Group Member: Yasheng Sun, Qichang Sun

Problem Restatement

Render model with different color based on its curvature.

Methodology

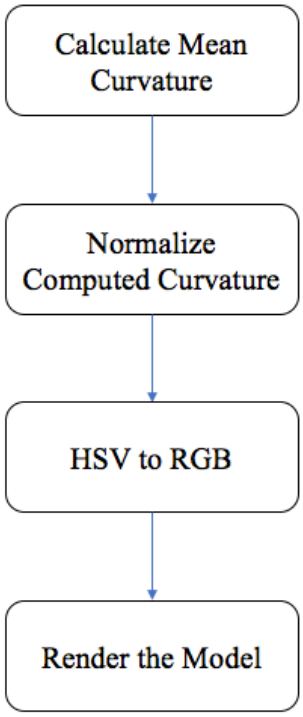

Fig. 1. Workflow of model rendering

The whole workflow of model rendering according to the related curvature is described in Fig. 1. Firstly, the mean curvature, one of the most common curvature, is calculated. And then the calculated curvature need to be normalized for following computation as below.

$$\begin{cases} 1.0 & H > 2\sigma \\ 0.5 + \dfrac{H - average}{2\sigma} & \\ 0.0 & H < 2\sigma \end{cases}$$

We denote mean curvature by H. Average stands for the mean of distribution and σ is for mean square error. The extreme value whose absolute distance to mean is greater

than 2 σ is ruled out in order to render the model. After we applying the normalization, we scale curvature to between zero and one. However, this normalized value is still unable to be directly used for rendering the model. In order to make the model more vivid, we represent the H component in HSV color space proportional to curvature. As is shown in fig.2, every separate component in HSV space is interpretable. Specifically, H represents hue in color space which describe the color information vividly. After obtaining the H component, we convert HSV color space to RGB color space for rendering. Result of our strategy is shown as fig.4. Details of the achievement is shown in appendix.

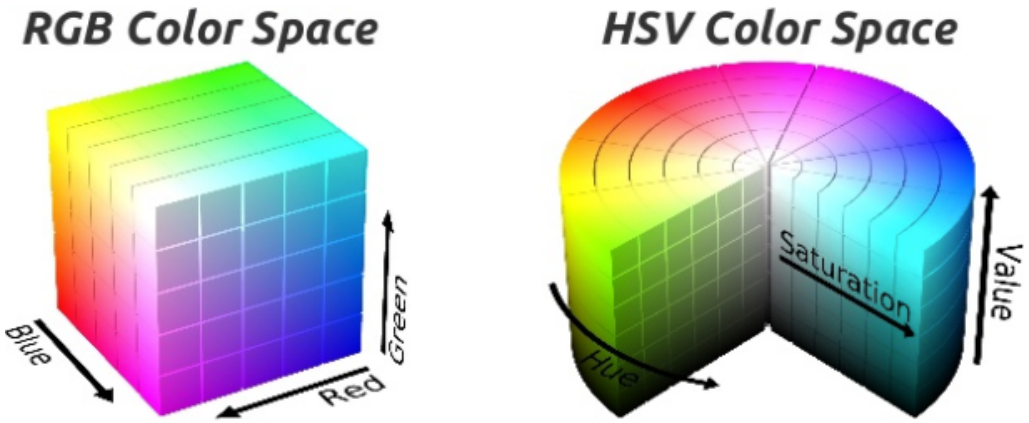

Fig. 2. HSV and RGB color space

$$C = (1 - |2L - 1|) \times S$$

$$X = C \times (1 - |(H / 60°) \bmod 2 - 1|)$$

$$m = L - C/2$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \le H < 60° \\ (X, C, 0) & , 60° \le H < 120° \\ (0, C, X) & , 120° \le H < 180° \\ (0, X, C) & , 180° \le H < 240° \\ (X, 0, C) & , 240° \le H < 300° \\ (C, 0, X) & , 300° \le H < 360° \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

Fig. 3. Conversion from HSV to RGB

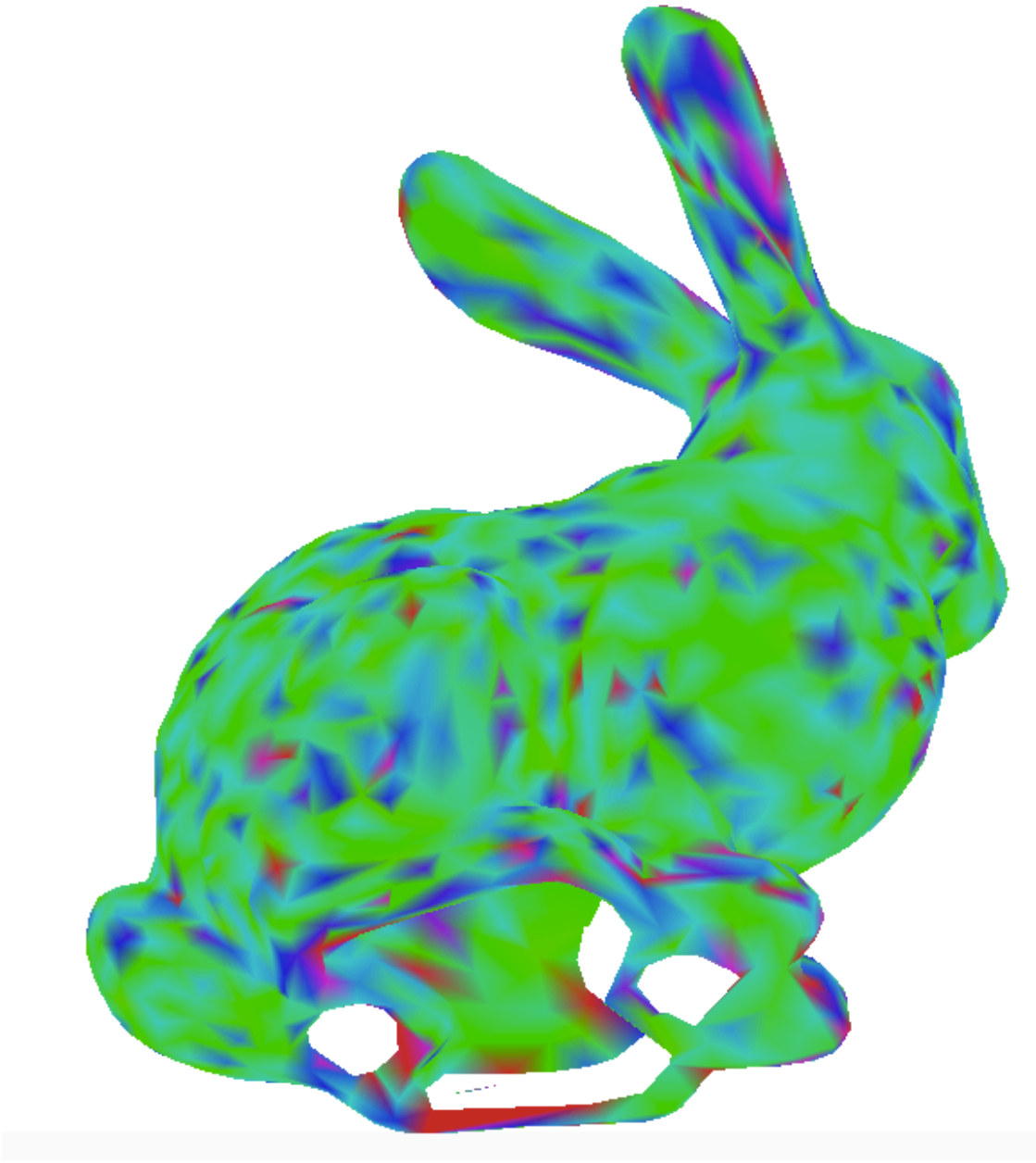

Fig. 4. Result of our strategy

Appendix

Code for curvature calculation

```
double aver = 0.0,sigma2 = 0.0;
for(int i =0;i<vList.size();i++)if(vList[i]!=NULL){
    Vertex* vi = vList[i];
    HEdge* he = vi->HalfEdge(); HEdge* curr = he;
    Vector3d tot(0,0,0);
    double area = 0.0;
    while(1){
        if(!curr->IsBoundary()){
            Vertex* vj = vi->HalfEdge()->End();
            Vertex* vk = vi->HalfEdge()->Next()->End();
            area += Area(vi->Position(),vk->Position(),vj->Position());
        }
        if(!curr->IsBoundary()&&!curr->Twin()->IsBoundary()){
            Vertex* vj = vi->HalfEdge()->End();
            double cot_alpha,cot_beta;
            Vertex* vk1 = vi->HalfEdge()->Next()->End();
            Vertex* vk2 = vi->HalfEdge()->Twin()->Next()->End();
            cot_alpha =Cot(vi->Position(),vk1->Position(),vj->Position());
            cot_beta = Cot(vi->Position(),vk2->Position(),vj->Position());
            tot += (cot_beta + cot_alpha)*(vj->Position()-vi->Position());
        }
        curr = curr->Twin()->Next();
        if(curr == he)break;
    }
    vi->H = -0.5*tot.L2Norm()/area;
}
```

Code for curvature normalization

```
aver /= vList.size();
for(int i=0;i<vList.size();i++)if(vList[i]!=NULL)sigma2 += (vList[i]->H-aver)*(vList[i]->H-aver);
sigma2 /= vList.size();
double sigma = sqrt(sigma2);
for(int i=0;i<vList.size();i++)if(vList[i]!=NULL){
    if(vList[i]->H-aver>=2*sigma)vList[i]->H = 1;
    else if(vList[i]->H-aver<=-2*sigma)vList[i]->H =0;
    else{vList[i]->H = 0.5+(vList[i]->H-aver)/4/sigma;}
    unsigned char r,g,b;
    HSVtoRGB(r, g, b, int(359-vList[i]->H*359), 80, 80);
    Vector3d h_color(int(r)/255.0,int(g)/255.0,int(b)/255.0);
    vList[i]->SetColor(h_color);
}
```