ETER: Elastic Tessellation for Real-Time Pixel-Accurate Rendering of Large-Scale NURBS Models, Supplementary Material

RUICHENG XIONG*, University of Science and Technology of China, China
YANG LU* and CONG CHEN, Sheyun Technology, China
JIAMING ZHU, YAJUN ZENG, and LIGANG LIU†, University of Science and Technology of China, China

1 A DISCUSSION ON SLEFES METHOD

Yeo et al. [2012] uses the tensor product form of slefes, i.e., piecewise bilinear patches, to determine the tessellation factors of surfaces. However, the tensor product form of slefes only measures the error between surfaces and piecewise bilinear patches rather than triangle meshes. An example is that Yeo's method cannot determine the tessellation factors of bilinear patches themselves, as the approximation error is always 0.

In addition, there is also an issue deriving bound improvement. Let $f(t) = \sum_{j=0}^{n} C_i B_n^i(t)$ be a polynomial function. Let $\nabla_j^2 f = C_{j-1} - 2C_j + C_{j+1}$, and the piecewise upper and lower bounds of f are:

$$\overline{f}(t) = l(t) + \sum_{j=1}^{n-1} \max(0, \nabla_j^2 f)\overline{a}_j^n(t) + \sum_{j=1}^{n-1} \min(0, \nabla_j^2 f)\underline{a}_j^n(t),$$

$$\underline{f}(t) = l(t) + \sum_{j=1}^{n-1} \min(0, \nabla_j^2 f)\overline{a}_j^n(t) + \sum_{j=1}^{n-1} \max(0, \nabla_j^2 f)\underline{a}_j^n(t),$$

such that $\underline{f}(t) \le f(t) \le \overline{f}(t),\ \forall\, t \in [0,1]$, where:

$$a_j^n(t) = \frac{1}{\frac{j-1}{j} + \frac{n-j-1}{n-j} - 2}\left(\sum_{k=0}^{j} \frac{k}{j} B_k^n(t) + \sum_{k=j+1}^{n} \frac{n-k}{n-j} B_k^n(t)\right).$$

and $\overline{a}_j^n(t), \underline{a}_j^n(t)$ are the linear upper and lower bounds of $a_j^n(t)$ ([Lutterkort 2000; Peters 2003]). Let $w = \overline{f} - \underline{f} = \sum_{j=1}^{n-1} |\nabla_j^2 f|(\overline{a}_j^n - \underline{a}_j^n)$, then $\|w\| = \max_{t\in[0,1]} w(t) = \max_{j=1,\cdots,n-1} w(t_j)$. We re-represent f on the interval $[t, t+h]$ as $\tilde{f}$, and the corresponding error function is $\tilde{w}$. Then we have:

$$\max_{j=1,\cdots,n-1} |\nabla_j^2 \tilde{f}| \le h^2 \max_{j=1,\cdots,n-1} |\nabla_j^2 f|.$$

[Yeo et al. 2012] claimed that this implies $\|\tilde{w}\| \le h^2\|w\|$. However, this conclusion is not generally true, because the maximum coefficient in w is scaled by h^2 does not necessarily mean that every coefficient is scaled by h^2. A simple example is a cubic polynomial with control points $[0.1967, 0.1983, 0.1704, 0.3678]$, and its piecewise linear approximation with 3 segments has error 0.0162, and according to [Yeo et al. 2012] its reparameterization on interval $[0.5, 1]$ should have error less than 0.00405, but actually, it is 0.0044.

*Joint first authors
†The corresponding author

Authors' addresses: Ruicheng Xiong, University of Science and Technology of China, China, xiongruicheng@mail.ustc.edu.cn; Yang Lu, luyang@lyteflow.cn; Cong Chen, Sheyun Technology, China, chencong@lyteflow.cn; Jiaming Zhu, sanlayn@mail.ustc.edu.cn; Yajun Zeng, yajun_zeng@mail.ustc.edu.cn; Ligang Liu, University of Science and Technology of China, China, lgliu@ustc.edu.cn.

2 PSEUDO CODE OF COMBINED CRACK FILLING

ALGORITHM 1: Crack Filling

```
Input: The visibility buffer of pixels whose centers are inside
       triangles (NonConsBuf), and the visibility buffer of pixels
       whose centers are outside triangles (ConsBuf).
Output: The ID of the triangle to be rendered.
PixPos = GetPixelPosition(ThreadID);
NonConsVis = NonConsBuf[PixPos];
ConsVis = ConsBuf[PixPos];
if (NonConsVis.Z == 1) and (ConsVis.Z == 1) then
    return background;
end
if (NonConsVis.Z == 1) then
    return ConsVis.TriID;
end
if (ConsVis.Z == 1) or (NonConsVis.Z < ConsVis.Z) then
    return NonConsVis.TriID;
end
Nurbs_n = GetNurbsID(NonConsVis.TriID);
Nurbs_c = GetNurbsID(ConsVis.TriID);
if (Nurbs_n==Nurbs_c) then
    return NonConsVis.TriID;
end
NonConsRepeat = 1;
for each pixel position PixPos_i around PixPos do
    NonConsVis_i = NonConsBuf(PixPos_i);
    if (GetNURBSID(ConsVis_i.TriID)==Nurbs_n) then
        NonConsRepeat++;
    end
end
if NonConsRepeat > 3 then
    return NonConsVis.TriID;
else
    return ConsVis.TriID;
end
```

3 CRACK FILLING OPTIMIZATION

Through some simple derivations below, we can know that under pixel accuracy, the distance between the triangular meshes of adjacent patches at the boundary is less than 0.5 pixels. Assume a function $f(x) \in C^2[0,1]$, and its piecewise linear interpolations $l_1(x), l_2(x)$, with different knot vectors $0 = t_0 < t_1 < \cdots < t_K = 1$ and $0 = s_0 < s_1 < \cdots < s_L = 1$, and we have:

$$f(t_i) = l_1(t_i),\ i = 0, 1, \cdots, K,$$

$$f(s_j) = l_2(s_j),\ j = 0, 1, \cdots, L.$$

Assume $\|f - l_1\| < \epsilon$, $\|f - l_2\| < \epsilon$, where $\|g\| = \max_{t \in [0,1]} |g(t)|$. Then for any interval $[t_{i-1}, t_i]$, we have:

$$|l_1(t_{i-1}) - l_2(t_{i-1})| = |f(t_{i-1}) - l_2(t_{i-1})| < \epsilon,$$
$$|l_1(t_i) - l_2(t_i)| = |f(t_i) - l_2(t_i)| < \epsilon.$$

- If $s_j \notin (t_{i-1}, t_i), \forall\ j = 0, \cdots, L$, then l_1 and l_2 are linear functions on $[t_{i-1}, t_i]$, which means:

$$\begin{aligned} &\|l_1 - l_2\|_{[t_{i-1}, t_i]} \\ =&\max(|l_1(t_{i-1}) - l_2(t_{i-1})|, |l_1(t_i) - l_2(t_i)|) \\ <&\epsilon \end{aligned}$$

$$\|l_1 - l_2\|_{[t_{i-1}, t_i]} = \max(|l_1(t_{i-1}) - l_2(t_{i-1})|, |l_1(t_i) - l_2(t_i)|) < \epsilon$$

.

- Assume $t_{i-1} = s_{j_0} < s_{j_0+1} < \cdots < s_{j_0+d} = t_i$, then l_1 and l_2 are linear functions on each interval $[s_{j_0+r}, s_{j_0+r+1}]$, $r = 0, \cdots, d-1$, therefore $\|l_1 - l_2\|_{[s_{j_{k-1}}, s_{j_k}]} < \epsilon$.

Thus, $\|l_1 - l_2\| < \epsilon$.

From the above conclusion, we know that in the pixels where crack occurs there must be at least two triangle edges from different patches. In order to minimize the impact of the conservative buffer on crack filling, we only need to utilize conservative rasterization on the left and bottom boundaries of the patch. This can be done by only writing fragments whose barycentric coordinates are outside the range $[0, 1]$ to the conservative buffer in fragment shaders.

REFERENCES

David C Lutterkort. 2000. *Envelopes of nonlinear geometry.* Ph. D. Dissertation. Purdue University.

Jörg Peters. 2003. Mid-structures linking curved and piecewise linear geometry. In *SIAM Conference on Geometric Design and Computing (GD03), Seattle, Washington, US.* 10–13.

Young In Yeo, Lihan Bin, and Jörg Peters. 2012. Efficient Pixel-Accurate Rendering of Curved Surfaces. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* Association for Computing Machinery, New York, NY, USA, 165–174.