Least Squares Conformal Maps

Putian YUAN
putian.yuan@ip-paris.fr

December 7, 2022

1 Motivation and Problem Statement

Given a surface representation (e.g. vertices and triangles), the task for parameterization is to find a map which converts this surface onto a 2D plane. Parameterization has many applications, such as texture mapping shown in Figure 1. Although there exists some technique such as Mesh Colors which allows artists to paint on 3D mesh directly without mapping, painting on 2D plane is still popular and common used in many fields. As this survey[1] summarized: parameterization almost always introduce a distortion in either angles or areas. A good mapping should minimize this distortion. Many approaches studying on this topic is to define the representation for this kind of distortion (energy), and try to minimize it. Basically there are 3 types of mapping: conformal mapping which preserves angles, equiareal mapping which preservers areas, and combination of these two mapping. My goal of this project is to study and implement one of these mapping technique: Least Squares Conformal Maps[2].

Figure 1: Texture Mapping

2 Summary of LSCMs

LSCMs is a conformal (angle-preserving) mapping and a conformal mapping must satisfy the Cauchy Riemann equations. Starting from it, Lévy et al.[2] rewrote it into a complex form and derived an equation which associates each vertex with a complex number satisfying the conformal condition. Then the distortion of mapped angles becomes a linear equation written in complex form:

$$C(\mathbf{x}) = \|\mathcal{A}\mathbf{x} - \mathbf{b}\|^2$$

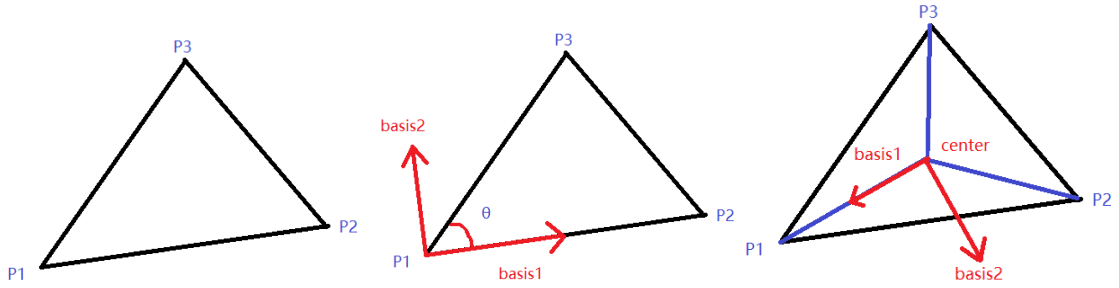
Once we follow the method described in their paper[2] to build and solve such an equation, we can easily get the final uv coordinates from $\mathbf{x}$.

3 Implementation

Following the paper LSCMs[2], I make used of the library **Eigen** to implement the least squares conformal maps method. The library **libigl** is also included in my project but I only use the LSCMs from **libigl** for comparing my results.

3.1 Local orthonormal basis

The LSCMs method supposed each triangle is provided with a local orthonormal basis, where (x_1, y_1), (x_2, y_2), (x_3, y_3) are the coordinates of its vertices in this basis, but they didn't provide the detail for building such basis. In my implementation, I have implemented two methods to build the 2D basis. Considering we have a triangle with three points P_1, P_2, P_3 shown as figure 2(a), we can choose P_1 as origin, edge $\overline{P_1P_2}$ as basis1, and the height to edge $\overline{P_1P_2}$ as basis2, shown as figure 2(b). Alternatively, we can choose the centroid of triangle as origin, edge $\overline{CP_1}$ from centroid to P_1 as basis1, rotate basis1 90 degrees around the *normal* of triangle to get basis2, as figure 2(c) shown. Once we have chosen our basis, computing (x_i, y_i) for each vertex is very easy: we can just project each point onto these two basis. In my experiment, these two methods give the same results.



(a) (b) (c)

Figure 2: local orthonormal basis

3.2 Pinned points

Although LSCMs doesn't demand the mesh boundary to be mapped onto a fixed shape, they mentioned that some points need be pinned (it means the uv coordinates of those pinned points are precomputed or preassigned) in order to have a non-trivial solution for the optimization problem. They also recommended to select two points from the boundary to avoid that triangle flips occur. Therefore, I chose two points which have maximal distance from boundary. For theirs uv coordinates, I provided two ways to set them: one is that we can compute a bounding box of the mesh, and squash the bounding box and two pinned points onto xz plane, then we can compute the uv coordinates, shown as figure 3

Let's assume P_{min}, P_{max} are the minimal point (3D vector) and maximal point of the bounding box, then the pinned uv can be computed:

$$u_i = \frac{(Pinned_i - P_{min}).x}{(P_{max} - P_{min}).x}$$
$$v_i = \frac{(Pinned_i - P_{min}).z}{(P_{max} - P_{min}).z}$$

where $i \in \{1, 2\}$, $Pinned_i$ is pinned point (3D vector), and (u_i, v_i) is its pinned uv coordinates. Alternatively, we can assign them directly. In my experiment, these pinned uv coordinates determine the distortion, and bad pinned points will cause very obvious distortion.

What's more, in the paper, the Hermitian Gram matrix $\mathcal{M}$ is split into two part

$$\mathcal{M} = (\mathcal{M}_f \quad \mathcal{M}_p)$$

which means we should always put the pinned points at the right columns of $\mathcal{M}$. In my implementation, I simply swap these two pinned points with last two points of vertices V in order to achieve

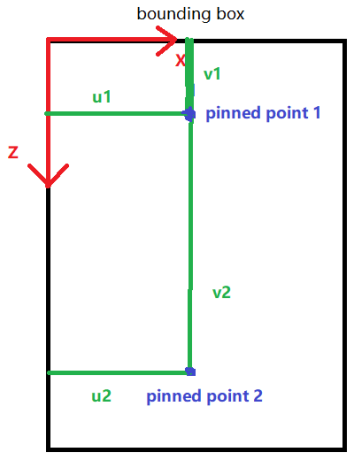


Figure 3: Compute pinned uv

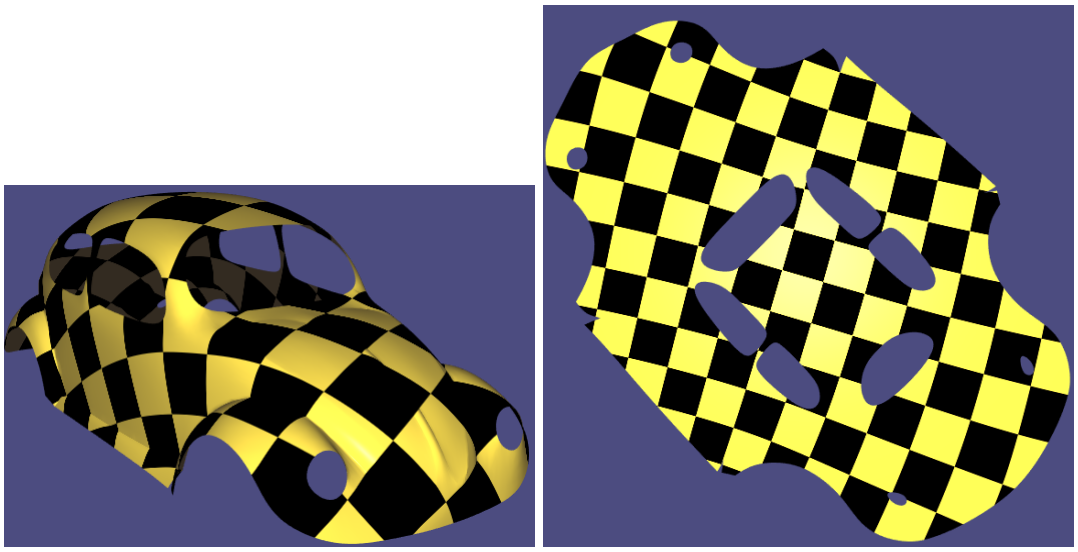
this goal. However, swapping the points of V won't change the indices in its corresponding triangle (facet) T. Therefore I also need to create a mapping to associate them in order to get the correct vertex information when iterating all triangles $\mathcal{T}$.

3.3 Solver choices

In my experiment, I tried the Least Squares Conjugate Gradient solver, and it took me more than 5 minutes to get the result. The reason why it was so slow is that the $\mathcal{A}$ matrix is extremely sparse, where there are only six elements(three complex number in each vertex of each triangle) not being zero at each row. Therefore, I turned to use the simplicial LDLT solver, and use it to solve the equivalent equation $\mathcal{A}^T\mathcal{A}\mathbf{x} = \mathcal{A}^T\mathbf{b}$, which is much faster.

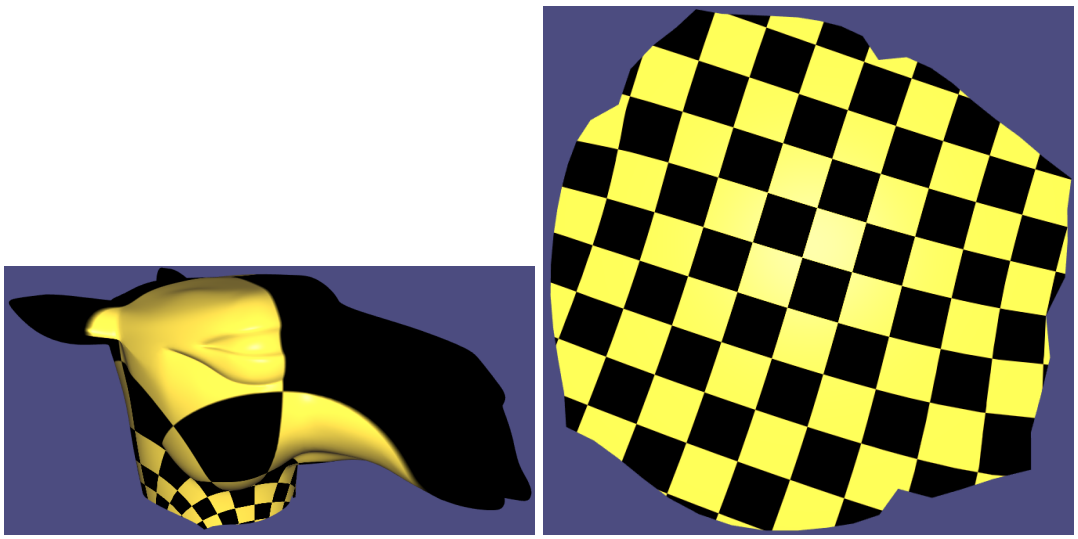
4 Results and Limitation

As mentioned above, LSCMs doesn't require to map the mesh boundary onto a fixed shape (such as circle or rectangle), but they require setting the pinned points before solving the linear equation. The limitation of this method is that we have to choose good pinned points to have a good result because the choice of pinned points can affect dramatically the mapping results. Figures 4(b), 5(b), 6(b) are the mapped results with good pinned points. Here I uses two points from boundary, and they have maximal distance, then set their uv as $(0, 0.7)$, $(0.7, 0)$. Comparing preassigned method, using computed uv from bounding box will not change the distortion, as 7(a) shown. When we choose bad pinned points, the distortion will be very obvious as we can see from figure 7(b). Here I choose the first two points from non-boundary points as pinned points(Choosing the firs two points from boundary will also give a similar distortion).



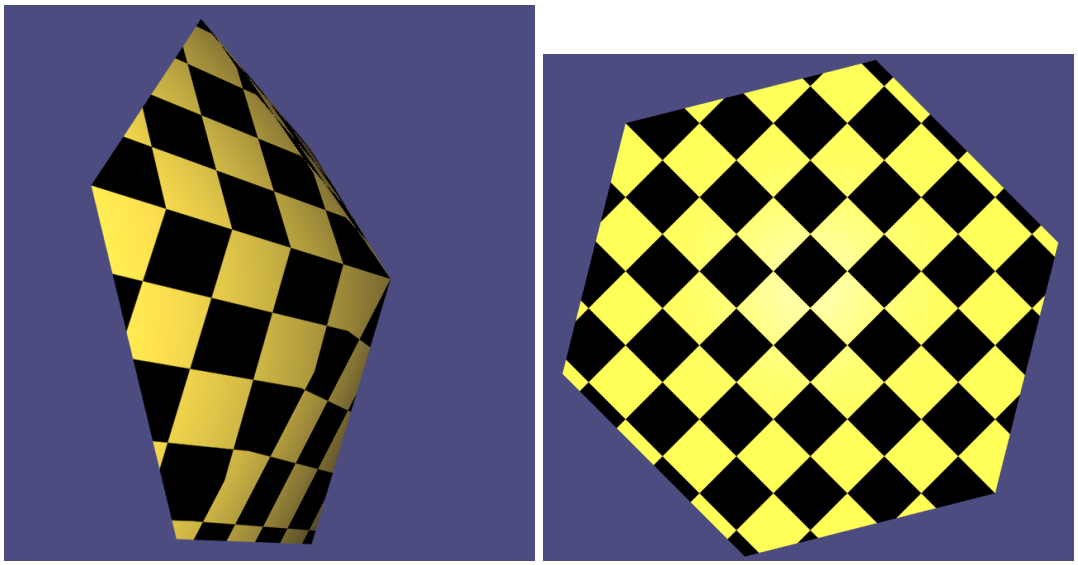
(a) beetle car shell (b) beetle car shell mapping with good pinned points

Figure 4: results 1



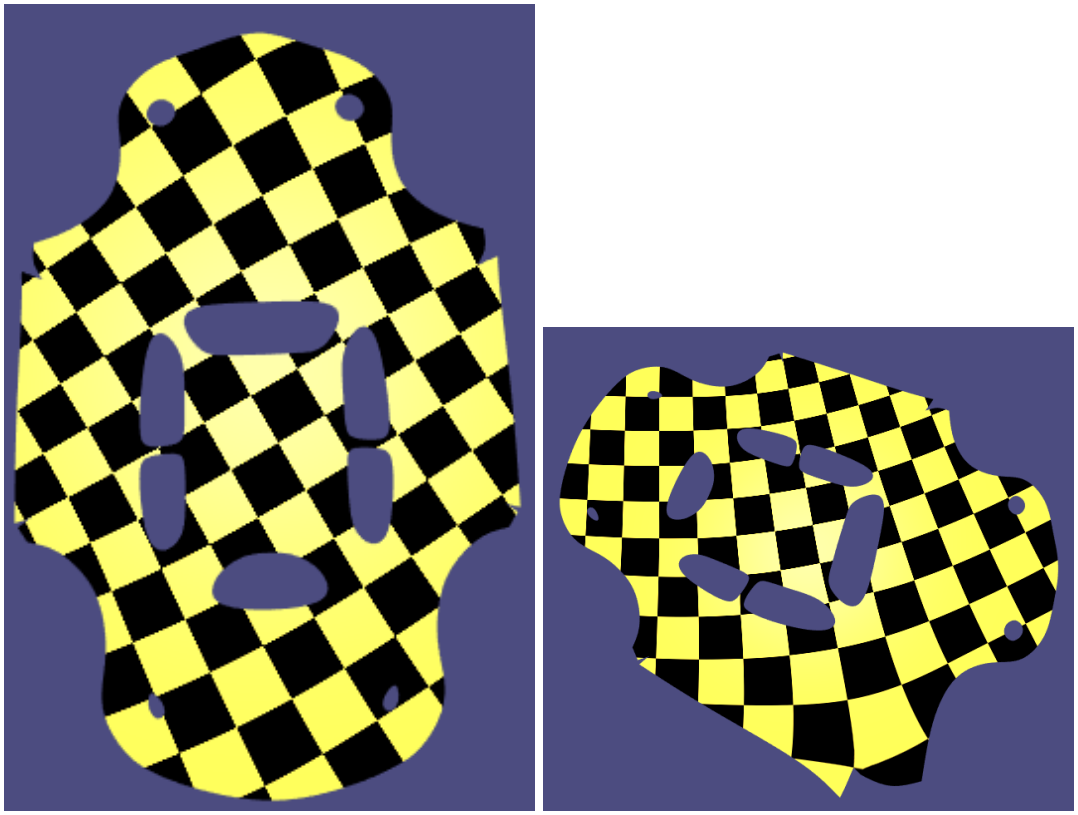
(a) camel head (b) camel head mapping with good pinned points

Figure 5: results 2



(a) hexagon (b) hexagon mapping with good pinned points

Figure 6: results 3



(a) uv of pinned points are computed from bounding box (b) beetle car shell mapping with bad pinned points

Figure 7: results 4

5 Conclusion

In order to implement LSCMs, at first I chose two pinned points, then computed or assigned their corresponding uv coordinates, next built the linear system by setting $\mathcal{A}$ matrix, $\mathbf{b}$ vector following the paper, finally used **Eigen** solver to solve it and construct the final uv results. Except being used in texture mapping, this technique can also be applied in re-meshing and data compression possibly. The most possible direction for it could be high performance parameterize huge models when using LSCMs criterion.

The source code of this project can be found here: https://github.com/pyuan-21/INF574-project.

References

[1] M. S. Floater and K. Hormann, "Surface parameterization: a tutorial and survey," *Advances in multiresolution for geometric modelling*, pp. 157–186, 2005.

[2] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation," *ACM transactions on graphics (TOG)*, vol. 21, no. 3, pp. 362–371, 2002.