

Homework 5 作业报告 by 76-朱雨田

本次作业要求实现 ARAP, ASAP 与 Hybrid 三种非固定边界的曲面参数化算法。我在理解并实现了这些参数化算法的基础上，对 ASAP 做了迭代方法和单一方程组方法的两种实现，并对这些算法做了进一步研究。

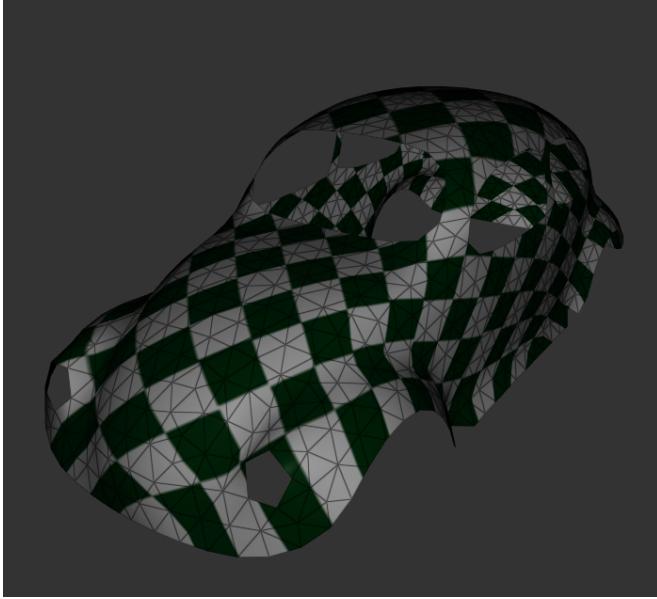
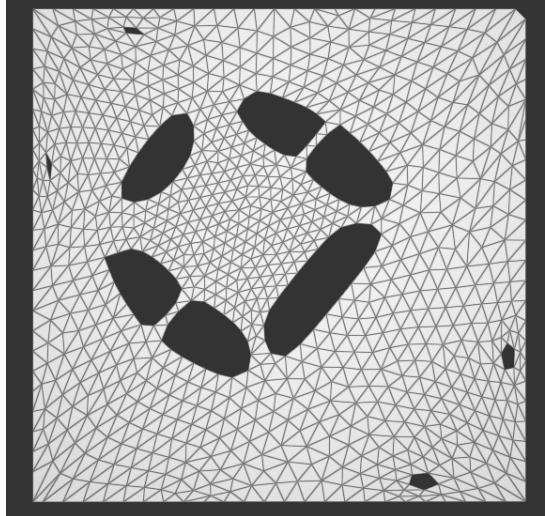
这次作业在数学方面的知识对我来说还有许多我个人不太能独立理解的部分。感谢吴汶政助教的耐心解答，让我能够对论文的各种细节做出能让我满意的理解。在本次作业报告中，我也会试着用刚做完 Homework 4 的大一学生也能理解的语言讲解我对这篇论文的理解。

0. 准备工作

本次作业新增了四个模型。它们的共同特点都是有比较好的切割（没有像 Bunny Head 那样过细的瓶颈了），并且都有各自的难点，如下表所示：

模型	特点
Cow	面数较多；网格较复杂，难以避免重叠；切割后仍保留了较多模型本身的几何特征（头部、眼睛、肢体），易于观察展开算法对形状的处理，确认算法的保形效果。
Beetle	高亏格曲面（即存在多条边界的曲面）
Isis	有较多“细长”的三角形
Gargoyle	网格顶点数、面数非常多

Beetle 模型打破了“模型只有一个边界”的假设，因此我们需要对 Homework4 的边界映射算法做出调整，对每个边界都进行一次遍历，并取长度最长的边界作为映射的边界。相应地，对 Tutte 参数化中用到边界相关的代码进行调整，不再视不为主要边界的点位边界。这样，Tutte 参数化的结果就能够较好地处理 Beetle 模型，并将其作为 ARAP 的 Initial Guess 了。下图是 Floater 参数化处理后的 Beetle 模型。

纹理映射结果	参数化结果 (Floater)
	

对于本次作业的类型设计，因为每种算法都要考虑每个三角面的映射、 L_t 矩阵与能量，因此我单独为每个算法中的**三角面**设计了结构体，负责存储纹理坐标、映射结果、变换矩阵，并更新变换矩阵、计算能量，每次 Local 步骤中只需要遍历三角面并依次执行更新变换矩阵函数即可。我对每个算法本身也设计了类型结构。

但是本次作业的类型设计有较多偷懒的地方，有较多为了编程便利而浪费的内存（如纹理坐标在每个三角面中又存了一遍（其实可以考虑用引用？）；三个顶点的映射结果必有一个是原点，其实不必存储）。我也并没有设计类的继承结构，而是将相似的结构对每个类型都复制了一遍。如果以后有机会重构代码，这些问题是有必要解决的。

1. ARAP

思路解析

本次论文中的几个算法都采用了相似的设计思路：先从参数化的理想结果出发，定义一个参数化能量；再通过解方程或者 Local-Global 迭代方法求这个能量的最小值。ARAP 定义的能量如下：

$$E(u, L) = \frac{1}{2} \sum_{t=1}^T \sum_{i=0}^2 \cot(\theta_t^i) \left\| (u_t^i - u_t^{i+1}) - L_t(x_t^i - x_t^{i+1}) \right\|^2 \quad (1)$$

它衡量的是网格中每个三角面“变形程度”的和。最外部的求和是对所有三角面能量的求和。每个三角面中，

- u 是将曲面上的点参数化映射后的实际结果；
- $L_t(x_t^i - x_t^{i+1})$ 描述了三角面中线段**理想的**变形结果——全等映射，即 **As Rigid as Possible**。
- x 是将三维网格中的单个三角面全等映射到平面后的结果，以便用二维矩阵进行全等变换；
- L_t 则是一个描述全等变换的矩阵。
- θ_t^i 描述的是原三角网格中，与该线段相对的角的余切值。

算法将全等变换作为参数化的理想结果，用对三角形每条线段映射后的实际结果与理想结果的向量作差并取距离平方，并用 \cot 值加权，从而定义了每个三角形与其理想变换结果的差距。非可展曲面不可能将所有三角面全等地参数化映射到一个平面上，因此这一能量不可能取到 0 值。但是，这一能量存在最小值，我们可以将其作为我们的目标。

从以上能量描述中，我们提取出自由的量：每个顶点都有一个参数坐标 u 待求解，每个三角面都有一个全等变换矩阵 L_t 待定。这些量必有一个取值组合能使得总能量最小。求出这个取值组合，也就求出了所有点的参数坐标，完成了曲面的参数化。因此，我们将几何上的参数化问题转换成了一个最优化问题。

数学分析课程中介绍的极值问题解法是对待求极值函数（假设有 n 个自由变量）中的每个自由变量求偏导，令它们全部等于 0，列出 n 个含有 n 个变量的方程；解出该方程组成立的所有自由变量组合，并代入进行进一步验证。

这一方法对于 ASAP 是可行的，但对于 ARAP 存在一些困难。全等变换的矩阵只有一个自由度，即 2D 旋转变换的角度（关于平移的问题会在后面解释）；无论如何用这个自由度表示 L_t ，将得到的能量对这一自由度求导，总是无法让这一自由度在方程组中以线性形式出现（如 $\sin \theta + 2 \cos \theta = 0$ 虽然只有一个自由度，但却并不是一个线性的方程组，对于计算机来说很难找到合适的机械算法进行求解）。

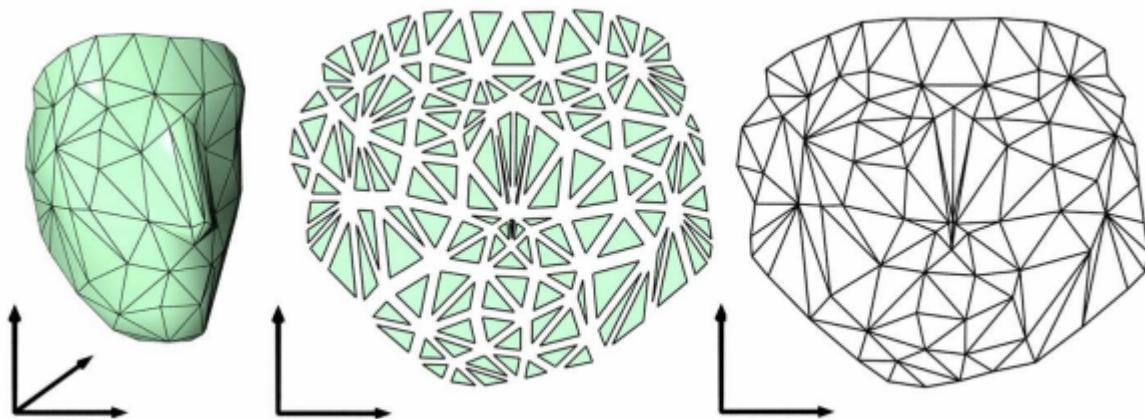
因此，我们不考虑一口气解出所有的自由变量。我们注意到当 u 固定的时候， L_t 存在一个最优值：将 $\triangle x_t^0 x_t^1 x_t^2$ 全等变换到离 $\triangle u_t^0 u_t^1 u_t^2$ “最接近”的三角形时对应的矩阵。论文将这一“最接近”描述成了严谨的数学概念，这在后文中会讲解。论文给出了对每个三角形，在 u 作为已知量的情况下，求解 L_t 的方法，即 Local 步骤。

而当 L_t 固定的时候，能量中 u 的每个分量都是以二次、一次的形式出现的。因此，我们对该能量以 u 的每个分量求偏导，就能得到一个关于 $u_{1x}, u_{1y}, u_{2x}, u_{2y}, \dots, u_{nVx}, u_{nVy}$ 的线性方程组，使用 Eigen 求解即可，即 Global 步骤。

将这两个步骤相互迭代，每次固定一组量，求出能让当前能量最小的另一组量，就能让能量值不断降低。这一能量最终会收敛于全局最小值。这样，经过多次迭代，我们也能在数值上求解能使能量最小的参数组合。

论文给出了一种形象的描述，如下：

- Local 步骤：先将三角网格的每个三角形拆卸下来，参考已有的参数化结果，把每个三角形分散放在平面上；
- Global 步骤：参考分散放置的三角形，对它们原本共有的顶点粘在一起，得到一个参数化结果以给 Local 步骤参考。



具体操作

(1) 映射三角面

如果想要描述一个 2D 到 2D 的全等变换，我们需要描述平移、旋转的量。如果不使用齐次坐标，我们很难用矩阵描述这样的变换。但是，我们观察定义能量的定义式，可以发现，能量只关心三角形的边向量，无论怎样平移三角形都不会改变能量的值。**全等变换对能量的自由度实际上只有旋转一个**。因此，我们不妨在将三角形映射到平面时，采取最简单的规则，即**将其中一个顶点 x_0 映射到原点上，一个顶点 x_1 映射到 x 轴上**，通过三角形的不翻转的全等变换唯一确定顶点 x_2 的位置。这样，我们就能大大简化问题。

先考虑从平面三角形到参数化后三角形的映射 $J \in \mathbb{R}^{2 \times 2}$ ：

$$\begin{cases} J(x_1 - x_0) = u_1 - u_0 \\ J(x_2 - x_1) = u_2 - u_1 \\ J(x_0 - x_2) = u_0 - u_2 \end{cases}$$

这一组方程实际上可以去掉一个，写成如下的矩阵形式：

$$J \begin{pmatrix} x_{1x} - x_{0x} & x_{2x} - x_{0x} \\ x_{1y} - x_{0y} & x_{2y} - x_{0y} \end{pmatrix} = \begin{pmatrix} u_{1x} - u_{0x} & u_{2x} - u_{0x} \\ u_{1y} - u_{0y} & u_{2y} - u_{0y} \end{pmatrix}$$

从而得到

$$J = \begin{pmatrix} u_{1x} - u_{0x} & u_{2x} - u_{0x} \\ u_{1y} - u_{0y} & u_{2y} - u_{0y} \end{pmatrix} \begin{pmatrix} x_{1x} - x_{0x} & x_{2x} - x_{0x} \\ x_{1y} - x_{0y} & x_{2y} - x_{0y} \end{pmatrix}^{-1}$$

这个矩阵在论文中也被称为离散三角网格的 Jacobi 矩阵。ARAP 算法还给出了另一种矩阵 S ，在算法迭代过后能得到与 Jacobi 矩阵相同的结果（在迭代 ASAP 算法中不等价）：

$$S_t(u) = \sum_{i=0}^2 \cot(\theta_t^i) (u_t^i - u_t^{i+1}) (x_t^i - x_t^{i+1})^T$$

这样，我们用数学描述了三角面从三维网格全等映射到平面，再用 Jacobi 矩阵映射到与参数平面中对应三角形全等的目标三角形的过程。

(2) Local 步骤

在 Local 步骤中，我们视已有的参数化结果（在第一次迭代时，将 Tutte 参数化的结果作为 u ）为固定量，寻找能让能量最小的 L_t 组合。论文给出了寻找这种 L_t 的方法，并证明了其正确性。

- 先将 J （或者 S ）作 SVD 分解 $J = U\Sigma V^T$ ；
- 将其中的 Σ 分量替换为单位矩阵；
- 将替换后的结果作为 L_t ，即 $L_t = UV^T$.

SVD 分解中的 U 与 V^T 是行列式为正的正交阵，对应两次旋转变换； Σ 是对角阵，其两个对角元记为 σ_1, σ_2 ，称为奇异值，对应两个方向上的伸缩变换（奇异值可以为负，对应翻转）。将 Σ 分量替换为单位阵，即**取消了变换中的伸缩部分**，只保留旋转部分。Eigen 提供了二阶矩阵奇异值分解的类。

论文在附录中证明了 ARAP 能量等于下式，从而证明了以上方法的正确性：

$$\sum_{t=1}^T A_t \left[(\sigma_{1,t} - 1)^2 + (\sigma_{2,t} - 1)^2 \right]$$

(3) Global 步骤

在 Global 步骤中，我们固定 L_t ，寻找能使总能量最小的 u 的组合。这一组合能通过对所有顶点的 u 的每个分量分别求偏导后解线性方程组得到。论文已经给出了对顶点 i 的 u 求偏导，并使其等于 0，列出的方程：

$$\begin{aligned} & \sum_{j \in N(i)} [\cot(\theta_{ij}) + \cot(\theta_{ji})] (u_i - u_j) \\ &= \sum_{j \in N(i)} [\cot(\theta_{ij}) L_{t(i,j)} + \cot(\theta_{ji}) L_{t(j,i)}] (x_i - x_j), \end{aligned}$$

这是我们擅长求解的线性方程组。但在求解之前，还要注意方程组的一些特点：

- 该方程左侧的系数对于同一个顶点的 x, y 分量都是相等的，从而我们可以把矩阵从 $\mathbb{R}^{2nV \times 2nV}$ 简化成 $\mathbb{R}^{nV \times nV}$ 的形式，并通过对右侧的 x, y 分量分别求解得到 u 的 x, y 分量。
- 观察该方程，它只关心 u 分量之间的差值。**如果对每个 u 都做相等的平移，该方程组仍然成立！**因此，该方程组的增广矩阵并非行满秩。为了让方程组只有唯一解，我们需要**固定**其中一个 u 点，来防止参数化结果的平移。
- 最朴素的固定方法是将该顶点对应的单个方程改成 $u_i = (u'_{ix}, u'_{iy})$ ，右侧为固定住的位置。但是这样的改写破坏了系数矩阵的对称性。我们可以进一步将 u_i 对应列的系数也全部设置为 0，在方程右侧减去用 u_i 代入得到的量，从而列出对称正定的方程组，用 Cholesky 分解求解，优化求解效率。
- 该方程的系数矩阵仅仅与原网格的性质与被固定的量有关，因此我们可以在最开始就列出该矩阵并进行预分解，在迭代过程中用已有的求解器进行 Global 步骤，从而优化效率。

在求解之后，我们还需要注意：

- 参数化结果的任意旋转也不会改变整体的能量。对于部分数据，可能会出现这样的情况：随着 Local / Global 步骤的进行，参数化结果会不断旋转，而得不到一组收敛的 u, L_t 值。为了解决这个问题，我们除了固定一个位置锚点外，在算法开始时还需确定一个方向锚点，在 Global 步骤完成后，对整体以位置锚点为轴旋转，将方向锚点转回最初位置锚点-方向锚点确定的射线上。这个旋转可以通过复数比较漂亮地完成。

综上，我们分别需要确定两个锚点：一个位置锚点和一个方向锚点，分别是为了保证方程有唯一解，和保证参数化坐标最终收敛到一组确定的值。因为这是保持刚性的参数化方法，因此我们不能固定两个位置锚点，否则参数化结果会被“折断”。

小心地实现了以上注意点后，我们就能得到一个稳定高效的 Global Phase 了。

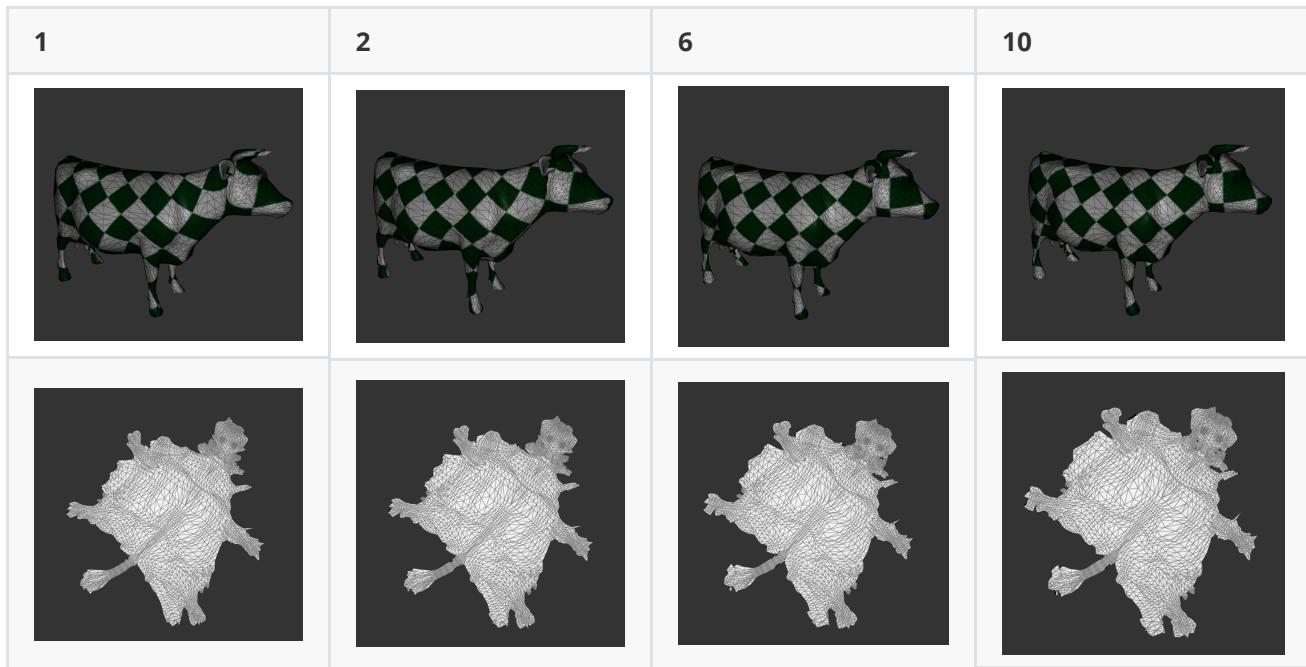
(4) 将步骤连接起来

在参数化算法开始前，我们可以预先确定要迭代的次数；反复进行 Local/Global 步骤后，我们就能得到一个理想的参数化结果。

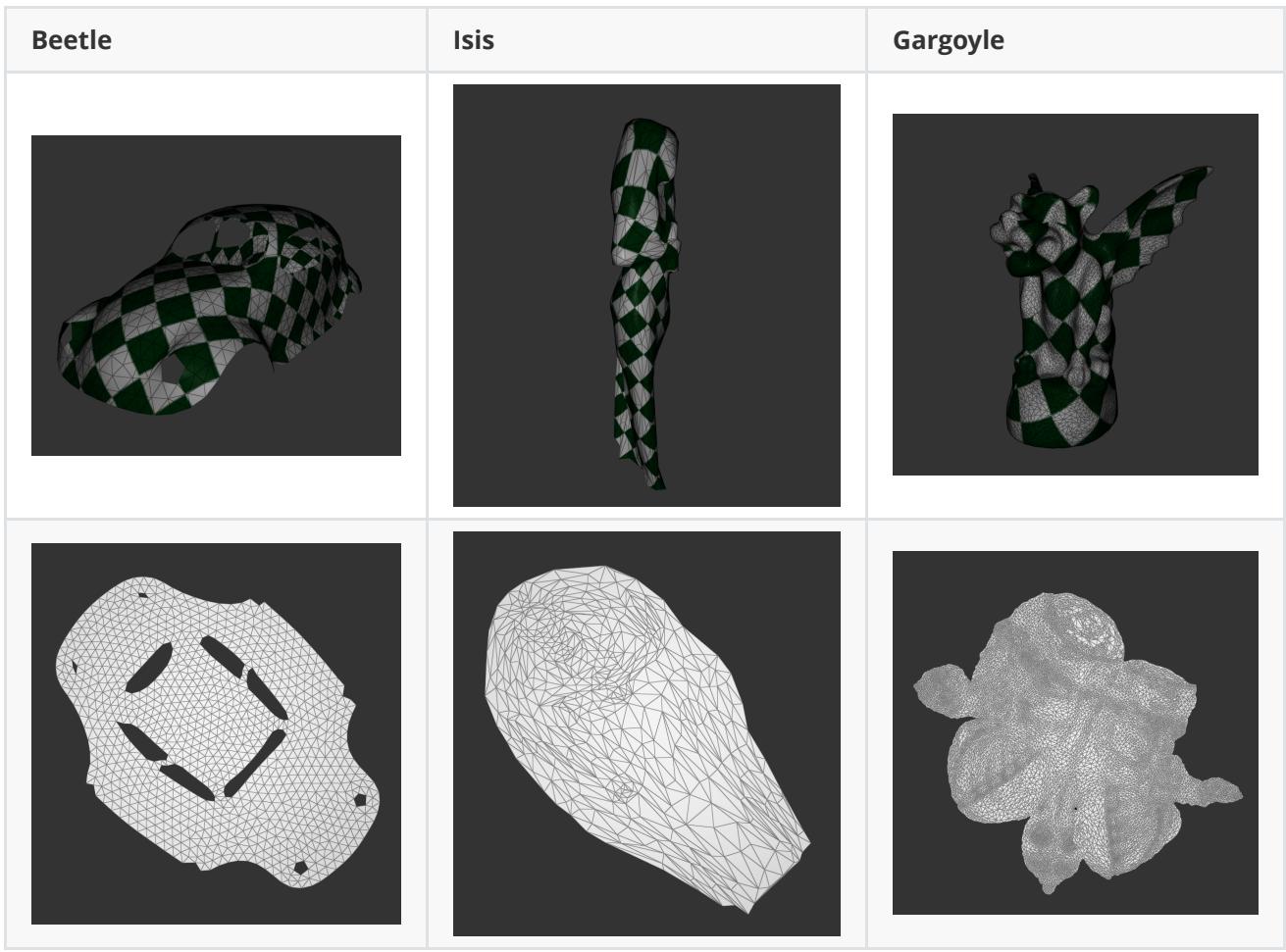
在得到结果后还有事情要干！因为 ARAP 是尽可能全等的变形，因此最后得到的参数平面中的三角形大小应与原三角网格中的三角形相近。但是，为了将纹理贴上参数平面，我们需要将参数平面进一步放缩，将其归一化到 $(0, 1) \times (0, 1)$ 的范围内。这样，我们就完成了 ARAP 算法的全部内容。

效果展示

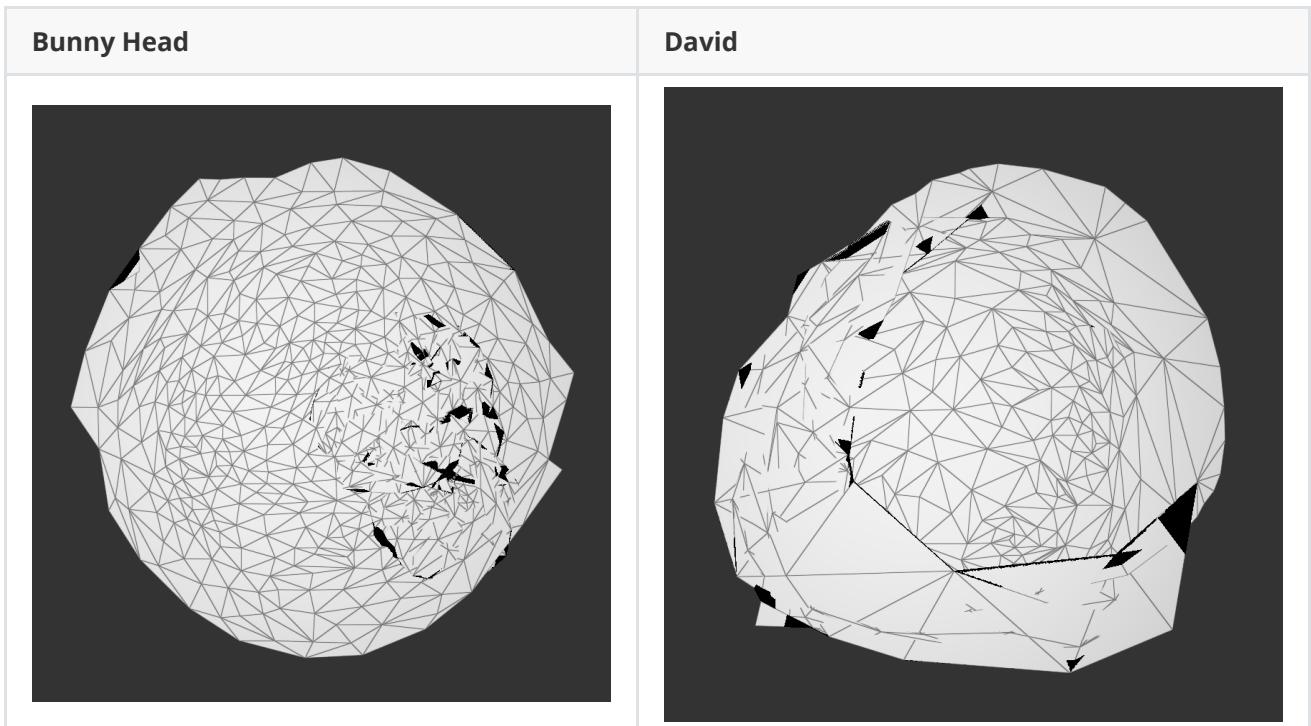
以下展示了 Cow 模型在不同迭代次数下的 ARAP 结果（正方形边界的 Floater 参数化作初值）：



以下展示了其他模型在 ARAP 收敛（10 次迭代）后的参数化结果：



因为 ARAP 是一种尽可能刚性的变换，所以会难以处理有较细瓶颈的、切割的不太好的曲面。以下是 Bunny_Head 与 David 的参数化结果，可以看到明显的三角形反转：



事实上，在 Cow 模型的边界与 Gargoyle 模型的局部也存在三角形翻转的问题。这也说明了这一算法并不完美，需要更优秀的 UV 展开才能得到好的参数曲面。

2. ASAP

ASAP 方法是尽可能相似的参数化算法，其中的相似即是相似三角形中的相似。与 ARAP 的全等变换相比，ASAP 的变换矩阵新增了一个缩放自由度，使得我们能用一个方程组直接求出所有的 u 。

ASAP 定义的能量与 ARAP 相同，只是对 L_t 的要求从全等变换（旋转变换）变成了相似变换。这样，我们可以将 L_t 写成 $\begin{pmatrix} a_t & b_t \\ -b_t & a_t \end{pmatrix}$ 的形式，并对每个顶点的 u_{ix}, u_{iy} 每个三角形的 a_t, b_t 求偏导（这个时候求偏导后的 a, b 也是线性的，不再受 ARAP 那种限制了），得到一个自由度为 $2nV + 2nF$ 的线性方程组。直接对其求解即可。

单一方程组方法

论文并没有直接给出对 a_t, b_t 求偏导的结果，但我们可以从对附录给出的 Hybrid 求导的结果里抄作业。

对于顶点对应的行，我们直接在 ARAP 的方程组基础上，将方程右侧的 $L_{t(i,j)}, L_{t(j,i)}$ 展开为关于 a_t, b_t 的线性式，求出方程右侧关于 a_t, b_t 的系数，并取负填入系数矩阵中 a_t, b_t 对应列即可。注意这里顶点坐标 x, y 分量对应的 a_t, b_t 系数不再相等了，因此不能再将系数矩阵缩小一倍，对分量分别求解了

对于三角面对应的行，我们参考 Hybrid 中对能量的 a_t, b_t 分量求偏导的结果：

$$\begin{cases} C_1 a + 2\lambda a(a^2 + b^2 - 1) = C_2 \\ C_1 b + 2\lambda b(a^2 + b^2 - 1) = C_3 \end{cases}$$

$$C_1 = \sum_{i=0}^2 w_i ((\nabla v_{ix})^2 + (\nabla v_{iy})^2),$$

$$C_2 = \sum_{i=0}^2 w_i (\nabla u_{ix} \nabla v_{ix} + \nabla u_{iy} \nabla v_{iy}),$$

$$C_3 = \sum_{i=0}^2 w_i (\nabla u_{ix} \nabla v_{iy} - \nabla u_{iy} \nabla v_{ix}).$$

$$(\nabla u_{ix}, \nabla u_{iy}) = u_i - u_{i+1}$$

$$(\nabla v_{ix}, \nabla v_{iy}) = v_i - v_{i+1}$$

$$\omega_i = \cot \theta_t^i$$

考虑 ASAP 的能量，方程组可以简化成如下形式，这是关于 $a, b, u_{ix}, u_{iy}, u_{(i+1)x} \dots$ 的线性方程组：

$$\begin{cases} C_1 a - C_2 = 0 \\ C_1 b - C_3 = 0 \end{cases}$$

其中 C_2, C_3 中的 ∇u_i 中含有未知量，将其展开成 u_{ix}, u_{iy} 的形式，整理出其系数，并分别填入系数矩阵中即可。

在将方程组填入代码前，我们同样需要检查自由度的问题。我们首先注意到， $u \equiv \mathbf{0}, a \equiv 0, b \equiv 0$ 时，能量取最低值 0，对应了方程的退化情形。进一步思考，当参数化结果平移、旋转时，能量大小不变；当参数化结果缩小时，能量也会减小。为了防止参数化结果的平移、旋转、缩放、退化，我们可以固定两个位置锚点——因为参数化结果不再刚性，因此可以固定其尺寸。与 ARAP 相同，直接朴素地将两个顶点对应行设置为形如 $u_{ix} = u'_{ix}$ 的等式即可。这个矩阵也可以进一步化为对称正定的形式，但因为我目前的代码结构这样做会很麻烦，所以我还没有这样做。也许有重构机会的话我会试着把它化为对称阵。因为不传入 Initial Guess，所以我分别将三维模型中 x 方向的两个最远点固定在了 $(0, 0)$ 和 $(1, 0)$ （所有点的 x 坐标都相同时特判）。因为最后会归一化，所以固定点位置的设置影响并不大。

但是当我选取其他的固定点组合时，参数化的结果似乎也会发生变化...感觉不太应该发生这样的现象，但已经来不及检查了。

值得一提的是，单一方程组方法不需要传入一个 Initial Guess，也不需要设置迭代次数；该方法可以天然地处理高亏格曲面（Beetle 模型自动地从最大边界展开了），因此可以作为其他方法的 Initial Guess 以加快收敛。

迭代方法

除了单一方程组方法之外，论文还给出了求 ASAP 参数化的另一种思路，即 Local/Global 方法。这种方法只需要对 ARAP 的代码做很小的更改。

论文将 ASAP 的能量最小值也化成了奇异值表示，形式如下：

and for the ASAP case, $k_{1,t} = k_{2,t} = (\sigma_{1,t} + \sigma_{2,t})/2$, so

$$E(u) = \sum_{t=1}^T A_t (\sigma_{1,t} - \sigma_{2,t})^2, \quad (A3)$$

因此，在 Local 步骤中，只需要将 Σ 变为 $\frac{\sigma_{1,t} + \sigma_{2,t}}{2} \cdot I^{2 \times 2}$ 即可使能量收敛于 ASAP 的最小能量。

需要注意的是，在 ASAP 算法中， S 矩阵的作用结果不再等于 J 矩阵的作用结果。我们需要求出原始的 J 矩阵进行奇异值分解。

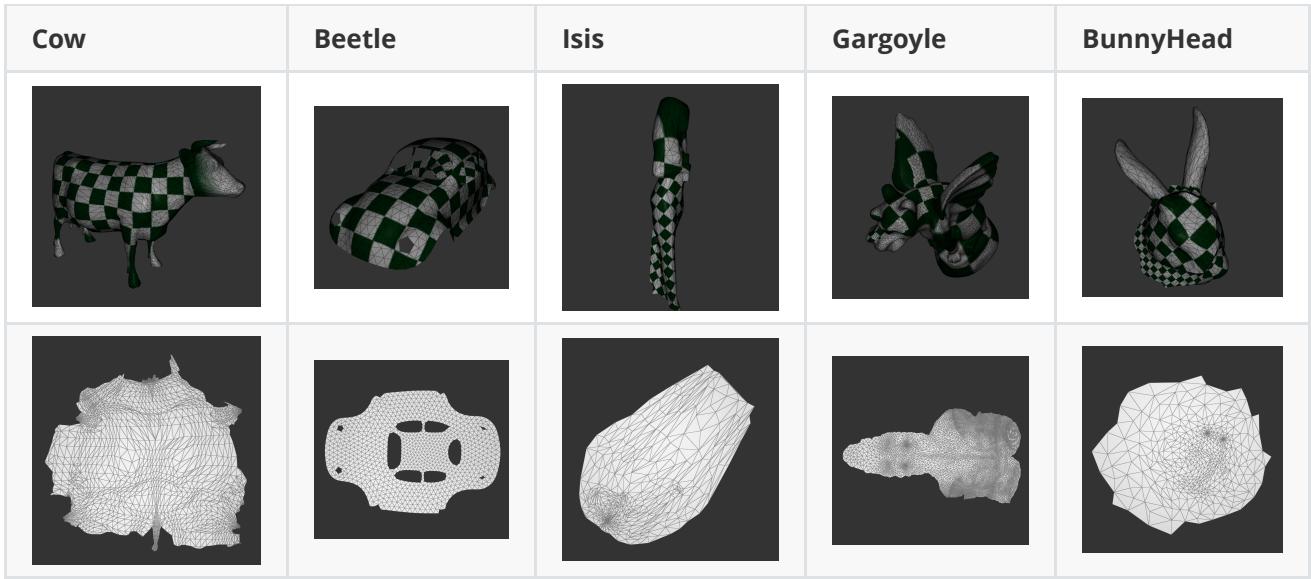
固定锚点的代码不需要修改，方程组求解过程中仍然只需要一个位置锚点。这是因为参考的参数化结果给定了一个三角形的大小，使得三角形不再能在 Global 步骤中自由地缩放。事实上两个位置锚点会造成参数化结果的扭曲。

但是，和 ARAP 中的旋转类似，ASAP 的参数化结果有可能在迭代过程中产生**缩放**，使得最终结果无法收敛（并且这非常容易触发。能量减小的速度异常之快，实际上发生的是参数化结果不仅在趋于相似目标结果，还在不断缩小）。因此，在求解方程组之后，除了将图形转回锚点确定的方向之外，还需要将图形整体缩放，将“方向锚点”拉回其所在的位置。（事实上也算是固定了两个位置锚点，只是其中一个锚点的固定是在求解方程组过后才进行的）。

虽然 ASAP 的迭代方法非常容易实现，但是其相比求解方程组方法，效率实在低下。其原因在于迭代收敛的速度实在太慢——Cow 模型需要迭代 500 次左右才能收敛至与单一方程组求解相近的形状。（而 ARAP 迭代 10 次左右就能得到相当令人满意的结果，尽管还没有完全收敛）

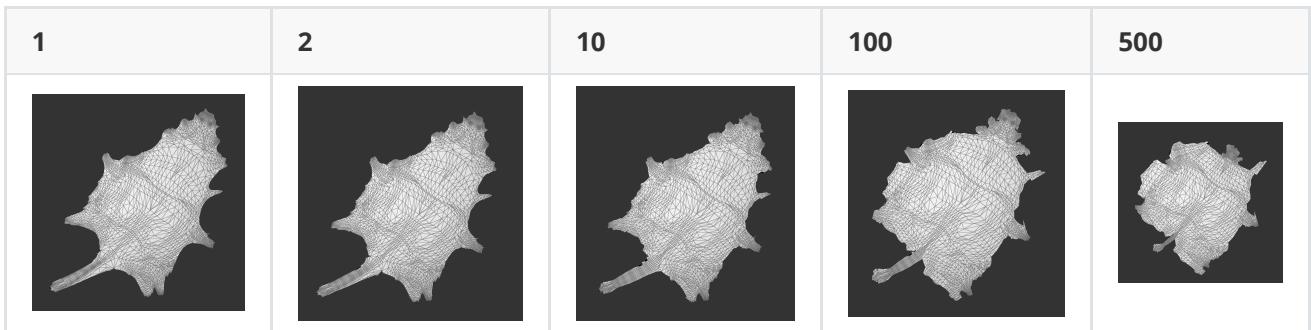
效果展示

以下展示了不同模型在单一方程组求解 ASAP 后的参数化结果：



可以看出相似变换的 ASAP 能正确处理 BunnyHead 模型。David 的参数化结果有明显问题，感觉是我哪里没写对。

以下展示了 Cow 模型在不同迭代次数下的迭代方法 ASAP 结果：



3. Hybrid

思路与操作

论文还发现，可以定义一种能量，通过控制能量中的参数 λ 来混合 ARAP 和 ASAP 两种参数化方法。Hybrid 能量定义如下：

$$E(u, a, b) = \frac{1}{2} \sum_{t=1}^T \left[\sum_{i=0}^2 \cot(\theta_t^i) \|\nabla e_t^i\|^2 + \lambda(a_t^2 + b_t^2 - 1)^2 \right], \quad (5)$$

where

$$\nabla e_t^i = (u_t^i - u_t^{i+1}) - \begin{pmatrix} a_t & b_t \\ -b_t & a_t \end{pmatrix} (x_t^i - x_t^{i+1}).$$

其中 $\lambda = 0$ 时，该能量的最优化与 ASAP 等价； $\lambda = +\infty$ 时，该能量的最优化与 ARAP 等价。

可以这样理解：当 $\lambda = 0$ 时，该式在形式上与 ASAP 等价；当 $\lambda = +\infty$ 时，该能量**强制要求** $a_t^2 + b_t^2 = 1$ ，也就是要求变换矩阵为全等变换；当满足全等变换后，能量在形式上与 ARAP 等价。（我其实理解了很久才看出“强制要求全等变换”的含义）

论文附录中指出，这里对 a_t, b_t 求偏导会得到三次的形式，如下：

$$\begin{cases} C_1 a + 2\lambda a(a^2 + b^2 - 1) = C_2 \\ C_1 b + 2\lambda b(a^2 + b^2 - 1) = C_3 \end{cases}$$

$$C1 = \sum_{i=0}^2 w_i ((\nabla v_{ix})^2 + (\nabla v_{iy})^2),$$

$$C2 = \sum_{i=0}^2 w_i (\nabla u_{ix} \nabla v_{ix} + \nabla u_{iy} \nabla v_{iy}),$$

$$C3 = \sum_{i=0}^2 w_i (\nabla u_{ix} \nabla v_{iy} - \nabla u_{iy} \nabla v_{ix}).$$

$$(\nabla u_{ix}, \nabla u_{iy}) = u_i - u_{i+1}$$

$$(\nabla v_{ix}, \nabla v_{iy}) = v_i - v_{i+1}$$

$$\omega_i = \cot \theta_t^i$$

因此，和 ARAP 一样，我们无法通过解线性方程组求解该问题，从而转向 Local/Global 求解。Hybrid 的 Global 步骤和 ARAP 相同，因为求偏导后多出来的能量项全部消去了。下面研究 Local 步骤，即固定参数化坐标，求解 a_t, b_t 。

关于 a, b 的三次方程组可以继续整理成如下形式：

$$\frac{2\lambda(C_2^2 + C_3^2)}{C_2^2} a^3 + (C_1 - 2\lambda)a - C_2 = 0,$$

$$b = \frac{C_3}{C_2} a$$

- 当 $\lambda = 0$ 时有特解 $a = \frac{C_2}{C_1}, b = \frac{C_3}{C_1}$. 事实上我们在单一方程组求解的 ASAP 算法中就得到过该形式，但在该算法中 C_2, C_3 并不是常数。
- 当 $\lambda = +\infty$ 时有特解 $a = \pm \frac{C_2}{\sqrt{C_2^2 + C_3^2}}, b = \pm \frac{C_3}{\sqrt{C_2^2 + C_3^2}}$.

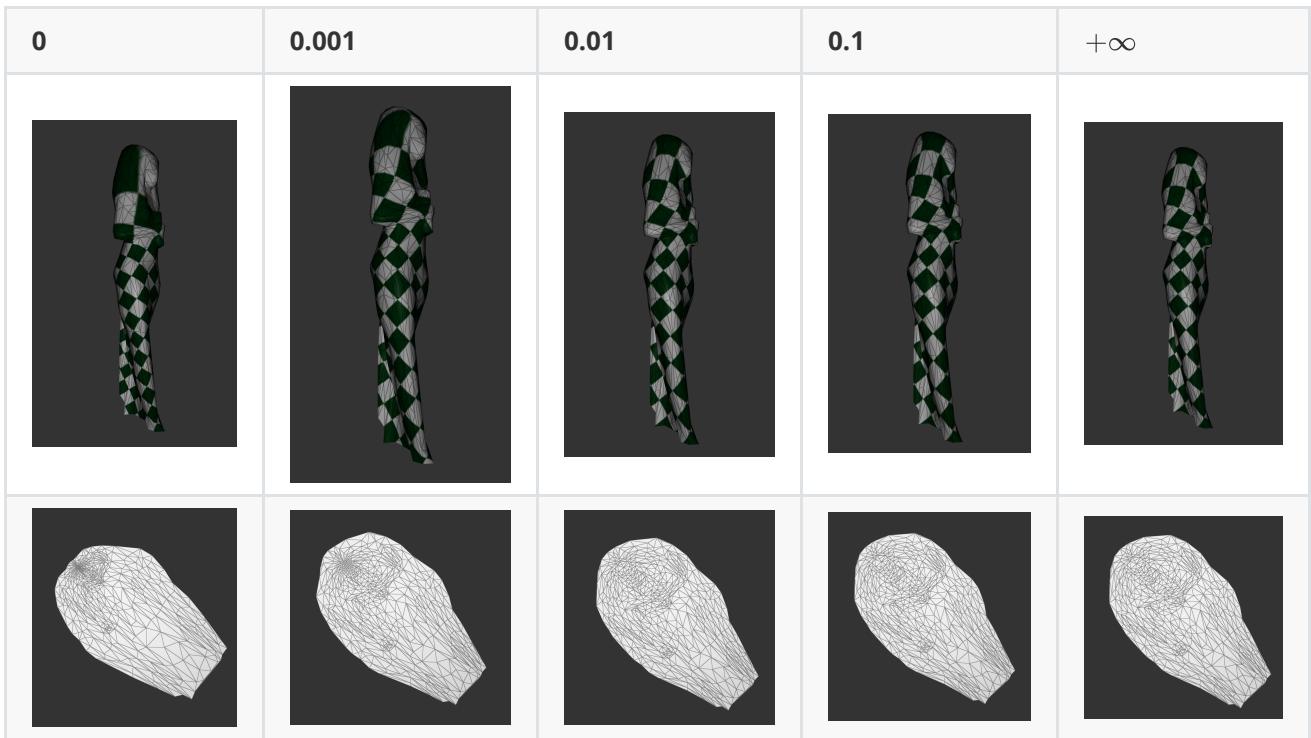
当 λ 不为特殊值时，我们要用牛顿迭代法解该三次方程。初值的选取相当关键，因为如果初值选取不当，选取的结果可能无法得到全局极小值点，导致能量的发散。事实上， $a = \frac{C_2}{\sqrt{C_2^2 + C_3^2}}$ 是一个相当错的初值。

对 ARAP 算法的 Local 部分作出如上更改后，我们就实现了 Hybrid 算法。其实 Hybrid 算法的实现反而是最简单的？

但由于 Hybrid 只能固定一个位置锚点和一个方向锚点，不能在每次 Global 后放缩参数化结果，因此在含有 ASAP 成分的参数化中，能量一定会有不断减小的趋势。同样地，Hybrid 实现的 ASAP 参数化的收敛也非常慢。

效果展示

以下是 Isis 模型分别取 $\lambda = 0, 0.001, 0.01, 0.1, +\infty$ ，迭代 50 次的参数化结果：



心得体会

在一周目的时候，我就确定了这个作业是所有作业里最难的一项。最优化，SVD 分解，Jacobi 矩阵，Local/Global 迭代方法...众多全新的数学概念给我幼小的心灵带来了极大的震撼。当时带着 ChatGPT 啃了好几天这篇论文，始终看不懂怎么去实现，也找不到看懂的方法，在网上搜索这些数学概念后也不觉得自己学会了，只能对“把三角形拆开再拼起来”的流程作感性的理解。最后作业的完成也非常 Cheat：在网上参考了一位学长的代码，把不知道该怎么实现的地方都看了一遍，再糊出一个 ARAP 的效果，不看 optional，直接跑路。当时“做完了”作业我也不觉得我学到了什么。

后来做完弹簧质点模型加速算法后，我觉得我开始理解 Local/Global 迭代方法了。但因为寒假还在忙别的，所以没有回去重看 ARAP 的论文。现在二刷 Homework 5 后，我虽然还有一些数学概念没能完全理解，但我知道我该怎样去找方法理解了——边画图、查资料边和助教确认自己的理解，把没搞懂的东西各个击破，现在我觉得我已经能独立地实现 ARAP，ASAP，Hybrid 参数化算法，对论文的内容有了足够的理解了。寒假时丢掉的成就感在这里找回来了。

这次作业绝对是这学期以来耗时最长的作业，不管是代码还是报告。从周二晚上肝到周六晚上，一些烂摊子（类结构设计的不好，ASAP 矩阵没来得及化成对称，ASAP 的固定点似乎还有问题，David 模型 ASAP 参数化出错等等）实在没精力收拾了（其他课还有好多作业！）。一些很普通的原因导致的 bug 耗了我很长时间，比如三角面初始化的时候所有三角形都翻转了，引起的 bug 花了我至少三个小时去各种地方查找；求 Jacobi 矩阵的时候 $u_1 - u_0$ 直接写成了 u_1 ，又花了我两个小时差错，等等。不过在查错的过程中我又反复翻了几遍论文，对论文看得更仔细了，是好事。

在实现单方程组求解 ASAP 的时候还有过突发奇想，绕过辅助矩阵，直接以奇异值差的平方作为能量求解参数化坐标。在研究这个方法的过程中我手动求出了每个面用 u_i 表示的能量（其中有一个行列式的绝对值，我武断地去掉了绝对值，怀疑是这里出错了），用 Sympy 求偏导，再将求出的偏导式的 Latex 作为注释写进代码中，让 Github Copilot 帮我写构造线性方程组的代码，人工检查正确性（效果意外地不错，只填错了一项）。最后得到的结果是：局部还能看，但有大量很明显的三角形翻转。虽然我现在也不敢肯定这个错误是去掉绝对值导致的，但起码体验了一遍尝试发明算法的感觉。还是很惊叹于最新的科技，2008 年的时候可还没这么多减少试错成本的工具呢。

对我而言最难的作业已经完成了，继续带着 optional 全收集的目标做完整个计算机图形学的作业吧。