Continuous Unfolding of Polyhedra – a Motion Planning Approach

Zhonghua Xi and Jyh-Ming Lien

Abstract—Cut along the surface of a polyhedron and unfold it to a planar structure without overlapping is known as Unfolding Polyhedra problem which has been extensively studied in the mathematics literature for centuries. However, whether there exists a continuous unfolding motion such that the polyhedron can be continuously transformed to its unfolding has not been well studied. Recently, researchers started to recognize continuous unfolding as a key step in designing and implementation of self-folding robots. In this paper, we model the unfolding of a polyhedron as multi-link tree-structure articulated robot, and address this problem using motion planning techniques. Instead of sampling in continuous domain which traditional motion planners do, we propose to sample only in the discrete domain. Our experimental results show that sampling in discrete domain is efficient and effective for finding feasible unfolding paths.

I. INTRODUCTION

Designing self-folding machine [1, 2, 3] that can resume or approximate a target shape requires careful foldability analysis. Foldability analysis usually involves two steps. The first is unfolding, namely how one can cut the 3D surface of the target shape (polyhedron P) so that the remainder can be unfolded into a single flat and non-overlapping structure. Edge unfolding is a special case that cuts are only allowed along the edges of P. These problems have been extensively studied in mathematics literature even back to 1525 [4]. However, whether every convex polyhedron has an edge unfolding is still open, while non-convex shapes have been shown that are not always have such an unfolding. The second step, which is often overlooked, involves folding the flat sheet along its hinges back to the original 3D surface, while avoiding stretching, bending and self-intersection.

This paper mainly focuses on the second step. We model the unfolding as a multi-link articulated robot and solve the foldability issue using motion planning techniques. First, we propose a tessellation independent unfolding heuristic called Regularized Unfolding which generates identical unfoldings among polyhedra with different tessellations but who share the same geometry. The proposed method allows unfoldings to reuse unfolding motions. For a polyhedron with n faces, its unfolding has n – 1 hingers which equals to the degree-of-freedom (DOF) of the system or the dimensionality of the configuration space. Planning motion in such high dimensional space is nontrivial. We use the idea from [5], in which instead of sampling in continuous domain, we sample in the discrete domain. In our experiments, we show that Discrete Sampler has much higher probability to generate a valid and critical (e.g., on the endpoints of a narrow passage) configuration while the probability of generating a valid configuration in higher dimensional space is extremely low for traditional sampling methods. We also propose a novel connector called Ordered Connector to generate a more human interpretable unfolding motion. With these new sampler and connector, we propose a motion planner for unfolding the polyhedron under the LazyPRM framework [6], which successfully unfold many types of polyhedra.

The main contributions of this paper include a new heuristic method which can generate tessellation independent unfoldings whose unfolding motions can be reused interchangeably and a simple but efficient motion planner that samples from discrete domain. This planner allows us to find feasible folding path while existing methods failed to find any path especially in higher dimensional space. These methods are discussed in detail in Section IV. To better understand the reasons behind the efficiency, we study properties of C-Space of unfolding problems. We show that the C-Space is sparse but configurations sampled from discrete domain are highly like to be valid even in high dimensional space. We also show that the C-space is linear connectable. These results can be found in Section V.

II. RELATED WORK

Unfolding Polyhedra Both edge unfolding and general unfolding of polyhedra have been studied extensively in the mathematical literature, here we refer interested readers to this short survey paper [4]. For the edge unfolding case that we are interested in, Schlickenrieder [7] proposed 19 different heuristics for unfolding a 3D polyhedron to a net (unfolding without overlapping). One heuristic called Steepest Edge Unfolding is shown to be able to unfold arbitrary convex polyhedra with 100% probability. [8, 9] extend [7] to unfold non-convex polyhedra, however, it becomes
much harder to generate non-overlapping unfoldings for non-convex shapes in a single connected piece. Thus, they either use splitting or post merging to ensure that the unfolding does not overlap while trying to minimize the number of pieces in the final unfolding. If cuts are not restricted on edges, star-unfolding [10] and source-unfolding [11] have been proved to be able to unfold all polyhedra into a single connected and non-overlapping piece. All aforementioned works generate non-overlapping unfoldings as final results, however, whether there exists a continuous unfolding motion from the polyhedron to its unfolding are not considered in their works.

Planning Unfolding Motion In order to make a physical copy of self-folding robot that can be folded back to its original shape from the non-overlapping unfolding, we need to find a feasible folding path that can bring the unfolding to its target shape without self-intersection. Recently, Demaine et al. [12] proposed continuous unfolding algorithms for all convex polyhedra. They also proved that the source unfolding [11] of any convex polyhedra can be continuously unfolded without self-intersection. With their mathematically proved methods, continuous unfolding path can be efficiently constructed in polynomial time, however, computing a source unfolding is nontrivial. Both algorithms require to cut at arbitrary location on the surface of the polyhedron. Further more, their methods only work on convex polyhedra. These limitations make it hard to be applied in practice. Tachi, Xi and Lien [13, 14, 15] proposed numerical methods for finding the folding/unfolding motion for rigid origami. However, these methods either not consider self-intersection or take advantage of symmetry property of the crease pattern to reduce the dimensionality of the C-space. Rigid origami is known to has more constraints (even over-constrained) which usually has only one degree of freedom (DOF), thus these method can not be directly applied to unfold polyhedra. Song and Amato’s work [16] studied a problem closely related to those in this paper. They also proposed a PRM-based planner to plan folding motion for tree-structure origami, however, their work mainly take the advantage of the linear connectable property of the C-Space without notice it (e.g., the start state and goal state are directly linear connectable). Therefore, their method does not scale to large and complicated unfoldings whose start state and goal state cannot be connected directly.

III. UNFOLDINGS OF POLYHEDRA

In this section, we briefly introduce how to cut a polyhedron to generate the unfolding, and propose a method to unfold tessellated polyhedra which can be combined with any other heuristics. Note, in this paper we are not focusing on generating the unfoldings for (non-)convex shapes so we assume it can be obtained externally. We are more interested in finding the continuous unfolding motion.

A. Dual Graph and Unfolding of the Mesh

Let \( M \) be a mesh, the graph of mesh is defined as \( G(M) = (V,E) \), where \( V \) = \{vertices of \( M \)\} and \( E \) = \{edges of \( M \)\}. The dual graph \( G'(M) \) of \( G(M) \) is defined as \( G'(M) = (V',E') \), where \( V' = \{faces of \( M \)\} \) and \( E' = \{(u,v)\} \), where \( u, v \) are faces and share an edge in \( M \).

The unfolding can be obtained by finding a spanning tree of the dual graph \( G'(M) \) [7]. Folding hinges will be those edges that are crossed by dual edges in the spanning tree. An example of dual graph of a mesh and its unfolding is shown in Fig. 2.

Since we do not know whether there exists an unfolding that has no overlapping until we try all possible spanning trees. However, it shows that for a mesh with \( F \) faces, there are \( O(2^\sqrt{F}) \) possible unfoldings [8] which can not be examined exhaustively in practice.

B. Heuristic Methods

In order to find a non-overlapping unfolding in polynomial time, many heuristic methods were proposed in the literature [7, 8]. The idea is to assign different weights on the dual edges such that the unfolding generated by the minimum spanning tree of the dual graph will have a higher probability that it has no overlapping. The running time of generating an unfolding is dominated by finding the minimum spanning tree of the dual graph which runs in \( O(|E|\log|E|) \), where \( |E| \) is the number of edges in the mesh.

C. Unfold Tessellated Polyhedra

If the faces of polyhedron are tessellated or densely triangulated, though it still represents the same geometry (surface), however, the topology is different which will lead to have a different unfolding. In order to address this problem, we propose a method called Regularized Unfolding, the idea is not to cut those flat edges (edges with folding angles equal to 0) to reduce the impact caused by topological changes. The simplest way to do that is to assign infinite weights on those flat edges thus they are less likely to be selected as cut edges during the minimum spanning tree procedure. Regularized Unfolding will generate identical or similar unfoldings on different tessellations obtained from the same polyhedron surface. An example of tessellated polyhedra with their Regularized Unfoldings is shown in...

---

**Fig. 2.** Left: Mesh and its dual graph, Right: Spanning tree of the dual graph and the unfolding of the mesh.
Fig. 3. If two unfoldings are identical (polygonal boundary of the unfoldings are the same), then the folding motions can be reused interchangeably.

IV. PLANNING UNFOLDING MOTIONS

A. Unfolding Model

We model the unfolding of a polyhedron as a multi-link tree-structure articulated robot, where folding positions (shared edges in the unfolding) are modeled as hinges and faces are modeled as rigid panels. For a polyhedron with \( n+1 \) faces, its unfolding contains \( n \) hinges: \( \{h_1, h_2, \ldots, h_n\} \). We use all the folding angles of the hinges to represent a configuration. The folding angle of hinge \( h_i \) is the complementary of the dihedral angle of two adjacent faces that connected by \( h_i \). We use \( \rho_i \) to represent the initial folding angle (folded state), and \( \theta_i \) to represent the current folding angle of \( h_i \), then the configuration can be represented as \( C = \{\theta_1, \theta_2, \ldots, \theta_n\} \). And we have two special configurations \( S = \{\rho_1, \rho_2, \ldots, \rho_n\} \) and \( G = \{0,0,\ldots,0\} \) which represent the initial state (polyhedron) and the goal state (unfolding) respectively.

B. Samplers

Sampling based motion planners like PRM [17] and RRT [18] are widely used nowadays since they show the advantage over exact methods especially in higher dimensional space or when the configuration space obstacles are hard or impossible to construct.

1) Continuous Sampler: The simplest but widely used sampling method in motion planning is uniform sampling: a configuration is uniformly sampled from the entire continuous configuration space. For the unfolding problem, we restricted the range of a folding angle to \([−\pi, \pi]\) to avoid local self-intersection so that adjacent faces will not penetrate each other.

2) Discrete Sampler: Contrary to Continuous Sampler, Discrete Sampler only samples the folding angle for a hinge \( h_i \) from its important angle set: \( \{0, \rho_i, \pi\} \) which are corresponding to the flat state, target state and fully folded state. By assuming the unfolding motion is monotonic (all the folding angles move towards 0 monotonically and neither hinge needs to be fully folded as an intermediate state in order to pass some narrow passage), the important angle set can be reduced to \( \{0, \rho_i\} \). Thus the total number of unique configurations for the unfolding with \( n \) hinges is \( 2^n \). A hashtable of bitset can be used to filter out duplicated samples in \( O(1) \).

C. Connectors

Connecting two valid configurations, e.g., planning a local path from configuration \( C_1 \) to \( C_2 \), usually dominating the entire planning time since it requires feasibility tests to avoid self-intersection and maintain certain properties of the system, for example, rigidity for rigid origami. These feasibility tests are computationally expensive, thus we want to avoid checking the connectivity between two configurations that are unlikely to be connectable. A good distance metric plays an important role in motion planning, connectable configurations are close to each other while far away configurations are unlikely to be connectable. Thus we can quickly filter out those far away configuration pairs. Unfortunately, there does not exist a good distance metric for all the unfolding problems and widely used Euclidian distance quickly become less meaningful when DOF increases.

1) All-pairs Connector: Connectivity of all pairs of configurations will be tested, which captures the maximum possible connectivity of the configuration space.

2) Ordered Connector: A novel connector is proposed for unfolding. We assume the unfolding motion is monotonic, all the folding angles decrease from \( \rho \) to 0 monotonically during the entire unfolding process. Connectivity will be tested only if the edge holds the following conditions. a) Before unfolding a hinge, all its children hinges (in the spanning tree) should be fully unfolded first. b) Only one hinge can be unfolded at each step, which is similar to the TreeUnroll method in [12]. The unfolding path found by Ordered Connector will be much more human interpretable since it only unfold one hinge at each time.

D. Motion Planner

The method Continuous Unfolding Polyhedron (Alg. 1) operates on initially empty roadmap graph \( R \) under the LazyPRM framework. First step is try to connect start configuration \( S \) and goal configuration \( G \) directly by linearly interpolating the intermediate configurations. If they are connectable, then we found the path. In our experiments we found that some unfoldings especially those of convex polyhedra have a linear path from start to goal even in high-dimensional space. And
the computation cost for test a single edge is relatively low, thus we always perform this check first.

If start and goal are not directly connectable, then we will plan the motion in the following manner. In each batch, we first use the sampler to sample certain amount of samples, and then use connector to connect them to $\mathcal{R}$ by adding edges, all the edges are not validated until really needed. Then a path $\Pi$ is extracted from $\mathcal{R}$, if $\Pi$ is valid then we found the unfolding path, otherwise a new path will be extracted again until there is no such a path. After that a new batch will be started. This process is repeated until a path was found or certain amount of samples were generated or time limit was exceeded.

Algorithm 1 Continuous Unfolding Polyhedron

Input: Polyhedron $\mathcal{P}$ and its Unfolding $\mathcal{U}$. 
Output: Self-intersection free path that unfolds $\mathcal{P}$ to $\mathcal{U}$.

1: while True do
2:     for each $i$ in range(0, SampleSize) do
3:         Sampler.sample()
4:     Connector.connect()
5:     end for
6:     while $\exists \Pi \in \mathcal{R}$ do
7:         if IsValidPath($\Pi$) then
8:             return ConstructPath($\Pi$)
9:         else
10:             $\mathcal{R} = \mathcal{R} \backslash \{e \mid e \in \Pi \land \neg IsValidEdge(e)\}$
11:         end if
12:     end while
13: end while

V. EXPERIMENTAL RESULTS

A. Experiment Setup

We implemented the proposed motion planner in C++, which will be released to the public domain after this work is published. All data reported in this paper were collected on a 2012 MacBook Pro with a 2.9 GHz Intel Core i7 CPU and 16GB Memory running Mac OS X. We have implemented four heuristic unfolding methods: Steepest Edge, Flat Spanning Tree and Minimum Perimeter proposed in [7] and Regularized Unfolding proposed in this paper to obtain the unfoldings of polyhedra for our experiments. All polyhedra used in this experiment are shown in Fig. 4 with their non-overlapping unfoldings.

B. Samplers: Continuous V.S. Discrete

We conduct the experiment for comparing the performance of two samplers, Continuous Sampler and Discrete Sampler, with respect to ability to generate valid configurations which is measured as the number of valid configurations generated. Average number of valid samples of 20 runs are shown in Table I, from which we can see though tree-structure unfoldings are less constrained than rigid origami as mentioned before, sampling in discrete domain still enable us to find more valid configurations especially in high dimensional space. We can see from Table I that the valid sample ratio of Continuous Sampler drops rapidly from around 40% in 7D space to only 0.06% in 43D space while Discrete Sampler has that ratio over 70% in 7D space and still maintains 58% in 43D space which means it has much higher scalability. It is worth to note that all valid configurations in discrete domain are unique, and many of them lay on the endpoints of the narrow passages.

C. Path Planning Time

We compare the path planning time of Continuous and Discrete Samplers using All-pairs Connector. In this experiment, the maximum sample size was set to 100000 (100 samples each batch, 1000 batches) and time limit was set to 1800 seconds. Average results of 20 runs are shown in Table II, from which we can see that compare to continuous domain sampling, sampling in discrete domain enable us to have higher success rate (Continuous Sampler failed to unfold Periscope2 in all trials) and path planning time can be reduced significantly.

Though planning unfolding motions for those models shown in Table II can be regarded as hard problems for motion planners due to their high dimensionality, however, from Table II we can see that 250 valid configurations are sufficient to find feasible paths for all the unfoldings we tested even using Continuous Sampler (though it might not be easy for it to generate that much). And surprisingly, all paths only contain a few (usually one) intermediate configurations. Here we show the unfolding sequence found by the Discrete Sampler in Fig. 5 with one intermediate configuration whose corresponding shape is show in Fig. 5(d).

D. Linear Connectivity of the Configuration Space

The reason why we did not report the result of Octahedron and Cube model in Table II is that both of them have a
linear path that connect start and goal directly, which makes the problem much easier. That might be the reason why [16] categorized folding a 31 DOF convex polyhedron as an easy problem without explanation while they marked another 5 DOF box as a hard problem. From the data they reported the number of nodes in the biggest connectable component is 2 for the polyhedron model which means among all valid samples, only start and goal are linearly connected while other samples are not able to connect to any of the rest samples.

In order to study the connectivity of the configuration space via only those configurations sampled from discrete domain, we further obtained the complete connectivity graph of Octahedron model in the entire discrete domain by trying all pairs of valid configurations. In the connectivity graph it has 126 nodes (only 2 configurations are invalid in the discrete domain), and 7731 out of 7875 edges (98.17%) are linearly connected. All the nodes form one and only one connected component in the graph. We can say that the configuration space is sparse, however, it is likely to be linearly connectable via those valid configurations sampled from discrete domain.

Whether an unfolding has a linear path that directly connects start and goal is usually unknown, and it mainly depends on the shape and topology of the unfolding. Unfoldings of convex polyhedra are more likely to have this property. However, we also found that some non-convex polyhedra also have this property, in Fig. 1 we show such a linear unfolding path of a non-convex shape.

E. Continuous Blooming of Convex Polyhedra

For all the convex polyhedra (including those shown in Fig. 4 but not limited to) that we have tested, we found that the unfoldings generated by using Steepest Edge heuristic can always be continuously bloomed while other heuristics do not have this property. The monotonic unfolding motion is obtained by linearly interpolating from start state to goal state, which can be considered as a simpler method compare to those proposed in [12] that requires the unfolding to be a source unfolding otherwise additional cuts need to be added to refine the unfolding before the continuous motion could be found. The intuition behind that is blooming of the flowers, since the unfoldings are very similar to bloomed flowers while the convex shapes like unbloomed ones. We show such a continuous blooming of a sphere in Fig. 6. Schlickenrieder [7] conjectured that Steepest Edge heuristic can generate a non-overlapping edge unfolding for every convex polyhedron. Here we extend the conjecture that every convex polyhedron can be continuously bloomed to a edge unfolding generated by the Steepest Edge unfolding.

![Fig. 6. Continuous blooming of a 161 DOF sphere. Unfolding path found by the proposed method within 2.5 seconds.](image-url)
Though previous methods are efficient in finding feasible paths, the unfolding motions found by those methods are hard to be understood by human since multiple hinges fold simultaneously. Ordered Connector, on the contrary, only allows to unfold one hinge at each step, and we assume children faces in the spanning tree need to be unfolded first before a parent face could be unfolded. The unfolding path found by it is ordered and thus more human-interpretable. We show the result of using Ordered Connector to unfold a 11 DOF cube in Fig. 7.

![Unfolding sequence generated by Ordered Connector for a 11 DOF Cube Model.](image]

**VI. CONCLUSION**

In this paper, we study the problem of continuous unfolding polyhedra using motion planning techniques. Sampling in the discrete domain enables us to efficiently generate not only valid but also critical intermediate configurations to help the planner effectively find feasible paths while uniform sampling in the continuous domain failed to do so. And with those valid configurations sampled from discrete domain, we study the property of the configuration space of unfoldings and find that it is sparse but linearly connectable. We also propose the **Ordered Connector** to generate more human interpretable unfolding motions. Though the proposed method successfully unfold several Polyhedra that existing methods failed to unfold, it is still challenge to unfold large and complicated non-convex polyhedra in general if the start and goal states are not directly linear connectable.

**REFERENCES**


