Master's Thesis of Enginnering

Computation of Configuration Space and Voronoi Structure for Planar Moving Object with 3DOF

평면 상에서 움직이는 3자유도 물체의

Configuration 공간과 Voronoi 구조의 계산

AUGUST 2021

Graduate School of Engineering

Seoul National University

Computer Science and Engineering Major

MinGyu Jung

Computation of Configuration Space and Voronoi Structure for Planar Moving Object with 3DOF

Advisor Myung-Soo Kim

Submitting a master's thesis of Engineering

August 2021

Graduate School of Engineering
Seoul National University
Computer Science and Engineering Major

MinGyu Jung

Confirming the master's thesis written by
MinGyu Jung
August 2021

Abstract

We present efficient algorithms for computing configuration space obstacles and Voronoi structures for a planar moving object bounded by arc−splines. This approach is based on simple geometric properties of circular arcs. Moving objects and obstacles represented with Bezier curves or B−splines curves can be converted to arc−spline models using biarc approximation. For the computation of configuration space obstacle, we can derive the exact equation and the parameter domain of contact surfaces from the boundary of configuration space obstacle. For the computation of Voronoi structure, we first compute the Voronoi diagrams for some fixed rotations and then stack them up to form a network of surface patches. Using the constructed network of surfaces, we generate a maximum clearance path by following the path on the Voronoi surface. The path can be further optimized based on various different criteria depending on specific applications.

주요어 : Configuration space, Minkowski sum, Voronoi structure, medial axis, motion planning, circular arc
학　번 : 2019−23686

Table of Contents

Table of Figures

Chapter 1

Introduction

1.1 Research Background

Motion planning is one of the most important research topics discussed in computer graphics, computer animation, and virtual reality [1] [2] [3]. The main goal is to find a collision−free path for a moving object when the environment with obstacles is known [4].

There are many different types of moving objects. Some move in a two−dimensional space, while others in 3D. Flexible robots may even rotate and deform their shapes with joints [5]. The specific type is a crucial aspect in motion planning. The problem complexity and the approach to the problem are dependent on the robot type. In this thesis, we focus on a planar motion that performs two−dimensional rigid body transformation with translation and rotation.

A two−dimensional rigid body's motion planning algorithms are applicable in many real−life problems. One of the most familiar examples that can be seen is the floor cleaning robot. The robot should not touch obstacles in the ground and move through places. Such a robot is confined to the ground—it cannot rise nor rotate with axes other than the vertical one. Therefore, such a robot can be seen as a two−dimensional rigid body robot. Another example that is commonly used in industry is a 2.5−axis CNC machining tool [6].

The cutter first changes its vertical level to work on and then only moves in that level with translation for a while. The machining tool should never collide with the parts of the material that is a part of the product. Another area of use is Amazon's transporting robots in the warehouse [7]. Some promising future areas of use include medical surgery, VR environment, etc.

There are many kinds of approaches to the motion planning problem. One of the most popular approaches is to convert the problem for volumetric robots with workspace obstacles to a point robot with configuration space obstacles [8], namely a set of configurations which make the robot collide with workspace obstacles. This approach simplifies the motion planning problem, as a collision test against a moving point becomes simpler than testing a moving volume against the surrounding environment. On the other hand, the difficulty is in computing the configuration space obstacles.

The computation becomes much more difficult when freeform objects are under consideration. Many previous results on motion planning with the configuration space approach employed polygonal approximation to the robot and the workspace obstacles [9]. While this choice of object representation greatly simplifies the computation of configuration space obstacles, it leads to some undesired side effects as a result. One of the most notable drawbacks is the loss in precision [10].

Another way we can represent objects is to use circular arcs as the basic primitive [11] [12]. This is the approach we take in this thesis. Compared with polygonal approximation, circular arcs can better represent the object as it has G^1−continuity and more degrees of freedom in curvature. Owing to these desirable geometric features, circular arc approximation is known to have cubic convergence [13].

This means that we can approximate a given model with a much smaller number of circular arcs compared with polygons, when a certain fixed level of precision is required.

After the computation of configuration space obstacles, we need to find a collision-free path for a moving point. There are many approaches toward this goal, including those such as roadmap method, cell decomposition, potential functions, etc [4].

Voronoi structures can be useful for all these methods in many ways by providing distance information. Points on a Voronoi structure form the so-called medial axis. The name comes from the fact that they have two or more closest points to the configuration space obstacles. Points on the medial axis contain distance information to the closest configuration space obstacle. The medial axis also works as the common boundary between adjacent Voronoi cells (consisting of points having the same closest configuration space obstacle).

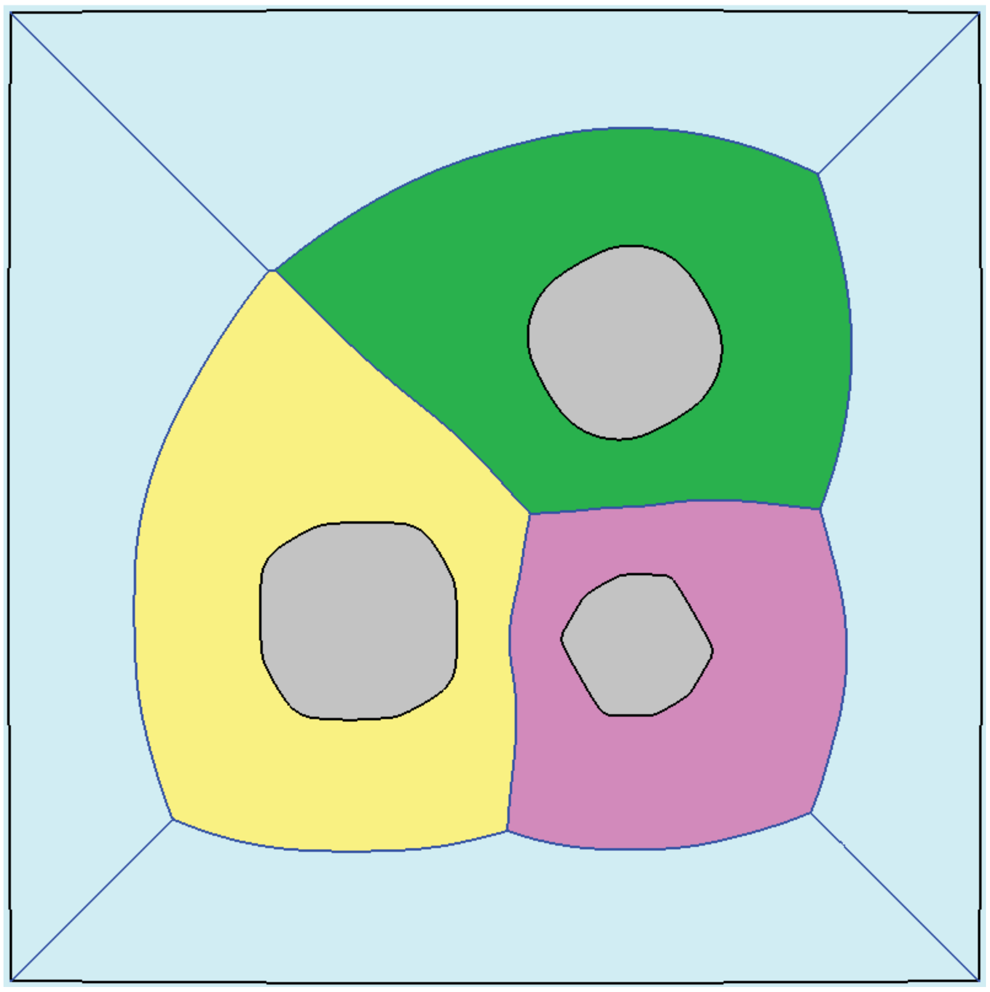

Figure 1 An example Voronoi Structure. The gray objects are configuration space obstacles. The blue curves are the medial axis. Each of the yellow, green, purple and sky blue areas represent a Voronoi Cell.

For the roadmap method, Voronoi structure itself can be a good roadmap and is commonly used. For the potential function method, the computation of distance to the nearest configuration space obstacles is often needed. For the cell decomposition method, Voronoi cells can be used as a good decomposition of the free space. When the object boundary is represented with a set of circular arcs, a set of ellipses or hyperbolas will form the Voronoi cell boundary. For other object boundary representations, such as polygons or B-splines, the medial axis curves have degrees higher than that of the input object boundary [10]. For circles, the bisector curves are also conics. This makes the boundary representation for Voronoi cells much easier and precise. Other ways to decompose the cells using Voronoi structures may be proposed, such as using the maximum touching disks which will be discussed later in this thesis, as the Voronoi structure contains rich information about the free space.

Finally, the configuration space of a two-dimensional rigid body motion is a three-dimensional space. In this case, the configuration space can be visualized, which can improve understanding the underlying structure of free space and facilitate further research in motion planning.

1.2 Main Contributions

In this thesis, the configuration space of a two-dimensional robot with rigid body motion is mainly discussed. To be specific, we compute the three-dimensional configuration space obstacles and their Voronoi structure.

We compute a set of three-dimensional surfaces that form a superset of the configuration space obstacle boundary. Some segments of the surface may not form the obstacle boundary. But these redundant segments all lie in the inner region of a configuration space obstacle. This property allows collision detection and visualization. For instance, when a moving point in free space tries to move along a trajectory that has collision, it will have collision with at least one of the computed surfaces. On the other hand, if such trajectory has no collision, it will not intersect the computed surfaces. This is because, the redundant segments all stay inside the configuration space obstacles and do not stick out to the free space.

An algorithm to compute Voronoi structures in the three-dimensional free space for the above configuration space obstacles is proposed. In the xyθ-space, points that have multiple closest points with a fixed θ value is computed. This means that slicing this Voronoi structure perpendicular to the θ-axis leads to a two-dimensional Voronoi structure.

What distinguishes this thesis from other related results is the choice of geometric primitives. We propose to use circular arcs for the representation of robots and obstacles. One of our goals is to propose a new computational framework for the arc-spline based motion planning.

1.3 Thesis Organization

The thesis is organized as follows. In Chapter 2, we briefly introduce background material to understand the configuration space approach and the roadmap approach using Voronoi structure. In the Chapter 3, we review other previous results related to topics such as motion planning, Minkowski sum, and Voronoi diagrams. Details of our algorithms are described in Chapter 4. After that, in Chapter 5, we demonstrate the effectiveness of our approach by showing some visual results of the successful construction for the configuration space and actual path of maximum clearance. Finally, we conclude this thesis.

Chapter 2

Preliminaries

2.1 Configuration Space Obstacles and Minkowski Sum

Motion planning is to find a collision−free path for a moving robot when the shape of robot and obstacles are known. There are some cases where the robot can be thought of as a point [14]. But in most of the cases, the robot usually takes up some area or volume. Directly testing whether the motion of a robot has collision with obstacles is not easy. To do so, we first need to compute the space that the robot sweeps in the motion. Then, we need to test whether the swept volume has an overlap with the obstacles. Repeating this process every time a motion is generated is quite costly.

A *configuration space* approach is a method that can make this problem relatively easy [8]. A *configuration* is a minimal set of variables needed to determine the positions of all the points that constitute the robot. The dimension of this set is called the *degrees of freedom* and the space that these variables make is referred to as the configuration space [4]. For instance, in this thesis we focus on a two−dimensional robot with translation and rotation. The corresponding configuration can be represented with 3 variables, namely (x, y, θ). The first two variables indicate the amount of translation, while the last one is the rotation. For other types of robots, more features such as joints or deformation add variables to the configuration [5].

At some configurations, the corresponding robot may have collision with the obstacles. A set of such configurations is called the *configuration space obstacles*. On the other hand, some configurations would be collision−free. A set of these configurations is called the *free space*.

Finding a collision−free path for the point in the configuration space is equivalent to the original problem. This simplifies the problem and makes the motion planning algorithms efficient, since the intersection tests on points or curves are much easier than that of testing overlaps among areas or volumes. On the other hand, the drawback of this approach is in the generation of the objects in the configuration space, which can be quite time−consuming.

Minkowski sum is a binary operation on two sets A and B. It is formally defined as follows [15]:

$$\mathrm{A} \oplus \mathrm{B} = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathrm{A}, \mathbf{b} \in \mathrm{B}\}.$$

The Minkowski difference can be defined in a similar way:

$$\mathrm{A} \ominus \mathrm{B} = \{\mathbf{a} - \mathbf{b} \mid \mathbf{a} \in \mathrm{A}, \mathbf{b} \in \mathrm{B}\}.$$

The relation between the two operations are as follows:

$$\mathrm{A} \ominus \mathrm{B} = \mathrm{A} \oplus (-\mathrm{B}), \text{where } (-\mathrm{B}) = \{-\mathbf{b} \mid \mathbf{b} \in \mathrm{B}\}.$$

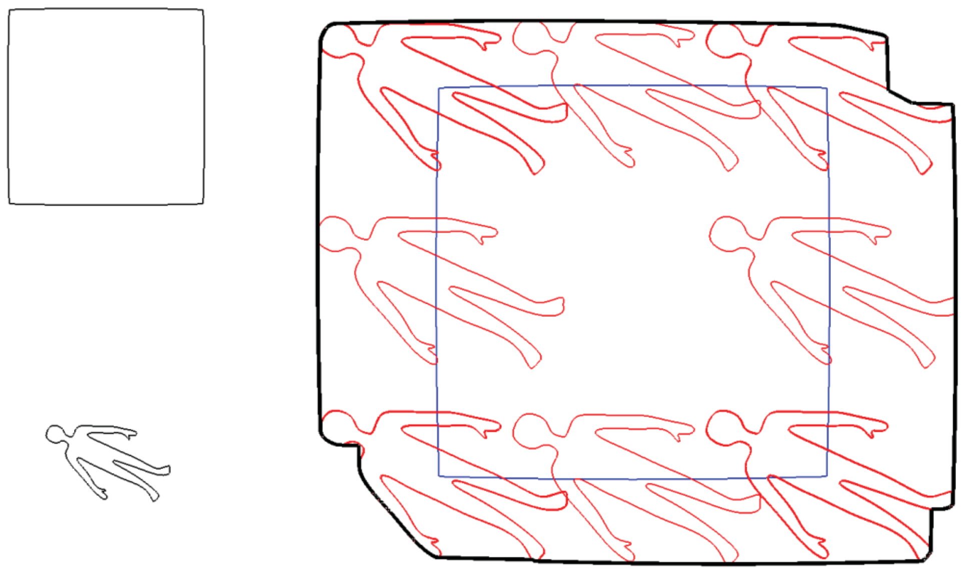

Figure 2 Example of Minkowski sum. Two objects on the left are inputs. The thick line on the right is the Minkowski sum boundary.

Minkowski sum can be used to compute a configuration space obstacle QO when the robot can translate with a fixed orientation. Given a robot A and an obstacle WO, the configuration space obstacle QO is given as follows:

$$QO = WO \oplus (-A).$$

When the robot at **t** and an obstacle WO have some intersection, an overlap point can be represented as either **w** $\in$ WO or **t** + **a** for **a** $\in$ A. Thus **t** = **w** - **a**. In conclusion, if there is intersection between the robot and an obstacle, **t** $\in WO \oplus (-A)$ holds. On the other hand, if there is no collision between the robot and an obstacle, it is impossible to choose a **a** $\in$ A and **w** $\in$ WO so that **t** = **w** - **a**. Thus, $QO = WO \oplus (-A)$.

This is possible only when the rotation θ is fixed. From now on, a subspace of the configuration space with a fixed θ value will be called a *slice*. The Minkowski sum lies in a two-dimensional slice, while the configuration space obstacle in the xyθ-space is three-dimensional. Therefore, we need further processing to stack those slices. This will be discussed in Chapter 4.

2.2 Voronoi Structure Computation

Given a set of objects in a space, the Voronoi structure is a decomposition of the space with distance information. The given space is decomposed into Voronoi cells, and the boundary of these cells form a medial axis [16]. *Voronoi cells* are a set of points that have the same closest object. Thus, the *medial axis* is a set of points which is equidistant to two or more objects [17].

Each point in the medial axis contains distance information to the closest object. Growing a circle from a point on the medial axis, it will stay collision-free until its radius reaches that distance. In reverse, we can start to grow a circle tangent to a point in an obstacle's boundary, until the circle touches another point in some obstacle's boundary. The center of such circle becomes an element of the medial axis. Such a circles is called the *maximum touching disk* [16], as it has the largest radius which does not make the inner region of the circle have contact with any obstacles. Circles that grew tangential to a point in the obstacle's boundary will be referred to as *touching disks* throughout this thesis.

With dense sampling and criteria to restrict approximation errors, many points on the medial axis can be found using the above method. But the difficulty of this approach comes into sight when we consider the correct topology [18], namely, how those points are connected and form the boundary curves between Voronoi cells. *Bifurcation point* is a point that has three or more closest points to the obstacle boundary. At that point, three or more medial axis curves meet and form a branch of the network. This complicates the topological structure of the medial axis.

One way to handle this problem is to choose certain subregions, in each of which there exists only one single branchless medial axis in it. It is already known that an area bounded by two monotone curvature curve segments has this important feature [16].

2.3 Object Representation and Biarc Approximation

There are many ways to represent an object. One simple way is to store some sampled points of the target object, like pixels. However, this approach is quite inappropriate for solving problems concerning high precision. Thus, in this thesis, we use boundaries of the connected components of the object.

The most common tools to represent boundaries are the Bezier curve and B-spline curves. Both of them are defined with control points, which is quite intuitive for designers. This led to their popular use in the current industry.

But directly using these curves has some difficulties. The Minkowski sum and the Voronoi structure generally have higher degrees than that of the input curves [10]. The computation is known to be notoriously hard, and the results have poor precision even if approximated.

A circular arc is another geometric primitive, which is efficient enough while well approximating these frequently used curves. Unlike Bezier or B-spline curves, the result of Minkowski sum and the Voronoi structure can be represented using conics.

Although the circular arc is not a much-used primitive compared to Bezier or B-spline curves, Biarc approximation [11] allows the transformation from those curves to arc−splines, a set of circular arcs. Biarc approximation recursively breakdowns a given curve into two G^1−continuous arcs. The tangents at the endpoints are also preserved, leading to the preservation of G^1−continuity of the whole object. The recursive process is known to have a cubic convergence. After each breakdown process, the error between the original object and the approximation is cut to asymptotically 1/8. This leads to a small number of total circular arcs, which generally leads to a faster computation time.

Chapter 3

Related Work

Lozano-Perez first introduced the notion of configuration space approach in motion planning [8]. The Minkowski sums of polytope objects were computed to plan collision-free a path for a moving object that can only translate. He also proposed a method to compute configuration space obstacles for a polygonal rigid body in 2D. Later, he further extended his work to visibility graphs [9], Where motions with translation and rotation are generated by connecting 2D slices, instead of computing the exact configuration space obstacles with full three-dimensions.

Han et al. [15] considered a planar object as a union of balls and represented the boundary using arc-splines. A sampling of these balls forms a subset of the inner region, therefore convolution arc-splines that fall into this region were trimmed. To accelerate this process, grid and cache data structures were used. Lee et al. [10] provide various approximation methods for the Minkowski sum boundary of two planar rational curves.

The Minkowski sum of three-dimensional B-spline surface was constructed by Mizrahi et al [19]. Surfaces that are neither convex nor C^1-continuous are allowed as inputs to the algorithm.

Blum first introduced the notion of medial axis transformation [17]. Multiple equivalent definitions and properties of the structure were given. Some important properties of the medial axis transformation were discovered by Choi et al. [20]. One of the most important properties is domain decomposition which allows the given problem to be divided into smaller subproblems. Many recent results including this thesis are based on this property as it stabilizes computation.

Medial axis for polynomial spline curves was computed by Aichholzer et al. [18]. Zhu et al. [21] approximate the medial axis of an arbitrary planar object with B−spline curves. The algorithm applies trimming to approximated branches that have Hausdorff distance larger than some user defined value.

Lee et al. [16] computed bisectors between inflection−free monotone B−spline segments in 2D. Unlike arc−splines, finding the maximum touching disk of B−spline models is not easy. Möbius transformation was applied to efficiently search maximum touching disks between B−spline curves.

There have been many approaches to directly compute the Voronoi diagram in 3D space. Culver et al. [22] algebraically derived the exact medial axis for polyhedron models. For this case, they showed that the medial axis surface segments are quadric surfaces, planes, or lines. They also showed that the seam curves between these segments can be lines, conics, or quartic curves. Musuvathy et al. [23] computed three−dimensional medial axis for a region bounded by C^4−continuous B−spline surfaces. The medial axis surfaces were computed by offsetting the boundary and computing self−intersections.

A parallelizable algorithm for medial axis computation of both planar and volumetric models was proposed by Lee et al. [24].The algorithm first divides the point cloud into subsets and then fits the balls so that they become maximum touching balls.

A method to compute arc−spline approximation of planar curves in CNC machining was proposed by Meek et al [11]. The arc−splines were constructed by connecting series of biarcs. Biarc is a set of two circular arcs which share endpoints with G^1−continuity [12].

There are other approaches toward motion planning besides the configuration space approach. For instance, probabilistic approaches are frequently used. Qureshi et al. [25] proposed Motion Planning Networks (MPNet) which is based on neural networks. The algorithm takes point clouds as input and recursively finds a midpoint for the path. Motion planning with reinforcement learning was proposed by Gao et al. [26]. The algorithm first finds path graphs in the scene. The path graph connects some points in the free space. Edges with collision are removed and then Q−learning is applied to find a path to the goal. Butyrev et al. [27] also proposed a reinforcement learning based approach toward motion planning. This approach takes non−holonomic constraints into account.

In this thesis, to verify the generated Voronoi structure, roadmap method was used. However, other methods such as cell decomposition or potential functions can be used with the configuration space approach [4]. The cell decomposition approach divides the free space into cells. Then a graph structure is generated by using the cells as vertices and the adjacencies of the cells as edges. A path in that graph is found and then refined. Potential function intuitively considers the objects to be electrically charged. The goal has attractive force while obstacles have repulsive force. This forms a surface of potentials and the robot slides down this surface. The method has a drawback. It might fall into local minimum and may not reach the goal.

Chapter 4

Geometry Construction Algorithms

4.1 3D Configuration Space Obstacle Computation

The robot can translate in two directions and rotate with its center. Therefore, the configuration of this robot will be represented with three-variables x, y, θ through the rest of this thesis. Also, the robot has a degree of freedom of 3.

Confining the robot into a slice of fixed θ restricts its movement to translation only. For this slice, the two-dimensional configuration space obstacle can be calculated with the Minkowski sum. Intuitively, the 3D configuration space obstacle can be made by stacking them in θ-direction. This is the approach taken.

For a point $\mathbf{p}$ from the Minkowski sum between two G^1-continuous curves A, B that form a loop, $\mathbf{p} \in \{\mathbf{a}+\mathbf{b} \mid \mathbf{a} \in \partial A, \mathbf{b} \in \partial B, \mathrm{normal}(\mathbf{a}) \parallel \mathrm{normal}(\mathbf{b})\}$ holds. In short, the Minkowski sum of curves is inside a set formed by a sum of points with parallel normal. Notice that such set is a super set of the Minkowski sum, but it has a good property that the segments to be trimmed lies in the inner region of it.

The robot or the obstacles may not be G^1−continuous, but we can think of those none G^1−continuous points to be a circle with radius 0 to solve this problem.

This thesis uses Biarc approximation to represent freeform objects. Therefore, all objects are defined with circular arcs. The normal of a point on a circular arc can be easily calculated by connecting its center with the point. Also, the change of normal direction is very simple. This greatly reduces the computational cost of normal matching.

We can calculate the whole superset of the Minkowski sum boundary, by a union of $C_A * C_B = \{\mathbf{a}+\mathbf{b} \mid \mathbf{a} \in C_A \subset \partial A, \mathbf{b} \in C_B \subset \partial B, \mathrm{normal}(\mathbf{a}) \parallel \mathrm{normal}(\mathbf{b})\}$. C_A and C_B are circular arcs. The process of finding $C_A * C_B$ is called the *arc convolution* of two circular Arcs C_A and C_B.

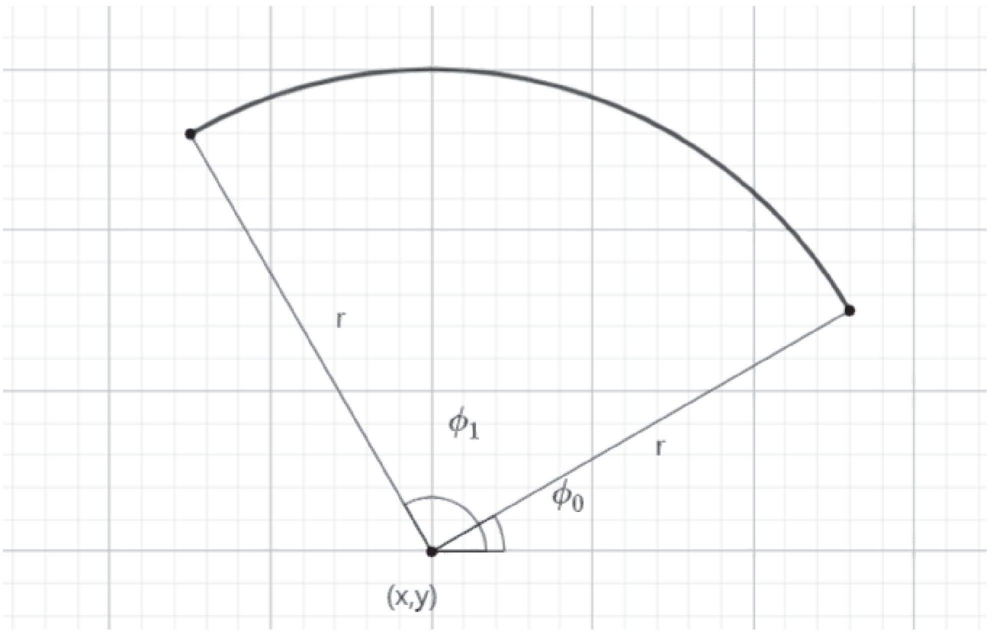

Figure 3 Definition of a circular arc

A random circular arc C in 2−dimensional space can be represented with 5 variables. To be specific, $C(x, y, r, \phi_0, \phi_1)$ represents a circular arc with its circle center at (x, y), with a radius r, and with parameter ϕ confined to an interval between ϕ_0 and ϕ_1. Here ϕ is a variable that parameterizes the circular arc. The value of ϕ is the angle between the two lines, the positive x-axis and the line connecting the circle center and the point corresponding to ϕ. In other words, for a point $C(\phi)$ on the circular arc C, below holds.

$$C(\phi) = (x, y) + r * (\cos\phi, \sin\phi)$$

Without loss of generality, we can always assume that the arc is parameterized in a counterclockwise manner. This means that we can assume $\phi_0 < \phi_1$.

We can always assume that $|\phi_0 - \phi_1| < \pi$, as we can always divide an arc with its central angle larger than π into two separate arcs. This assumption simplifies finding common intervals. Normally, two continuous intervals would have a single continuous interval. But angles have a periodic property, in other words, ϕ is equivalent to $\phi + 2\pi$. This could result into multiple common intervals, as can be seen in Figure 4. But, restricting the arc's central angle to be smaller than π resolves this problem. There are still multiple intervals that are not continuous, but they are congruent to 2π. This is illustrated in Figure 4.

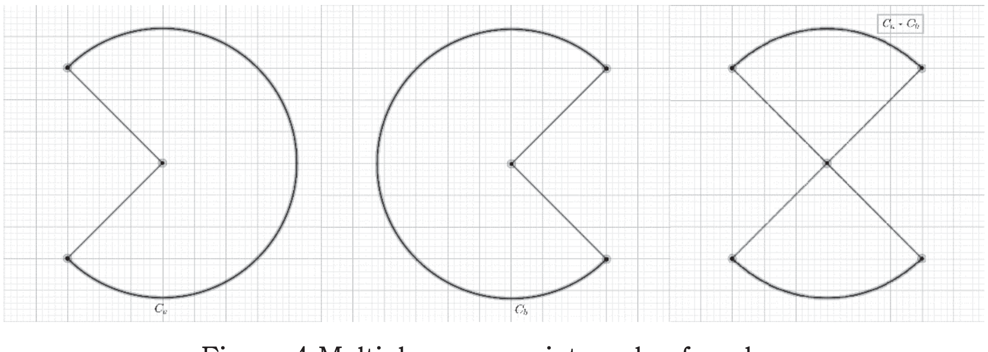

Figure 4 Multiple common intervals of angles.

The arc convolution of two arcs $C_0(x_0, y_0, r_0, \phi_{00}, \phi_{01})$ and $C_1(x_1, y_1, r_1, \phi_{10}, \phi_{11})$ results into two new arcs.

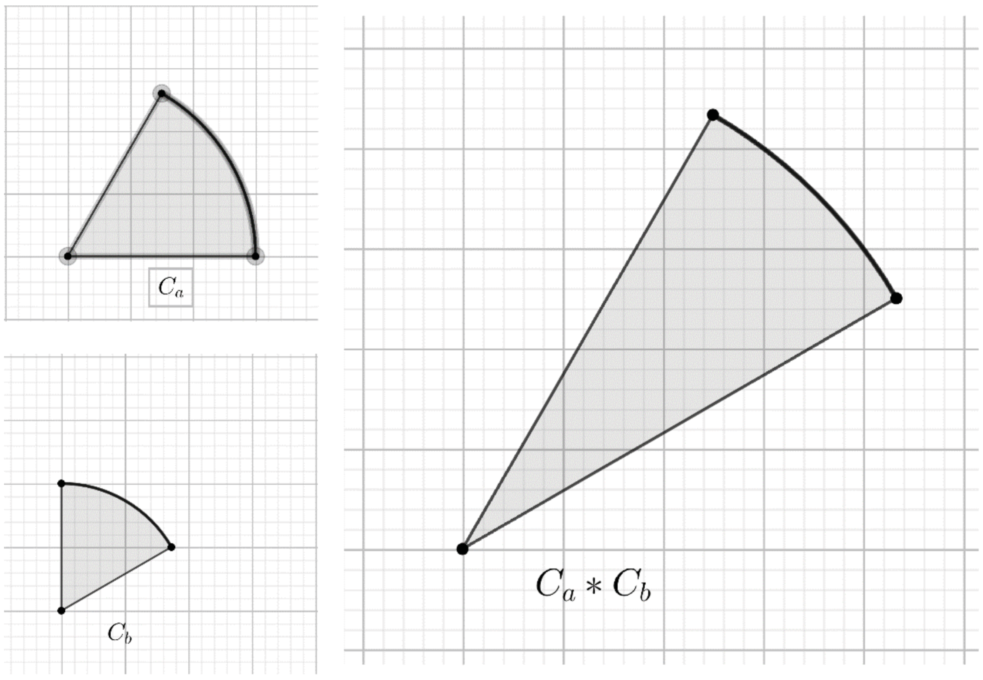

Figure 5 Arc Convolution, the first case

The first resultant arc is $C(x_0 + x_1, y_0 + y_1, r_0 + r_1, \phi_0, \phi_1)$, where $[\phi_0, \phi_1]$ denotes the common interval between the two intervals $[\phi_{00}, \phi_{01}]$ and $[\phi_{10}, \phi_{11}]$. This is the case where a point from C_0 and a point from C_1 have parallel normals since their ϕ value is in an equivalent class congruent to 2π. The Minkowski Sum of those two points will be as follows:

$$\begin{aligned} C_0(\phi) + C_1(\phi) &= (x_0, y_0) + r_0 * (\cos\phi, \sin\phi) + (x_1, y_1) + r_1 * (\cos\phi, \sin\phi) \\ &= (x_0 + x_1, y_0 + y_1) + (r_0 + r_1) * (\cos\phi, \sin\phi) \end{aligned}$$

The collection of those points is also a circular arc with its center at $(x_0 + x_1, y_0 + y_1)$, its radius $(r_0 + r_1)$, and its parameters confined to $[\phi_0, \phi_1]$. This is visually illustrated in Figure 5.

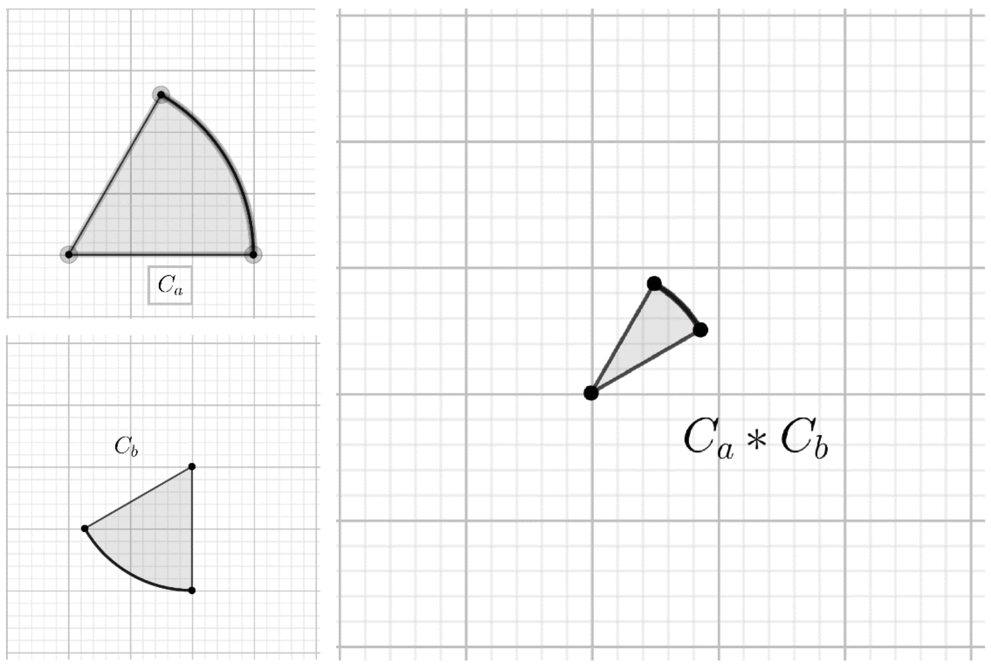

Figure 6 Arc Convolution, the second case

Without loss of generality, we can assume that $r_0 \geq r_1$. Both the arc convolution operator and the Minkowski Sum operator are commutative. Therefore, we can always change the operands order to qualify this assumption.

The second resultant arc that comes from arc convolution can be expressed as $C(x_0 + x_1, y_0 + y_1, r_0 - r_1, \phi_0, \phi_1)$. For the second case, $[\phi_0, \phi_1]$ is the common interval between the two intervals $[\phi_{00}, \phi_{01}]$ and $[\phi_{10} + \pi, \phi_{11} + \pi]$. This is the case where a point with parameter ϕ and another point with $\phi + \pi$ share parallel normals. Those two points results into a point as follows:

$$\begin{aligned} C_0(\phi) + C_1(\phi) &= (x_0, y_0) + r_0 * (\cos\phi, \sin\phi) \\ &+ (x_1, y_1) + r_1 * (\cos(\phi + \pi), \sin(\phi + \pi)) \\ &= (x_0 + x_1, y_0 + y_1) + (r_0 - r_1) * (\cos\phi, \sin\phi) \end{aligned}$$

Again, the collection of these points forms a circular arc. But for the second case, the new arc's radius is the difference of the original arcs' radii. This is illustrated in Figure 6.

So far, we have seen that the boundary of a 2d configuration space obstacle is a set of circular arcs. The resultant arcs in two-dimensional space will stack up in the θ-direction and form a curved surface in three-dimensional space. The exact equation and the parameter domain of that surface can be found by observing how arc convolution changes with θ.

The given robot changes its orientation as the θ value changes. This results in a change in the circular arcs that constitute the robot's boundary. All such arcs need to be rotated with a uniform center point, the *robot center*, which necessarily is not the arc's center. Assume that the robot center is represented with (x_r, y_r). After some arbitrary rotation of θ, $C\ (x, y, r, \phi_0, \phi_1)$ becomes as follows:

$$C_\theta\ (x'(\theta), y'(\theta), r, \phi_0+\theta, \phi_1+\theta)$$

$$\text{Where, } \begin{bmatrix} x'(\theta) \\ y'(\theta) \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \end{bmatrix} + \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

On the other hand, the arcs that constitute the boundary of obstacles do not change with θ. Therefore, an arc convolution between an arc from a robot at rotation θ and an arc from an obstacle would be as follows:

$$C_\theta\ (x', y', r, \phi_0+\theta, \phi_1+\theta) * C_{obs}\ (x_o, y_o, r_o, \phi_2, \phi_3)$$

As we stack these arcs through the θ-direction, they form a surface in 3d.

$$S(\theta, \phi) = \begin{bmatrix} x(\theta,\phi) \\ y(\theta,\phi) \\ z(\theta,\phi) \end{bmatrix} = \begin{matrix} x'(\theta) + x_o + |r1 \pm r2| * \cos(\phi) \\ y'(\theta) + y_o + |r1 \pm r2| * \sin(\phi) \\ \theta \end{matrix}$$

We also need to find the parameter domain of the surface. Assume that the ϕ-value of a robot's arc is bounded to $[\phi_{00} + \theta, \phi_{01} + \theta]$, while that of the obstacle is $[\phi_{10}, \phi_{11}]$. A (θ, ϕ)-value is a part of the parameter domain only when the normal directions are shared. As the ϕ-value interval of the obstacle's arc does not change, we can easily see that two inequalities $\phi_{10} \leq \phi \; and \; \phi \leq \phi_{11}$ must hold. These are the inequalities that restrict parameter domain.

There are two more inequalities. For there to be an overlapping ϕ−value, the two intervals must have an intersection. As θ increases, the first contact occurs at $\theta = \phi_{10} - \phi_{01}$. As θ keeps increasing, there will be an overlapping region until θ reaches $\phi_{11} - \phi_{00}$. This can be seen as a line segment moving diagonally in the $\phi-\theta$ parameter space, forming a band. The inequalities of the band are $\phi - \phi_{01} < \theta$ and $\theta < \phi - \phi_{00}$. Eventually, the four inequalities form a parallelogram in the parameter space.

Figure 7 and Figure 8 show an example of the computed parameter domain. On the left are the input circular arcs, one from the robot and the other from the obstacle. On the right is the parameter space, with the x−axis as ϕ and y−axis as θ. Note that the diagonal part of the band always has a slope of 1.

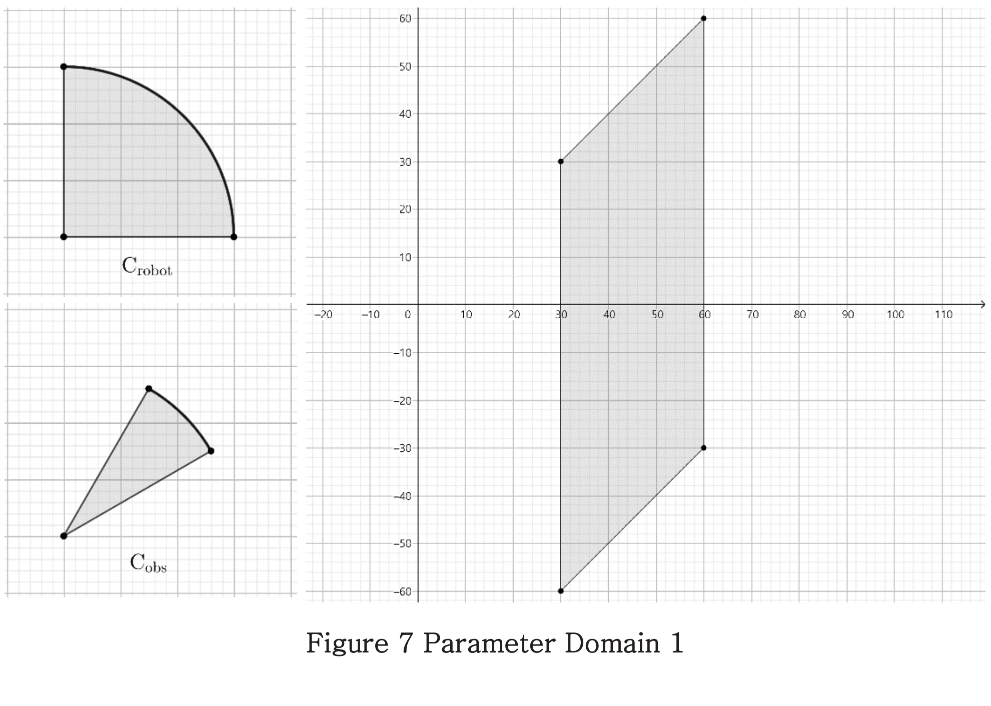

Figure 7 Parameter Domain 1

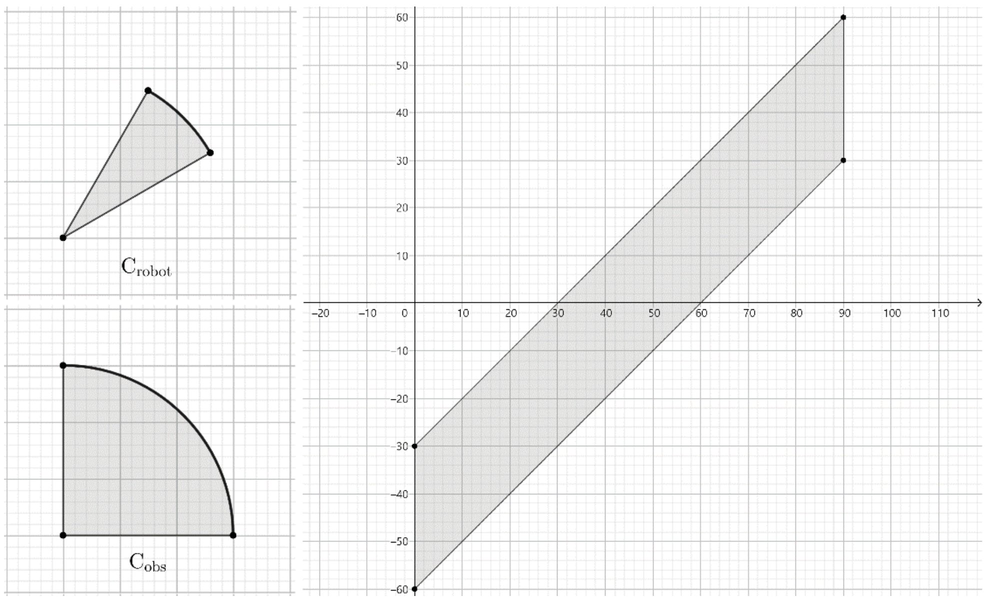

Figure 8 Parameter Domain 2

4.2 3D Voronoi Structure Computation

In this section we discuss how the Voronoi Structure is calculated. We first start with the computation in a single slice. After some sampled slices are properly computed, we stack them up through the θ−direction to get a three−dimensional structure.

The Voronoi Structure can be seen as a representation of the free space with two components, the medial axis and the maximum touching disks. The trails of the center of moving maximum touching disks form the medial axis. In reverse, by sweeping the medial axis with maximum touching circles, we get the free space. We can see the Voronoi Structure as a compression of the free space.

To find the medial axis in the free space, we use the fact that two curve segments have a single branchless bisector under the following conditions [16] :

1. Each endpoint of a curve segment shares a maximum touching disk with an endpoint of another curve segment.
2. The curvature of the two curve segments should be monotone. This means that either the curvature never decreases or never increases.

The bisector segment is a part of the medial axis. We also need to find the endpoints of this bisector curve segment. The endpoints are the two circle centers of the maximum touching disks described in the first condition.

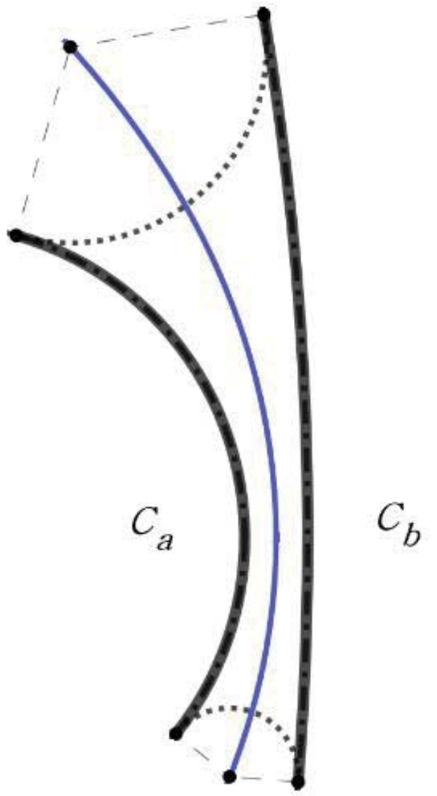

Figure 9 A single branchless bisector curve in a cycle.

We are using circular arcs to represent the boundaries of the configuration space obstacles. A circular arc is a subset of some circle. A circle has a constant curvature. Therefore, it is obvious that the second condition always suffices.

We can divide the whole free space into segments, which is bounded by two circular arcs and two lines connected to a maximum touching disk's center. Then we can evaluate each of these free space segments separately and get a single curve corresponding to that segment.

The free space segment that was segmented following the above paragraph will be called a *cycle* throughout this thesis. This is because the segment is bounded by a loop of alternating circular arcs and transitions between arcs. A *transition* between arcs is a set of 2 line segments, which each of them connects the maximum touching circle's center with its closest circular arc's endpoint. A cycle's boundary alternates between circular arcs and transitions, for instance (Arc_0, $Transition_0$, Arc_1, $Transition_1$, … , Arc_n, $Transition_n$). The last point of the last transition coincides with the first point of the first circular arc. Thus, the structure is named a cycle.

To find a cycle, we first need to find transitions. Since a transition is between two arcs' endpoints that share a maximum touching disk, we first need to find such pairs.

A random arc's endpoint's maximum touching disk would generally touch a point from a different arc, that is not an endpoint. This is because the arcs are a result of biarc approximation, which chooses the arc endpoints in each step to be the one which could minimize the error. Therefore, we need to subdivide the arcs into two smaller arcs at some point whenever some other arc's endpoint shoots a maximum touching disk toward it.

Subdividing all the arcs in the scene with the above process would yield the following statement to hold; each endpoint from some arc in the scene has a paired endpoint which shares a maximum touching disk with it. Therefore, we can now always assume that for every endpoint, there is a transition that leads to another valid endpoint.

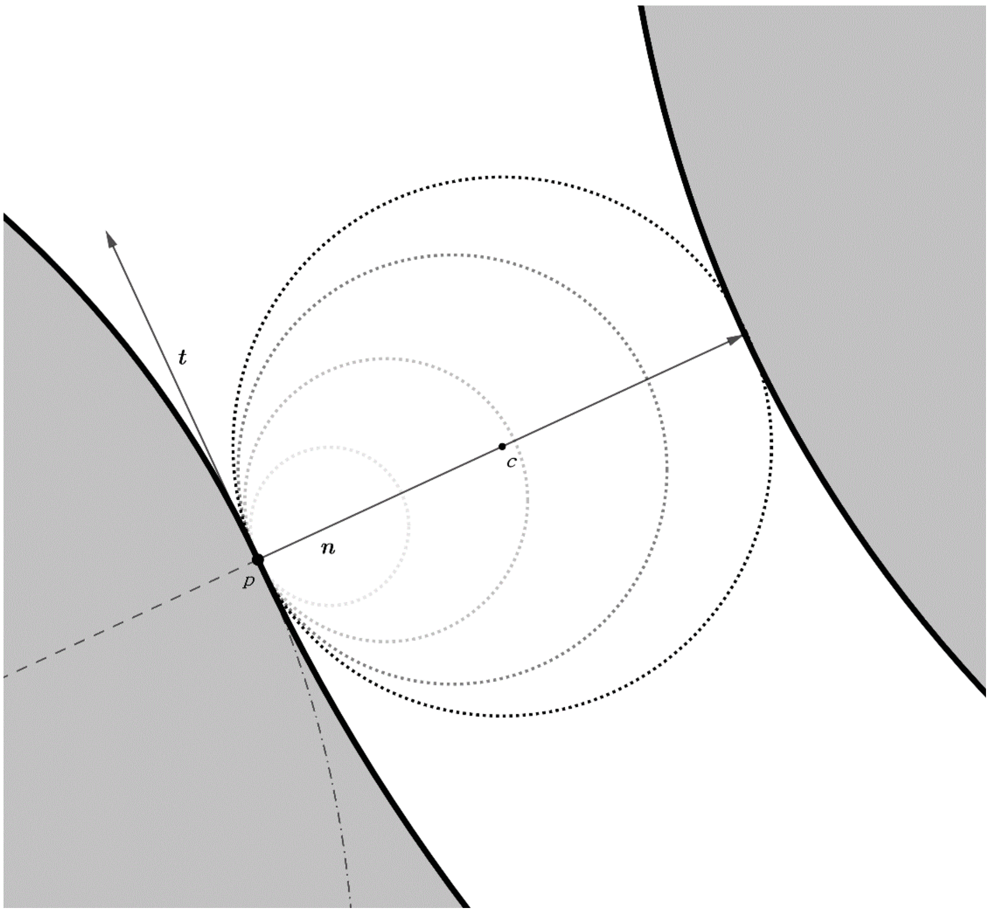

Figure 10 Growing touching disks. The one with the maximum radius is called the maximum touching disk.

The process of finding a maximum touching disk for an arc endpoint is done as follows. Let point **p** denote the endpoint we want to find its maximum touching disk and vector **t** denote its tangent. Without loss of generality, we can assume that all of the configuration space obstacles are parameterized in a counterclockwise manner. This means that for a point on the boundary, the inner region of a configuration space obstacle exists on the point's left side of **t**. On the other hand, the free space exists in the right side. Let $\mathbf{n}$ denote a vector that is perpendicular to $\mathbf{t}$ and pointing towards the free space. As the boundary is counterclockwise, rotating $\mathbf{t}$ with -90 degrees yields $\mathbf{n}$. Due to tangential contact, the maximum touching disk's center $\mathbf{c}$, the circular arc's center, and $\mathbf{p}$ will be all in the same line of direction $\mathbf{n}$. As the disk lies in the free space, we can say that $\mathbf{c} = \mathbf{p} + r * \mathbf{n}$ for some non−negative real value r.

Finding the maximum touching disk is now equivalent to finding a maximum value of a real value r, while the disk not having contact with any of the inner region of the configuration space obstacles. This is also equivalent to finding the minimum value of r that has any contact with the boundary. We can first compute the maximum touching disk radius for each of the circular arcs that constitutes the configuration space obstacles and then apply the minimum to get the maximum touching disk radius for the whole scene.

If a circular arc is extended to have a full angle, for instance $\phi_0 = 0$ and $\phi_1 = 2\pi$, it becomes a circle. Given a point−normal pair $(\mathbf{p}, \mathbf{n})$, computing the maximum touching disk radius for a circle is much simpler than that of a circular arc. But as circular arcs are just mere subsets of some circle, there is no guarantee that such radius is valid.

Depending on the interval of the circular arc $[\phi_0, \phi_1]$, any point on the superset circle can become the point forming the maximum touching disk. For a superset circle C_s, $r(\mathbf{c})$ for $\mathbf{c} \in C_s$ is the maximum touching disk which has contact with $\mathbf{c}$.

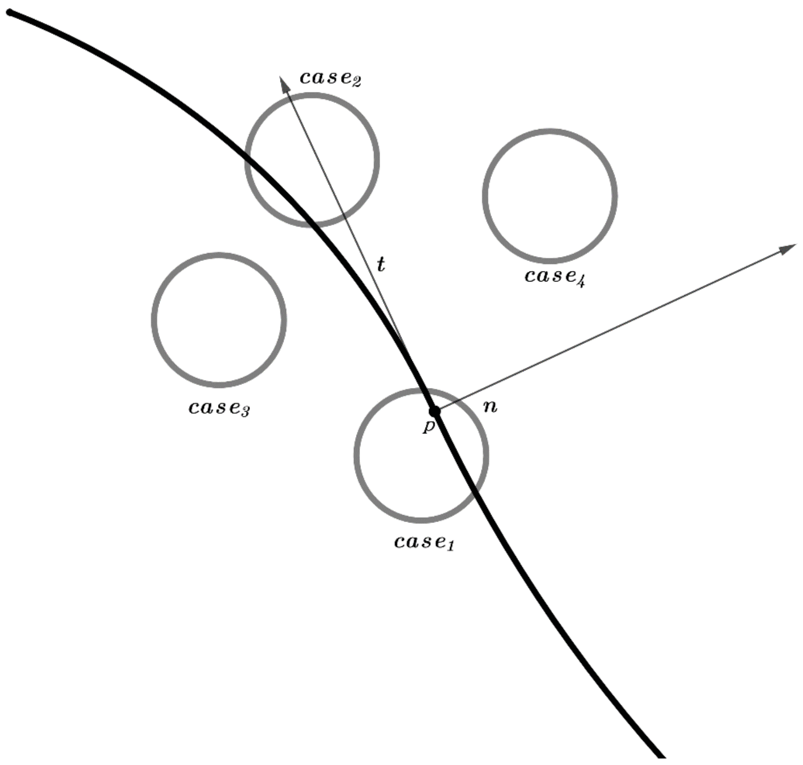

Figure 11 Four cases of C_s.

The relation between a point−normal pair $(\mathbf{p}, \mathbf{n})$ and a random circle C_s, can be divided into four cases.

1. $\mathbf{p}$ is inside C_s.
2. $\mathbf{p}$ is outside of C_s and C_s has contact with the tangential line at $\mathbf{p}$.
3. $\mathbf{p}$ is outside of C_s and $(\mathbf{c} - \mathbf{p}) * \mathbf{n} < 0 \; \forall \mathbf{c} \in C_s$.
4. $\mathbf{p}$ is outside of C_s and $(\mathbf{c} - \mathbf{p}) * \mathbf{n} > 0 \; \forall \mathbf{c} \in C_s$.

The four cases each have a different distribution of $r(\mathbf{c})$.

If a maximum touching disk is shared with a point $\mathbf{c}$ below the tangent line, $r(\mathbf{c})$ will be negative. Such points are meaningless in the perspective of free space. Thus, we can easily ignore all arcs that have a superset of case 3.

For case 1 and 2, the circle has intersections with the tangent line. $r(\mathbf{c})$ is not G^0−continuous near the intersection points. $r(\mathbf{c})$ approaches positive or negative infinity as $\mathbf{c}$ goes close to those points. The circle can be divided into two segments. We ignore the segment with negative r value.

One property the touching disk has is that all touching disks with a radius smaller than $\mathbf{R}$ are contained in the inner region of a touching disk of radius R. Therefore, When the touching disk passes through some point $\mathbf{c}$ in a circle, whether $\mathbf{r(c)}$ increases or decreases is related to whether the tangent heads to the inner region or the outer region. If the inner product between the $\mathbf{c}$'s tangent and the outward normal of the touching disk at $\mathbf{c}$ is positive, $\mathbf{r(c)}$ will increase. If it is negative, $\mathbf{r(c)}$ will decrease. When the inner product is zero, $\mathbf{r(c)}$ is at its critical point. Therefore, we can conclude that $\mathbf{r(c)}$ monotonically increases or decreases between tangential contact points. The value of $\mathbf{r}$ corresponding to the tangential contact point can be found by solving the following equation. (The circle $\mathbf{C_s}$ has a center of $\mathbf{c_c}$ and a radius of r_c)

$$\mathrm{r} = \left| |\mathbf{c_c} - \mathbf{p} - \mathrm{r} * \mathbf{n}| \pm \mathrm{r_C} \right|$$

Above is a tweaked version of the formula which states the relationship between two circles' radius and center distance when there is tangential contact. After getting r, we can easily get the coordinate of the tangential contact point and whether that point is the minimum or not.

If such minimum point is a member of the circular arc in interest, picking that point as the maximum touching disk radius is sufficient. However, if it is not a member of the circular arc, more works should be done. For case 4, where the whole circle lies above the tangent line, we can simply choose one of the endpoints with a smaller $\mathbf{r(c)}$ value. If the circular arc contains the tangential contact point with maximum $\mathbf{r(c)}$, the $\mathbf{r(c)}$ monotonically increases until reaching that tangential contact point. Thus, the only the endpoints have a local minimum. On the other hand, if the tangential contact point with maximum is not included, one endpoint will have a minimum while the other will have a maximum.

For case 1 and 2, some **c** has negative $r(\mathbf{c})$. Getting rid of those parts from the circular arc leads into at most two new circular arcs to evaluate. Again, if the minimum point is contained, simply picking that gets the minimum. If not, picking the best among the endpoints will get the minimum. This is because each of the new circular arcs have a continuous, monotonic, and positive $r(\mathbf{c})$. There is no tangential contact point with local maximum, as the radius can diverge to infinity.

As we can find the maximum touching disk radius for a single circular arc, we can simply iterate through all of the circular arcs and find the minimum value.

Notice that **p** is a point in the boundary. The whole boundary should be G^0−continuous, so there should be two arcs that have **p** as an endpoint. If the model is not G^1−continuous at p, there is no extra modification to the algorithm. But if it is G^1, the arcs that contain **p** should be excluded in the iteration. This is because, numerical errors can lead to a maximum touching disk of radius 0.

As there always exists a transition for a segmented circular arc's endpoint, we can simply form a cycle by alternating between circular arcs and transitions. We start from a circular arc and travel counterclockwise. If the endpoint is reached, we follow the transition and move to another endpoint of some arc. We again follow that arc counterclockwise and repeat. This alternating between circular arcs and transitions terminates when we reach a point that has already been visited, in other words, when the trail forms a cycle. It is obvious that this method would eventually form a cycle and terminate in finite time, as the circular arcs are finite. Iterating this method until all of the circular arcs are visited leads to a decomposition of the free space into cells bounded by cycles.

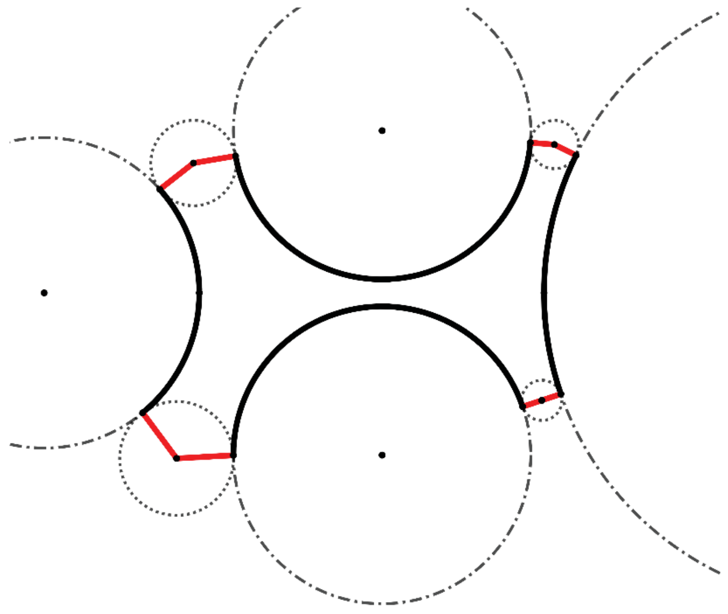

Figure 12 Cycles with 3 or more arcs.

Figure 12 illustrates an example of a cycle. The thick black curves are the circular arc segments from the obstacles. The two red lines are the transitions. You can see that they form a loop and bound a free space segment. The dotted circle is a maximum touching disk at the circular arc endpoints. The dashed and dotted circles are the superset circles of the circular arcs.

You can see in Figure 12 that a cycle might have more than two circular arcs in it. This can easily happen as the second arc's maximum touching disk need not have contact with the first arc and have contact with any other arc in the scene.

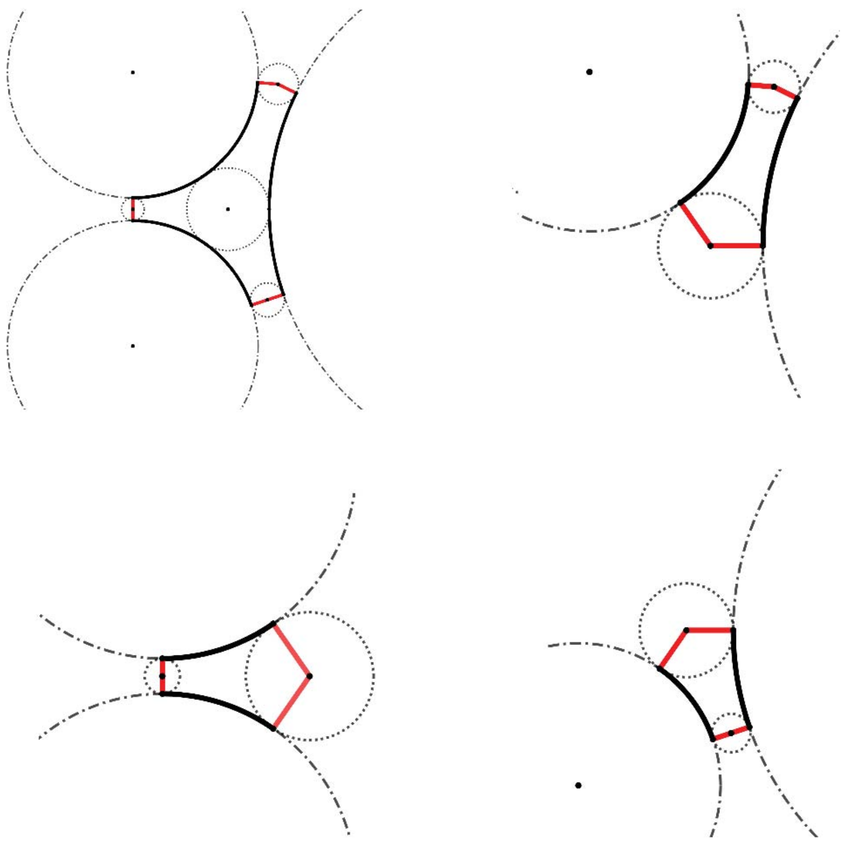

Figure 13 Division of a cycles with 3 circular arcs (upper left) using the Apollonius circle. The Apollonius circle has contact with all three circular arcs. Using the Apollonius circle, the cycle can be divided into subproblems (upper right, lower left, lower right).

A cycle with exactly two circular arcs is needed to evaluate a branch−less medial axis. We can divide a cycle with three circular arcs into three cycles with two circular arcs. Cycles with three circular arcs will have a Y−shaped medial axis inside it. The bifurcation point is the point where medial axis branches and has triple contact with the circular arcs. By finding that point, we can divide the cycle into three parts. Each of the smaller cycle will contain circular arcs segmented at the contact point of the bifurcation point's maximum touching disk. Also, the smaller cycle will contain two transitions, where one of them is from the original cycle and the other corresponding to the maximum touching disk with triple contact.

Such a maximum touching disk is called a *circle of Apollonius*. Circles of Apollonius refers to a set of circles which have tangential contact with the 3 circles given. In our case, the 3 circles correspond to the superset circles of the 3 circular arcs. There are at most 8 solutions to the Circles of Apollonius problem. The Apollonius circle and a given circle can have two types of tangential contact, whether one circles includes the other or not. And there are 3 circles given. These leads to 8 cases of contact types that a solution circle can have and results into a change in the equation. We can easily know whether the two centers lie in the same side or not, as the solution circle's center should always appear on the free space side.

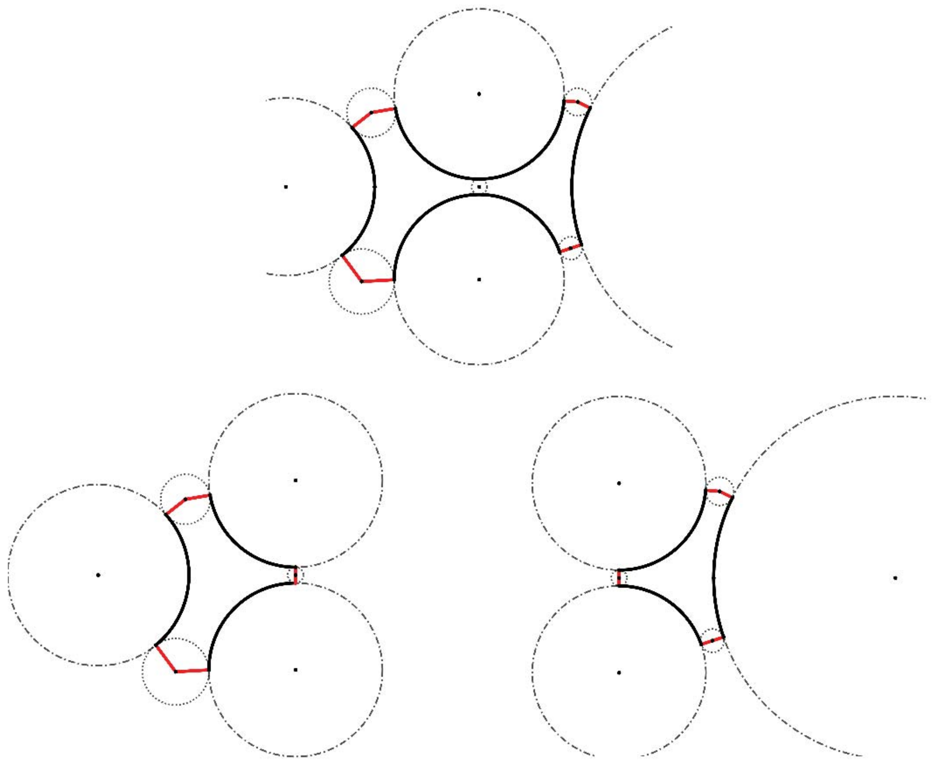

Figure 14 For cycles with 4 or more arcs, randomly sample maximum touching disk to make their children smaller.

There are cases where cycles contain 4 or even more circular arcs in it. A circle that has tangential contact with four circular arcs generally do not exist. Instead, we can find any maximum touching circle and use it to divide the cycle. Adding a transition inside a cycle will cut the cycle in two. The two cycle can have at minimum two circular arcs inside it, while at maximum it can have the same number of arcs before the cut. There is no guarantee that the cut reduces the number of circular arcs in the cycle at each cut operation. But it is guaranteed that the free space size is reduced, as the two new cycles' free space are mutually exclusive and collectively exhaustive to the input cycle's free space. Therefore, we can recursively cut the cycle with many arcs, and it will eventually fall under some error bound.

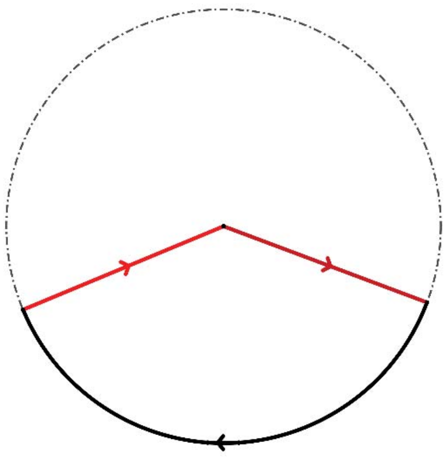

Figure 15 The trivial case of a cycle with single circular arc.

A cycle might have only a single circular arc inside it. This can happen when a circular arc has its concave part to the outside. The touching disk has first contact at another endpoint of the same circular arc. We can simply discard this free space segment. The shape of the free space is a circular sector. No disk in the circular sector can have multiple contacts with the given circular arc. Therefore, there is no medial axis in the inner region.

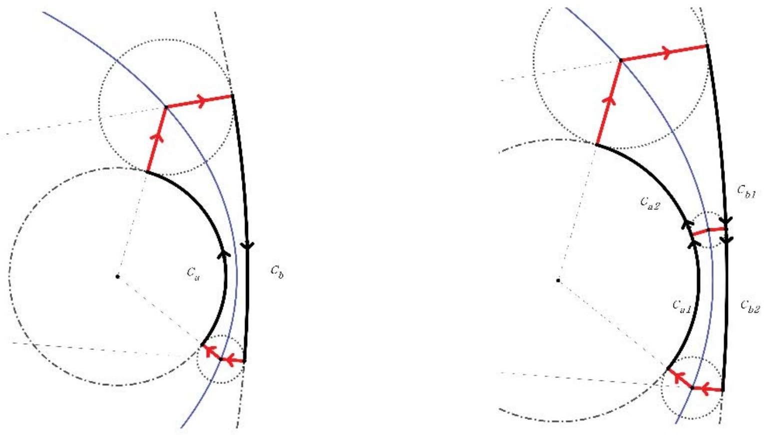

Figure 16 A recursive approach to evaluate a free space with single medial axis segment.

Through cutting, all cycles have two circular arcs in it. Evaluating these cycles leads to a single branch−less medial axis. To evaluate a cycle into a curve, we can take two approaches. The first one is the recursive approach. We can further cut these cycles of two circular arcs. These leads into two cycles with two circular arcs. We can recursively cut these cycles. The free space related to the cycle always decreases at each recursion. Eventually, the circles will almost be very small, and the two transitions' midpoints will come very closer to each other. If some error bound conditions are met, we can simply connect the two transition midpoints to form a very small line segment. As the segmentation of the free space was mutually exclusive and collectively exhaustive, we can get a union of all of these tiny line segments to form the medial axis.

The second approach is geometric. Concerning the fact that the medial axis is a set of equidistant points, we can use the geometric features of circular arcs to get the exact equation of the medial axis in that cycle. The medial axis segment is a conic section.

The type of conic section changes with whether the free space is on the convex part of the circular arc or not. The distance to a circular arc can be represented with the circular arc's radius r and the distance d to the circular arc's center. But this changes with the convexity. If the random point is on the convex side of the arc, the distance to the circular arc is $(d - r)$. On the other hand, if the point exists in concave part of the circular arc, the distance to the circular arc becomes $(r - d)$.

Assume that for a point $\mathbf{p}$ in the free space of a cycle, its distance to the first arc of radius r_1 is d_1, while the distance to the second arc of radius r_2 is d_2. If both circular arcs have a free space on its convex side, the distance between $\mathbf{p}$ and the circular arcs are $(r_1 - d_1)$ and $(r_2 - d_2)$ respectively. The point $\mathbf{p}$ is on the medial axis, therefore those two distances are the same. The equation therefore becomes $r_1 - r_2 = d_1 - d_2$. The left−hand side is a constant. This means that for $\mathbf{p}$ to be a medial axis, its difference of the distances to two points should be a constant. This is a definition of a hyperbola with its foci as the circular arcs' center. The resulting equation is the same when both circular arcs are concave. Therefore, if the convexities of the two circular arcs are the same, the medial axis is a hyperbola. If the radii of the two circular arcs are the same, the hyperbola degenerates into a line segment.

If the convexities of the two circular arcs differ, the equation becomes $r_1 + r_2 = d_1 + d_2$, which is an equation of an ellipse.

We know the foci positions and the major axis length. Therefore, we can evaluate the explicit form of the conic sections using $\cos$, $\sin$, $\cosh$, and $\sinh$. The endpoints of these conics are the maximum touching disk centers. We can know which parameter value it corresponds to using **arccosh** and **arcsinh**. Therefore, with equation and parameter range, we can evaluate a conic section segment.

Notice that the geometric approach uses algebraic formulas, this is usually faster than the recursive approach when size of the free space between the circular arcs is big. Or else, hyperbolic trigonometry functions and their inverse are relatively a heavy operation, and this will make the routine slower.

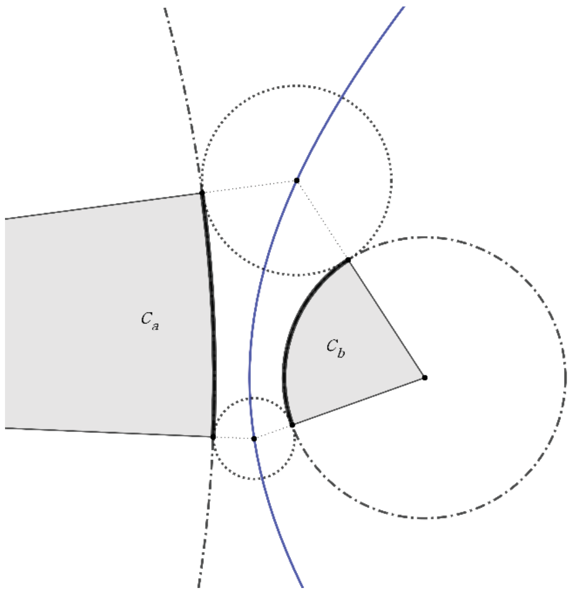

Figure 17 The bisector(blue) between two convex circular arcs is a hyperbola.

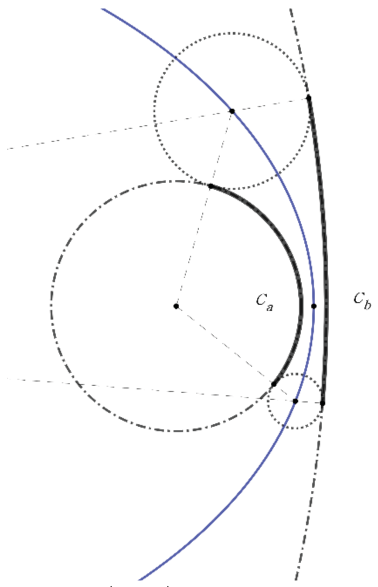

Figure 18 The bisector(blue) between a convex circular arc and a concave circular arc is an ellipse.

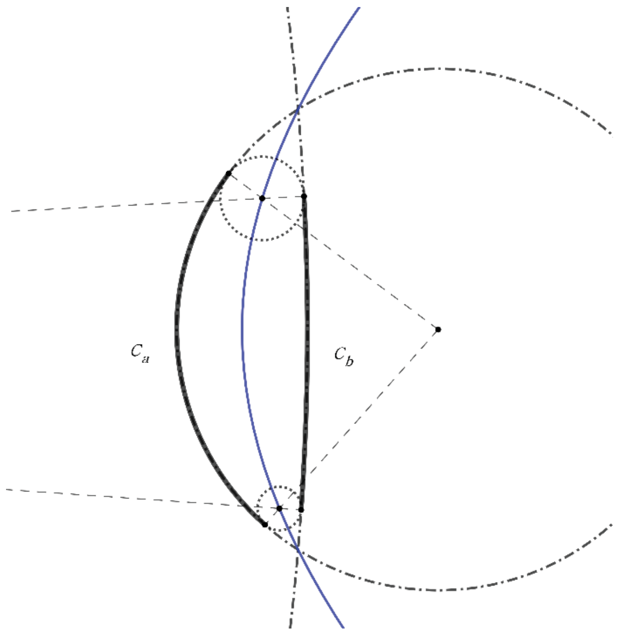

Figure 19 The bisector(blue) between two concave circular arcs is a hyperbola.

From above, Voronoi structure in 2D is computable. The method to construct a 3D structure is to use multiple samples of 2D Voronoi structures. We connect them between slices to get a 3d structure. We can see the medial axis in a slice as a graph structure. The bifurcation points correspond to vertices in the graph, while the medial axes are the edges. The main task is to find which point in one slice corresponds to which point in another slice. Intuitively, if the two slices are very close to each other, the topology of the graph will be the same and the bifurcation points would not move largely. There could be ambiguous correspondence between bifurcation points when they get clustered in a small region, but we can perform denser sampling to reduce the ambiguity to some extent under the error bound. We add edges between bifurcation points if they seem to match. Through this process we get a curve that connects the bifurcation points. These curves are boundaries between Voronoi surface segments.

A surface segment can be formed by picking four neighboring bifurcation points **a, b, c, d** according to the following conditions. The points **a** and **b** should be inside the same slice, while points **c, d** should be inside the same neighboring slice. There should be 4 edges, each of which connects **(a, b), (c, d), (a, c)**, and **(b, d)**. The graph structure should be that of a quadrilateral. The first two edges, which are medial axis component curves confined to a single slice, are almost the same when the slices are close. The latter two edges are line segments almost parallel to the θ−axis, connecting two slices. Therefore, the four edges would form a boundary of a surface that is almost a ruled surface. Collection of these surfaces would result into the Voronoi structure in 3d.

One thing to take attention is the *X−junction*. As θ changes, two bifurcation points with three branches, could come together and become a single bifurcation point with 4 branches. This bifurcation point is called a X−junction and can be thought of as a change in the topology of the Voronoi structure. If there is a large θ value difference between the two slices near a X−junction, the matching of the bifurcation points can get erroneous. To prevent this, a denser sampling is needed when there are two bifurcation points near each other. It is hard to compute the exact value of θ with X−junction. Through dense sampling, an approximate θ where the two bifurcation points are near enough is computable.

Chapter 5

Experiment Results

The proposed algorithms were implemented in C++ under an environment of Windows 10. The tests were performed with a CPU of i7−6700k, a GPU of GTX1070, and a memory of 32GB.

Two pairs of a robot and a scene filled with obstacles were tested. One of them will be called the easy case and the other one will be called the hard case.

The easy case uses a simple square robot and a scene with some regular polygons. The gray object in the left is the robot, while the blue objects in the right are obstacles.

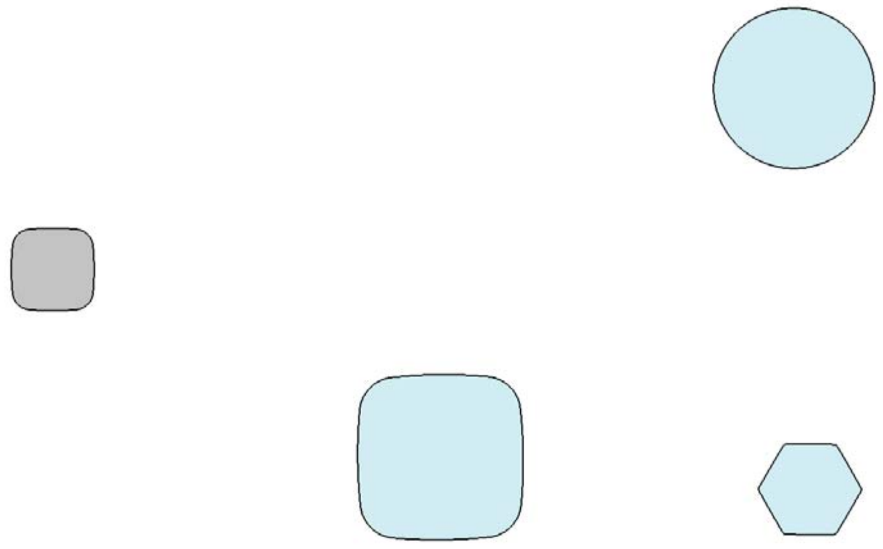

Figure 20 Robot (left, gray) and Obstacles (Right, blue)

Below in Figure 21 is some construction results of the Minkowski Sum and the Voronoi structure in a single slice. The gray objects are the configuration space obstacles. The blue curves are the medial axes. The black square surrounding the scene exists to give a bound to the Voronoi structure, as medial axis can have infinite length. The surrounding square can be thought of as a bound to the robot's working region.

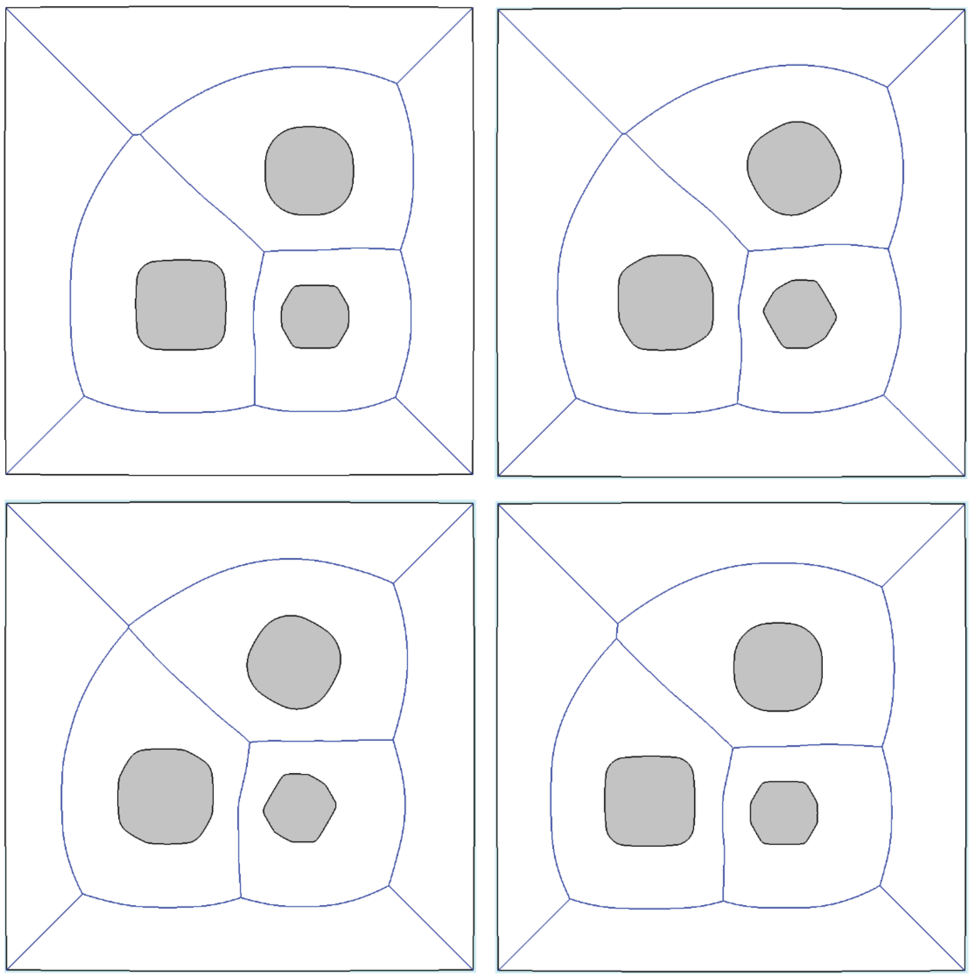

Figure 21 Configuration space obstacles and Voronoi structure at **θ** = 0° (upper left), 30° (upper right), 60° (lower left), 90° (lower right). The result of upper left and lower right is different as the robot's rotational center is not exactly the square's center.

The red objects below in Figure 22 is the configuration space obstacles in 3D. The four images show the same set of configuration space obstacles, with different camera positions and angles.

From the lower right image, you can see that some of the surfaces that is not part of the boundary exist, but they only exist in the inner region of the configuration space obstacles.

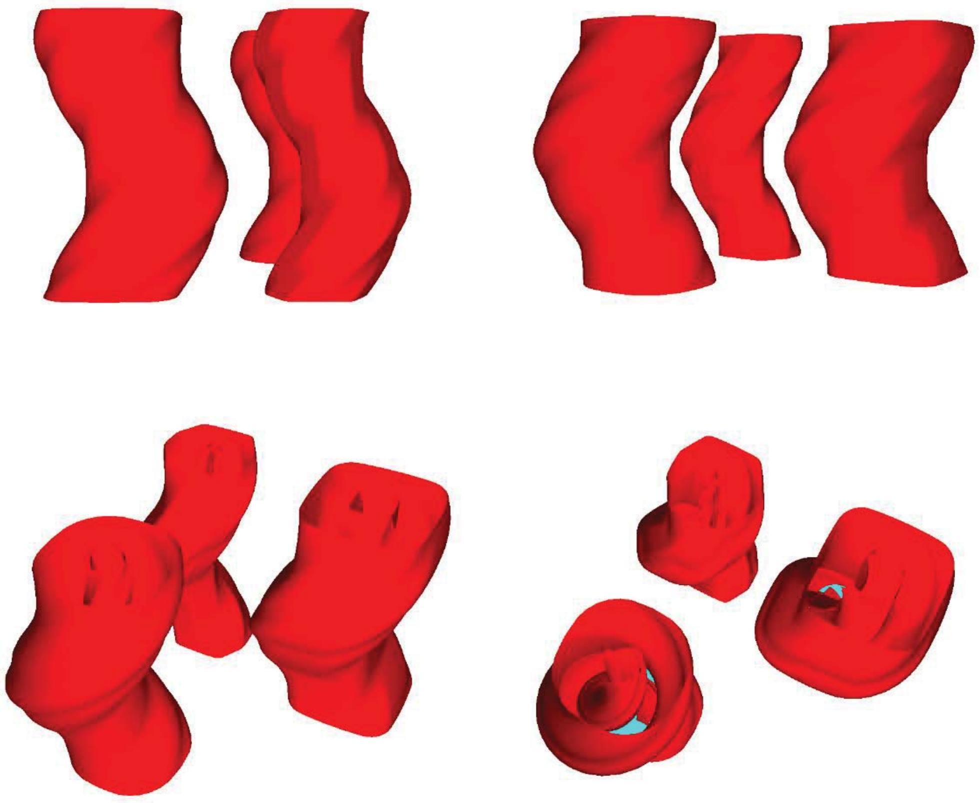

Figure 22 Configuration space obstacles in 3D, with different camera configurations

Stacking up the Voronoi structures in 2D leads to the green object you can see in Figure 24 . The magenta curves in Figure 23 visualizes the connection between the bifurcation points.

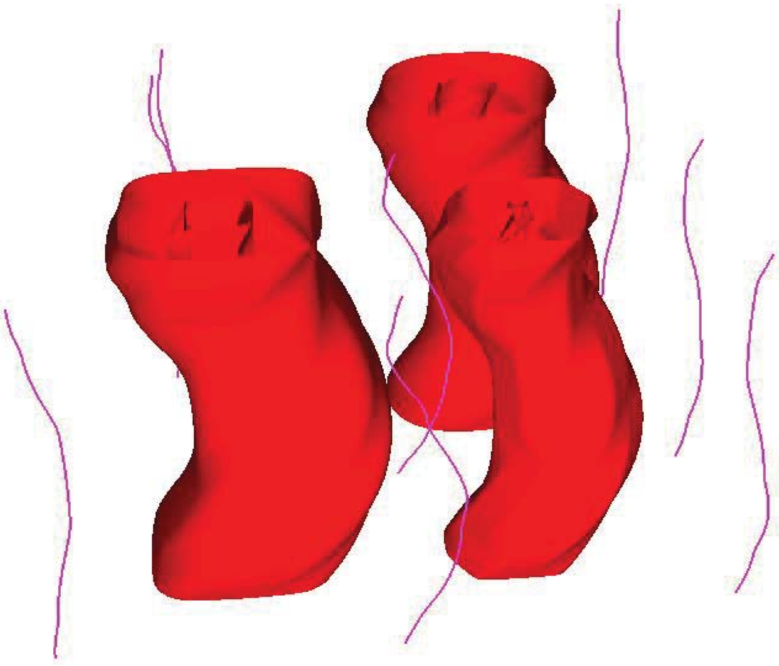

Figure 23 Connected Bifurcation Points in 3D

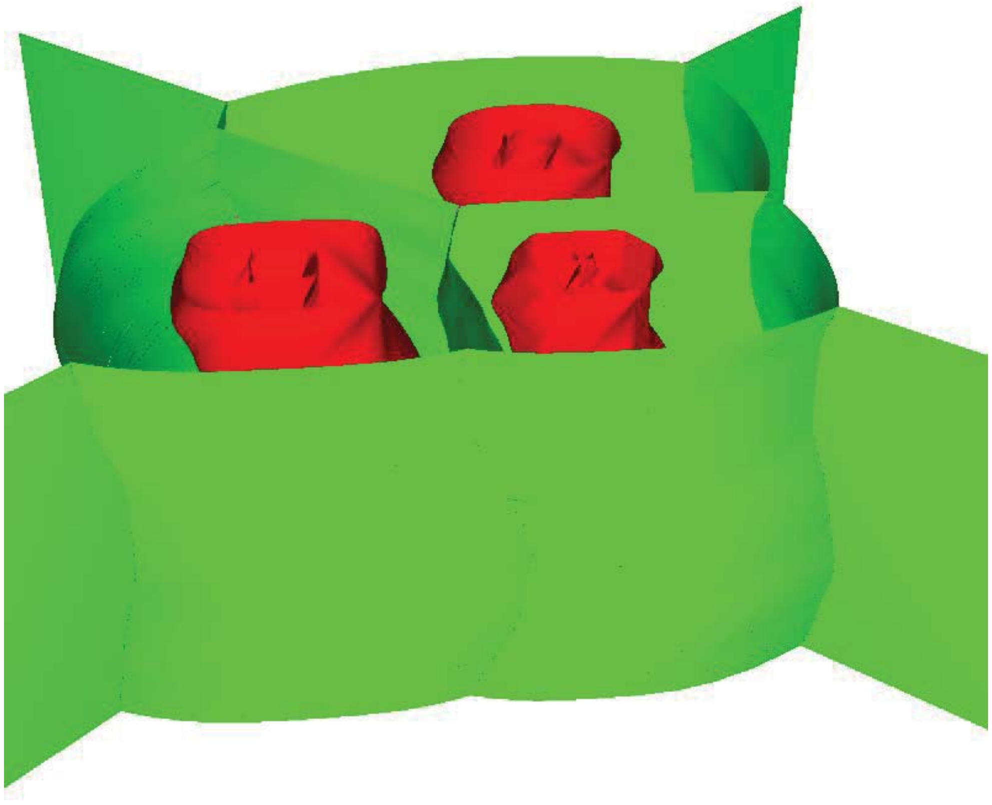

Figure 24 Voronoi structure stacked in 3D for the easy case.

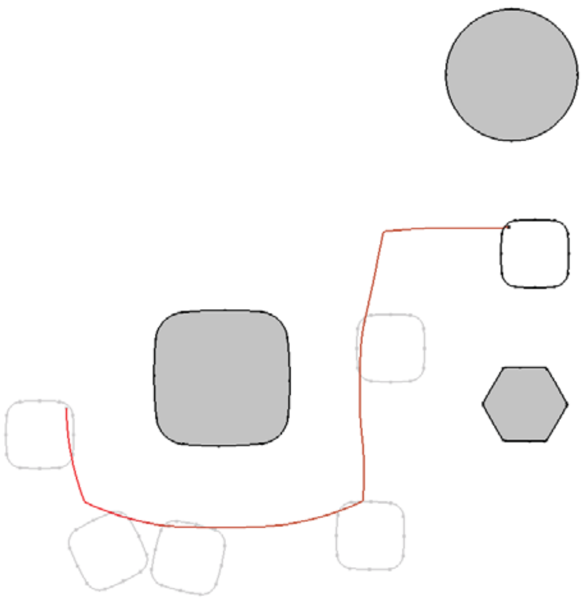
Figure 25 Motion planning Result 1

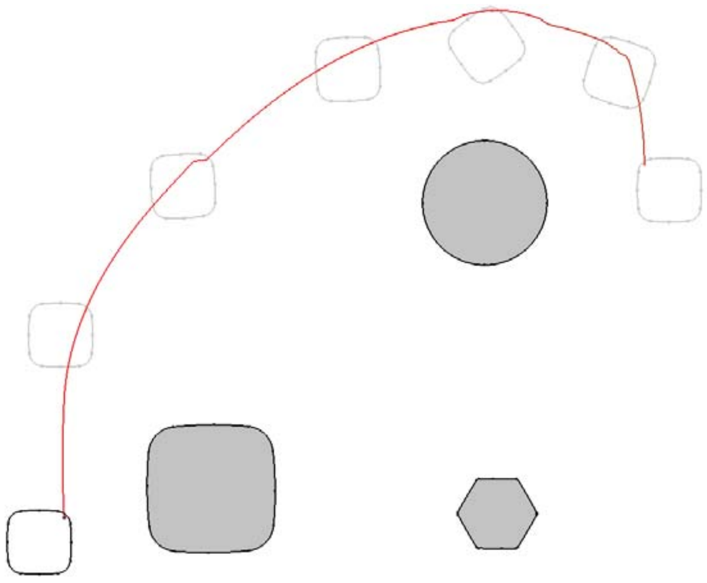
Figure 26 Motion Planning Result 2

What you can see in Figure 25 and Figure 26 are examples of motion planning results, using the network constructed by connecting the two−dimensional Voronoi structures. A total of 360 slices each were used to find the path. The color coding of the path represents the rotation of the robot, to be specific, the RGB value is $\left(1-\frac{\theta}{2\pi}\right)*$

$(1,0,0) + \left(\frac{\theta}{2\pi}\right) * (0,1,0)$.

A* algorithm was used for searching the path in the network of Voronoi structures. The code for the A* algorithm was from an open-source library called C++ Boost.

For the case of Figure 25, it took 2.28 seconds to compute all the slices. It took 3.379 seconds to execute the A* algorithm. And for the case of Figure 26, it took 2.275 seconds to compute all 360 slices, and 3.372 to run A*.

Each of Figure 27 and Figure 28 is a visualization of a path in three-dimensional configurations space of Figure 25 and Figure 26.

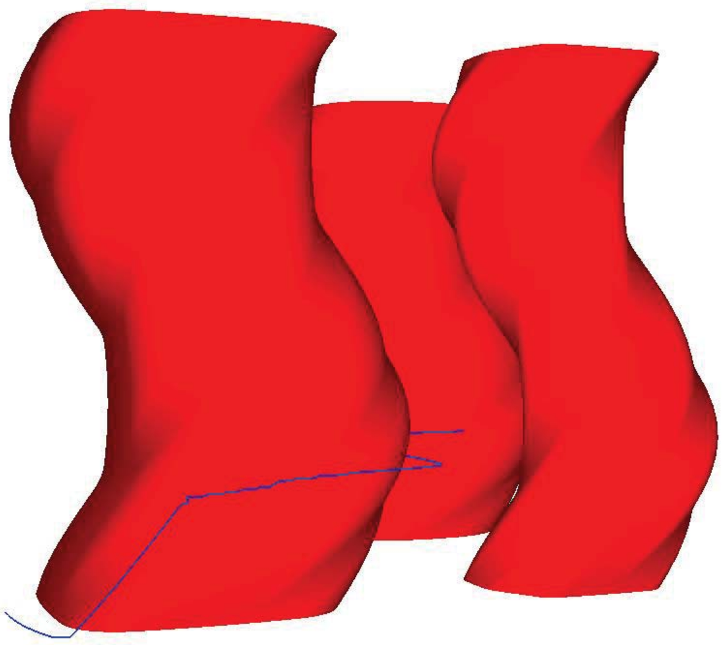

Figure 27 Path of Figure 25 in 3D

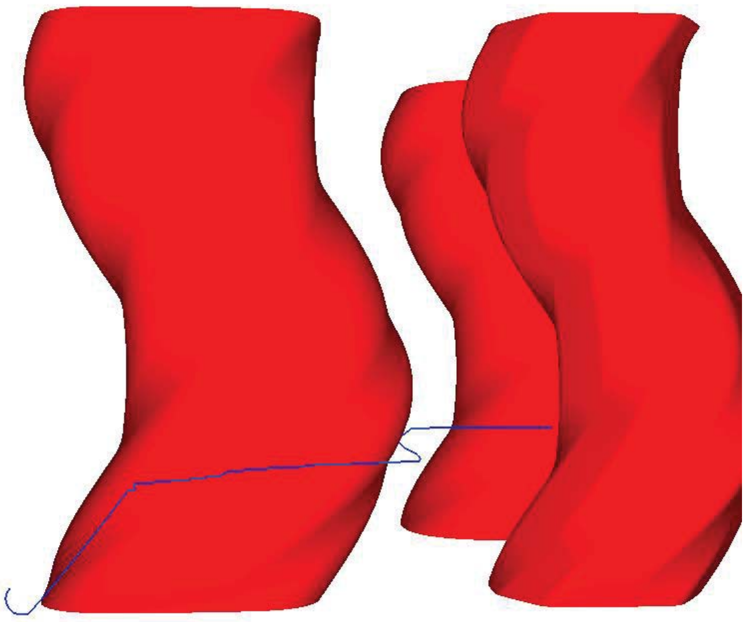

Figure 28 Path of Figure 26 in 3D

For the hard case, we use a 1° rotational sweep volume of a rectangle as the input of a robot. If a path does not have contact with this rotational sweep volume, the original rectangle is guaranteed to be collision−free when all the differences between slices are below 1°.

The scene of obstacles was intentionally designed so that the robot must pass through a narrow corridor while performing rotation simultaneously.

The robot is colored gray and the obstacles in the scene are colored blue in Figure 29.

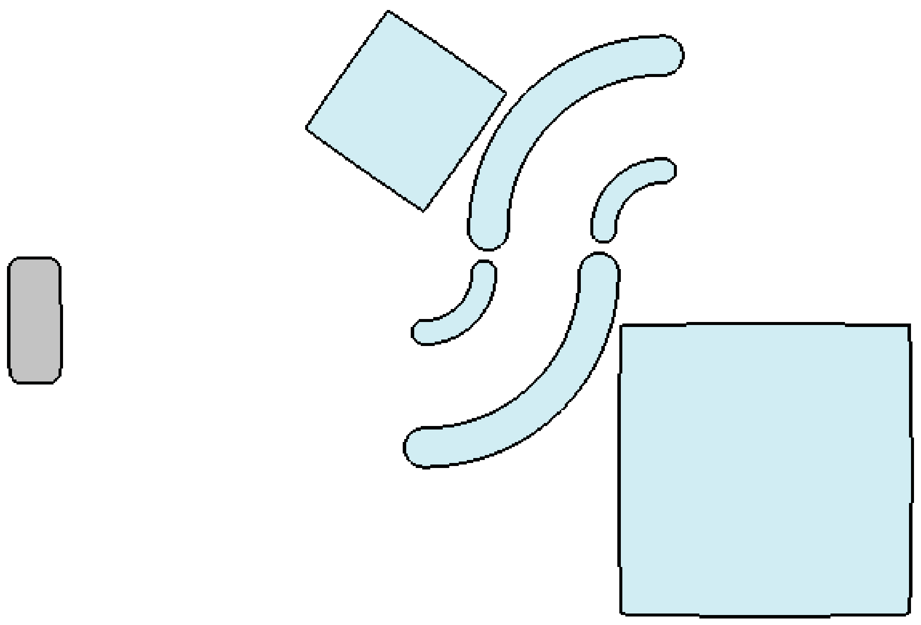

Figure 29 Hard case, Robot (left) and Obstacles (Right)

Figure 30 shows the slices containing configuration space obstacles and the Voronoi structure. Unlike the easy case, the robot has its center at the center of the rotation sweep volume. Therefore, the robot is equivalent to itself rotated with an angle of 180°.

One thing you can find in Figure 30 is that there are much more medial axes in the configuration space. This comes from the fact that the robot is a rotational sweep volume. Rotational sweep volumes have points that are not G^1−continuous. Minkowski sum with such object causes the medial axis to have contact with configuration space obstacles at some points.

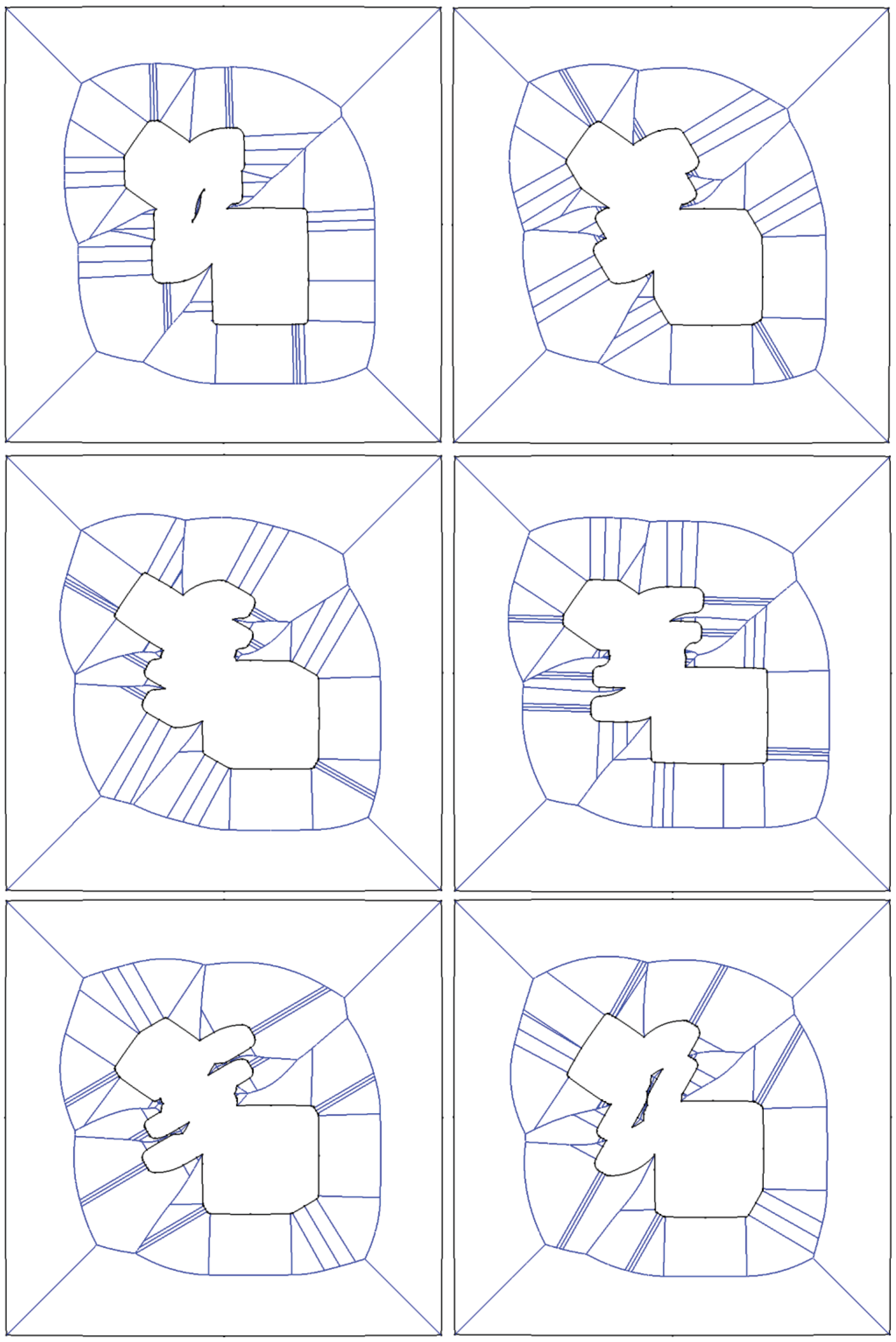

Figure 30 Hard case, slices with rotation of 0°, 30°, 60°, 90°, 120°, 150°

Figure 31 shows the 3D configuration space obstacle's surfaces calculated with the discussed algorithms. You can see in lower left that the hole, which the robot can traverse with rotation, is preserved. This is one example that shows that no redundant surfaces invade free space.

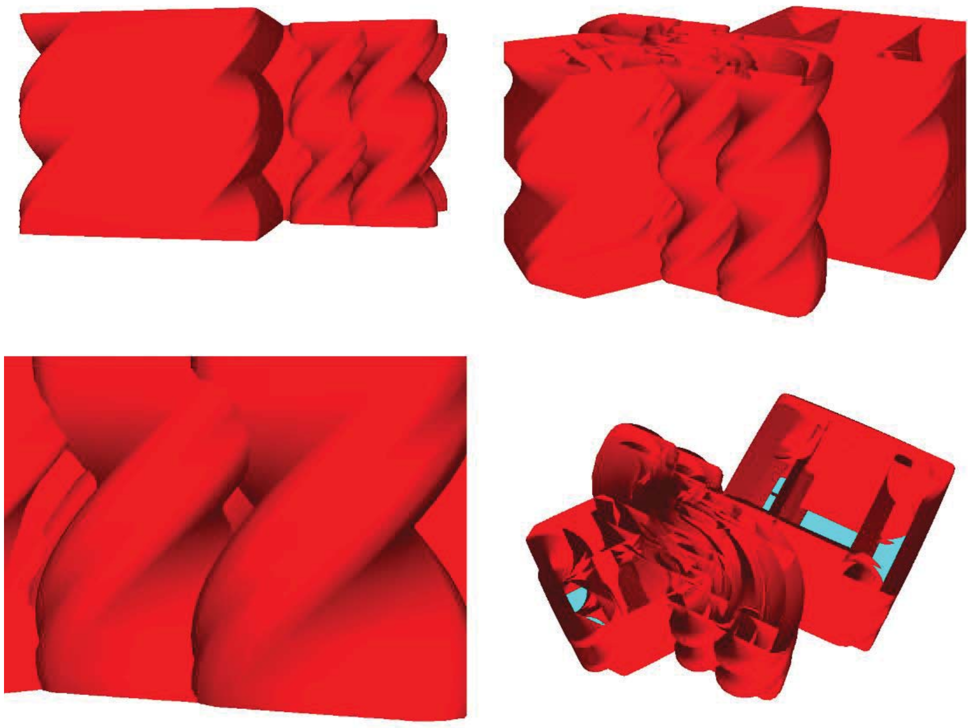

Figure 31 Configuration space obstacles in 3D, for the hard case

The Voronoi Structure in this setting is in Figure 32. The picture above only shows the configuration space obstacles, while the bottom one shows the Voronoi structure too. The camera configuration is the same.

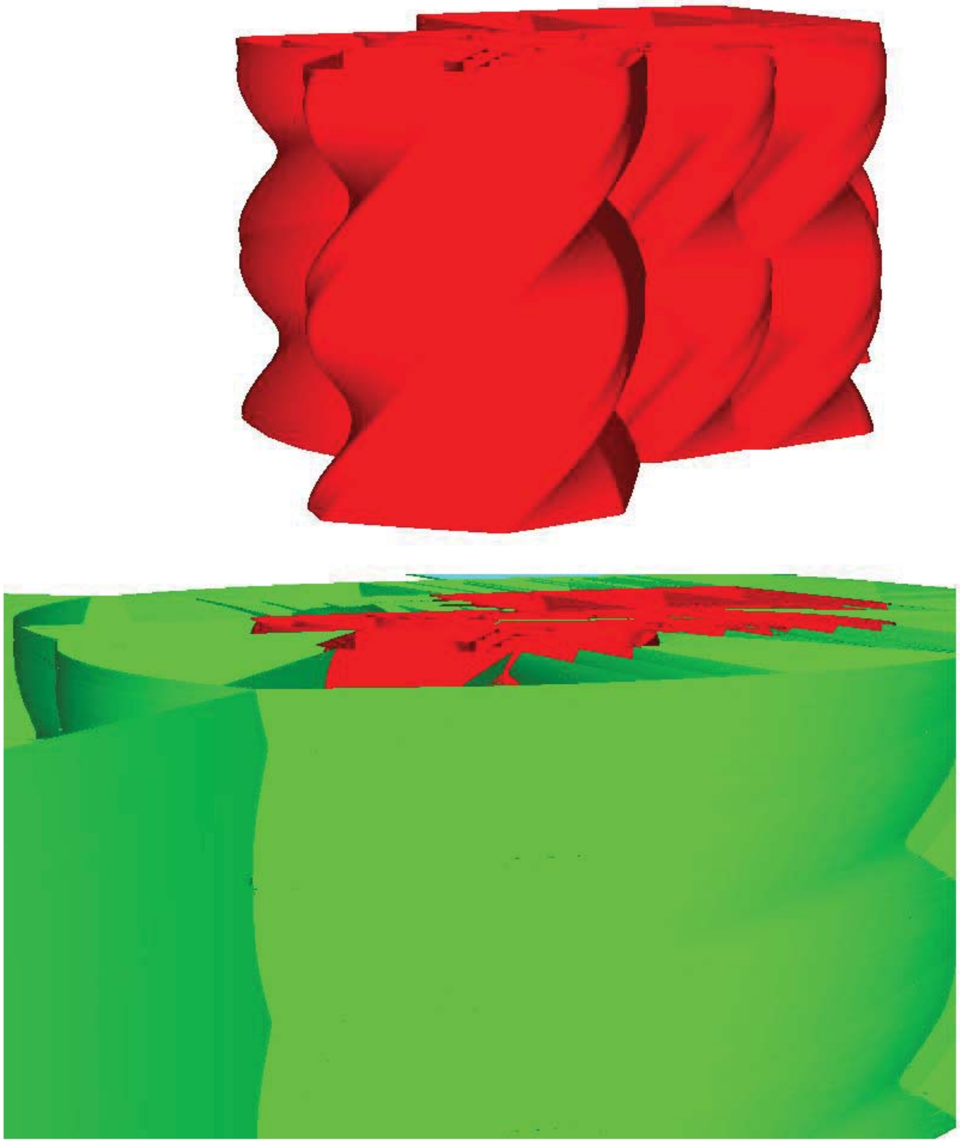

Figure 32 Voronoi structure for hard case.

The path that is made for the hard case is in Figure 33. You can see that the robot rotates through to corridor as intended.

The time needed to generate the Voronoi structure was 2.896 seconds. While the time needed to perform A^* search was 5.93 seconds.

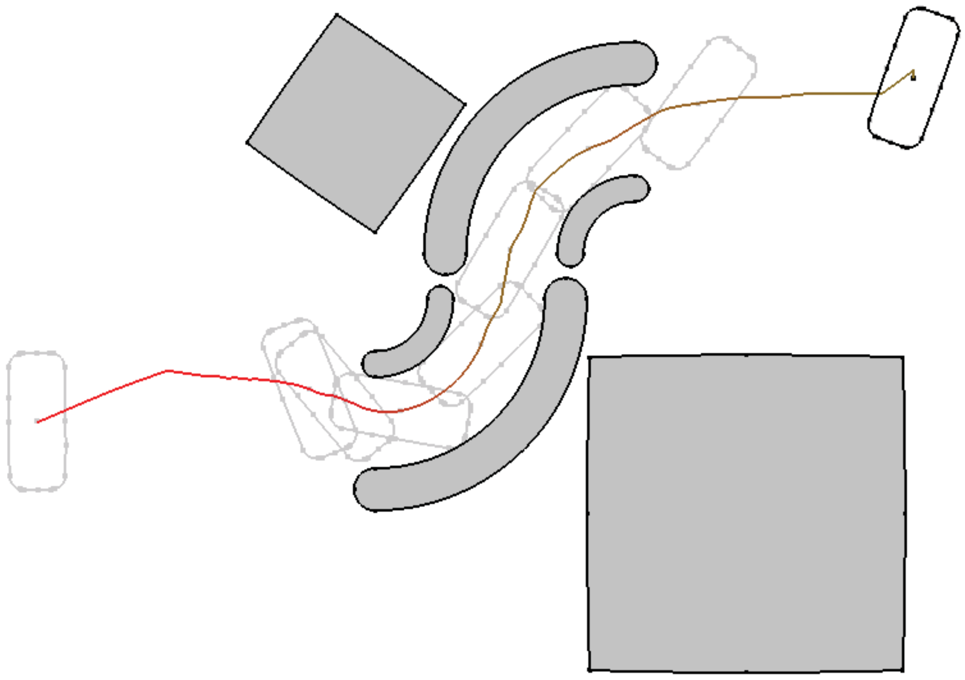

Figure 33 Path for the hard case

Finally, the path in 3D is shown in Figure 34. The figure shows that the narrow corridor is properly generated and followed. The generated path enters the narrow free space inside the object at the upper left image and leaves it at the lower right image.

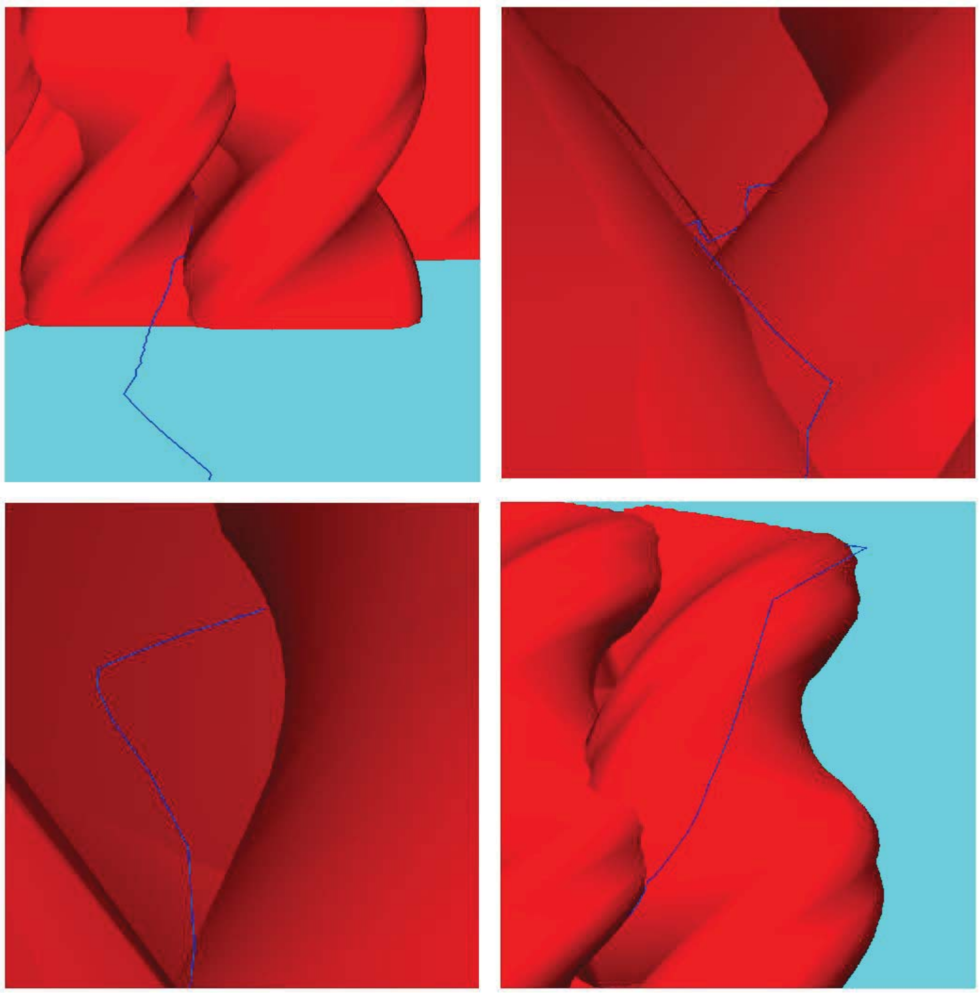

Figure 34 Path in 3D, hard case

Chapter 6

Conclusion

In this thesis we proposed an algorithm to compute the configuration space obstacles for a moving planar robot with translation and rotation in the xyθ−space. Based on the result, the Voronoi structure is also computed. It is generated by stacking two−dimensional medial axes. Notice that the result is different from directly constructing a medial surface in the xyz−space. This is because the distance metric is different from the Euclidean metric.

Another contribution of this thesis is a motion planning framework that is based on an arc−spline representation for two−dimensional objects. This is because our algorithms use circular arcs as input. The circle−based geometric structures greatly simplify motion planning using the configuration space approach. Providing a visualization tool for the 3D configuration space, we can facilitate further research in motion planning.

Bibliography

[1] S. M. LaValle, Planning Algorithms, USA: Cambridge University Press, 2006.

[2] X. Sheng, "Motion Planning for Computer Animation and Virtual Reality Applications," in *Proceedings of the Computer Animation*, USA, 1995.

[3] A. D. Mali, "Motion Planning in Computer Games," in *Encyclopedia of Computer Graphics and Games*, N. Lee, Ed., Cham, Springer International Publishing, 2018, p. 1–6.

[4] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki and S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, MIT Press, 2005.

[5] Y.-J. Kim, G. Elber and M.-S. Kim, "Precise contact motion planning for deformable planar curved shapes," *Computer-Aided Design,* vol. 70, pp. 126-133, 2016.

[6] R. Narayanaswami and J. Pang, "Multiresolution analysis as an approach for tool path planning in NC machining," *Computer-Aided Design,* vol. 35, pp. 167-178, 2003.

[7] D. Edwards, "Amazon now has 200,000 robots working in its warehouses," 21 01 2020. [Online]. Available: https://roboticsandautomationnews.com/2020/01/21/amazon-now-has-200000-robots-working-in-its-warehouses/28840/. [Accessed 01 06 2021].

[8] Lozano-Perez, "Spatial Planning: A Configuration Space Approach," *IEEE Transactions on Computers,* Vols. C-32, pp. 108-120, 1983.

[9] T. Lozano-Pérez and M. A. Wesley, "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles," *Commun. ACM,* vol. 22, p. 560–570, 10 1979.

[10] M.-S. K. G. E. In-Kwon Lee, "Polynomial/Rational Approximation of Minkowski Sum Boundary Curves,," *Graphical Models and Image Processing,,* Vols. Volume 60, Issue 2,, pp. 136-165, 1998,.

[11] D. S. Meek and D. J. Walton, "Approximating smooth planar curves by arc splines," *Journal of Computational and Applied Mathematics,* vol. 59, pp. 221-231, 1995.

[12] K. M. Bolton, "Biarc curves," *Computer-Aided Design,* vol. 7, pp. 89-92, 1975.

[13] J. Lee, Y. Kim, M. Kim and G. Elber, "Comparison of three bounding regions with cubic convergence to planar freeform curves," *The Visual*

Computer, vol. 31, pp. 809-818, 2015.

[14] V. J. Lumelsky and A. A. Stepanov, "Path-Planning Strategies for a Point Mobile Automaton Moving amidst Unknown Obstacles of Arbitrary Shape," *Algorithmica,* vol. 2, p. 403–430, 11 1987.

[15] S. Han, S.-H. Yoon, M.-S. Kim and G. Elber, "Minkowski Sum Computation for Planar Freeform Geometric Models Using G^1-Biarc Approximation and Interior Disk Culling," *Vis. Comput.,* vol. 35, p. 921–933, 6 2019.

[16] J. Lee, Y.-J. Kim, M.-S. Kim and G. Elber, "Efficient Voronoi diagram construction for planar freeform spiral curves," *Computer Aided Geometric Design,* vol. 43, pp. 131-142, 2016.

[17] H. Blum, "A Transformation for Extracting New Descriptors of Shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed., Cambridge, MIT Press, 1967, p. 362–380.

[18] O. Aichholzer, W. Aigner, F. Aurenhammer, T. Hackl, B. Jüttler and M. Rabl, "Medial axis computation for planar free–form shapes," *Computer-Aided Design,* vol. 41, pp. 339-349, 2009.

[19] J. Mizrahi, S. Kim, I. Hanniel, M. Kim and G. Elber, "Minkowski sum computation of B-spline surfaces," *Graph. Model.,* vol. 91, pp. 30-38, 2017.

[20] H. Choi, S. Choi and H. Moon, "Mathematical Theory Of Medial Axis Transform," *Pacific J Math,* vol. 181, 12 1997.

[21] Y. Zhu, F. Sun, Y.-K. Choi, B. Jüttler and W. Wang, "Computing a compact spline representation of the medial axis transform of a 2D shape," *Graphical Models,* vol. 76, pp. 252-262, 2014.

[22] T. Culver, J. Keyser and D. Manocha, "Exact computation of the medial axis of a polyhedron," *Computer Aided Geometric Design,* vol. 21, pp. 65-98, 2004.

[23] S. Musuvathy, E. Cohen and J. Damon, "Computing medial axes of generic 3D regions bounded by B-spline surfaces," *Computer-Aided Design,* vol. 43, pp. 1485-1495, 2011.

[24] Y. Lee, J. Baek, Y. M. Kim and F. C. Park, "IMAT: The Iterative Medial Axis Transform," *Computer Graphics Forum,* vol. n/a.

[25] A. H. Qureshi, Y. Miao, A. Simeonov and M. C. Yip, "Motion Planning Networks: Bridging the Gap Between Learning-Based and Classical Motion Planners," *IEEE Transactions on Robotics,* vol. 37, pp. 48-66, 2021.

[26] P. Gao, Z. Liu, Z. Wu and D. Wang, "A Global Path Planning Algorithm for Robots Using Reinforcement Learning," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019.

[27] L. Butyrev, T. Edelhäußer and C. Mutschler, *Deep Reinforcement Learning for Motion Planning of Mobile Robots,* 2019.

[28] J. Connors and G. Elkaim, "Analysis of a Spline Based, Obstacle Avoiding Path Planning Algorithm," in *2007 IEEE 65th Vehicular*

Technology Conference - VTC2007-Spring, 2007.

Abstract

평면 상에서 움직이는 3자유도 물체의 Configuration 공간과 Voronoi 구조의 계산

이 논문에서는 평면 위에서 움직이는 arc−spline으로 표현된 물체의 Configuration 공간 장애물과 Voronoi 구조의 효율적인 계산 방법을 제시한다. 이를 위해, 원호가 가지는 단순한 기하학적 성질들이 활용된다. Bezier 또는 B−spline 곡선으로 표현된 움직이는 물체의 경우에는, biarc approximation을 통해 arc−spline으로 표현이 가능해진다. Configuration 공간 장애물의 경계면 계산은 대수적인 방법을 통해 그 정확한 식과 정의역을 구할 수 있다. 3차원 Voronoi 구조를 이루는 곡면은 θ가 고정된 2차원 Voronoi Diagram을 쌓는 방식을 사용하여 만들어진다. 마지막으로, 만들어진 곡면 위에서 움직이는 최대 clearance 경로를 생성한다.

Keywords : Configuration 공간, 민코프스키 덧셈, Voronoi 구조, 중심축, 동작 계획, 원호
Student Number : 2019−23686