Institute of Parallel and Distributed Systems
University Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Bachelorarbeit

Efficient Sampling of Transition Constraints for Motion Planning under Sliding Contacts

Marie Therese Khoury

Course of Study:	Technische Kybernetik
Examiner:	Prof. Dr. rer. nat. Marc Toussaint
Supervisor:	Dr. Andreas Orthey
Commenced:	January 17, 2020
Completed:	August 12, 2020

Abstract

In contact-based motion planning we consider for humanoid and multiped robots problems like going up a staircase, walking over an uneven surface or climbing a steep hill. Solving such tasks requires finding sequences of fixed and sliding contacts and planning the transition from one contact in the environment to another. However, most existing algorithms do not take sliding contacts into account for navigation problems or consider them only for manipulation scenarios.

We propose an approach to contact-based planning that uses sliding contacts and exploits contact transitions. Such transitions are elementary operations required for whole contact sequences. To model sliding contacts, we develop a sliding contact constraint that permits the robot to slide on an object's surface. To exploit contact transitions, we utilize three constraint modes to enable passage: contact with a start surface, no contact and contact with a goal surface. We develop a sampler that samples these transition modes uniformly. In this thesis we focus on the motion of one robot link's end from an initial contact point toward a designated goal surface while the other end of the robot remains in sliding contact with the initial surface.

Our method is evaluated by testing it on manipulator arms of two, three and seven degrees of freedom with different objects and various sampling-based planning algorithms. From the considered manipulator arm, it would be possible to transfer our concept to more complex robots and scenarios and extend it to a whole sequence of contacts.

Kurzfassung

In der Kontakt-basierten Bewegungsplanung betrachten wir Problemstellungen für humanoide und mehrbeinige Roboter, wie zum Beispiel Treppensteigen, Laufen auf einem unebenen Grund oder Besteigen eines steilen Hügels. Um solche Aufgaben zu lösen, benötigt man Sequenzen von festen oder gleitenden Kontakten und Transitionsbewegungen von einem Kontakt in der Umwelt zu einem anderen. Allerdings berücksichtigen die meisten existierenden Algorithmen gleitende Kontakte nicht für Navigationsprobleme oder erwägen sie nur für Manipulationsszenarien.

Wir stellen eine Herangehensweise zum Kontakt-basierten Planen vor, welche gleitende Kontakte benutzt und Kontakttransitionen verwenden. Solche Transitionen sind elementare Vorgänge, die bei Kontaktsequenzen benötigt werden. Um gleitende Kontakte zu modellieren, entwickeln wir eine Gleitkontakt-Einschränkung, welche es dem Roboter ermöglicht auf einer Objektoberfläche zu gleiten. Um Kontakttransitionen zu verwenden, benutzen wir drei Einschränkungs-Modi um Übergänge zu ermöglichen: Kontakt mit einer Startoberfläche, kein Kontakt und Kontakt mit einer Zieloberfläche. Wir entwickeln einen Sampler, der diese Transitions-Modi gleichmäßig sampelt. In dieser Thesis konzentrieren wir uns auf die Bewegung eines Roboterarm-Endes von einem initialen Kontaktpunkt zu einer spezifizierten Zieloberfläche, während das andere Ende in Gleitkontakt mit der Startoberfläche bleibt.

Unsere Herangehensweise wird evaluiert durch Tests mit Manipulator Armen mit zwei, drei und sieben Freiheitsgraden mit unterschiedlichen Objekten. Wir testen mit verschiedenen Sampling-basierten Planungsalgorithmen. Ausgehend vom betrachteten Manipulator Arm, wäre es möglich unser Konzept auf komplexere Roboter und Szenarien zu übertragen und auf eine ganze Sequenz von Kontakten zu erweitern.

Contents

List of Figures

List of Algorithms

1 Introduction

Robots that act in the real-world require the use of contacts. Contacts are important to locomote by walking, climbing and navigating [Bre06; EKMG08; HBL06] and to manipulate the environment and complete various tasks like exploring the shape of an object by sliding an end-effector over its surface [DET17] or holding an object and readjusting the grasp [CHR19].

Algorithms that enable robots to accomplish those tasks are studied in the field of contact-based motion planning. The objective is to complete a manipulation task or reach some goal configuration collision-free while also including explicitly chosen points of the robot to be in contact with specified points in the environment. We want to explore this concept and focus on the sub-problem of sliding contacts and transitioning between two contact points.

Our idea to solving this planning problem is projection-based constrained planning. We project sampled configurations of the robot onto specified constraints that we develop. The robot's configuration space is modified such that it only contains constraint-satisfying configurations.

While projection-based planning works well for fixed contact planning, there does not yet exist an extension to incorporate transitions between sliding contacts. For sliding contacts we constrain the robot to be in constant contact with the area that it should move on. Transition motions are realized by combining this contact constraint with a sampling method to switch between being in contact and moving freely toward the goal.

1.1 Goal

The contribution of this thesis is a new method for contact-based planning. We develop sliding contact and transition constraints and a sampler for the transition constraint. This thesis focuses on the first contact break, the transition and the following contact creation of a whole contact sequence. We realize a simultaneously active constant contact with the ability to slide on a surface. We test and evaluate our method on manipulator arms of two, three and seven degrees of freedom (DoF) with different objects and various sampling-based planning algorithms.

1.2 Structure

This thesis is structured as follows:

Background This chapter explains the knowledge required to understand this thesis.

Related Work This chapter gives an overview of recent research in contact planning and manipulation tasks with similar approaches.

Sliding Contact and Transition Constraints This chapter explains our approach to planning sliding contacts and transitions using constraints and a sampler.

Evaluation and Results This chapter analyses the applied algorithms on their performance in the different scenarios and presents the results.

Conclusion The last chapter summarizes the results of this thesis and makes suggestions for further work on the presented approach.

2 Related Work

This chapter outlines related works in the field of contact-based motion planning.

One way to solve contact planning problems is using the notion of simplification. Simplifications can be achieved through various ways. We can divide the problem into two sub-problems[TDP+18]. The first part plans a guide path for the root of the robot or a specified scaled version of the root. We can use a reachability condition for projecting the space of robot configurations. Reachable configurations are defined such that the root of the robot is collision-free while the limbs are able to reach the contact surface. In this case the efficiency is increased by sampling configurations in the projected space of lower dimension. The second part takes the root guide path and expands it into a set of consecutive contact configurations. Similarly we can use a guide route for the robot's root link [GAL16]. To ensure a collision-free path, we apply necessary and sufficient conditions that check for collisions with the bounding geometry of the robot's root and bounding geometry for the whole body respectively. To achieve a different simplification, we can approximate the robot's body as boxes encasing the limbs' motion range [DT15] and only explicitly plan the the footstep placements.

Another approach to contact-planning problems utilizes robotic constraints. We can describe contact points through a set of contact constraints [EKM06] by assigning points on the robot to accord with points on predefined support objects. This method consists of two parts. The first part is a tree builder and explorer that creates neighboring reachable contacts to a given set of contacts and the second part is a posture generator that takes these sets of contacts to compute stable, collision-free and contact constraint compliant configurations. Trajectories between the resulting configurations are then planned using classical motion planners. A second way to use constraints is approached by sampling useful contacts first and then sampling a set of sequential configuration that satisfy collision and contact constraints, before computing the transition motions between these configurations [HBL06]. We can extend this planner by using motion primitives that are generated offline and chosen with a set of criteria in order to compute more efficient and natural-looking motions [HBHL06]. For these motion primitives, a transition between given start and goal configurations is sampled randomly and a trajectory then planned using a sampling-based algorithm.

Sliding contacts as opposed to fixed contact points have various application possibilities. We can for example use sliding contacts to explore object shapes [DET17] and address the problem of uncertainty in robotics that stems from an unknown environment. By keeping constant contact with a touch sensor end-effector and moving over a previously unexplored object, data for learning the surface is gathered. Using such sliding contacts yields better results than singular contact points. Another application for sliding contacts are balance-keeping tasks for humanoid robots [SCTK19]. One such task for example has the robot making contact with a wall and its hand and moving it across the wall in constant contact. A different usecase is object manipulation and grasping with a multi-fingered robot hand [SWUL17].

3 Background

This chapter explains the needed background knowledge for this thesis. We explain the basic robotic concept of configuration spaces and motion planning. Then we go into sampling-based motion planning and the algorithms that are utilized later in this work. Finally we explain our notion of contacts and transitions and describe constrained planning.

Large parts of this chapter are based on Part II of [LaV06] and [KMK18].

3.1 Configuration Space and Motion Planning

A configuration q defines the independent variables of a given robot needed to uniquely specify the robot's position relative to a reference frame. Considering a manipulator arm with n number of rigid links connected by n joints, one possible configuration would define n number of variables. For such a robot with n degrees of freedom q is defined as $q \in \mathbb{R}^n$. The configuration space Q of a robot is the set of all such possible configurations q. To avoid collisions with any obstacles or the robot itself, typically a free space $Q_{free} \subseteq Q$ is defined that only contains collision-free configurations.

A basic motion planning problem is defined as finding a continuous path in Q_{free} that connects from a given start configuration $q_{start} \in Q_{free}$ to a goal configuration $q_{goal} \in Q_{free}$. Explicitly computing Q_{free} is a complex problem and the complexity grows with an increase of a robot's degrees of freedom. We use the concept of sampling-based motion planning to avoid this by working with a smaller subset of Q_{free} that is chosen through sampling strategies.

3.2 Sampling-based Motion Planning

Sampling-based motion planning relies on sampling collision-free configurations q_{free} in subsets of Q, instead of computing all possible configurations and paths in-between. These samples are connected to a tree or graph. With a longer planning time and increasing numbers of q_{free}, more space can be mapped out. If the problem is solvable, the probability of finding a path then converges to one. The algorithms provide an efficient solution for finding feasible paths for high-dimensional problems.

3.2.1 Sampling-based Planning Algorithms

In this thesis, we use the following five sampling-based planning algorithms.

The Rapidly Exploring Random Tree-Algorithm (RRT) builds a tree from q_{start} toward q_{goal}. In its basic form it samples random configurations from Q_{free} and in each iteration connects one to the closest configuration already in the tree if the path is collision-free. This is repeated until a branch can be connected to the goal.

Sparse Stable RRT (SST) [LLB14] is an asymptotically near-optimal variant of RRT. During each of N iterations a random configuration is sampled and chosen with a best first strategy. A new tree node is added if the path connecting to it is collision-free and provides better path cost than previous nodes in that region. These inferior previous nodes are then discarded.

The algorithm Search Tree with Resolution Independent Density Estimation (STRIDE) [GMK13] is another tree-based planner that detects less investigated regions of Q. It uses a nearest-neighbor access tree data structure to assess the density of configurations. The detected configurations are then sampled to build a tree toward the goal state.

The Probabilistic Roadmap Method (PRM) is a different type of sampling-based planner [KSLO96]. First a roadmap, a graph consisting of nodes and edges is generated. The nodes correspond to randomly sampled configurations in Q_{free} and the edges are connections between nodes that are determined collision-free by a local planner. When given q_{start} and q_{goal}, the method tries to find two nodes on the roadmap closest to these configurations. Then a path is computed along the roadmap's edges connecting the two initial nodes.

Sparse Roadmap Spanners (SPARS) is another roadmap-based algorithm similar to a variant of PRM [DKB13]. This planner utilizes two graphs, a sparse and a dense graph. The dense graph is constructed with randomly sampled configurations from Q_{free} to be asymptotically optimal. The sparse spanner graph contains only a subset of these configurations, selected by predefined path length and cost maxima.

3.3 Contacts and Transitions

We make use of point contacts in this thesis. A point contact is the junction of a point of the robot with a point on a surface in the environment. For a three-dimensional world $W = \mathbb{R}^3$ a contact can be formulated as follows:

$$\begin{pmatrix} x_{robot} \\ y_{robot} \\ z_{robot} \end{pmatrix} = \begin{pmatrix} x_{surface} \\ y_{surface} \\ z_{surface} \end{pmatrix}$$

Such a junction that is not broken during a continuous motion over a surface S is a sliding contact.

$$\begin{pmatrix} x_{robot} \\ y_{robot} \\ z_{robot} \end{pmatrix} \in (S)$$

A contact transition is the process of breaking a contact, moving freely and creating a contact elsewhere.

$$\begin{pmatrix} x_{robot} \\ y_{robot} \\ z_{robot} \end{pmatrix} = \begin{pmatrix} x_{startSurface} \\ y_{startSurface} \\ z_{startSurface} \end{pmatrix} \xrightarrow{\text{contact-free motion}} \begin{pmatrix} x_{robot} \\ y_{robot} \\ z_{robot} \end{pmatrix} = \begin{pmatrix} x_{goalSurface} \\ y_{goalSurface} \\ z_{goalSurface} \end{pmatrix}$$

3.4 Constrained Planning

Constrained planning can be used when a robot performs tasks that in some way limit its possible motions [KMK18]. When we use a contact-based method with point contacts, sliding contacts or contact transitions, we put such limits on a robot's configuration space Q. In order to express these limits, we formulate a contact as a singular task constraint and a transition as a set of successive task constraints. We extend the motion planning objective of finding collision-free configurations to simultaneously satisfy given constraints. These constraints are specified by a constraint function

$$f(q) : Q \to \mathbb{R}^n$$

that is satisfied when the real-value vector $f(q) = \mathbf{0}$ for a given configuration q. This function is then used to construct an implicit constrained configuration space.

$$X = \{q \in Q \mid f(q) = \mathbf{0}\}$$

It contains all configurations that satisfy the defined constraint. We can now define a configuration space that includes all collision-free configurations from Q_{free} (defined in Section 3.1) that also satisfy the constraints.

$$X_{free} = X \cap Q_{free}$$

A basic constrained motion planning problem is thus defined as finding a continuous path in X_{free} that connects from a given start configuration $q_{start} \in X_{free}$ to a goal configuration $q_{goal} \in X_{free}$.

4 Sliding Contact and Transition Constraints

This chapter explains our approach to contact and transition planning with constraints and presents our sampling method for transition modes.

4.1 Problem Description

The goal of this work is realizing a new method for planning sliding contacts and the transition between two given contact points. We develop a sliding contact constraint and a transition constraint to project the configuration space and a sampler needed for the transition constraint.

The sliding contact constraint keeps an end joint in constant touch with an object's surface. We define the constraint function as the distance between joint and surface.

The transition encompasses a contact break from a given start surface, the contact-free motion toward a goal surface and a contact creation at the goal. The steps of a transition motion is divided into three separate transition modes. We combine the concept of our contact constraint with a mode sampler that samples these three modes uniformly.

For a robot with k contact joints we specify k number of either contact or transition constraints. Only these k designated joints can be in contact, the rest of the body is planned to avoid any collisions. At least one joint is in constant contact with a surface and can slide on it. In Figure 4.1 we can see a graph for a specific robot with $k = 2$ contact joints and two contact surfaces. It depicts the three different modes a robot's contact joint can be in and which modes can be accessed afterward. The first entry of a tuple corresponds to the first joint and the second entry to the second joint.

Mode 0 is a state without contact or constraint, during which the motion between two contacts occurs.

Mode 1 is the state of contact with a specified object surface one.

Mode 2 is the state of contact with a different specified object surface two.

State (0,0) describes a free floating robot without any contacts. From there either joint can make contact with object surface one or two. This mode change is depicted by the edges between the different states. State (1,1) corresponds to a robot with both contacts on the same initial surface one (see the green stance in Figure 4.2) and state (2,2) corresponds to full contact with surface two. A step cycle describes the process of a joint breaking contact with surface one and creating a new contact on surface two and the other joint doing the same afterward. At least one of the two joints is in contact with one of the surfaces during the whole process. A full cycle is completed by changing modes along the depicted edges starting at state (1,1) and ending in state (2,2). From there, the cycle can be repeated to form a sequence of steps and transitions in context of e.g. a walking or climbing scenario.

Highlighted in green is the sub-problem we address in this thesis in context of a full step cycle. Starting from state (1,1), we consider the task of moving along the edges over (1,0) or (0,1) to state (1,2) or state (2,1) respectively.

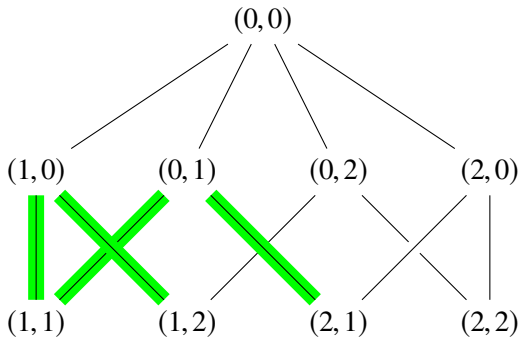

Figure 4.1: Constraint Graph. This graph depicts the possible modes and mode changes of a robot with $k = 2$ contact joints and two contact surfaces. Relevant for this thesis are the parts highlighted in green.

In this thesis we consider simple manipulator arms that can make contact with its environment via two end joints, i.e. $k = 2$. Figure 4.2 shows such a manipulator arm with three DoF [1]. We label the end of the thinner link as the first contact joint and the thicker link end as the second contact joint. We place the sliding contact constraint on the last joint and the transition constraint on the first link.

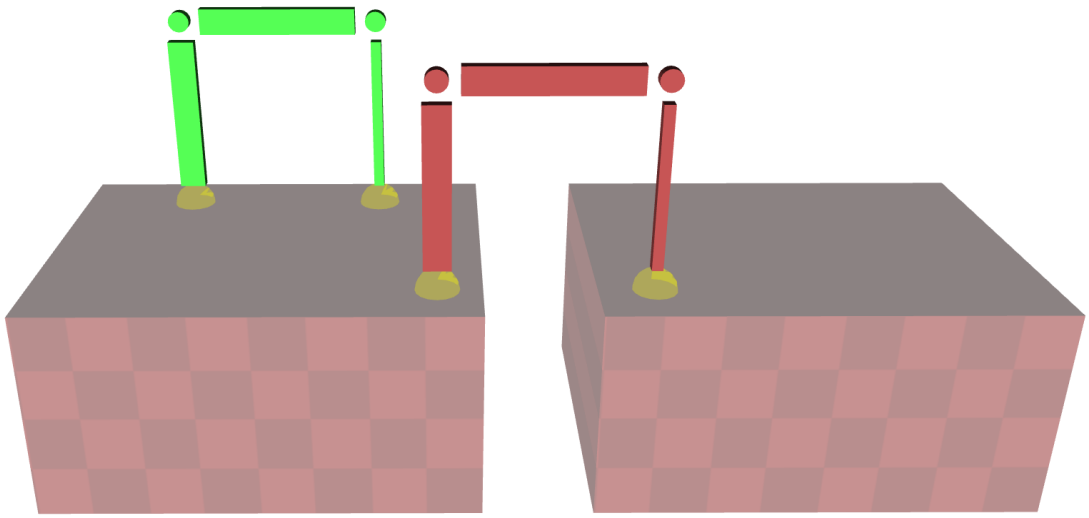

Figure 4.2: Three DoF manipulator arm. q_{start} is depicted in green, q_{goal} in red and the yellow spheres symbolize the end joints that are in contact with the brown cuboids.

[1] Videos of this example and other scenarios can be found here: https://www.youtube.com/playlist?list=PLGFX_osUncaPGvBRV23DbFr3a4ScZMCyK

4.2 Sliding Constraint

First we present the sliding constraint that is put on the robot's contact joint specified to be in constant contact. We compute the distance between this contact joint and the closest point on the given surface at the current configuration q and assign this value to the constraint function $f(q)$.

Algorithm 4.1 shows the pseudo-code for computing sliding constraints. Information on the $robot$, the environment $world$, the desired contact $surface$ and the robot's $joint$ to be in contact is given. First the joint's position in world coordinates is determined. We split the given surface into mesh polygons and save them into a list. The variable $varDistance1$ is a number greater than the distances we compare in the following loop. This list is iterated over to determine the polygon that is closest to the current $jointPosition$. During each iteration we save the coordinate on one such polygon that is closest to $jointPosition$ into $varClosestPoint$ and the distance between these two coordinates into $varDistance2$. We then compare the two distance variables and if $varDistance1$ is greater than $varDistance2$, we assign the value of $varDistance2$ to $varDistance1$. The closest coordinate $varClosestPoint$ corresponding to the smaller distance is then assigned to be the overall $closestPoint$. After this loop, we have determined the coordinates of the surface point closest to the current position of the robot joint. Finally we compute the distance between $jointPosition$ and current $closestPoint$ again and return it .

Algorithm 4.1 Sliding Constraint

```
1:  procedure SlidingConstraint(robot, world, joint, surface)
2:      jointPosition ← getRobotWorldPosition(robot, joint);
3:      surfaceList ← getSurfaceMesh(world, surface);
4:      varDistance1 = 1000;
5:      closestPoint;
6:      for k ← (0, surfaceList.size()) do
7:          varClosestPoint ← closestPosition(surfaceList(k), jointPosition);
8:          varDistance2 ← getDistance(jointPosition, varClosestPoint);
9:          if varDistance1 > varDistance2 then
10:             varDistance1 ← varDistance2;
11:             closestPoint ← varClosestPoint;
12:         end if
13:     end for
14:     return distance ← getDistance(jointPosition, closestPoint);
15: end procedure
```

It is also possible to use this constraint on multiple contact joints, without including a transition. This results in the robot sliding across an object's surface on all its contact joints to reach a given goal configuration. Figure 4.3 and Figure 4.4 show such a sliding scenario for a seven DoF manipulator arm with two contact joints.

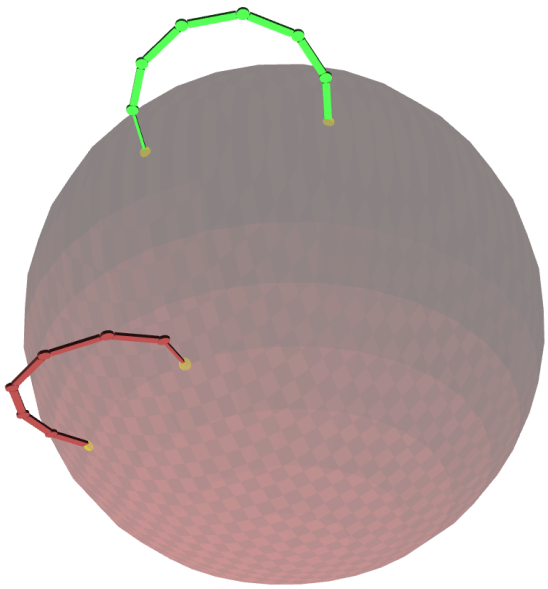

Figure 4.3: Seven DoF manipulator sphere scenario. q_{start} is depicted in green and q_{goal} in red.

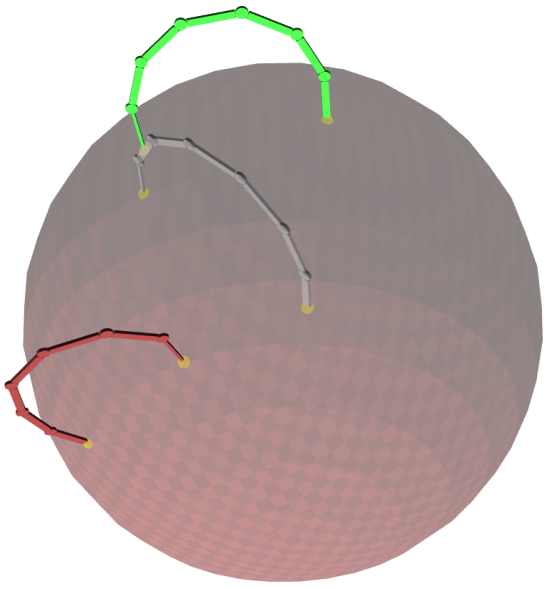

Figure 4.4: Seven DoF manipulator sphere scenario in motion. q_{start} is depicted in green, q_{goal} in red and a intermediate step in gray.

4.3 Transition Constraint

This section presents the implementation of our transition constraint. It combines the contact constraint described in Section 4.2 with a constraint-free state into three transition modes.

Algorithm 4.2 shows the pseudo-code for our transition constraints. It requires information on the $robot$, the environment $world$, the desired contact $startSurface$ and $goalSurface$ and the robot's $joint$ that performs the transition. First the joint's position in world coordinates is determined. We split the given surfaces into mesh polygons and save them into a $startSurfaceList$ and $goalSurfaceList$ respectively. The transition $mode$ of the current iteration is set by our sampling method explained in the next section. We enforce a different constraint depending on the value. In the case of $mode = 0$ the returned value is 0 which means the constraint is satisfied, independent of the robot's position in relation to a surface. If $mode = 1$ then we compute a sliding constraint using the $startSurfaceList$. The last case of $mode = 2$ computes a sliding constraint for $goalSurface$.

4.4 Sampling Method

In order to use this transition constraint for path planning, we implement a method to sample the three different constraint modes. The mode is set before calling Algorithm 4.2 and the transition constraint is enforced accordingly.

In Algorithm 4.3 we can see the pseudo-code for sampling transition modes. The input is a list of k constraints for a robot with k number of contact joints. Each constraint could be either a sliding or a transition constraint. We iterate over this list and check each entry if it is a transition constraint. The variable $sampledMode$ is a number uniformly sampled from (0, 1, 2). For each transition constraint in $constraints$ we then assign the value of the $sampledMode$ to the transition $mode$.

Algorithm 4.2 Transition Constraint

```
1:  procedure TransitionConstraint(robot, world, joint, startSurface, goalSurface)
2:      jointPosition ← getRobotWorldPosition(robot, joint);
3:      startSurfaceList ← getSurfaceMesh(world, startSurface);
4:      goalSurfaceList ← getSurfaceMesh(world, goalSurface);
5:      mode;
6:      if mode = 0 then
7:          return distance = 0;
8:      else if mode = 1 then
9:          varDistance1 = 1000;
10:         closestPoint;
11:         for k ← (0, startSurfaceList.size()) do
12:             varClosestPoint ← closestPosition(startSurfaceList(k), jointPosition);
13:             varDistance2 ← getDistance(jointPosition, varClosestPoint);
14:             if varDistance1 > varDistance2 then
15:                 varDistance1 ← varDistance2;
16:                 closestPoint ← varClosestPoint;
17:             end if
18:         end for
19:         return distance ← getDistance(jointPosition, closestPoint);
20:     else if mode = 2 then
21:         varDistance1 = 1000;
22:         closestPoint;
23:         for k ← (0, goalSurfaceList.size()) do
24:             varClosestPoint ← closestPosition(goalSurfaceList(k), jointPosition);
25:             varDistance2 ← getDistance(jointPosition, varClosestPoint);
26:             if varDistance1 > varDistance2 then
27:                 varDistance1 ← varDistance2;
28:                 closestPoint ← varClosestPoint;
29:             end if
30:         end for
31:         return distance ← getDistance(jointPosition, closestPoint);
32:     end if
33: end procedure
```

Algorithm 4.3 Transition Mode Sampler

```
procedure SampleMode(constraints)
    for k ← (0, constraintsList.size()) do
        if constraints(k) is TransitionConstraint then
            sampledMode ← uniformSampledNumber(0, 2);
            constraints(k).mode ← sampledMode;
        end if
    end for
end procedure
```

5 Evaluation and Results

This chapter describes various planning scenarios and evaluates the algorithms that are used to test the contact planner. We test with a manipulator arm of two, three and seven DoF and different obstacle scenarios in two-dimensional (2D) and three-dimensional (3D) space. The scenarios are run ten times with each planning algorithm and with a specified computation time. We apply and compare the sampling-based planning algorithms SST, RRT, PRM, STRIDE and SPARS. (Explained in Chapter 3.2). We evaluate the performance of our method by comparing the computation times of the different algorithms. The results of our evaluation is visualized in the form of bar charts.

5.1 Programming Setup

The contact planner is implemented and tested utilizing the libraries Open Motion Planning Library [KMK19; MŞK15; ŞMK12] and Kris' Locomotion and Manipulation Planning Toolbox [Hau16]. We used an OpenGL-based graphical user interface to visualize the output. The code for this thesis was written in the programming language C++.

5.2 Test Scenarios

In this section we describe our test scenarios. We test our method on four different scenes using five planning algorithms. We consider a manipulator arm of two DoF in 2D and two rectangular objects, one of three DoF in 3D using two cuboid objects on one and a sphere and a cuboid in a second scene and finally an arm of seven DoF in 2D with three rectangular objects.

5.2.1 Two-DoF Manipulator Arm

This scenario is in 2D, there are two rectangular objects and a manipulator arm with two rigid links, two joints and two ends to make contacts with (Figure 5.1 and Figure 5.2). The robot starts in full contact with the lower rectangle and transitions up along the computed path (depicted in purple). In its goal position the robot keeps one contact with the initial surface and makes one contact with the upper rectangle.

Figure 5.3 shows the average computation time of ten runs of this scenario with a specified maximum planning time of two seconds. We can see that all planning algorithms find a path successfully and quickly in under a second.

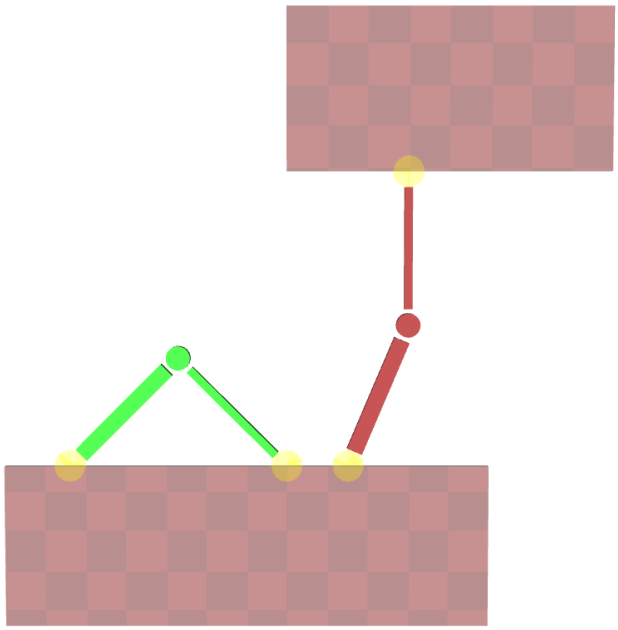

Figure 5.1: Two DoF rectangle scenario. q_{start} is depicted in green and q_{goal} in red.

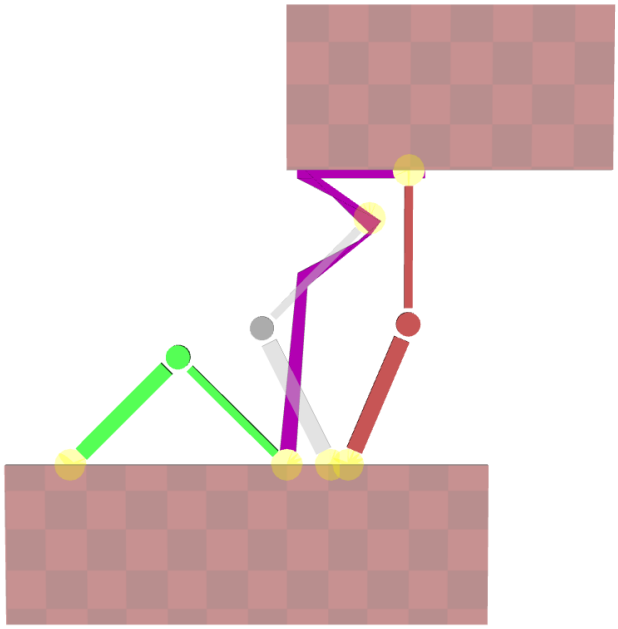

Figure 5.2: Two DoF rectangle scenario. An intermediate step of Figure 5.1 is shown in gray and the path of the first joint in purple.

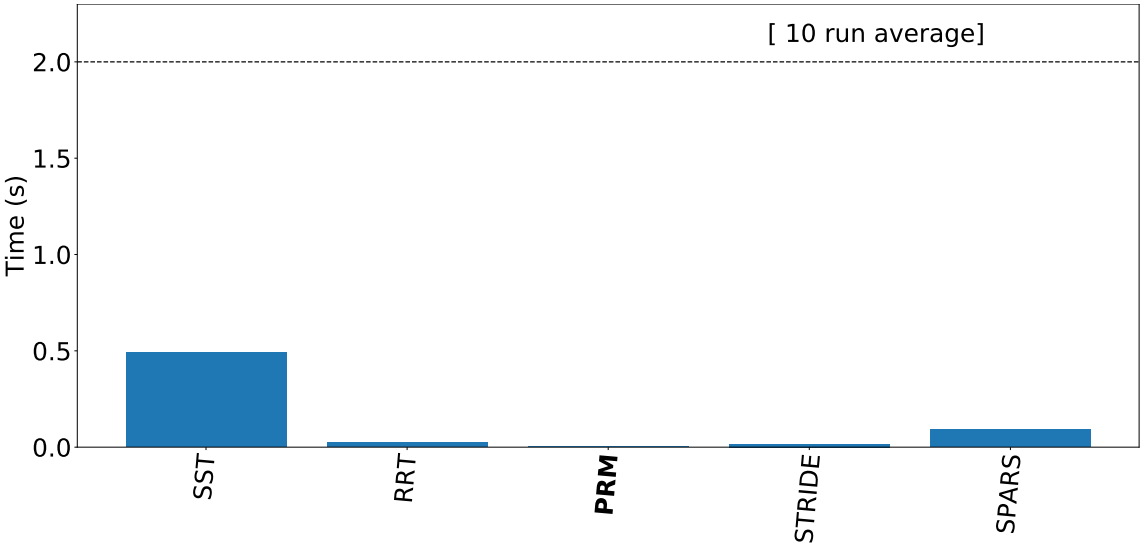

Figure 5.3: Computation Time. Two DoF Scenario. Maximum planning time of two seconds.

5.2.2 Three-DoF Manipulator Arm

We test the three DoF manipulator arm in two different scenarios.

Cuboid Scenario

The first 3D scenario contains two cuboid objects and a manipulator arm with three rigid links, three joints and two ends to make contacts with (Figure 5.4 and Figure 5.5). The robot starts in full contact with the bigger lower cuboid and moves up along the computed path (depicted in purple). To its goal position the robot has slid in contact with the initial surface and made one contact with the upper cuboid after transitioning upward.

Figure 5.6 shows the average computation time of ten runs of this scenario with a specified maximum planning time of two seconds. We can observe that all five planners find a path successfully and in under a second. On average the planning takes more time than for the previous scenario in Section 5.2.1.

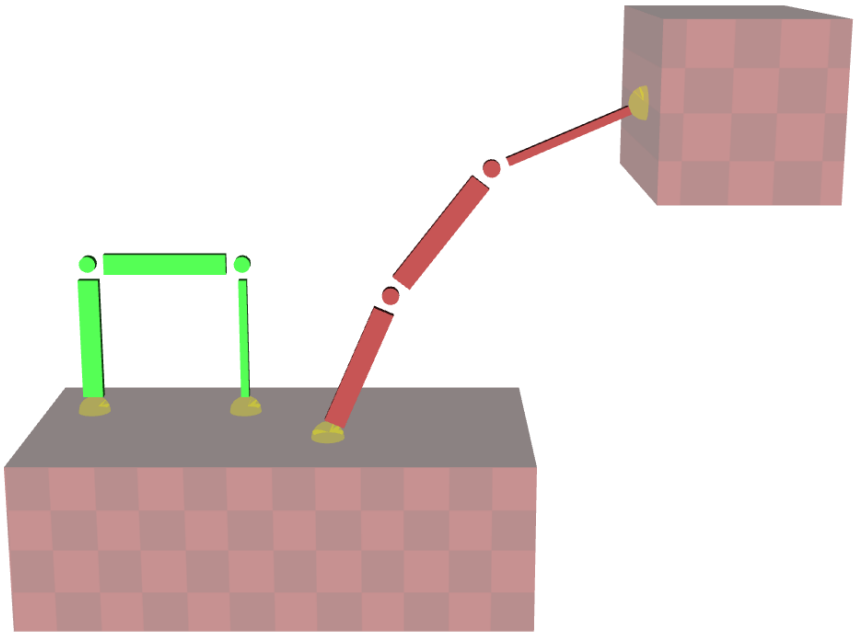

Figure 5.4: Three DoF cuboid scenario. q_{start} is depicted in green and q_{goal} in red.

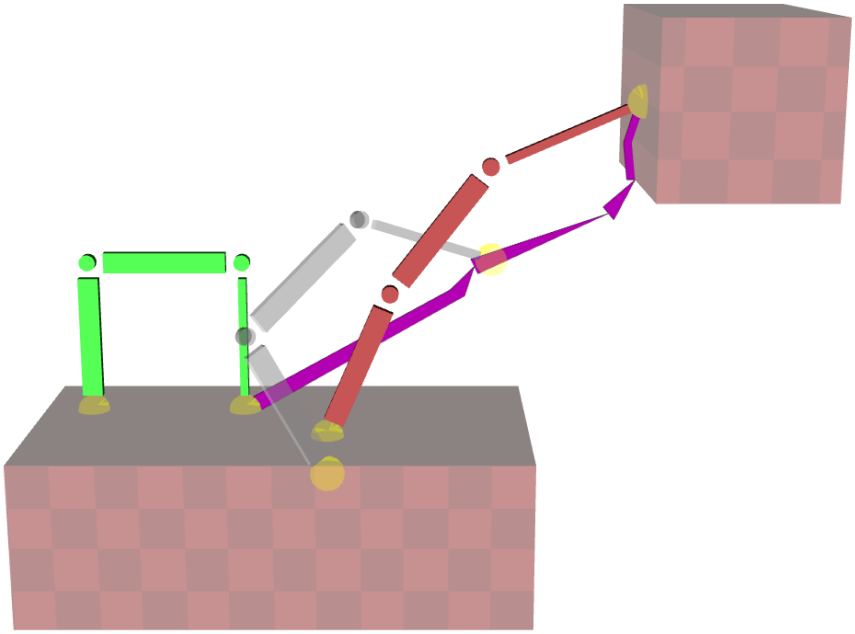

Figure 5.5: Three DoF cuboid scenario. An intermediate step of Figure 5.4 is shown in gray and the path of the first joint in purple.

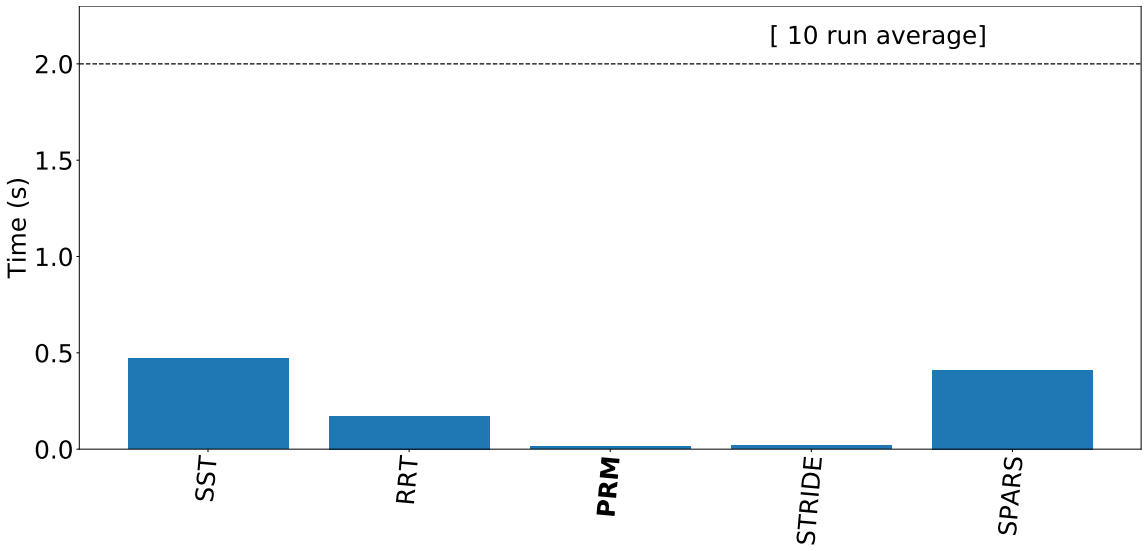

Figure 5.6: Computation Time. Three DoF cuboid scenario. Maximum planning time of two seconds.

Sphere and Cuboid Scenario

The second 3D scenario with the 3 DoF manipulator arm contains a sphere and a cuboid object (Figure 5.7 and Figure 5.8). The robot starts in full contact with the sphere and transitions up along the computed path [1]. At its goal position the robot has one contact with the initial sphere surface and the other contact with the cuboid on the right above.

Figure 5.9 shows the average computation time of ten runs of this scenario with a specified maximum planning time of 30 seconds. We can see that not all planning algorithms find a path in the maximum planning time. The algorithms STRIDE and PRM solve the planning problem the fastest in under 5 seconds and SPARS exceeds the time limit.

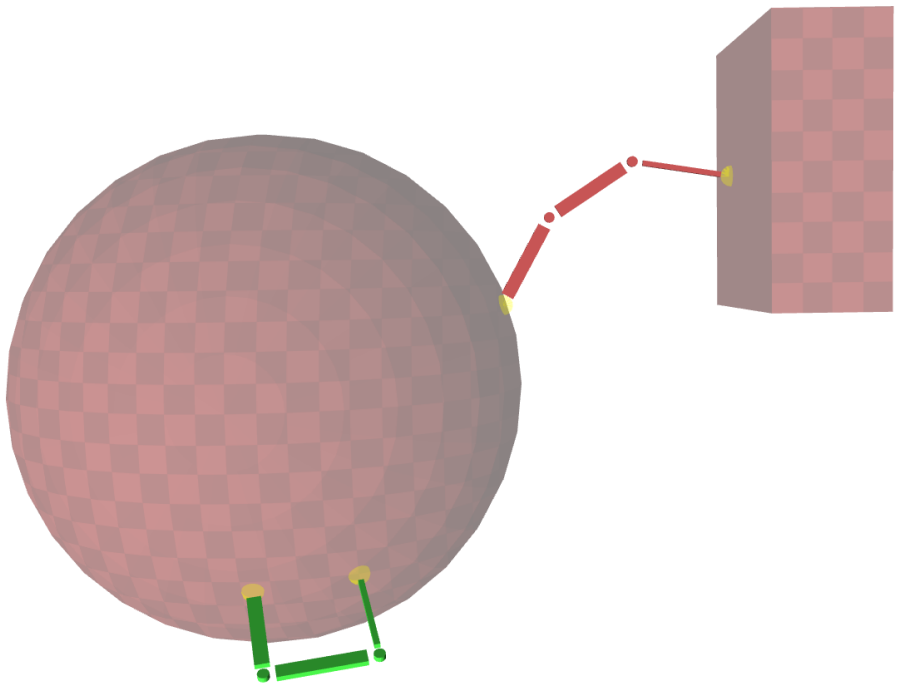

Figure 5.7: Three DoF cuboid scenario. q_{start} is depicted in green and q_{goal} in red.

[1] The sliding part of the path is not visible due to a visualization fault of the sphere

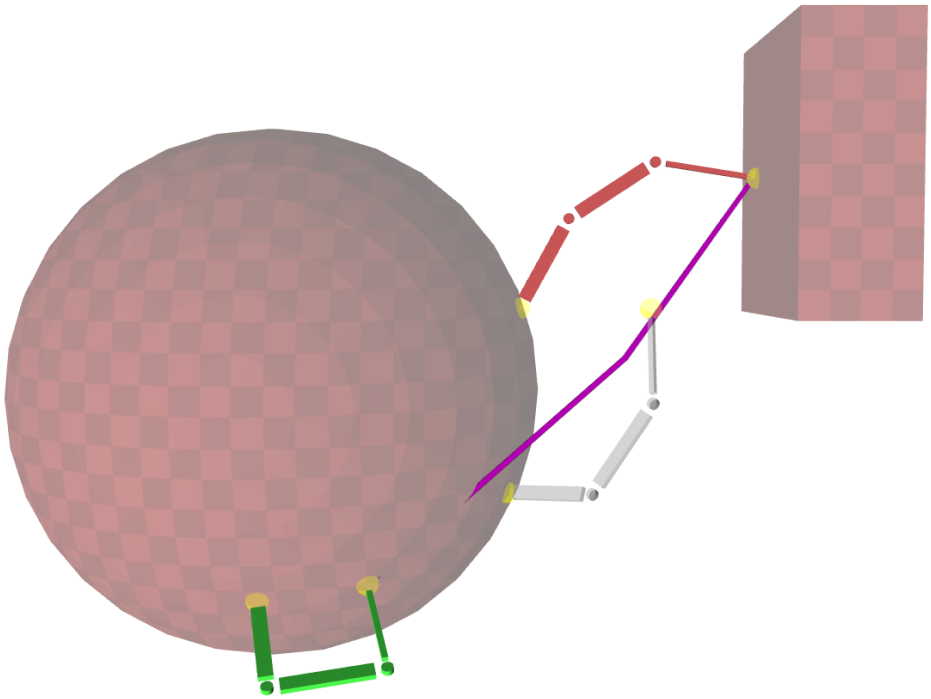

Figure 5.8: Three DoF cuboid scenario. An intermediate step of Figure 5.7 is shown in gray and the path of the first joint in purple.

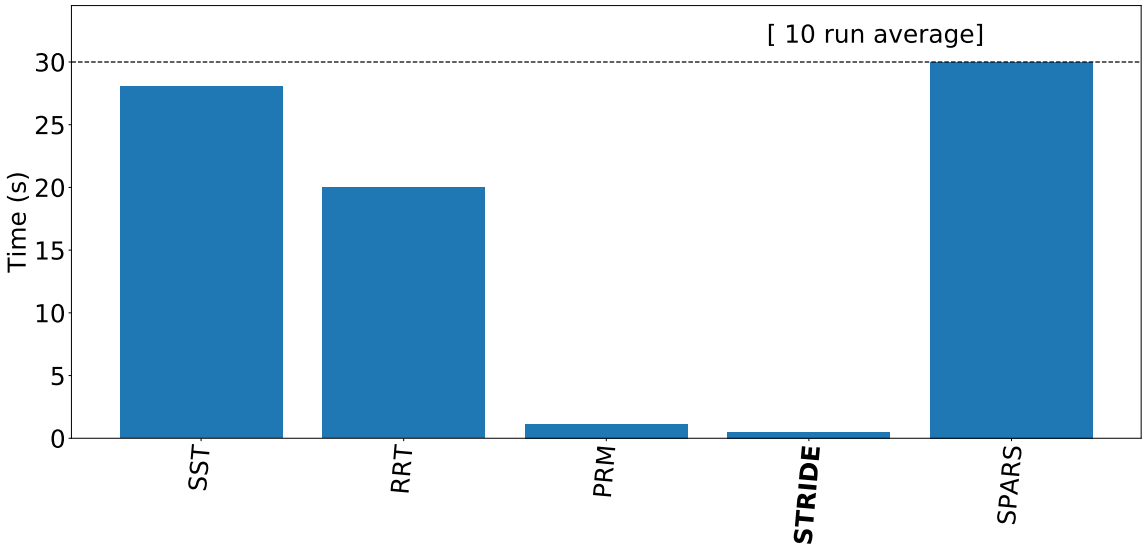

Figure 5.9: Computation Time. Three DoF sphere scenario. Maximum planning time of 30 seconds.

5.2.3 Seven-DoF Manipulator Arm

This scenario is planned in 2D. It contains three rectangular objects and a manipulator arm with seven rigid links, seven joints and two ends to make contacts with (Figure 5.10 and Figure 5.11). The robot starts with one joint in contact with the lower rectangle and the other joint in contact with the left-hand rectangle. It transitions the upper end joint toward the right-hand rectangle. To its goal position the robot has slid the lower joint along the lower rectangle's surface and has made contact with the upper right rectangle.

Figure 5.12 shows the ten run average computation time graph of this scenario with a specified maximum planning time of 30 seconds. We can observe that all planning algorithms find a path successfully in five seconds or less. SPARS and SST require more time in comparison to the other three planners but they still solve the problem quickly.

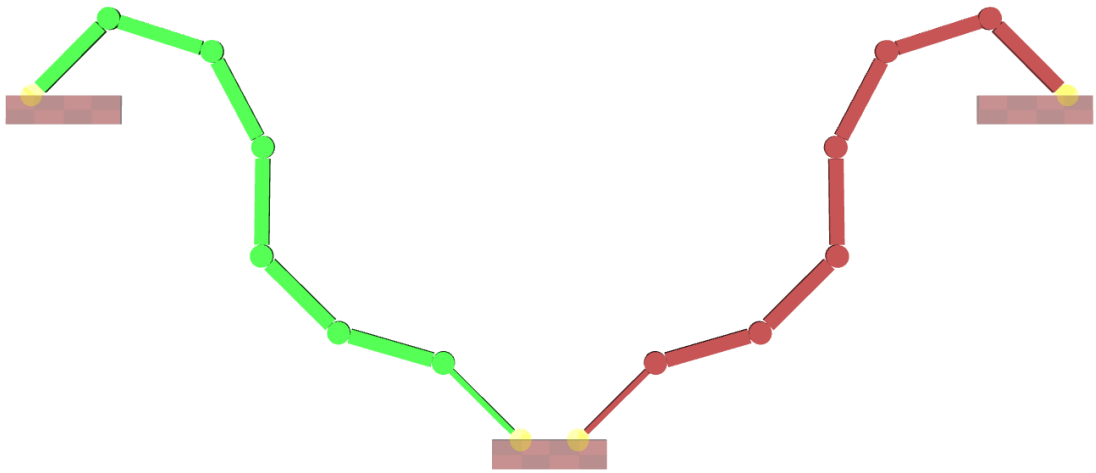

Figure 5.10: Seven DoF manipulator rectangle scenario. q_{start} is depicted in green and q_{goal} in red.

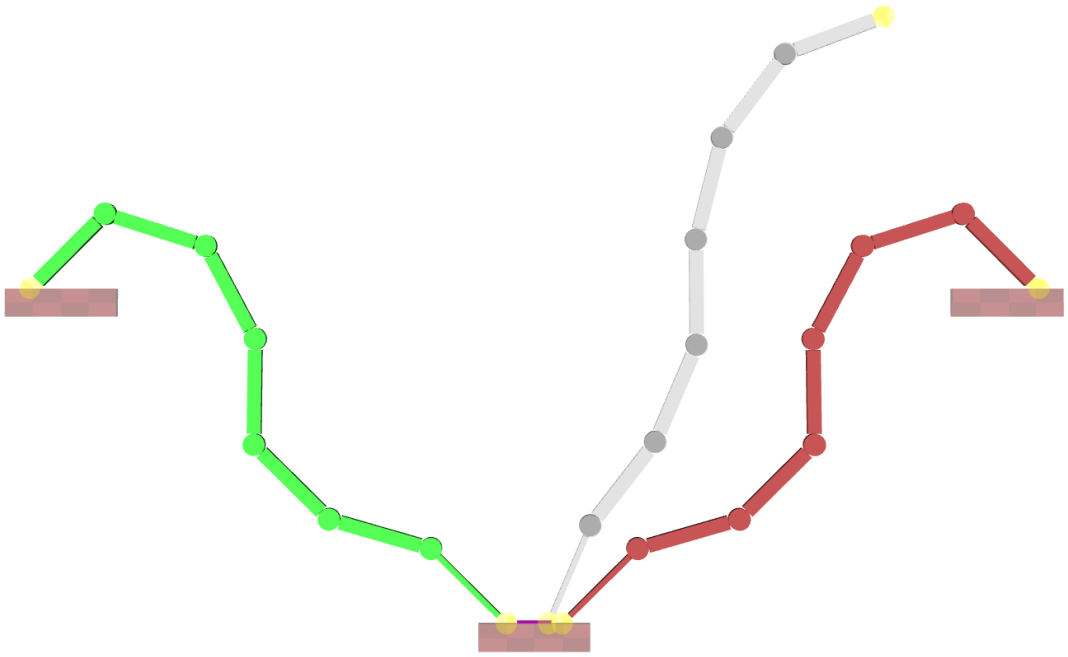

Figure 5.11: Seven DoF manipulator rectangle scenario in motion. An intermediate step of Figure 5.10 is shown in gray and the path of the first joint in purple.

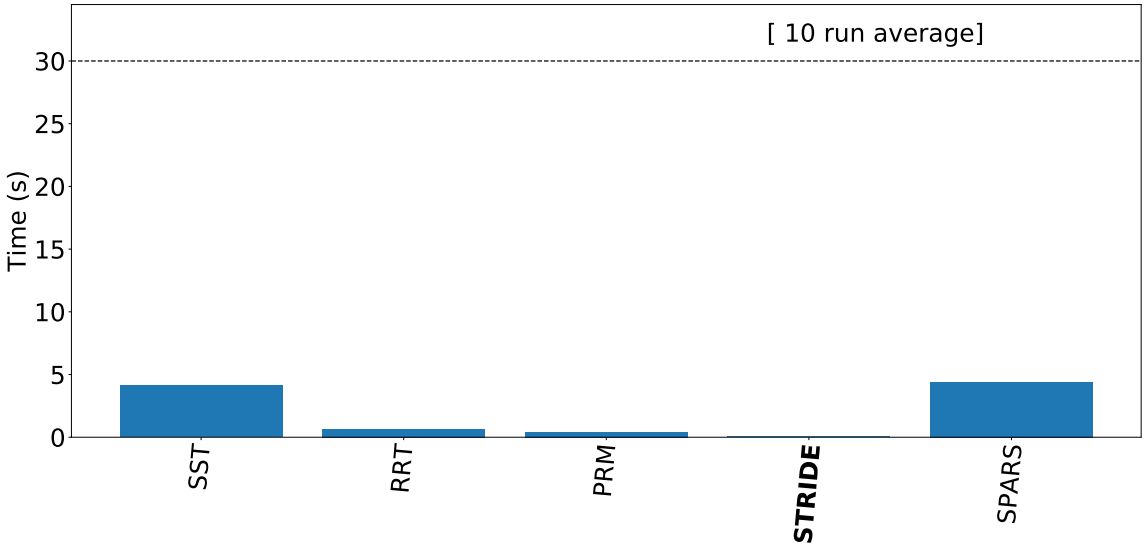

Figure 5.12: Computation Time. Seven DoF Scenario. Maximum planning time of 30 seconds.

5.3 Limitations

During the setup of our testing scenarios we came across a limitation due to the different ways to set the transition. One setup, shown in Figure 5.13, uses the first joint (end of thinner link) to stay in contact with the initial surface while the last joint (end of thicker link) transitions. The other setup is vice-versa, the first joint transitions and the last joint keeps contact (Figure 5.14). Figure 5.15 and Figure 5.16 show the computation time graphs of the aforementioned setups with a maximum planning time of 30 seconds. We can see a significant difference in the planning duration until a path was found. The setup where the first joint is in sliding contact was solved in under two seconds by all five algorithms. In the other setup the path was found after a much longer time by three of the algorithms and the other two exceeded the maximum planning time.

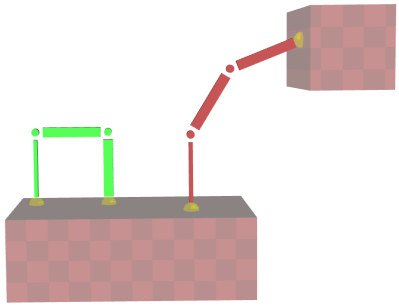

Figure 5.13: Three DoF with first joint in sliding contact. q_{start} is depicted in green and q_{goal} in red.

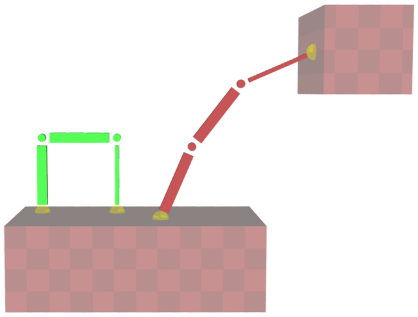

Figure 5.14: Three DoF with last joint in sliding contact. q_{start} is depicted in green and q_{goal} in red.

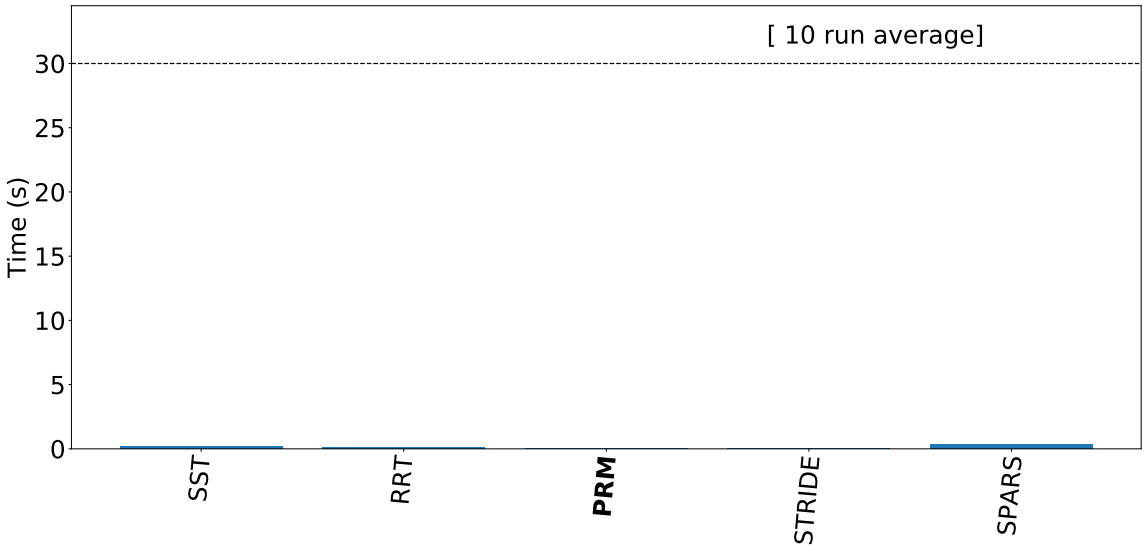

Figure 5.15: Computation times for the first joint in sliding contact. Maximum planning time of 30 seconds.

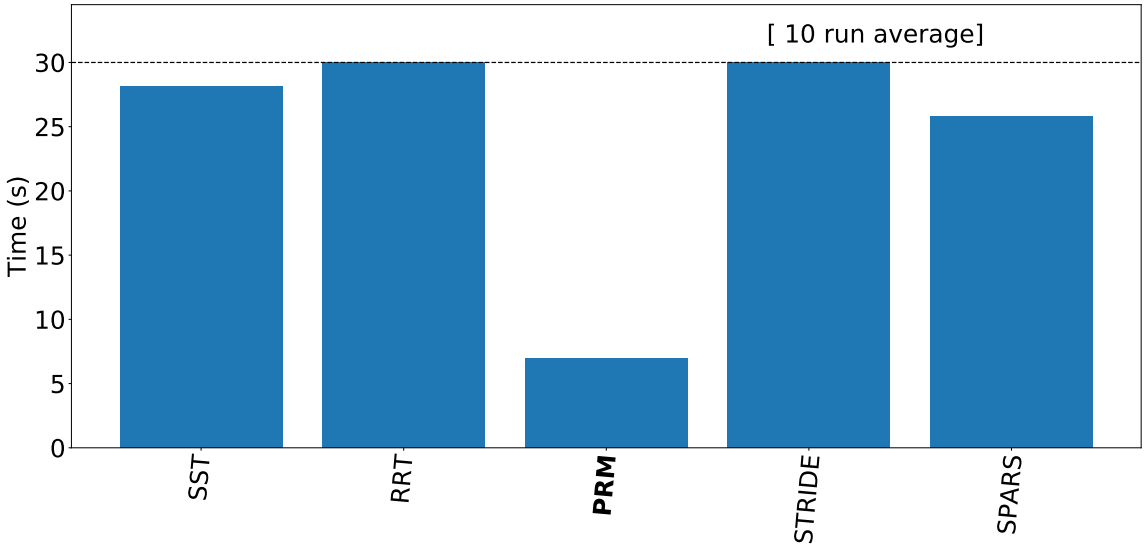

Figure 5.16: Computation times for the last joint in sliding contact. Maximum planning time of 30 seconds.

Due to this effect a fully planned scene for the seven DoF manipulator from Section 5.2.3 can only be shown with the first joint in sliding contact. When we use the first joint in transition, a complete path can not be planned in feasible maximum planning time. Figure 5.17 shows the result of planning the transition for a computation time of ten minutes. We can see that even after this extensive run time, the path in purple is not connected to the goal state.

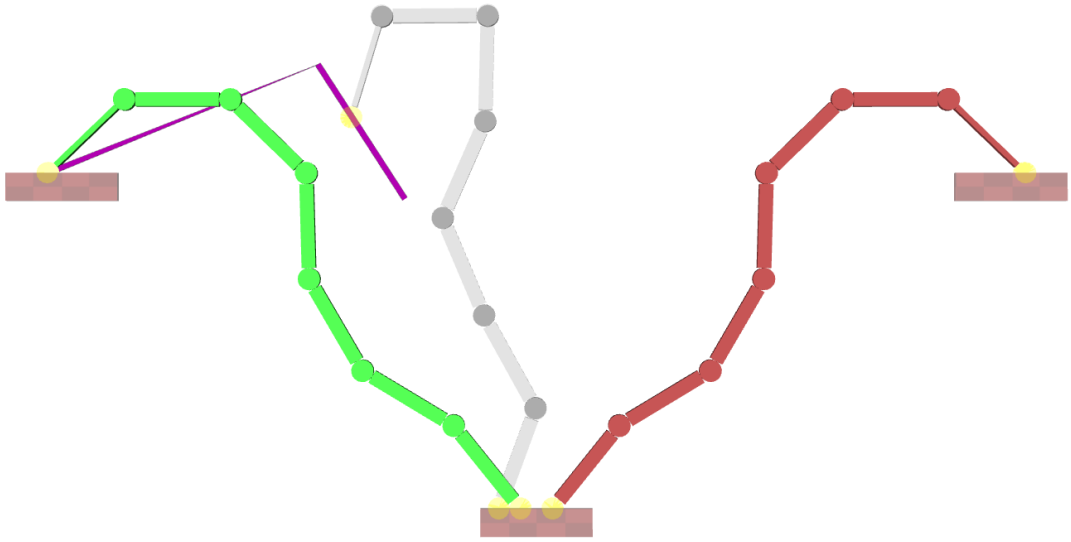

Figure 5.17: Seven DoF manipulator with the last joint in sliding contact. q_{start} is depicted in green and q_{goal} in red, an intermediate step is shown in gray and the path of the first joint in purple.

5.4 Results

As a result of our tests, we can see that our proposed method using constraints works for the considered manipulator arms with two contact joints. The applied sampling-based planning algorithms were able to find transitioning paths between the given start and goal configurations while keeping the designated joint in sliding contact with the initial surface.

We can see that the required computation time differs depending on the setup. We considered three kinds of objects in our scenarios: rectangles in the 2D scenarios, and cuboids and a sphere in the 3D scenarios. The sphere in the 3D scene leads to a longer computation time than the cuboid and rectangular objects. This could be attributed to a much higher number of surface polygons that has to be iterated over during the planning process. We also observe that planning in 2D took less time than in 3D. A 3D scenario leads to more potential contact surfaces and thus also results in more polygons.

The observed limitations may also stem from the way the configurations are projected. We can plan the path for any part of the robot, which we call the root, by projecting onto it. In this thesis we only consider setups where the root position is put on the robot's first contact joint. When we place the root on the robot's end joint that underlies a transition constraint, the free space X_{free} contains more configurations compared to when the joint is constrained to be in contact with a single object's surface. Thus the probability of sampling a feasible sequence of configuration from X_{free} is lower. Additionally, X_{free} also grows with the distance of q_{start} and q_{goal} like e.g. the scenario in Figure 5.17.

6 Conclusion

In this thesis we contribute a new approach to planning sliding contacts using constrained planning and develop a sampler method for transition motions. We formulate two constraints for a robot's contact joints: a sliding contact constraint and a transition constraint. For the transition constraint we split the problem into three constraint modes and implement a sampling method that samples them uniformly. We address the planning of a single contact joint breaking contact, moving the robot toward a goal and creating a new contact there while the other joints remain in sliding contact.

Our tests and evaluations using sampling-based planning algorithms show that the presented concept works on manipulator arms in 2D and 3D scenarios albeit with some limitations depending on the scene setup. We observe that the planning time depends on the placement of the root on one of the robot's body parts. The path is planned for that body part by projecting configurations onto the root. In our tests we only consider setups where the root position is put on the robot's first contact joint. This positioning is critical so it is an important aspect to study further in order to improve planning.

Despite these limitations we can successfully combine sliding contacts with contact transitions. Using this approach we are able to solve the addressed planning problem for a robot with two contact joints.

Bibliography

[Bre06] T. Bretl. “Motion Planning of Multi-Limbed Robots Subject to Equilibrium Constraints: The Free-Climbing Robot Problem”. In: *I. J. Robotic Res.* 25 (Apr. 2006), pp. 317–342. DOI: 10.1177/0278364906063979 (cit. on p. 13).

[CHR19] N. Chavan-Dafle, R. Holladay, A. Rodriguez. “Planar in-hand manipulation via motion cones”. In: *The International Journal of Robotics Research* (Oct. 2019), p. 027836491988025. DOI: 10.1177/0278364919880257 (cit. on p. 13).

[DET17] D. Driess, P. Englert, M. Toussaint. “Active learning with query paths for tactile object shape exploration”. In: 2017. DOI: 10.1109/IROS.2017.8202139 (cit. on pp. 13, 15).

[DKB13] A. Dobson, A. Krontiris, K. Bekris. “Sparse Roadmap Spanners”. In: Jan. 2013, pp. 279–296. DOI: 10.1007/978-3-642-36279-8_17 (cit. on p. 18).

[DT15] R. Deits, R. Tedrake. “Footstep planning on uneven terrain with mixed-integer convex optimization”. In: 2015 (2015). DOI: 10.1109/HUMANOIDS.2014.7041373 (cit. on p. 15).

[EKM06] A. Escande, A. Kheddar, S. Miossec. “Planning support contact-points for humanoid robots and experiments on HRP-2”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006, pp. 2974–2979 (cit. on p. 15).

[EKMG08] A. Escande, A. Kheddar, S. Miossec, S. Garsault. “Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2”. In: July 2008, pp. 293–302. DOI: 10.1007/978-3-642-00196-3_35 (cit. on p. 13).

[GAL16] M. Grey, A. Ames, C. Liu. “Footstep and Motion Planning in Semi-unstructured Environments Using Possibility Graphs”. In: (Oct. 2016) (cit. on p. 15).

[GMK13] B. Gipson, M. Moll, L. Kavraki. “Resolution Independent Density Estimation for motion planning in high-dimensional spaces”. In: May 2013, pp. 2437–2443. ISBN: 978-1-4673-5641-1. DOI: 10.1109/ICRA.2013.6630908 (cit. on p. 18).

[Hau16] K. Hauser. “Robust Contact Generation for Robot Simulation with Unstructured Meshes”. In: Apr. 2016, pp. 357–373. ISBN: 978-3-319-28870-3. DOI: 10.1007/978-3-319-28872-7_21 (cit. on p. 27).

[HBHL06] K. Hauser, T. Bretl, K. Harada, J.-C. Latombe. “Using Motion Primitives in Probabilistic Sample-Based Planning for Humanoid Robots”. In: vol. 47. 2006. DOI: 10.1007/978-3-540-68405-3_32 (cit. on p. 15).

[HBL06] K. Hauser, T. Bretl, J.-C. Latombe. “Non-gaited humanoid locomotion planning”. In: 2006. ISBN: 0-7803-9320-1. DOI: 10.1109/ICHR.2005.1573537 (cit. on pp. 13, 15).

[KMK18] Z. Kingston, M. Moll, L. E. Kavraki. “Sampling-Based Methods for Motion Planning with Constraints”. In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018). DOI: 10.1146/annurev-control-060117-105226 (cit. on pp. 17, 19).

[KMK19] Z. Kingston, M. Moll, L. Kavraki. “Exploring implicit spaces for constrained sampling-based planning”. In: *The International Journal of Robotics Research* (2019), p. 027836491986853. DOI: 10.1177/0278364919868530 (cit. on p. 27).

[KSLO96] L. Kavraki, P. Svestka, J. Latombe, M. Overmars. “Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces”. In: *Robotics and Automation, IEEE Transactions on* 12 (Sept. 1996), pp. 566–580. DOI: 10.1109/70.508439 (cit. on p. 18).

[LaV06] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877 (cit. on p. 17).

[LLB14] Y. Li, Z. Littlefield, K. Bekris. “Asymptotically Optimal Sampling-based Kinodynamic Planning”. In: *The International Journal of Robotics Research* (July 2014). DOI: 10.1177/0278364915614386 (cit. on p. 18).

[MŞK15] M. Moll, I. A. Şucan, L. E. Kavraki. “Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization”. In: *IEEE Robotics & Automation Magazine* 22.3 (Sept. 2015), pp. 96–102. DOI: 10.1109/MRA.2015.2448276 (cit. on p. 27).

[SCTK19] S. Samadi, S. Caron, A. Tanguy, A. Kheddar. “Balance of Humanoid robot in Multi-contact and Sliding Scenarios”. In: Sept. 2019 (cit. on p. 15).

[ŞMK12] I. A. Şucan, M. Moll, L. E. Kavraki. “The Open Motion Planning Library”. In: *IEEE Robotics & Automation Magazine* 19.4 (Dec. 2012). https://ompl.kavrakilab.org, pp. 72–82. DOI: 10.1109/MRA.2012.2205651 (cit. on p. 27).

[SWUL17] J. Shi, J. Z. Woodruff, P. B. Umbanhowar, K. M. Lynch. “Dynamic In-Hand Sliding Manipulation”. In: *IEEE Transactions on Robotics* 33.4 (2017), pp. 778–795 (cit. on p. 15).

[TDP+18] S. Tonneau, A. Del Prete, J. Pettre, C. Park, D. Manocha, N. Mansard. “An Efficient Acyclic Contact Planner for Multiped Robots”. In: *IEEE Transactions on Robotics* PP (2018). DOI: 10.1109/TRO.2018.2819658 (cit. on p. 15).

All links were last followed on August 10, 2020.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature