论文

> Fast, Interactive Origami Simulation using GPU Computation

2 Simulation Methods

2.1 Meshing

任务：将折痕离散化为三角面片。

面片中有三种折痕：

1. facet crease

   人为添加的折痕，折纸被折起来时仍然保持平整。添加这些折痕的原因是为了将边大于3的面片拆分成三角面片。

2. mountain crease

   折纸被折起来时向外凸出的折痕（论文中以红色表示）

3. valley crease

   折纸被折起来时向内凹的折痕（论文中以蓝色表示）

2.3节中的目标折叠角度的正负利用这里的三种折痕来构建。

在最后得到的模型的mesh中，每个edge都被当成一个销连接桁架中的梁。

而每一根销连接桁架中的梁都被建模为线性弹簧，受到轴向和角度的约束。

2.2 Axial Constraints

> 轴向约束：避免纸张伸长、收缩

通过计算每根梁轴向上的力（局部坐标），再转换为全局坐标上的力，最后施加到node上来实现轴向约束。

根据胡克定律和chain rule:

$$\vec{F}_{axial} = -k_{axial}(l - l_0)\frac{\partial l}{\partial \vec{p}}$$

$$\vec{k}_{axial}：\text{梁的轴向胡克系数}$$

$$\vec{p}：node\text{的位置}$$

$$l：\text{梁当前长度}$$

$$l_0：\text{梁的原长}$$

每根梁的两个node：

$$\frac{\partial l}{\partial \vec{p}_1} = -\hat{I}_{12}, \frac{\partial l}{\partial \vec{p}_2} = \hat{I}_{12}$$

$$\hat{I}_{12}：从node1到node2的unit\ vector$$

$$p_1：node1在全局坐标中的3D位置。$$

对于胡克系数：

$$k_{axial} = \frac{EA}{l_0}$$

$$其中E是杨氏模量，A是梁的横截面，在这里为了方便计算就把EA看做常量$$

于是每根梁的胡克系数只与自己的原长有关。

2.3 Crease Constraints

> 增加对二面角的约束，来完成、约束折叠

> 角度约束：建模为线性弹性扭转弹簧（linear-elastic torsional springs），将相邻三角形面朝鲜某个目标折叠角度驱动

$$\vec{F}_{crease} = -k_{crease}(\theta - \theta_{target})\frac{\partial \theta}{\partial \vec{p}}$$

$$\vec{F}_{crease}：全局坐标中的3D力向量$$

$$\theta_{target}：目标折叠角$$

$$\vec{p}：node位置$$

$$k_{crease}：约束的刚性（stiffness）$$

这里的stiffness（也就是K_crease) 受2.1中提到的折痕的种类的影响：

$$k_{crease} = \begin{cases} l_0 k_{flod} & \text{moutain or valley crease} \\ l_0 k_{facet} & \text{facet crease} \\ 0 & \text{boundary edge or undriven crease} \end{cases}$$

论文中提到这里通过选择stiffness来使得（初步猜想是使得轴向约束远大于角度约束）

$$k_{axial} >> k_{flod}, k_{axial} >> k_{facet}$$

目标折叠角度和折痕种类也是有关的（这个关系比较符合常理）

$$\theta_{target} = \begin{cases} < 0 & \text{mountaion crease} \\ > 0 & \text{valley crease} \\ 0 & \text{facet crease} \end{cases}$$

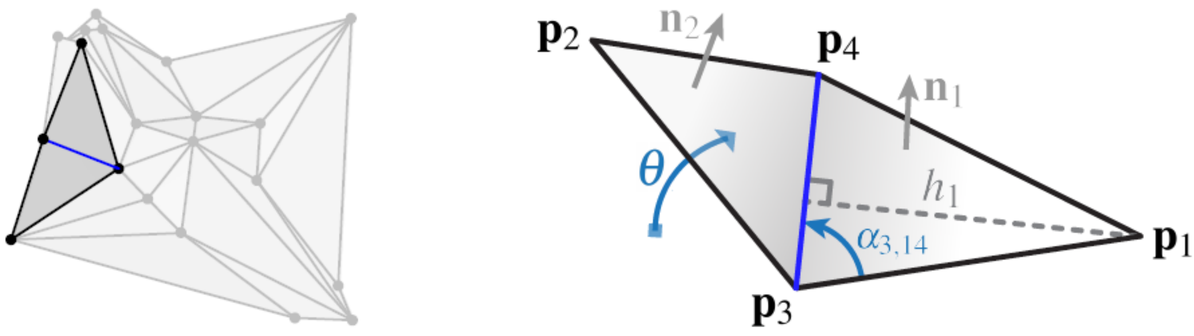


Figure 3: *Formulation of crease constraints.*

每个角度约束会给4个node施加力（角度和nodes如上图）

$$\frac{\partial \theta}{\partial \vec{p}_1} = \frac{\vec{n}_1}{h_1},$$
$$\frac{\partial \theta}{\partial \vec{p}_2} = \frac{\vec{n}_2}{h_2},$$
$$\frac{\partial \theta}{\partial \vec{p}_3} = \frac{-cot\alpha_{4,31}}{cot\alpha_{3,14} + cot\alpha_{4,31}}\frac{\vec{n}_1}{h_1} + \frac{-cot\alpha_{4,23}}{cot\alpha_{3,42} + cot\alpha_{4,23}}\frac{\vec{n}_2}{h_2},$$
$$\frac{\partial \theta}{\partial \vec{p}_4} = \frac{-cot\alpha_{3,14}}{cot\alpha_{3,14} + cot\alpha_{4,31}}\frac{\vec{n}_1}{h_1} + \frac{-cot\alpha_{3,42}}{cot\alpha_{3,42} + cot\alpha_{4,23}}\frac{\vec{n}_2}{h_2}$$
$$\alpha_{1,23}\text{：以}p_1\text{为顶点，}p_2, p_3\text{分别在两边的角}$$
$$h_1, h_2\text{：以}p_1, p_2\text{为顶点的高}$$

2.4 Face Constraints

> 随着三角面片逐渐扁平，轴向约束越小。然而随着三角形形变，面约束带来的力却越来越大。

仅使用梁和折痕的约束（就是轴向约束和角度约束），就可以进行折叠的模拟了。然而这样的模拟会有折叠的表面剪切的问题。论文作者通过实践发现增加面约束可以增加模拟的稳定性。

面约束也之前的约束一样，被建模为线性弹性弹簧。

$$\vec{F}_{face} = -k_{face}(\alpha - \alpha_0)\frac{\partial \alpha}{\partial \vec{p}}$$
$$\alpha\text{：当前角度}$$
$$\alpha_0\text{：初始（还是}flat\text{状态的时候）状态下的角度}$$
$$\vec{k}_{face}\text{：是面约束的刚性（}stiffness\text{）}$$
$$\vec{p}\text{：这里的}p\text{是指邻接的}node\text{的位置}$$

偏导数公式：

$$\frac{\partial \vec{p}_1}{\partial \alpha_{2,31}} = \frac{\vec{n} \times (\vec{p}_1 - \vec{p}_2)}{\|\vec{p}_1 - \vec{p}_2\|^2}$$

$$\frac{\partial \vec{p}_2}{\partial \alpha_{2,31}} = -\frac{\vec{n} \times (\vec{p}_1 - \vec{p}_2)}{\|\vec{p}_1 - \vec{p}_2\|^2} + \frac{\vec{n} \times (\vec{p}_3 - \vec{p}_2)}{\|\vec{p}_3 - \vec{p}_2\|^2}$$

$$\frac{\partial \vec{p}_3}{\partial \alpha_{2,31}} = -\frac{\vec{n} \times (\vec{p}_3 - \vec{p}_2)}{\|\vec{p}_3 - \vec{p}_2\|^2}$$

2.5 Numerical Integration

> 这部分显式的计算了在轴向约束和角度约束下node的小位移，也是本篇论文的创新点。

一个node受到的总的力:

$$\vec{F}_{total} = \sum_{beams} \vec{F}_{beam} + \sum_{creases} \vec{F}_{crease} + \sum_{faces} \vec{F}_{face}$$

于是得到该node的加速度（三维）：

$$\vec{a} = \frac{\vec{F}_{total}}{m}$$

在论文的推导过程中，将m假定为1，在真正模拟时，选用更加准确的质量值应该可以获得更加真实的力学模拟效果。

计算node的速度和位置时使用了显式欧拉法（就是模拟单摆时用的第一种方法）

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}\Delta t,$$
$$\vec{P}_{t+\Delta t} = \vec{P}_t + \vec{v}_{t+\Delta t}\Delta t$$

对于步长的选择，为了使得模拟过程能够保持稳定，需要满足

$$\Delta t < \frac{1}{2\pi\omega_{max}},$$
$$\omega_{max}\text{：模型中所有约束中的最大固有频率}$$
$$\text{由于}k_{axial} \gg k_{flod}, k_{axial} \gg k_{face}\text{，所以}$$
$$\omega = \sqrt{\frac{k_{axial}}{m_{min}}},$$
$$m_{min}\text{：一个}beam\text{中质量较小的}node\text{的重量}$$
$$\text{同时给定初始值:}$$
$$\vec{v}_0 = \vec{0}$$

为了能够在有限的迭代中得到稳定的状态，在邻接的vertices中加入了粘性阻尼:

$$\vec{F}_{damping} = c(\vec{v}_{neighbor} - \vec{v}),$$
$$c\text{：粘性阻尼系数}$$

使用这种方法来计算祝你可以减少对浮点数的计算。

论文作者发现即使使用了严格的阻尼，在全局状态下仍然会有欠阻尼现象。

使用更加精确的阻尼和使用更精确的质量一样，都可以得到更加真实的力学模拟效果。

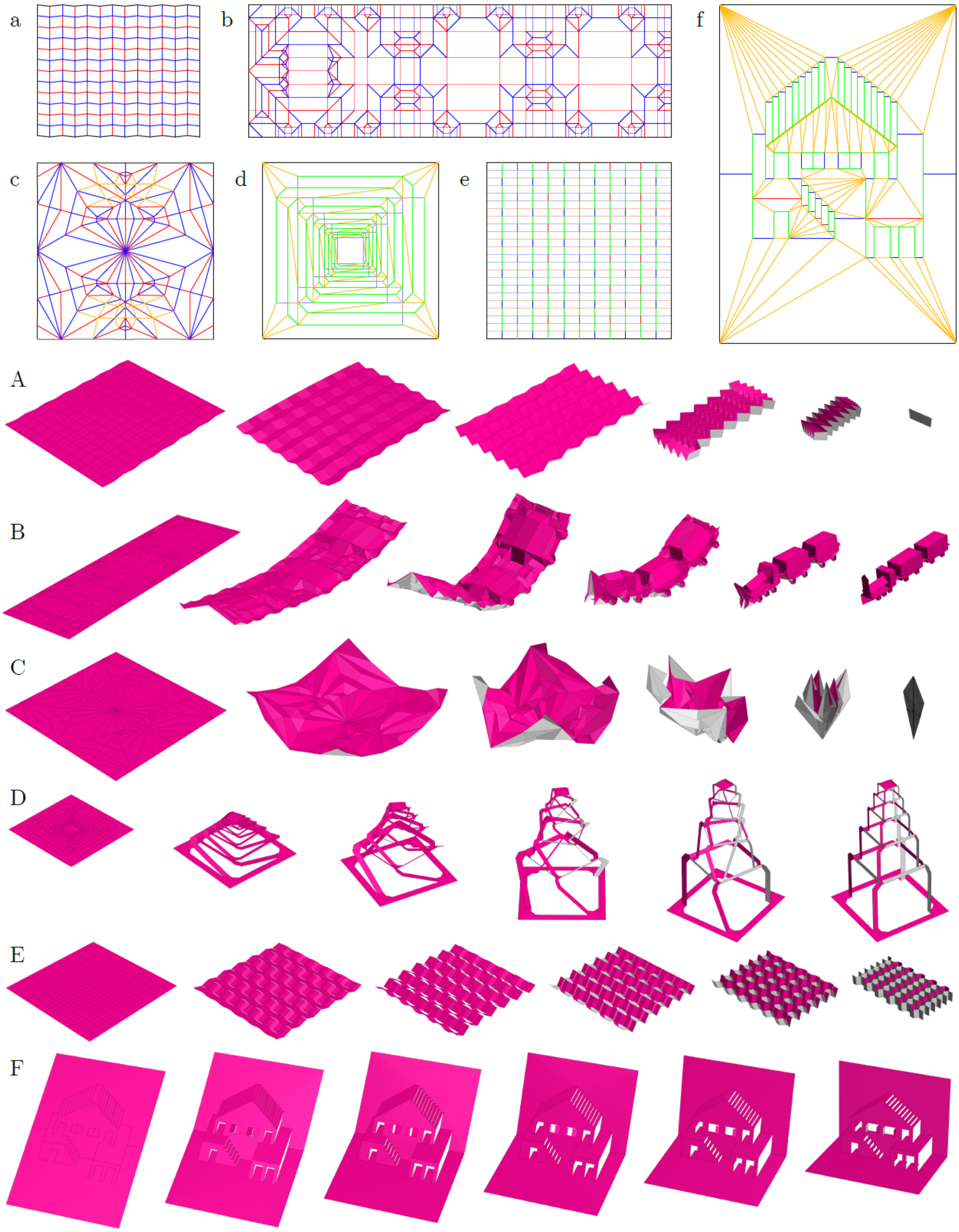
$$c = 2\zeta\sqrt{k_{axialm}}$$

$$\zeta：\text{阻尼系数，较为合适的值为：}0.01 \le \zeta \le 0.5$$

$$m：\text{质量，在推导中选用}1\text{作为}m\text{的值}$$

3 Implementation

> 论文作者实现了一个开源的web应用，也就是 [Simulator]



demo中，上文提到的常量选取的数值为：

$$1.EA = 20$$
$$2.k_{flod} = 0.7$$
$$3.k_{facet} = 0.7$$
$$4.k_{face} = 0.2$$

在求解器开始工作之前，需要预计算：

1. stiffness和damping之间的关系
2. 构建一个elements之间几何关系的查找表。

在GPU（WebGL）中，模拟的每一次更新需要进行的计算：

1. 并行计算mesh中所有三角面片的表面法矢
2. 并行计算mesh中所有边的当前折叠角度
3. 并行计算mesh中所有边在2.3节中最后4个公式中的系数
4. 并行计算mesh中所有node的力和速度
5. 并行计算mesh中所有node的位置

作者提到，如果使用NVIDIA CUDA之类的GPU框架的话，可以使用更少的步骤完成上述操作。

在给的在线模拟器中，多次更新（specified number of simulation steps）之后才会更新一次几何形状。

3.1 Strain Visualization

将纸张的张力可视化出来，帮助用户理解纸张的几何结构。

3.2 User Interaction

这里提到了VR交互以及后续希望添加的功能。

两部分都是在Web应用中加入的feature，和原理无关所以此处略去。