Project on Surface Parameterization

Xuan Li[1,*]

[1]Department of Computer Science, Stony Brook University
[*]SBU ID: 111676019

ABSTRACT

In this project, I implement three parametrization algorithms: Euclidean orbifold Tutte embedding [Aigerman and Lipman, 2015], hyperbolic orbifold Tutte embedding [Aigerman and Lipman, 2016], and boundary first flattening [Sawhney and Crane, 2017]. The Euclidean orbifold Tutte embedding uses orbifold structure to generalize the algorithm in [Gortler et al., 2006] to sphere-type meshes. Hyperbolic orbifold Tutte embedding generalize the Euclidean orbifold Tutte embedding in the sense of geometry. But this generalization in non-linear. Besides, the Euclidean orbifold Tutte is also claimed to be conformal, so this algorithm provide a new linear method to compute conformal maps. Boundary first flattening is another linear conformal map solver. But this algorithm is based on differential geometry.

1 Theoretical Backgroud

1.1 Euclidean Orbifold Tutte Embedding

The first part of my project is Euclidean orbifold Tutte embedding [Aigerman and Lipman, 2015]. This algorithm is a generalization of so-called Tutte graph embedding based on the following theorem:

Theorem 1.1 *Let $G = \langle V, E, F \rangle$ be a 3-connected planar graph with boundary vertices $B \subset V$ defining a unique unbounded exterior face f_e . Suppose ∂f_e is embedded in the plane as a (not necessarily strictly) convex planar polygon, and each interior vertex is positioned in the plane as a strictly convex combination of its neighbors, then the straight-line drawing of G with these vertex positions is an embedding. In addition, this embedding has strictly convex interior faces.*

The embedding problem is the following full-rank linear system:

$$\begin{aligned}
\sum_{v_j \in N(v_i)} w_{ij} x_j &= x_i, i = 1, ..., |V - B| \\
\sum_{v_j \in N(v_i)} w_{ij} y_j &= y_i, i = 1, ..., |V - B| \\
x_i &= b_i^x, i = |V - B| + 1, ..., |V| \\
y_i &= b_i^y, i = |V - B| + 1, ..., |V|
\end{aligned} \tag{1}$$

where $\sum_j w_{ij} = 1, \forall v_i$, and $w > 0$. The result is guaranteed to be injective.

But this algorithm cannot be generalized onto topological sphere directly. They [Aigerman and Lipman, 2015] use a geometric structure called orbifold, making this method computable on sphere.

An orbifold can be treated as a tiled plane. The tile transformation is isometries. In this paper Euclidean orbifolds are used. We cut the mesh into disk to get a maximal submesh. The maximal submesh can be mapped to a tile of Euclidean plane. So besides Eq.1, we should make the embedding satisfy the following two conditions:

1. The embedding can tail the whole plane seamlessly.
2. The image of every vertex satisfy weighted convex combination on tiled plane.

For example, in Fig.1, not only interior but also boundary vertices of the maximal submesh satisfy convex combination condition on the tiled plane.

The system needs to be solved is:

$$\begin{aligned}
\sum_{v_j \in N(v_i)} w_{ij}(z_j - z_i) &= 0, \quad v_i \in \bar{V} - \bar{B} \\
\sum_{v_j \in N(v_i)} w_{ij}(z_j - z_i) + \sum_{v_j \in N(v_{i'})} w_{i'j} R_{i'i}(z_j - z_i') &= 0, \quad v_i \in \bar{B} - \bar{C} \\
z_i &= z_i^0, \quad v_i \in \bar{C}
\end{aligned} \tag{2}$$

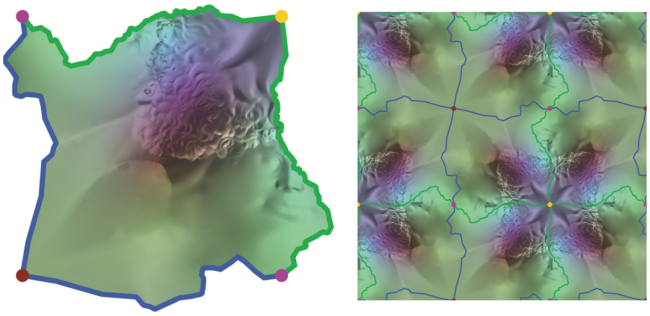

Figure 1. Tiled plane.

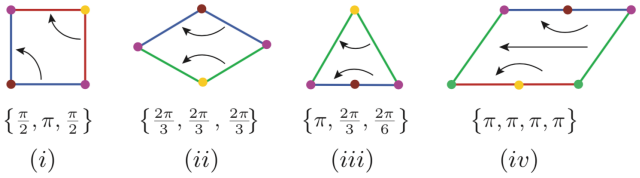

Figure 2. Four kinds of sphere-type Euclidean orbifolds

Here we cut the M into a submesh $\bar{M}$, which is a disk. $\bar{C}$ is the cones of the orbifold. $\bar{B}$ is the boundary of the submesh. And $R_{i'i}$ is the rotation part of the isometry from i' to i.

Note that we have only four Euclidean sphere-type orbifold, shown in Fig. 2. And one intresting property is that the map is actually confomal map.

1.2 Hyperbolic Orbifold Tutte Embedding

The second part of my projects is hyperbolic orbifold Tutte embedding [Aigerman and Lipman, 2016]. Hyperbolic orbifold Tutte embedding is the generalization of Euclidean orbifold Tutte embedding. We treat a hyperbolic orbifold as a tiled Poincare disk by a basic tile. But hyperbolic isometries are non linear. Besides we use a non linear generalization of Tutte embedding:

$$\begin{aligned} &\min_{\Phi} \quad E(\Phi) = \frac{1}{2} \sum_{(i,j)\in E} w_{ij} d(z_i, z_j)^2 \\ &s.t \quad z_i = z_i^0, \quad v_i \in B \end{aligned} \tag{3}$$

where d is the distance function in corresponding geometry. This system is equivalent to Eq.1 in Euclidean geometry.

We use this generalization to get hyperbolic orbifolds:

$$\begin{aligned} &\min_{\Phi} E(\Phi), \\ &s.t \quad z_i = m_{i'i}(z_{i'}), \quad v_i \in \bar{B} - \bar{C} \\ &\qquad z_i = z_i^0, \qquad\qquad v_i \in \bar{C} \end{aligned} \tag{4}$$

Here d is hyperbolic distance function and $m_{i'i}$ is hyperbolic isometry from i' to i.

The distance has explicit expression whose first order derivative is easily computed. So we can use first-order optimization algorithm to optimaize this problem.

A result from paper is showned in Fig.3

1.3 Boundary First Flattening

The third part of my project is boundary first flattening [Sawhney and Crane, 2017].

2 Implementation Details

2.1 Overview

2.2 Dependencies

[Guennebaud et al., 2010] [Jacobson et al., 2016] [Qiu, 2016] [Botsch et al., 2002]

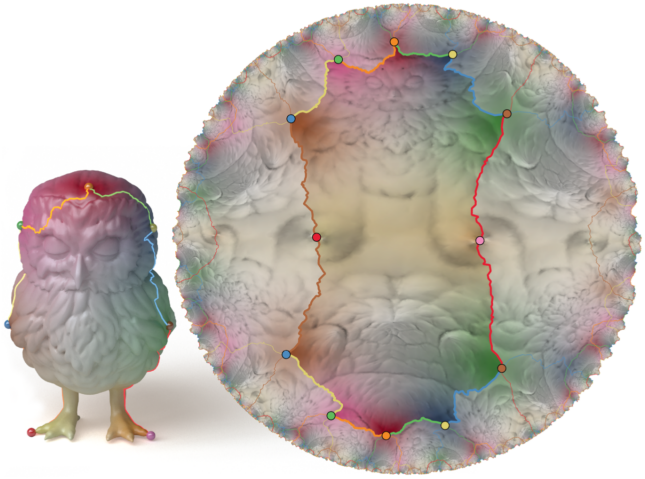

Figure 3. Hyperbolic orbifold Tutte embedding

3 Experiments

References

Aigerman and Lipman, 2015. Aigerman, N. and Lipman, Y. (2015). Orbifold tutte embeddings. *ACM Trans. Graph.*, 34(6):190:1–190:12.

Aigerman and Lipman, 2016. Aigerman, N. and Lipman, Y. (2016). Hyperbolic orbifold tutte embeddings. *ACM Trans. Graph.*, 35(6):217:1–217:14.

Botsch et al., 2002. Botsch, M., Steinberg, S., Bischoff, S., and Kobbelt, L. (2002). Openmesh - a generic and efficient polygon mesh data structure. https://www.openmesh.org.

Gortler et al., 2006. Gortler, S. J., Gotsman, C., and Thurston, D. (2006). Discrete one-forms on meshes and applications to 3d mesh parameterization. *Comput. Aided Geom. Des.*, 23(2):83–112.

Guennebaud et al., 2010. Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

Jacobson et al., 2016. Jacobson, A., Panozzo, D., et al. (2016). libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Qiu, 2016. Qiu, Y. (2016). Lbfgs++. https://github.com/yixuan/LBFGSpp.

Sawhney and Crane, 2017. Sawhney, R. and Crane, K. (2017). Boundary first flattening.