Unions of Balls and Configuration Space

Yinan Zhang
Department of Computer Science
Dartmouth College
Hanover, New Hampshire 03755, US

Devin Balkcom
Department of Computer Science
Dartmouth College
Hanover, New Hampshire 03755, US

August 23, 2014

Abstract

Covering both the configuration space, or *C-space*, and represent the right topology is hard to achieve at the same time with traditional sampling-based motion planning algorithms, such as Probabilistic Roadmap and Rapidly-Exploring Random Tree. In this paper, we proposed a method that is able to cover near optimal path and study the shape and topology of configuration space. Our method samples a serious of balls to cover the space and use the dual shape of the union of balls to re-construct the space. The output of our algorithm can be viewed as a rich representation of *C-space*. Benefits include (1)low memory for planning, (2) ability of analyzing shape and topology of *C-space* and (3) fast planning for large space. We analysis the coverage of the space and show how these balls and their dual shape can be used to generate a graph that maintains the topology of *C-space*.

1 Introduction

A robot state can be represented using n parameters $< x_1, x_2, ..., x_n >$ as a point in a n-dimensional space X, called *Configuration Space*, or *C-space*[15]. In this paper, we present a new method to model *C-space*. Our algorithm is sampling-based, We will sample balls in *C-space* to cover the space. By using the dual shape of unions of balls we construct the shape of free space, and capture its topology. A topology roadmap will then be

built which we can use to determine path homotopy type, to find paths in different homotopy classes, and to answer other topology questions. We implemented the algorithm in a 2D world with a point robot.

Representing *C-space* is very fundamental for motion planning. Covering the space and characterize the topology are two concerns. By covering the space better, we can find better paths. By studying its topology, we can answer questions like "does a path exist between two configurations?", "does two paths has the same topology?" and "can we find two path that have different homotopy types?". These questions arise frequently in applications as predicting paths for unpredictable entities and deployment of multiple agents for efficient exploration of an environment[6]. Because of the extensive applications of motion planning to areas like augmented reality [9], computer-aided design [10] and bioinformatics [11]. Answering these questions could benefit many researches.

Previous methods have their disadvantages. Some sampling-based methods, such as *PRM** and *RRT** [7], are able to cover the space, but failed to capture the topology [4] because of "holes" in the roadmap and inability to determine if a "hole" contains obstacles. *Visibility-base Probabilistic Roadmap*, or *vis-PRM* [8], tried to capture the topology by placing "guards" to "see" the space. However, they can't describe "visible regions" and overlapping regions explicitly, which makes it difficult to analyze coverage and find optimal paths. Medial Axis sampling attracted researchers attention, such as *MAPRM*[18] and *UMAPRM*[15], for their property of characterizing the topology of *C-space* and safe clearance to obstacles, however, no structures are made with these samples to properly represent the space. We have seen some studies in low dimensional space trying to characterize the topology of *C-space*, such as [12] and [13], these methods suffer dimension curse in higher dimensional space. In [5] and [6], Bhattacharya, et al, studied searching path in a graph with topological constrains, and their method can be extended to higher dimensional space

Our method addresses these problems by sampling weighted configurations, or balls, and constructing the dual shape of union of balls. Thus we are able to analyze the coverage of the space and answer topology-related questions. Besides, our algorithm has many advantages. It requires low memory for planning paths, allows analysis of shape and topology of *C-space* and fastens planning for large space.

In this paper, we are sampling in (X, ρ) metric space. Let x be a configuration in X(usually *C-space*), $\rho(x)$ gives the clearance of configuration x, negative if infeasible, positive if feasible. By *"clearance"*, we mean its (estimated) distance to the nearest obstacles. Our experiments environment is a

point robot with polygonal obstacles, so we can fast determine the clearance of a configuration. (In practice, workspace doesn't have to be the same as configuration space, as long as we can get the clearance of a configuration which can be done by checking the collision distance of a robot.)

2 Related Work

In this section, we first discuss the basics of sampling-based motion planning and the method of sampling on medial axis. With the information of positions and their clearance in *C-space*, we have samples as spheres, or balls. We then discuss the dual shape of the union of balls which plays an important role in understanding the shape and topology of *C-space*. Some works on graph theory and topology will finally be introduced.s

A *Sampling-based Motion Planning*

A robot is a movable object whose state can be described by n parameters, or *degrees of freedom* ($DOFs$). A point $< x_1, x_2, ..., x_n >$ in a n-dimensional space uniquely defines the configuration of a robot. Such space is called *"Configuration Space"* or *"C-space"* (C). The subset of all feasible (collision-free) configurations is called the *free space* (C_{free}), while $C_{obst} = C \setminus C_{free}$ is the union of all infeasible configurations[15]. Path planning will generally be viewed as a search in a metric space X for continuous paths from an initial state x_{init} to a goal region $X_{goal} \subset X$. For a standard problem, $X = C$ [16]. What needs to be pointed out is that *C-space* is very different from workspace which is usually formed by polygons. The shape of *C-space* is usually unknown to us, unless the robot is a point, in which case workspace is the *C-space*.

Sampling-based algorithms have been very successful and have seen many applications in industry for planning in high dimension space. Probablistic Roadmap [17] and Rapidly-Exploring Random Tree [16] are two outstanding methods. The basic idea under these methods is to randomly sample feasible configurations in *C-space* and connect nearby valid samples as edges to construct a graph or tree structure, which careless about the dimension problem. By connecting start and goal configurations to the roadmap, a graph search, e.g., A*, can be performed to extract a near optimal solution path. Problem is it's hard to analyze coverage of the space with these methods. And "holes" in a roadmap cannot tell if there are obstacles, thus we can't use it to analysis path topology. Our method can avoid these problems by analyzing balls and their dual shape.

Some methods do captures topology of *C-space*. *Visibility-based Probabilistic Roadmap*[8] might be one of them. By randomly sampling "guards" that can't see each other, the algorithm can produce guards that cover most of the space. Then connection nodes will be generated to connect guards. Our method is very much related to *Vis-PRM*. "Guards" of *Vis-PRM* describe their "visible regions" implicitly, while our method limit a configuration to have a "visible range", the clearance. Thus, we can describe overlapping regions and visible regions explicitly.

It's still possible to use *PRM* to find paths with topology constrains. In [5] and [6], by combining integration theory with search techniques, Bhattacharya, et al, studied searching path in a graph with topological constrains, and their method can be extended to higher dimensional space.

Methods like *PRM** requires much memory to store the roadmap because of the large amount of vertices and greater amount of edges. Some algorithms, such as [1] and [2], have been proposed to reduce the memory requirement by reducing edges of *PRM**. These methods try to solve the problem after it happened, while our method tries to avoid the problem from happening.

C *Medial Axis Sampling*

A simpler method that maintains the topology of *C-space* but doesn't cover near optimal path is medial axis sampling. Medial Axis is also called Generallized Voronoi Diagram. The medial axis $MA(C_{free})$ of free space C_{free} is a strong deformation retract of C_{free}, which means C_{free} can be continuously deformed onto $MA(C_{free})$ while maintaining its topological structure. However, medial axis is hard to compute explicitly.

In [14], sec. 5.6.3, S. LaValle defined Medial Axis as "Let (X, ρ) be a metric space. Let a *maximal ball* be a ball $B(x, r) \subseteq X$ such that no other ball can be a proper subset. The centers of all maximal balls trace out a one-dimensional set of points referred to as the *medial axis*".

Mapping a pair of points A and B in free space onto $MA(C_{free})$, there is a path between A and B if and only if there is a path between their images on $MA(C_{free})$. Such path has a very nice property of the largest clearance from obstacles.

S. Wilmart [18] sampled on the medial axis of free space to build up a probabilistic roadmap planner. He also showed that "it is possible to efficiently retract a configuration, free or otherwise, onto the medial axis of the free space without having to compute the medial axis explicitly".

The basic idea of the algorithm proposed in [18] is: given a point A and

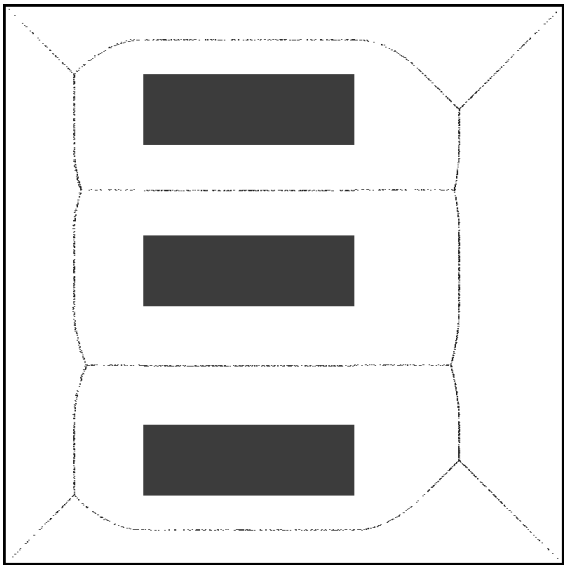

Figure 1: Sampling on the Medial Axis of a rectangle space. The black blocks are obstacles. We will use this world as our working space in the following sections.

its nearest point on the boundary of obstacles A', a ray start from A with direction $\overrightarrow{A'A}$ will always intersect with $MA(C_{free})$.

This method results in sampling more nodes in narrow corridors, which is beneficial for improving performance on problems requiring traversal of such corridors.

However, such property makes the sampling method biased heavily towards certain portions of the medial axis. These biased property tend to ignore more critical parts of the space: large open free space. Sampling uniformly on the medial axis is sometimes more helpful.

Yeh [15] solved this problem by putting some random "sticks" with random directions in free space. Any surfaces will intersect uniformly with these sticks, thus searching on sticks to find points that intersect with medial axis will give nodes uniformly distributed on the medial axis.

Figure 1 visualizes the shape of medial axis of a rectangle.

C *Dual Shapes of Unions of Balls*

Approximating the shape of an unknown area to study its properties is a very commonly used method. We have seen research ideas in the study of

biogeometry to solve the problem. As for robotics, a few people began to notice the achievements from biogeometry study and tried to apply similar methods to study the shape of *C-space*. [21]

We have seen papers concerning the shape of *C-space*. Robin, et al, developed a computational method for detecting large convex regions of obstacle-free space[3]. Knowing such information can be helpful to plan robot footstep locations while avoiding obstacles and other optimization problems. But this method can only capture local features.

Z. McCarthy, et al, [24] used samples generated by PRM to form the α-shape of *C-space*. She then proved there doesn't exist a path between a pair of configurations by proving they are in two different disconnected components of the shape. She declared it was the first time α-shape is being used in motion planning.

Define a *simplicial complex* D: a collection of simplices such that if $\triangle_T \subseteq D$ ($\triangle_T$ is the convex hull of T), then if $U \subseteq T$, we have $\triangle_U \subseteq D$ and also that the intersection of two simplices in D is either empty or another simplex in D. The α-shape of a set of points S is a generalization of the convex hull of those points[24]. H. Edelsbrunner first introduced them in [22]. By choosing different values of the one parameter α, we can have a family of shapes with the same set of points. An example of α-shape is in Figure 2.

Choosing the best α value could be a bothering work. Assuming weighted points in the point set, by using a generalization of the Delaunay Triangulation, the Regular Triangulation, we can construct another shape using weights as flexible α's. Points with weights can be considered as balls. H. Edelsbrunner gave us this idea in [20]. Such shapes are also called Dual Shapes or Dual complex of unions of balls. The weighted α-shape captures the topology of the union of balls with different radii. Figure 3 shows an example of such weighted α-shape.

D *Betti Numbers*

In algebraic topology, the Betti Numbers are used to distinguish topological spaces based on the connectivity of n-dimensional simplicial complexes. Informally, the *n-th* Betti number (β_n) represents the rank of the *n-th* homology group, which tells us the maximum number of cuts that can be made before diving a surface into two pieces. [25]

An example of Betti numbers will be a torus. Figure 4

The first Betti number of a graph is very easy to compute. Given a graph G, with n vertices, m edges and k components. It's 1st Betti number

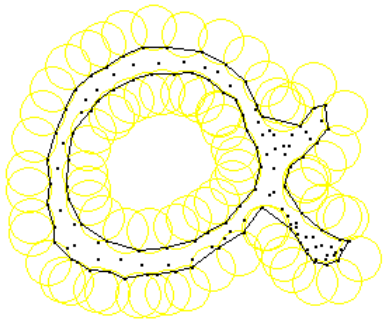

Figure 2: An example of α-shape

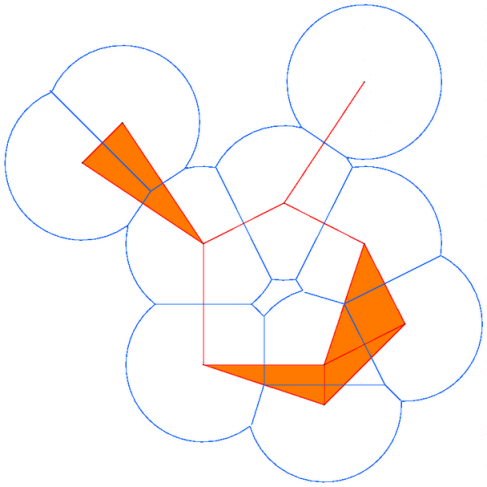

Figure 3: An example of weighted α-shape for a union of balls. Copied from [20]

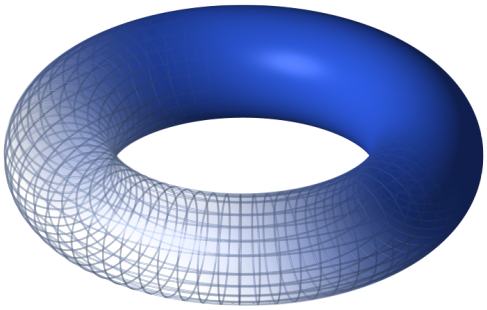

Figure 4: The torus has one connected component(β_0), two circular holes (β_1,the one in the center and the one in the middle of the "donut"), and one two-dimensional void (β_2, the inside of the "donut"). Copied from [25]

is

$$\beta_1 = m - n + k \tag{1}$$

Proof of this equation can be found in [26].

H. Edelsbrunner introduced a way to compute *n-th* Betti number in [27], where he uses gaussian elimination to compute the Betti numbers with adjacency matrix.

V. Robins [28] provided a computational technique for computing Betti numbers with detailed explains and proofs. What's more, she put more thoughts on computing Betti numbers for alpha shapes.

3 Our Method

Traditionally, sampling based algorithms will build a graph for configuration space. Some of these graphs, such as *PRM** and *RRT** [7], are proved to be able to find near optimal path, but can hardly be used to analysis the topology of paths. We want to to build a stronger representation of *C-space* that is both able to cover optimal path and study the topological structure. In this section, we are trying to solve the problem in 2D and answer the question "Is two paths in the same homotopy class" as a start point of using our method to study the topology of *C-space*. In this section, we will first give the outline of the algorithm, and explain each step in detail.

In the rest of the section, we will use the term "configuration" to refer to a point in configurations space. Ball $B(x, \rho(x))$ is referred to as a "*sample*"

1: **function** RICHCSPACEREPRESENT(*C-space*, number)
    // Phase One
2: $randomConfigs \leftarrow SampleRandomConfigurations(C\text{-}space, number)$
    // Phase Two
3: $maSamples \leftarrow SampleOnMedialAxis(randomConfigs)$
4: $regSamples \leftarrow SampleLeftArea(maSamples, randomConfigs)$
    // Phase Three
5: $dualShape \leftarrow WeightedAlphaShape(maSmaples)$
6: $topologyRoadmap \leftarrow Retract(dualShape)$
    // Phase Four
7: $brokenRoads \leftarrow BreakGraph(topologyRoadmap)$

8: **end function**

Figure 5: Algorithm Outline. *C-space* here means the metric space (X, ρ) where we can know if a configuration is feasible and its clearance to nearest obstacles.

in the rest of the section.

A *The Algorithm outline*

The algorithm is shown in Figure 5. It has four phases. The first phase is to generate some random configurations in *C-space*, like all other sampling-based planning algorithms do, and keep only those feasible ones. Every feasible configuration will be kept until the third phase. ($randomConfig$ keep unchanged.) In the second phase, we will sample balls centered on medial axis first and then try to cover the left area of C_{free}. Samples on the medial axis $MA(C_{free})$ will be used in phase three to construct the dual shape that captures the topology of C_{free}. After simplifying the dual shape, we will have a roadmap while maintaining the topological structure. Phase four breaks the topology roadmap into pieces such that no two piece can be merged into a new graph without forming a loop.

We will now explain each phase in detail with demos.

B *Sampling in C-space*

As pointed out in Figure 5, this phase has two parts: Sampling on medial axis, and sampling the rest of free space.

Sampling on medial axis is well studied in [18] and [15]. Besides the

positions of configurations, we also keep the clearance of each configuration, thus have some balls, or samples. We also require that no new ball, when sampling, is allowed to be centered inside any existing ones. For details of the algorithms, please refer to these two papers. One thing worth our notice is that using only algorithm mentioned in [18], we can get good result in narrow corridor but are more likely to ignore large open area. While using only [15] method, we can cover large open area well, but more likely to produce disconnected balls in narrow corridors. Therefore, running both algorithms is highly recommended.

The reason we sample on medial axis first is that 1.these balls are maximal ones as the definition of medial axis suggests, thus may reduce the number of samples we get, and 2.medial axis $MA(C_{free})$ is a SDR of *C-space*. [19]

Figure 6 describes how we sample area not yet covered by balls centered on medial axis. The basic idea behind the algorithm is to keep sampling new balls on the boundary of existing ones by "pushing" random configurations inside existing balls to the boundary of them and determine if they are in any other balls. If not, we sample a new ball. We restrict that all balls to be larger than a minimal radius ball, thus void getting a lot of very small samples.

Getting configurations inside a ball doesn't have to be $O(n)$ costly. It can be done using KNN algorithm, which is more efficient than iterating everything. Since we generated random configurations n times(not all are feasible), in a 2D space with area A. We expect to have n/A configurations in an unit area. A ball with radius r has area πr^2, approximately

$$k = n \cdot \pi r^2 / A \tag{2}$$

configurations are inside the ball. We then project these (uniformly distributed) configurations to the boundary of the ball to sample new spheres. Actually, choosing a larger k could be better: we virtually extend the radius of our discs, which results in a smaller density in arcs that are very close to obstacles than those far from obstacles. Samples got in this part are called *regular samples*.

Figure 7 is a result of this phase running in a rectangle *C-space* with 3 obstacles in the middle. Gray balls are centered on medial axis, while green ones are not. Red points are ball centers.

C *Constructing Topology Roadmap of C-space*

After getting these samples, we can now start to build a structure that captures the topology of the space.

```
1: function SampleLeftArea( maSamples, randomConfigs )
2:     regSamples ← EmptySet                          ▷ Set of balls not on MA
3:     newBall ← True                                 ▷ If we can get new balls
4:     lastRoundBalls ← maSamples
5:     while newBall do
6:         newBall ← False
7:         bndConfigs ← EmptySet          ▷ A set of ball boundary configs
8:         for ball ∈ lastRoundBalls do
9:             innerConfigs ← KNearest(ball, randomConfigs);  ▷ KNN
10:            bndConfigs.append(getBoundaryConfigs(ball, innerConfigs))
11:        end for
12:        lastRoundBalls ← EmptySet
13:        for config ∈ bndConfigs do
14:            if config not inside any existing ball then      ▷ O(n^2) time
15:                ball ← getNewBall(config)                  ▷ get a new ball
16:                if ball.radius ≥ min_radius then
17:                    newBall ← True
18:                    lastRoundBalls.append(ball)
19:                    regSamples.append(ball);
20:                end if
21:            end if
22:        end for
23:    end while
24:    return regSamples
25: end function
```

Figure 6: Algorithm to sample left free area

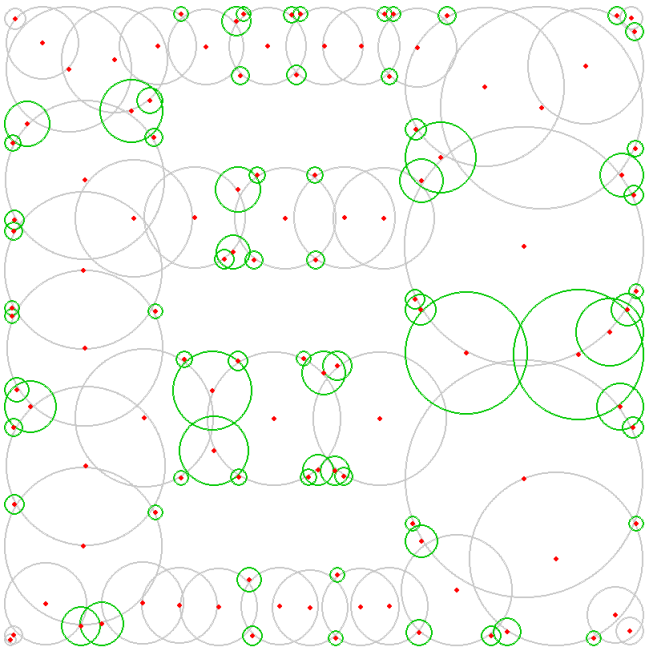

Figure 7: Samples got by the algorithm. We have 43 medial axis samples(gray balls), and 76 regular samples(green balls).

This phase, like the previous one, has two parts: 1. Construct the dual shape of medial axis samples got from phase one. 2. Simplify the dual shape to a simple graph.

In the first part, constructing the dual shape using only the medial axis sample could break the topology of medial axis. The reason is shown in Figure 8: three balls are all centered on medial axis, but their dual shape will be an empty triangle, which introduces a "hole" in that area. To solve this problem we can try to sample a ball centered at the radical center after getting an empty triangle, and rebuild the dual shape for these four balls[23]. A simple proof: if there is no obstacles in that area, the newly sampled ball will connect three balls, if there is obstacle, the ball will either not be sampled or not connect three of the balls.

With these balls, we then compute their dual shape, which is formed by some 1-simplices and 2-simplices in 2D. The runtime of computing dual shape will be $O(n^2)$ which is the worst case when doing regular triangulation of weighted points. Although using all balls will avoid problem as in Figure 8, we are making n much larger, which results in a bad runtime of this step. (However, constructing the dual shape of all balls will be helpful to study the shape of *C-space*)

One might argue that we can simply get configurations on medial axis and connect them as a roadmap. However, this roadmap doesn't maintain the topology of *C-space*. At some corners, we will still get triangles which don't have information if there are obstacles inside the triangle. By building the dual shape, this will no longer be a problem.

The second part of this phase can be stated as: retract all connected 2-simplices into one point, reconnect these retracted points with other ones using their previous connection. By doing so, we got a simple graph that characterize the topology of *C-space*. We would like to call it *Topology Roadmap*. This roadmap can also be used to determine if paths exist between two configurations. An example of such roadmap is shown in Figure 10. Filled triangles, along with large balls, are suggesting some large open areas, which is a problem [3] is trying to address.

This is not the only method of building the topology roadmap of *C-space*, another option will be to build a visibility roadmap, like [8], but using only ball centers as roadmap nodes. One might come up with other heuristics to achieve the same thing. Due to the limited number of balls, any algorithms can have a good performance.

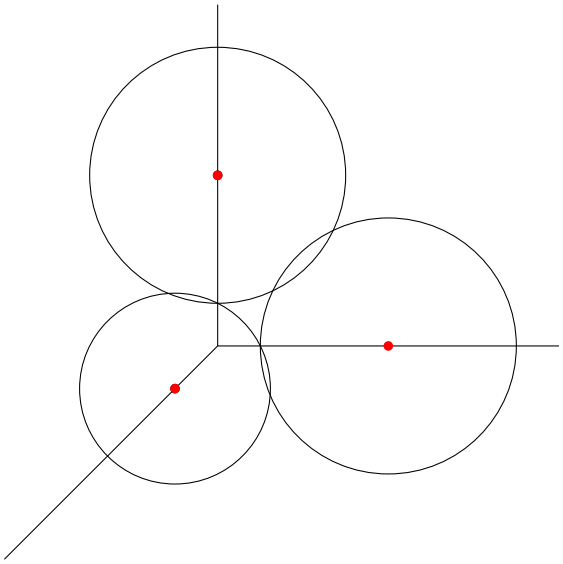

Figure 8:
An example that medial axis samples break the topology structure of the medial axis.

D *Breaking A Graph into Loop-free Parts*

After getting the topology roadmap, we technically already have a planner, but still we can simplify it. We now discuss breaking a graph into several components, such that no two components can merge into a new graph without forming a loop. By doing this, a graph is even simplified. Knowing which components are the start and goal configurations in, we can find a path connecting these components immediately. This technique can be a great improve when *C-space* is very large and traditional *PRM* or *RRT* methods will take a long time to solve.

Let g_1 and g_2 be two graphs without loops. We say g_1 and g_2 to be *mergeable* if: 1. they share at least one common node, 2. the new graph formed by these two graph has no loop.

Determine if a simple graph has a loop is very easy using the first Betti number β_1 mentioned in section 2, equation (1).

Figure 11 shows our algorithm to break a graph. The idea behind this algorithm is to take every edges as an initial component, then iterate each component to find and merge with a mergeable component until there is no two mergeable components.

Result of breaking Figure 10 can be seen in Figure 13.

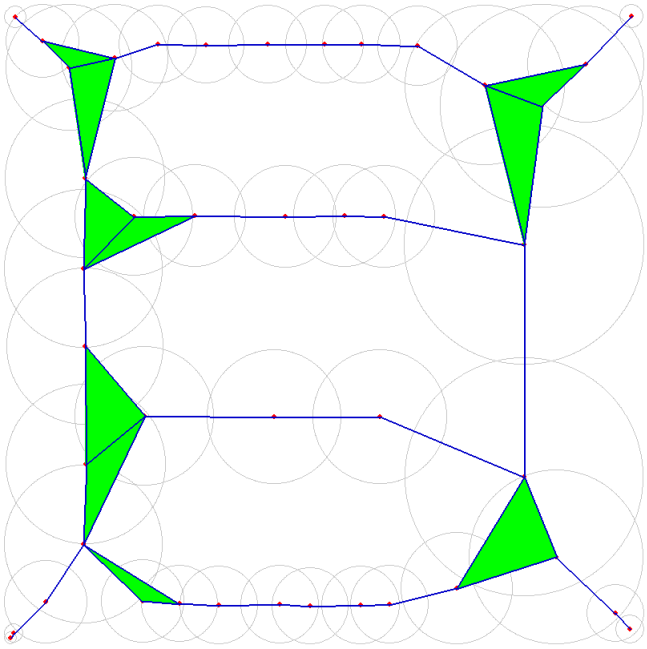

Figure 9:
Dual shape of balls generated in phase 1 (centered on medial axis). Filled triangles, along with large balls, are suggesting some large open areas.

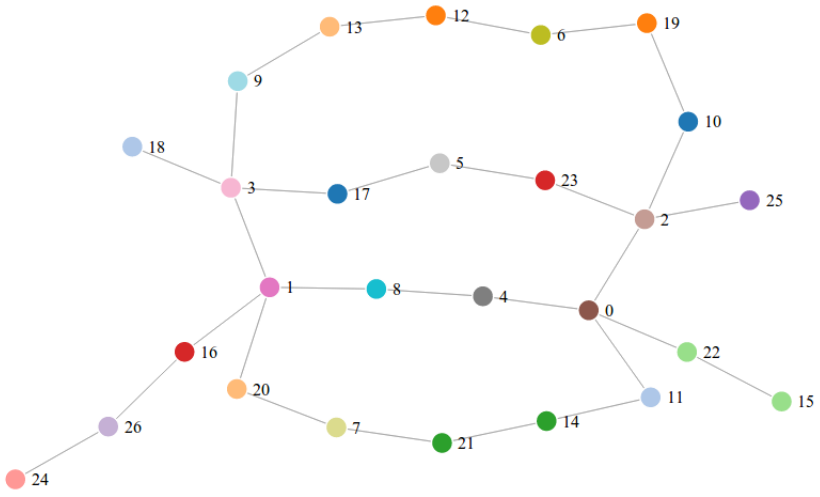

Figure 10:
Simplified graph of the dual shape in Figure 9.

```
1: function BreakGraph( graph )
2:     components ← EmptySet
3:     for edge ∈ graph.edges do
4:         components.append( new Graph(edge) )
5:     end for
6:     return MergeComponents( components )
7: end function
1: function MergeComponents(components)
2:     pq ← PriorityQueue()
3:     for component ∈ components do
4:         pq.push(component, component.size())
5:     end for
6:     while pq.size() > 1 do
7:         smallest_component ← pq.pop()
8:         mergeable_component ← find first mergeable component in pq
9:         new_component ← smallest_component.merge_with(mergeable_component)
10:        components.remove(smallest_component)
11:        components.remove(mergeable_component)
12:        pq.remove(mergeable_component)
13:        components.append(new_component)
14:        pq.push(new_component, new_component.size())
15:    end while
16:    return components
17: end function
```

Figure 11:
Breaking a graph into several loop-free components.

By breaking a graph into several parts, we virtually break the space into several subspaces. In this sense, our method can be viewed as a space partition strategy.

E *Determine Paths in the Same Homotopy Class*

A continuous function, or a map, $f : X \to Y$ is called a *homotopy equivalence* if there is a map $g : X \to Y$ such that there exist a continuous function $H : X \times [0, 1] \to Y$ with $H(x, 0) = f$ and $H(x, 1) = g$.

Besides using those components to plan motions for a large *C-space*, they can also be used to determine if two paths are in the same homotopy class.

Let σ_1 and σ_2 be two paths, with the same start and goal configuration,

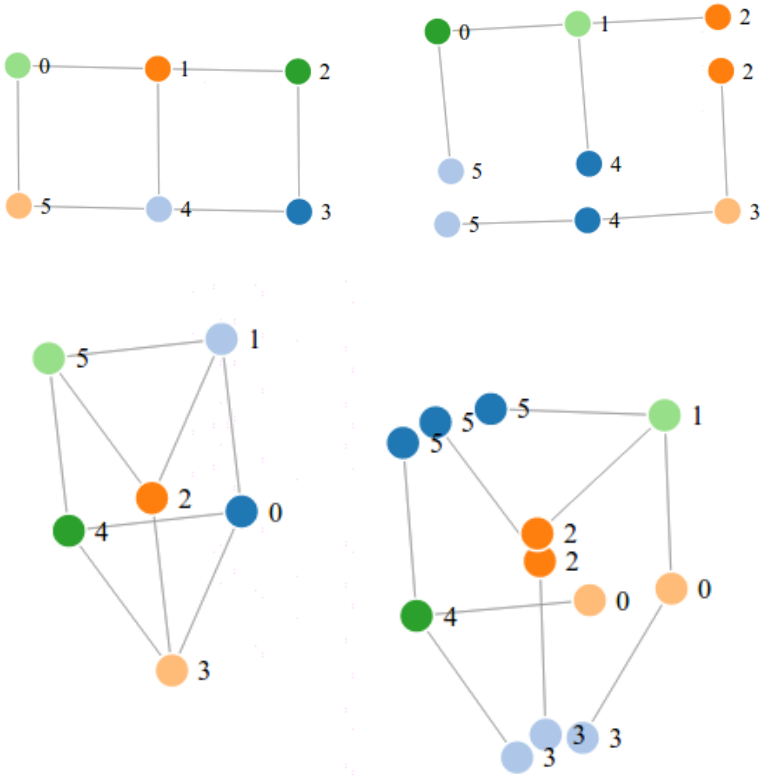

Figure 12: Some examples of breaking a graph. Graphs are visualized using *d3.js* (*http* : *//www.d3js.org*)

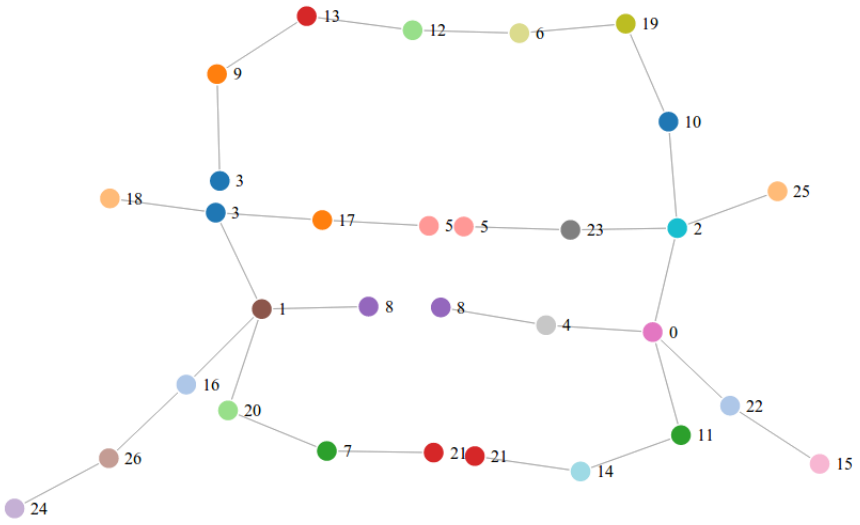

Figure 13:
Broken graph of Figure 10. The graph is divided two parts. Merging these two components will create loop.

going through from components g_1 to g_2 via two nodes A and B. σ_1 and σ_2 are homotopy equivalent if $A = B$.

Proof can be found in section 4.

4 Good Properties

This section should be re-organized as below:

4.1 Coverage Analysis

Theorem 4.1 *Let S_{init} be the set of initial random configurations. With minimum radius of r_{min}, configurations with clearance larger than r_{min} will be covered, when $|S_{init}| \to \infty$.*

Proof: When the number of initial random configurations approaches infinite, the boundaries of all existing balls are fully covered. So, every configuration not inside any existing balls will be sampled as a ball, unless it's less than r_{min} clearance.

As we can see, we can analyze the coverage of the space. Even more, we can explicitly describe the coverage of each ball in the algorithm. Using this property, we can find near optimal path. We will show some experiments in section 5.

Theorem 4.2 *Let $b \in B$ be an element of B, A and B are two configurations in* C-space. *The optimal path σ between A and B is in $\bigcup_{b \in B}$ if σ has clearance no less than r_min.*

Proof This is a direct conclusion of previous theorem.

4.2 Low Memory Representation

Theorem 4.3 *Let B be the set of ball samples got in phase one. Then $|B|$ is finite.*

Proof: We restrict balls to have no less than r_{min} radius and new balls to be centered outside existing ones. Assume we partition the space into $r_{min} \times r_{min}$ cells in 2D. The number of cells is finite, if the space is finite. Putting balls with exactly r_{min} radius in the centers of these cells will cover the space totally. The number of balls is finite. Phase one will generate balls no more than the number of balls generated in this way. Figure 9 shows the dual shape of balls generated by previous phase.

In practice, most balls are much larger than minimum radius. So the actually number of balls is very limited. We will see, in section 5, the number of balls required to cover a space tends to be stable.

4.3 Analysis of Topology

In this subsection, we will see how to use the structure built by our algorithm to determine path topology.

Theorem 4.4 *Any path σ can be continuously deformed onto medial axis.*

Proof This is the definition of Medial Axis $MA(C_{free})$ being a strong deformation retraction of C_{free}.

Given the theorem that any path can be deformed onto medial axis, we now use medial axis samples to determine if two paths have the same homotopy type.

Theorem 4.5 *Given path σ_1 and σ_2, we say they have the same homotopy type if their image in the topology roadmap is the same.*

Proof Let x be a point in the space X, $B = \{b_1, b_2, ..., b_n\}$ be the set of balls centered on medial axis. Define map $H(x)$:

$$H(x) = b_i \quad , if \quad |x - b| \leq |x - b'|, b' \in B \tag{3}$$

This continuous function maps any points in the space to balls centered on medial axis. While every point in the balls is mapped to the dual shape form by filled triangles, nodes and edges. Filled triangles are mapped to single nodes.

Therefore, if two paths have the same image in the topology roadmap. They have the same homotopy type.

4.4 Fast Planning

By breaking a graph into several components, we virtually break the space into several subspaces. Every subspace is managed by a tree, and trees are connected with shared common nodes. Every node in a tree represents a cell of power-diagram, as suggested by equation (3).

Consider each component as a graph node, connecting them will form a graph. A fast path searching algorithm will be: 1. search path between components, then 2. search path in each component.

In the worst case of searching a graph, the time is

fdaaaaaaaaaaaaaaaaaaaaaaaaagfdasgdsfagfdsafdsagdsafdsf

5 Experiments

We now describe some experiments and their results. We are planning for a point robot in workspace shown in Figure 1. Thus the workspace is the same as configuration space.

5.1 Coverage of the space

Figure 14 shows the relation between the number of initial configurations from Phase 1 and the number balls got in Phase 2. X axis ranges from 100 - 6000 with 100 increment as the number of initial configurations. Every setting runs 8 times. The number of balls in the Y axis is the average value of 8 results for each setting. With the increase of initial random configurations, the number of balls tends to be 120.

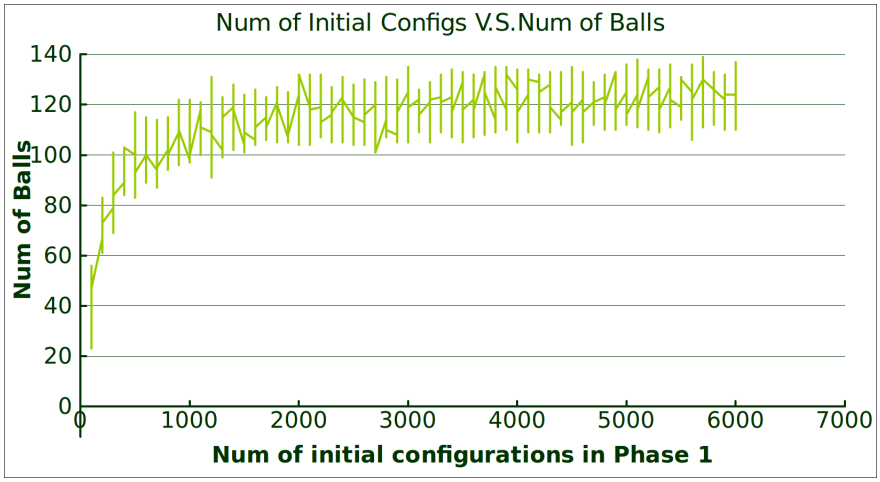

Figure 14:
Relation between number of initial configurations and balls. The number of balls tends to be around 120.

Figure 1 is a 800×800 workspace for a point robot. By analyzing each pixel, we can know whether it is covered by balls and get its distance from obstacle. Figure 15 shows the relation between the number of initial configurations, number of pixels uncovered by balls and their distance from obstacles. The Y axis ranges from 100-7000 with 200 increment as numbers of initial random configurations in Phase 1. X axis indicates the distance from obstacles. And Z axis is the number of pixels uncovered by balls. The minimum radius of balls is 10. As we can see, with the increase of initial configurations, the number of uncovered pixels drops sharply. The maximum distance uncovered pixel also decrease very quickly.

5.2 Searching for Optimal Path

Balls generated in Phase 2 can be used to find near optimal paths using $A*$ algorithm. We are virtually building a graph by discretizing balls to boundary configurations. Every configuration is "connected" to boundary configurations of the ball it is in. The difference between a *PRM* is that we are not storing any connection information between configurations explicitly.

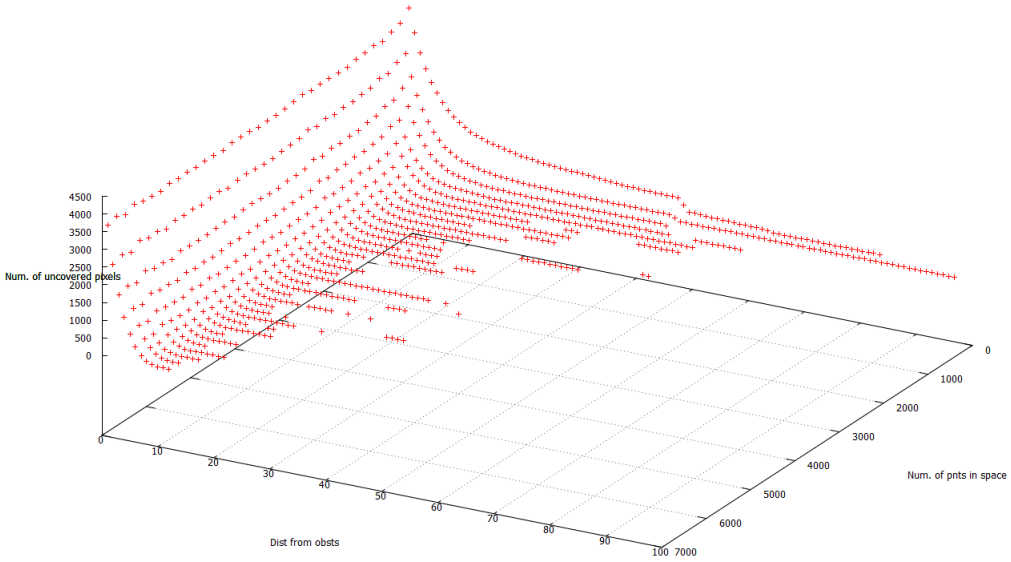

Figure 15:
Relation between number of initial configurations, uncovered pixels and their distance from obstacles.

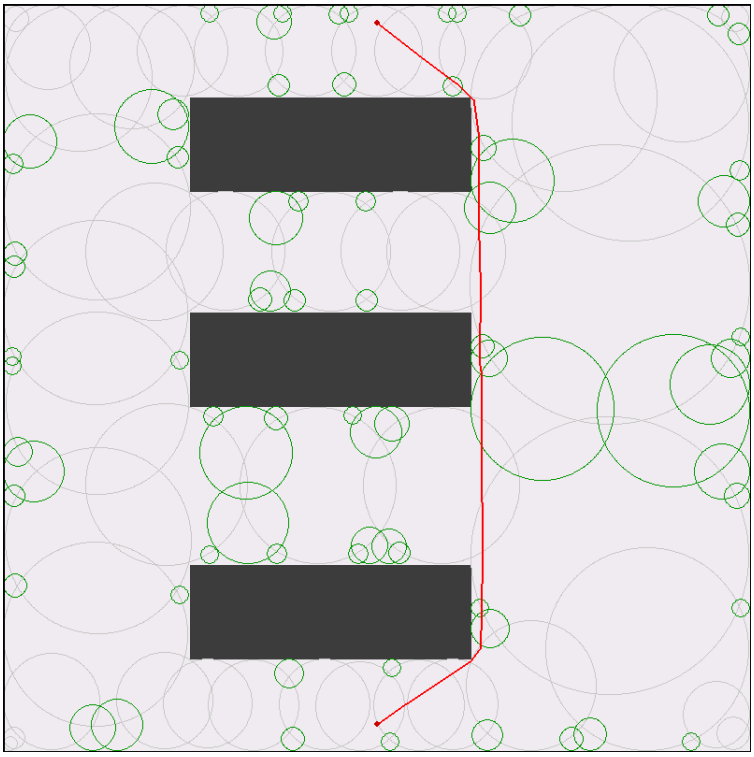

Figure 16:
Using $A*$ search algorithm to find an optimal path. We discretize balls to configurations on their boundaries. Thus, successors of a configuration are other boundary configurations of the ball current configuration is in.

6 Conclusions and Future Work

The results demonstrated with our algorithm are sound for covering the *C-space* and represent the topology.

However, we don't know what kind of issues there will be when implementing for high dimensional space. The time cost for constructing weighted α-shape(dual shape) of unions of balls is a concern. Because we need to check each *k-simplex* while $k = 1, 2, 3, ..., d$, d being the dimensions. So, our next goal is to implement the algorithm for 3D case, and then higher dimensions.

Our next direction is to study the properties and applications of weighted α-shape, especially applications in bioinformatics and chemistry, then extend to high dimensions.

Another direction will be studying the topology using the topology roadmap we constructed. Its value is underestimated in this paper, and I believe we can find more there.

References

[1] Weifu Wang, Devin Balkcom, Amit Chakrabarti, "A fast streaming spanner algorithm for incrementally constructing sparse roadmaps", IROS 2013.

[2] Marble J and Bekris KE. "Asymptotically near-optimal planning with probabilistic roadmap spanners", IEEE Transactions on Robotics, 29(3), 2013.

[3] Robin L H Deits and Russ Tedrake. "Computing large convex regions of obstacle-free space through semidefinite programming". In Proceedings of the Eleventh International Workshop on the Algorithmic Foundations of Robotics (WAFR 2014), Istanbul, 2014.

[4] Lindemann, Stephen R., and Steven M. LaValle. "Current issues in sampling-based motion planning." Robotics Research. Springer Berlin Heidelberg, 2005. 36-54.

[5] Bhattacharya, Subhrajit, Maxim Likhachev, and Vijay Kumar. "Topological constraints in search-based robot path planning." Autonomous Robots 33.3 (2012): 273-290.

[6] Bhattacharya, Subhrajit, Vijay Kumar, and Maxim Likhachev. "Search-based path planning with homotopy class constraints." Third Annual Symposium on Combinatorial Search. 2010.

[7] Sertac Karaman, Emilio Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning", International Journal of Robotic Research - IJRR , vol. abs/1105.1, no. 7, pp. 846-894, 2011.

[8] Thierry Simon, Jean-paul Laumond, Carole Nissoux, "Visibility-based probabilistic roadmaps for motion planning", Advanced Robotics , vol. 14, no. 6, pp. 477-493, 2000.

[9] J.-M. Lien, O. B. Bayazit, R.-T. Sowell, S. Rodriguez, and N. M. Amato. "Shepherding behaviors". In Proc. IEEE Int. Conf. Robot.Autom. (ICRA), pages 41594164, April 2004.

[10] O. B. Bayazit, G. Song, and N. M. Amato. "Enhancing randomized motion planners: Exploring with haptic hints". In Proc. IEEE Int. Conf. Robot. Autom. (ICRA), pages 529536, 2000.

[11] A. P. Singh, J.-C. Latombe, and D. L. Brutlag. "A motion planning approach to flexible ligand binding". In Int. Conf. on Intelligent Systems for Molecular Biology (ISMB), pages 252261, 1999.

[12] Delanoue, N.; Jaulin, L.; Cottenceau, B. (2006). "Counting the Number of Connected Components of a Set and Its Application to Robotics". Applied Parallel Computing, Lecture Notes in Computer Science 3732 (1).

[13] Jaulin, L. (2001). "Path planning using intervals and graphs". Reliable Computing 7 (1).

[14] Steven M. LaValle, "Planning Algorithms", Cambridge University Press, ISBN 0-521-86205-1, 2006.

[15] Yeh, Hsin-Yi Cindy, et al. "UMAPRM: Uniformly Sampling the Medial Axis."

[16] LaValle, Steven M. "Rapidly-Exploring Random Trees A New Tool for Path Planning." (1998).

[17] L. E. Kavraki, P. Svestka, L. E. K. P. Vestka, J. claude Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces" IEEE Trans. Robot. Autom., vol. 12, pp. 566580, 1996.

[18] Wilmarth, Steven A., Nancy M. Amato, and Peter F. Stiller. "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space." Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 2. IEEE, 1999.

[19] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space". Technical Report TR98-022, Department of Computer Science, Texas A&M University, College Station, TX, Nov. 1998.

[20] Edelsbrunner, Herbert. "The union of balls and its dual shape." Proceedings of the ninth annual symposium on Computational geometry. ACM, 1993.

[21] "Alpha Shape Introduction", Duck University, ($http : //biogeometry.duke.edu/software/alphashapes/$)

[22] Edelsbrunner, Herbert; Kirkpatrick, David G.; Seidel, Raimund (1983), "On the shape of a set of points in the plane", IEEE Transactions on Information Theory 29 (4): 551559, doi:10.1109/TIT.1983.1056714.

[23] "Radical Center", From Wolfram MathWorld – the web's most extensive mathematics resource. ($http : //mathworld.wolfram.com/RadicalCenter.html$)

[24] Z. McCarthy. Bretl, and S. Hutchinson, "Proving path non-existence using sampling and alpha shapes," in IEEE International Conference on Robotics and Automation (ICRA), May 2012.

[25] "Betti Number", From Wikipedia – The Free Encyclopedia. ($http : //en.wikipedia.org/wiki/Betti_number$)

[26] Gyula Karolyi, Ambrus Pal, "The cyclomatic number of connected graphs without solvable orbits", Periodica Mathematica Hungarica, 01/2007;

[27] Edelsbrunner, Herbert, and John Harer. "Computational topology: an introduction." American Mathematical Soc., 2010. page. 88 - 93.

[28] Robins, Vanessa. "Computational topology for point data: Betti numbers of -shapes." Morphology of Condensed Matter. Springer Berlin Heidelberg, 2002. 261-274.