call these sphere-swept lines, capsules. Note that a capsule can be reduced to a line segment plus a radius for distance computation. For instance, the closest distance from a manipulator link to an obstacle, is the closest distance to the obstacle from the line segment minus the radius. This makes a capsule computationally simple while reasonably approximating the shape of manipulator links. In our work we use capsules to represent the manipulator links.

The artificial potential field approach for obstacle avoidance was introduced by Khatib [7]. To quote Khatib, the goal point *"is an attractive pole for the the end effector, and obstacles are repulsive surfaces for the manipulator parts"*. Note that potential fields are akin to SDF Gradients - both are vector fields, where, at any point in the field, the magnitude of the field is proportional to the distance from the obstacle and the direction of the field indicates the direction in which the distance from the obstacle decreases[1].

On an abstract level, our method is similar to the artificial potential field approach. The SDF map can be thought of as an implementation of the potential fields concept. The artificial potential fields approach was further explored and integrated into a task hierarchy by Dietrich et. al. in their work[8]. Due to the absence of a joint effort interface to our robot, we could not use the effort-based control scheme.

Another important study is one by Kanehiro et. al. [17], where they explore a local method for collision avoidance. The obstacles are represented by non-strictly convex polyhedra. These polyhedra are then decomposed into simpler geometries from which constraints are generated. The main disadvantage is that in complex environments, the computation time can be drastically large (as mentioned in the work). Also, it is assumed that the full geometry of the obstacle is available.

[1] Other examples of potential fields are the Gravitational Field and Electric Field.

3.3 Scene Representation

3.3.1 TSDF and ESDF

In this work, we use the *Signed Distance Field* or *SDF* (sometimes also referred to as the Signed Distance Transform) to represent the scene in which manipulation is performed. The distance function maps an n-dimensional vector to a scalar using a distance metric. If $\mathbf{x}$ denotes a point in n-dimensional space, then the SDF is mapping given by:

$$D(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R} \tag{3.3}$$

If the metric used is Euclidean distance, it is called the Euclidean Distance Transform [18]. For example, distance from a circle (with centre at the origin) is given by the equation:

$$D(\mathbf{x}) = \|\mathbf{x}\| - r \tag{3.4}$$

where r is the radius of the circle. For points on the circle, the above equation becomes the equation of the circle, as expected. For points inside the circle, the function produces a negative value, and for points outside the circle, the function produces a positive value. Such a function is called a *Euclidean Signed Distance Function (ESDF)*. In 3-D, the notion of positive and negative distances can be used to know if a point is inside or outside a specific object.

A voxel grid is usually used to represent a discrete SDF. Each voxel contains the distance to the nearest surface. When using a depth camera, such as the Kinect or Xtion, one typically gets only one view of the scene if no reconstruction is performed. It should be easy to note that in the absence of a complete reconstruction of the scene, one can not compute the ESDF. However, it is easy to compute the distance to the surface along the direction of a ray from the center of the sensor. Such a distance metric is an approximation of the true Euclidean distance. This is because the distance is not computed along the normal to the surface. The SDF thus obtained is called a projective SDF as it uses a projection of the Euclidean distance along the ray. Hence, it is usually truncated to only have values very close to the surface. This representation is called a *Truncated Signed Distance Field* or *TSDF*. For more information on how a TSDF is generated and updated, please refer to the work by Canelhas [19].

3.3.2 Gradient of the SDF

The gradient of the SDF is a vector whose components are partial derivatives of the SDF with respect to each spatial dimension.

$$n(\mathbf{x}) = \nabla D(\mathbf{x}) = \begin{bmatrix} \partial D(\mathbf{x})/\partial x \\ \partial D(\mathbf{x})/\partial y \\ \partial D(\mathbf{x})/\partial z \end{bmatrix} \tag{3.5}$$

The gradient tells us how the SDF values vary along each dimension. In other words, it tells us along what direction the SDF value increases. In Euclidean distance grids, the gradient is normal to the obstacle surface. In practice, the gradient is computed numerically using central differences. In our implementation we use 2nd order central differences along each dimension.

$$\nabla D_x(\mathbf{x}) = \frac{D(\mathbf{l}+2\mathbf{h}) - D(\mathbf{l}-2\mathbf{h}) + 8D(\mathbf{l}+\mathbf{h}) \ - 8D(\mathbf{l}-\mathbf{h})}{12\ |\mathbf{h}|}$$

where **l** is the location where we wish to compute the gradient and **h** is a unit vector in the x-direction scaled by the resolution of the SDF grid. This process is repeated (by changing **h**) for y and z directions.

In order to be able to use the SDF Map for obstacle avoidance we need,

- Gradients in every cell of the SDF Map
- SDF value in every cell of the SDF Map

In a TSDF grid, the gradients will point towards the origin of the sensor. This is because the distances are computed along the direction of the ray. Also, SDF values are only available for cells close to the observed obstacle surfaces. Hence, we convert the TSDF grid into an ESDF grid. This is done using the TSDF grid as a starting point and computing the minimum distances along each dimension. The algorithm is described in detail in the work by Felzenszwalb and Huttenlocher [20].

3.4 SDF Maps and manipulator geometry

In this section we discuss the problem of obstacle avoidance when the manipulator is represented as a combination of points, spheres and capsules.

3.4.1 Keeping a manipulator point away from obstacles

Suppose the manipulator is represented, for the purpose of obstacle avoidance, by several points at different locations on it. Let us call these the *test points*. And we want to keep each of these points away from the obstacle surface. We discussed the task of avoiding collision with a surface (represented by an SDF map) in Section 3.1. Note that the SDF value at each of the test points gives us the distance to the closest point on the surface. Also, the SDF gradient gives us the normal to the surface.

Note that since the generalized function in Equation 2.4 contains a greater-than-or-equal-to relation, it is necessary to add a safety distance to the error formulation to prevent the controlled point from actually reaching the obstacle surface should there be a competing task of lower priority.

3.4.2 Keeping a manipulator sphere away from obstacles

When the manipulator is represented as a combination of spheres, the formulation is quite similar to that of a point with a slight difference. The jacobian stays the same. The radius r of the sphere is subtracted from the error value. The error is simply

$$e(\mathbf{q}) = D(\mathbf{p}(\mathbf{q})) - r \tag{3.6}$$

Note that the shortest distance to a sphere is simply the shortest distance to the centre of the sphere minus the radius, and hence, the equation for error.

3.4.3 Keeping a manipulator capsule away from obstacles

When each link of the manipulator is represented as a capsule, the formulation is not quite straight-forward. Several questions arise:

- *What points on the capsule should be constrained?* In the above formulations, both primitive geometries used can be reduced to a point. So in effect, the velocity of this point is constrained. Kanehiro et al. [17] show

that at least a pair of points (namely the end points) on a line segment need to be constrained to ensure collision avoidance between that line segment and a plane. However, this over simplifies the problem. Ideally we want to contrain the velocity of a link taking every obstacle into consideration.

- The naïve approach is to sample points on the capsule's axis at the resolution of the underlying environment representation, in our case the SDF map. This leads us to our next question, *"are all the constraints necessary?"* If not, it should be possible to reduce these constraints to a smaller number of quivalent constraints. For instance when avoiding collision with a plane only two constraints are needed and these need to be at the end points of the line segment.

In this work, we will adopt the aforementioned naïve method for obstacle avoidance and evaluate it thoroughly. In the following section, we present two simple methods to reduce constraints.

3.5 Constraint Reduction

The naïve method mentioned in the previous section works well in practice but results in a large amount of time spent on optimization when used at the highest resolution. Per iteration, it takes about 12 *ms* for optimizaiton[2]. This means optimization can be run at a frequency of utmost 83 Hz. It is also quite clear from experimentation (please refer to Figure 4.8 in Section 4.2.1) that a reduction in the sampling length reduces the time spent on optimization.

At this juncture, we wish to answer the question, *"can we intelligently prune out constraints such that we have as much information as possible while reducing the number of constraints significantly"*? Although the answer is *yes*, the problem of reducing these constraints is not quite straight forward. In this work we present a few methods to prune out constraints. These are dealt with in the following sections.

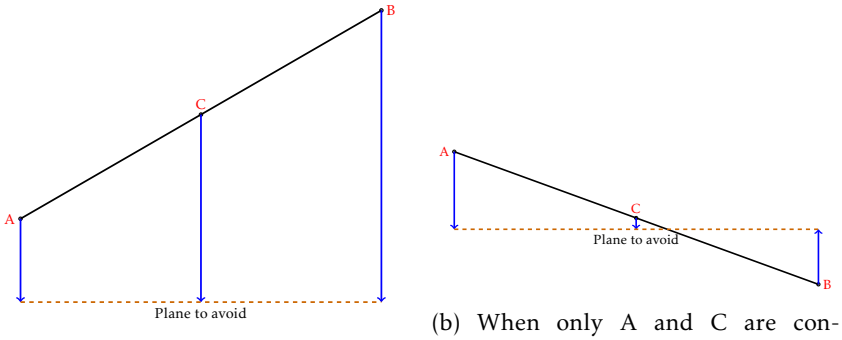
3.5.1 Keeping a line segment away from a Plane

In order to keep a line segment *above*[3] the plane, at least two points need to be constrained and the two points should be the end points. This statement is easily proved graphically. For instance, consider the line segment in Figure 3.1. When points A and C are constrained with respect to the plane below, the segment is allowed to move to configuration shown in Figure 3.2 in the next time instant. This is because, point C has not violated its velocity constraints in the direction of the plane and point A has zero velocity in the direction of the plane. However, the line segment can still collide with the plane.

When points A and B are constrained, any point between A and B on the line segment, and hence point C, is automatically constrained. For instance, for point C to collide with the plane either point A or B or both should also collide with the plane. This is not possible since A and B have already been constrained. This result gives us the easiest way to prune out constraints when there are multiple constraints against the same plane or set of planes. We can

[2]Th experiments were performed on an Intel Core i7 (3.4 GHz × 8) machine running GNU/Linux.

[3]Above = on the same side as the normal

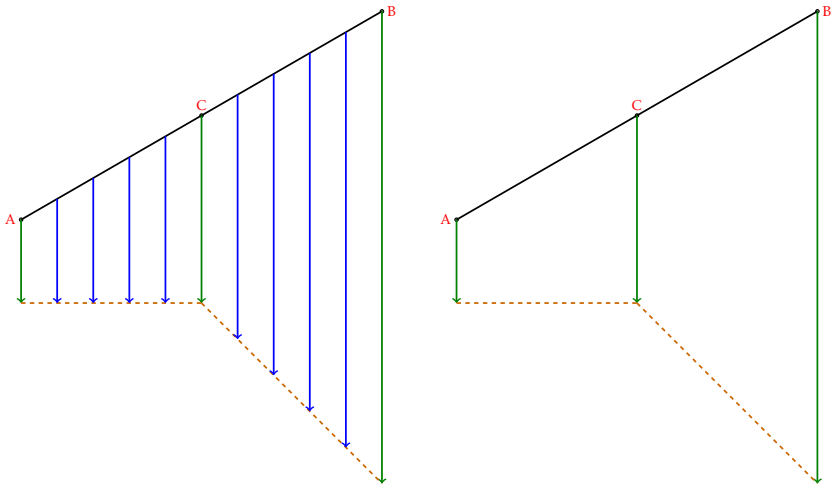


(a) Line segment vs Plane

(b) When only A and C are constrainted

Figure 3.1: Constraining a line-segment with respect to a plane.

keep the outer most constraints and dispense with the others as shown in Figure 3.2a and Figure 3.2b. In practice, this method helps reduce the number of constraints if there are planar surfaces like tables, boxes, etc. When planar surfaces are scarce, this method would not help much in reducing the number of constraints.



(a) All constraints

(b) Redundant constraints removed

Figure 3.2: Reducing constraints related to the same plane.

3.5.2 Minimum-in-region sub-sampling

Another way of reducing constraints is by sub-sampling the constraints in each capsule. In this particular method we divide the length of the line-segment (longitudinal axis of the capsule) into several equal parts. We then compute the SDF gradients at the lowest resolution and retain only the gradient that is smallest in every region.

Suppose l is the length of the capsule associated with the link in question. Suppose d is the discretization length and s is the subsampling length. Total number of constraints in the naïve method would be,

$$no.\ of\ constraints = \frac{l}{d}$$

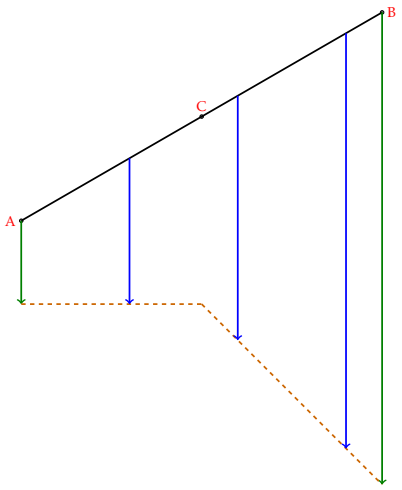


Figure 3.3: Application of minimim-in-region subsampling to Fig. 3.2a

As an example, for a 30-centimeters-long link, and a discretization length of $0.003m$ the number of constraints would be $\frac{0.3}{0.003} = 100$. However, subsampling would reduce it to a fraction of that value. For instance if we choose one constraint for every 2 constraints (by picking the minimum SDF Gradient), we would already reduce the no. of constraints by half. An illustration of this method is shown in Figure 3.3.

Chapter 4
Experiments

To evaluate our system, three different types of experiments are performed. This chapter deals the purpose and context of each type of experiment including the the description of the set-up used (Section 4.1) and a discussion of the results (Section 4.2).

4.1 Outline of experiments

This section describes the context and purpose of each experiment we wish to perform. Additionally we describe the setup used for each experiment in detail.

For all the experiments performed the manipulators' links are modelled as shown in Figure 4.1. Each capsule begins at frame i to the origin of frame $i+1$. Each capsule is empirically made to fit the entire link it is attached to. Only the last four links are modelled for our experiments since the other links are unlikely to collide with obstacles in our particular set-up.

To recreate each of the above shown scenarios on the real robot, the setup shown in Figure 4.2 is used. On a sheet of paper, the positions of objects (taken from the gazebo models) are marked out for easy repeatability of experiments on the robot. Every obstacle avoidance experiment was repeated 8-10 times.

We performed all the experiments on the Yumi robot. We use ROS as the middle-ware[1]. In simulation the Yumi robot is simulated using Gazebo.

4.1.1 Goal pose in collision (pre-generated ESDF)

What: The end effector is given a final pose where it would definitely collide with the environment. The setup used is shown in Figure 4.3a. The model of a plane and the stanford bunny is loaded onto Gazebo. The same model is used to generate an ESDF.

In order to generate an ESDF for use in these experiments the following is done:

- Create a simple 3D model of the setup[2]. For this experiment, we used a flat object to represent the table and we used the Stanford bunny as the obstacle. The target pose is in collision with the bunny.

[1] ROS drivers for Yumi can be found at `https://github.com/OrebroUniversity/yumi`.
[2] We used the open-source program, blender, to create our models.

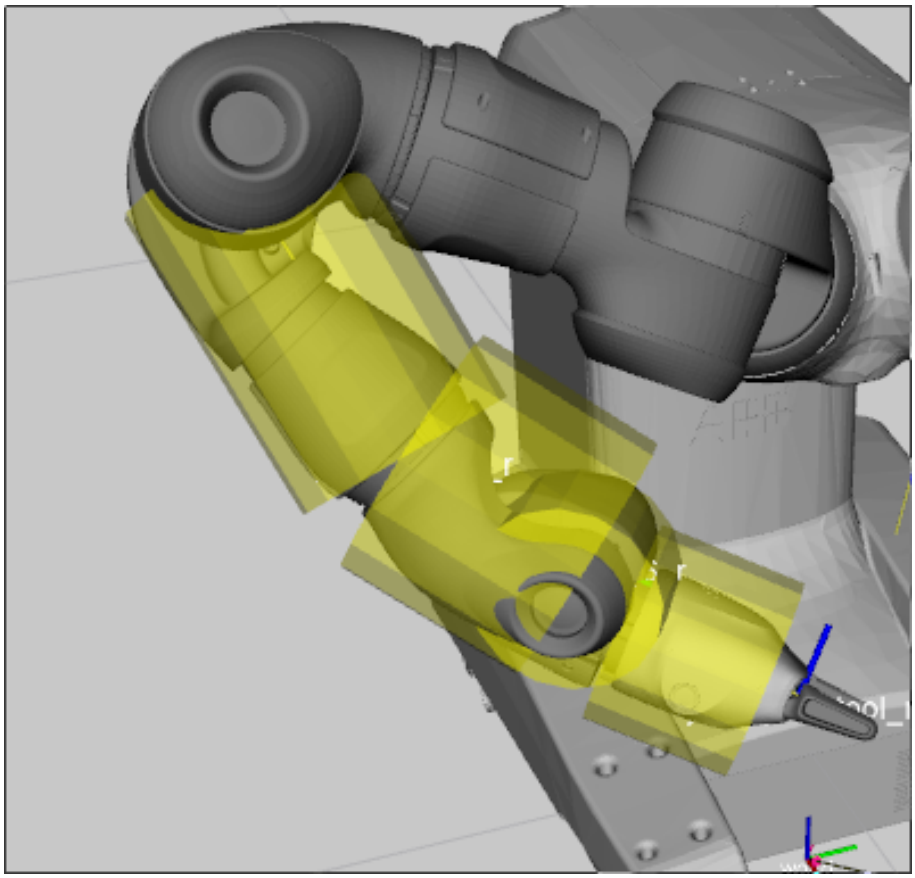


Figure 4.1: Cylinders/capsules around each manipulator link.

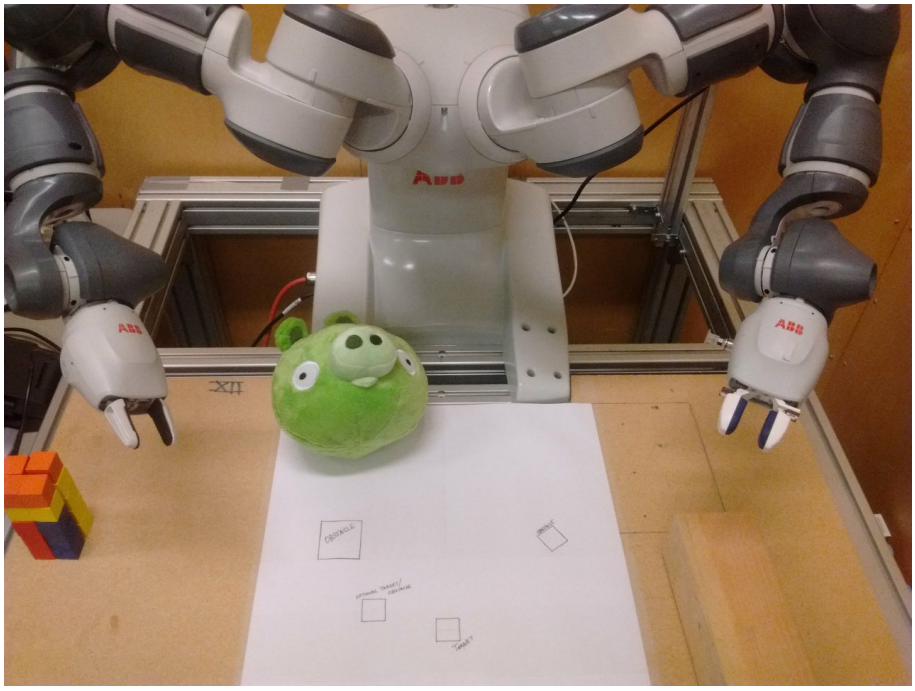
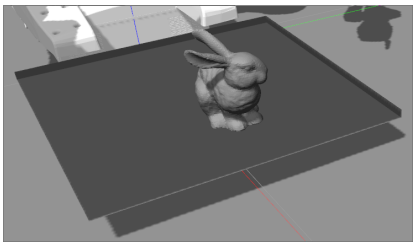
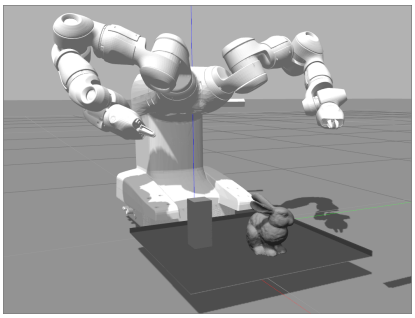


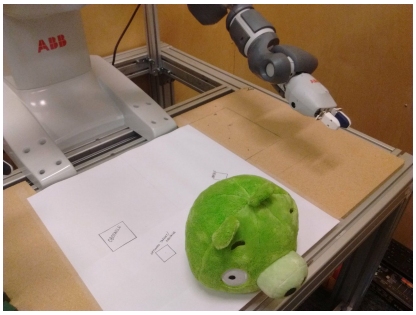
Figure 4.2: The setup used for experiments on the robot



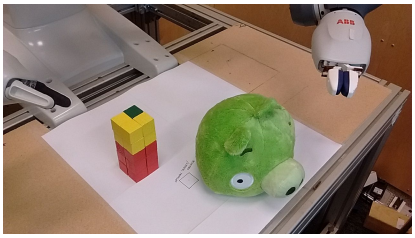
(a) Bunny setup - gazebo



(b) Bunny and box setup - gazebo



(c) Pig setup - real



(d) Pig and box setup - real

- Import the above 3D model into gazebo to create a world. Use the 3D model to create an ESDF[3]. This is computed once only.

Why: The most basic goal of the obstacle avoidance system is that it should not collide with the environment. This is the simplest case to test the system since there is only one obstacle and the end effector is prevented from colliding with it.

Measured quantity: Binary result - either collision occurs or not.

4.1.2 Goal Pose in collision with obstacles in the way (pre-generated ESDF)

What: In this experiment we use the same setup as in Section 4.1.1 except that now there is a box in the way. In the previous experiment (when there is no box), the path the robot takes is noted and an obstacle is placed where it would likely collide with the robot. The system is run once without obstacle avoidance to make sure that the box is a potential obstacle. The experimental setup is shown in Figure 4.3b (simulation) and in Figure 4.3c. The experiment is performed for varying values of sampling length (of the capsule primitive) for the computation of SDF gradients.

Why: Aim is *not* to evaluate if the system can make the manipulator successfully navigate around the obstacle. It is to make sure that manipulator can avoid obstacles. Also we wish to determine what sampling length is needed to successfully avoid obstacles and to understand if the sampling length has an impact on the optimization time.

Measured Quantities:

[3]We used the SDFGen program to generate the ESDF from STL meshes. The package can be found at `https://github.com/tstoyanov/SDFGen`

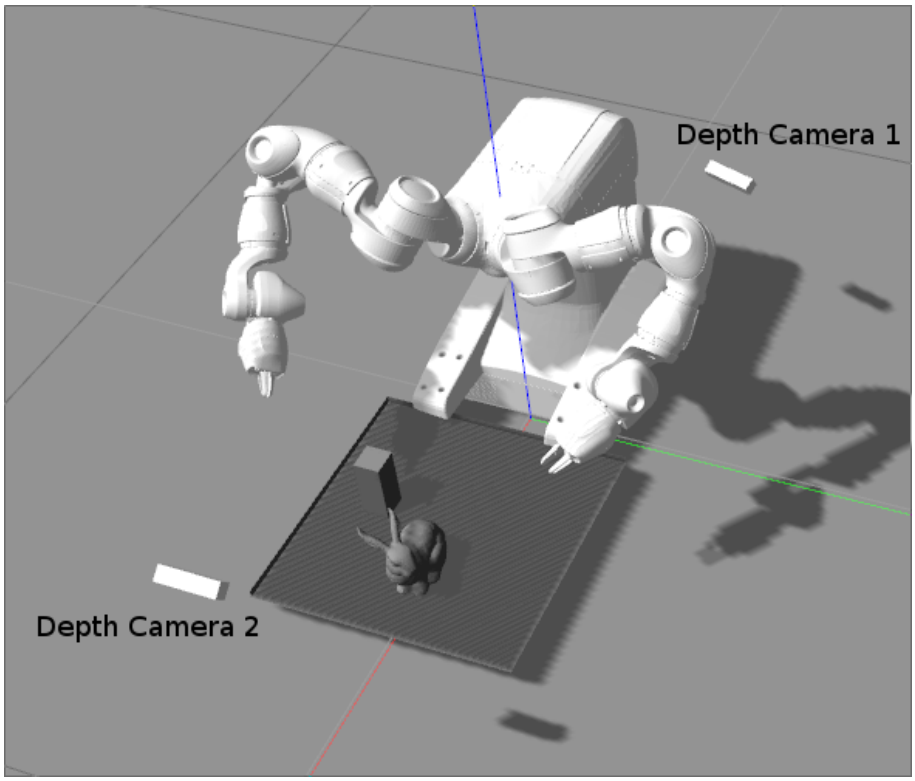


Figure 4.4: The world: A ground plane, a table setup and two depth cameras.

- Binary result - either collision occurs or not.
- Velocity controls computation time - This is the time taken for HiQP to solve the optimization problem. We would like to study how the optimization time increases as the number of constraints increases. We vary the sampling length to vary the number of constraints.

4.1.3 Goal Pose in collision with obstacles in the way (online SDF)

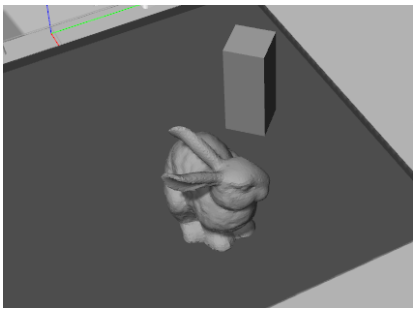
This experiment is exactly the same as experiment in Section 4.1.2 with one key difference - the SDF Map is generated online using depth sensors. These depth sensors are added to gazebo and gazebo plugins are used to stream the depth data back to ROS. The gazebo setup is shown in Figure 4.4.

Alternative setup

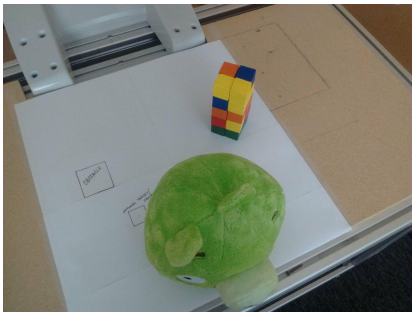
For the experiment in Section 4.1.3, the obstacle place in the way is occluded from one of the sensors. So the same experiment is done again with obstacles placed on the opposite side (right-hand-side for the viewer). The setup is shown in Figure 4.5.

4.1.4 Picking with obstacles in the way

What: Pick up an object with obstacles in the way, using our framework and using Moveit. Similar to the obstacle avoidance experiments, the picking operation is first conducted without the obstacle avoidance task and it is made sure that the manipulator would collide with the obstacle.

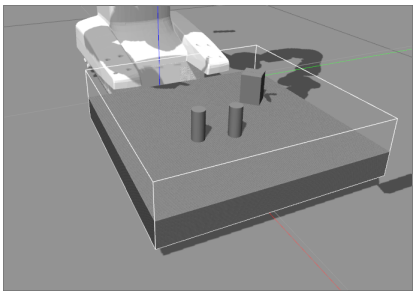


(a) Bunny and box alternative setup - gazebo

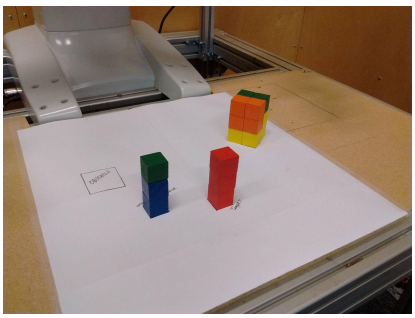


(b) Pig and box alternative setup - real

Figure 4.5: Alternative setup for obstacle avoidance experiments



(a) Picking experiment setup - gazebo



(b) Picking experiment setup - real

Figure 4.6: Setup for picking experiment

Why: To compare the movement of joints due to HiQP and Moveit. We do not wish to test if HiQP can successfully navigate around complex obstacles. We know that HiQP would fail to navigate around the obstacles if local minima exist. However, we wish to consider cases where HiQP would be able to find a path. We then feed the destination pose into moveit (we know this pose is collision free).

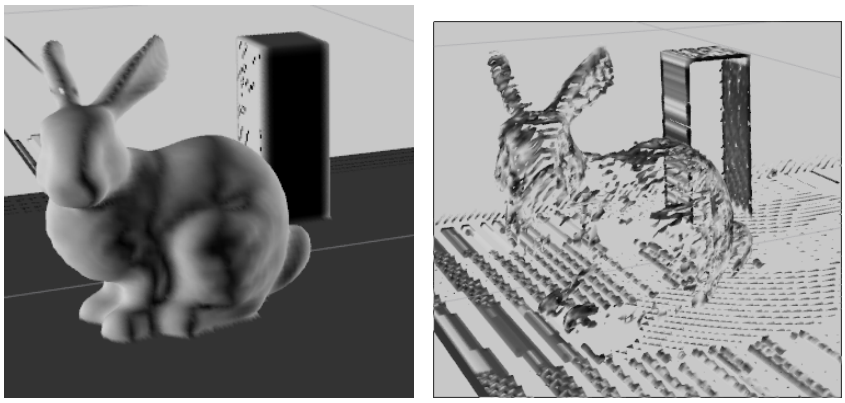
Measured Quantities: We record all joint states. In the end we wish to measure how much the joints have moved. To do this we incrementally sum up all the joint displacements.

4.2 Results

4.2.1 Simple obstacle avoidance

For the simplest obstacle avoidance experiment, the experiments showed a 100% success rate for various sampling lengths ranging from 0.003 m (resolution of the underlying SDF map) to 0.05 m in both simulation and on the robot[4].

[4]It should be noted that in the absence of any obstacles in the way, a sphere at the tip of the last link would be enough to prevent end effector collision.



(a) Pre-computed ESDF (b) Online Map

Figure 4.7: Online and offline computed SDF maps

4.2.2 Goal Pose in collision with obstacles in the way (pre-generated ESDF)

Table 4.1 outlines the results from the obstacle avoidance experiment when there is a box in the way and when a pre-generated ESDF is used.

Sampling length (m)	**Success rate (%)**
0.007	100
0.009	100
0.01	100
0.02	100
0.03	25

Table 4.1: Success rate for different sampling lengths - Obstacle in the way (pre-generated ESDF)

It is quite clear that increasing the sampling length above 0.02 m affects success rate. In the naïve method for adding constraints, we are essentially approximating the capsule with several spheres. When we increase the sampling length, the spheres might be too far apart that when an obstacle appears in between two such spheres, there is a greater chance of collision (due a laxer velocity constraint).

4.2.3 Goal Pose in collision with obstacles in the way (online SDF)

Simulation

When the SDF tracker is used online, some issues begin to show up. Firstly, due to their positions relative to the workspace, the cameras do not get a view of the space from every angle. This leads to gaps in the SDF map. For instance (see Figure 4.7) the lateral faces of the box are not visible to the sensors. This explains the results from using the online tracker. It is quite clear that even at highest resolution the success rate is quite low. These results are summarized in the table below.

sampling length (m)	Success rate (%)
0.003	50
0.005	50
0.008	25
0.01	37.5
0.03	0

Table 4.2: Obstacle in the way using online SDF in Simulation

Robot

Table 4.3 summarizes the results from the robot.

sampling length (m)	Success rate (%)
0.003	33
0.006	12
0.008	0

Table 4.3: Obstacle in the way using online SDF on the Robot

The success rate is slightly worse than in simulation. This could be due to various factors including extrinsic calibration between the cameras. For our experiments we measured the camera placement using a tape to have a starting point for the transformation. We then adjusted the transforms until the point clouds from the two sensors aligned with the robot model in RViz.

Alternative setup

In order to verify that the failures are indeed due to occlusion, the experiment is conducted with the box placed at another location. The table in Figure 4.5b shows the results for the alternative setup.

sampling length (m)	Success rate, simulation(%)	Success rate, robot(%)
0.003	100	100
0.006	100	88
0.008	100	45
0.015	0	-

Table 4.4: Obstacles in the way, alternative setup using online SDF

Placing the obstacle at a different location (as shown in Figure 4.5) drastically improves obstacle avoidance success rate. In the new location, the obstacle is more clearly visible and hence the resulting SDF map more accurately models the real world.

4.3 Minimum-in-region sub-sampling

We performed an evaluation of the minimum-in-region sub-sampling method for reducing the number of constraints. This experiment was performed on the alternative setup mentioned in the Section 4.1.3. This experiment was performed only in simulation.

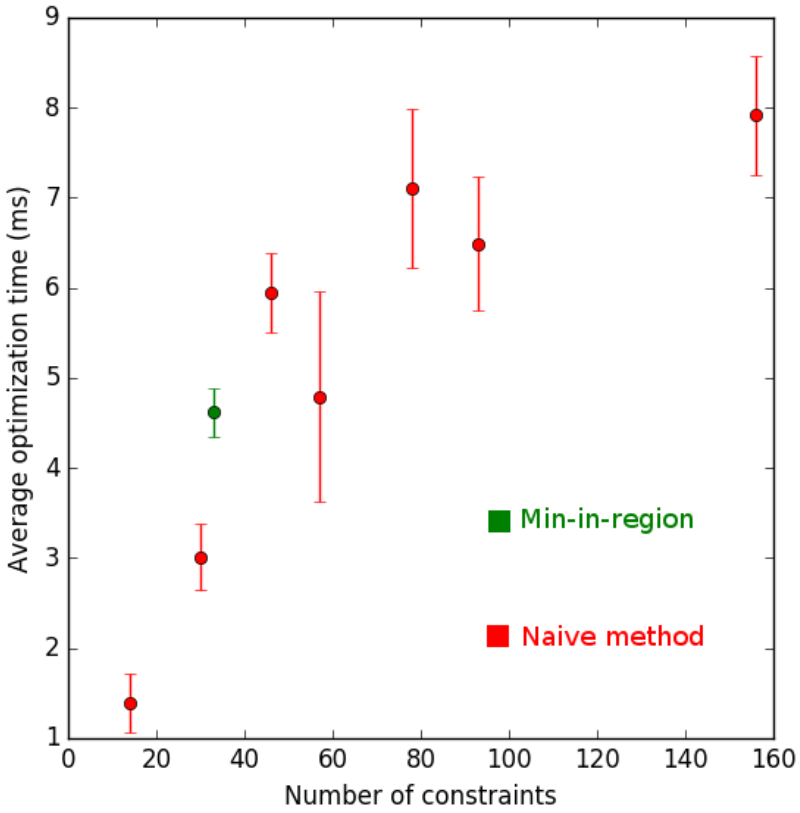


Figure 4.8: Average Optimization time vs number of constraints

When a sub-sampling length of 0.015 m was used along with a sampling length of 0.003 m, the success rate was 100% (for 10 runs). Another interesting observation is the relationship between the time required for computing velocity controls (optimization time) and the sampling length. It is clear from Figure 4.8 that average optimization time increases significantly with an increase in the number of constraints.

4.4 Object picking

As already mentioned, the object picking experiments are mainly designed to compare the joint motions resulting from our framework and those from Moveit. For these experiments, the SDF tracker is initialized with a cell size of 0.005 m (side of each cell cube) and Moveit's Occupancy Map updater is initialized to the same cell size. With Moveit the default RRT planner from OMPL is used.

Firstly, the picking is conducted with our framework (HiQP + obstacle avoidance + grasp planner). This experiment is performed several times. The resulting end effector pose from these experiments is fed into Moveit's planning pipeline and the resulting motion is also recorded.

The graph in Figure 4.9 shows the results from the experiment. Here the joint motions are calculated by measuring the norm of the joint distance vec-

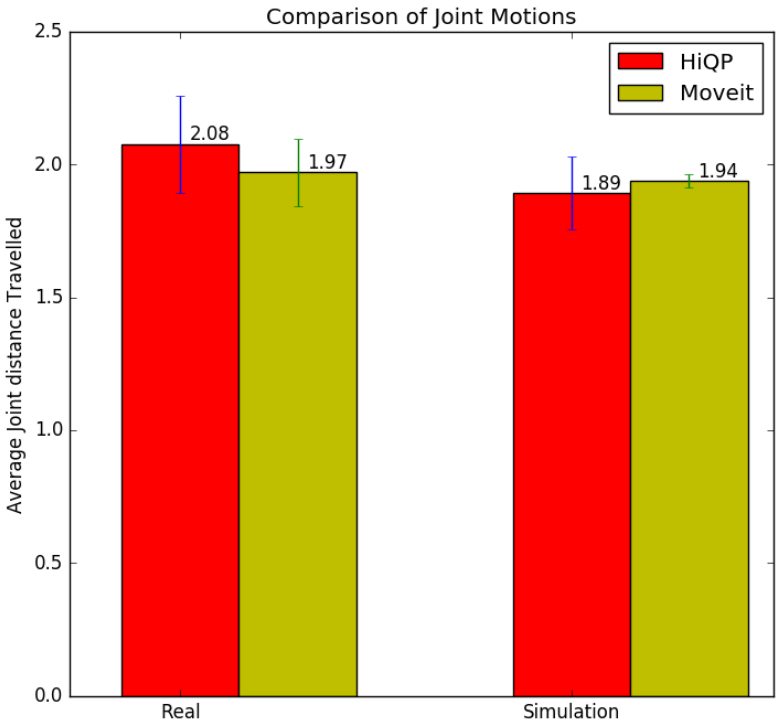


Figure 4.9: Comparison of Joint Motions

tor. Note that this is not the joint displacement vector. Although very different joint trajectories may have the similar joint displacement, the joint distances will be different. We integrate incremental joint displacements to compute the joint distance vector. It is clear that in both simulation and real experiments, Moveit and the HiQP-based picking framework result in similar joint motions.

4.5 Discussion

The results from these experiments can be summarized as follows:

- The obstacle avoidance experiments serve as a strong baseline for further experimentation and future work.
- The experiments suggest that in the presence of a perfect SDF Map, collisions do not occur when the sampling length is low enough.
- Velocity controls computation time seems to increase drastically with increase in resolution (decrease in sampling length) at which the capsule is sampled. This suggests strongly that constraint reduction techniques could be of great advantage.
- Results from the minimum-in-region subsampling indicate that this method is probably an effective method for computing constraints.
- The joint motions resulting from the system are similar to those resulting from Moveit.

Perhaps the result that is most worthy of discussion is the one from the minimum-in-region sub-sampling. As expected this reduces the velocity computation time drastically. And though only one sub-sampling length ($0.015\ m$) was tested, it produced a 100% success rate. When a sampling length of $0.015\ m$ was used, it produced a 0% success rate. This can be due to several reasons. Firstly, although sub-sampling is performed, all the information available for a capsule is being used. Another reason could be that the sub-sampling makes kinematic sense. The sub-sampling actually splits the line segment into multiple smaller line segments. Lets call these smaller segments sub-segments. The method literally chooses the most critical constraint for each of these sub-segments. It is possible that this critical constraint supplants other constraints in the sub-segment. Further theoretical and practical study on constraint reduction would certainly help us understand the problem more clearly.

Chapter 5
Conclusion

5.1 Future Work

In section 3.5 we have suggested potential methods for reducing constraints. Although one of the methods (minimum-in-region subsampling) has been evaluated, implementation and thorough evaluation of the other might prove to be useful in speeding up the computation of velocity controls when using obstacle avoidance. Furthermore, intuition suggests that we could use theory from kinematics of linkages to make inferences regarding velocity limitations. The method mentioned in section 3.5.1 is an example of this. It might be useful to develop more generalized methods in the same light.

It is quite clear from section 4.2.2 that the performance of obstacle avoidance deteriorates when the obstalces lie in regions that are partially occluded. Object segmentation and reconstruction can help obviate these problems. Also, reconstruction of objects can provide additional information that can be exploited by the Grasp Planner.

This project has thus far focussed on obstacle avoidance using SDF Maps. However, self-collision avoidance has not been implemented. The problem of self collision avoidance requires us to determine the closes points on pairs of capsules and then constraining these points. Such a constraint should be applied to every pair of links that can potentially collide.

The ideal future goal is to have an online obstacle avoidance behaviour. In our current setup, the SDF Tracker and obstacle avoidance task run on different processes. This entails that we undertake the costly operation of sending the map as a ROS Message from the SDF Tracker to the obstacle avoidance task. Running the two processes as two threads of the same process will help eliminate the costly data transfer. To have an online obstacle avoidance task, we would also need procedures to filter out depth pixels that correspond to the robot itself.

5.2 Contributions

In this project we set out to achieve a few development goals and research goals. In essence the developmental contributions from this work are as follows:

- The HiQP framework has been used to port existing software for picking operations.

- The HiQP framework has been extended to allow tasks to be loaded as ROS plugins.

Research contributions resulting from this project are:

- We have implemented a local obstacle avoidance behaviour based on the SDF representation.
- We have performed base-line evaluation of the obstacle avoidance behaviour.
- We have suggested a couple of methods to address the difficult problem of reducing constraints related to obstacle avoidance. We have also evaluated one of the suggested methods in this work.

Bibliography

[1] T. Stoyanov, N. Vaskevicius, C. A. Mueller, T. Fromm, R. Krug, V. Tincani, R. Mojtahedzadeh, S. Kunaschk, R. M. Ernits, D. R. Canelhas, *et al.*, "No more heavy lifting: Robotic solutions to the container unloading problem," *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 94–106, 2016.

[2] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[3] R. Krug, T. Stoyanov, V. Tincani, H. Andreasson, R. Mosberger, G. Fantoni, and A. J. Lilienthal, "The next step in robot commissioning: Autonomous picking and palletizing," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 546–553, 2016.

[4] M. Johansson, "Online whole-body control using hierarchical quadratic programming: Implementation and evaluation of the hiqp control framework," 2016.

[5] T. Stoyanov, R. Krug, R. Muthusamy, and V. Kyrki, "Grasp envelopes: Extracting constraints on gripper postures from online reconstructed 3d models," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 885–892, IEEE, 2016.

[6] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS Workshop on Geometry and Beyond*, 2016.

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

[8] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Integration of reactive, torque-based self-collision avoidance into a task hierarchy," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1278–1293, 2012.

[9] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pp. 1211–1216, IEEE, 1991.

[10] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

[11] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 249–254, ACM Press/Addison-Wesley Publishing Co., 2000.

[12] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303–312, ACM, 1996.

[13] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

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

[15] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time collision avoidance with whole body motion control for humanoid robots," in *IROS 2007. IEEE/RSJ International Conference on Intelligent Robots and Systems.*, pp. 2053–2058, IEEE, 2007.

[16] O. Kanoun, *Contribution à la planification de mouvement pour robots humanoïdes*. PhD thesis, Université Paul Sabatier-Toulouse III, 2009.

[17] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," *Proceedings of robotics: science and systems IV*, 2008.

[18] Q.-Z. Ye, "The signed euclidean distance transform and its applications," in *Pattern Recognition, 1988., 9th International Conference on*, pp. 495–499, IEEE, 1988.

[19] D. Ricao Canelhas, "Scene representation, registration and object detection in a truncated signed distance function representation of 3d space," Master's thesis, Örebro University, School of Science and Technology, Örebro University, Sweden, 2012.

[20] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," tech. rep., Cornell University, 2004.