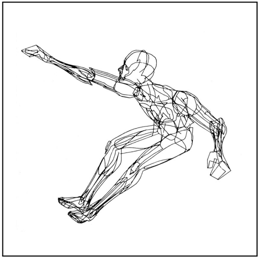
A Hybrid Method for Computing Apparent Ridges

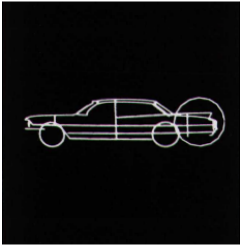
Eric Jardim
Luiz Henrique de Figueiredo

impa

Evolution of realism in computer graphics

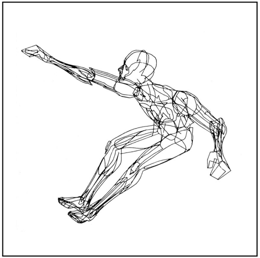
the 60s

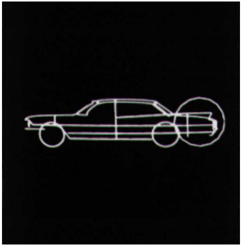




Evolution of realism in computer graphics

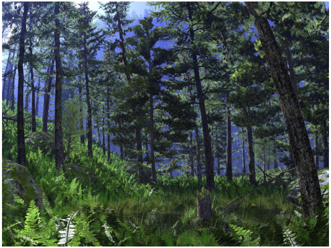
the 60s





⟹

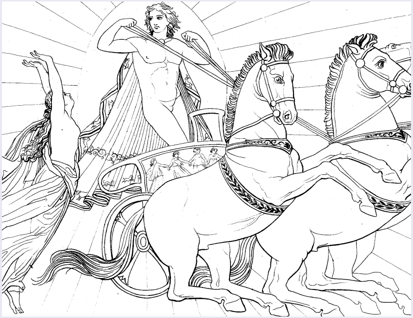
nowadays

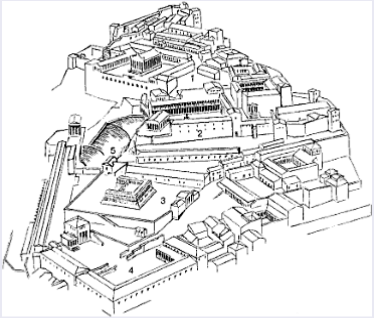


Line drawings

- Line drawings are the most basic type of artistic expression
- Compact, clear and effective
- Expressive line drawing of 3D models $\Rightarrow$ NPR problem

Examples of line drawings

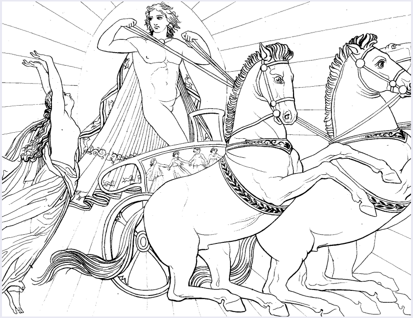


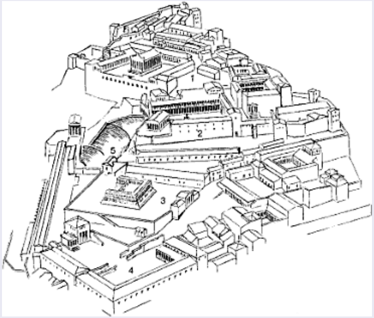


Line drawings

- Line drawings are the most basic type of artistic expression
- Compact, clear and effective
- Expressive line drawing of 3D models ⇒ NPR problem
- Apparent ridges is a type of line drawing from 3D models

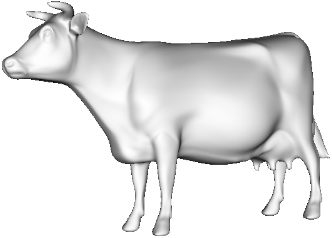
Examples of line drawings

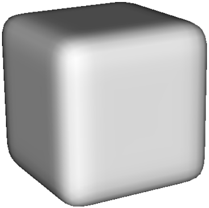




Previous work

- There are several line definitions and methods for extraction
- We will review some related curves
- They will be extracted them from the following shaded models



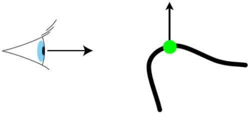


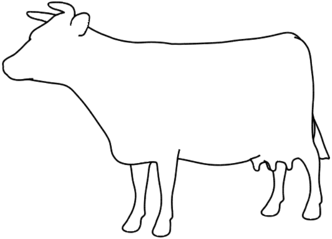
Contours

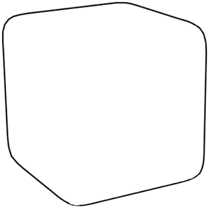
Definition

Normal perpendicular to view direction: $\langle n, v \rangle = 0$

- First-order, view-dependent
- Essential, but don't capture enough information





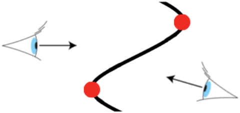


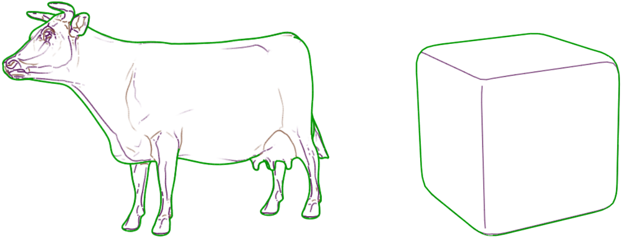
Ridges & valleys

Definition

Extrema of curvature in principal direction: $D_{e_1} k_1 = 0, \; |k1| \geq |k2|$

- Second-order, not view-dependent
- Angles are too sharp and are visually rigid



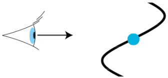


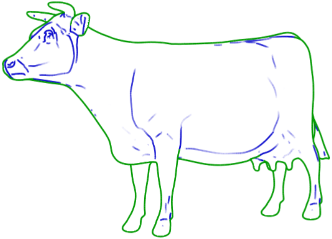
Suggestive contours

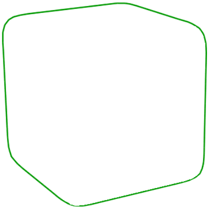
Definition

Zeros of radial curvature in view direction projected onto the tangent plane: $D_w n = 0, \ w = (v)_{T_p S}$

- Second-order, view-dependent
- Naturally extend contours, but don't appear on convex regions

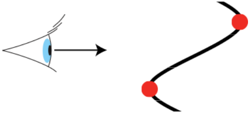


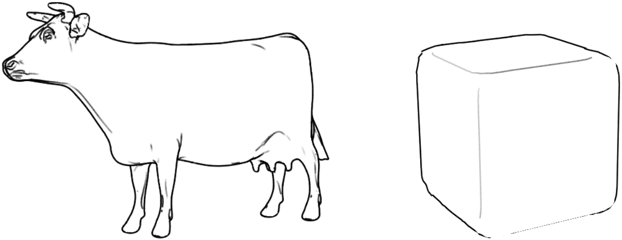




Apparent ridges

- Second-order, view-dependent
- Appear also on convex regions
- Does not need to be combined with contours





Apparent ridges definition

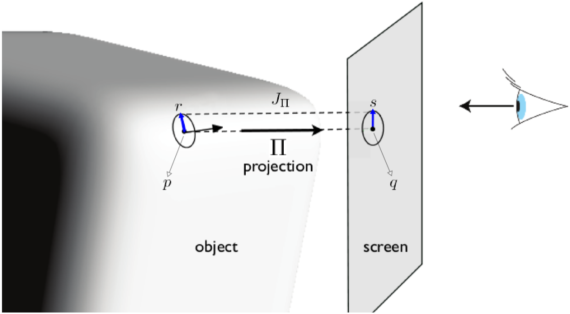
Apparent ridges are based on a new geometric property:

View-dependent curvature

- Key idea: measure how the surface bends with respect to the viewpoint, taking into account the perspective transformation
- Plays analogue role for apparent ridges as ordinary curvature does for ridges and valleys

View-dependent curvature

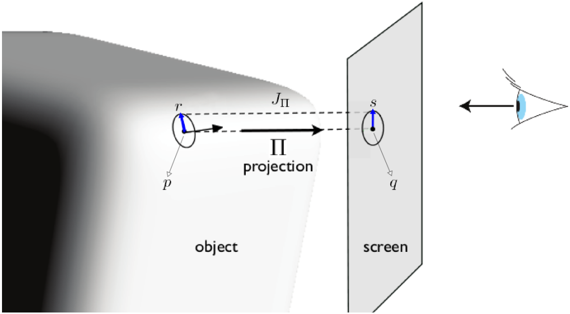
- Let $p \in M$, n normal, Π parallel projection and $q = \Pi(p)$
- If p is not a contour point $\Rightarrow$ Π is locally invertible
- J_Π maps vectors from T_pM to the screen
- Let $\tilde{n}(q) = n \circ \Pi^{-1}(q)$



View-dependent curvature

- The shape operator on T_pM is defined as $S(r) = D_r n$
- Define $Q(s) = D_s \tilde{n}$
- Since $D_s \tilde{n} = D_{r(s)} n = S(r(s))$ and $r = J_\Pi^{-1}(s)$ then

$$Q = S \circ J_\Pi^{-1}$$

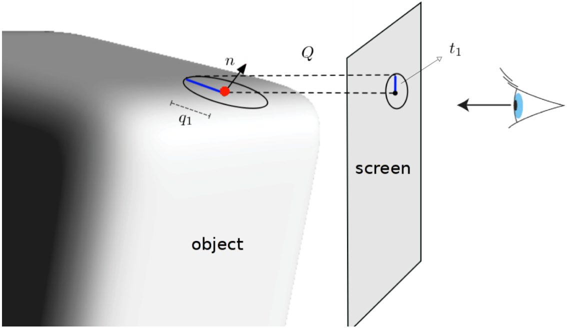


View-dependent curvature

Definition

Maximum view-dependent curvature and principal direction

$$q_1 = \max_{||s||=1} ||Q(s)|| \qquad t_1 = \text{ direction } ||Q(s)|| \text{ is maximum}$$

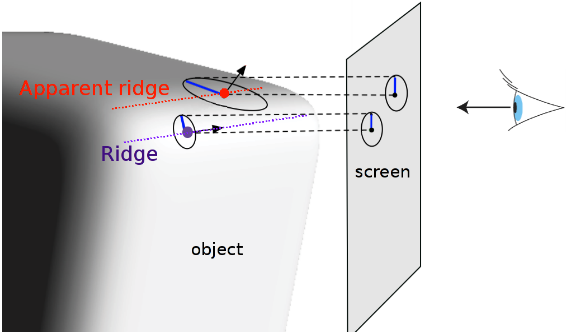


Apparent ridges

Definition

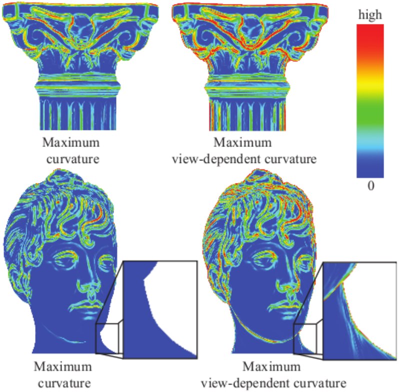
Apparent ridges := local maxima of q_1 in the t_1 direction

$$D_{t_1} q_1 = 0 \quad \text{and} \quad D_{t_1}\left(D_{t_1} q_1\right) < 0$$



Contours

- When $p \longrightarrow$ contour $\Rightarrow q_1 \longrightarrow \infty$
- Extending the definition, q_1 achieves a maximum at ∞
- Contours are apparent ridges and can be extracted together



Motivation

- Apparent ridges were defined in Judd's work [1]
- Definition + CPU implementation on triangle meshes
- Excelent visual results compared to other lines, but...

[1] T. Judd, *Apparent Ridges for Line Drawing*. Masters Thesis, Computer Science, MIT, Jan 2007

Motivation

- Apparent ridges were defined in Judd's work [1]
- Definition + CPU implementation on triangle meshes
- Excelent visual results compared to other lines, but...
- Lower performance compared to other lines

[1] T. Judd, *Apparent Ridges for Line Drawing*. Masters Thesis, Computer Science, MIT, Jan 2007

Motivation

- Apparent ridges were defined in Judd's work [1]
- Definition + CPU implementation on triangle meshes
- Excelent visual results compared to other lines, but...
- Lower performance compared to other lines

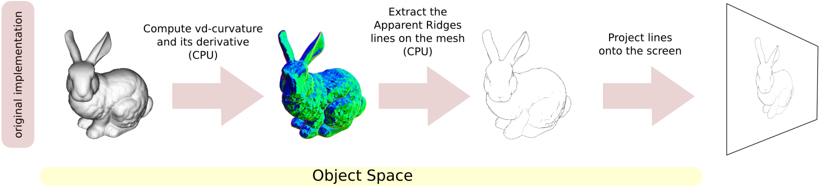
Main motivation

Exploit the GPU processing power to speed up extraction

[1] T. Judd, *Apparent Ridges for Line Drawing*. Masters Thesis, Computer Science, MIT, Jan 2007

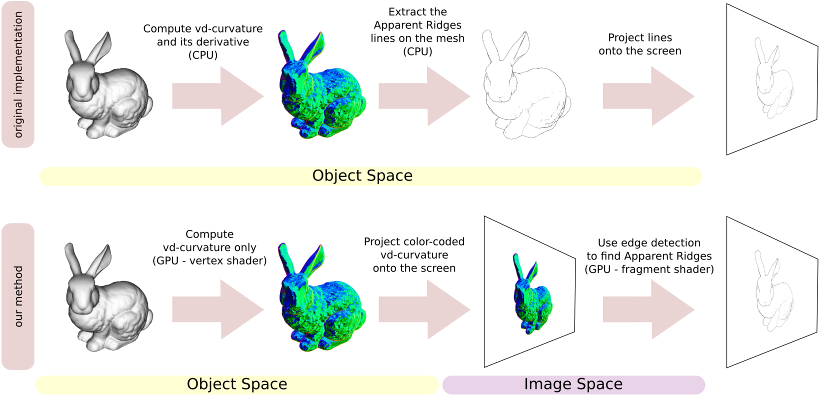
Overview

- Judd's is an object-space method: compute per-vertex $D_{t_1} q_1$
- Direct approach: port Judd's code to a GPU vertex shader
- Problem: vertex adjacency $\Rightarrow$ not easy to port



Overview

- Judd's is an object-space method: compute per-vertex $D_{t_1} q_1$
- Direct approach: port Judd's code to a GPU vertex shader
- Problem: vertex adjacency $\Rightarrow$ not easy to port
- Our solution: split the method in vertex and fragment stages



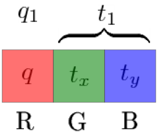
Object-space stage

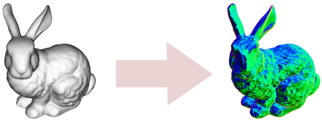
- Compute q_1 and t_1 at each vertex of the mesh
- Ported Judd's code to vertex shader
- The required data are passed as color and texture
- Since $q_1 \in [0, \infty] \Rightarrow$ scaled and truncated to fit $[0, 1]$

$$q = 2^{\tau} q_1 \qquad \tau \text{ is user-controlled}$$

- Each t_1 coordinate $\in [-1, 1] \Rightarrow$ affine transform to $[0, 1]$

$q_1, t_1 \mapsto q, t_x, t_y$





Object-space stage

- Each (q, t_x, t_y) is rasterized as vertex color to an off-screen buffer.
- Usual bilinear interpolation by the graphics pipeline
- The off-screen buffer is the input to the next stage

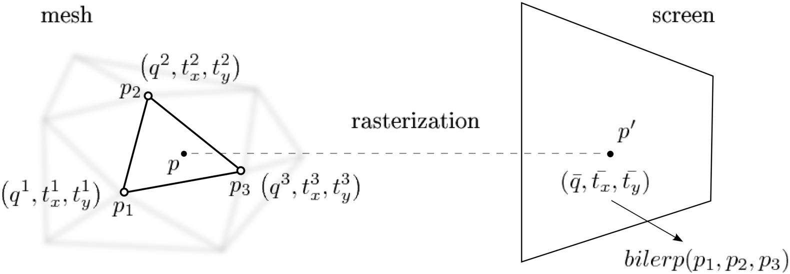
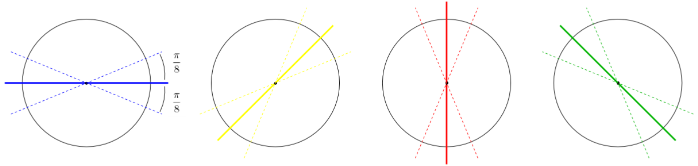


Image-space stage

- Edge detection to find the maxima of q_1 in t_1 direction
- Laplacian-like filter $\Rightarrow$ t_1 quantized: $\theta \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right\}$
- The λ parameter weights the filter in the θ direction

$t_1 \in [0-\frac{\pi}{8}, 0+\frac{\pi}{8}] \Rightarrow \theta = 0$ $\quad$ $t_1 \in [\frac{\pi}{4}-\frac{\pi}{8}, \frac{\pi}{4}+\frac{\pi}{8}] \Rightarrow \theta = \frac{\pi}{4}$ $\quad$ $t_1 \in [\frac{\pi}{2}-\frac{\pi}{8}, \frac{\pi}{2}+\frac{\pi}{8}] \Rightarrow \theta = \frac{\pi}{8}$ $\quad$ $t_1 \in [\frac{3\pi}{4}-\frac{\pi}{8}, \frac{3\pi}{4}+\frac{\pi}{8}] \Rightarrow \theta = \frac{3\pi}{4}$



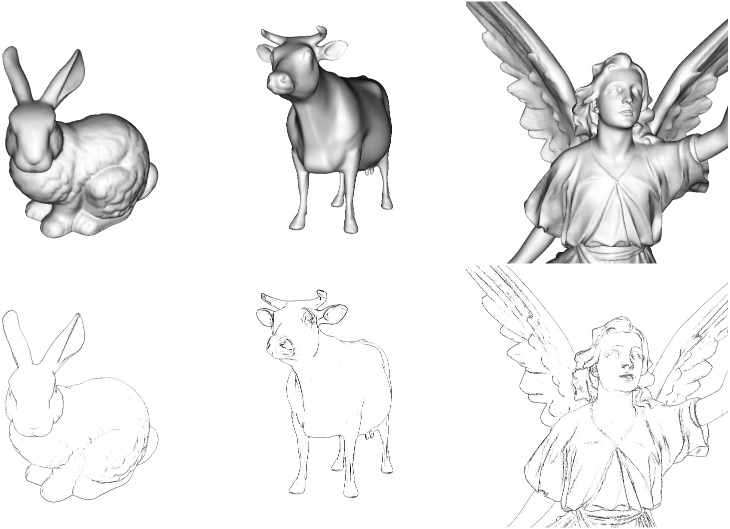
$1+\lambda$	1	$1+\lambda$
$1+2\lambda$	$-8-8\lambda$	$1+2\lambda$
$1+\lambda$	1	$1+\lambda$

1	$1+\lambda$	$1+2\lambda$
$1+\lambda$	$-8-8\lambda$	$1+\lambda$
$1+2\lambda$	$1+\lambda$	1

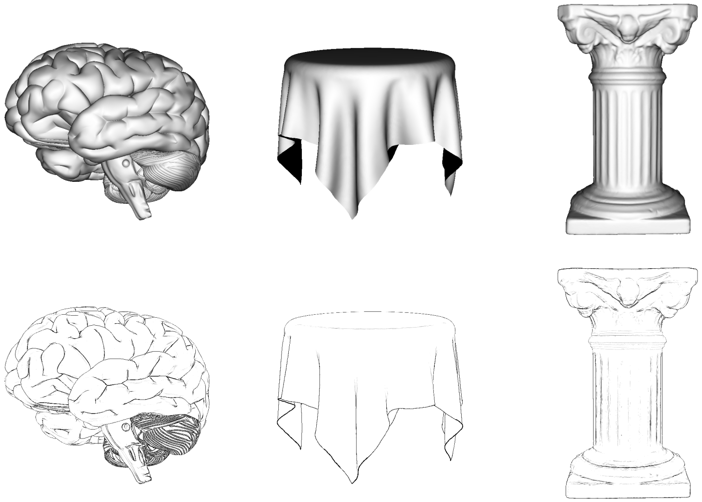
$1+\lambda$	$1+2\lambda$	$1+\lambda$
1	$-8-8\lambda$	1
$1+\lambda$	$1+2\lambda$	$1+\lambda$

$1+2\lambda$	$1+\lambda$	1
$1+\lambda$	$-8-8\lambda$	$1+\lambda$
1	$1+\lambda$	$1+2\lambda$

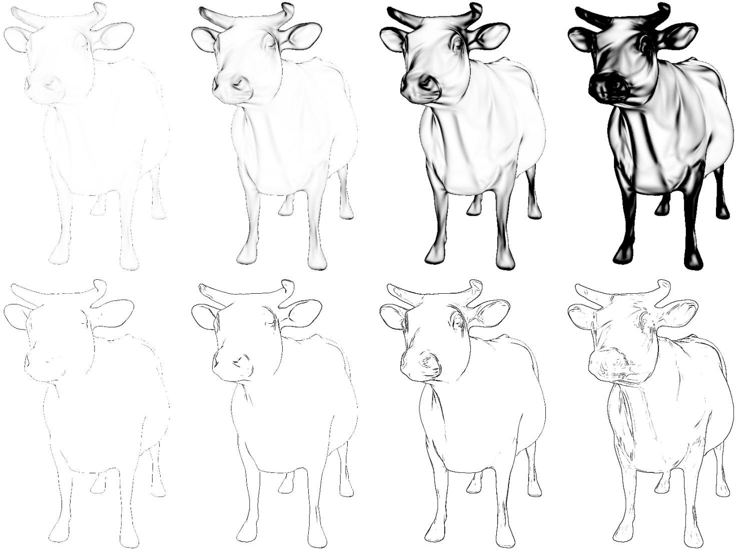
Our results



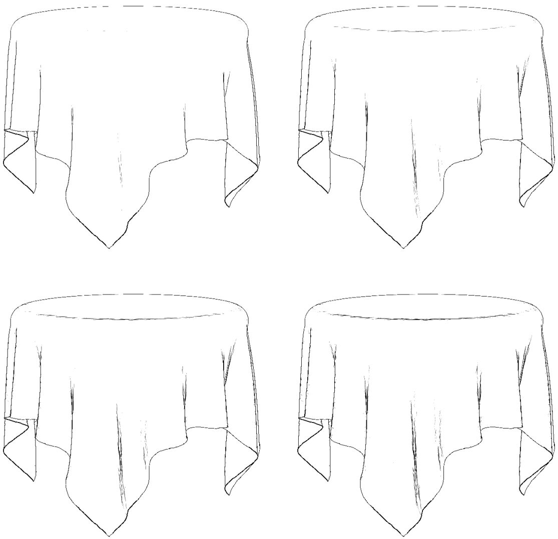
Our results



Parameter variation τ

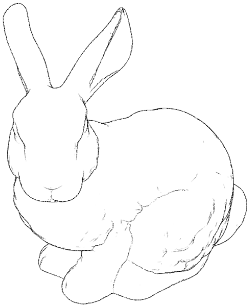


Parameter variation λ

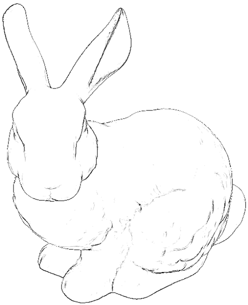


Comparison with Judd's method

Judd's

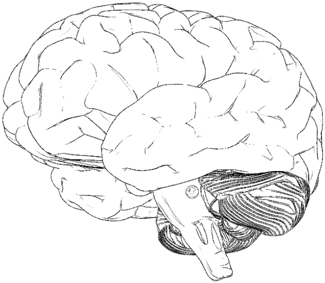


Our

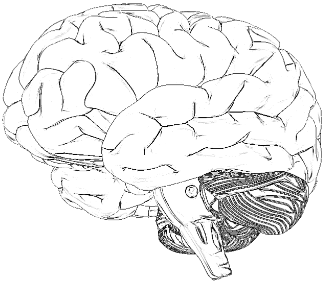


Comparison with Judd's method

Judd's

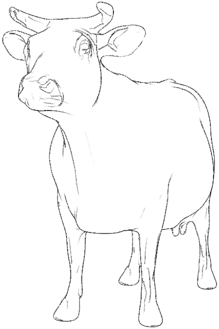


Our



Comparison with Judd's method

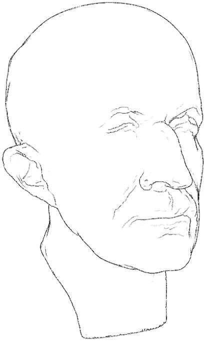
Judd's



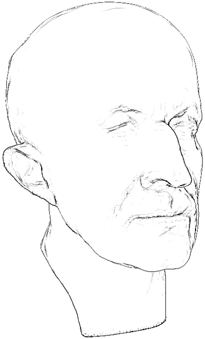
Our

Comparison with Judd's method

Judd's



Our



Performance comparison on laptop

Model	# Vertices	Judd's (FPS)	Ours (FPS)	Speedup
roundedcube	1538	202	90	0.5
torus	4800	148	84	0.6
tablecloth	22653	32	64	2.0
hippo	23105	41	66	1.6
cow	46433	24	58	2.4
horse	48484	22	52	2.4
maxplanck	49132	20	46	2.3
bunny	72027	15	42	2.8
elephant	78792	14	47	3.4
golfball	122882	9	30	3.3
igea	134345	8	29	3.6
armadillo	172974	5	18	3.6
column	262653	3	9	3.0
lucy	262909	4	13	3.3
heptoroid	286678	4	21	5.3
brain	294012	4	20	5.0

Performance comparison on workstation

Model	# Vertices	Judd's (FPS)	Ours (FPS)	Speedup
roundedcube	1538	600	1100	1.8
torus	4800	270	870	3.2
tablecloth	22653	32	160	5.0
hippo	23105	42	160	3.8
cow	46433	23	198	8.6
horse	48484	26	147	5.6
maxplanck	49132	22	90	4.1
bunny	72027	16	102	6.4
elephant	78792	15	113	7.5
golfball	122882	9	52	5.8
igea	134345	8	58	7.3
armadillo	172974	5	30	6.0
column	262653	3	15	5.0
lucy	262909	4	26	6.5
heptoroid	286678	5	19	3.8
brain	294012	4	34	8.5

Performance comparison (laptop and workstation)

Model	# Vert	Judd's L	Judd's W	Ours L	Ours W
roundedcube	1538	202	600	90	1100
torus	4800	148	270	84	870
tablecloth	22653	32	32	64	160
hippo	23105	41	42	66	160
cow	46433	24	23	58	198
horse	48484	22	26	52	147
maxplanck	49132	20	22	46	90
bunny	72027	15	16	42	102
elephant	78792	14	15	47	113
golfball	122882	9	9	30	52
igea	134345	8	8	29	58
armadillo	172974	5	5	18	30
column	262653	3	3	9	15
lucy	262909	4	4	13	26
heptoroid	286678	4	5	21	19
brain	294012	4	4	20	34

Conclusion

- Apparent ridges are perceptually pleasant and visually competitive
- We presented a new method that:
  - Replaces estimation of $D_{t_1} q_1$ with simple edge detection
  - Performs computations on GPU
  - Produces faster results
  - Provides similar image quality
- Apparent ridges are now even more competitive

Future work

- Enhace image quality:
  - Try other edge detection filters
  - Phong-like shading of q_1 and t_1
- Modular image-space stage: extract apparent ridges from volume data and implict models
- Adaptation of the method to extract other lines
- View-dependent curvature: shading and modeling

A Hybrid Method for Computing Apparent Ridges

Eric Jardim
Luiz Henrique de Figueiredo

impa

SIBGRAPI 2010
GRAMADO.RS.BRAZIL