GPU Gradient Decent Shape Optimizer Design

December 16, 2021

1 Requirements

Write a GPU accelerated shape optimizer to practice GPU optimizations and investigate speed up for Morpho first version should be able to:

- Optimize surface Area with constant Volume

- Read in morpho meshes (for ease of use)

- Run on the GPU

- Input is Morpho mesh print area and volume. output is optimized morpho mesh print area and volume

2 Conceptual Method

A mesh is a list of vertices $\vec{r_i}$ with rules for how the connect. One dimensional connections are lines, Two dimensional connections are facets. A connection is a list of vertex indices that are part of the higher dimensional element. Let i label verties, k label facets and l label the vertex in a facet. Consider a facet f_k with three vertices labeled $\vec{r}_{f_{k_l}}$ I.E $\vec{r}_{f_{k_1}}, \vec{r}_{f_{k_2}} \vec{r}_{f_{k_3}}$. The area of this facet is:

$$A_k = (\vec{r}_{f_{k_2}} - \vec{r}_{f_{k_1}}) \times (\vec{r}_{f_{k_3}} - \vec{r}_{f_{k_2}})$$

To minimize area we need to find the gradient of the area with respect to the vertices of the facet.

$$\vec{S}_0 = (\vec{r}_{f_{k_2}} - \vec{r}_{f_{k_1}}) \tag{1}$$

$$\vec{S}_1 = (\vec{r}_{f_{k_3}} - \vec{r}_{f_{k_2}}) \tag{2}$$

$$\vec{S}_{01} = \vec{S}_0 \times \vec{S}_1 \tag{3}$$

$$\vec{S}_{010} = \vec{S}_{01} \times \vec{S}_0 \tag{4}$$

$$\vec{S}_{011} = \vec{S}_{01} \times \vec{S}_1 \tag{5}$$

$$\nabla_{\vec{r}_{f_{k_1}}} A_{f_k} = \frac{\vec{S}_{011}}{2|\vec{S}_{01}|} \tag{6}$$

$$\nabla_{\vec{r}_{f_{k_2}}} A_{f_k} = -\frac{\vec{S}_{011} + \vec{S}_{010}}{2|\vec{S}_{01}|} \tag{7}$$

$$\nabla_{\vec{r}_{f_{k_3}}} A_{f_k} = \frac{\vec{S}_{010}}{2|\vec{S}_{01}|} \tag{8}$$

Now that we have the area per vertex per facet we need to collate them to find the gradient. To do this on the GPU avoiding write collisions we use a map from $F(i) : i \to (k,l)_{m_i}$ where m_i counts up to the number of facets vertex i is included in. The gradient of the are with respect to a single vertex i is then:

$$\nabla_{\vec{r}_i} A = \sum_{(k,l)_{m_i}} \nabla_{\vec{r}_{f_{k_l}}} A_{f_k}$$

Likewise for volume integrand on a facet we have

$$V_{f_k} = \left| \frac{(\vec{r}_{f_{k_1}} \times \vec{r}_{f_{k_2}}) \cdot \vec{r}_{f_{k_3}}}{6} \right|$$

The gradient of the volume integrad for each vertex on a facet is then

$$s = sign((\vec{r}_{f_{k_1}} \times \vec{r}_{f_{k_2}}) \cdot \vec{r}_{f_{k_3}}) \tag{9}$$

$$\nabla_{\vec{r}_{f_{k_1}}} V_{f_k} = \frac{s}{6} \vec{r}_{f_{k_2}} \times \vec{r}_{f_{k_3}} \tag{10}$$

$$\nabla_{\vec{r}_{f_{k_2}}} V_{f_k} = \frac{s}{6} \vec{r}_{f_{k_3}} \times \vec{r}_{f_{k_1}} \tag{11}$$

$$\nabla_{\vec{r}_{f_{k_3}}} V_{f_k} = \frac{s}{6} \vec{r}_{f_{k_1}} \times \vec{r}_{f_{k_2}} \tag{12}$$

Then we perform the sum in the same manor

$$\nabla_{\vec{r}_i} V = \sum_{(k,l)_{m_i}} \nabla_{\vec{r}_{f_{k_l}}} V_{f_k}$$

To minimize area with constant volume we project the area gradient vector to the space where the volume change is zero. We call the negative of this

projection the force.

$$\vec{F}_i = -\left(\nabla_{\vec{r}_i} A - \frac{\nabla_{\vec{r}} A \cdot \nabla_{\vec{r}} V}{\nabla_{\vec{r}} V \cdot \nabla_{\vec{r}} V} \nabla_{\vec{r}_i} V\right)$$

Noting here that as this is a global constraint the dot products are computed across every vertex.

This only preserves volume to linear order however so we must reproject the verteices onto the global constraint. This is done is done via gradient decent. We first calculate how far off we are from the target volume $\delta V = V_{target} - V_{actual}$. We then recalculate the gradient of the volume for the now moved vertices using the above method. The amount to move the vertices to project onto the constraint of the volume is then

$$\frac{\delta V}{\nabla V \cdot \nabla V} \nabla V$$

Gradient decent then consists of calculating this force and moving the vertices on the mesh until the force is below a threshold value.

3 Implementation Method

First pass of this will be done in C++ using CUDA for GPU acceleration Class stucture is below

```
            GradientDecentOptimizer
const char* myFileName
Mesh myMesh
DeviceMesh GPUMesh
Gradient myGradient
GradientDecentOptimizer(const char * inputMesh)
optimize(bool useGPU)
gradDecentStep(Mesh)
gradDecentStep(DeviceMesh)
print()
```

```
                   Mesh
unsigned int _numVert
double* _vert
unsigned int _numFacet
unsigned int * _facet
Mesh(const char *fileName)
print(const char *fileName)
updateFromGradient(Gradeint
```

```
                   Gradient
double *_gradA
double *_gradV
double *_gradAProjected

bool _onGPU;
-- Mesh methods are CPU and for time comparison
Gradient(Mesh) // uses CPU memory, sets _onGPU = fal
Gradient(DeviceMesh) // uses GPU memory sets _onGPU

calculateGradA(Mesh)
calculateGradA(DeviceMesh) // calls calcA kernal
calculateGradV(Mesh)
calculateGradV(DeviceMesh) // calls calcV kernal
projectGradA(Mesh)
projectGradA(DeviceMesh)    // calls projectA Kernal
```

```
                   DeviceMesh
unsigned int _numVert
double* _vert //(on the GPU)
unsigned int _numFacet
unsigned int * _facet //(on the GPU)
vector <unsigned int, unsigned int> *_vertToFacet
Gradient _grad
DeviceMesh(const char *fileName)
DeviceMesh(Mesh) // copies a mesh to the GPU

Mesh copyToHost() // copies a mesh from the GPU

updateFromGradient() //calls moveMesh kernal
```

Figure 1: For this project there will be a optimizer class that manages a gradient and a mesh to perform the optimization. It will load in a mesh from file, create a device mesh from this mesh and tell the gradients to calculate and the mesh to move.