2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Proving Path Non-existence Using Sampling and Alpha Shapes

Zoe McCarthy, Timothy Bretl, and Seth Hutchinson

Abstract— **In this paper, we address the problem determining the connectivity of a robot's free configuration space. Our method iteratively builds a constructive proof that two configurations lie in disjoint components of the free configuration space. Our algorithm first generates samples that correspond to configurations for which the robot is in collision with an obstacle. These samples are then weighted by their generalized penetration distance, and used to construct alpha shapes. The alpha shape defines a collection of simplices that are fully contained within the configuration space obstacle region. These simplices can be used to quickly solve connectivity queries, which in turn can be used to define termination conditions for sampling-based planners. Such planners, while typically either resolution complete or probabilistically complete, are not able to determine when a path does not exist, and therefore would otherwise rely on heuristics to determine when the search for a free path should be abandoned. An implementation of the algorithm is provided for the case of a 3D Euclidean configuration space, and a proof of correctness is provided.**

I. Introduction

We address the decision problem for path planning: Given initial and goal configurations for a robot, does a collision free path exist that connects them? Most modern path planning algorithms rely on iterative sampling schemes to address this problem using incremental search of some sort. These algorithms are typically not guaranteed to find a solution if one exists, but they are typically either probabilistically complete [1] or resolution complete [2]. One of the key problems for both probabilistically complete and resolution complete planners is, thus, the practical question of how long the algorithm should search for a solution before concluding that none exists. This question is the key motivation for our research. In particular, given an algorithm that can prove that a collision-free path does not exist for given initial and goal configurations, a sampling-based planner can use this algorithm to determine when to terminate its search.

To determine whether a path exists for two configurations, our algorithm attempts to provide a constructive proof that the two configurations lie in disjoint components of the free configuration space. In particular, our algorithm uses a sampling-based approach to construct a set of simplices that (i) lies entirely within the configuration space obstacle region $Q \backslash Q_{\text{free}}$, and (ii) separates the initial and goal configurations. We use simplices derived from the alpha shape of a set of weighted sample points in the configuration space obstacle region—in this context they are n-balls [3]. For a sample $x \in Q \backslash Q_{\text{free}}$, we calculate a lower bound on the generalized penetration depth ρ_x at x [4] (which gives a bound for the shortest path that would remove the robot to Q_{free}). The ball of radius ρ_x at x lies within $Q \backslash Q_{\text{free}}$, and thus the resulting simplices will also be contained within $Q \backslash Q_{\text{free}}$. As sampling continues, the union of balls asymptotically approximates the configuration space obstacle region.

Our algorithm can be viewed as a supplementary check for probablistic or resolution complete planners to determine whether or not they should keep searching for a path or increase resolution, respectively. It can be run in parallel with a more traditional planner attempting to find a collision-free path. We hope to replace the heuristics of these planners with a way to determine whether or not search should be continued. Currently, only $\mathbb{R}^3$ is supported in an implementation, but the proposed method extends easily to higher dimensions.

The main contributions of this paper are to offer an alternate solution to the non-termination problem faced by many modern, sampling-based planners, and to introduce α-shapes [5] as an effective tool for path planning. By focusing only on the topological question of whether or not a path exists between two points (are they in the same connected component?), we demonstrate that an answer to this question can be generated for spherical obstacle sets in $\mathbb{R}^3$. The worst case runtime of our algorithm on a set of balls is $O(n^2)$ with n the number of balls to construct the query structure, and $O(m)$, where m is the number of cells generated by the algorithm, to check if two points are connected.

In what follows, Section II discusses related work, Section III gives a mathematical problem statement, Section IV provides the necessary background on α-shapes, Section V describes our proposed algorithms, and Section VI shows preliminary results.

II. Related Work

A key aspect of the motion planning problem is that configurations that place the robot in collision with obstacles must be avoided. Early attempts at motion planning focused on the exact motion planning problem, that is, given a configuration space, an obstacle space, and two points in the free configuration space(configuration space with the configurations that intersect obstacles removed), does there exist a continuous path between the two that does not intersect any obstacle, and if one exists, specify one, i.e. complete. This as stated for polygonal obstacles is NP-hard as shown in [6]. Classical work in complete motion planning include the piano mover's problem, [7], [8], cylindrical algebraic decomposition, [9]–[11], and Canny roadmap algorithm [6],

Z. McCarthy and S. Hutchinson are with the Department of Electrical and Computer Engineering and T. Bretl is with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA {mccarth4,tbretl,seth}@illinois.edu

[12]. These planners worked with very general obstacle sets, and as such their complexity was high.

As the exact motion problem has been seen to be intractable in high dimensions, newer approaches to the motion planning problem are based on approximation and sampling. The cell decomposition approaches have been generalized to approximate cell decomposition [2]. Other methods that have been very successful and have seen many applications in industry for planning in higher dimensional spaces are the Probablistic Roadmap [13] and Rapidly-Expanding Random Tree [14]. These methods take random sample points in the configuration space, test if they intersect the obstacle, and if they do not then connect the point to some graph structure. This graph structure is searched to find paths between sample points, and then query points are connected to the graphs. There is no stopping condition for when the resolution of the planner is small enough or when enough points have been sampled and connected to the graph to capture all of the topology of the ambient configuration space; there are just heuristics. This results in poor performance when a large number of path planning problems must be executed, especially when paths do not exist in several of the planning queries. This problem of sampling methods must be addressed before path planning can effectively be used as a black box subroutine in larger robotic applications.

There have been several attempts to solve this problem. In [15], a method is given to construct a polynomial such that the zero set is contained within the obstacle, and the start and end configurations evalute positive and negative respectively. In the paper by Basch, et al. [16], an approach is used to determine the connectivity of small passages within the obstacle space, an area which sampling based methods have a difficulty solving. The method examines the point when a particular ratio of the robot has passed through the gap, shrinks the robot, and then samples orientations of the robot to show that none can exist that are not in collision at that ratio. In the papers on fat obstacles by van der Stappen et al. [17], [18], definitions of "fat" obstacles are given and the amount of multiple contacts between polygonal robots and these fat obstacles is proved to be O(n), and it is shown that motion planning using fat obstacles is less complex than in general. In these papers, the definition of fat given imply that spheres are the maximally fat objects, and so in our framework, we have reason to believe that the complexity of the motion planning problem will be reduced.

More recently, new methods in the disconnection problem have been applied by L. Zhang et al. Our work is inspired by these results. In [4], [19]–[21], the authors provided a method of using approximate cell decomposition and c-obstacle queries to determine when no path exists between two configurations. They decompose a configuration space into a set of geometric primitives as cells and use c-obstacle queries, built upon from methods in [22], to prove that the entire cell is contained within the free space or collision space. The geometric primitives used in this case are rectangles. The authors are able to efficiently prove path non existence for queries in configuration spaces with low dimensions. The method divides the space into cells of varying width, and then for each cell perform a c-obstacle query. They utilize the notion of penetration depth in order to obtain a lower bound on the minimum distance that the robot can travel before escaping collision with the obstacle. If an upper bound on the robot's motion within a cell is lower than this penetration depth, then we may conclude that the entire cell is contained in the collision space. They have a corresponding check for a cell's inclusion within the free space, by generating a bound on the separation distance between the robot and the obstacle, and thus can prove that a cell is contained within the free space. By partitioning the configuration space in this way, they show that if every path from the start cell to the end cell travels through a cell that is completely contained within the obstacle, then no path can exist between the start configuration and end configuration. Their notion of generalized penetration depth in [4] is what allows us to calculate configuration space spheres.

We adopt a similar framework but use a different geometric primitive—a simplex (within a triangulation). We construct simplices from sampled points. This allows our cell decomposition to have less axis-constrained structure, as we can sample points from any distribution and construct a cell decomposition around those points. However, this comes at an increased cost of maintaining a triangulation. We discuss the advantages and disadvantages of using simplices as the primitive instead of rectangles, and present preliminary results with an implementation of the proposed algorithms.

III. Problem Statement

We are given: $[a_1, b_1] \times [a_2, b_2] \times [a_3, b_3] = Q \subseteq \mathbb{R}^3$, with the standard metric, two points x_s, x_g, and a set of closed spherical obstacles $O = \{B_{r_i}(x_i)\}_{i \in A}$ for some finite indexing set $A \subset \mathbb{N}$. We assume $B_{r_i}(x_i) \subseteq Q \backslash Q_{\text{free}}$ for each i, given as a set of ordered 4-tuples $\{(x_i, y_i, z_i, r_i) |\ i \in A\}$ where (x_i, y_i, z_i) are the coordinates of the center of the sphere and r_i is the radius of the sphere. In our notation, $B_r(v) = \{x \in Q |\ d(x, v) \leq r\}$. We want to know if there exist a continuous curve $\gamma : [0, 1] \to \tilde{Q}_{\text{free}}$, where $\gamma(0) = q_s$, $\gamma(1) = q_g$, and $Q_{\text{free}} \subseteq \tilde{Q}_{\text{free}} = Q \backslash \mathcal{O}$, for $\mathcal{O} = \cup O \subseteq Q \backslash Q_{\text{free}}$. The question is true if we can find such a path. Showing that there exists no path is in general more difficult than finding a path, since only one path need be found to show that one exists, but proving that there exists no path is a statement about the space of all paths.

IV. Alpha Shapes as Geometric Primitives

We discuss α-shapes in some detail since they are the primary tool for our algorithm. To the authors' knowledge, α-shapes have not been used explicitly for motion planning, but they have been widely applied in geometry, biology, and chemistry as a general purpose geometric modeling and topological tool. α-shapes are the analogue in our algorithm to the rectangular cell decomposition used in [19], and they contain all of the simplices used as cells. If the reader is familiar with α-shapes, weighted α-shapes, and the nerve theorem specialized to α-shapes, this section may be skipped.

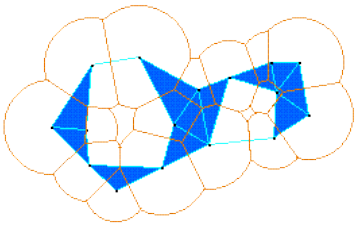

Fig. 1. Example of 2D weighted α-shapes. The α-shape is the simplicial complex defined by the blue lines; the balls it represents are in yellow. Image from http://www.geom.uiuc.edu/ mucke/GeomDir/shapes95/xabs.html.

For a complete treatment of α-shapes, see [5], for weighted α-shapes, [3], and for a thorough discussion of these and related topics, [23]. Note that everything in the following generalizes to $\mathbb{R}^n$, instead of just $\mathbb{R}^3$ as treated here.

A. Introduction to α-shapes

We first define a *simplicial complex* $\mathcal{D}$: a collection of simplices such that if $\Delta_T \subseteq \mathcal{D}$, then if $U \subseteq T$, we have $\Delta_U \subseteq \mathcal{D}$ and also that the intersection of two simplices in $\mathcal{D}$ is either empty or another simplex in $\mathcal{D}$. The α-shape of a set of points S is a generalization of the convex hull of those points. H. Edelsbrunner et al. introduced them in [24] for a set of points in the plane. They were later extended to the three dimensional case in [5], and to arbitrary dimension in [3]. The α-shape is a one-parameter family of polytopes that are an attempt to quantify the 'shape' of a point set(parameterized by postive real α). When $\alpha = 0$, the alpha shape of S is simply S itself. When $\alpha = \infty$, we have that the alpha shape of S is equivalent to the convex hull of S. For intermediate α, the α-shape is more complicated.

The α-shape for $0 < \alpha < \infty$ is best described by analogy. Imagine that the points in S are hard rocks and every other point in $\mathbb{R}^3$ is soft dirt. Then take a spherical shovel of radius α and carve out everywhere that the shovel can fit without colliding with a point in S. Then flatten out the curves so that what is left consists of points in S, edges between two points, triangles between three points, and tetrahedra between four.

α-shapes are important to us because they have a natural interpretation for a union of balls, or a space-filling diagram as they are known in chemistry and biology. Let S be a set of points, $S = \{v_i\}$. Take a ball centered at each point in S with radius r, $S_B = \{B_r(v_i)\}$, and intersect each with the voronoi cell of the center of the voronoi diagram of S. The result is a set of truncated balls, where they are truncated along intersections with other balls. We construct a simplicial complex S_r on S, by letting the 0-simplices in S_r be S. A 1-simplex exists in S_r between two points in S if their truncated balls intersect. A 2-simplex exists in S_r between three points in S if their truncated balls intersect, and finally a 3-simplex exists in S_r between four points in S if their truncated balls intersect. The result is that $S_r = S_\alpha$ for $\alpha = r$.

The result is that α-shapes are a very natural way to reduce a set of balls to a simplicial complex. The nerve theorem, [23], specialized to this case, states that the union of balls in S_B is homotopy equivalent to the alpha shape S_α for $\alpha = r$. Thus, whenever S_B has a void or tunnel, S_α does too, and they correspond exactly. There is actually a stronger property, that the union of balls deformation retracts onto the alpha shape [5]. S_α captures the topology of the union of balls, and packages it in a nice computable form.

B. Connection to Delaunay Triangulation

The α-shape is closely related to another object, the Delaunay Triangulation. The *Delaunay Triangulation* of S in $\mathbb{R}^3$, denoted $\mathcal{D}$, is the collection of k-simplices Δ_T for $0 \leq k \leq 3$ such that there are open empty balls b (of any radius) with $\partial b \cap S = T$. This forms a simplicial complex. So a simplex $\Delta_T \in \mathcal{D}$ if and only if $\exists \alpha \geq 0$ so that Δ_T is α-exposed. This implies that $S_\alpha \subseteq \mathcal{D}$ for each α. The union of the simplices in the Delaunay Triangulation of S is the convex hull of S, so we have that $S_\infty = \mathcal{D}$. The Delaunay Triangulation has been studied extensively, and there exist algorithms for computing it in arbitrary dimension [25], [26].

We can exploit the relationship between the α-shape of S and the Delaunay Triangulation of S. To calculate the entire family of α-shapes, we use the fact that each α-shape is a subset of the Delaunay Triangulation, and simply calculate the Delaunay Triangulation of S. For each simplex in $\mathcal{D}$, we calculate the α-interval in which it is a member of S_α. We would also like to know the intervals for which each simplex in $\mathcal{D}$ is on the boundary of the α-shape, and also when it is interior or exterior to the whole α-shape as a solid. Refer to [5] for a discussion of these intervals and how to compute them. To us, only the 3-simplices that are exterior to S_0, and the boundary faces to the 0-shape will be important.

C. Weighted α-shapes

The above results are only for unions of balls that have the same radius, but extend using a generalization of the Delaunay Triangulation, the Regular Triangulation. The Regular Triangulation is constructed on a set of weighted points in Euclidean space, where the weight becomes the squared radius of a ball centered at each point. The α-shape framework extends to subsets of the Regular Triangulation by using these weights [3]. The weighted α-shape captures the topology of the union of a set of balls with different radii, with the radii specified by a combination of the weights on the points and α. Figure 1 shows an example weighted α-shape in 2D. Algorithms for computing the Regular Triangulation also exist, and we follow the same method for computing the weighted α-shape by first computing the Regular Triangulation and then computing the α-interval for each simplex in the Regular Triangulation such that the simplex is in the α-shape. In $\mathbb{R}^3$, the Regular Triangulation of weighted points may be computed in worst-case time $O(n^2)$ where n is the number of points. This runtime dominates the calculation of the whole family of α-shapes. We use CGAL to compute weighted α-shapes for our input set.

V. Algorithm to Prove Path Non-Existence

In this section we develop the necessary algorithms to prove that no path exists between two points. We begin

by discussing how to generate balls to use as the weighted points as input to an α-shape. We then discuss how to use the resulting α-shape to show path non existence.

A. Generating balls

In [4], the authors calculate a lower bound on separation distance (or penetration depth) for the center of each of their cells. They then reason that if the maximum workplace path length by moving from that configuration to another is less than this distance, then the new configuration is still not in collision (in collision respectively). This approach was taken in [22] to prove that the path between two configurations is contained in the free space. The methods in [4] can be viewed as a multi-dimensional extension of [22] for free cell queries using separation distance, and for collision cell queries using penetration depth. The approach we take is to sample points using some distribution (uniform in our implementation), and then for each sampled point x, if x is in collision obtain a lower bound on penetration depth, say ρ_x. This defines a ball $B_{\rho_x}(x) = \{y \in Q |\ d(x, y) \leq \rho_x\}$, such that if the movement between x and any other point y results in every point on the robot travelling less than ρ_x distance in the workspace, then $y \in B_{\rho_x}(x)$. [4] utilizes this ball implicitly, by setting y to a corner of the given cell, and then determining the maximum distance traveled by the robot between x and y.

We choose a metric on the configuration space that is a lower bound on the length of the motion of any point on the robot between two configurations. In many cases we can scale the axis of the configuration space in order for the standard euclidean metric to be such a bound [27], and we restrict ourselves to those cases here. Thus, when we obtain a bound on the penetration depth of a point, $B_{\rho_x}(x)$ is actually a euclidean ball centered at x, and we also have that $B_{\rho_x} \subseteq (Q \backslash Q_{\text{free}})$. So every time we sample a point, if it is in collision, we can directly obtain some subset of the collision space that is not measure zero. In this way we can sample balls instead of points and thus construct an approximation of the collision space directly by using balls. We then use these sampled balls as the basis for our α-shape algorithm.

B. Path non-existence

Now we have established a framework for generating and asking questions about the union of a set of balls of varying radii: the family of weighted α-shapes. From the nerve theorem, the union of balls and the α-shape that corresponds to it have the same homotopy type. Thus, when a simplex is internal to the α-shape, we know that the simplex is completely contained within the union of balls, and by extension the collision space. We can exploit this fact to learn about the topology of the union of balls in space, and since the union of balls is a subset of the collision space, we learn about its topology by considering all of the solid tetrahedra in $\mathcal{R}$ that are not in the α-shape for a given α (those 3-simplices exterior to the shape for a given α). Taken together with all of the configuration space that is not in the convex hull of the points in S (and thus not in any simplex of the Regular Triangulation of S), they completely capture the topology of our approximation to the free configuration space. Whenever a path exists in the free configuration space between two points, we can find a path of connected 3-simplices in $\mathcal{R} \backslash S_\alpha$ (along with one cell for the complement of the convex hull) that corresponds to that path, and if our approximation has sufficient resolution, the converse is true as well.

Algorithm 1 Compute_Connected_Components(O)

Require: O, Sphere Set
Ensure: DS, Disjoint Set of connected exterior 3-simplices
  $\mathcal{R} \leftarrow$ Compute_Alpha_Shape(O, $\alpha = 0$)
  $T \leftarrow$ Exterior_Finite_3_Simplices($\mathcal{R}$)
  DS $\leftarrow$ Make_Sets_From(T)
  DS.Make_Set(infinite_tetrahedron)
  for all $T_i \in$ T **do**
    for all $T_j \in$ Neighbor(T_i) **do**
      if Classification(T_j) = EXTERIOR **then**
        if Classification(Joining_Face(T_i, T_j)) = EXTERIOR **then**
          DS.Union_Sets(T_i, T_j)
        end if
      end if
    end for
  end for

Algorithm 1 describes the process. Given an input set of balls O, we construct a weighted 0-shape S_0, where the points in S_0 are the centers of the balls of O, and the weights on the points are the various radii squared of the balls in O. Thus, the 0-shape we describe has the same homotopy type as the union of the balls in O. In order to calculate the 0-shape, we calculate the Regular Triangulation of these weighted points, $\mathcal{R}$, and calculate the α-shape when $\alpha = 0$.

We then find all of the 3-simplices that are in $\mathcal{R}$, but not in S_0, by finding all of the 3-simplices that are classified as EXTERIOR for $\alpha = 0$. Given a list of these tetrahedra, T, we find their connected components by using the query: for each tetrahedron in T, check the neighboring tetrahedra (there could be up to 4). If the neighboring tetrahedron is exterior (which includes the case when it is infinite) to S_0, and the face joining the two is exterior to S_0, then the two tetrahedra are connected. We use a disjoint set, or Union-find structure, to keep track of the connected components of T, since connected components are equivalence classes. We add one initial set for the space that is outside of conv(S), since it is connected, and then perform the same test for an exterior tetrahedron that bounds conv(S), if the face between the testing tetrahedron and $\mathbb{R}^3 \backslash$conv(S) is exterior to S_0, we join those sets.

This will build the structure that will allow us to test whether or not two points are disconnected by the sphere set O. We construct a Regular Triangulation in time $O(n^2)$ in the worst case because there can be as many as $O(n^2)$ simplices in $\mathcal{R}$ [5]. The rest of the algorithm runs in time $O(m)$, where m is the number of 3-simplices in $\mathcal{R}$. So the overall runtime is at most $O(n^2)$. We next describe the algorithm that will perform the query for two sample points. For any two given

Algorithm 2 Query_Connectedness(q_s, q_g)

Require: q_s, q_g, Query Points in Q_{free}
Ensure: Boolean, Existence of a path between q_s, q_g
  $T_s \leftarrow$ Tetrahedron_Locate(q_s)
  $T_g \leftarrow$ Tetrahedron_Locate(q_g)
  $S_s \leftarrow$ Find(T_s)
  $S_g \leftarrow$ Find(T_g)
  if $S_s = S_g$ **then**
    Return(TRUE)
  else
    Return(FALSE)
  end if

points, $q_s, q_g \in Q_{\text{free}}$, we simply locate the tetrahedra that each point is located in, T_s, T_g, if the points are in the convex hull of the centers of the balls in O. If they are not, we say that they are within the extra infinite tetrahedron that we have appended to the disjoint sets for Algorithm 1. We then find the sets within the disjoint set structure that the tetrahedra are located in, and check to see if they are the same set. If they are the same set, then they are connected and there exists a path between the two. If they are in different sets, there exists no path between the two points and they are disconnected. This is described below in Algorithm 2. Note that it also implicitly requires that Algorithm 1 has been run on an obstacle set O, so that the set query structure exists.

This algorithm's speed is dependent on the Tetrahedron_Locate query. In worst case it will run in time $O(m)$, again where m is the number of 3-simplices in $\mathcal{R}$. If an optimized search structure is in place for this location query, however, this can be brought down. Finding which sets the tetrahedra are in is essentially constant, as optimized Union-find structures have $O(n\alpha(m, n))$ runtime for m set operations (union, find) on n elements, where $\alpha(\cdot, \cdot)$ is the extremely slow growing inverse to the Ackermann function [28]. Thus, these operations are essentially constant time operations, and as such, the entire runtime is dominated by the location query. This algorithm is correct if the locate calls are correct and the disjoint set structure is initialized correctly. See Appendix A for a proof of the correctness.

We make use of Algorithms 1 and 2 by sampling balls and then periodically running both of the algorithms on the set of balls. If Algorithm 2 returns that no path exists between two points q_s and q_g in the 0-shape (Algorithm 2 returns FALSE), then we know that no path exists in the free space. If TRUE is returned, then we cannot say anything about path existence. If no path exists in the 0-shape, it implies that no path exists between q_s and q_g that does not pass through one of the balls that define the α-shape. Since each of the balls is contained within the obstacle, no path exists between the points that does not pass through the collision space.

We compare the choice of primitives with that of [19]. Rectangles used as a primitive instead of simplices offers some advantages and disadvantages. For instance, using rectangles limits the scaling of the algorithm directly, since the number of rectangles that a region must be subdivided into has hard combinatorial complexity. The expected construction time of a triangulation grows exponentially with respect to dimension, but this is a result of the worst case number of simplices. In many cases according to [3], the number of simplices is actually much lower. Rectangles offer an advantage in that they are easy to maintain. Subdivision in them always results in the same structure, whereas a triangulation of points must maintain the regularity property, and so simplices that were once in the triangulation may be replaced for others. However, triangulations offer the advantage that they are not constrained to be axis-aligned, and thus could approximate curved obstacles better.

VI. Preliminary Results

We have implemented the proposed algorithm for a simple 3-link robotic arm in the plane with polygonal obstacles. In this section we discuss the preliminary results, how performance could be improved, and an extension to n-dimensional α-shapes, and to a hybrid planner, as in [29].

A. Robot Link System

The robot system that we use for our simulation is a 3 revolute joint robotic arm with joint lengths $\{j_1, j_2, j_3\}$. The first joint's base is fixed at $(0, 0)$. From [27] 8.5.1.3 and 5.3.4, we can linearly scale our coordinate axes by the maximum radius of the robot arm that can rotate around each given joint. Then, the standard euclidean metric on the configuration space is an upper bound on the maximum path length by any point on the robot between two configurations, as described earlier. In our implementation, this results in a very conservative estimate on the collision space surrounding a point, as the euclidean metric assumption is fairly restrictive. A large number of samples are required to make up for this.

We can specify any two points in the configuration space and we can sample balls to determine if the points are disconnected. We utilized CGAL [30] to calculate α-shapes, and we utilized SOLID collision detection library (obtained at http://www.dtecta.com/) for collision detection and penetration depth calculations, which is based off of [31]. Several examples of configurations that our algorithm proved that no path exists between are shown below, along with the corresponding 0-shapes. In the first example, path non existence was proven after sampling 36000 points. In the second, path non existence was proven after sampling 30000. The large number of samples required to prove disconnection is likely a combined result of our sampling scheme, the joint limits, and our upper bound on the robot's motion. The approximation to $Q \backslash Q_{\text{free}}$ quickly approaches the overall shape, but the small holes between the regular collision points and joint limit collision points cause non-existence proofs to take longer than expected. Since we are sampling uniformly, we rely on random chance for holes to be filled.

B. Implementation Issues

There were a few implementation issues for this simulation. First, we impose joint limit constraints on the robot in order to avoid identifying equivalent cells in a periodic

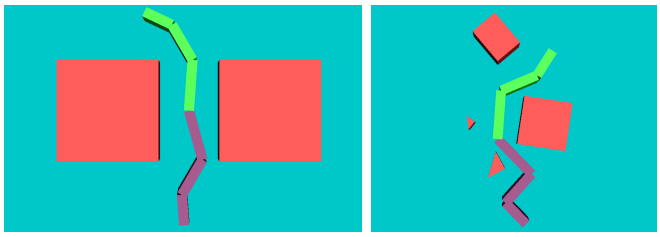

Fig. 2. A 3-link robot arm at two different configurations and obstacle configurations. The start configuration is in green, the end configuration is in purple, and the obstacles are red.

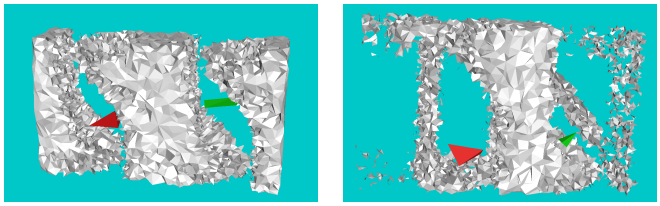

Fig. 3. The 0-shape for planning between the first configuration and the second in the above examples. The white tetrahedra are interior to the 0-shape(and thus a subset of $Q \backslash Q_{\text{free}}$), the green tetrahedron is the start cell, and the red tetrahedron is the end cell.

triangulation. Since the α-shape is homotopy equivalent to the union of balls, unless we sample balls to make up the boundary of the joint limits, our algorithm will likely not prove path non-existence. There was some nuance in selecting the correct balls to sample outside of the joint limits, since we would like to fill the entire boundary of the joint limit constraints with balls, but this increases the runtime of the triangulation and α-shape generation. A solution was to sample balls in a region that is some factor $R > 1$ larger than the configuration space that satisfies joint limits, and then consider anything outside of the limits to be in collision with penetration depth equal to the maximum of the distance to the joint limits and the actual penetration depth if the configuration is in collision. This ensures overlap into the configuration space for samples that are actually in collision and outside of joint limits. However, work is still being done as this scheme results in a large number of required samples to prove disconnection. The 0-shapes shown in the figures are not the shapes that prove disconnection; they are the shape that occurs when sampling within joint limits. When figures are produced when sampling outside of the joint limits as well, the entire 0-shape appears to be solid from the outside. If we would like to not enforce joint limits, we must take special care and utilize a periodic triangulation package that explicitly identifies cells that occur on a periodic axis such as CGAL's 3d Periodic Triangulation package.

An issue that affected performance was the CGAL library's implementation of α-shapes. Their implementation of the regular triangulation was fully dynamic (supports insertions and deletions), but the α-shape implementation was not. So the entire α-shape is periodically constructed from the regular triangulation, which has many redundant calculations. This re-computation dominated the runtime of the entire algorithm, and so a more dynamic solution is necessary to have competitive performance. Note that this is not a fundamental limitation of α-shapes, but rather just the given implementation, since we can simply compute the alpha structure of each new simplex in a dynamic triangulation as we insert (which we can do in $O(1)$ time in fixed dimension). This pushes the runtime of our algorithm to $O(n^3)$ in our implementation, as we must periodically reconstruct the entire triangulation (but is still theoretically only $O(n^2)$ with a better implementation).

C. Extensions

There are several possible useful extensions to this algorithm. Since the weighted α-shape construction proceeds with very few modifications for any dimension of euclidean space, we can apply the entire algorithm to determine whether or not query points are disconnected within a high-dimensional configuration space by hyper-balls. However, the authors are unaware of any library that computes high dimensional α-shapes, as their use as a geometric modeling and visualization tool is mostly lost in that setting. Since we can reconstruct any obstacle space with hyper-spheres, this may be an effective way of disconnection proving for high dimensional space. We note that again the runtime of such an algorithm would be dominated by the construction of the n-d Delaunay or Regular Triangulation, as it can contain up to $O(n^{\frac{d}{2}})$ simplices, for n spheres and d the dimension of the space, and thus is bounded by that runtime [3]. Note that this bound is lower than the bound for the Canny roadmap algorithm, however, so queries of disconnection run faster than constructing an exact path for these sphere sets.

In our construction, we only utilize the α-shape family for $\alpha = 0$. We could instead grow the obstacle set by increasing α. We could maintain a family of disjoint sets, parameterized by increasing α, such that for obstacles grown by any size, we could query whether or not two points can reach each other in the free space. This essentially deals with uncertainty about the obstacle set. By varying the size of the obstacle set, we can check the robustness of a path between two points, by noting the α in which the two points become disconnected by the obstacles. Since many algorithms that run for α-shapes run on them incrementally for increasing or decreasing α, and in particular those for calculating the betti numbers of the α-shape (which quantify the topological characteristics), we can use this information to filter for interesting α's in which the topology of the α-shape changes. We then only update the disjoint set structure for these α's, utilizing the simplex that the interesting α adds or removes. In this way, we can construct the set structure for all α, and allow for queries for different α's, or give the α-robustness of a given connectivity result, as a measure of how connected two points are.

We can utilize the framework in [29], [19] to augment our path non-existence prover with a smarter sampling strategy and path planner. Instead of simply using a disjoing set structure to store the connected components of the exterior simplices, we can instead attempt to find a path of external simplices between q_s and q_g. Then, if a potential path has been found, then we adaptively sample within the cells in

the returned path. This directs the sampling to potential paths which require more resolution to determine if a path exists or not. If the sampling strategy is made adaptive in this manner, then our algorithm can be directly compared to those given in [19] for performance and robustness issues.

VII. Conclusion

We have presented an algorithm that samples euclidean balls that are completely contained within the collision space obstacle. It then determines if no path exists through the union of balls, and if none does, then no path exists through the larger collision space. The main difference between this algorithm and the one proposed in [19] is in the geometric primitive used to decompose the configuration space. In our algorithm, the primitives used are simplices in a triangulation, and in [19], the primitives are rectangular cells. This research applies α-shapes, a geometric and topological modeling tool, in a new way applied to robotic path planning.

Acknowledgments

This work was supported by the National Science Foundation under CPS-0931871 and CMMI-0956362.

Appendix

We assume that path connectedness is equivalent to connectedness in our topological spaces.

Theorem 1.1: $q_s, q_g \in Y = \mathbb{R}^3 \backslash (\cup_{B_r(v_i) \in O} B_r(v_i))$ are path connected in Y if and only if they are path connected in $X = \mathbb{R}^3 \backslash S_0$.

Proof: Since $\cup_{B_r(v_i) \in O} B_r(v_i)$ deformation retracts onto S_0, we have that they have the same number of voids, and thus their complements have the same number of connected components. Since $S_0 \subseteq \cup_{B_r(v_i) \in O} B_r(v_i)$, we have that $Y \subseteq X$. Thus, each connected component of Y is a subset of some connected component of X, but since they have the same number, they correspond exactly. Since $q_s, q_g \in Y$, they must also be in X, and so we have that they are in the same connected component in Y if any only if they are in the same connected component in X. ∎

Now we show that Algorithm 1 produces the connected components of $\mathbb{R}^3 \backslash S_0$. Fix $q_s, q_g \in \mathbb{R}^3 \backslash (\cup_{B_r(v_i) \in O} B_r(v_i))$ in the same connected component. Fix a collision free path γ (with respect to $\mathbb{R}^3 \backslash (\cup_{B_r(v_i) \in O} B_r(v_i))$) between q_s and q_g. Let $\{S_i\}$ be the finite set of cells and faces that $\gamma([0,1])$ intersects. Each has a point in it that does not intersect any ball in O—the point on the path that intersects the face or cell. Thus, the face or cell is not fully contained in the obstacle space and is not a part of the alpha shape, hence is EXTERIOR. So the algorithm that links exterior cells by exterior faces produces the correct connected components.

References

[1] J. Barraquand, L. Kavraki, J.-C. Latombe, R. Motwani, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," *The Int. Journal of Robot. Research*, vol. 16, no. 6, pp. 759–774, 1997.

[2] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[3] H. Edelsbrunner, "Weighted alpha shapes," University of Illinois at Urbana-Champaign, Champaign, IL, USA, Tech. Rep., 1992.

[4] L. Zhang, Y. J. Kim, G. Varadhan, and D. Manocha, "Generalized penetration depth computation," *Computer-Aided Design*, vol. 39, no. 8, pp. 625–638, 2007.

[5] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Trans. Graph.*, vol. 13, pp. 43–72, January 1994.

[6] J. F. Canny, *The complexity of robot motion planning*. Cambridge, MA, USA: MIT Press, 1988.

[7] J. T. Schwartz and M. Sharir, "On the piano movers' problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on Pure and Applied Mathematics*, vol. 36, no. 3, pp. 345–398, 1983.

[8] ——, "On the "piano movers" problem. ii. general techniques for computing topological properties of real algebraic manifolds," *Advances in App. Math.*, vol. 4, no. 3, pp. 298 – 351, 1983.

[9] G. E. Collins, "Quantifier elimination for real closed fields by cylindrical algebraic decomposition–preliminary report," *SIGSAM Bull.*, vol. 8, pp. 80–90, August 1974.

[10] J. H. Davenport and J. Heintz, "Real quantifier elimination is doubly exponential," *J. Symb. Comput.*, vol. 5, pp. 29–35, February 1988.

[11] G. E. Collins and H. Hong, "Partial cylindrical algebraic decomposition for quantifier elimination," *J. Symb. Comput.*, vol. 12, pp. 299–328, September 1991.

[12] J. Canny, "A new algebraic method for robot motion planning and real geometry," in *Ann. Symp. Found. Comp. Sci.*, Washington, DC, 1987.

[13] 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. 566–580, 1996.

[14] S. M. LaValle and J. J. Kuffner, Jr., "Rapidly-exploring random trees: progress and prospects," in *WAFR*, 2000.

[15] T. Bretl, S. Lall, J.-C. Latombe, and S. Rock, "Multi-step motion planning for free-climbing robots," in *in WAFR*, 2004, pp. 1–16.

[16] J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen, "Disconnection proofs for motion planning," in *IEEE Int. Conf. Rob. Aut.*, 2001.

[17] A. F. van der Stappen, D. Halperin, and M. H. Overmars, "The complexity of the free space for a robot moving amidst fat obstacles," *Comput. Geom. Theory Appl*, vol. 3, pp. 353–373, 1993.

[18] ——, "Efficient algorithms for exact motion planning amidst fat obstacles," 1993.

[19] L. Zhang, Y. J. Kim, and D. Manocha, "Efficient cell labelling and path non-existence computation using c-obstacle query," *Int. J. of Robotics Research*, vol. 27, no. 11-12, pp. 1246–1257, 2008.

[20] L. Zhang, Y. Kim, G. Varadhan, and D. Manocha, "Fast c-obstacle query computation for motion planning," in *IEEE Int. Conf. Rob. Aut.*, May 2006, pp. 3035–3040.

[21] G. Varadhan and D. Manocha, "Star-shaped roadmaps - a deterministic sampling approach for complete motion planning," in *Robotics: Science and Systems*, 2005.

[22] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 338–353, 2005.

[23] H. Edelsbrunner and J. L. Harer, *Computational Topology. An Introduction.* Providence, RI, USA: Amer. Math. Soc., 2010.

[24] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," University of British Columbia, Vancouver, BC, Canada, Tech. Rep., 1981.

[25] N. P. Weatherill and O. Hassan, "Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints," *Int. Journal for Num. Methods in Eng.*, vol. 37, no. 12, pp. 2005–2039, 1994.

[26] P. Cignoni, C. Montani, and R. Scopigno, "Dewall: A fast divide and conquer delaunay triangulation algorithm in ed," *Computer-Aided Design*, vol. 30, no. 5, pp. 333 – 341, 1998.

[27] S. M. LaValle, *Planning algorithms*. New York, NY: Cambridge University Press, 2006.

[28] L. Kettner, S. Pion, and M. Seel, "Profiling tools timers, hash map, union-find, modifiers," in *CGAL User and Reference Manual (v3.8)*. CGAL Editorial Board, 2011.

[29] L. Zhang, Y. Kim, and D. Manocha, "A hybrid approach for complete motion planning," in *IEEE/RSJ Int. Conf. Int. Rob. Sys.*, 2007.

[30] T. K. F. Da and M. Yvinec, "3D alpha shapes," in *CGAL User and Reference Manual (v3.8)*. CGAL Editorial Board, 2011.

[31] G. van den Bergen, "Proximity queries and penetration depth computation on 3d game objects," in *Game Developers Conf.*, 2001.