Efficient Pixel-Accurate Rendering of Curved Surfaces

Young In Yeo
University of Florida

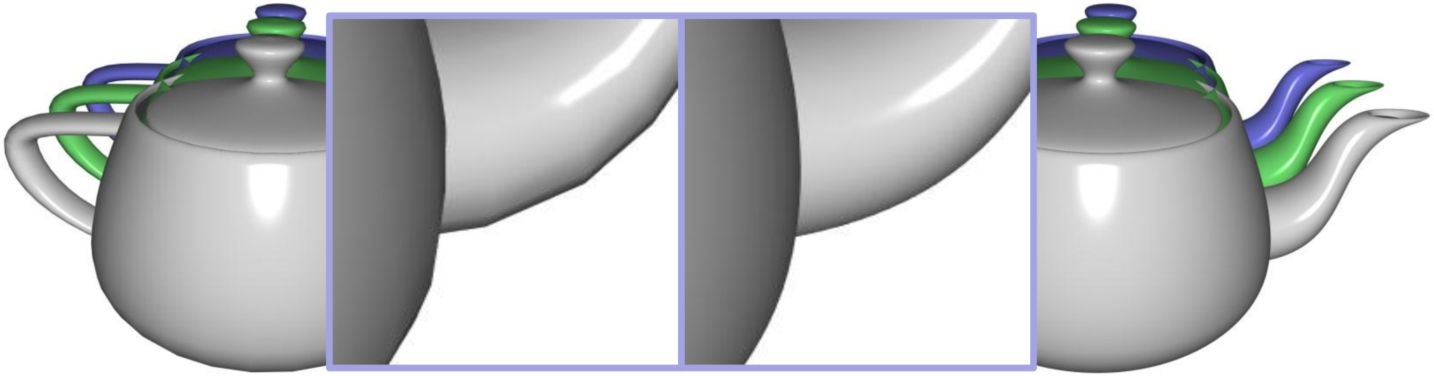
Lihan Bin
University of Florida

Jörg Peters
University of Florida

Optimal Tessellation?

Tessellation too low

Tessellation too high



> 2000 fps

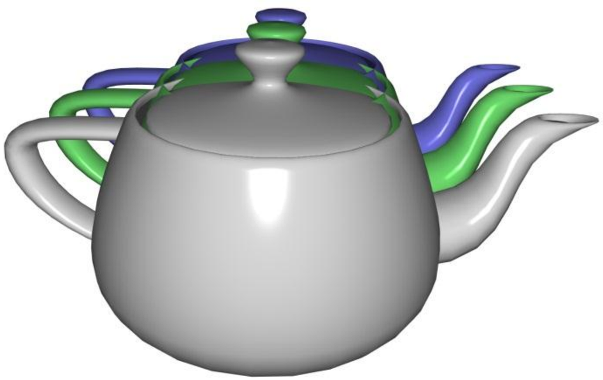
< 50 fps

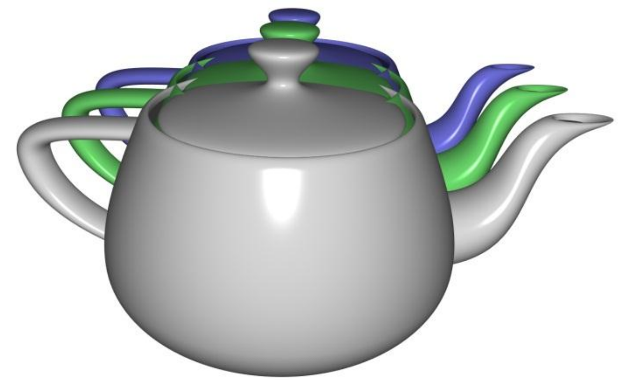
Outline

- **What is Efficiency?**
  - Triangle size
  - Non recursive
- What is Accuracy?
- Test & Predict (SLEFE)
- Implementation & Demo

Low Tessellation

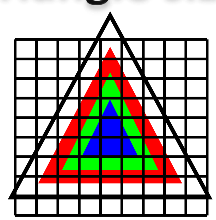
High Tessellation

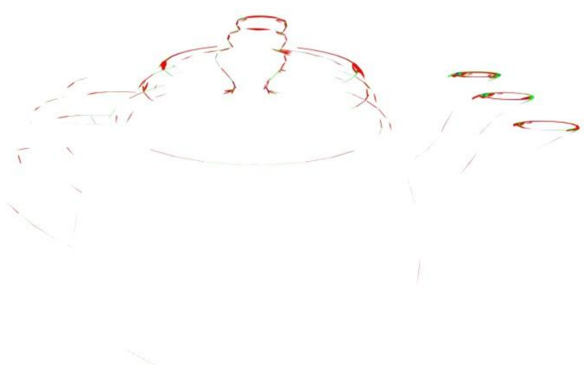


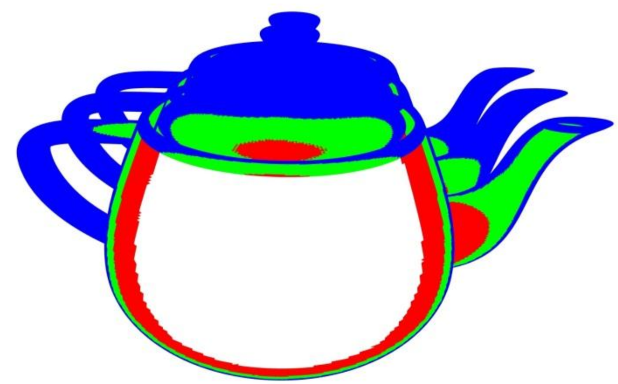


Efficiency

Triangle size





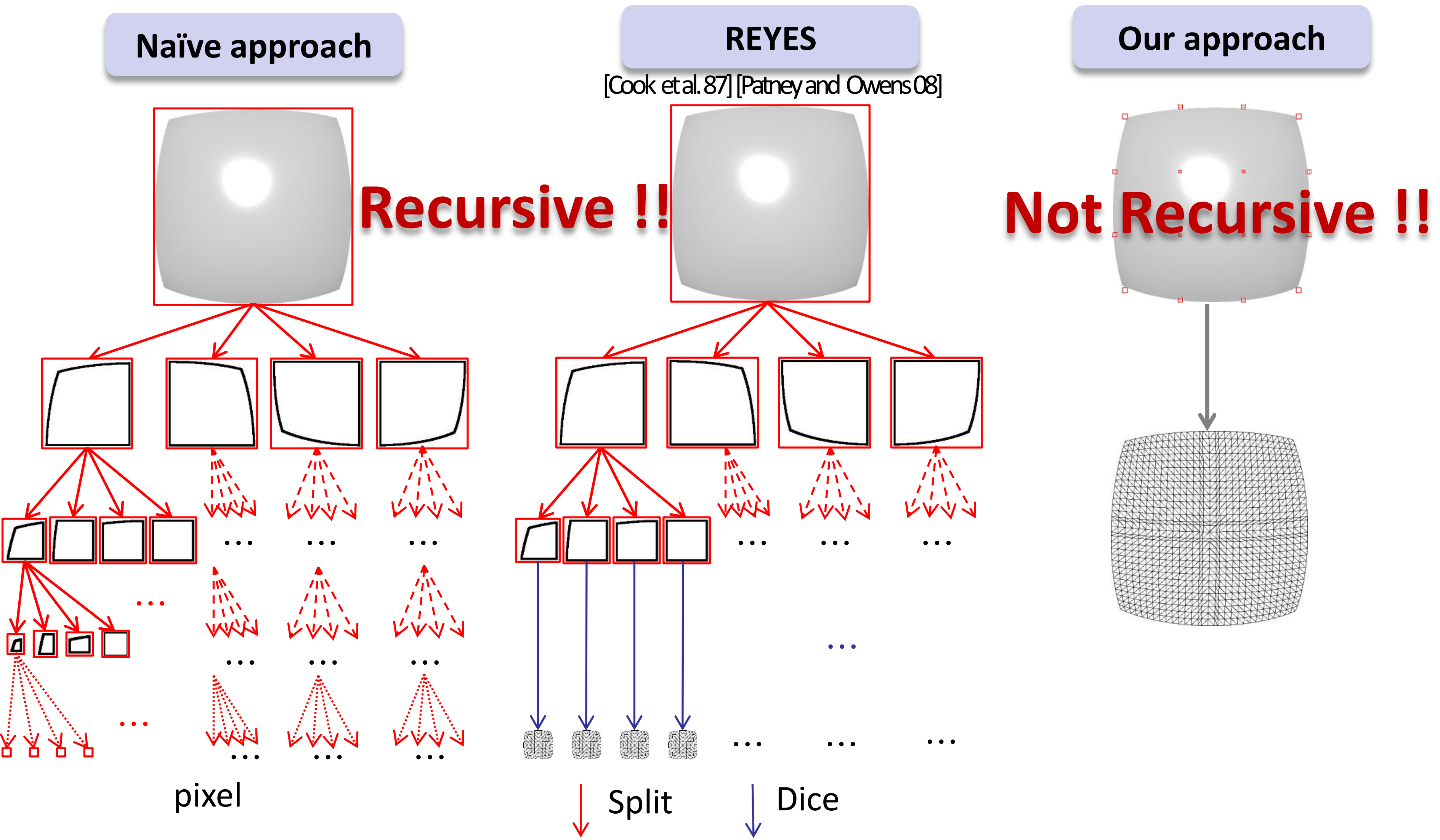


- size < 5 pixels
- size < 10 pixels
- size < 20 pixels
- size >= 20 pixels

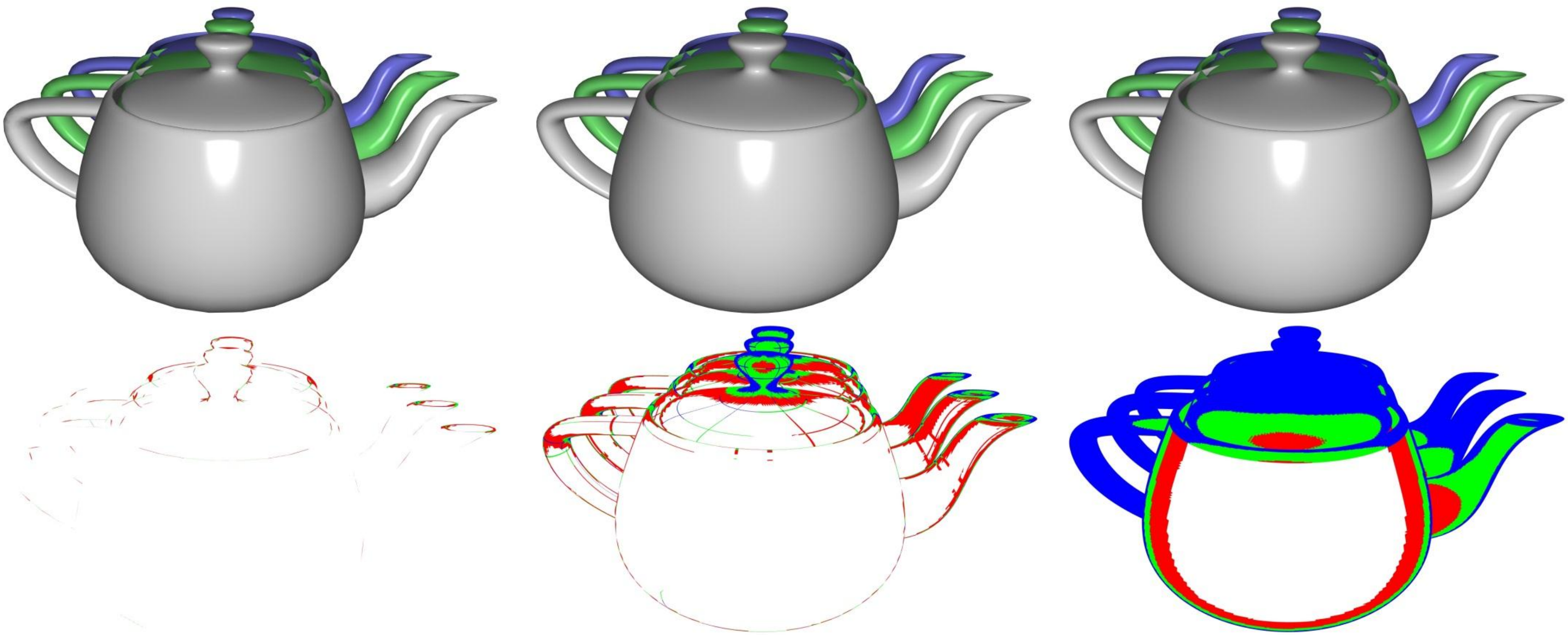
> 2000 fps

< 50 fps

Micropolygonization



Low Tessellation | Our Approach | High Tessellation



> 2000 fps | ~ 1500 fps | < 50 fps

▲ size < 5 pixels
▲ size < 10 pixels
▲ size < 20 pixels
△ size >= 20 pixels

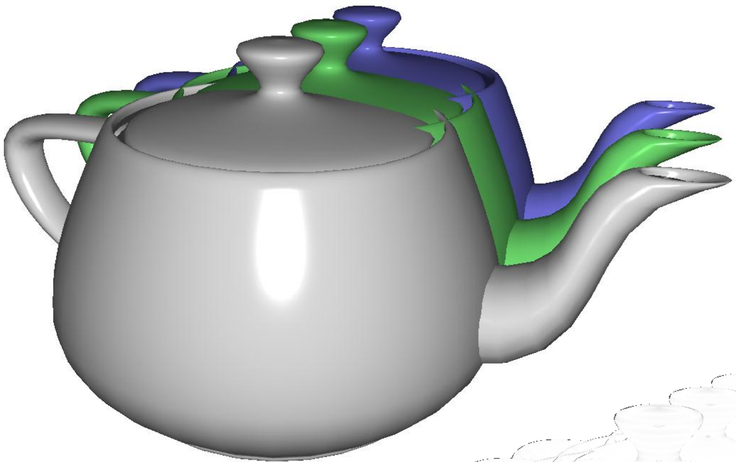
Outline

- What is Efficiency?
- **What is Accuracy?**
  - Covering accuracy (depth)
  - Parametric accuracy (distortion)
- Test & Predict (SLEFE)
- Implementation & Demo

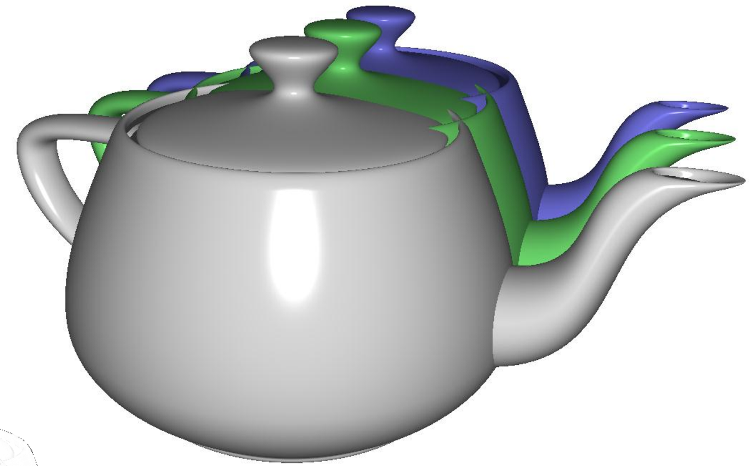
Covering accuracy (depth)

- Correct surface piece determines the pixel

Low Tessellation

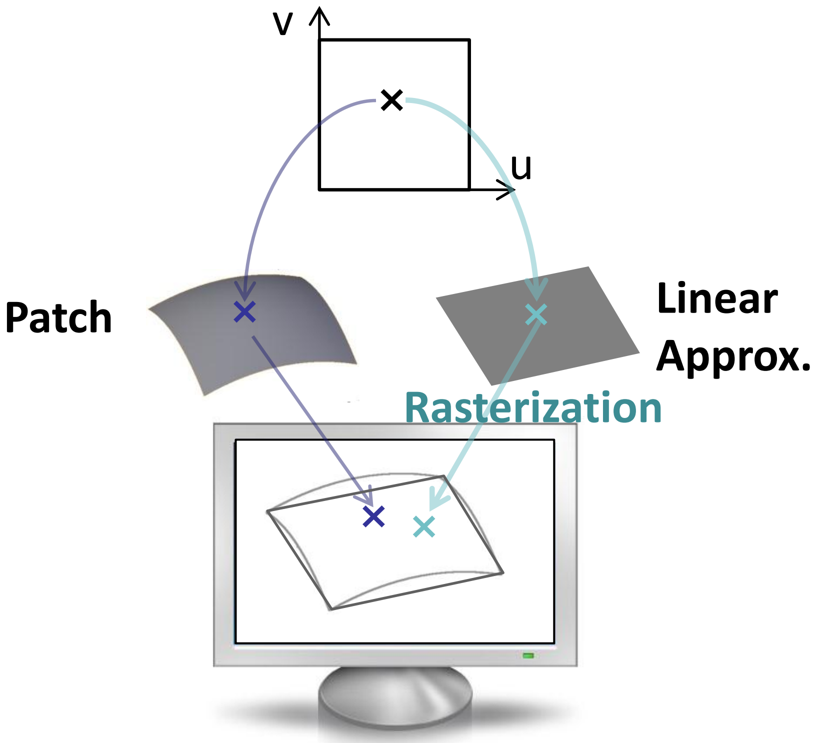


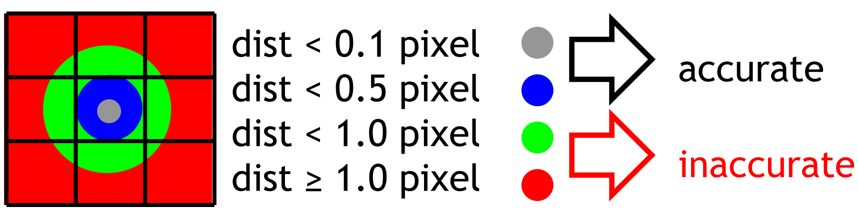
High Tessellation



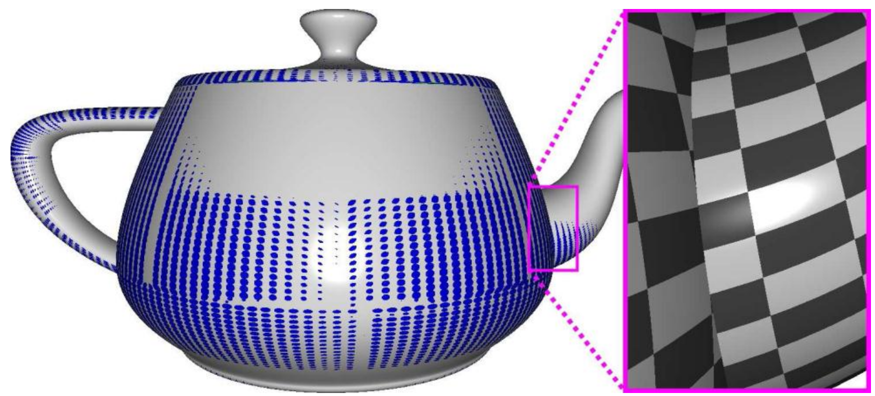
Inaccurate covering pixels

Parametric accuracy
(no distortion & wrong normals)

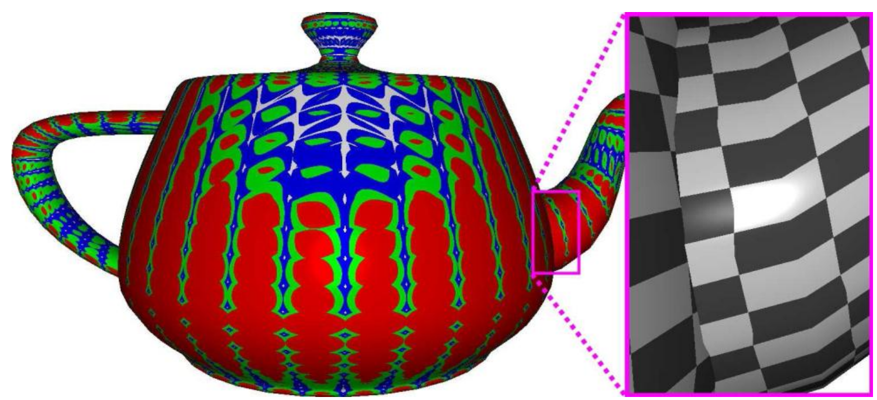




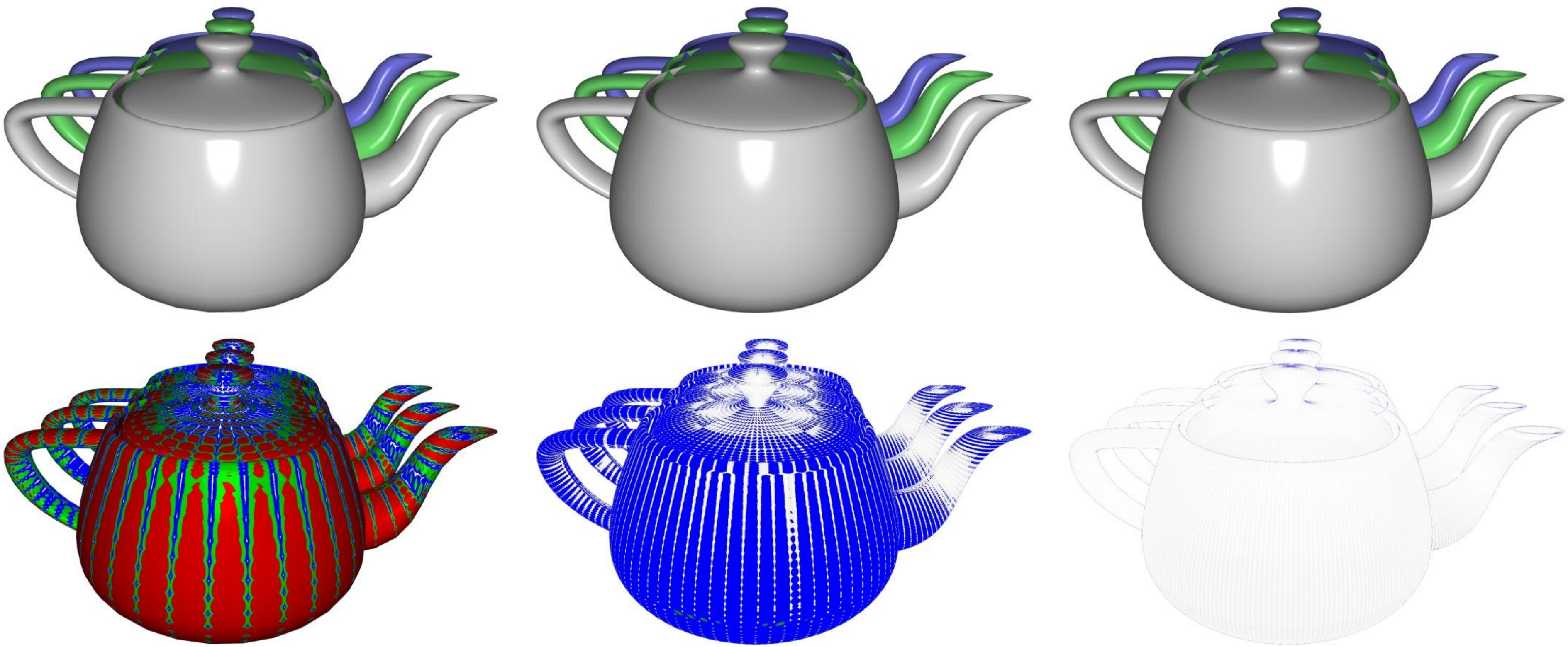
Our Approach



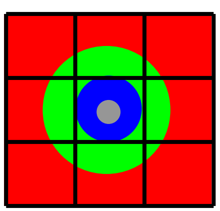
Low Tessellation



Low Tessellation | Our Approach | High Tessellation



> 2000 fps | ~ 1500 fps | < 50 fps



dist < 0.1 pixel
dist < 0.5 pixel
dist < 1.0 pixel
dist ≥ 1.0 pixel

accurate

inaccurate

Outline

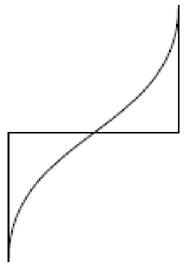
- What is Efficiency?
- What is Accuracy?
- **Test & Predict (SLEFE)**
- Implementation & Demo

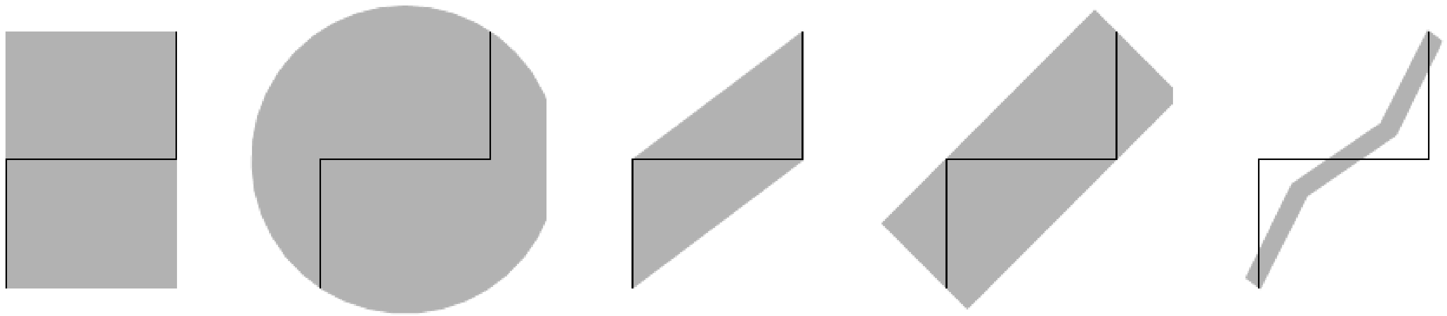
Key insight

- Triangulations are rendered accurately.
- Need to only control the **variance** of (the projection of) the triangulated surface to the exact surface.

What is a SLEFE?

Curve and Control Polygon





AABB Bounding Disk Convex Hull OBB m=3-piece slefe

SLEFE: Subdividable Linear Efficient Function Enclosure

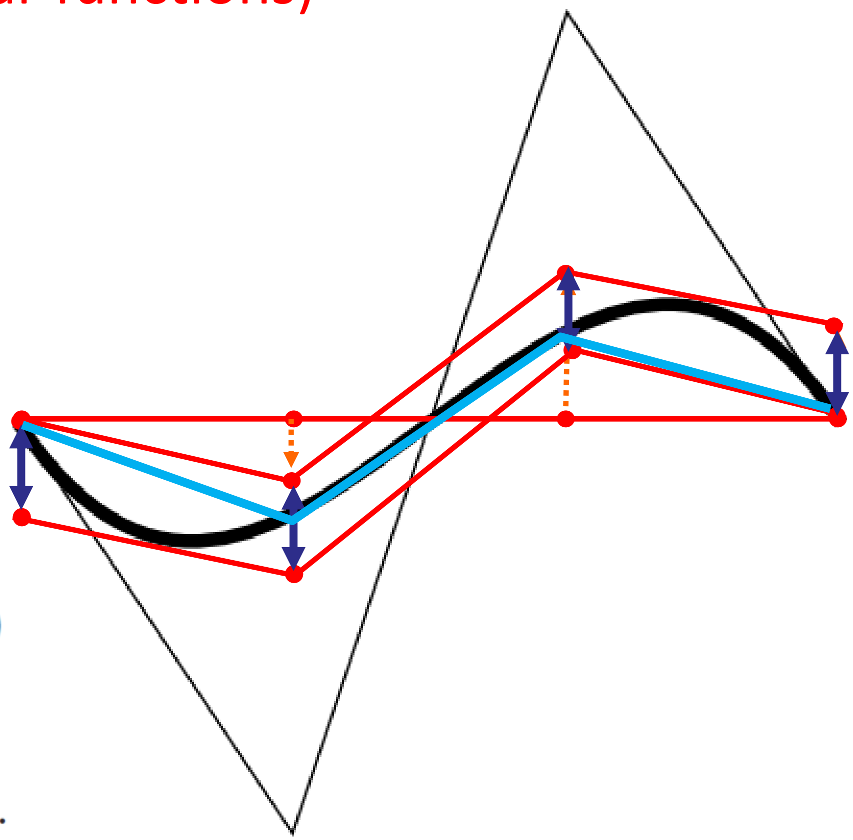
(tightly sandwich non-linear functions)

Non-linear functions

$$\underline{p} \leq p \leq \overline{p}$$

Lower **Upper**

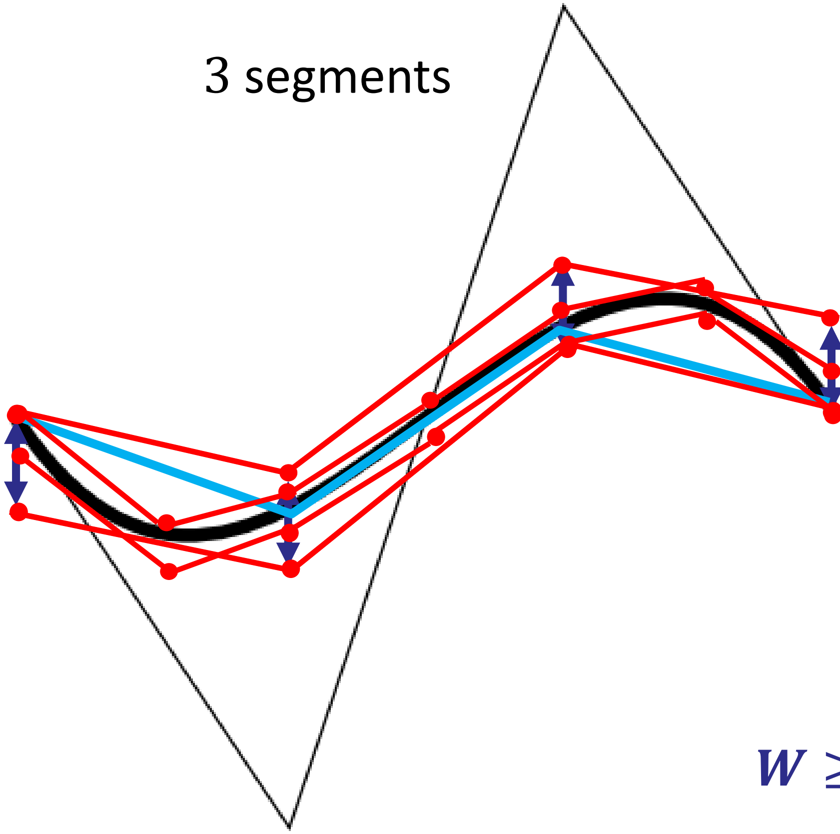
piecewise approximation

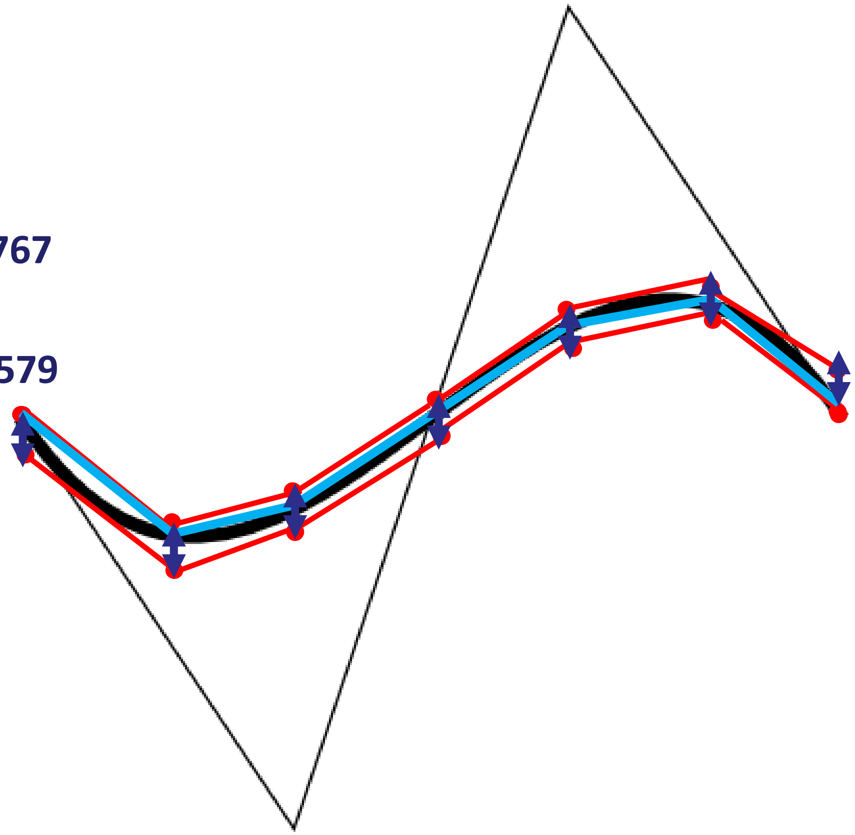


$$p(t) \leq \overline{p}(t) := \boxed{\ell(t)} + \sum_{j=1}^{d-1} \max\{0, \nabla_j^2 p\}\, \overline{\mathsf{a}_j^d}^m(t) + \sum_{j=1}^{d-1} \min\{0, \nabla_j^2 p\}\, \underline{\mathsf{a}_j^d}_m(t).$$

Slefe table

Why SLEFE ?

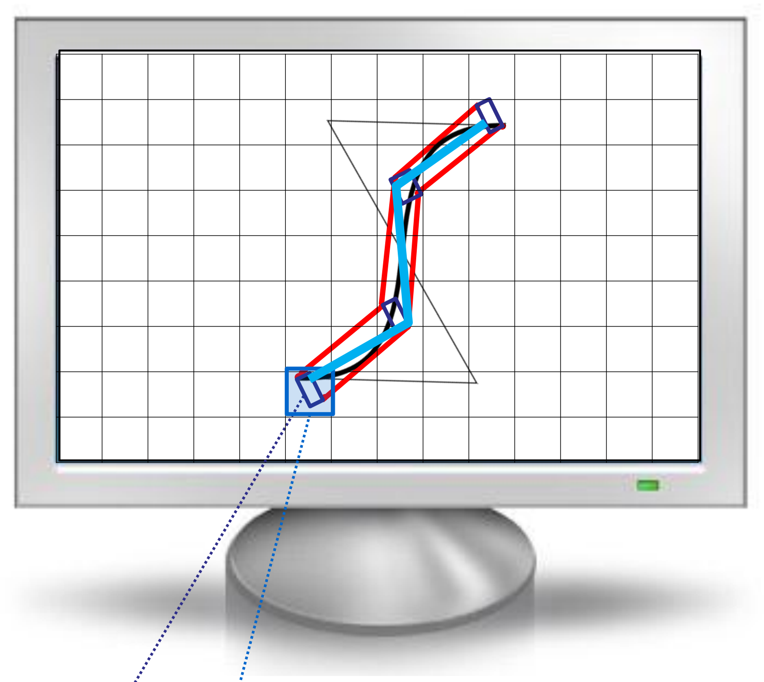




$$W \geq 4\overline{w_2}$$

$$W \geq h^2\overline{w_h}$$

Test & Predict Accuracy in Pixel



h fold subdivision $\frac{1}{h^2}\boxed{W} < \boxed{1}$ pixel $\Longrightarrow$ $h > \sqrt{W}$

$\Downarrow$

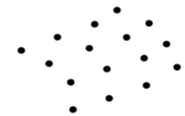
Tessellation Factor $3 * \sqrt{W}$

Outline

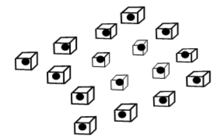
- What is Efficiency?
- What is Accuracy?
- Test & Predict (SLEFE)
- **Implementation & Demo**

Implementation

patch $\boldsymbol{p}$ (control points $\boldsymbol{c}_{ij}$), slefeTable

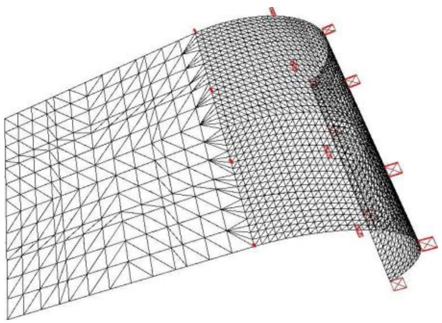


Compute Shader $\underline{\overline{\boldsymbol{p}}}$



TF$_p = 3 * \sqrt{W}$

TF per patch & edge
-> water-tight



Pixel-Accurate Rendering

Hull Shader — determine **TF**$_{p,q}$

Tessellator — (u, v) generation

Domain Shader — evaluation

Rasterizer

Pixel Shader — shading

Standard DX11 Rendering

Demo

Efficient Pixel-Accurate
Rendering of Curved Surfaces

Captured @ 1440 x 900 resolution
on NVIDIA GTX 480

Summary

Accurate

- Distortion < 1/2 pixel (= error not visible).

Efficient

- Triangles of maximal size & no recursion.

Automatic

- No need for manual LoD.

Fast

- > 300 fps for *130k* patches.

Thank You