MANAGING TEMPORAL ROBOT CONSTRAINTS USING REACHABLE VOLUMES

An Undergraduate Research Scholars Thesis

by

EVERETT SIYAN YANG

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Nancy M. Amato

May 2019

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Managing Temporal Robot Constraints using Reachable Volumes

Everett Siyan Yang
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Nancy M. Amato
Department of Computer Science and Engineering
Texas A&M University

This project focuses on planning the motion for high degree of freedom manipulator robots under dynamic (or temporal) constraints. Manipulator robots are widely used in industry and are important because they can do jobs that are either too tedious or too dangerous for humans. An example would be picking up toxic waste or exploring underwater archeological sites.

Motion planning for high degree of freedom (DOF) manipulators under task constraints is challenging because it gives rise to high dimensional configuration spaces (C_{space}) that are complex in structure. Our approach reduces the complexity by re-parameterizing the manipulator robots DOFs into a space that contains the valid regions that the end effector of the robot can reach, known as the Reachable Volume space (RV_{space}). In this way, we can sample valid configurations in C_{space} in linear time with the number of DOFs of the manipulator.

Current Reachable Volume theory only handles permanent constraints and cannot adapt to scenarios that require constraints that are enabled at certain times in the problem and dis-

abled at other times. For example, when a manipulator grabs an object, closure constraints on the grasper must be satisfied, but when the object is to be dropped, these constraints must be ignored. Additionally, certain scenarios require the cooperation of multiple robots. This is obvious if we consider problems that involve objects that are too large for a single robot to handle.

In this work, we produce a working computational framework for efficient motion planning of high degree of freedom manipulator arms under dynamic constraints through the extension of existing work in Reachable Volume spaces.

ACKNOWLEDGMENTS

This project would not be possible without the support of my advisor Dr. Nancy M. Amato and that of the researchers at Parasol Laboratory. In particular, Read Sandstrom, Diane Uwacu, and Shawna Thomas.

LIST OF FIGURES

1. INTRODUCTION

A robot is very often required to traverse an environment while satisfying different task constraint(s). Imagine an underwater robot that needs to open a barrel in order to uncover an artifact. Such a robot would need to constrain its graspers to be in a closed position so it can lift objects. This would be an example of a task constraint. When the robot is ready to drop the object, the constraint on its graspers must be released.

In order to have these robots do such high-level tasks autonomously, we use techniques in motion planning, which is the problem of guiding a robot through some environment while satisfying any constraints placed on it. However, motion planning is a computationally hard problem. Exact approaches are infeasible, and randomized approaches do not provide a general solution. From the latter, there is also the question of the precision of solutions obtained.

Recently, there have been several new developments that relieve some inefficiencies associated with constrained planning [1], [2], [3]. Notably, the reparameterization approach has proven to be effective in creating valid configurations for a high dimensional robot. In fact the approach is able to create such configurations in linear time with the number of degrees of freedom of a robot by constructing a space of all reachable points from a robots base, called the Reachable Volume space.

It is on this development that our current project rests. In particular, our project focuses on using such a method to handle cases where the constraints are not permanent. In these time sensitive cases, the Reachable Volume space must adapt to the changing constraints as the problem progresses. We will show methods to efficiently handle such changes.

We test our algorithms on a 14-DOF manipulator robot navigating through an empty environment. We show that the algorithm is able to guide such complex robots through the environment while continuously satisfying any constraints on its end effector. An immediate application of the algorithm is to the drawing problem, where the robots end effector must follow some curve in space.

2. PRELIMINARIES AND RELATED WORK

This section reviews existing work in the field and introduces concepts crucial to the understanding and solving of the motion planning problem.

2.1 The Motion Planning Problem

Navigating a robot from a point A in space to a point B is called the *robot motion planning problem*. A robot in space is an n-dimensional object, where n is equal to the number of degrees of freedom it exhibits. When we place a robot in the environment, we set its degrees of freedom, thus forming a robot *configuration*. The space of all possible configurations is aptly named the *configuration space*, or C_{space} [4].

The notion of configuration spaces allow us to formulate the motion planning problem as a problem of finding a continuous curve in C_{free} (the set of all feasible configurations of C_{space}) which connects a robot start and goal configuration.

2.2 Sampling Based Planning

Since configuration space can be quite complex, methods that give an exact solution are generally not feasible [5]. Instead, we sample probabilistically on the C_{space}. In this way we can achieve a polynomial time solution to the motion planning problem. A commonly used algorithm is the Probabilistic RoadMap (PRM) method [6], which samples configurations in C_{free}) and connects them using a local planner, which interpolates on each edges, generating feasible intermediate robot configurations. Once a connected roadmap is formed in the workspace (the space that the physical robot lives in) we use a graph search algorithm (e.g., Dijkstra's, A*) to find the shortest path between the start and goal nodes. Once this is done, the problem is solved.

There are many ways to sample robot configurations with some being more effective for a specific environment than other methods. For example, in an environment cluttered with obstacles we may want to use Obstacle-based sampling [7], which randomly samples configurations on the surface of workspace obstacles, yielding much more connective roadmaps, in these cluttered environments, than the classic uniform sampling method.

2.3 Reachable Volumes for Constrained Planning

Although sampling based methods are effective for sampling in unconstrained spaces, they fail when constraints are introduced . To understand this intuitively, we can see that the configuration space (sample space) is large compared to the sub-manifold containing constraint satisfying samples (the configurations we are interested in). Traditional sampling is so hard that the probability of generating a valid sample approaches zero as sampling continues [8].

To this end, the notion of a Reachable Volume space has been developed for efficient sampling of high-DOF manipulator configurations. Sampling valid nodes in RV space takes only polynomial time [3].

Instead of sampling directly on the constraint sub-manifold, we take into consideration all the points our robot (or a sub-section of our robot) can reach from a given base position. In this space, we can directly sample valid configurations without having to do low probability rejection sampling as with traditional sampling. For manipulators, this is done using the sampling algorithm detailed in [3].

Computing the RV-space for a given manipulator is quite efficient with the concept of Minowski sums. Figure 2.1 shows the reachable volume (RV) of a typical manipulator arm. Notice that the large reachable volume (dark green spherical shell) has smaller reachable volumes contained within it. This is because the RV of the robot is constructed from RVs of its linkages, which have smaller outer radii.

We convert a reachable volume sample into a sample of C_{free} with the use of concepts

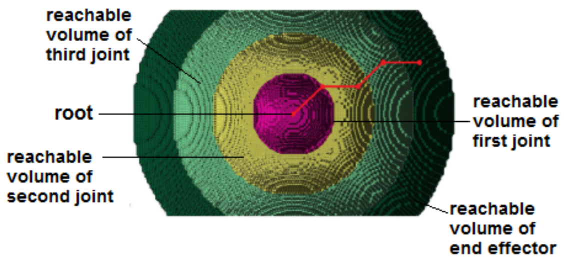

reachable volume of third joint
root
reachable volume of second joint
reachable volume of first joint
reachable volume of end effector

Figure 2.1: Reachable Volume of a simple manipulator arm

in trigonometry, as two robot linkages of known length form a triangle of angle θ which can be computed simply with the law of cosines. For unconstrained RVs, the position of the corresponding C_{free} configuration sample is assigned at random. If the RV is constrained to be at some point(s), it must use those points in the conversion.

Thus, reachable volume space acts as an interface to the configuration space, in the meantime speeding up the sampling process substantially.

3. APPROACH

We use sampling based planning to construct a roadmap as described in section 2.2. To construct constraint satisfying samples we use Reachable Volumes described in section 2.3.

Although the method used to keep track of what set of constraints is "on" at which time is invariant, the application of these constraints varies based on the type of constraints and robots involved. For a simple manipulator robot that is only constrained to have its end effector at some point, we simply construct the Reachable Volume around the point and sample valid configuration from that RV. This section will describe how we manage temporal constraints and how the Reachable Volume is constructed and reconstructed throughout planning.

3.1 Constraint Management

In order to efficiently manage constraint that can be either dynamic or static, we introduce new data structures to the pre-existing Reachable Volume framework.

3.1.1 Constraint Matrix

We define the *constraint matrix* to be an array of constraint sets. Each array of constraints represents a different phase in the problem. For example, when a robot is trying to retrieve an object from a closed barrel, the phase 1 constraint set consist of having the robots end effector be on the barrel lid and its body positioned next to the barrel, $\{C_1, C_2\}$. Phase 2 could then contain n constraints that deal with the robot grabbing the object from inside the barrel, so the set $S = \{C_1, C_2, C_3, ..., C_n\}$. The 2D array contains all of these sets of constraints for the problem. The structure will be used in the planning phase by the RV-sampler.

3.1.2 Temporal Roadmap

Once the environment contains enough samples that satisfy each of the constraints in the set, we are now tasked with connecting them so that a valid path from the starting node to the ending node can be found. Connection should be done in a way that allows for the problem to be solved correctly. In all cases, we will need to establish a notion of time to prevent the robot from forming an ill ordered path (e.g., moving to write something on a whiteboard before picking up a marker, etc.). To do so, we implement what is called a *temporal roadmap*. This is a roadmap that contains configurations that each store a value containing information about their position in time. This is necessary as the notion of time is considered when we are connecting nodes under different constraints. The mapping is done using a standard hashmap with $O(1)$ access time.

3.2 Algorithm Description

This section details the planning algorithm, which involves the structures introduced in section . We will discuss each of the essential functions and how they work together to form a plan for the robot.

The high level structure of the planning strategy described in Algorithm 1 is similar to the algorithm in [6]. In other words, we adopt the PRM algorithm and make changes to it's components in order to handle temporal constraints. Thus our approach is easily extended to other frameworks, such as a tree-based planning algorithms (e.g., RRTs [9]).

The changes made to handle temporal constraint lie in Algorithm 2 and 3, detailed in section 3.2.2 and 3.2.3, respectively.

3.2.1 Temporal Reachable Volumes

As mentioned above, Algorithm 1 adapts from the classic PRM algorithm. Unlike the PRM algorithm, we do preprocessing on a user inputted constraint matrix. This is done so the sampler can assign timesteps to the valid nodes generated.

Algorithm 1 Temporal Reachable Volume Strategy

```
Input:  $S$: temporal constraint set
 1: $env$: environment the robot lives in
 2: $robot$: model of the manipulator robot to be planned
 3: $start$: $robot$ start configuration
 4: $goal$: $robot$ goal configuration
Output:  $g$: constraint satisfying Temporal Roadmap for $robot$ in $env$
 5: function StartPlanning
 6:     ParseConstraintMatrix($S$)
 7:     while !ValidPathExistsBetween($start$, $goal$) do
 8:         TemporalSample($robot$, $S$)
 9:         TemporalConnect()
10:     end while
11: end function
```

3.2.2 Sampling in the Constraint Matrix

To ensure that no constraint set in the constraint matrix is starved of samples, we take the straightforward approach of uniform sampling in the constraint matrix. Formally, $\Pr(A_i) = \frac{1}{|S|}$, where A_i is the event that the Reachable Volume sampler produces a sample that satisfies $C_i \in S$.

Algorithm 2 makes clear the way we choose samples from the parsed list of constraints from the constraint matrix.

3.2.3 Connecting Samples in the Temporal Roadmap

Connecting is done after every sampling iteration. Each existing node in the graph is connected with it's nearest neighbors, with the added condition that two nodes must have consecutive timesteps. The importance of this is mentioned in section 3.1.2.

Algorithm 2 TemporalSample

```
Input:  $robot$: model of the manipulator robot to be planned
 1: $S$: temporal constraint set
 2: $rvSampler$: Reachable Volume sampler
Output:  Valid $robot$ samples added to $g$
 3: function TEMPORALSAMPLE
 4:     randNum = generateRandomNumber() % $|S|$
 5:     currentConstraintSet = S.get(randNum)
 6:     listOfValidSamples = $rvSampler$.sample(currentConstraintSet)
 7:     $g$.add(listOfValidSamples)
 8: end function
```

3.3 Simple Manipulator with End Effector Constraint

We implement our approach for the simple case of a manipulator that only requires its end effector to be constrained at some point. The constraint matrix would be reduced to a one dimensional list because each element in the matrix would only contain one element (the point that the end effector should be positioned at). That is $|C_i| = 1 \forall i \in 1, 2, ..., n$.

Next, we simply set the end effector point of the reachable volume constructed for the robot to be at this point, thus the reachable volume is simply being translated through C_{space} as constraints change. We demonstrate that, given information about the constraints, we can use this method to grasp objects and move them through complicated environments.

3.4 Closed Chain Constraint

Another constraint that we consider, in theory, is the closed chain constraint, where a grasper-like robot will need to configure its degrees of freedom to form a closed loop.

When we require the closed chain constraint to be "on" we simply construct the Reachable Volume representation as described in [3]. When we want it "off", or not considered during some interval in planning, we will simply discard that Reachable Volume and com-

Algorithm 3 TemporalConnect

```
Input:  $g$: temporal map with valid nodes
Output:  new edges added to $g$
 1: function TemporalConnect
 2:     for all nodes $n_1$ in $g$ do
 3:         neighbors = FindNeighbors($n$)
 4:         for all nodes $n_2$ in neighbors do
 5:             if ($n_2$.getTimeStep() - $n_1$.getTimeStep()) == 1 then
 6:                 if ConnectableUsingStraightLineLP($n_2$, $n_1$) then
 7:                     edge = ConstructEdge($n_1$, $n_2$)
 8:                     $g$.addEdge(edge)
 9:                 end if
10:             end if
11:         end for
12:     end for
13: end function
```

pute an unconstrained RV. Though sampling in the unconstrained RV may, with low probability, also produce a closed chained constraint, this should not be worrysome as it will be overwhelmed by non-closed samples. In other words, the limit, as we sample, of the ratio of closed to open sample should equal 0.

We do not run experiments that explicitly use closed chained constraints. One can easily see that our algorithm is extensible to such constraints once the corresponding RV sampler is implemented.

An example of a robot under closed chained constraints is shown in Figure 3.1 in its Reachable Volume.

3.5 Reconstructing the Reachable Volume Space

When multiple constraints are required to solve a problem, we require that the Reachable Volume be dynamically reconstructed so to represent the constraints currently active. Notice that given two reachable volumes R_1 and R_2, R_1 can be recomputed into R_2 if and only if both exist. If they both exist, then a trivial recomputation algorithm is to dis-

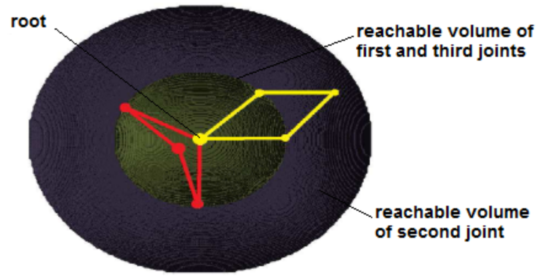


Figure 3.1: A closed chained robot in its Reachable Volume

card R_1 and compute R_2. For the simple end effector constraint, recomputation involves translating the reachable volume from a point a to a point b in the workspace. Finding a faster algorithm for recomputation of more complex structures is still an open problem (see Section 5.1).

4. EXPERIMENTS

In this section we demonstrate the effectiveness of the methods described in chapter 3.

4.1 Machine Setup

The algorithm was implemented in the Parasol Motion Planning Library (PMPL) in C++. The library uses a distributed graph from the Standard Template Adaptive Parallel Library (STAPL) [10]. All experiments were run on a Dell Optiplex 780 running CentOS Linux 7 with an Intel Core 2 Quad CPU Q9550 with 4 GB RAM.

4.2 Environmental Setup

The environment we use is the Empty Box environment. This environment provides a simple testbed that serves to show how drawing can be done with temporal roadmaps.

4.3 Simple Manipulator Robot

We use a 14 DOF manipulator robot with spherical joints shown in Figure 4.1. This is a representative model of a real-world robot as it contains joints and linkages, found in real robots.

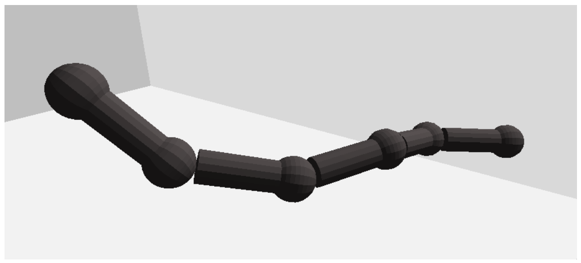


Figure 4.1: Simple manipulator arm with spherical joints

4.4 Drawing with Simple End Effector Constraints

Oftentimes, a robot arm will be asked to follow some smooth curve in space. For example, in industry a robot manipulator may be tasked with painting a car or some other task which requires precise movements.

In this section, we demonstrate the effectiveness of temporal constraints when coupled with Reachable Volumes in handling such constraints. In particular, we solve the drawing problem mentioned above using the simple manipulator robot.

4.4.1 Empty Box Environment

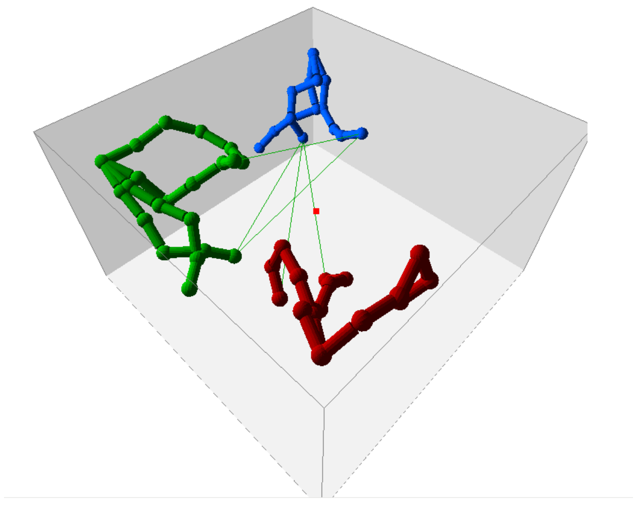
Figure 4.2 shows an empty environment with the robots end effector constrained at some corners of the box. Configurations in different corners represent the robot at different points in time of the problem. In our case, this also means that their end effectors are constrained to different points in workspace.

The temporal roadmap in Figure 4.2 is structured as follows:

- Red indicates phase 1
- Blue indicates phase 2
- Green indicates phase 3

Notice that phase 1 configurations do not connect with phase 3 configurations. As mentioned in section 3.2 this is the intended result as nodes should only connect in consecutive phases, so to maintain chronological order. Each cluster of nodes is centered around one point, indicating that it is the end effector constraint.

To construct a motion planning problem from this graph, set a goal configuration to be an robot configuration in phase 3 and a start configuration to be a robot configuration in phase 1. Then simply follow the edges in one direction in order, to construct a valid path


(a) 10 nodes

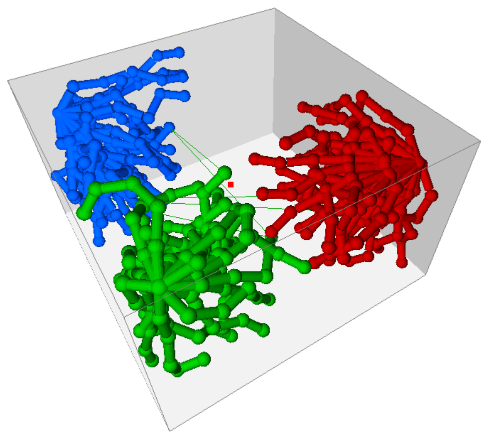

(b) 100 nodes

Figure 4.2: Simple 14-DOF manipulator arms constrained at different points in a simple box environment (no obstacles)

from start to goal. This step is usually done with a graph search algorithm (e.g., Dijkstra's algorithm). In this example the solution to the motion planning problem will provide a solution to the user provided drawing problem.

The robots are free to orient the linkages in their body that do not contain the end effector, thus giving many different instances of the robot for each constraint set. This results from the randomness of Reachable Volume sampler when assembling the robots from their constituent linkages.

4.5 Discussion

The demonstration above shows that Reachable Volumes can efficiently handle manipulator problems that require different constraint at different times in planning.

Specifically, we solved the drawing problem in an empty box environment.

We did not show a comparison between our method and the traditional PRM or RRT methods as it is known that it is with probability 0 that those methods will produce samples that lay on such a small subset of the configuration space. Thus, Reachable Volumes makes possible the handling of constraints, particularly those which require such precision as the ones shown above.

Figure 4.3 shows that the algorithm exhibits an approximate linear slowdown with the number of nodes sampled. This is expected as it takes linear time for the reachable volume sampler to produce a valid node and the added temporal tools should each take constant time.

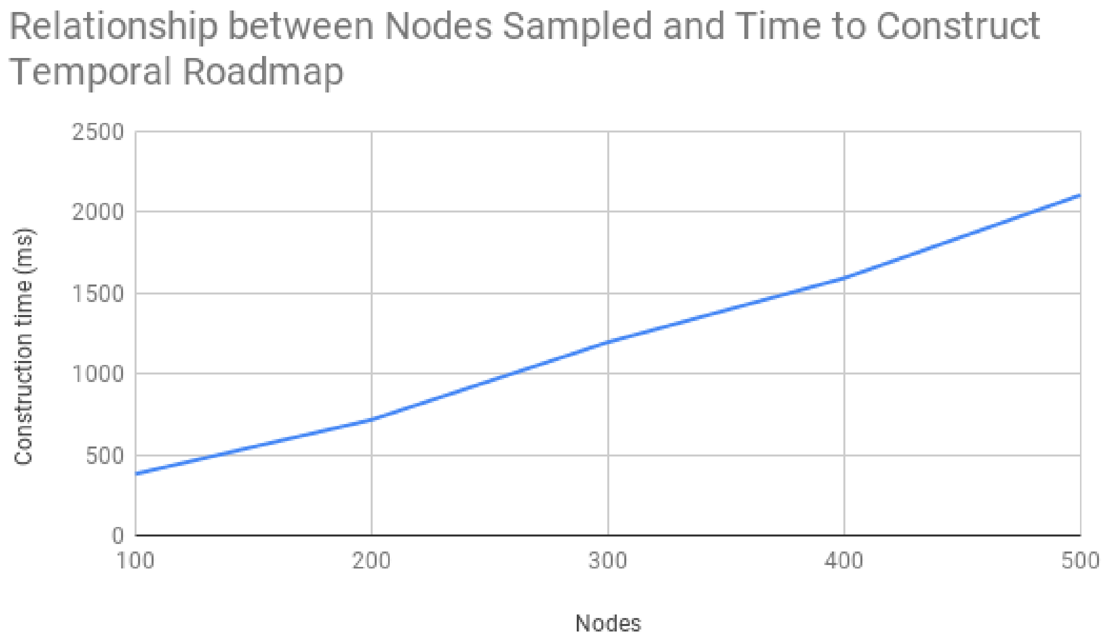


Figure 4.3: Roadmap construction times on the Empty Box environment with 3 end effector constraints

4.6 Limitations

Most of the limitations of the algorithm are mentioned in section 5.1 along with some proposed solutions.

Additionally, it is fair to mention that, in our experiments, we are using the standard local planner that does not sample constraint satisfying samples along the edges. The effects of this are not noticeable due to the fact that the samples exhaust every possible constraint satisfying configuration encountered in the problem. In other words, the constrained space in our case is so narrow (points) that a local planner would not introduce noticeable improvement. When the constrained space is a continuous manifold, a reachable volume local planner is necessary because intermediate configuration produced by the ordinary straight-line planner on edges may not adhere to the constraints in effect.

5. CONCLUSION

In this work, we present an approach to managing temporary constraints in high degree of freedom manipulators robots. Combined with pre-existing work in Reachable Volumes we are able to efficiently plan motion for these robots to solve complex real world problems. Our experiments on a 14-DOF manipulator arm demonstrate that our current approach is effective in planning robots that are under simple end effector constraints. In fact, we show that extending reachable volumes to handle these temporal constraints incurs only a constant time slowdown.

5.1 Further Study

A faster recomputation algorithm for RVs would be interesting and would speed up much of the current planning process. One potential way to do this is to keep a permanent template of the reachable volume, determined by only permanent constraints. Then using this template, compute the needed RV based on the currently needed constraints (stored in the constraint matrix).

Additionally, our current algorithm does not consider the orientation of the end effector, thus we are confined to using spherical graspers (which has a uniform orientation). [11] demonstrates a method for doing this.

We would also like to extend our method to handle grasping, which is a significantly harder problem than drawing, due to graspers being more complex in structure (more DOFs, thus higher dimensional C_{space}). It is also a problem to determine the optimal configuration for a robot arm to grasp an object so that the object is secure.

Finally, information about constraints is currently given by the user. We would like to develop an autonomously method for detecting where a constraint is needed. This brings us to the difficult problem of determining when a robot should release the object and how, if multiple robots are involved, they may collaborate correctly and effectively.

Bibliography

[1] L. Jaillet and J. Porta, “Path planning with loop closure constraints using an atlas-based RRT,” in *15th International Symposium on Robotics Research*, 2011.

[2] M. Bonilla, E. Farnioli, L. Pallottino, and A. Bicchi, “Sample-based motion planning for soft robot manipulators under task constraints,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 2522–2527, 2015.

[3] T. M. McMahon, *Sampling Based Motion Planning with Reachable Volumes*. Ph.D. dissertation, Dept. of Computer Science and Engineering, Texas A&M University, August 2016.

[4] T. Lozano-Perez, “Spatial planning: A configuration space approach,” *Computers, IEEE Transactions on*, vol. 100, no. 2, pp. 108–120, 1983.

[5] J. H. Reif, “Complexity of the mover’s problem and generalizations,” in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, (San Juan, Puerto Rico), pp. 421–427, October 1979.

[6] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Automat.*, vol. 12, pp. 566–580, August 1996.

[7] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, “OBPRM: an obstacle-based PRM for 3d workspaces,” in *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*, (Natick, MA, USA), pp. 155–168, A. K. Peters, Ltd., 1998. (WAFR ‘98).

[8] S. LaValle, J. Yakey, and L. Kavraki, “A probabilistic roadmap approach for systems with closed kinematic chains,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, (Detroit, MI), pp. 1671–1676, 1999.

[9] S. M. Lavalle, “Rapidly-exploring random trees: A new tool for path planning,” tech. rep., 1998.

[10] P. An, A. Jula, S. Rus, S. Saunders, T. Smith, G. Tanase, and N. T. N. M. Amato, “Stapl: A standard template adaptive parallel c++ library,” in *In Int. Wkshp on Adv. Compiler Technology for High Perf. and Embedded Processors*, p. 10, 2001.

[11] T. McMahon, R. Sandstrom, S. Thomas, and N. M. Amato, “Manipulation planning with directed reachable volumes,” in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, (Vancouver, Canada), September 2017.