CLEARANCE MAP FOR THE APPLICATION OF WORKSPACE SKELETON-BIASED MOTION PLANNING

An Undergraduate Research Scholars Thesis

by

QINGQING LI

Submitted to the Undergraduate Research Scholars program
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 2017

Major: Electrical and Computer Engineering

ABSTRACT

Clearance Map for the Application of Workspace Skeleton-biased Motion Planning

Qingqing Li
Department of Electrical and Computer Engineering
Texas A&M University

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

Motion planning is the problem of finding a valid path for a moveable object from a start to a goal state. Current state-of-the-art sampling based planners are able to solve a variety of problems. Sometimes, it is difficult for them to find a collision-free path if the solution path is highly related to the workspace geometry and the free space is relatively restricted.

Recent works have indicated that the planner can benefit from the inherent topological information in the workspace. These algorithms capture the topology of the free workspace in form of a graph called workspace skeleton. Nevertheless, they do not allow preferential selection of solution path if multiple paths exist. For example, some of the paths might not be feasible for robot of certain size.

In this work, we address the preferential selection of paths based on the clearance requirements of the planning problem. To achieve this, we annotate the workspace skeleton with clearance information. Given the clearance requirements of the problem in form of a filter function, we generate a subgraph that contains the edges from the original skeleton satisfying the function. This saves time in planning unnecessarily in regions of the

workspace that are infeasible based on the clearance requirements of the problem.

We study the impact of the annotated skeleton on the performance of a recent sampling based motion planner that uses workspace skeleton. We use two kind of filter function : one that filters edges based on the size of the robot, and other that forces following a specific route. Our results show improvement in both planning time (6.5% in some cases) and number of robot placements sampled before finding the solution path.

TABLE OF CONTENTS

Page

1. INTRODUCTION

A basic motion planning problem is to produce a continuous motion for a moveable object or a *robot* that connects a start placement and a goal placement with a collision-free path. Besides robotics, it has other applications including computational biology [1], virtual prototyping [2] and computer animation [3]. Currently, sampling-based motion planners are considered state-of-the-art solution for motion planning and have been widely applied to solve high-dimensional problems such as protein folding [1] and robotic manipulation [4].

Sampling-based motion planners work by constructing a graph or a tree of randomly generated valid placements of the robot or samples. Then graph search algorithm can be applied on the graph to find a valid path connecting start and goal positions. Rapidly-exploring Random Tree (RRT) [5] is one of the sampling based method adopted for to handle differential constraints of robot in motion planning. It works by generating a valid random sample, and extending the nearest sample in the current tree towards the sample until a solution path is found. However, exploring narrow passages or cluttered workspace remains a challenge for RRT.

Variants of RRT have been developed to address exploration in narrow passages by utilizing topological features of workspace. One such work, Medial Axis Biased Rapidly-Exploring Random Trees (MARRTs) [6], guide tree exploration to the medial axis of free space by pushing all samples from expansion steps towards the medial axis. MARRT also increases the clearance of the samples from the workspace obstacles. Recently, Dynamic Region-biased Rapidly-exploring Random Trees (DRRRTs) [7], present a novel planning process where it captures the topology of free space in Reeb Graph [8] and use it to bias the growth of RRT. In this work we generalize the graph embedded in the free work space

as *workspace skeleton*, that encodes the topology of the free workspace. Medial axis in 2D workspace and Reeb Graph are examples of workspace skeleton that could be used to inform the path search.

Although the workspace skeleton could capture the topological feature of free space and properly resolve the problem of the coverage of narrow passages in workspace, it cannot choose if multiple solution paths exist. Robot might struggle to pass through some of these paths considering its size as shown in Figure 1.1. In addition, workspace skeleton has little control of selection of specific robot paths. However, in some planning problems it might be preferred if robot could follow specific routes or avoid certain passages. For example, in a navigation problem it is more reasonable if vehicle could avoid roads under construction.

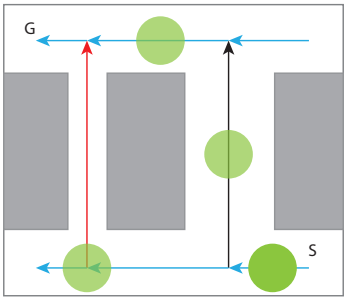


Figure 1.1: It is not feasible for a circular robot to follow red trajectory considering its size. However, the black trajectory is more preferred.

In this work, we annotate the workspace skeleton with a clearance map to allow preferential selection of skeleton edges during planning. Our method begins by processing a generated skeleton before the planning stage. We compute the minimum clearance along each edge in skeleton and store them as an additional information in the skeleton. Then, a custom filter function is defined and it checks the clearance of each route in skeleton to

generate a subgraph of the skeleton that satisfy the clearance requirement of the problem defined by filter function. If the clearance of certain edges do not satisfy the condition defined in filter function, the edges will be removed from the resulting subgraph. In this way, our work could potentially aid in avoiding planning time in parts of the workspace that do not satisfy clearance requirements of the planning problem.

We study the impact of the annotation of the workspace skeleton on the performance of the planner by pruning the skeleton edges impassable by robots of different sizes. We also examine whether we could bias the solution paths along specific passages or avoid certain passages. We implemented three kinds of experiments that include varying the size of robot, varying the filter function and application to real world problem.

Our contributions include:

- A algorithm to annotate workspace skeleton with clearance information.
- A framework for filtering edges in the workspace skeleton:
  - to routes passable by the robot based on its size.
  - to bias the paths along specific routes or avoid certain routes.
- Experimental validation and evaluation of our algorithm based on DRRRT with performance analysis and application to real-world motion planning problem.

2. PRELIMINARIES AND RELATED WORK

A basic motion planning problem is to produce a collision-free trajectory for a moveable object called robot from a given start state to a goal state. The state of robot is usually described by certain parameters including position, orientation, velocity, and link angles. So the state of robot, or *configuration*, could be uniquely represented by a point in an n-dimensional space, which is also known as *configuration space* [9]. The motion planner finds a path in the configuration space.

When the state of the robot is defined by just the position of the robot, the configuration space is same as the workspace. However, the configuration space is not always same as the workspace. As our contribution is heavily dependent on the workspace, it can be applied to problems where workspace obstacles have significant impact on the performance of the planner.

Sampling-based planners (e.g., Probabilistic RoadMap (PRM) [10] and Rapidly-exploring Random Tree (RRT) [5]) have solved variety of motion planning problems. PRM [10] constructs roadmaps of free space by randomly sampling collision-free configurations and connecting nearby samples with a valid path. RRT [5] is widely applied for kinodynamic of robot. Starting at a given configuration, RRT incrementally grow a tree to explore the configuration space and find a path connecting the start and a goal positions.

Although these basic planners are able to address variety of challenges, RRT and PRM have difficulty exploring the narrow passages in a cluttered space as the probability to land a valid sample in such cluttered environment is relatively low. Hence, many efforts have been made to extract workspace information to guide exploration in these narrow passages. For example, Obstacle-Based RRT (OBRRT) [11] guides expansion to narrow passage by sampling near the obstacle surface. Retraction-based RRT [12] can effectively cope with

narrow passages by retracting nodes along obstacle boundaries.

2.1 Medial Axis Biased RRT

Medial Axis-Biased RRT (MARRT) [6] is an approach that guides the growth of RRT on the medial axis skeleton of free space, resulting in greater path clearance. Instead of growing from closest node of the tree towards a random configuration, it forces the growth near the medial axis by generating a new configuration towards the random configuration and then pushing the new configuration to the medial axis. The process is iterated until a maximum expansion length is reached or expansion fails to make progress or two successive nodes are not visible to each other.

MARRT tries to explore every possible passage but with its objective is to place the robot at high clearance from the obstacles. In contrast, the objective of our work is to use clearance to avoid planning in the regions in workspace that does not satisfy the clearance requirement of the problem.

2.2 Spark PRM

Spark PRM [13] is a hybrid method that combines the PRM with RRT to help in exploring the narrow passage efficiently. It exploits the benefit of RRT which could efficiently explore the constrained space once inside it. Spark PRM generates PRM in wide free space while limit growing RRT solely in narrow passage. It works by performing narrow passage test every time a configuration is generated by PRM. Instead of studying the geometry of workspace, Spark PRM uses randomly generated configuration to probe the workspace information.

2.3 Dynamic Region-biased RRT

While beneficial in some scenario, RRT still finds it hard to discover a solution path if the passage has complicated geometry. Dynamic Region-biased RRT (DRRRT) [7] is

an approach designed to overcome this weakness by rendering the topological feature of workspace and using it to bias the exploring of RRT. DRRRT first computes the embedded graph that captures the topology of workspace with Reeb Graph [8]. A flow in the embedded graph is generated leading from source to goal position. During planning stage, multiple active regions are created and moved along the flow of each edge to guide the growth of RRT until a solution path is found. If multiple solution paths exist, DRRT cannot distinguish one from the other. In contrast, the skeleton annotated with clearance information would assist in preferring one solution path over other based on the clearance requirements of the problem.

3. ALGORITHM

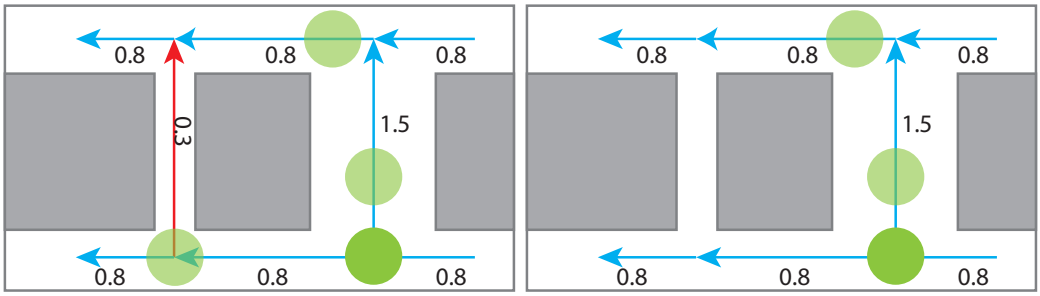
In this work, we annotate the workspace skeleton with clearance information and use it to bias planning along the skeleton edges that satisfy the clearance requirements of the problem. In this section, we detail the definition of workspace skeleton (the data structure used), algorithm for annotating the skeleton and its application in DRRRT to improve the performance.

3.1 Workspace Skeleton

The annotation of the skeleton is achieved by building a clearance map that associates each edge in the skeleton with its clearance information. The workspace skeleton is a graph embedded in the free workspace. (See Figure 3.1(a)). Each vertex of the graph indicates a point in the workspace. For example, if the skeleton is a Reeb graph, the vertices are the Reeb graph nodes. Each edge is represented as a set of interpolated workspace points connecting two vertices in the skeleton.

3.2 Annotated Workspace Skeleton

Algorithm 1 describes how we construct clearance map to annotate the skeleton. Each edge in the skeleton is traversed and the clearance for each workspace point in the edge is computed. Finally, the weight of the edge is assigned as the minimum clearance value of the interpolated points along the edge.



(a) Workspace Skeleton (b) Filtered Subgraph

Figure 3.1: The red trajectory in (a) of 0.3 clearance is too narrow for circular robot to pass through so the red edge is filtered. (a) Original Workspace skeleton of free space (b) Resulting subgraph of skeleton with edges from the original skeleton satisfying clearance requirement

Algorithm 1: InitializeClearanceMap

Input : Workspace skeleton $S(V, E)$
Output: Clearance Map, C

```
clearance_map initialization;
for each edge e in workspace skeleton edges E do
    minDist = ∞;
    for each unvisited workspace point p in e do
        dist = compute clearance of p ;
        if dist < minDist then
            minDist = dist;
        end
    end
    C.put(e, minDist);
end
```

Algorithm 2 describes how to obtain the filtered workspace skeleton satisfying the clearance requirements of the problem. The clearance requirements of the problem is defined as a filter function that takes a skeleton edge as an input and returns true if the clearance or weight of the edge satisfies the clearance requirements of the problem.

Given a workspace skeleton with a clearance map, and a custom filter function, a subgraph of the skeleton is generated with edges that satisfy the condition given in the filter function. As shown in Figure 3.1(b), the resulting subgraph preserves those edges

that are wide enough for the robot to pass through.

Algorithm 2: FilteredSubgraph

Input : Clearance Map C, Workspace skeleton $S(V, E)$, Custom filter function $F(c)$
Output: Subgraph of skeleton

```
Initialize a graph G with skeleton vertices V;
for each edge e in skeleton edges, E do
    if F(C.get(e)) then
        Add edge e to G ;
    end
end
return G ;
```

3.3 Application to Sampling based Motion Planner

We verify the impact of annotated skeleton in improving the performance of DRRRT algorithm. DRRRT algorithm works by first computing the Reeb Graph (workspace skeleton) of free space. And it guides the growth of RRT by creating sampling regions and moving them along the edge of skeleton. The application of the annotated skeleton in DRRRT algorithm is described in Algorithm 3. Instead of using the original Reeb graph, we use the filtered subgraph generated by the clearance map and the filter function. In this way, the clearance map is used to prune the skeleton edges that do not meet clearance requirement for DRRRT algorithm so that only feasible edges are used to guide robot.

Algorithm 3: Application of clearance map in DRRRT

```
Reeb graph, RG = InitializeDRRRT() ;
C = InitializeClearanceMap(RG);
Filtered Subgraph, SG = FilteredSubgraph(C,RG,F(c));
CreateSamplingRegions(SG);
while not done do
    Region biased RRT growth;
end
```

4. EXPERIMENTS AND RESULTS

In this section, we will analyze the impact of using the annotated skeleton on the performance of DRRT, a sampling based motion planner that uses workspace skeleton. We study the performance of the algorithm with respect to total planning time, overhead due to clearance annotation and size of the roadmap i.e., number of samples created before a solution path is found. All experiments are performed in Linux machine with 10 runs. All methods are implemented in the Parasol Lab motion planning library developed at Texas A&M University.

4.1 Performance impact due to filtering by robot size

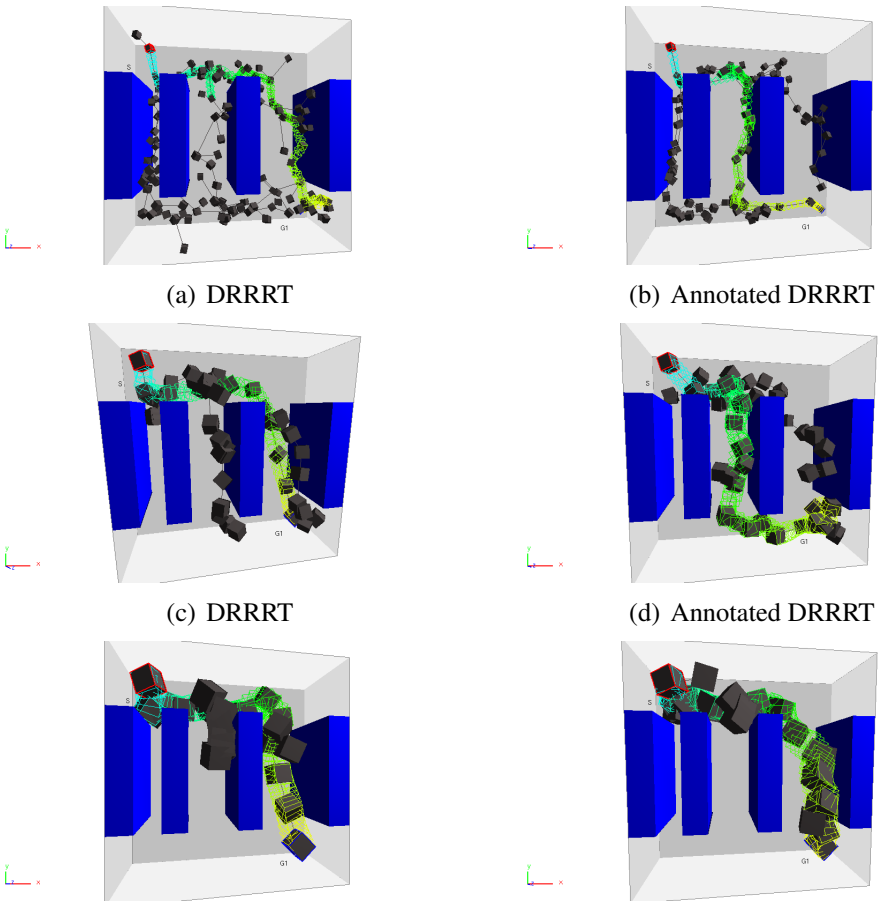
This experiment is designed to verify if annotated workspace skeleton will select more appropriate routes considering robot's size and study its impact on the planning time and roadmap size and the overhead of initialization of clearance map.

For the environment, four blocks with varying passage width are used. The calculated clearance of each passage are 0.125, 0.25, and 0.375 respectively (See Figure 4.1(a)). The square robot has six DOFs. The filter function filters any edge of skeleton whose clearance is smaller than robot radius.

DRRRT and annotated DRRRT generate similar roadmap. However, DRRRT might spend additional time exploring those unfeasible passages considering its size. And annotated DRRRT might spend less time exploring those regions since filter function filters the unfeasible edges. We will validate our claim in following subsection.

4.1.1 Performance

We ran each experiment shown above ten times and averaged the values for analysis. As Figure 4.2 (a) and (b) show, irrespective of robot size, size of roadmap generated by



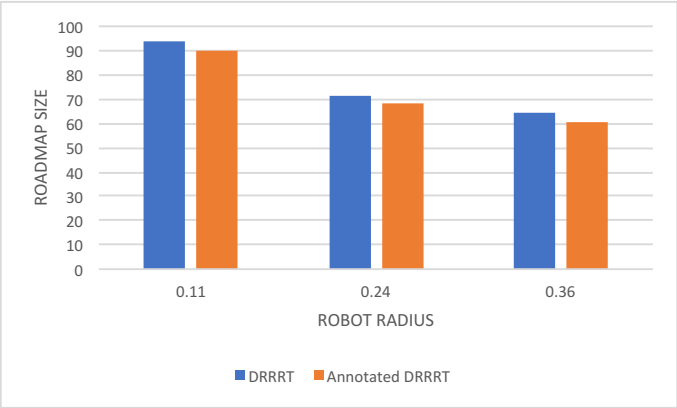
(a) DRRRT (b) Annotated DRRRT

(c) DRRRT (d) Annotated DRRRT

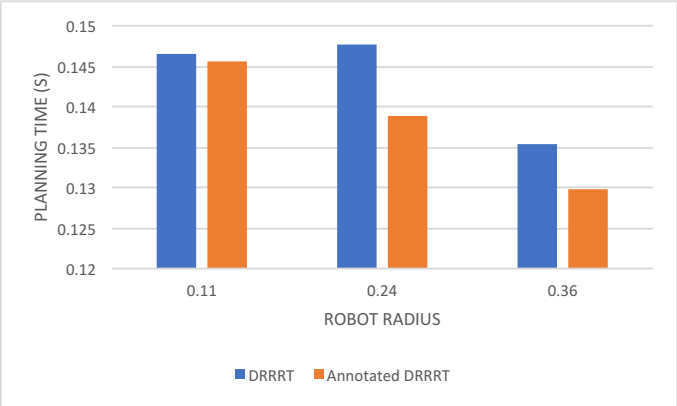
(e) DRRRT (f) Annotated DRRRT

Figure 4.1: (a) (b) Robot radius: 0.11; (c) (d) Robot radius: 0.24; (e) (f) Robot radius: 0.36

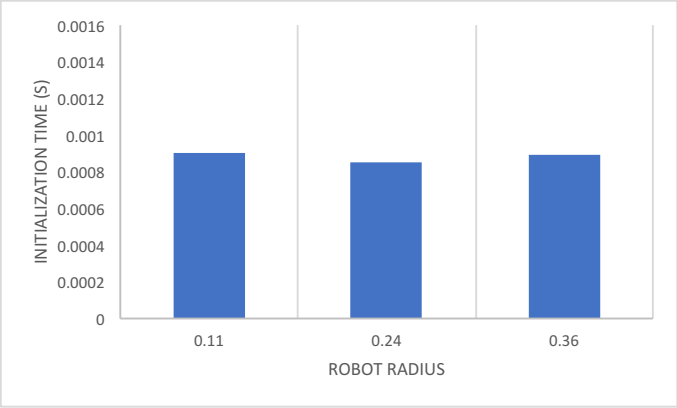
annotated DRRRT is smaller than DRRRT, and total planning time has been reduced. Regardless of robot of different size, the initialization overhead is almost the same as Figure 4.2 (c) indicates. Besides, the overhead time taken to create clearance map accounts for fairly small portion of the total planning time, around 0.6%.



(a) Roadmap Size



(b) Planning Time



(c) Initialization Time

Figure 4.2: (a) Averaged roadmap size of DRRRT and annotated DRRRT for each robot radius (b) Averaged initialization time of clearance map for each robot radius and (c) Averaged initialization time of clearance map for each robot radius

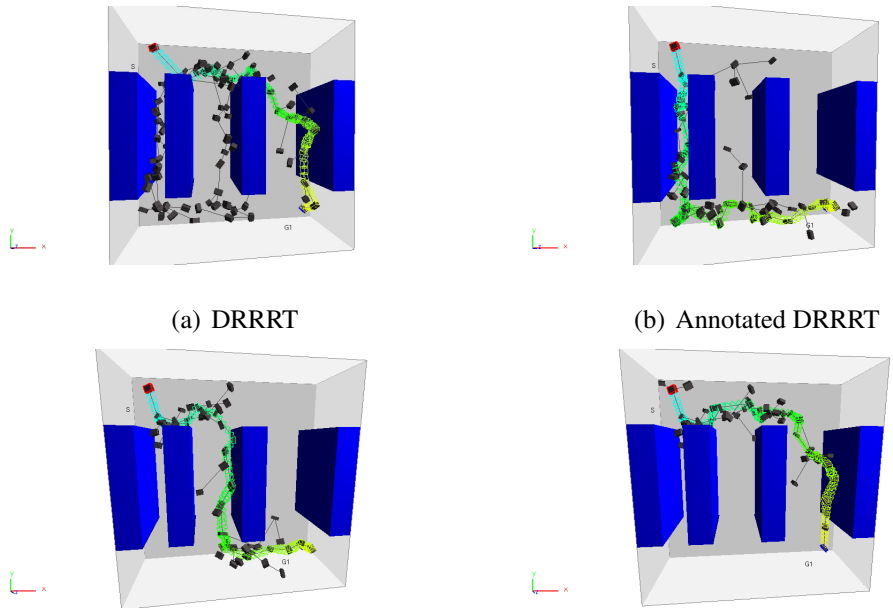
With the constraint in filter function, DRRRT annotated by clearance map selects an appropriate route in terms of robot radius. The clearance map could bias DRRRT to explore a more appropriate passage considering its size instead of exploring other passages robot might find struggling to pass through. Uninformed DRRRT might try to explore every edge (route) of its skeleton, which could potentially increase additional planning time.

4.2 Preferential selection of robot paths

This experiment is designed for discussing whether we could bias the paths along specific passages via customized filter function.

In this experiment, we use the same environment and type of robot as last experiment. The radius of the robot is fixed to 0.119. Different filter function has been employed to bias along specific passages. In Figure 4.3(a), DRRRT isn't applied on any filter function. In Figure 4.3(b), filter function returns true if clearance of the edges is greater than twice the robot size. In Figure 4.3(c), filter function returns true if clearance of edges is less than twice of robot radius or greater than third times robot size. In Figure 4.3(d), filter function returns true with clearance of edges greater than third times robot size.

In DRRRT as in Figure 4.3(a), all possible routes in different passages have been constructed or under construction. Usually, if a path is generated first, robot will take the first generated route. So it is hard to infer which route robot will take. In Figure 4.3(b), (c) and (d), we vary the filter function so that other two passages are filtered, leaving only one path for robot.



(a) DRRRT (b) Annotated DRRRT

(c) Annotated DRRRT (d) Annotated DRRRT

Figure 4.3: Varying filter function to bias along specific passages (a) DRRRT without any filter function (b) filter function: clearance > 2× robot radius (c) filter function: clearance < 2× robot radius or clearance > 3× robot radius (d) filter function: clearance < 3× robot radius

Instead of generating all possible routes as in DRRRT, only preferred paths are preserved and followed while other paths are filtered out in the updated algorithm. We could infer that custom filter is promising to bias path search to specific passages.

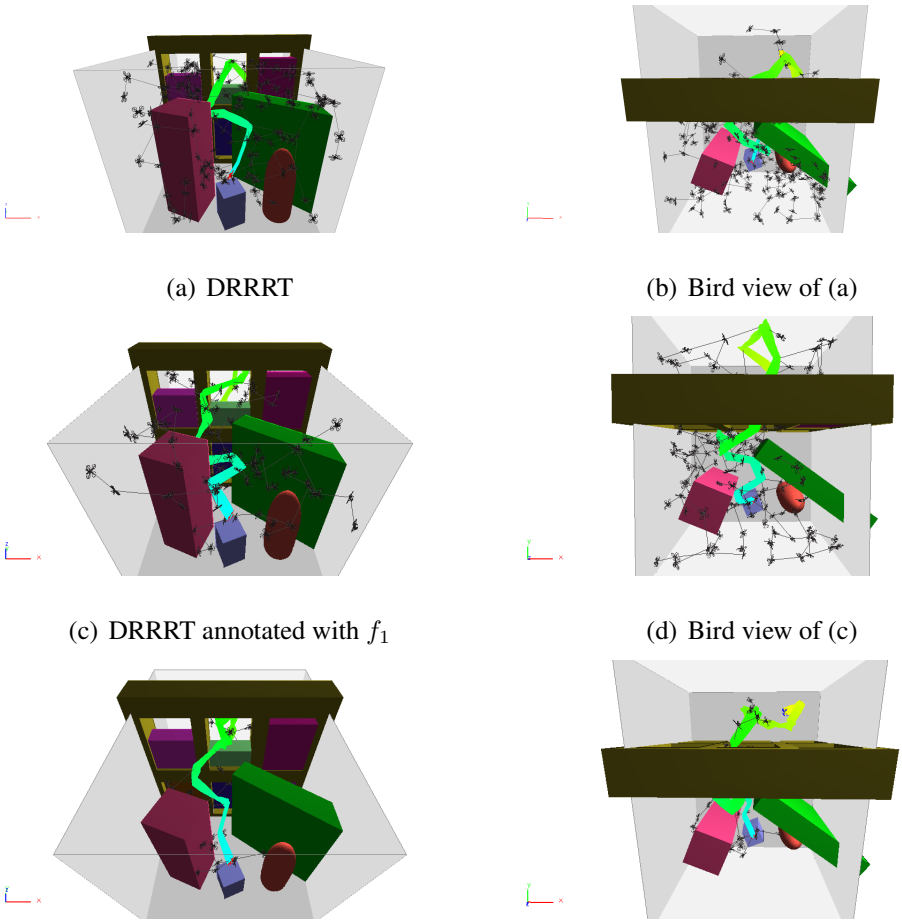
4.3 Real-life Motion Planning Problem

In this experiment, we will explore the performance of clearance map in a real-world motion planning problem, city environment, from the perspective of planning time and roadmap size. We will further consolidate the potential of preferential selection of routes in real-world environment with comparisons of two different filter functions.

We use the city environment (See Figure 4.4) and a drone robot of 6 DOFs for simulating a real-world motion planning problem. In the environment, shelf-like building is

being under construction with first floor finished and second floor unfinished. Three windows in second floor have blocks of different height in it while all of the windows in first floor are totally blocked. The rightmost window in second floor is almost blocked leaving little space such that drone could not pass through it at all. The window in the middle has highest clearance, 2.3. Nearby two windows have clearance less than 2.3, with 1.4 and 0.4 respectively. There are four buildings of different size and shape in front of shelf building. The query is placed at each side of shelf building. The radius of the robot is 0.9. We compare DRRRT against the annotated one.

We use two kinds of filter function, f_1, filtering the edges of skeleton by robot radius, and custom filter function, f_2, intentionally filtering the edges whose clearance is smaller than 1.5.



(a) DRRRT (b) Bird view of (a)

(c) DRRRT annotated with f_1 (d) Bird view of (c)

(e) DRRRT annotated with f_2 (f) Bird view of (e)

Figure 4.4: DRRRT and annotated DRRRT with two filter functions

We observe that f_1 has negligible effect on DRRRT (See Figure 4.4 (a) (b) (c) (d)) with respect to the resulting roadmap as compared to DRRRT filtered with f_2 (See Figure 4.4 (a) (b) (e) (f)). This is due to the resulting subgraph of workspace skeleton for f_2 only guides RRT passing through the window in the middle.

4.3.1 Performance

Table 4.1: The Comparison of performance between DRRRT and annotated ones

	Averaged Roadmap Size	Averaged Total Planning Time (s)	Averaged Initialization Time of Clearance Map (s)
DRRRT	263.14	1.640	N/A
Annotated DRRRT with f_1	252.8	1.629	0.0109
Annotated DRRRT with f_2	102	1.359	0.0110

As given in Table 4.1, we observe that annotated DRRRT with custom filter function, f_2, has significant improvements in roadmap size and total planning time as compared to those of the annotated DRRT with custom filter function, f_1. The initialization time of clearance map is negligible compared to the total planning time for (See Table 4.1) for both the filter functions. However, DRRRT annotated by f_1 shows minor improvements both in roadmap size and total planning time over the unannotated one.

Thus, various selections of filter function could significantly influence the resulting roadmap and performance. We further prove that DRRRT could be biased towards the paths in certain passages or avoid some routes by choosing appropriate custom filter funtion in a real-world problem.

5. CONCLUSIONS

In this thesis, we annotate the workspace skeleton with clearance information to improve the performance of sampling based planners that uses workspace skeleton. The additional annotations help in filtering out edges from the workspace skeleton that do not satisfy the clearance requirement of the planning problem.

We analyzed the performance of the annotated skeleton to filter out unfeasible paths considering robot size and the selection of preferential routes with customized filter function. Our experiments show that the annotated skeleton improves the planning time of a recent sampling based planning algorithm that uses workspace skeleton.

In the future, we would like to apply clearance map with a wider variety of skeleton planner, such as MARRT.

REFERENCES

[1] N. M. Amato and G. Song, "Using motion planning to study protein folding pathways," *Journal of Computational Biology*, vol. 9, no. 2, pp. 149–168, 2002.

[2] O. B. Bayazit, G. Song, and N. M. Amato, "Enhancing randomized motion planners: Exploring with haptic hints," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1, pp. 529–536, IEEE, 2000.

[3] J.-M. Lien, O. B. Bayazit, R. T. Sowell, S. Rodriguez, and N. M. Amato, "Shepherding behaviors," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 4, pp. 4159–4164, IEEE, 2004.

[4] T. McMahon, S. Thomas, and N. M. Amato, "Sampling based motion planning with reachable volumes: Application to manipulators and closed chain systems," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 3705–3712, IEEE, 2014.

[5] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[6] J. Denny, E. Greco, S. Thomas, and N. M. Amato, "Marrt: Medial axis biased rapidly-exploring random trees," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 90–97, IEEE, 2014.

[7] J. Denny, R. Sandström, A. Bregger, and N. M. Amato, "Dynamic region-biased rapidly-exploring random trees,"

[8] G. Reeb, "Sur les points singuliers dï£¡une forme de pfaff completement intégrable ou dï£¡une fonction numérique," *CR Acad. Sci. Paris*, vol. 222, no. 847-849, p. 2, 1946.

[9] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

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

[11] X. Tang, J.-M. Lien, N. Amato, *et al.*, "An obstacle-based rapidly-exploring random tree," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 895–900, IEEE, 2006.

[12] L. Zhang and D. Manocha, "An efficient retraction-based rrt planner," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3743–3750, IEEE, 2008.

[13] K. Shi, J. Denny, and N. M. Amato, "Spark prm: Using rrts within prms to efficiently explore narrow passages," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 4659–4666, IEEE, 2014.