DGP 第三次作业：ASAP，ARAP 参数化

SA22001009 陈泽豪

March 31, 2024

1 作业介绍

在数字几何处理和计算机图形学领域，网格参数化是一种将三维表面网格映射到二维平面的过程。这一过程对于纹理贴图、网格编辑、形状分析等应用至关重要。近年来，ASAP（As-Similar-As-Possible）和 ARAP（As-Rigid-As-Possible）参数化技术因其出色的性能和灵活性而受到广泛关注。

ASAP 参数化旨在保持网格局部相似性，使得映射后的网格尽可能保留原网格的形状。这是通过最小化网格顶点移动前后边长比例的变化来实现的。具体来说，ASAP 寻求一种映射，使得每个三角形的形变尽可能接近等距变换，从而减少纹理拉伸和压缩，保持纹理细节。ASAP 参数化特别适用于那些需要保持图案或纹理细节准确度的应用场景。

相较之下，ARAP 参数化强调保持局部刚性，旨在使得映射过程中每个小片网格尽可能保持其原始形状。在 ARAP 模型中，每个三角形都尽量维持其原始的角度，使得整个变形过程中网格的局部刚性最大化。这种方法在处理需要保持物体特征明显和边缘锐利的模型时表现出色，例如机械零件或建筑模型的参数化。

ASAP 和 ARAP 参数化方法的关键优势在于它们的能力，即在保持网格特征的同时实现从三维到二维的平滑映射。这两种技术采用了能量最小化框架，通过迭代求解来寻找最优映射，这使得它们能够处理包括具有复杂拓扑结构在内的广泛网格类型。此外，这些方法通常伴随着边界固定或自由边界的策略，以满足不同的应用需求。

然而，尽管 ASAP 和 ARAP 参数化在许多方面表现出色，它们也面临着挑战，如处理具有大量扭曲或高度非线性变形的网格时的稳定性和效率问题。此外，选择适当的边界条件和初始映射对于获得高质量的参数化结果至关重要。

总的来说，ASAP 和 ARAP 参数化技术为三维网格的二维展开提供了有效的工具，它们在 3D 建模、纹理映射、数字制造和其他许多领域都有着广泛的应用前景。未来的研究将继续探索这些方法的改进，以提高它们的适用性和性能，特别是在处理高度复杂模型时。

2 整体的算法框架

整个代码的实现主体实际上可以分为如下的几步。

2.1 ASAP,ARAP 的主要步骤

对于有边界的三角网格文件，我们可以将三维空间中的三角形投影至二维空间，采用等距投影的方式，因此对任意 $triangle\ t$，在平面上都可以变成 $\{x_t^0, x_t^1, x_t^2\} \subset \mathbf{R}^2$，其中可以直接令

$x_t^0 = [0., 0.]$。然后希望得到所有的二维参数化结果实际上为 $\{u_t^0, u_t^1, u_t^2\}$，那么考虑：

$$\mathbf{L} : \{x_t^0, x_t^1, x_t^2\} \to \{u_t^0, u_t^1, u_t^2\} \tag{1}$$

是一个三角形之间的变换，可以是保角变换，可以是保形变换等等。因此为了让最终的 $\{u_t^0, u_t^1, u_t^2\}$ 与 $\mathbf{L}\{x_t^0, x_t^1, x_t^2\}$ 尽可能的接近，就有如下的能量函数优化问题：

$$E(u, L) = \sum_{t=1}^{T} A_t \| J_t(u) - L_t \|_F^2 \tag{2}$$

$$= \frac{1}{2} \sum_{t=1}^{T} \sum_{i=0}^{2} cot(\theta_t^i) \| (u_t^i - u_t^{i+1}) - L_t(x_t^i - x_t^{i+1}) \|^2 \tag{3}$$

其中 θ_t^i 是 (x_t^i, x_t^{i+1}) 正对着的角度。

2.2 ASAP

对于 ASAP 来说，取 L_t 形如：

$$L_t = \begin{bmatrix} a_t & b_t \\ -b_t & a_t \end{bmatrix} \tag{4}$$

然后直接采用构建矩阵的方式，固定住两个点的 u_t^i 在固定的点处，例如 $(0, 0), (1, 1)$ 处。然后对其他所有的 u_t^i, a_t, b_t 进行求偏导操作，最终得到一个大型的稀疏矩阵等式，直接求解出 u_t^i 最终的值即可。

2.3 ARAP

对 ARAP 来说，取 L_t 形如：

$$L_t = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \tag{5}$$

它是一个迭代的过程，首先得输入一个初始的 u_t^i 作为参数化信息，这里我们可以直接用 ASAP 的结果作为 ARAP 的输入。然后固定住所有的 u_t^i ，求出最佳的 L_t 。然后再固定住 L_t，求出最佳的 u_t^i，这样循环多次得到结果。

求 L_t 的方法，首先令 $J = UX^{-1}$，然后对 J 进行 SVD 分解：

$$J = U_1 \Sigma V_1^T \tag{6}$$

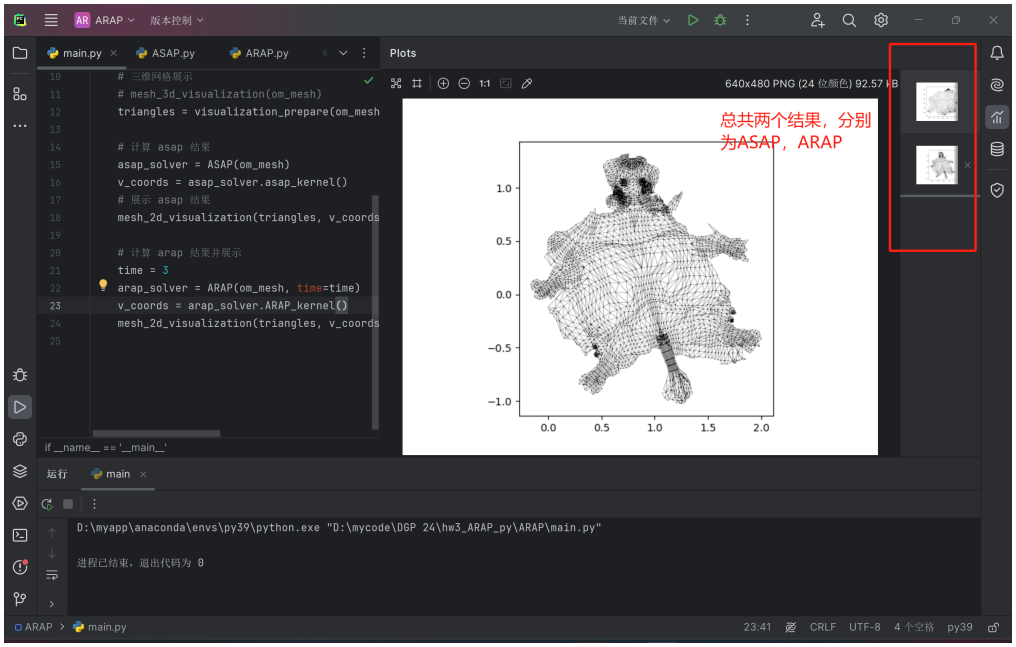
则最终得到的 L 即为：

$$L = \begin{cases} U_1 V_1^T, det(J) > 0 \\ U_1 \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} V_1^T, else. \end{cases} \tag{7}$$

得到 L 之后求 u_t^i 的方式依然是利用偏导建立方程求解。

3 最终结果展示

实现采用 python 3.9.19（openmesh 在 python 里只有 python 3.9 支持），openmesh 1.2.1，numpy 1.26.4，scipy 1.12.0，matplotlib 3.8.0，pyvista 0.43.4. 其中 matplotlib 用来显示二维参数化结果，pyvista 用来显示输入的三维网格图样（pyvista 可能存在与 numpy 的兼容性问题，代码中暂时将这部分注释了）。直接在 main.py 内点击运行即可输出 ASAP，ARAP 参数化结果，如下所示：



3.1 示例一：Cow_dABF

首先输入如下所示的牛：

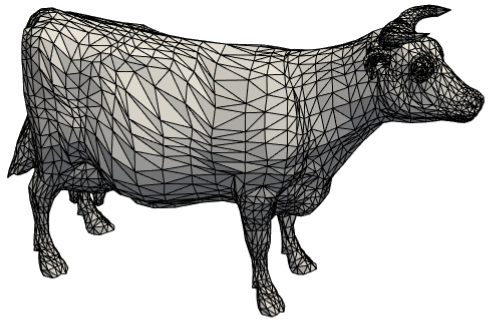


Figure 1: 牛牛

然后得到它的 ASAP 结果，ARAP 结果如下所示，其中 ARAP 迭代三次。

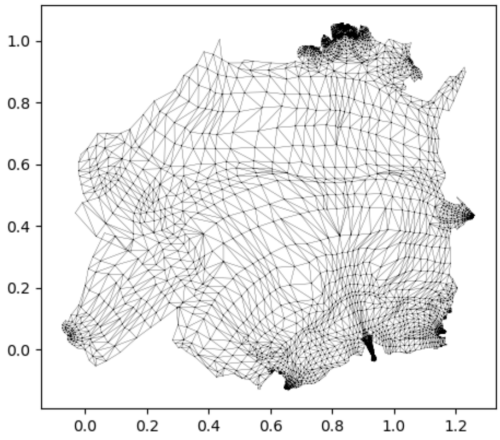


Figure 2: 牛牛 ASAP 结果

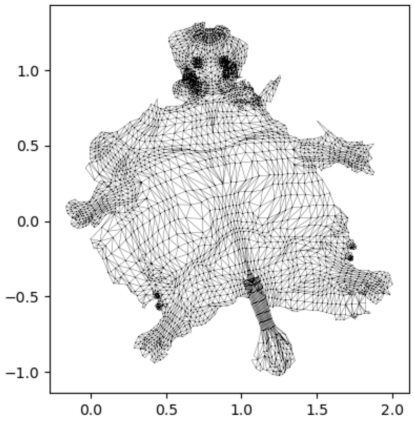
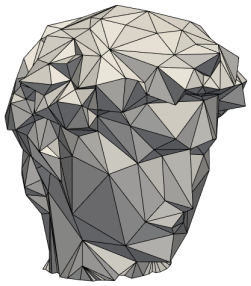


Figure 3: 牛牛 ARAP 结果

3.2 示例二：David

输入如下所示的 David，同时给出它的 ASAP，ARAP 结果：



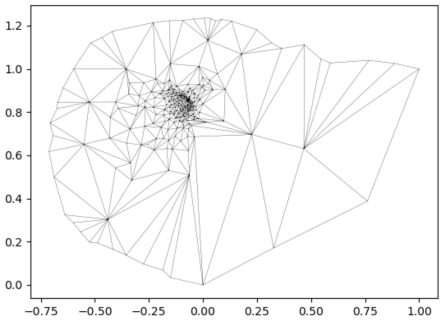


Figure 4: David ASAP 结果

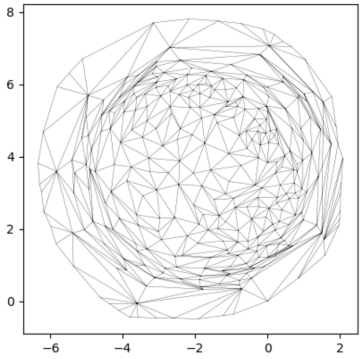


Figure 5: David ARAP 结果

3.3 示例三：Gargoyle

输入如下所示的 Gargoyle，同时给出它的 ASAP，ARAP 结果：

Figure 6: Gargoyle

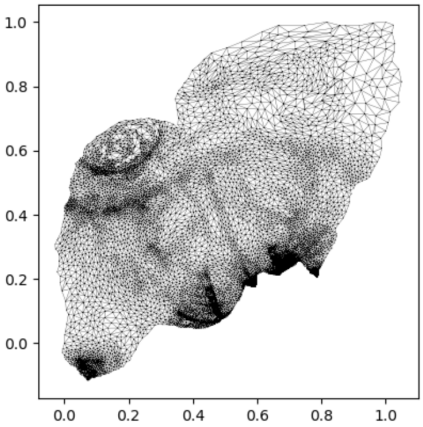


Figure 7: Gargoyle ASAP 结果

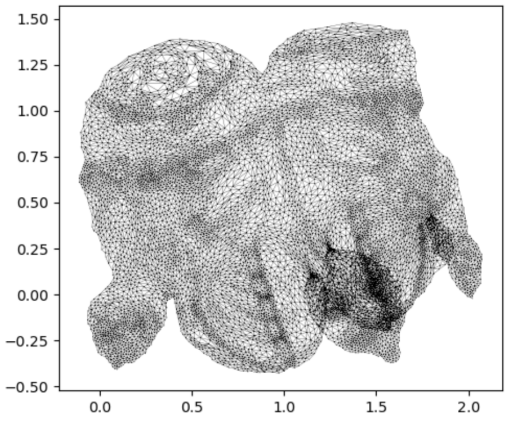


Figure 8: Gargoyle ARAP 结果

4 总结

发现利用 ARAP 相比 ASAP 具有更好的效果。本次实验更加了解了参数化的信息。