Planning via Convex Transformations

Nima Rahnemoon, Xiyuan (Cyrus) Liu, Siddharth Raina, Shreshta Shetty

The Algorithm

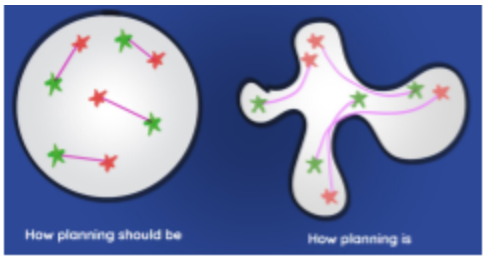

Figure 1: Planning in convex shapes, as shown in the right, simply involves drawing a line between start and goal. The complexities of planning arise when planning in non-convex shapes (left).

This project proposes a method for planning on Eucliean-like spaces of any dimension. With regards to the scope of what was implemented we limited the Euclidean-spaces to $\Re^2$. The algorithm is based on the premise that finding the optimal plan in convex-shaped maps is simply the straight line path between start and goal. Most of the challenges in planning in Euclidean spaces are due to the non-convexity of the planning space, resulting in curved paths, as shown in Figure 1.

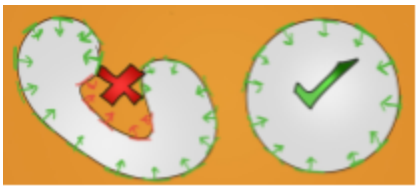

Figure 2: A shape is convex if all the normals to the boundary points point inside the shape.

What defines a convex shape? As shown in Figure 2, a convex shape is one in which the normals at each point on the map's boundary point inside the shape. The objective of our planning algorithm is quite simple. As shown in Figure 3, our algorithm involves two steps: (1) transforming the object from a non-convex shape to a convex shape and storing the mapping and then (2) planning in the convex shape and using the mapping to convert the plan back to the non-convex shape. Note that once the transformation is made, the mapping between shapes may be used for efficiently calculating any number of plans between start and goal.

Figure 3: Our planning approach consists of two steps: transformation and planning.

Related Work

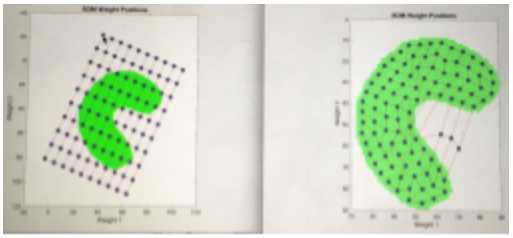

Figure 4: Using self-organizing maps to transform a grid to a non-convex shape produces imperfect results.

While researching existing work in this area, we were not able to find any work that followed the two step process of transforming a map from a non-convex shape to a convex shape, planning in the transformed map and converting back to the original map. This approach appears to be novel. In terms of techniques we considered, we first explored using self-organizing maps [1] to create the mapping. Self-organizing maps take a grid input and a desired shape and then pass this information through a neural network which tries to shape the input such that it matches the desired shape. The only reference we found for SOM being used for path planning was [2], but it was used for feature detection instead of map transformations. The limitation of this approach is that it does not guarantee that 100% of vertices are inside the desired shape, see results in Figure 4.

We then explored a number of methods in Riemannian geometry to perform the transformation. The first method we examined was Ricci Flow. Described in a 2003 paper by Grigori Perelman, Ricci Flow produces a transformation that is conformal and angle-preserving. Conformal means adjacency of areas is maintained through the transformation; i.e., if two areas are adjacent in the non-transformed shape, they will be adjacent in the transformed shape as well. Angle-preserving means that if two edges have an angle θ between them, they will likewise have an angle θ between them unless otherwise specified. Ricci Flow uses linear programming to keep the angle-preserving and conformal

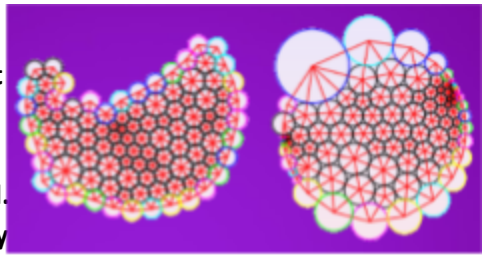

Figure 5: Ricci Flow finds a transformation that is conformal and angle-preserving. Left is the original non-convex shape, right is the transformed convex shape.

constraints to solve the transformation. An example transformation using Ricci Flow is shown in Figure 5. Note that all the internal shapes are circles in the original shape and all the shapes are still circles in the transformed shape -- this is because Ricci Flow is angle-preserving so it maintains shapes. In 2013, Zhao and Gu published a modified version of Ricci Flow, [4], that solves Minowski's Optimal Mass Transport problem (OMT). Namely, instead of preserving angles, OMT preserves area for surfaces and volumes for closed surfaces (3D objects). Figure 6 shows the results of Ricci Flow and OMT side by side. Because OMT preserves areas it produces convex spheres in 3D and convex circles in 2D that are better-suited for path-planning. Ideally, there would be a transformation that would be distance-preserving; however, such a transformation currently doesn't exist. Nevertheless, as the resolution of the grid increases (meaning, the area containing in each grid approaches 0), the area-preserving transformation becomes distance-preserving.

Figure 6: Ricci Flow preserves angles whereas OMT preserves volume/area. As such, OMT is better suited for path planning applications.

Implementation

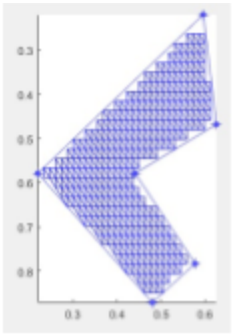

Figure 7: Matlab program which allows the user to draw the map and then discretizes the map via filling it with triangles.

The math to generate the OMT transformation is very complex and would have been infeasible given the time we had for the project. Fortunately, Dr. Gu, author of [4], has output his code that implements the OMT algorithm, albeit with a few minor bugs. The first step involved allowing the user to input a map shape. Figure 7 shows the process we used to generate the map. We then used Dr. Gu's OMT program to take the newly generated shape and convert it to the transformed shape, as shown in Figure 8. Notice that the transformed map in Figure 8 consists of hexagons. The OMT program outputs an OFF file, the format of which is described in [6], containing the transformed map. We then took the transformed map and planned on it, as shown in Figure 9. Note that the transformed shape in Figure 9 is not conformal -- we believe there's a bug in the OMT code which outputs the OFF file as triangles incorrectly. Nevertheless, despite this bug, we perform some manual checks to ensure the path is conformal and thereby our plans are conformal. Note that the path distance of our plan is 0.478 whereas the optimal plan's distance is 0.4179. This is due to two reasons. First, we discretize our plan so that it snaps to the triangles' centers. This results in curvy suboptimal paths, as shown in Figure 10. Secondly, because OMT is area-preserving, not distance preserving, it does not guarantee that the plan will be optimal. However, as the resolution of the grid increases (i.e., the area of each triangle approaches 0), the plan will become optimal.

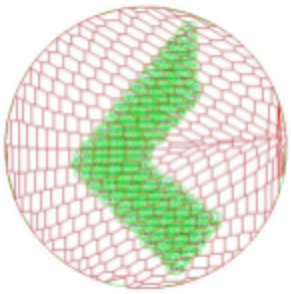

Figure 8: The transformed map after OMT has been run.

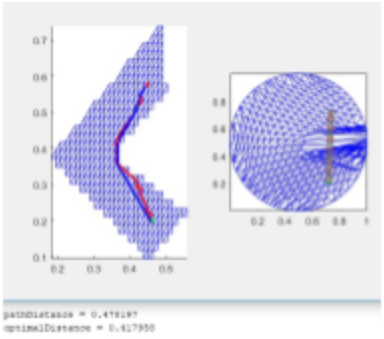

Figure 9: Path planning in the transformed map and converting back to the original non-convex map. Blue is the optimal path, red is our path.

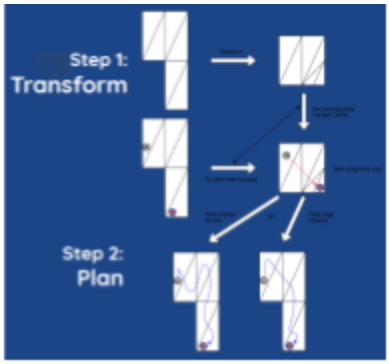

Figure 10: We discretized our plan such that it goes through the centers of the triangles. This simplifies our logic, but produces suboptimal plans.

Future Work

There are a number of other cases for which the OMT algorithm is useful as it relates to path planning which we didn't have time to implement. First, we have considered a method of implementing the OMT transformation on a map that has islands. This technique is briefly described in Figure 11. Figure 12 describes the transformation complexity for such

an algorithm. As shown in Figure 13, the algorithm may also be extended for 3D shapes. Dr. Gu has code on his website that does volume-preserving OMT for 3D objects.

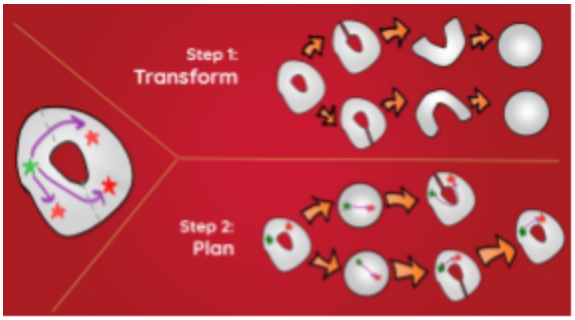

Figure 11: To plan in maps that have islands, invisible walls need to be drawn 180 degrees apart from the island to the walls. This generates two separate maps that have to be transformed separately. Finally, planning may take place in both maps and then the shortest path produced is returned as the final optimal path.

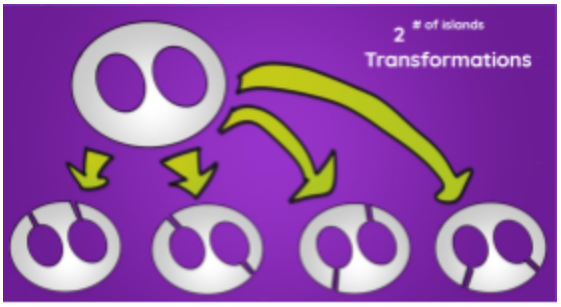

Figure 12: Planning in a shape that has multiple islands requires there to be 2^(# of islands) different maps that have to be transformed. Planning takes place in all the maps and then the shortest path would be returned.

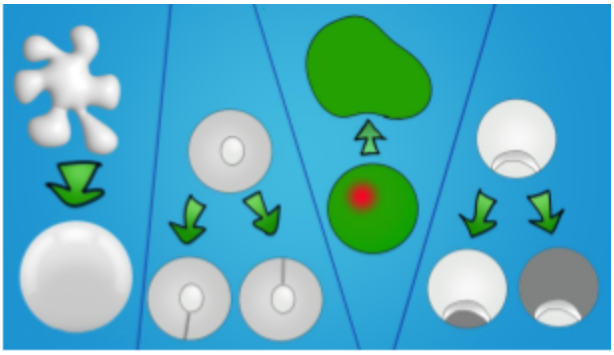

Figure 13: From left to right: (1) OMT may be used to generate a sphere from any non-convex 3D shape, (2) Similar to islands in the 2D case, if there is a 3D obstacle in the middle of the map, two separate infinitely thin columns must be added 180 degrees apart to generate two separate 3D transforms, (3) OMT may be applied on cost maps by first expanding the area of the map such that the costs are captured in the Euclidean distance, and (4) 3D shapes that have arches must have two separate infinitely thin walls added to generate 2 separate transformations. One thin wall captures the area within the arch, the other is a wall surrounding the arch.

Paper References

1. Teuvo Kohonen, The self-organizing map, In Neurocomputing, Volume 21, Issues 1–3, 1998, Pages 1-6. (source)
2. Ishii and K. Yano, "Path planning system for a mobile robot using self-organizing map," 2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479), Beijing, 2001, pp. 32-37 vol.4. (source)
3. Perelman, Grisha. The entropy formula for the Ricci flow and its geometric applications. 2002. (source)
4. X. Zhao et al., "Area-Preservation Mapping using Optimal Mass Transport," in IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 2838-2847, Dec. 2013. (source)
5. Zeng W, Gu X. Ricci Flow for Shape Analysis and Surface Registration. Springer New York, 2013. (source)
6. John Burkardt. Florida State University. OFF file format. (source)

Slides/Talks References

The papers are somewhat convoluted. We relied much more on the talks given by Dr. David Guo of Stony Brook. If you would like to learn more about Ricci Flow, we'd recommend [2], and if you'd like to learn more about Optimal Mass Transport, we'd recommend [3].

1. David Gu. Tutorial on Conformal Geometry for Computer Vision. Computer Vision, IPAM, UCLA 2013. (Slides, Talk)
2. David Gu. Tutorial on Surface Ricci Flow for Shape Registration and Analysis, IPAM, UCLA 2013. (Slides, Talk)
3. David Gu. Tutorial on Surface Ricci Flow for Shape Registration and Analysis, IPAM, UCLA 2013. (Slides, Talk)
4. John A. Bullinaria. Self Organizing Maps: Fundamentals. 2004. (Slides)

Code References

1. Our code that performs planning via Matlab given the non-convex OFF file and the transformed convex OFF file (instructions are in the ReadMe). (source)
2. C++ Code for A Simple Mesh Viewer Written. (source)
3. C++ Code for Surface Ricci Flow Tutorial. (source)
4. C++ Code for Optimal Mass Transportation Map Surface Area-Preserving Parameterization; i.e., for 3D surfaces or for 2D spaces (source)
5. C++ Code for Optimal Mass Transportation Map Volume-Preserving Parameterization; i.e., for 3D closed objects -- we didn't use this directly, but we tested it. (source)

Fun Videos

A lot of this material is very dense math; we'd recommend watching the following videos for a high-level introduction into some of the concepts:

1. Grigori Perelman documentary. (source -- a bit long, maybe watch the other videos first)
2. Ricci Flow - Numberphile. (source)
3. Poincaré Conjecture - Numberphile. (source)

Special Thanks

We'd like to thank Tianlong Yu, a mathematics PhD student at CMU who helped guide our work in the right direction. Without his assistance, we wouldn't have converged onto the Ricci Flow and Optimal Mass Transport solutions. If you would like more details about the low-level math, feel free to reach out to Tianlong -- he's done extensive work in this area. (directory)