Correct resolution rendering of trimmed spline surfaces

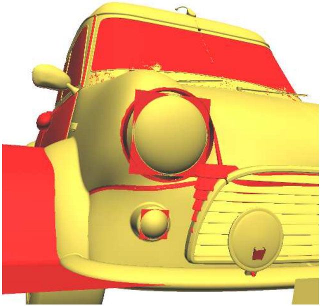
Ruijin Wu and Jörg Peters

University of Florida

SPM 2014

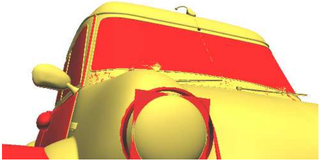
Industrial Workflow: Overfit + Trim

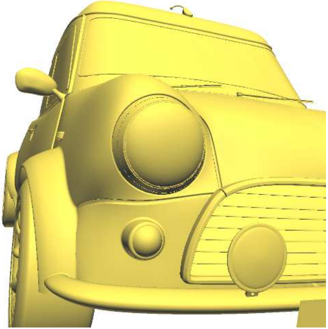
- Capture shape w/o considering other surfaces



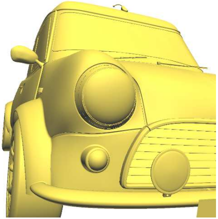
Industrial Workflow: Overfit + Trim

- Capture shape w/o considering other surfaces
- Trim the surfaces back to match constraints



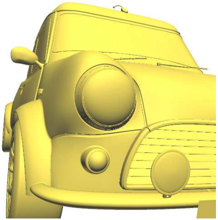


Benefits



Practice

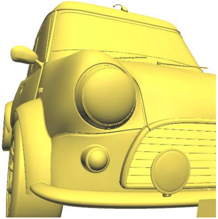
Benefits



Practice

- Accuracy currently: predefined triangulation level

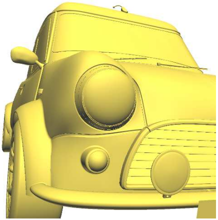
Benefits



Practice

- Accuracy currently: predefined triangulation level
- Latency adjust trim, modeler recomputes triangles

Benefits

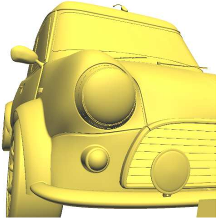


Practice

- Accuracy currently: predefined triangulation level
- Latency adjust trim, modeler recomputes triangles

Theory

Benefits



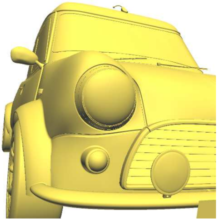
Practice

- Accuracy currently: predefined triangulation level
- Latency adjust trim, modeler recomputes triangles

Theory

- Accuracy domain $\leftrightarrow$ projected image resolution

Benefits



Practice

- Accuracy currently: predefined triangulation level
- Latency adjust trim, modeler recomputes triangles

Theory

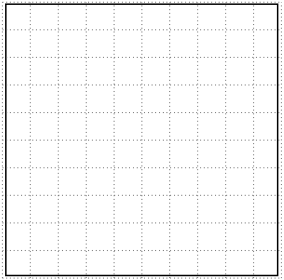
- Accuracy domain $\leftrightarrow$ projected image resolution
- Latency Lean parallel data structures

Goal

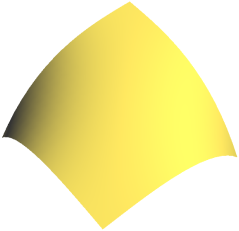
- High precision rendering
- Interactive frame rate

Outline

Surface and Trim Curve

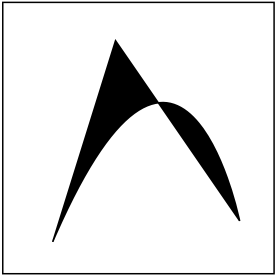


$$\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix} \xrightarrow[S]{}$$

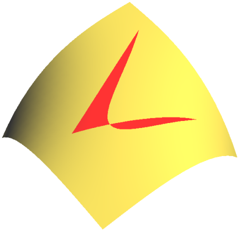


- Surface piece S maps the unit rectangle to 3-space.

Surface and Trim Curve

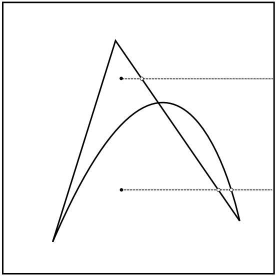


$$\xrightarrow[S]{\begin{pmatrix}u\\v\end{pmatrix}\to\begin{pmatrix}x\\y\\z\end{pmatrix}}$$

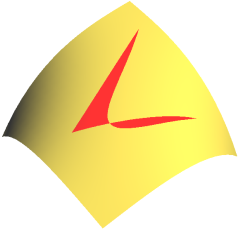


- Surface piece S maps the unit rectangle to 3-space.
- Trim curves define and restrict the domain of the surface.

Surface and Trim Curve



$$\xrightarrow[S]{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$

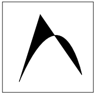


- Surface piece S maps the unit rectangle to 3-space.
- Trim curves define and restrict the domain of the surface.
- "inside" is determined by ray test *in the uv domain*.

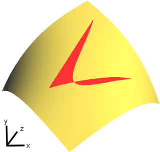
Outline

Ray trace or trim texture?

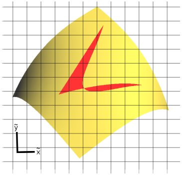
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$

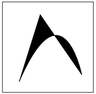


$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$

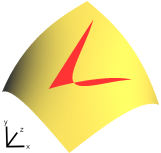


Ray trace or trim texture?

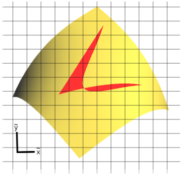
Map in from (u, v) to $(\tilde{\mathrm{x}}, \tilde{\mathrm{y}})$:



$$\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix} \longrightarrow$$



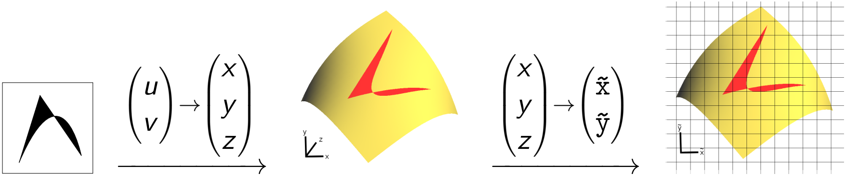
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{\mathrm{x}} \\ \tilde{\mathrm{y}} \end{pmatrix} \longrightarrow$$



Assume: Each pixel P(x(u,v)) knowns its uv.

Ray trace or trim texture?

Map in from (u, v) to $(\tilde{x}, \tilde{y})$:

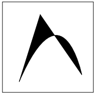


Assume: Each pixel P(x(u,v)) knowns its uv.

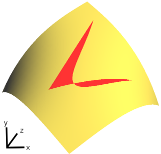
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]

Ray trace or trim texture?

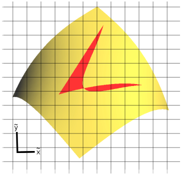
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$



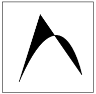
$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$



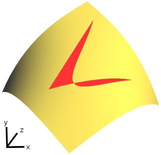
Assume: Each pixel P(x(u,v)) knowns its uv.

- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  + precise

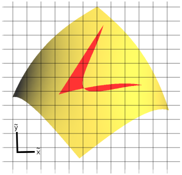
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$



$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$

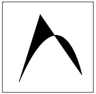


Assume: Each pixel P(x(u,v)) knowns its uv.

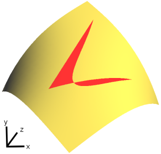
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  $+$ precise - several trim curves, many curve segments

Ray trace or trim texture?

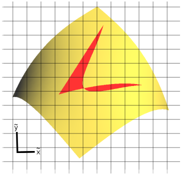
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$



$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$

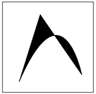


Assume: Each pixel P(x(u,v)) knowns its uv.

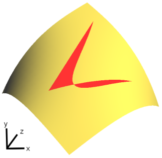
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  $+$ precise   - several trim curves, many curve segments
  - numerical stability (multiple roots, curves not implicit) [LB05]

Ray trace or trim texture?

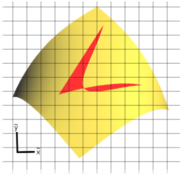
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

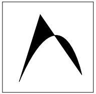


Assume: Each pixel P(x(u,v)) knowns its uv.

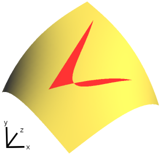
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  $+$ precise - several trim curves, many curve segments
  - numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]

Ray trace or trim texture?

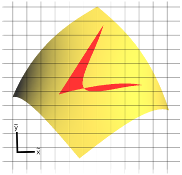
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$



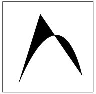
$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$



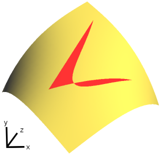
Assume: Each pixel P(x(u,v)) knowns its uv.

- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  $+$ precise - several trim curves, many curve segments
  - numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]
  $+$ fast lookup

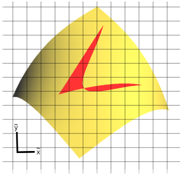
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

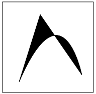


Assume: Each pixel P(x(u,v)) knowns its uv.

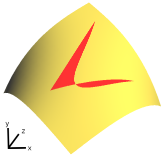
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  + precise - several trim curves, many curve segments
  - numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]
  + fast lookup + robust

Ray trace or trim texture?

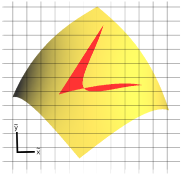
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\begin{pmatrix} u \\ v \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} \longrightarrow$$



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \longrightarrow$$

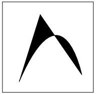


Assume: Each pixel P(x(u,v)) knowns its uv.

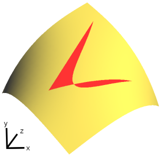
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  + precise - several trim curves, many curve segments
  - numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]
  + fast lookup + robust
  - resolution

Ray trace or trim texture?

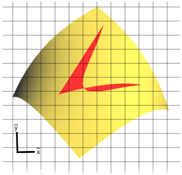
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$



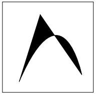
$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$



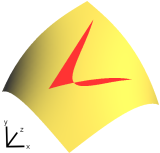
Assume: Each pixel P(x(u,v)) knowns its uv.

- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  + precise - several trim curves, many curve segments
  - numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]
  + fast lookup + robust
  - resolution - (re)computation (parallel write),

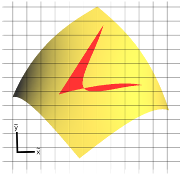
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\begin{pmatrix}u\\v\end{pmatrix}\to\begin{pmatrix}x\\y\\z\end{pmatrix}$$



$$\begin{pmatrix}x\\y\\z\end{pmatrix}\to\begin{pmatrix}\tilde{x}\\\tilde{y}\end{pmatrix}$$

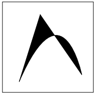


Assume: Each pixel P(x(u,v)) knowns its uv.

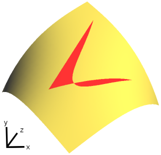
- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  \+ precise - several trim curves, many curve segments
  \- numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]
  \+ fast lookup + robust
  \- resolution - (re)computation (parallel write), - space

Ray trace or trim texture?

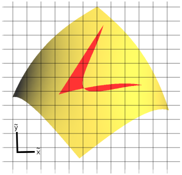
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



$$\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix} \longrightarrow$$



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \longrightarrow$$

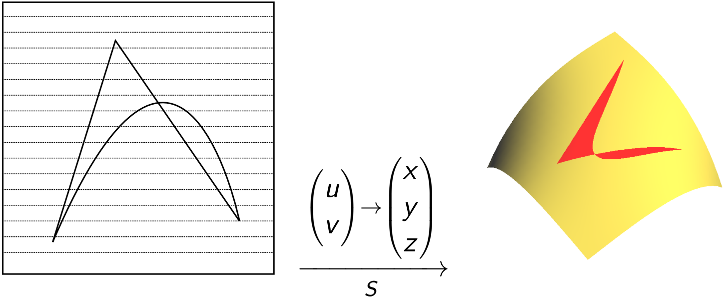


Assume: Each pixel P(x(u,v)) knowns its uv.

- ray trace (in uv domain) [e.g. Pabst06, Schollmeyer09]
  \+ precise - several trim curves, many curve segments
  \- numerical stability (multiple roots, curves not implicit) [LB05]
- in/out 'trim texture' (uv grid) [e.g. Guthe05]
  \+ fast lookup + robust
  \- resolution - (re)computation (parallel write), - space
- hybrid = robust + fast + sufficient precision ?

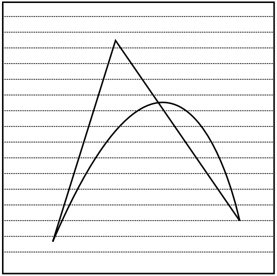
Outline

Predicting Correct Resolution

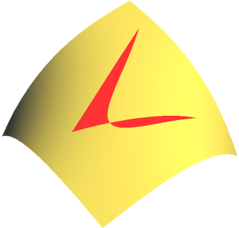


- Surface piece S maps the unit rectangle to 3-space.
- Trim curves define and restrict the domain of the surface.
- "inside" is determined by ray test *in the uv domain*.
- 'fat' rays – how fat?

Predicting Correct Resolution



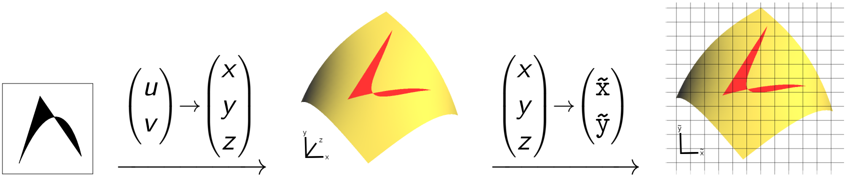
$$\xrightarrow[S]{\begin{pmatrix}u\\v\end{pmatrix}\to\begin{pmatrix}x\\y\\z\end{pmatrix}}$$



- Surface piece S maps the unit rectangle to 3-space.
- Trim curves define and restrict the domain of the surface.
- "inside" is determined by ray test *in the uv domain*.
- 'fat' rays – how fat? 'simplify' trim curve – how simple?

Predicting correct resolution

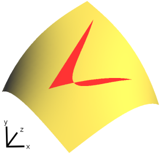
Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



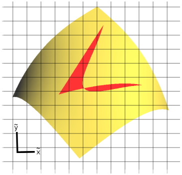
Predicting correct resolution

Map in from (u, v) to $(\tilde{x}, \tilde{y})$:

$$\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

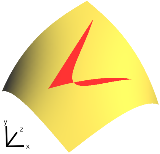


'pull back' pixel grid   need distinct pre-images

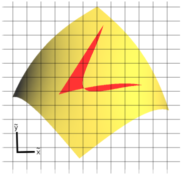
Predicting correct resolution

Map in from (u, v) to $(\tilde{x}, \tilde{y})$:

$$\xrightarrow{\begin{pmatrix} u \\ v \end{pmatrix} \to \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$



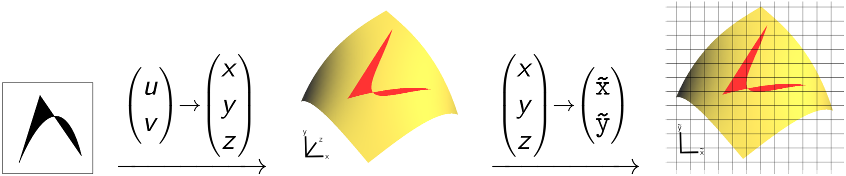
$$\xrightarrow{\begin{pmatrix} x \\ y \\ z \end{pmatrix} \to \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}}$$



'pull back' pixel grid   need distinct pre-images avoid non-linear grid

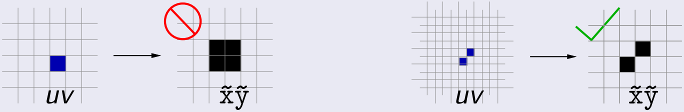
Predicting correct resolution

Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



'pull back' pixel grid   need distinct pre-images avoid non-linear grid

Correct Resolution



Math 101: Determine the v-scan line density

The screen space distance between two v-scan lines is:

$$\left|\tilde{\mathtt{x}}(u,v) - \tilde{\mathtt{x}}(u,v+h)\right| = h\left|\tilde{\mathtt{x}}_v(u,v^*)\right|, \qquad \tilde{\mathtt{x}}_v := \frac{\partial\tilde{\mathtt{x}}}{\partial v}.$$

Math 101: Determine the v-scan line density

The screen space distance between two v-scan lines is:

$$|\tilde{\mathtt{x}}(u,v) - \tilde{\mathtt{x}}(u,v+h)| = h\,|\tilde{\mathtt{x}}_v(u,v^*)|\,, \qquad \tilde{\mathtt{x}}_v := \frac{\partial \tilde{\mathtt{x}}}{\partial v}.$$

If, for all $v \in V_i$ and the v-scan line *spacing* $h > 0$,

$$h\ \rho_i(v) < 1,$$
$$\rho_i(v) := \max\{\sup_u |\tilde{\mathtt{x}}_v(u,v)|\,, \sup_u |\tilde{\mathtt{y}}_v(u,v)|\},$$

Math 101: Determine the v-scan line density

The screen space distance between two v-scan lines is:

$$|\tilde{\mathtt{x}}(u,v) - \tilde{\mathtt{x}}(u,v+h)| = h\,|\tilde{\mathtt{x}}_v(u,v^*)|\,, \qquad \tilde{\mathtt{x}}_v := \frac{\partial \tilde{\mathtt{x}}}{\partial v}.$$

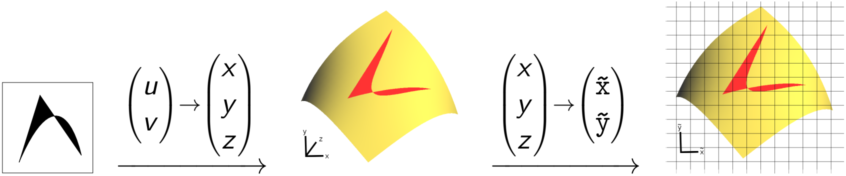
If, for all $v \in V_i$ and the v-scan line *spacing* $h > 0$,

$$h\ \rho_i(v) < 1,$$
$$\rho_i(v) := \max\{\sup_u |\tilde{\mathtt{x}}_v(u,v)|\,, \sup_u |\tilde{\mathtt{y}}_v(u,v)|\},$$

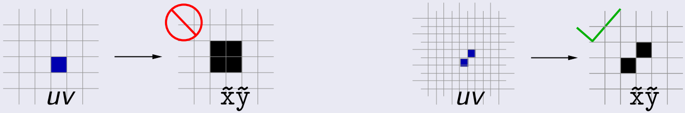
then the $\tilde{\mathtt{x}}$-distance between the screen images of the two v-scan lines $\tilde{\mathtt{x}}(u,v_j)$ and $\tilde{\mathtt{x}}(u,v_j+h)$ is less than a pixel and so is the $\tilde{\mathtt{y}}$-distance.

Summary: Predicting correct resolution

Map in from (u, v) to $(\tilde{x}, \tilde{y})$:



Correct Resolution

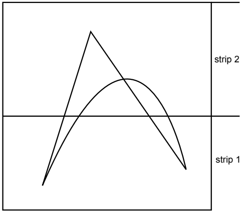


v-scan line spacing

$$h < 1/\rho_i(v) \qquad \rho_i(v) := \max\{\sup_u |\tilde{x}_v(u, v)|, \sup_u |\tilde{y}_v(u, v)|\}$$

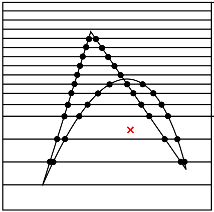
Outline

Two-level scanline hierarchy



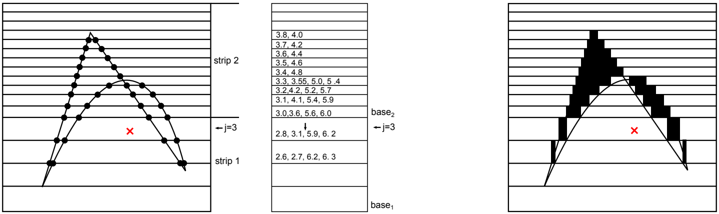
- Uniformly partition the domain into n_V v-strips. (First level)

Two-level scanline hierarchy

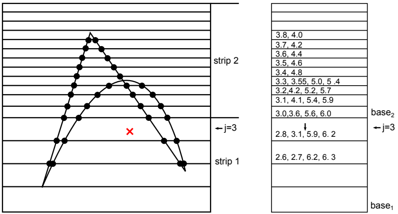


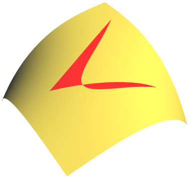
- Uniformly partition the domain into n_V v-strips. (First level)
- Uniformly partition each v-strip into its own number of v-scan lines that guarantees correct resolution. (Second level)

Two-level scanline hierarchy



- Uniformly partition the domain into n_V v-strips. (First level)
- Uniformly partition each v-strip into its own number of v-scan lines that guarantees correct resolution. (Second level)
- Store the u-coordinate of intersections for each v-scan line.



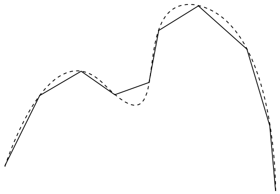


- Uniformly partition the domain into n_V v-strips. (First level)
- Uniformly partition each v-strip into its own number of v-scan lines that guarantees correct resolution. (Second level)
- Store the u-coordinate of intersections for each v-scan line.
- Trim test: Look up position of each pixel's pre-image in the table.

Outline

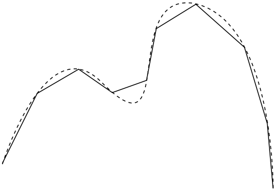
Tessellation of curve and surface

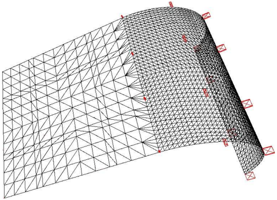
- The trim curve is tessellated into line segments.



Tessellation of curve and surface

- The trim curve is tessellated into line segments.
- The surface is tessellated into triangles.

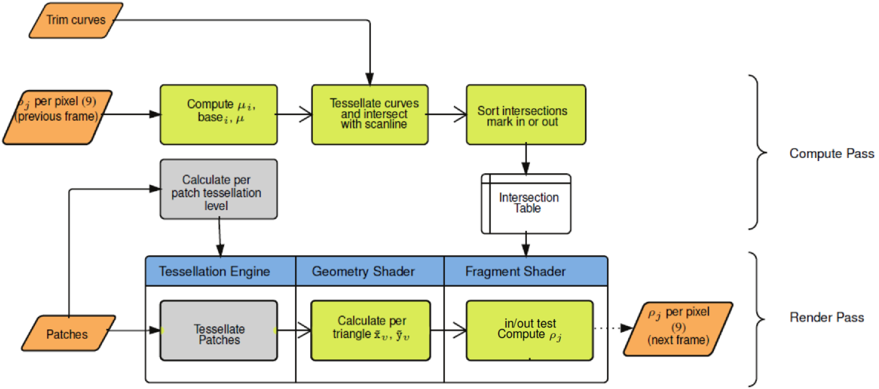




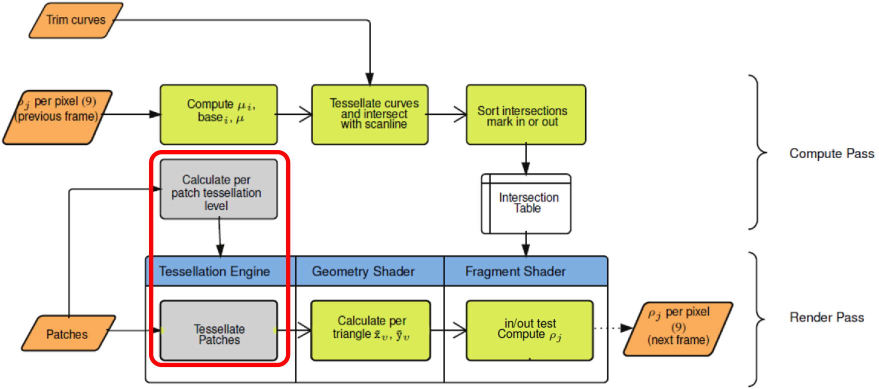
Subdividable linear efficient function envelopes = SLEFE

A tight bound of the deviation between the curve/surface and its piecewise linear approximation [Yeo et al 2012].

Using the GPU pipeline

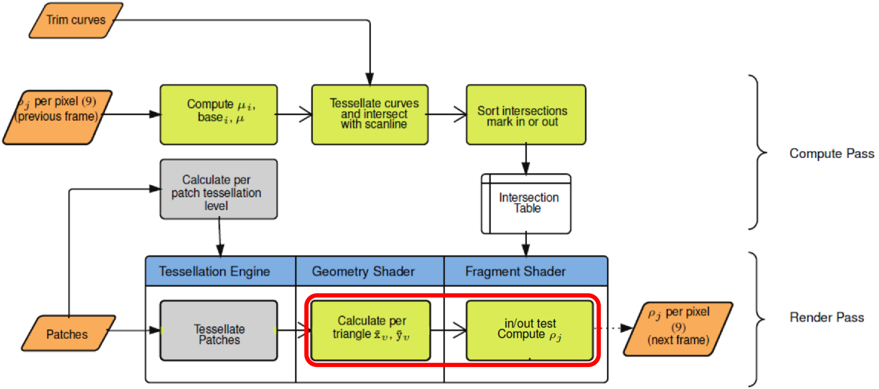


Using the GPU pipeline



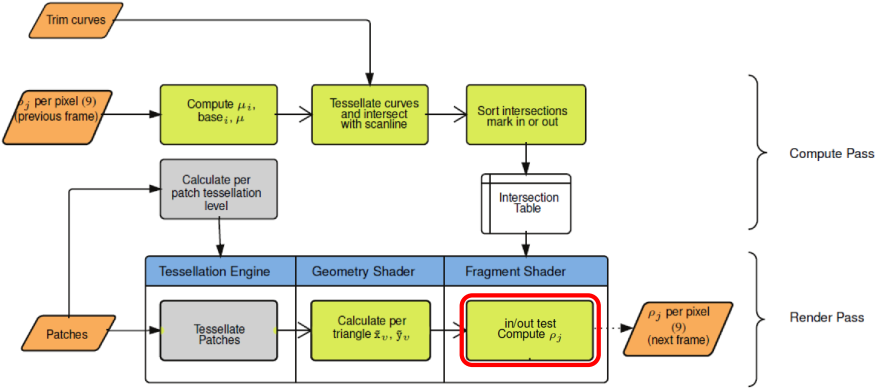
- iPass algorithm [Yeo et al. 2012] for surface rendering
  https://bitbucket.org/surflab/ipass_gl4

Using the GPU pipeline

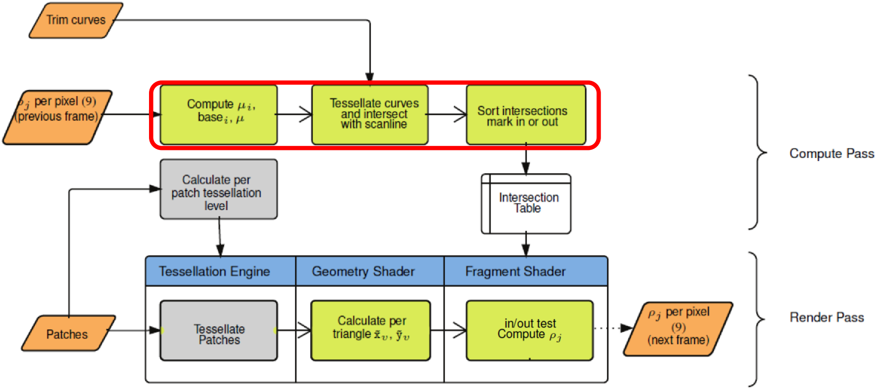


- iPass algorithm [Yeo et al. 2012] for surface rendering
- Calculate v-scan line spacing, ρ_j per pixel

Using the GPU pipeline



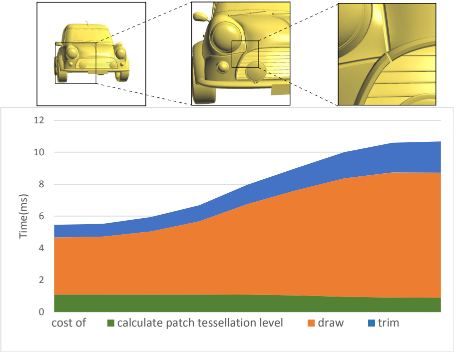
- iPass algorithm [Yeo et al. 2012] for surface rendering
- Calculate v-scan line spacing, ρ_j per pixel
- in/out test per pixel



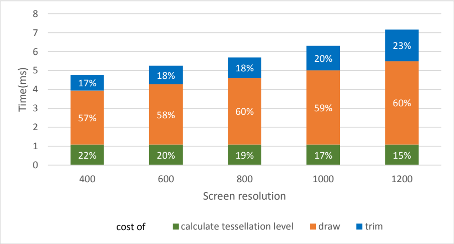
- iPass algorithm [Yeo et al. 2012] for surface rendering
- Calculate v-scan line spacing, ρ_j per pixel
- in/out test per pixel
- Build u-intercept table

Outline

Real-time interactive frame rate.



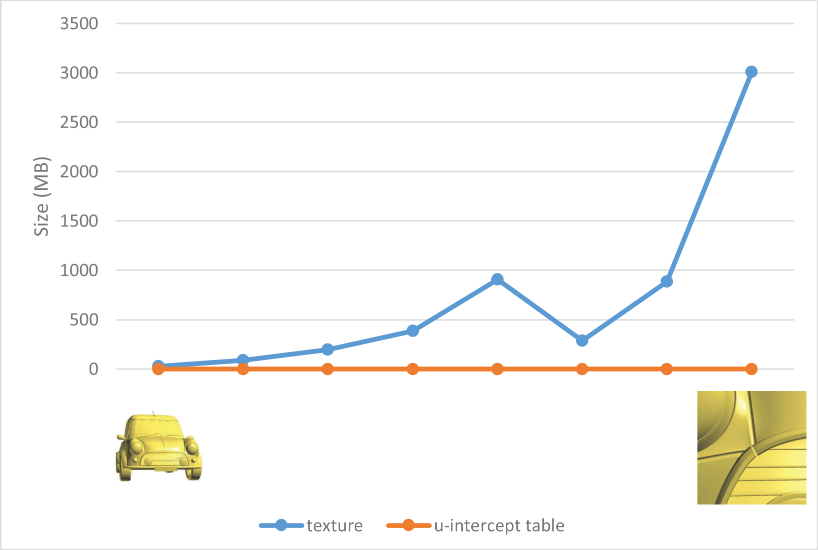
(a) Performance at different zoom levels.



(b) Performance at different screen resolution.

GPU memory usage

Compare to texture based technique



Outline

Contributions

- Determine a *provably* optimal scan density for trim test.

Contributions

- Determine a *provably* optimal scan density for trim test.
- Efficient *semi-uniform* scan-line data structure.

Contributions

- Determine a *provably* optimal scan density for trim test.
- Efficient *semi-uniform* scan-line data structure.
- GPU friendly: real-time trim curve editing with simple add-on.

Contributions

- Determine a *provably* optimal scan density for trim test.
- Efficient *semi-uniform* scan-line data structure.
- GPU friendly: real-time trim curve editing with simple add-on.

- Questions?