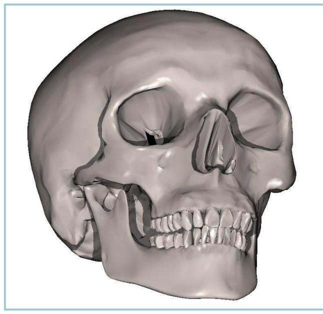
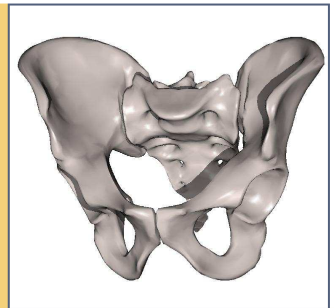
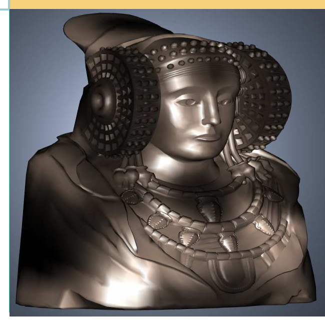
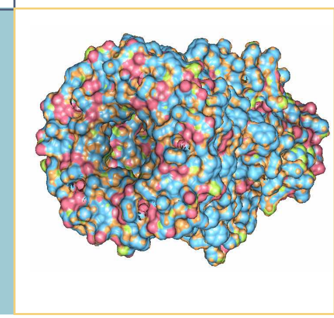
Visual Computing: At the Crossroads of Realism, Modeling, and Perception

Amitabh Varshney

Department of Computer Science

University of Maryland at College Park

Graphics: The First Revolution

- The birth of Raster Graphics three decades ago
- Consumer-driven demand for TVs
- VLSI-driven fall in memory prices
- Affordable graphics systems with CRTs and framebuffers

Computer Games

Age of Empires II
Microsoft Games

NFL Madden
Electronic Arts

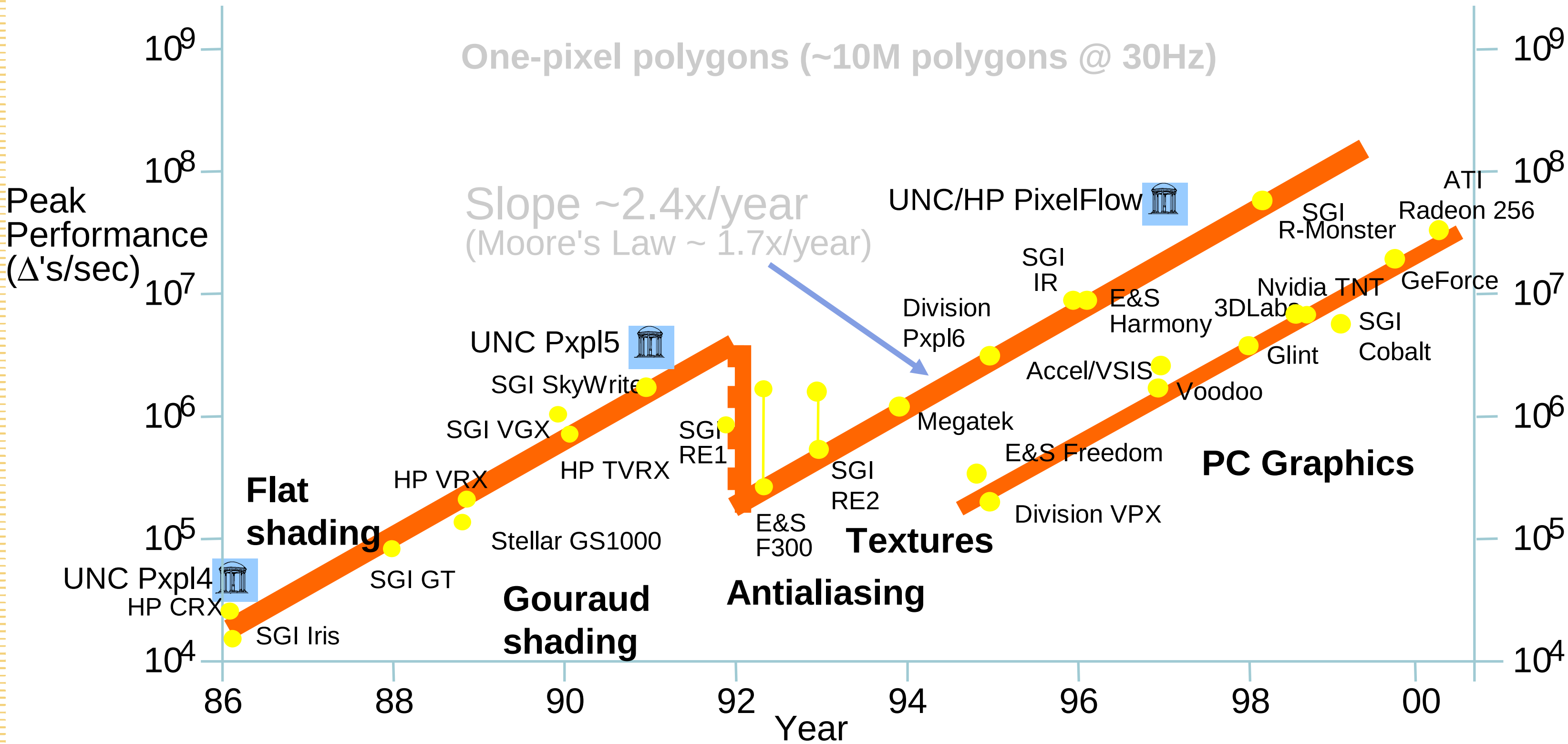
Battlefield II
Electronic Arts

Ice Age 2: The Meltdown
Sierra Entertainment

Halo 2
Bungie Studios

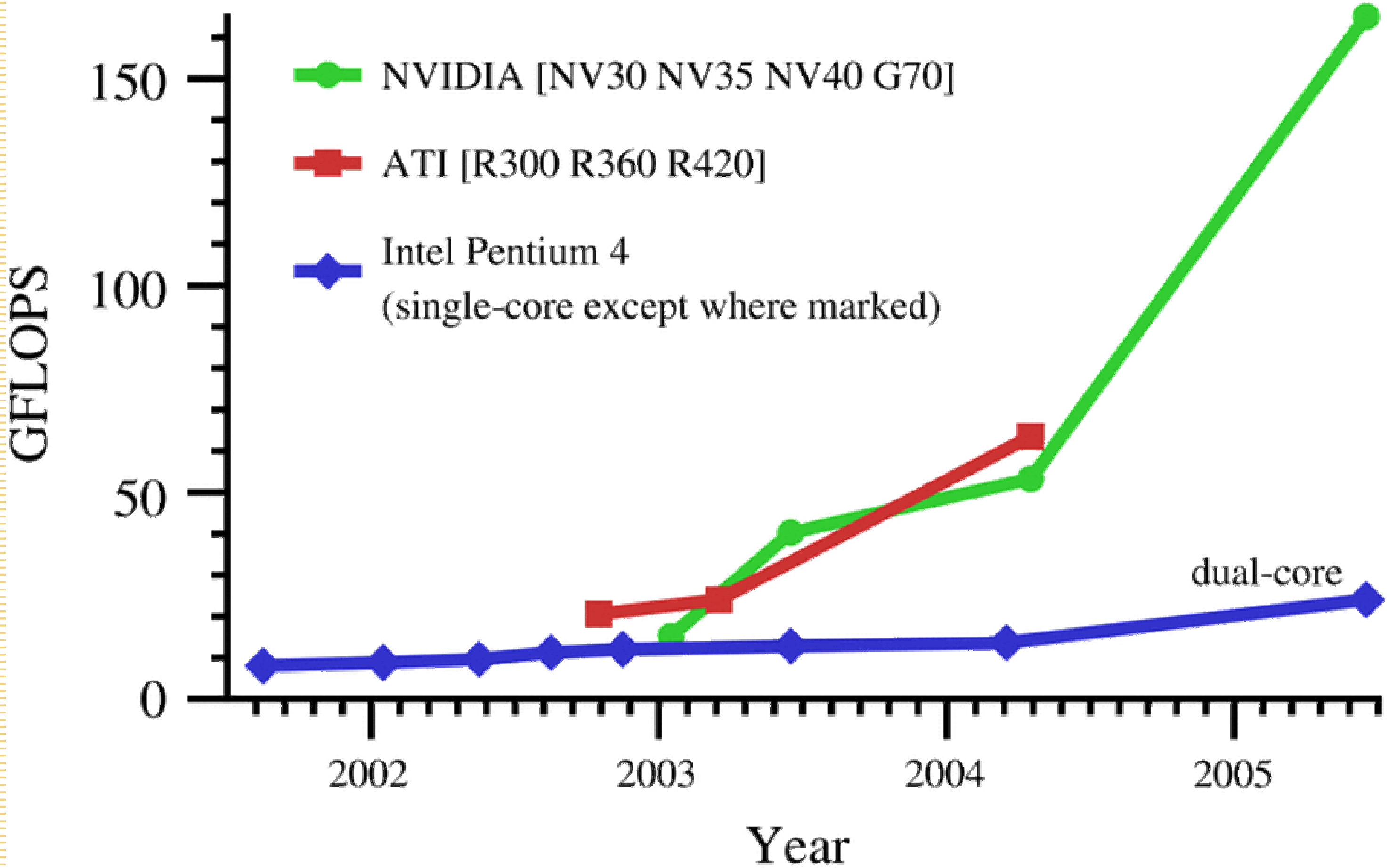
Graphics: The Second Revolution

- High-resolution Displays (LCDs, HDTVs)
- Consumer-driven demand for Games
- Emergence of Graphics Processing Units (GPUs)



Graph Courtesy: John Poulton, UNC

Current Trends



Courtesy Ian Buck, John Owens

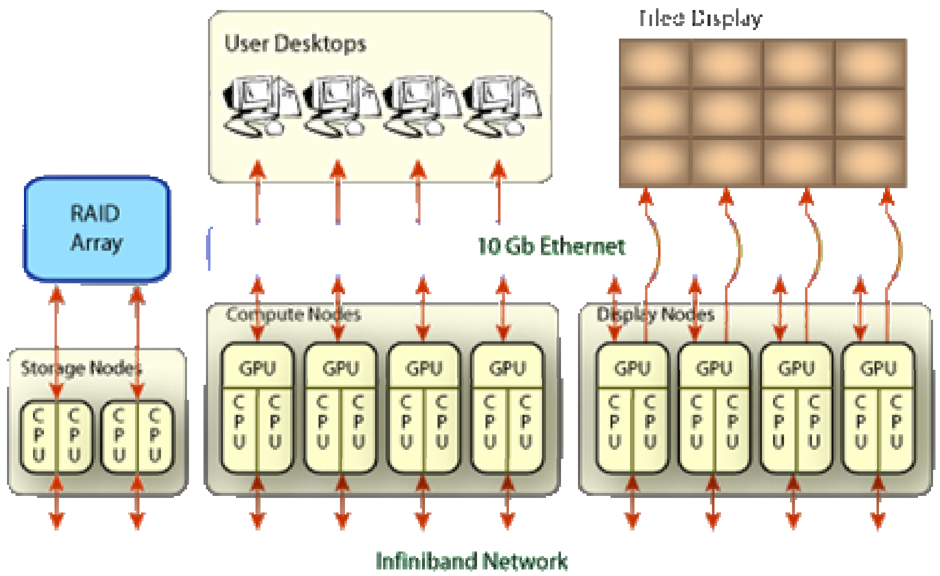
Do GPUs have a Future?

- The GPUs are cheap and are already commercially successful on a massive scale
  - PCs, game consoles, and now cell phones (100M units/year)
- Programming model
  - Restrictive, but high-level support is rapidly emerging
  - Optimizing compilers do great because of constraints
- GPUs are following a disruptive innovation path
  - Instead of appealing to the high-end market scientific computing market, they first appealed to the low-end teenagers market and are now moving up-market

GPU-CPU Cluster Overview

The Team:

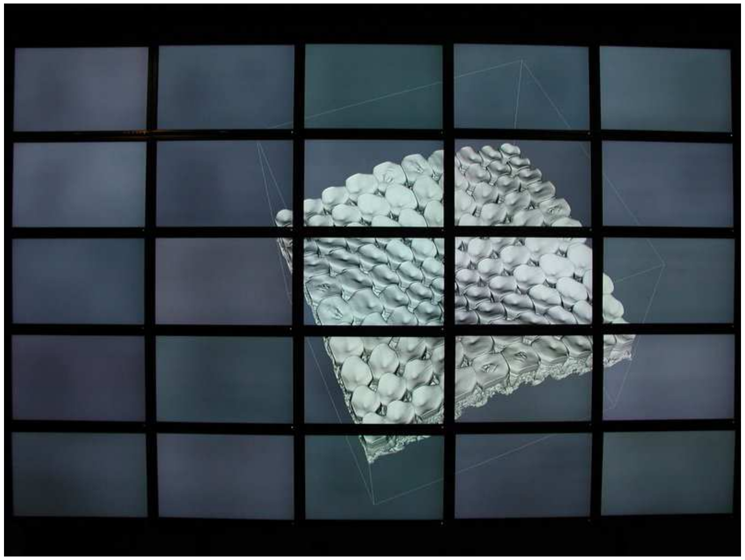
Shuvra Bhattacharyya
Rama Chellappa
Michael Cummings
Larry Davis
Leila DeFloriani
Ramani Duraiswami
Howard Elman
Francois Guimbretiere
David Jacobs
Joseph JaJa
Fritz McCall
David Mount
Dianne O'Leary
Hanan Samet
Alan Sussman
Amitabh Varshney



Summary:

Build upon the synergies in coupling GPUs (Graphics Processing Units), CPUs, displays, and storage to address a variety of computational and scientific problems

UMD Tiled Display Wall



Motivations for Streaming

- When input and/or output data is real time:
  - video, audio, databases, network data, 3D graphics
- When data is expensive to obtain or send:
  - the memory wall problem
- When the problem is compute intensive:
  - Traditional architectures favor memory-intensive apps
  - One or two CPUs, a few GB of RAM, 100’s GB of disk
  - A large fraction of on-chip CPU area is devoted to caches and their management
  - Examples: graphics, scientific computing, computational biology, multimedia processing, real-time computer vision, machine learning, real-time planning, …

Conventional vs Streaming Computers

von Neumann Computers:

- Designed to do *many* operations on each datum
- Hence data stays (mostly) still, while instructions flow past
- Substantial set of different operations, but each has fixed function

Data Streaming Computers:

- Designed to do *same* operation on *many* data
- So operation stays still (set up), and the data flows past
- Want very powerful vector operations, so as to flow the data few times
- A whole different way of thinking and programming

GPUs and Data-Stream Computers

Similarities:

- Have some fixed operations
  - vertex transformation, lighting, rasterization,…
- Data (vertex/fragment) flows through processors
- Instructions use streaming table lookup (textures)
- Streaming to-memory operations
  - Copy to texture, render to texture, Z-buffering

Differences:

- Read/write access only for local memory, but not the entire on-chip memory
- No producer-consumer locality or working set

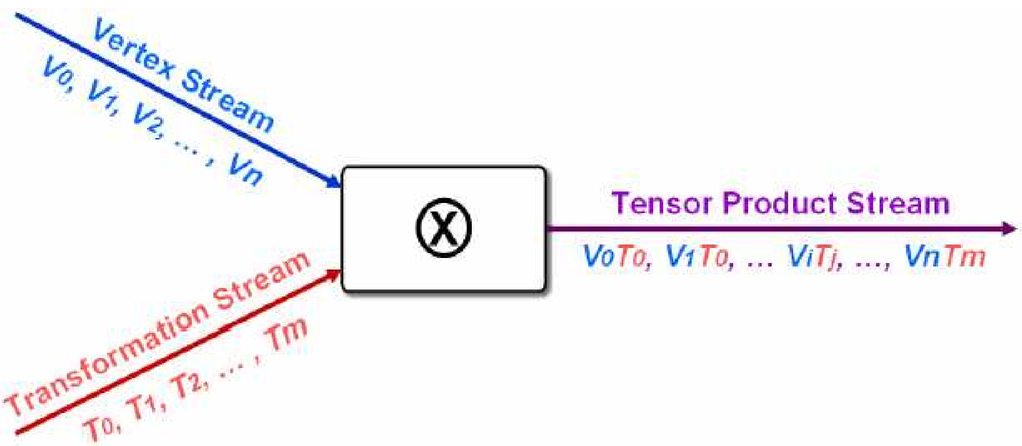
Stream Programming for GPUs

- Stream programming abstraction for GPUs
  - Streams of records: graphics primitives
  - Kernels: graphics operations (*Vertex shader* and *Pixel shader)*
  - Enabled a variety of applications in scientific computing, machine learning, and computer vision
- Arithmetic intensity
  - the compute to bandwidth ratio
  - the size of a floating-point unit has decreased
  - arithmetic is cheap and bandwidth is the critical problem
  - maximizing the returns from the available bandwidth
- Insight
  - two interacting streams are significantly more powerful than a single stream.

Interacting Streams

- Our goal is to reduce the input geometry bandwidth by factoring an input vertices into two interacting streams of *vertices* and *transformations*

- They are combined on a GPU, resulting in an *output stream of vertices*

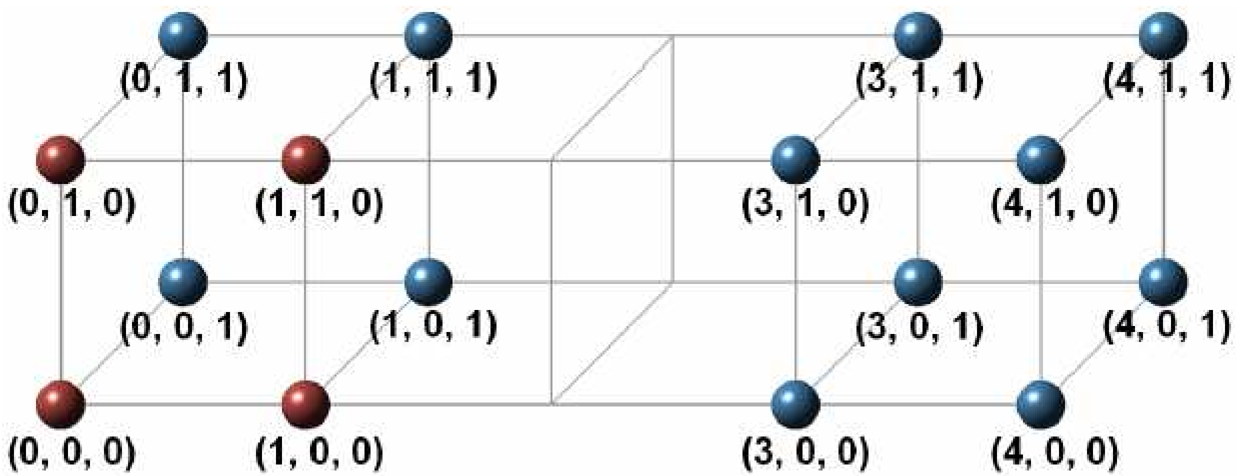


- Best case: factor *n* vertices into two streams of size $\sqrt{n}$ each

Interacting Streams

- An example in ideal case:

  4 vertices $\otimes$ 4 translations $\Rightarrow$ 16 vertices

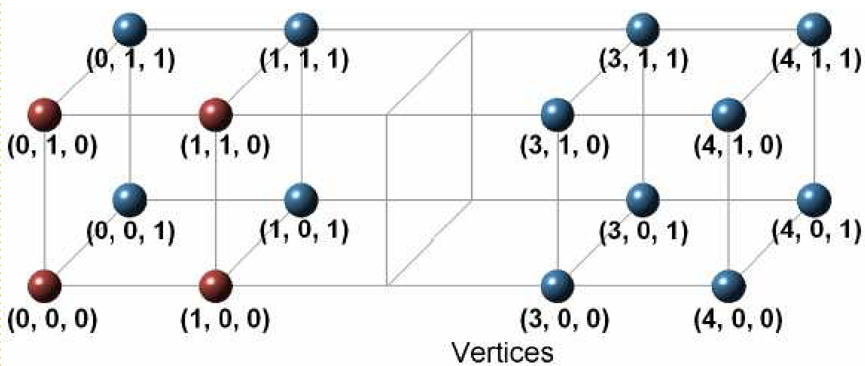


4 Source Vertices = (0, 0, 0), (1, 0, 0), (1, 1, 0), (0, 1, 0)

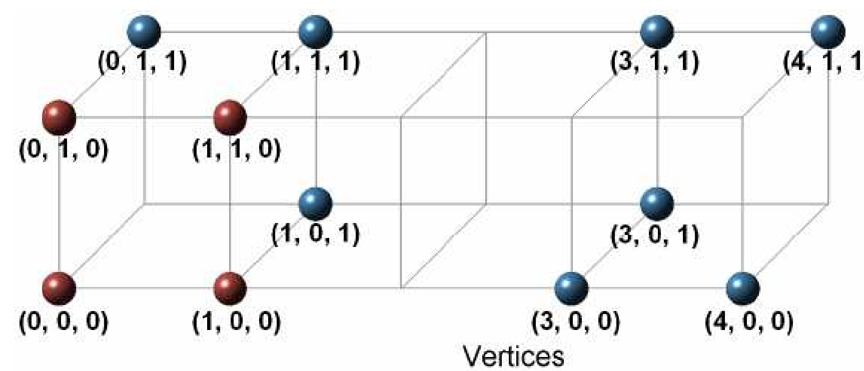
4 Translations = (0, 0, 0), (0, 0, 1), (3, 0, 0), (3, 0, 1)

Interacting Streams

- In practice, it is not always possible to find a perfect mapping between vertices and transformations
- We have generalized our interaction by tagging the elements of two streams.
- Interactions are represented by binary tables.
  - 1: interaction between a vertex and a translation
  - 0: no interaction



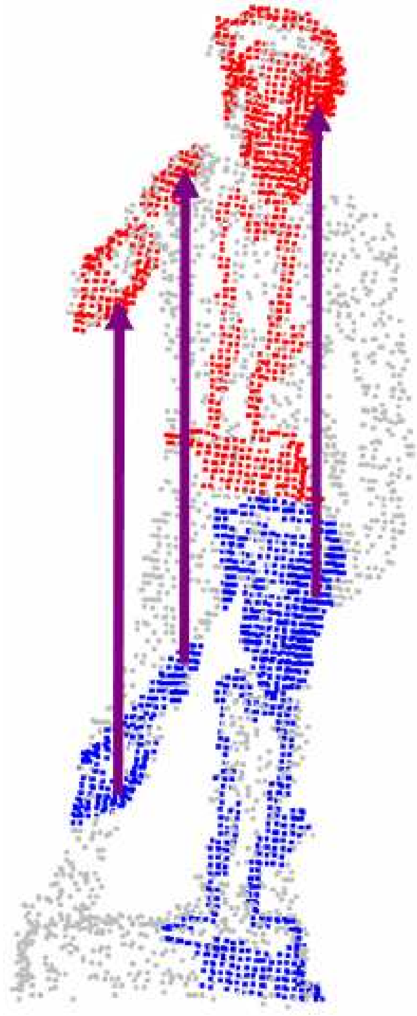
Translations	(0, 0, 0)	(1, 0, 0)	(1, 1, 0)	(0, 1, 0)
(0, 0, 0)	1	1	1	1
(0, 0, 1)	1	1	1	1
(3, 0, 0)	1	1	1	1
(3, 0, 1)	1	1	1	1



Translations	(0, 0, 0)	(1, 0, 0)	(1, 1, 0)	(0, 1, 0)
(0, 0, 0)	1	1	1	1
(0, 0, 1)	0	1	1	1
(3, 0, 0)	1	1	0	0
(3, 0, 1)	1	0	1	1

Transformation Palettes

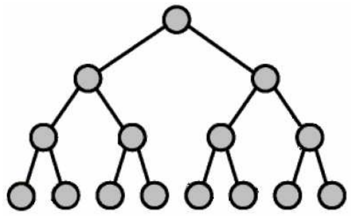
- Identifying the most common transformations amongst vertices
- Restrict ourselves to translations
- n vertices => n^2 translations possible => m (<< n^2) unique translations in practice if the vertices are quantized
- We want to identify the most common *m* (= 256) translations after quantizing points on a 128^3 grid
- Use FFT to find self-similar regions fast
- We use the results to build the *vertex-transformation pools*



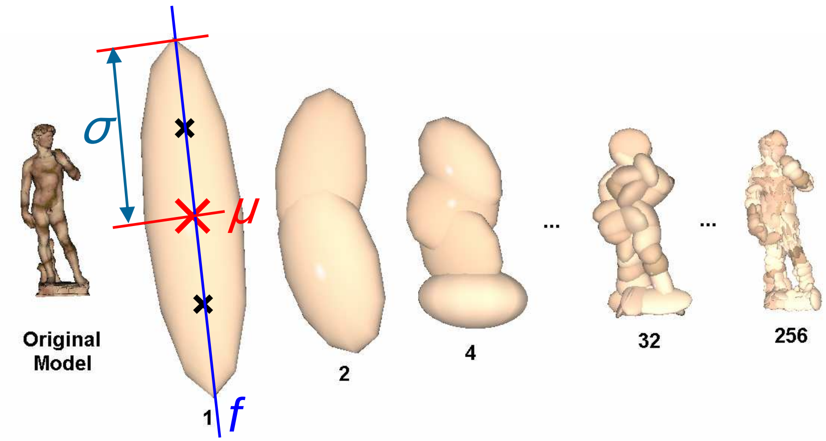
Kim, Lee, and Varshney, Graphical Models, 2006 (to appear)

View-dependent Rendering

- Render different regions of a scene at varying detail based on their perceptual significance



- First build a binary hierarchy over the input points by PCA-based partitioning
  - Mean(μ), an orthogonal frame (f), standard dev. (σ) of the data
  - k-means clustering (k = 2)



- For each node in the binary tree, we carry out three steps (which have been discussed) as a pre-process:
  - Identify the most common transformations appropriate for that node (transformation palettes)
  - Build vertex-transformation pools
  - Identify the transformation and vertex streams and store

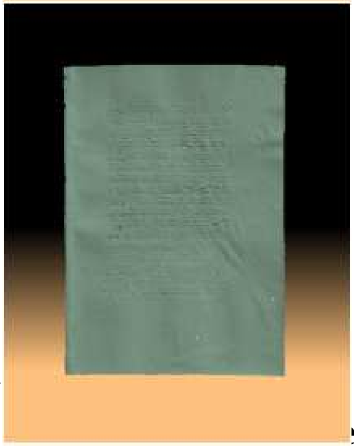
Results

- 200% to 500% improvement in the communication requirements to the GPU
- 17% to 32% improvement in frame rates
- As the gap between processing speeds and memory access times grows ever wider, the impact of our method should rise further

1.37M (81%) out of 1.69M

878K (74%) out of 1.17M

926K (89%) out of 1.02M



Research Applications

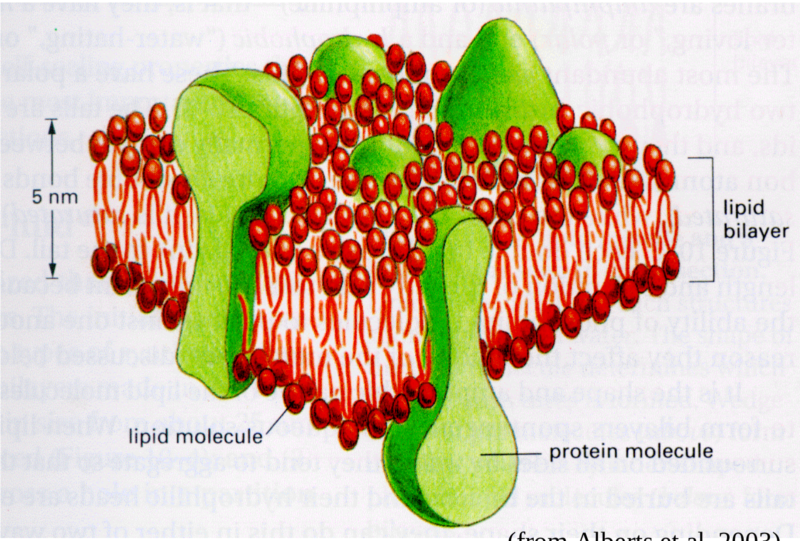
Interleaved Computation and Visualization

- *High-performance Computing*
  - Querying and Visualization of Large Scientific Datasets
- *Scientific Computing*
  - Solving Stochastic Differential Equations with CPU-GPU capabilities
- *Scientific Visualization*
  - Visualization-assisted Computational Steering for Protein Studies
- *Virtual 3D Audio*
  - Real-time Soundscape Rendering using CPU-GPU cluster
- *Computer Vision*
  - Modeling and Visualization of Humans and their Activities

Data Streaming Applications

Ion Channels

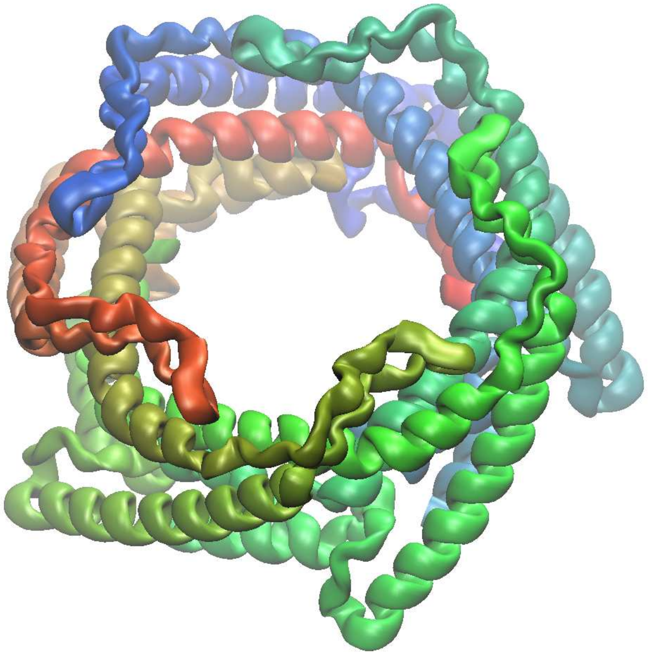
- Proteins that regulate the flow of ions into and out of the cell membranes.



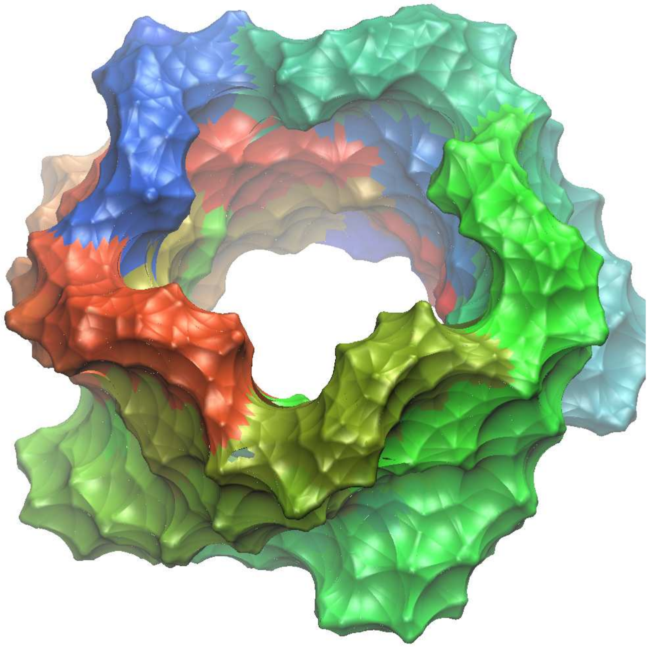
(from Alberts et al. 2003)

- Highly specific cell filters
- Transitions are very fast
- Ion-channel-driven processes responsible for Alzheimer's Parkinson’s, epilepsy, schizophrenia, stroke, and cystic fibrosis.
- Nearly a third of the top 100 pharmaceutical drugs target the ion-channels.

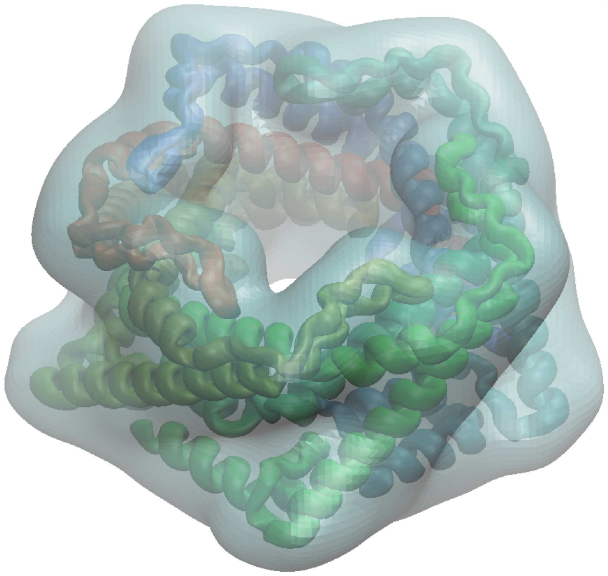
E.Coli Mechanosensitive Ion Channel



Ribbon Representation

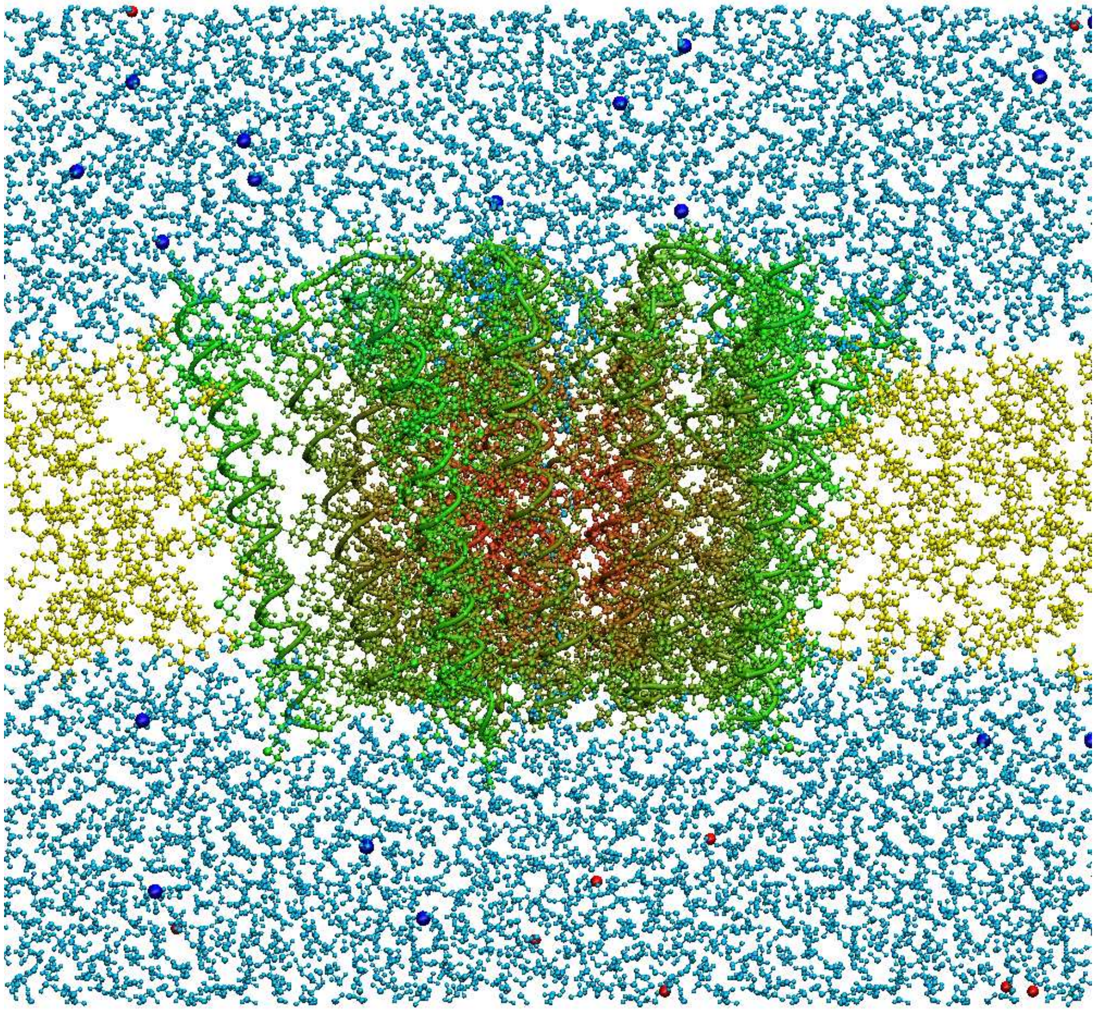


Solvent-Accessibility Representation



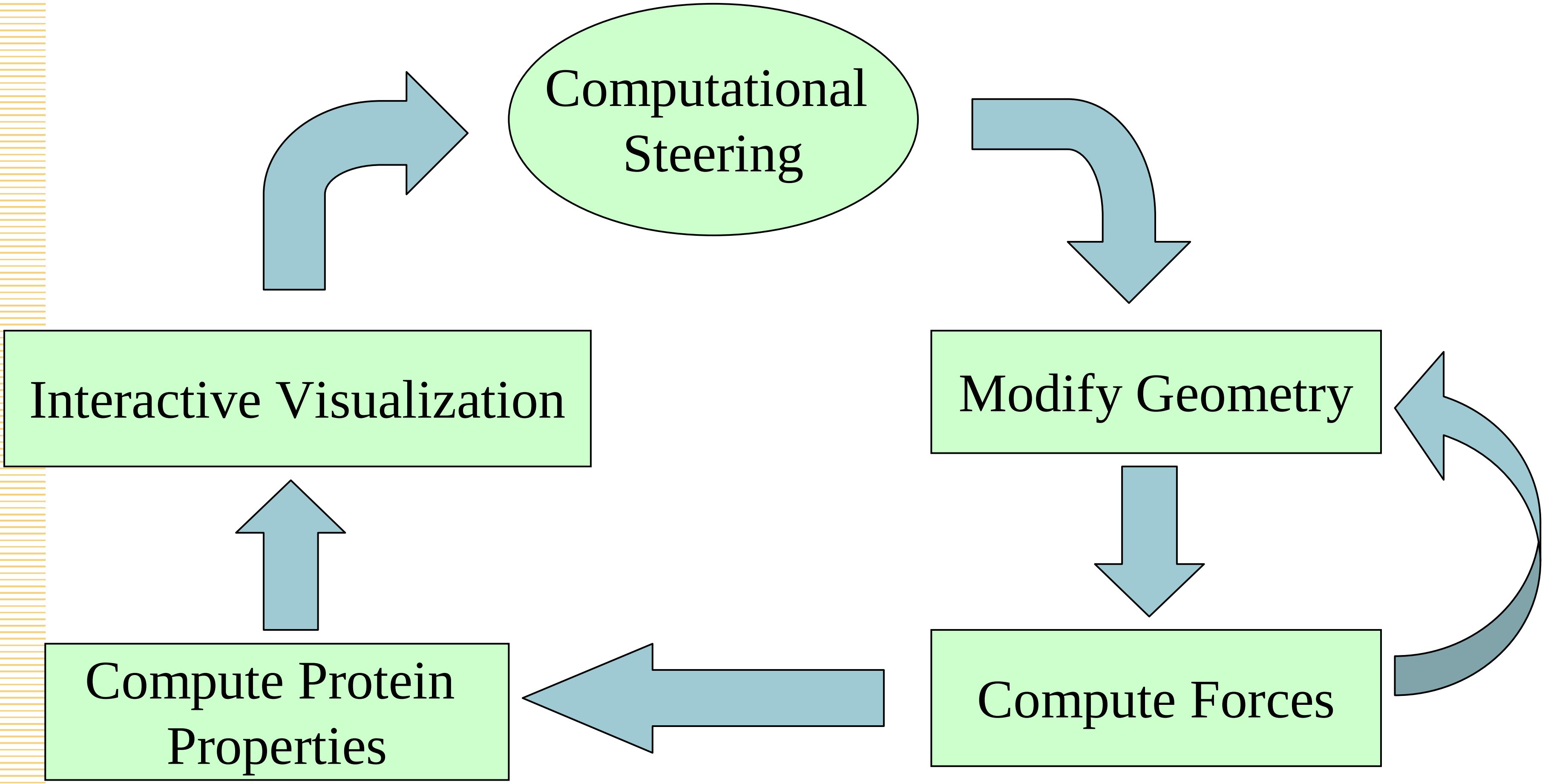
Uncertainty Envelope Representation

Membrane Ion Channels



Nicotinic Acetylcholine Receptor in Membrane mimetic Slab
(77K atoms)

Overview of Planned System



Computational Modeling of Proteins

- Bonded Properties
  - bond length, bond angle, dihedral angle
  - proton donor/acceptor distributions (H-bonds)

- Non-bonded Properties
  - Lennard-Jones potential (Van Der Waal's radius)
  - Electrostatic charge (Poisson-Boltzmann Equation)
  - Solvent interactions (Richards' smooth molecular surface)

SURF: A system to generate Solvent-Accessible Molecular Surfaces

- Analytically precise
- Parallelizable to one atom per compute node
- Nearly linear scalability in number of atoms
- Based on power diagrams (generalization of the Voronoi diagrams)
- Currently handles solvent-accessibility for several public-domain protein packages:
  - VMD System (UIUC)
  - Protein Viewer (IBM Life Sciences)
  - Sculpt (Elsevier Science)
  - Naval Research Labs
  - VeraChem/UMBI, …

Mapping on the CPU-GPU Cluster

- Electrostatics computed on GPUs using vertex programs
- Solvent-accessible surfaces computed on CPUs
- Surfaces texture-mapped with electrostatics
- Explore direct volume rendering on GPUs

Modeling Inter-atomic Forces

- Non-bonded properties take up most of the time

$$F_i(\mathbf{r_{ij}}) = \left(\underbrace{\frac{1}{4\pi\varepsilon_0} \frac{q_i q_j}{\varepsilon_r r_{ij}^2}}_{\text{Coulomb Interaction}} + \underbrace{12\frac{C_{12}}{r_{ij}^{12}} - 6\frac{C_6}{r_{ij}^6}}_{\text{Lennard Jones}} \right) \frac{\mathbf{r_{ij}}}{r_{ij}}$$

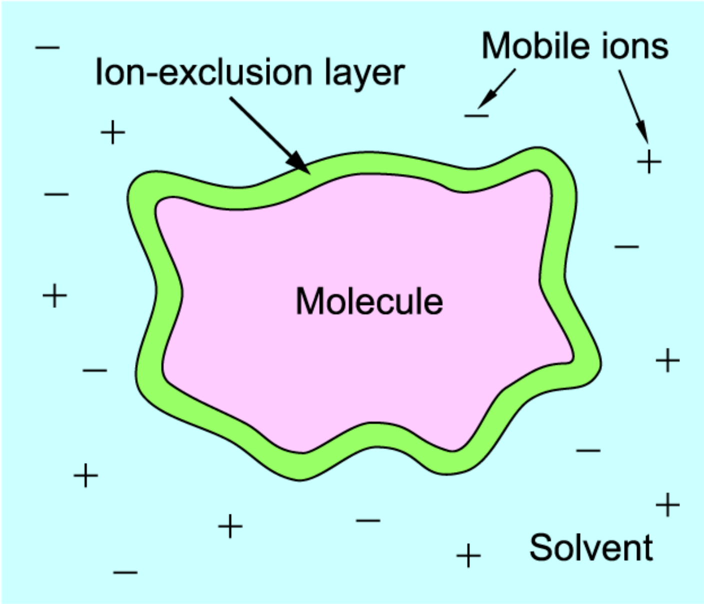
Modeling Electrostatics

- Quantum Mechanical Methods
  - Accurate
  - Require immense computational power
  - Currently possible only for small molecules

- Classical electrostatics
  - Model interactions between partial atomic charges
  - Depend on 3D structures of molecules, charge distributions, and environment (solvent)
  - Described by Poisson-Boltzmann equation (PBE)

Poisson-Boltzmann Equation (PBE)

$$\nabla\left[\varepsilon(\vec{r})\nabla\phi(\vec{r})\right]-\kappa'^2(\vec{r})\sinh(\phi(\vec{r}))+4\pi\rho(\vec{r})=0$$

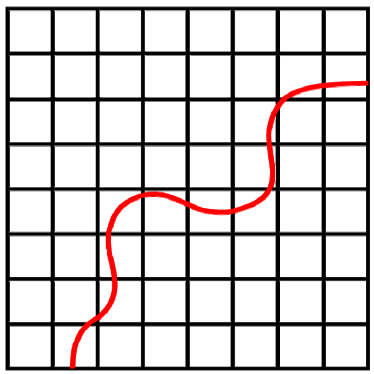


Dielectric constant

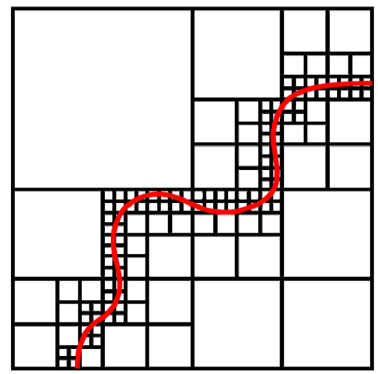
$$\varepsilon \approx \begin{cases} 2 & \text{inside molecule} \\ 80 & \text{inside solvent} \end{cases}$$

PBE parameters change largely across solvent-molecule interface

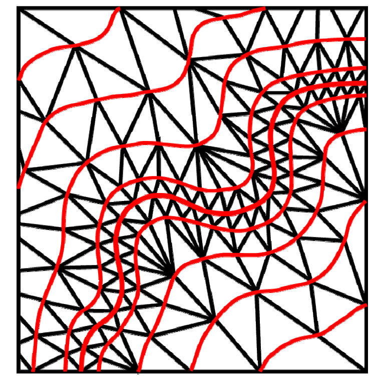
Molecular Electrostatics



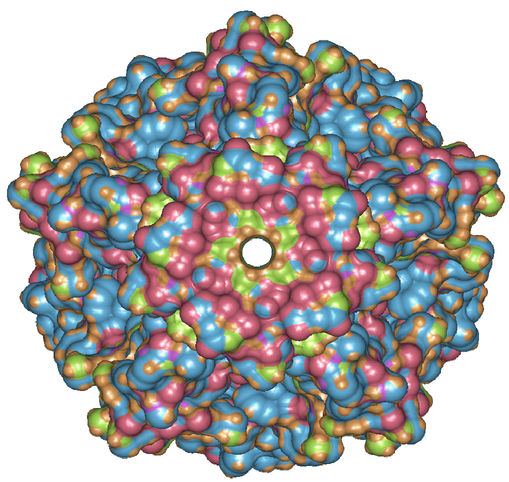
Regular grid

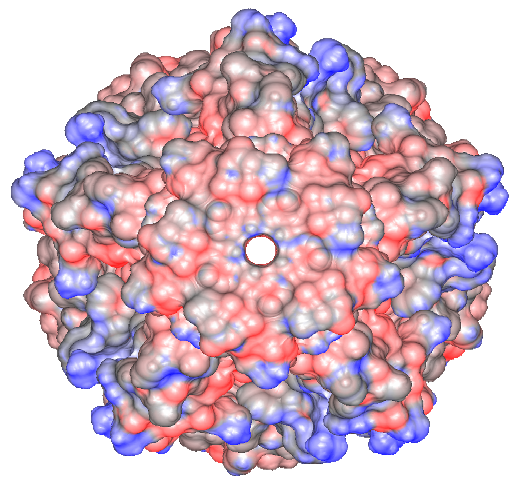


Adaptive Subdivision



Interface-focused

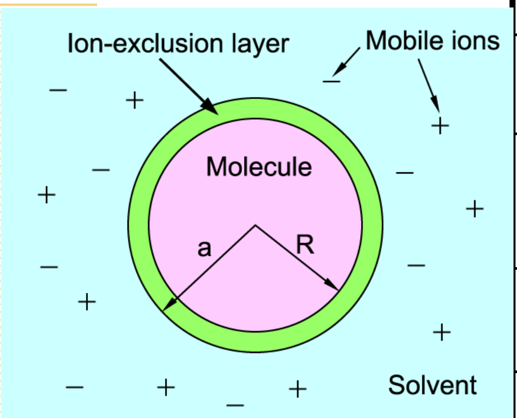




E. coli Mechanosensitive Channel

Molecular surface **Surface Electrostatics**

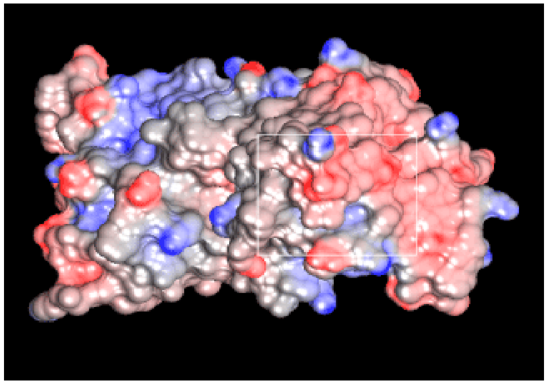
Comparisons with DelPhi

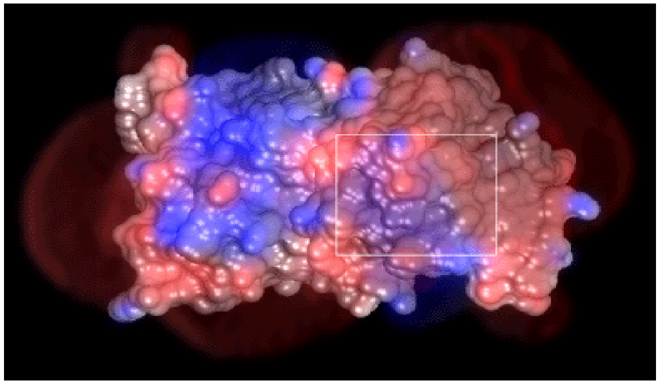


	DelPhi (from Columbia University)			Our Method
Grid size	67^3	133^3	199^3	NA
# of pts	300,763	2,352,637	7,880,599	26,987
PSNR	8.17	19.1	25.1	27.7
Avg Error	30.88%	17.91%	13.27%	15.98%
PBE time	0.31 sec	4.5 sec	20.1 sec	0.25 sec

X. Hao and A. Varshney, "Efficient Solution of Poisson–Boltzmann Equation for Electrostatics of Large Molecules", *High Performance Computing Symposium,* April 18 - 22, 2004, Arlington, VA

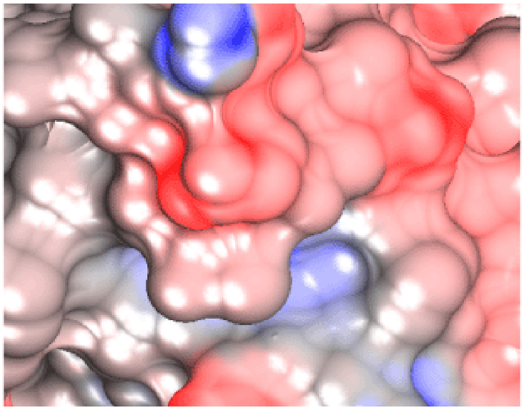
Volume Rendering of SOD Electrostatics

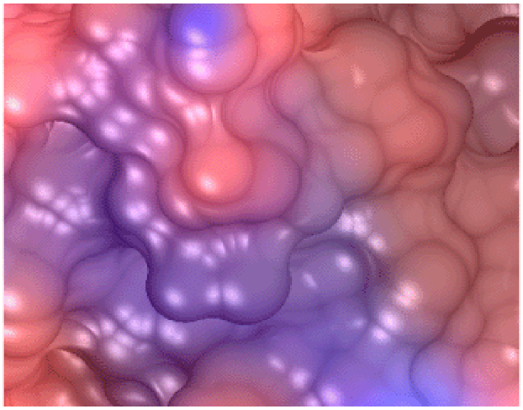




Red: negative potential

Blue: positive potential



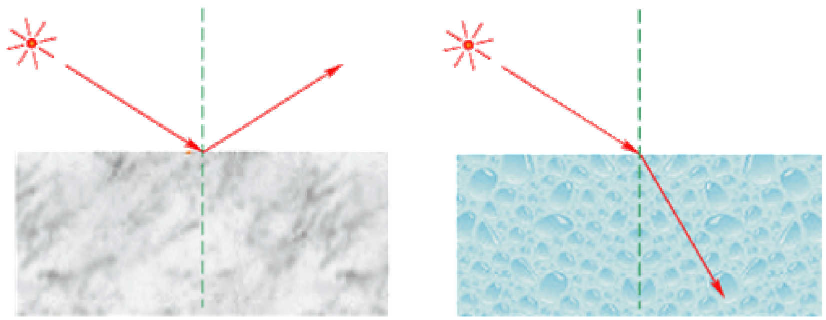


On-surface display

Our splatting (from viewer up to surface)

Interaction of Light and Matter: A Graphics View

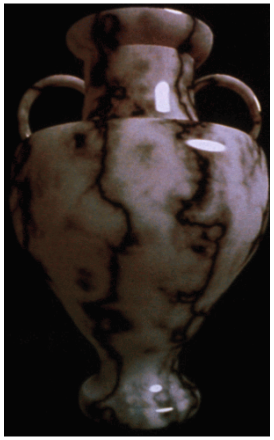
- Propagation
- Reflection and Refraction
  - Happen at the interface between two materials



- Can be described by both geometric (ray) optics and wave optics

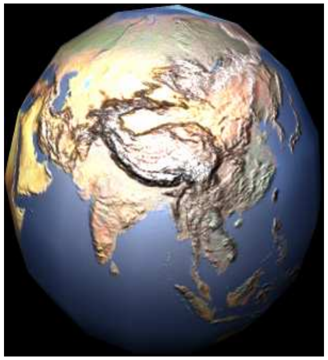
Rough Surfaces with Varying Properties

- Texture mapping (Catmull 74)
  - Greatly enhances realism
  - Hard to adjust to different lighting
  - 2D or 3D, real-time



Perlin *85*

- Bump mapping (Blinn *78*)
  - Used to perturb surface normal
  - No local shadowing or inter-reflections
  - 2D, real-time



NVIDIA *02*

Rough Surfaces with Varying Properties

- Bidirectional Texture Functions (BTF, Dana *et. al. 99*)
  - 6D function
  - Allow texture to adjust to different lighting
  - 1 second/frame
- Translucent materials
  - Absorption, scattering inter-related
  - Described by integral equations involve functions of 8D or higher
  - More complicated than BTF
  - 1250 minutes/frame (Monte-Carlo ray tracing, SIGGRAPH 2001)

Xin Tong et al. 02

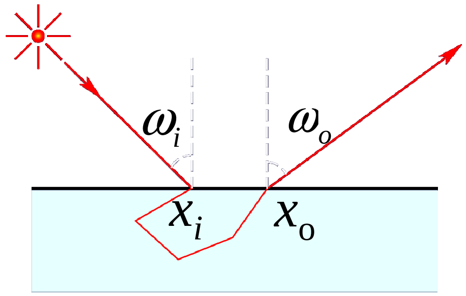
Jensen and Buhler *02*

Subsurface Scattering in Real World

BSSRDF Model

- Relates the illumination of a surface point with light distribution at other surface points (S is a 8D function):

$$dL_o(x_o,\vec{\omega}_o) = S(x_i,\vec{\omega}_i;x_o,\vec{\omega}_o)\,d\Phi_i(x_i,\vec{\omega}_i)$$



- Total outgoing radiance is an integral over all the incoming directions and the area

- Takes 1250 minutes to generate one image with Monte-Carlo ray-tracing (Jensen *et al.*, SIGGRAPH 2001)

Reduce the Dimension of the Integral

$$L_o(x_o,\vec{\omega}_o,\vec{\omega}_i) = \left\{\int_A \frac{1}{\pi} F_t(\eta,\vec{\omega}_i) R_d L_i(x_i,\vec{\omega}_i)(\vec{n}_i \cdot \vec{\omega}_i) dA\right\}$$

$$\bullet F_t(\eta,\vec{\omega}_o)$$

$$\equiv q(\eta, x_o, \vec{\omega}_i) \bullet F_t(\eta, \vec{\omega}_o)$$

Now *q* is just a 4D function, can be pre-computed

Discrete mesh geometry, integral $\Rightarrow$ summation

Compression of Scattering Integrals

- Original vertex data (per vertex)
  - Location and normal (both 3D vectors)
  - $(3 + 3) \times 4 = 24$ bytes with floating representation
- Pre-computed integrals requires *200* bytes/vertex
  - $200 \times 1M = 200M$ bytes for object with *1M* vertices
- Compression is necessary
  - Diffuse nature of scattering, and integral is defined in directional space $\Rightarrow$ Spherical Harmonic compressions

Results of SH Compression

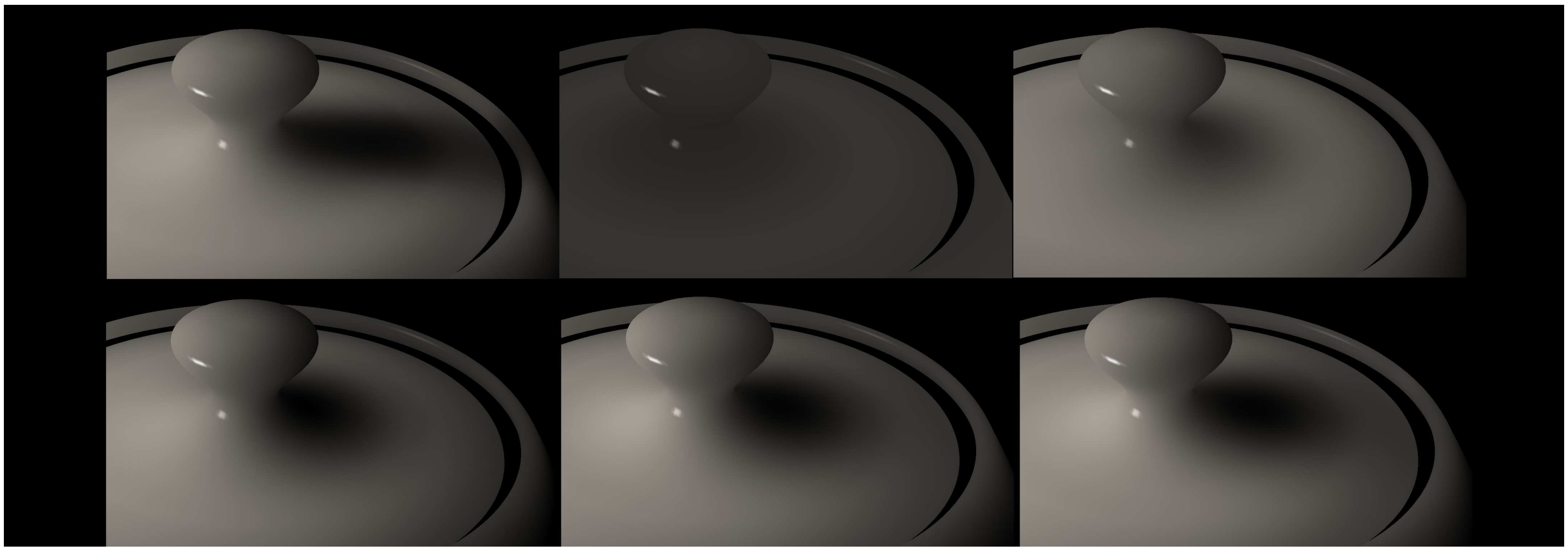
Utah Teapot with 150K vertices

Original (200 bytes/vertex)   1 SH (2 bytes/vertex)   4 SH (8 bytes/vertex)

9 SH (18 bytes/vertex)   16 SH (32 bytes/vertex)   25 SH (50 bytes/vertex)

Close-up Comparison

Original 1 SH 4 SH



9 SH 16 SH 25 SH

Reference Points with 9 SH Functions

Original
(200 bytes/vertex)

390 Ref pts
(13 bytes/vertex)

614 Ref pts
(14 bytes/vertex)

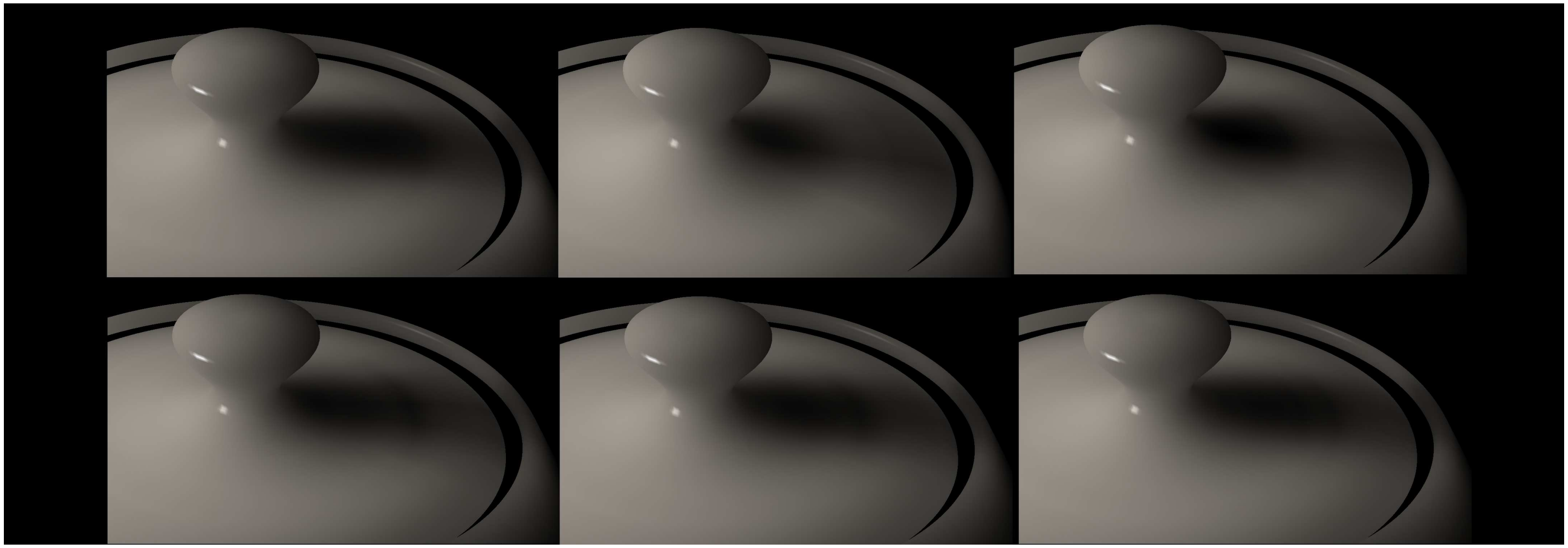
1.2K Ref pts
(15 bytes/vertex)

2.5K Ref pts
(17 bytes/vertex)

5K Ref pts
(20 bytes/vertex)

Close-up of Reference Points with 9 SH

Original　　390 Ref pts　　614 Ref pts



1.2K Ref pts　　2.5K Ref pts　　5K Ref pts

Results: Venus

No Scattering (62.5 fps)   With Scattering (27.3 fps) : 42K vertices

X. Hao and A. Varshney, "Real-Time Rendering of Translucent Meshes", *ACM Transactions on Graphics*, 23(2), April 2004

Current Trends

1. Data complexity is rising
   Anecdotal evidence suggests Super-Moore's Law increases
2. Processing capabilities are increasing
   Parallelism, Streaming, Interleaved ALUs and memory
3. Rendering Realism is improving

We are rapidly making progress towards handling very large models with impressive detail and realistic lighting.

If these trends continue we will soon reach …

Here …

And we will be able to fly through this at 60 Hz *with* collision detection

The Problem with Photo Realism

- Photorealism does not necessarily improve comprehension

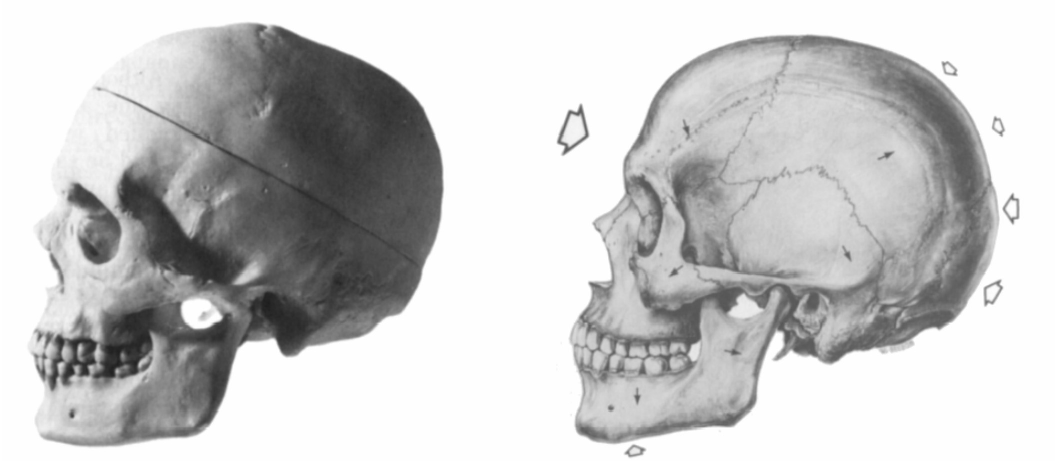


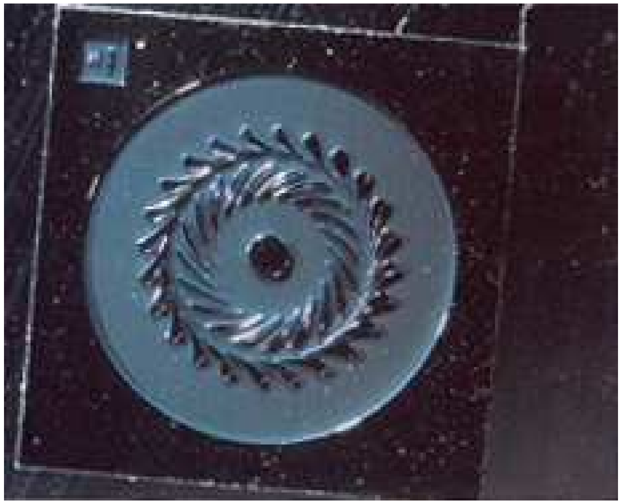
Image from The Guild Handbook of Scientific Illustration by Hodges, 1989

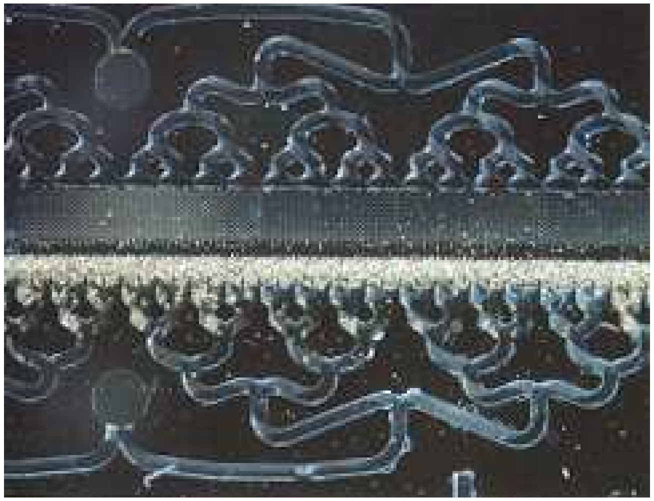
- What does one mean by *photorealistic* visualization anyway?

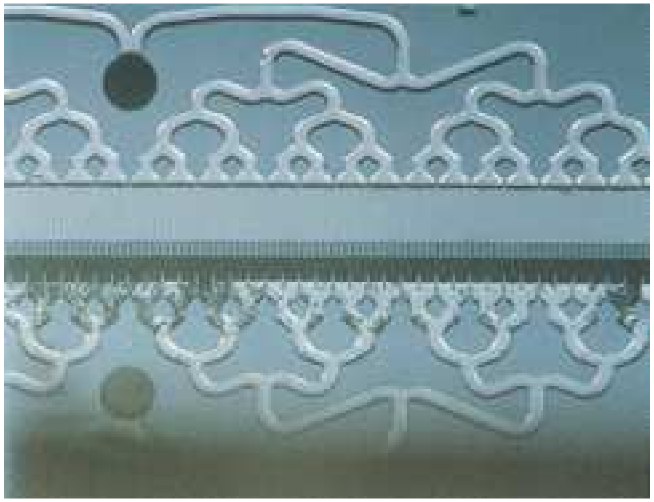
Directing Visual Attention by Light

Joseph's Bloody Coat Brought to Jacob, Velasquez 1630

Lighting in Photography







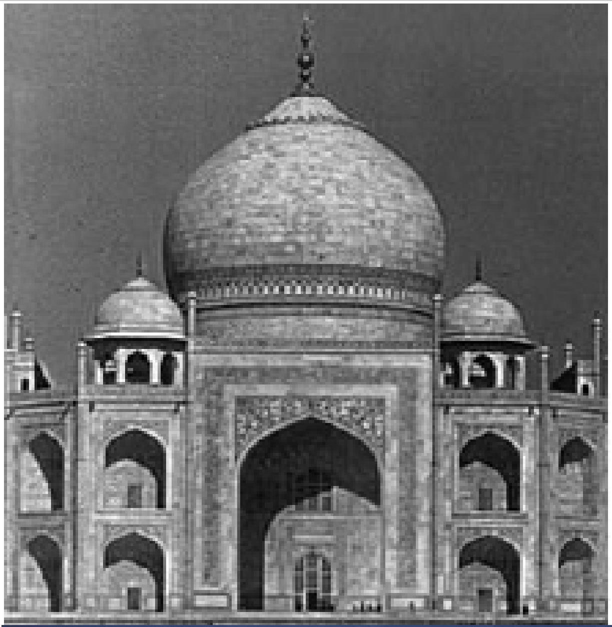
from Envisioning Science by Felice Frankel

Cinematic Lighting Design

- Direct viewer's attention
- Establish a mood and an atmosphere
- Create a sense of depth and realism

High key, bright lights vs
low key, high contrast

Some Eye Candy



What do you think of these images?

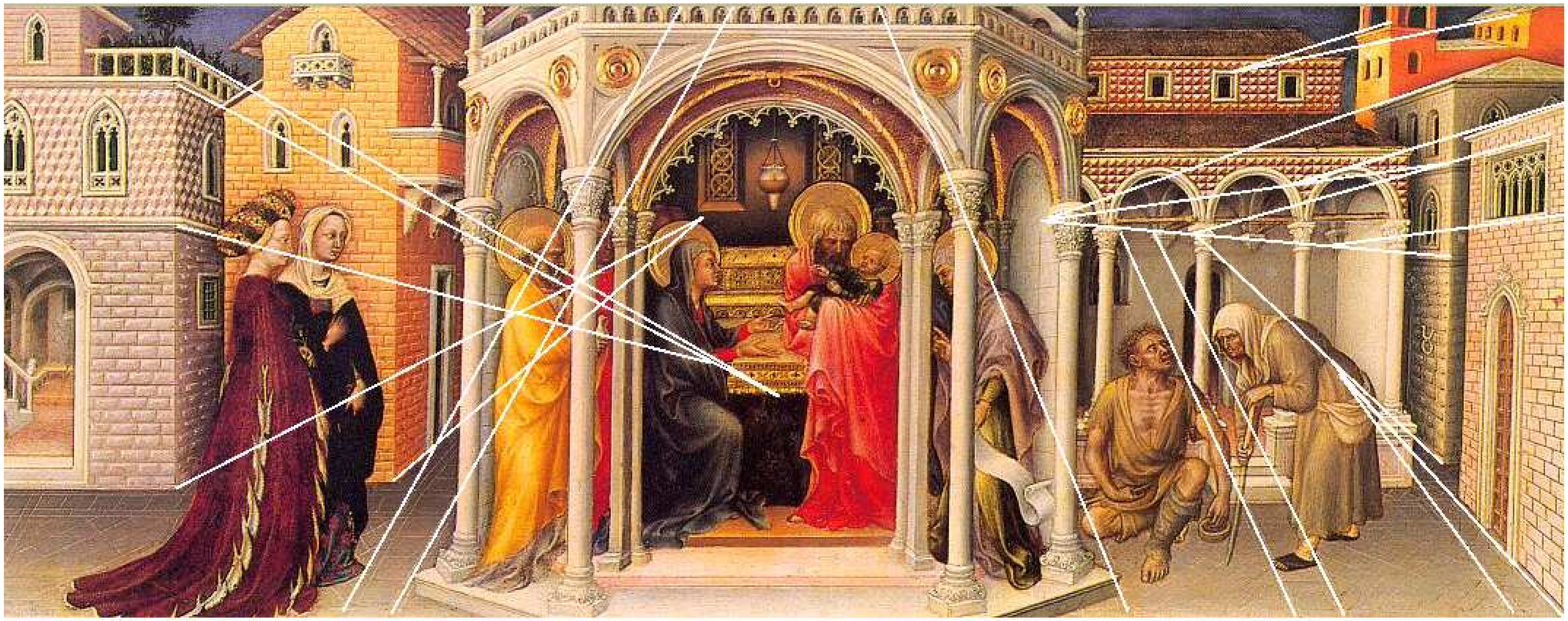
Visual Continuity & Consistency

- Shots often composited from multiple takes
- Lighting is painstakingly made consistent
- Consistent lighting considered important for visual continuity and storytelling

Jurassic Park, © Universal Studios

Is Consistent Lighting Necessary?

- Nature has one dominant light source
- Evolution might have endowed us with an ability to discern inconsistency in illumination
- Just as it has inconsistency in perspective

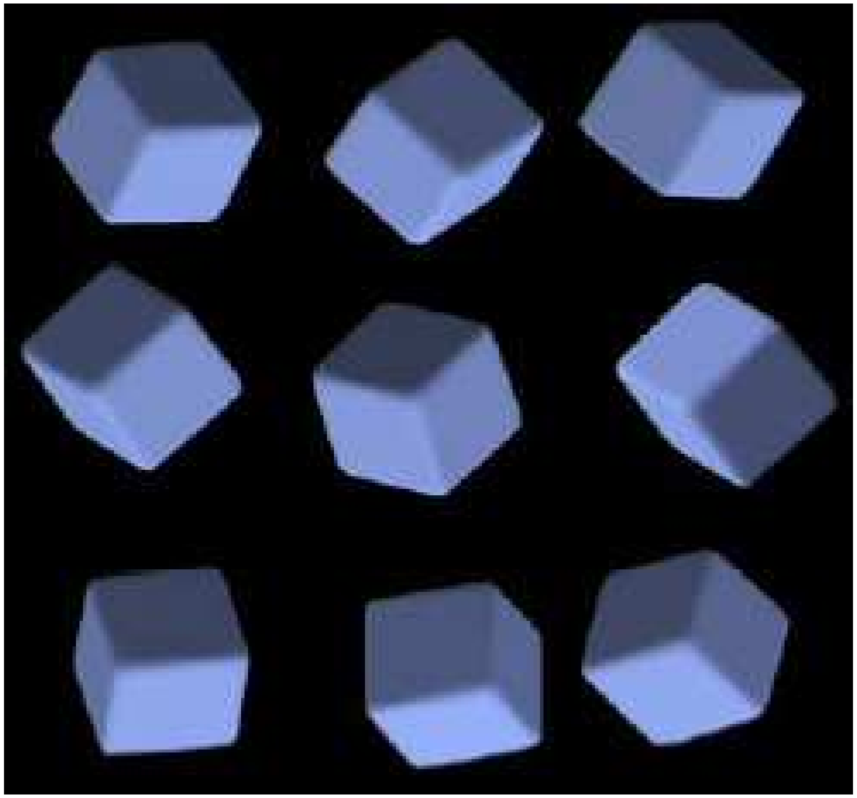


The Presentation in the Temple by Gentile da Fabriano (1423)

Illumination Inconsistencies

Recent research suggests that illumination consistency is *not* resolved at the low-level human vision

Find the cube lit inconsistently with respect to others:

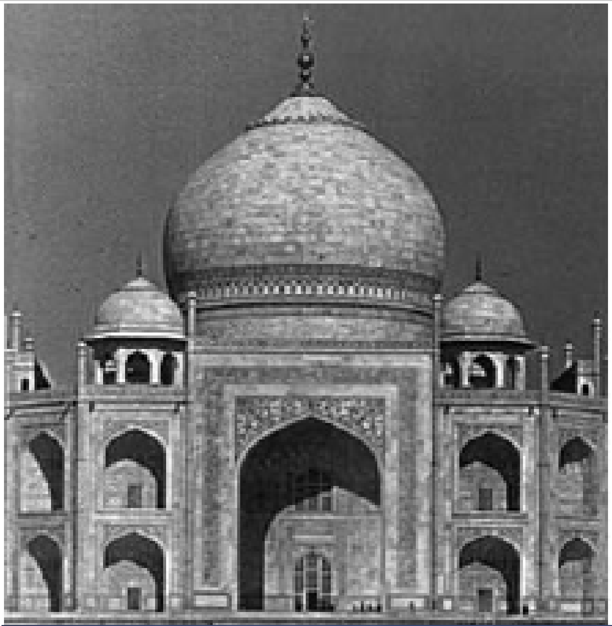


On average, users take 8 seconds to answer and are then wrong 30% of the time

Illumination inconsistencies are not perceptually salient

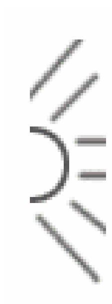
Ostrovsky, Sinha, Cavanagh, *Perception 2006*

Illumination Perception



Ostrovsky, Sinha, Cavanagh, *Perception 2006*

Discrepant Lighting in Art



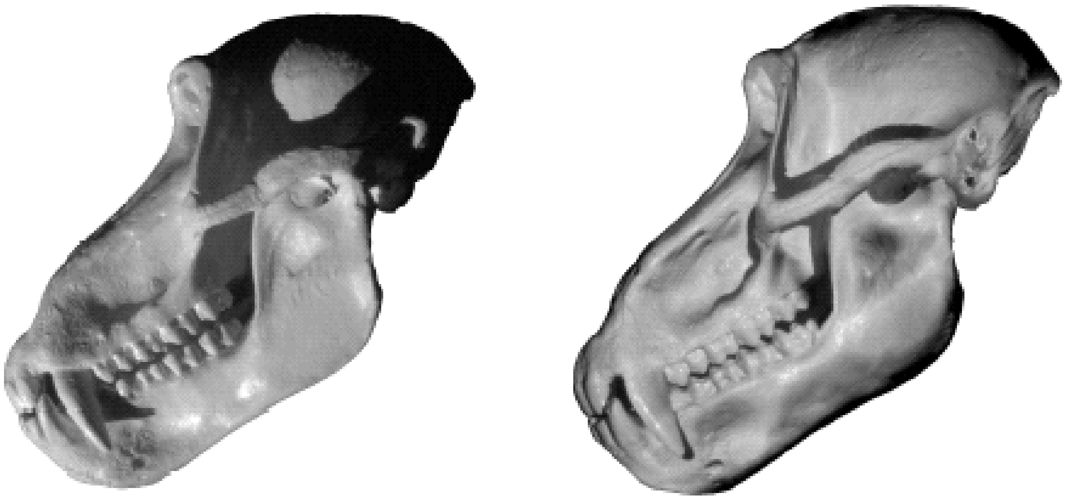
George Washington Crossing the Delaware
by Emanuel Gottlieb Leutze

Discrepant Lighting in Art

The Music Lesson by Jan Vermeer

Discrepant Lighting

- Scientific visualization need not strive for photorealism
- Discrepant lighting can yield compelling results

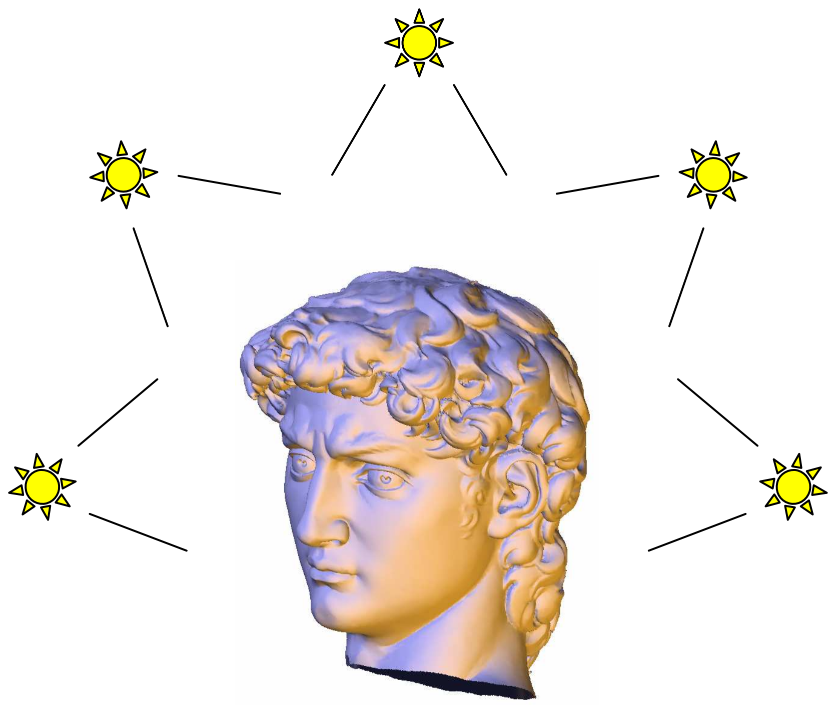


Consistent Discrepant

Akers, Losasso, Klingner, Agrawala, Rick, Hanrahan, *Visualization 2003*

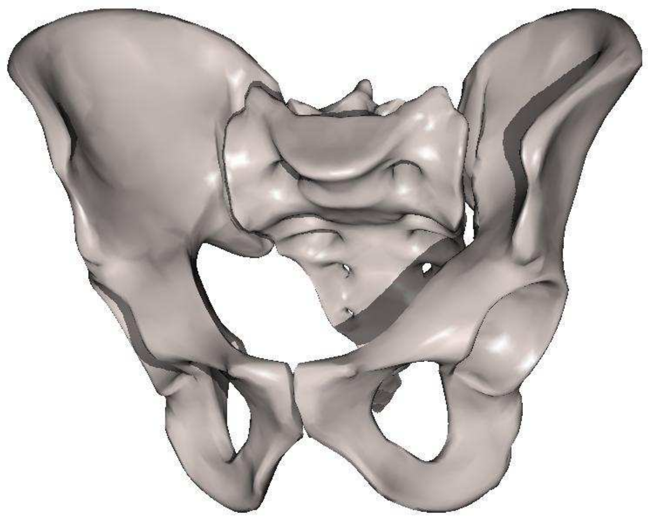
Light Collages: Basic Idea

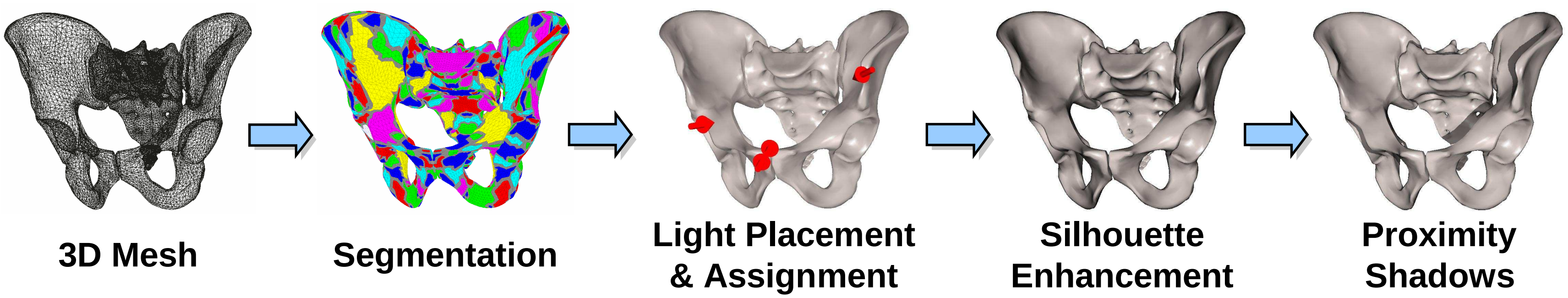
Allow local lighting parameters to be defined independently at local regions



Light Collages Overview

- Segmentation
- Light Placement and Assignment to patches
- Silhouette Enhancement
- Proximity Shadows

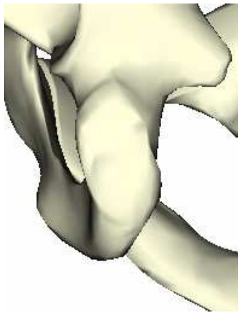




Lee, Hao, Varshney, IEEE Transactions on Visualization and Graphics 2006

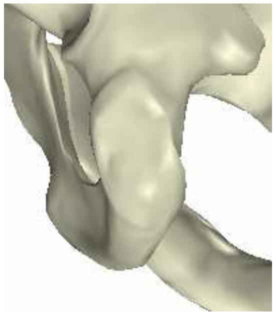
Light Placement

Specular light



on less curved surfaces

⇒ over-exposure hides details



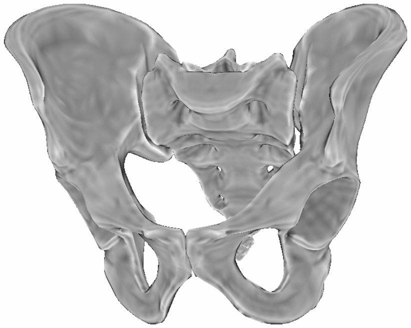
on highly curved surfaces

⇒ useful for illustrating details

Diffuse Light

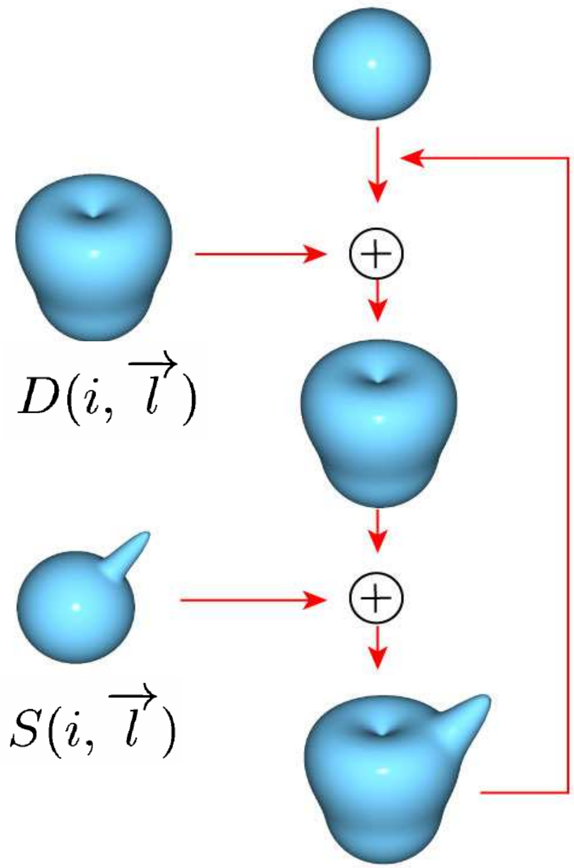
Curvature is informative

(Gumhold *Visualization 02*)



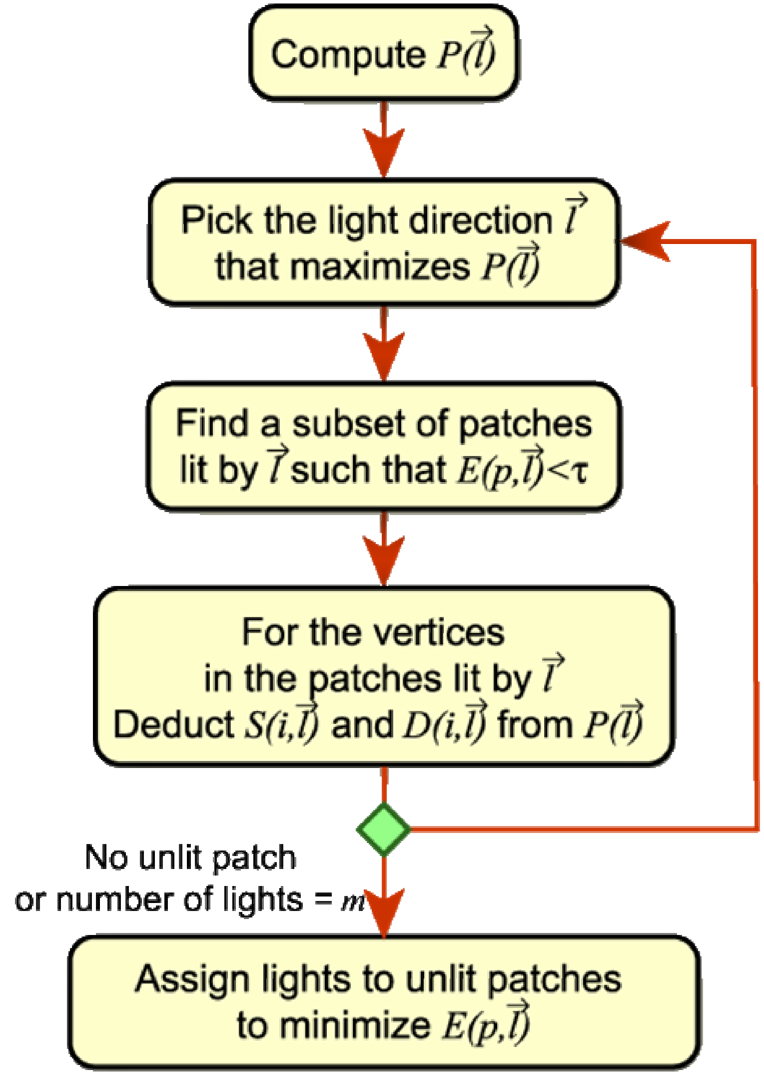
Curvature intensity c_i at a vertex i:
normalized mean curvature

Light Placement



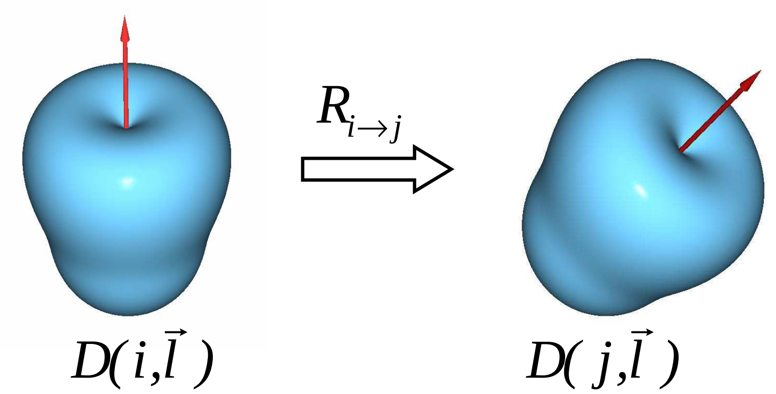
Light Placement Function:

$$P(\vec{l}) = \sum_i (S(i, \vec{l}) + D(i, \vec{l}))$$

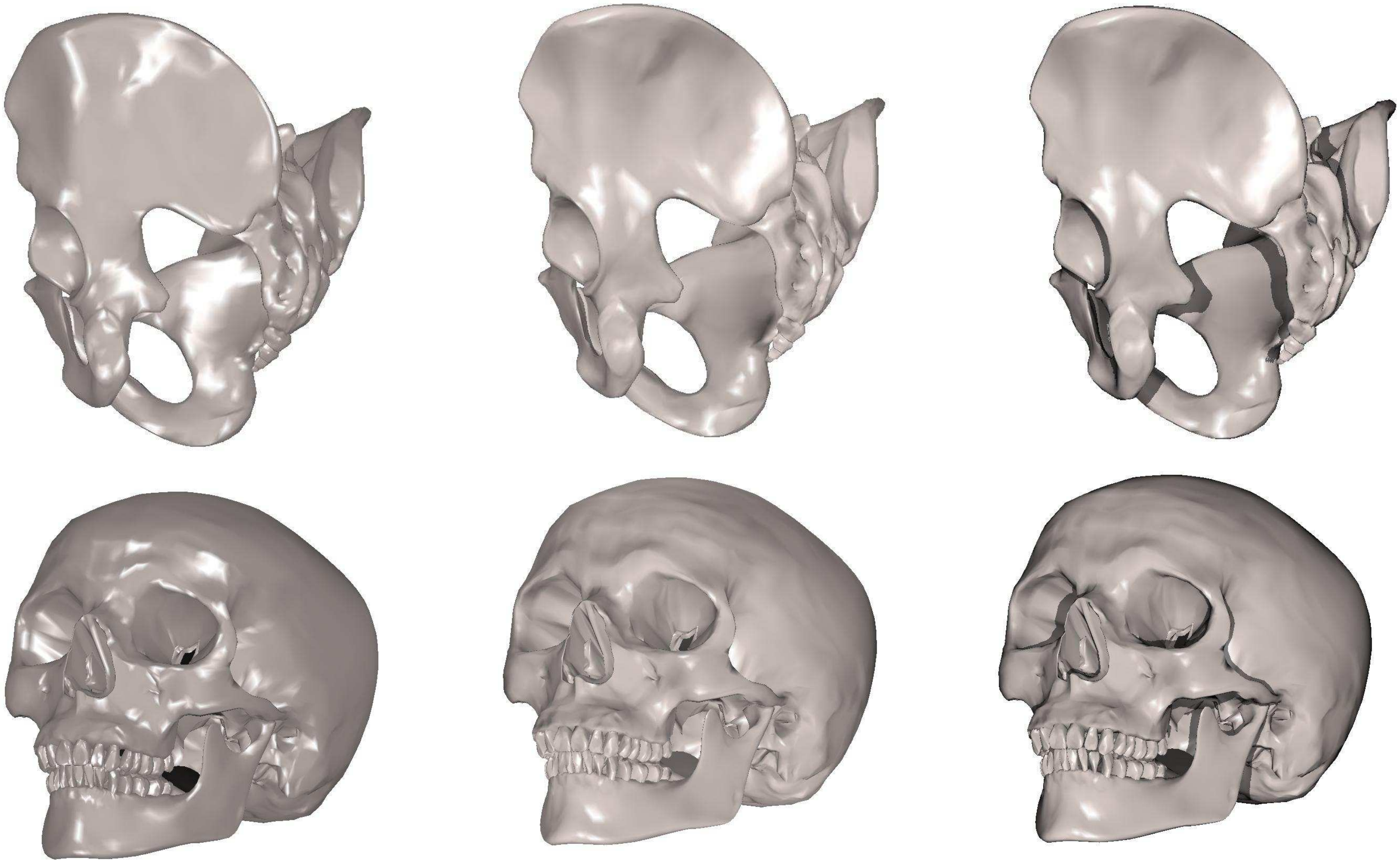


Spherical Harmonic Speedup

- Specular & diffuse weight functions
  - The rotation-invariant shape depends on the curvature value
  - Pre-computed & encoded in spherical harmonic representation for different curvatures
- Light placement function
  - For each vertex, pre-computed weight function is rotated and added
- For 5 SH bands (considered sufficient) 20x speedup and memory reduced by 500x



Results – Pelvis & Skull

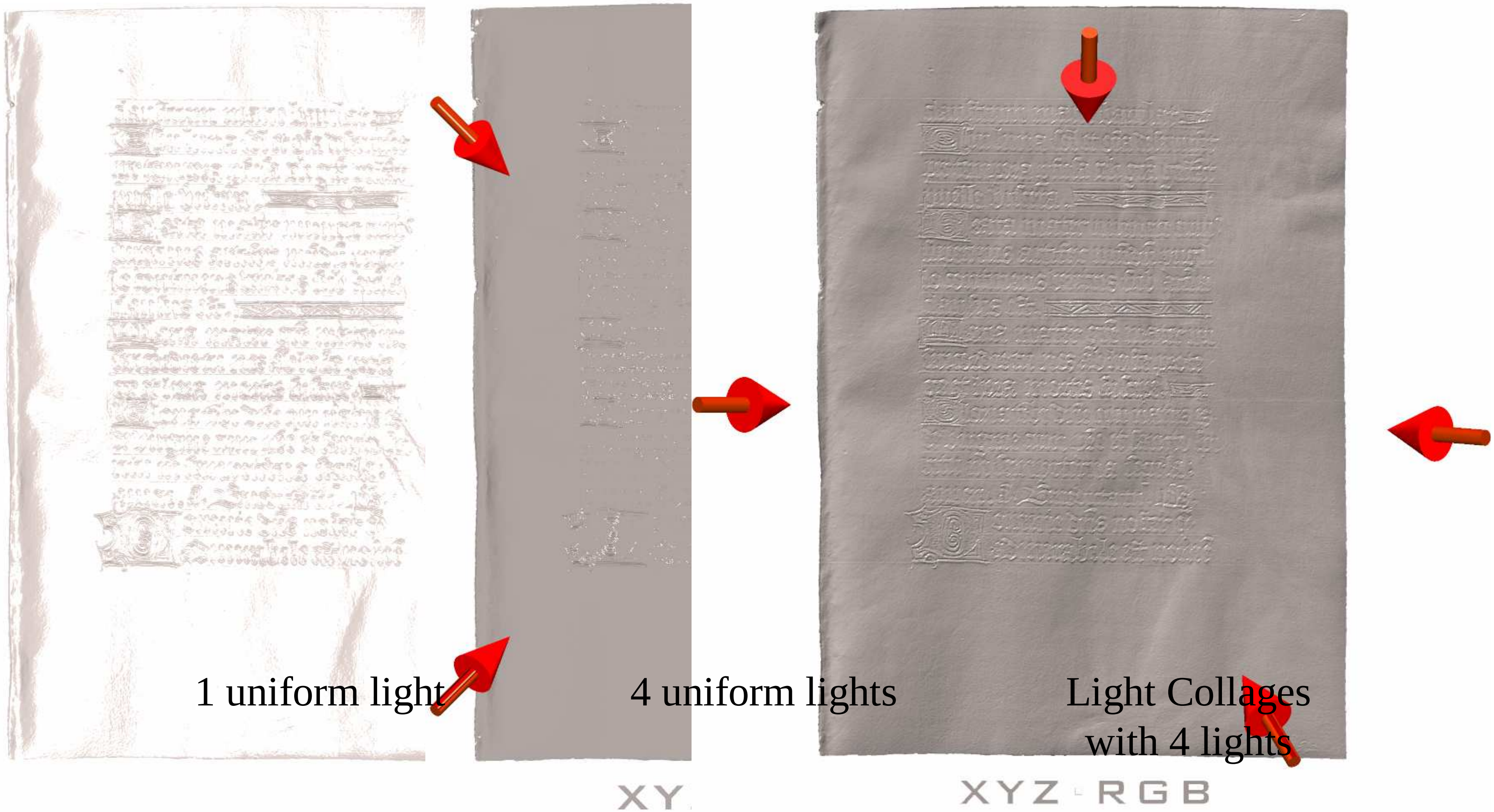


(a) Uniform 4 lights

(b) Light Collages: with 4 lights

(c) Light Collages: Silhouettes+ Shadows

Results - Manuscript

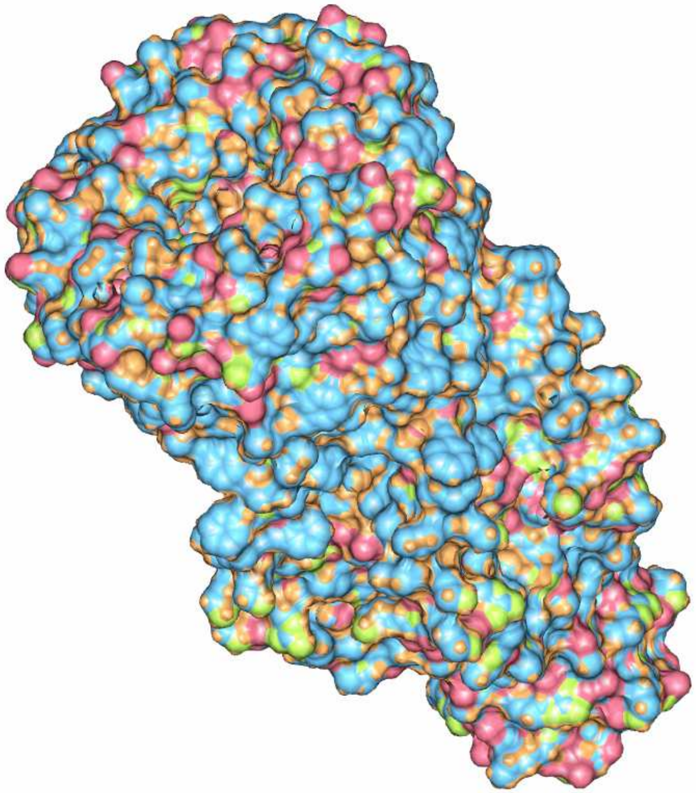


Manuscript courtesy of Paul Debevec, USC and XYZ RGB Inc.

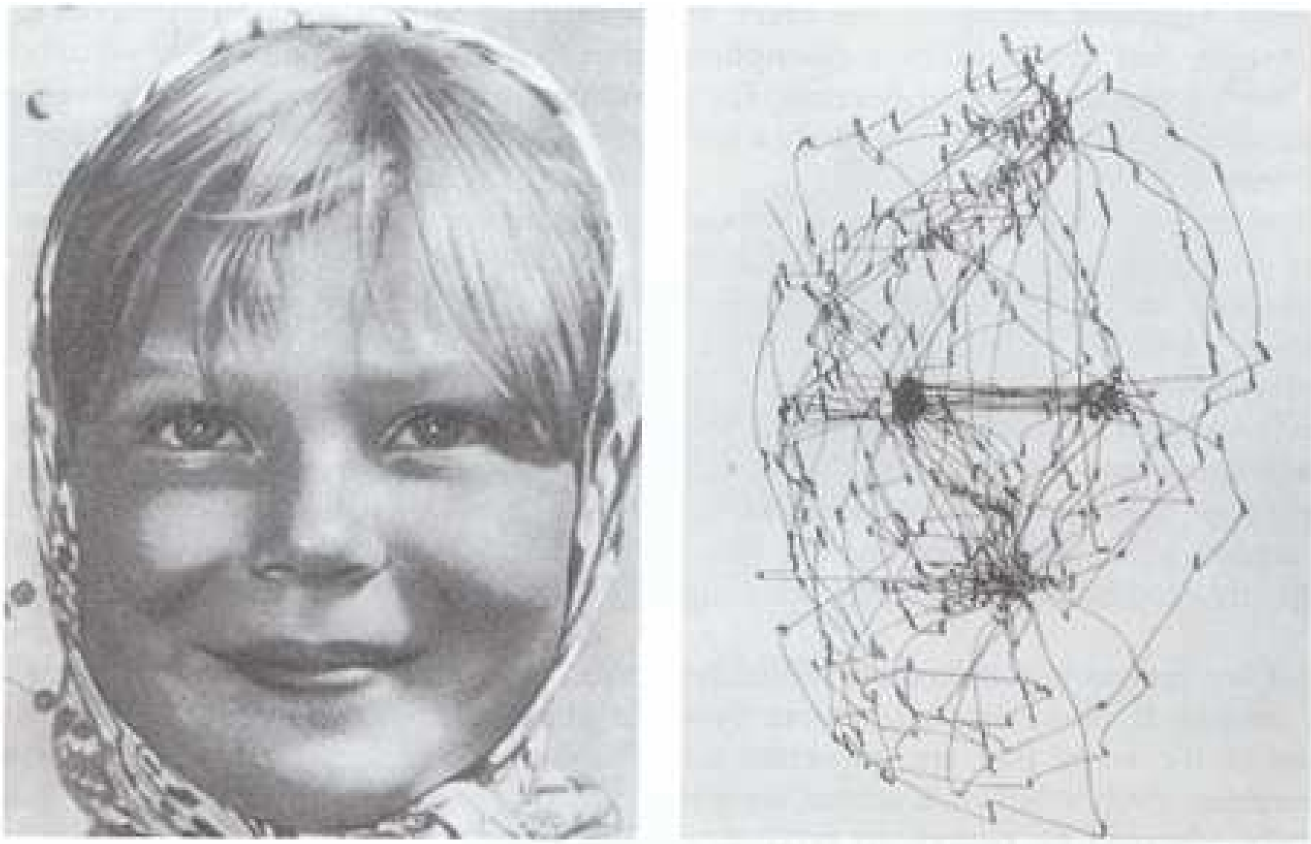
Lighting can be Distracting

As datasets and displays increase in size:

- Too many visual distractions
- Lots of low-information inconsequential detail
- Visual discovery hampered by low SNR



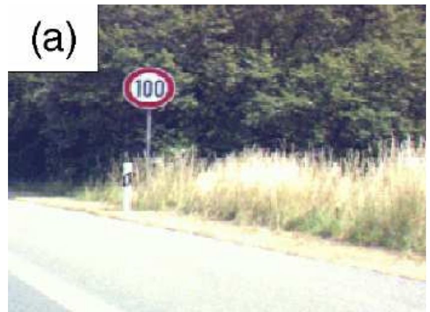
What is Salient?

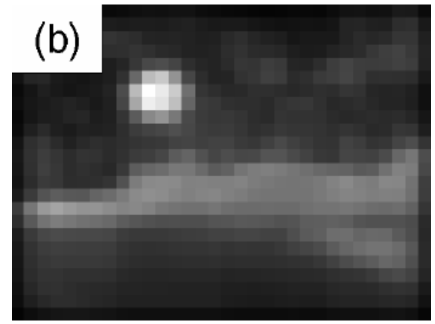


from [Yarbus 1967]

Related Work

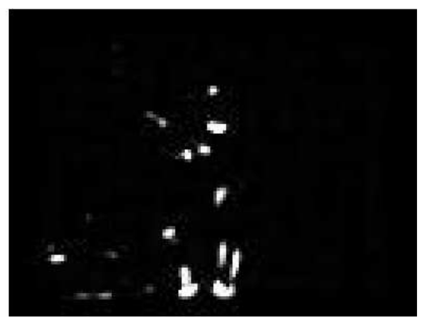
- Image saliency maps
  - Tsotsos *et al.* 95, Milanese *et al.* 94, Itti *et al*. 98, Rosenholtz 99





Itti *et al.* PAMI 98

- Applications: compression and cropping
  - Privitera and Stark 99, Chen *et al*. 03, Suh *et al*. 03



Without cropping

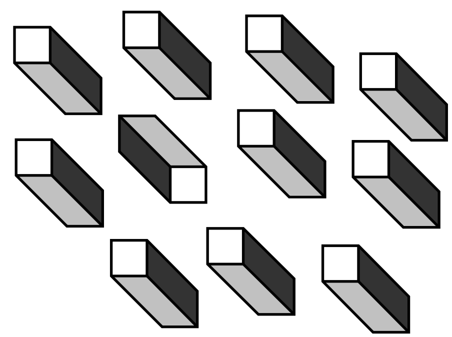
Saliency-based cropping

Suh *et al.* UIST 03

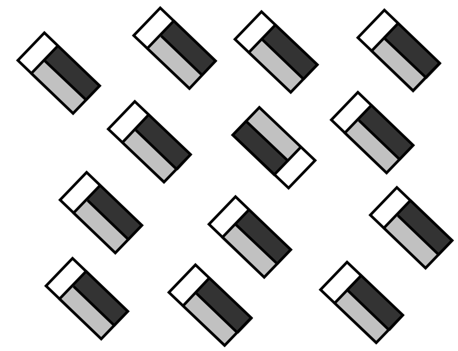
Related Work

3D object

- A distinctive 3D structure pops out pre-attentively



3D features
pop out quickly



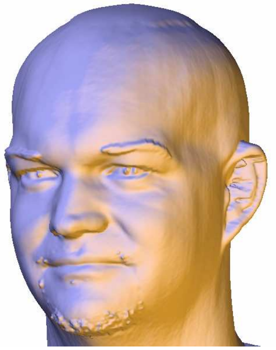
2D features
not pre-attentive

from [Enns and Rensink 90]

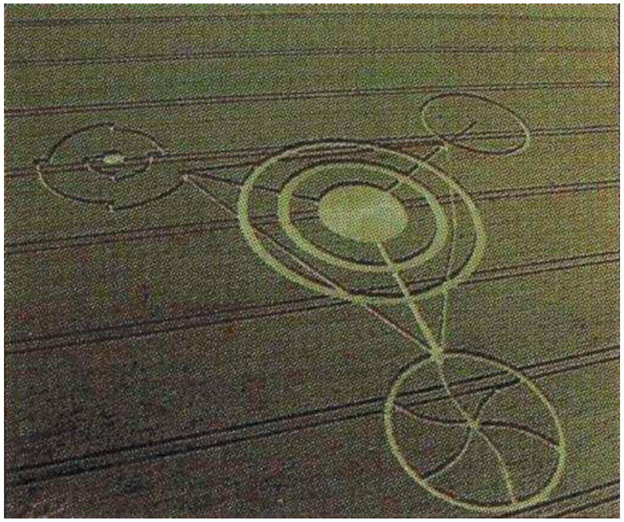
- Curvature
  - Watanabe and Belyaev *Eurographics 01*
  - Hisada *et al. Eurographics 02*
- Eye tracking
  - Howlett *et al. APGV 04*

What is Salient?

- High curvature is important



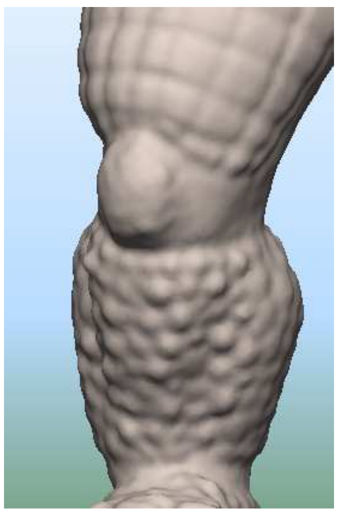

- but not always…

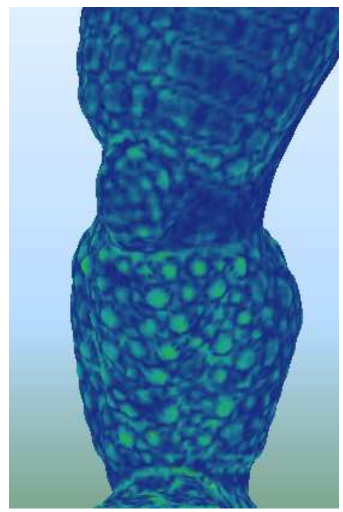



Mesh Saliency

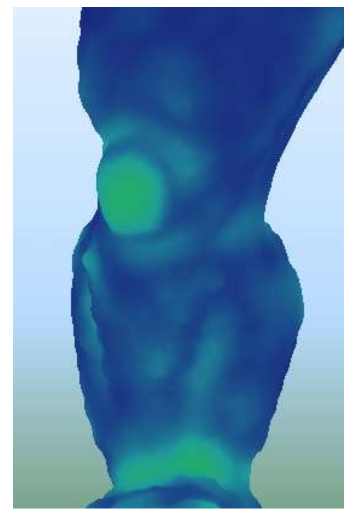
Center-Surround Mechanism

– Identify regions different from their surrounding





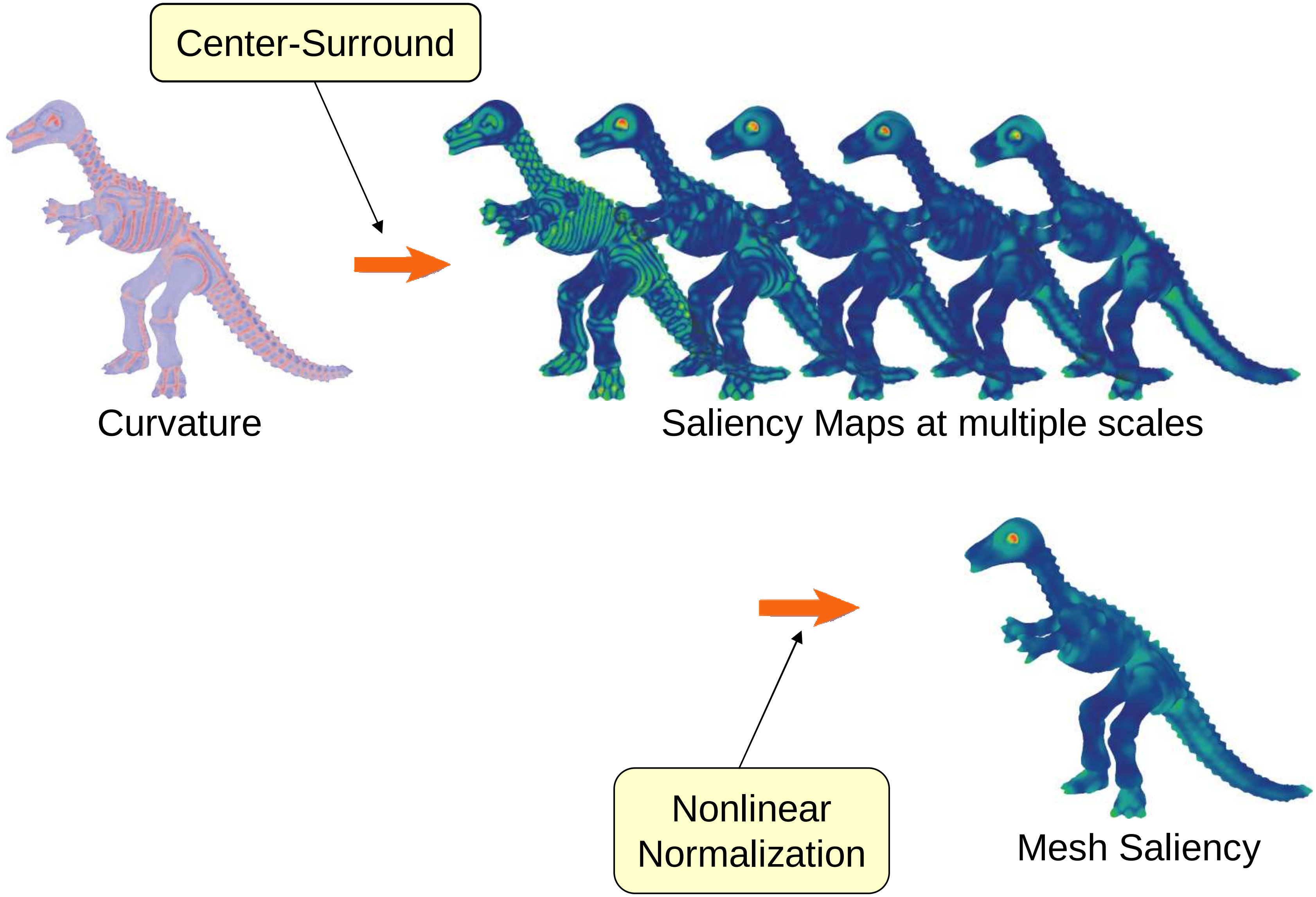
Curvature



Saliency

Lee, Varshney, Jacobs, *SIGGRAPH 2005*

Saliency Computation Overview



C. Lee, A. Varshney, D. Jacobs, *SIGGRAPH 2005*

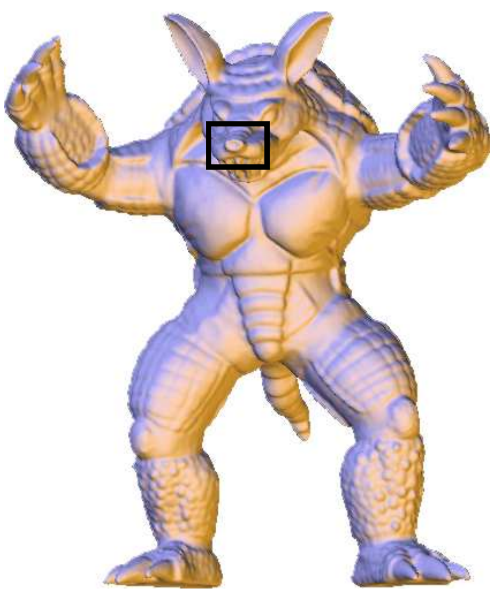
Center-Surround Operator

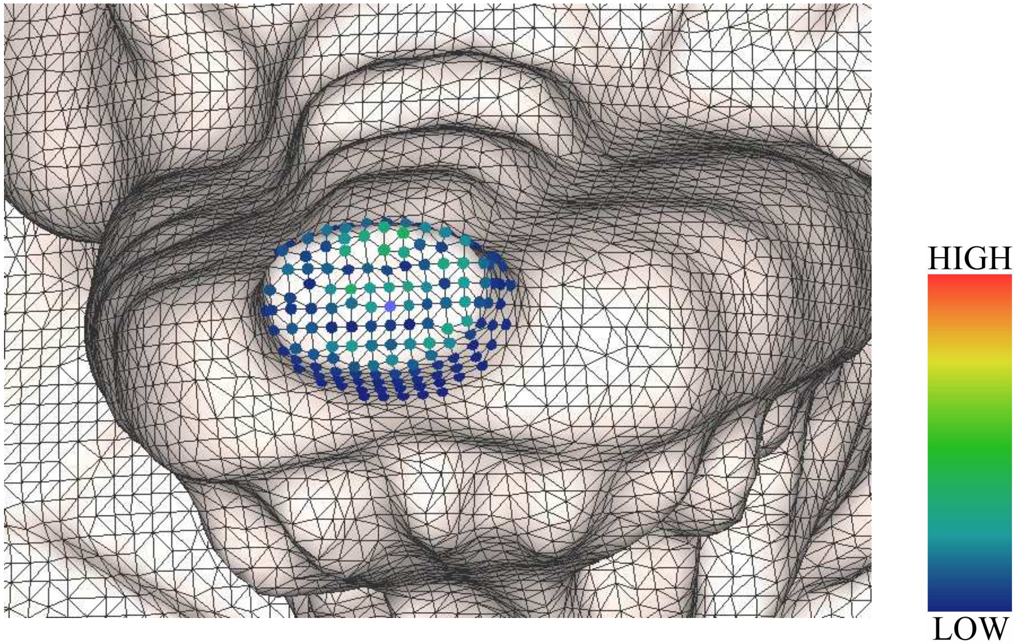
Gaussian-weighted average is:

$$G(\mathscr{C}(v),\sigma)=\frac{\sum_{x\in N(v,2\sigma)}\boxed{\mathscr{C}(x)}\,\boxed{exp[-\|x-v\|^2/(2\sigma^2)]}}{\boxed{\sum_{x\in N(v,2\sigma)}exp[-\|x-v\|^2/(2\sigma^2)]}}$$

$\mathscr{C}(x)$: Mean curvature at vertex v

Gaussian Weights



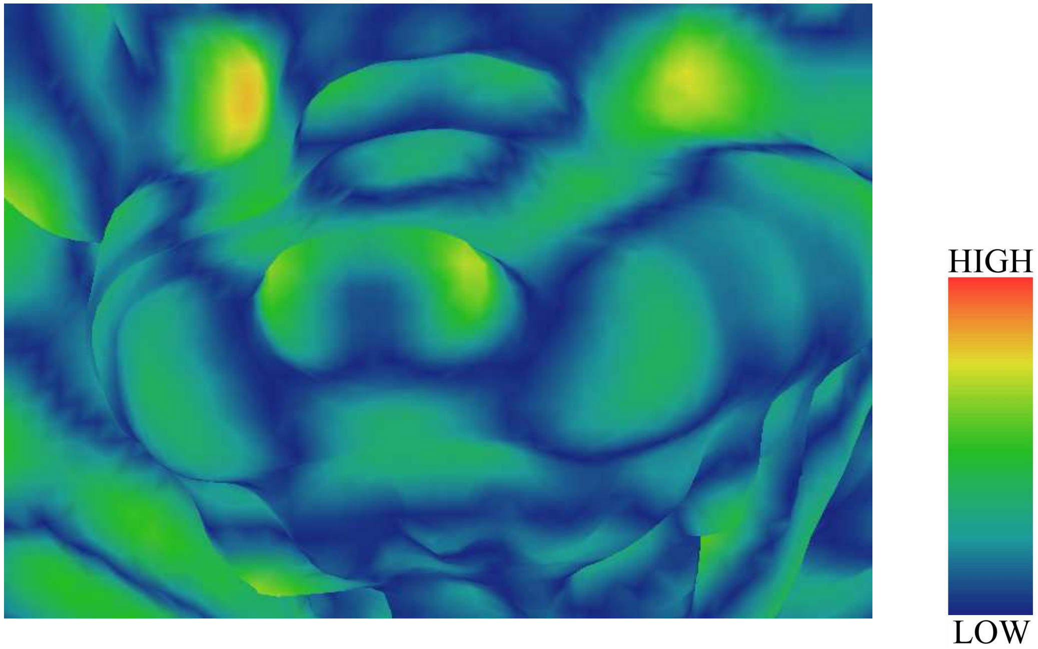


Center-Surround Operator

Saliency map at each scale *i* is:

$$\boxed{\mathscr{S}_i(v) = \left| \boxed{G(\mathscr{C}(v), \sigma_i)} - \boxed{G(\mathscr{C}(v), 2\sigma_i)} \right|}$$

$$\sigma_i \in \{2\varepsilon, 3\varepsilon, 4\varepsilon, 5\varepsilon, 6\varepsilon\}, \varepsilon = 0.3\% \text{ of the diagonal of the object}$$



Center-Surround Operator

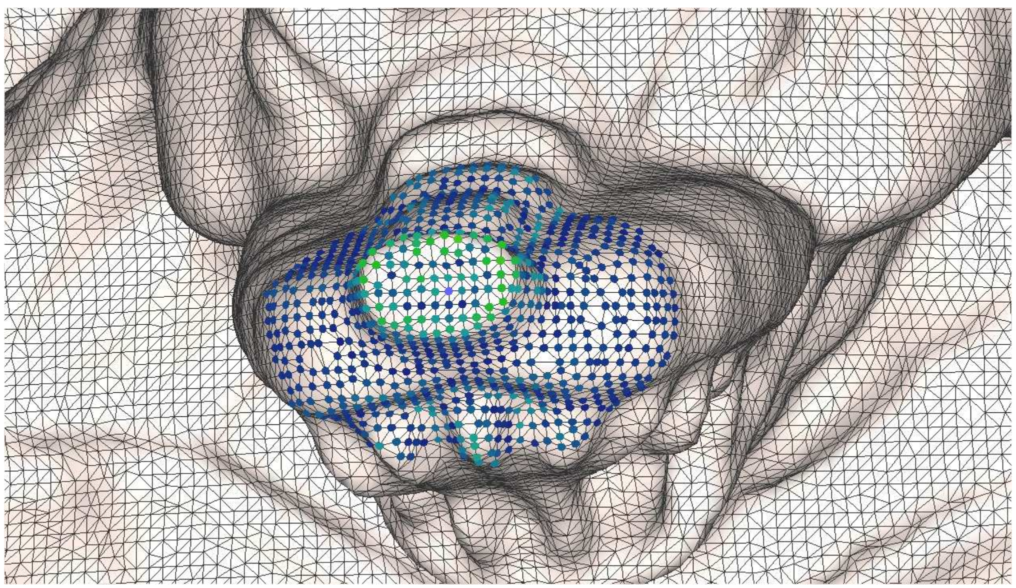
Saliency map at each scale i is:

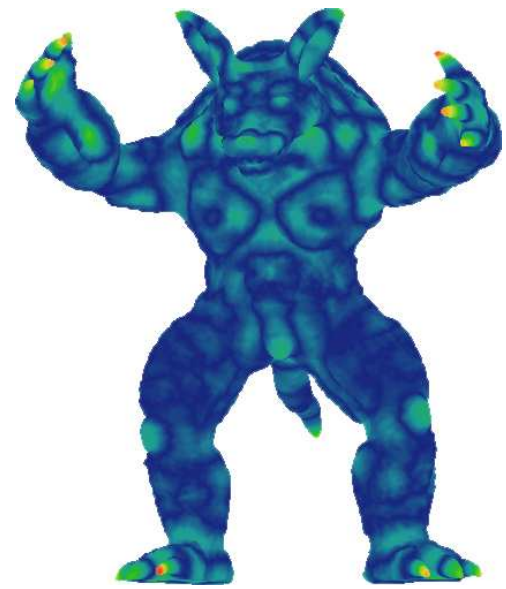
$$\mathscr{S}_i(v) = \left| G(\mathscr{C}(v), \sigma_i) - G(\mathscr{C}(v), 2\sigma_i) \right|$$

$$\sigma_i \in \{\boxed{2\varepsilon}, \boxed{3\varepsilon}, \boxed{4\varepsilon}, \boxed{5\varepsilon}, \boxed{6\varepsilon}\}$$

$$\mathscr{S}_i \in \{\boxed{\mathscr{S}_0}, \boxed{\mathscr{S}_1}, \boxed{\mathscr{S}_2}, \boxed{\mathscr{S}_3}, \boxed{\mathscr{S}_4}\}$$

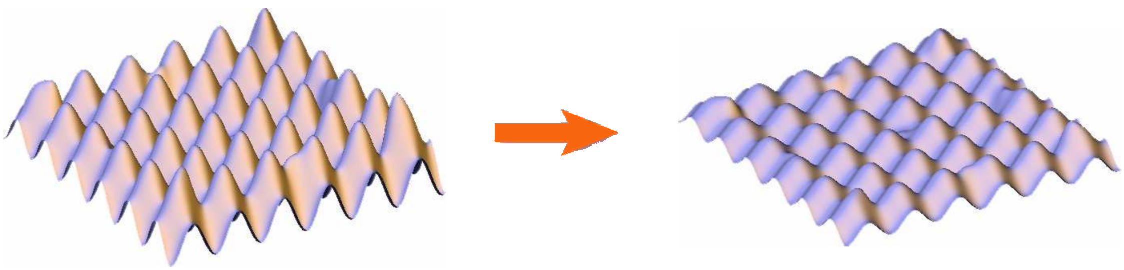
$\varepsilon = 0.3\%$ of the diagonal of the object

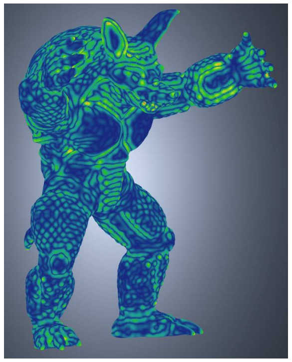




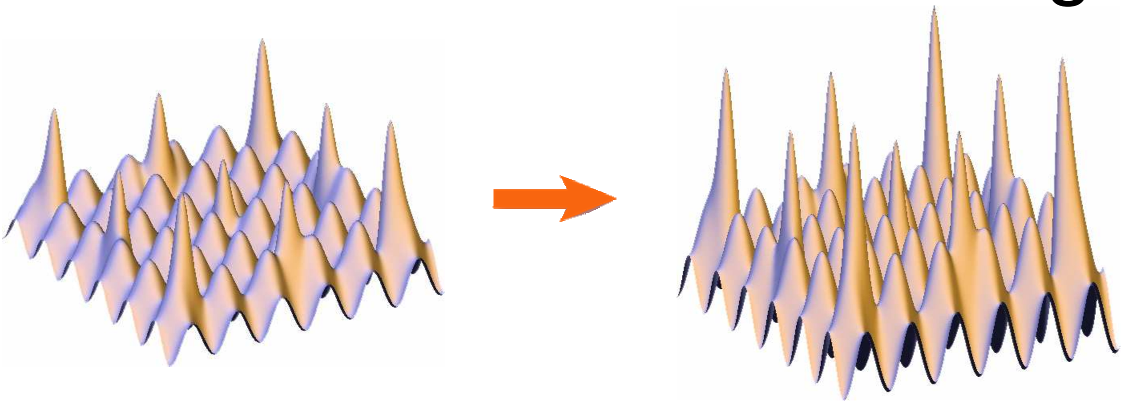
Nonlinear Normalization

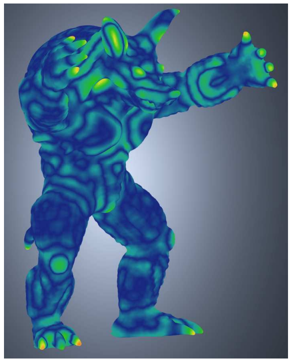
Suppress a large number of similar peaks





Promote a small number of high peaks





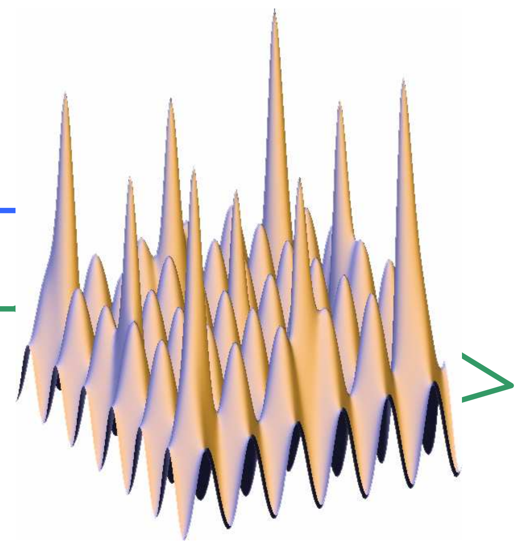
Nonlinear Normalization

The suppression operator is defined as:

$$S(\mathcal{S}_i) = (M_i - \overline{m}_i)^2 \mathcal{S}_i$$

M_i : The maximum saliency value

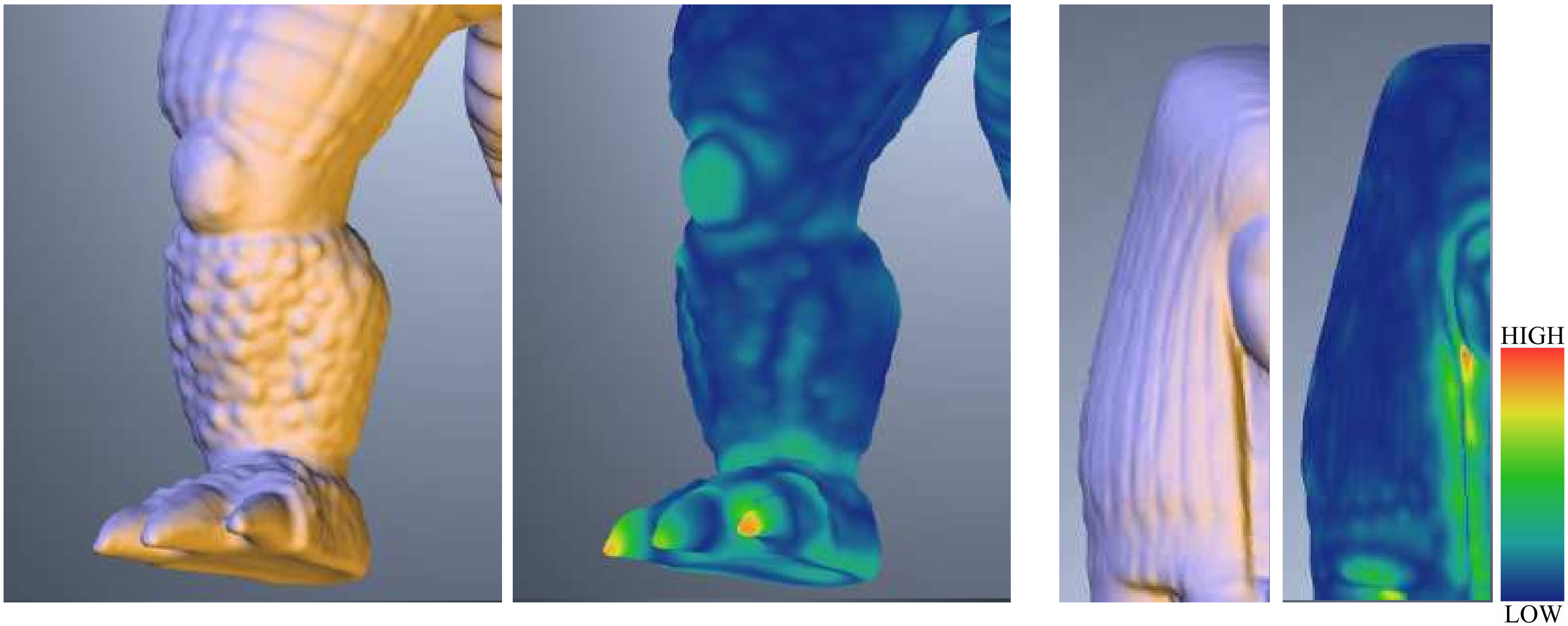
$\overline{m}_i$: The average of the local maxima



The final saliency map is:

$$\boxed{\mathcal{S} = \sum_i S(\mathcal{S}_i)}$$

Mesh Saliency Results



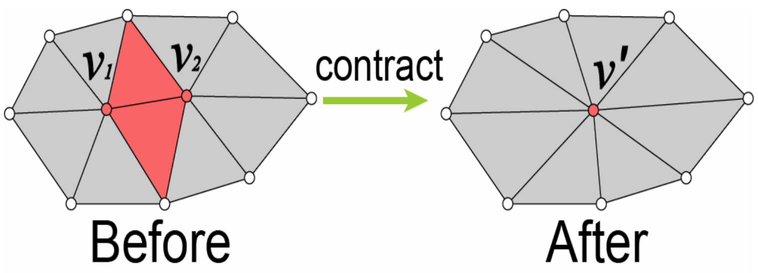
Stanford Armadillo

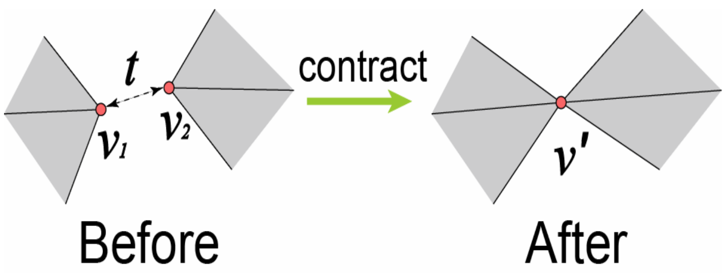
Cyberware Isis

Mesh Simplification

Qslim [Garland and Heckbert *SIGGRAPH 97*]

- Contracts edges until we get desired level of detail

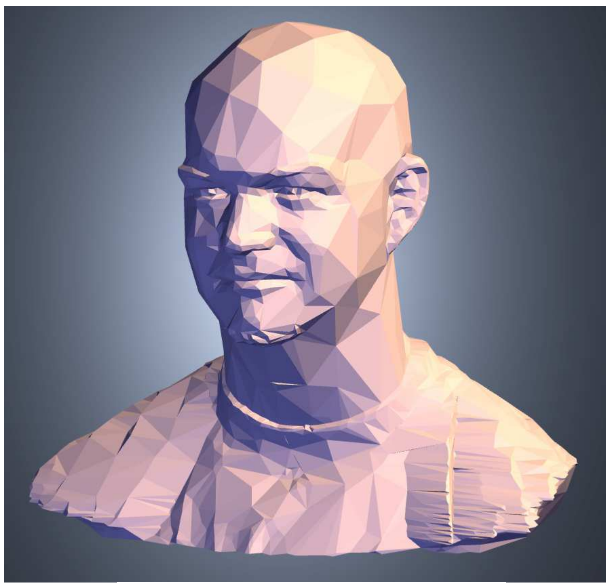




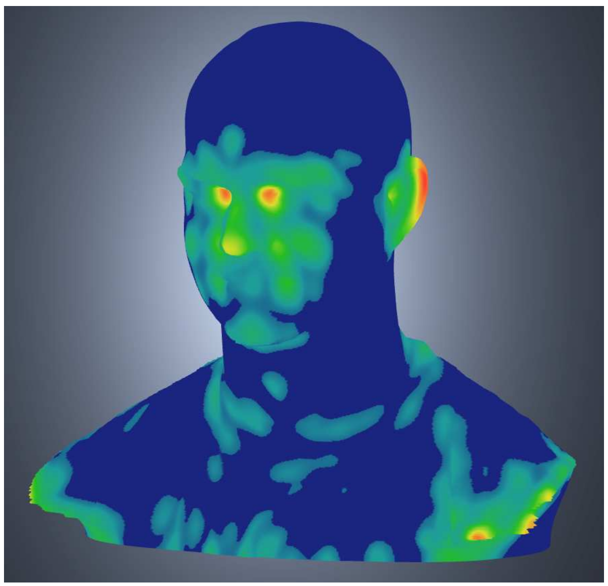
- Uses quadric error for determining the order of contraction

Saliency-guided Simplification

Scale the quadric error by the saliency to preserve more triangles for salient regions



Cyberware Male

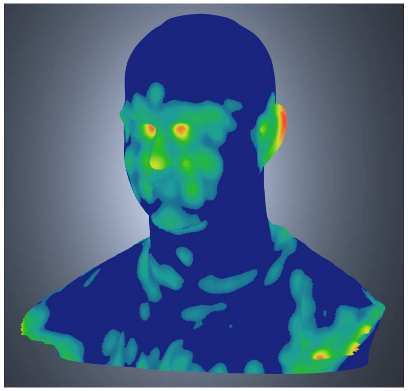


Mesh Saliency

Simplification Results

Simplification by Qslim
(4K tris)

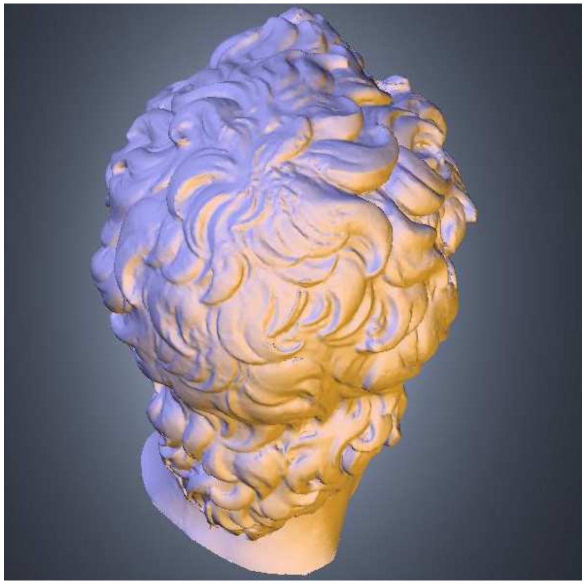
Simplification guided by Saliency
(4K tris)

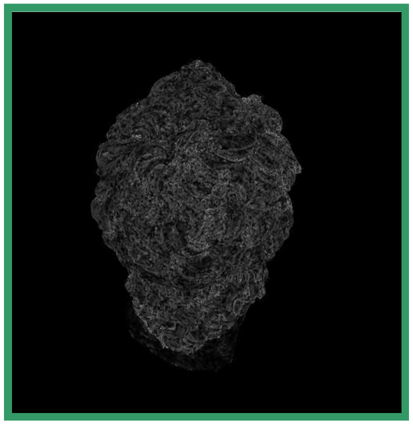


Saliency-guided Viewpoint Selection

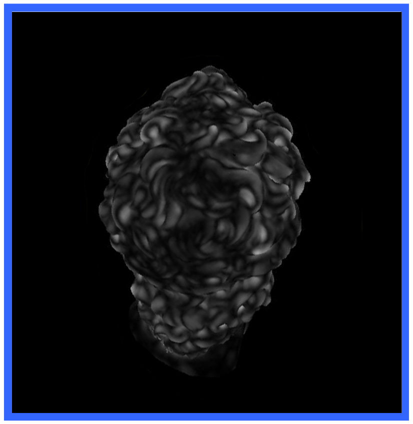
- Find the viewpoint that maximizes the sum of the visible saliency
- Gradient-descent-based optimization method for efficiency
  - Start with random points (1% of sample directions)
  - Try only 6% of 12K sample directions

Viewpoint Selection

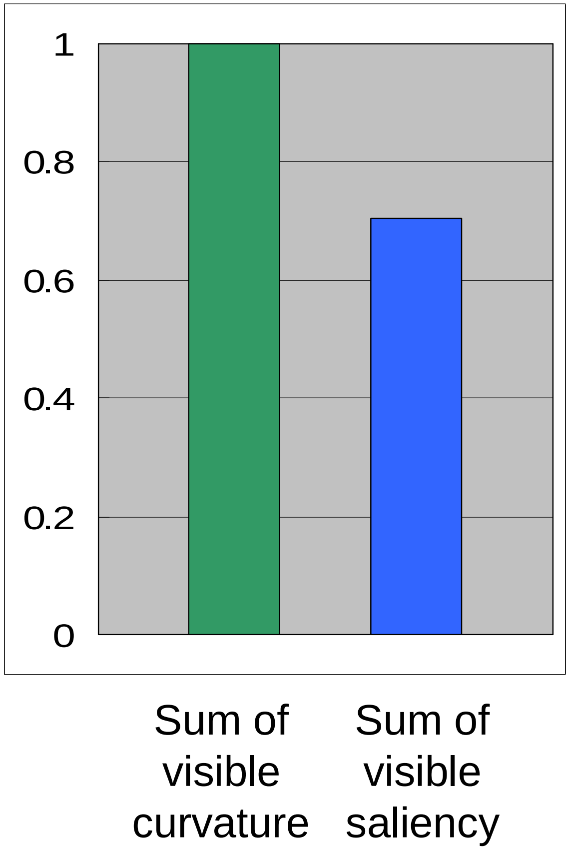




Curvature

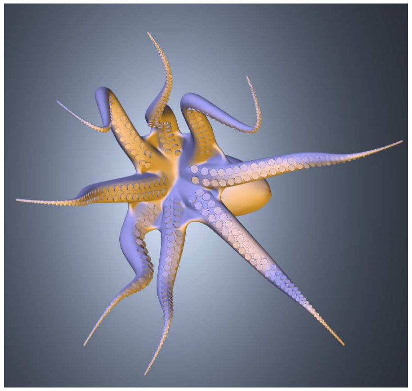


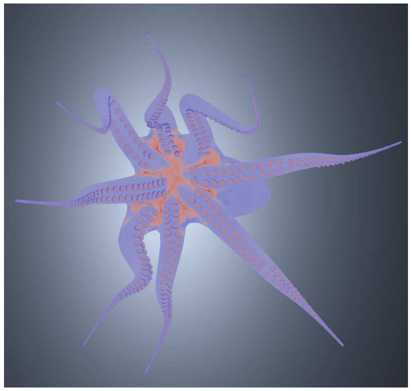
Saliency



Viewpoint Selection Results

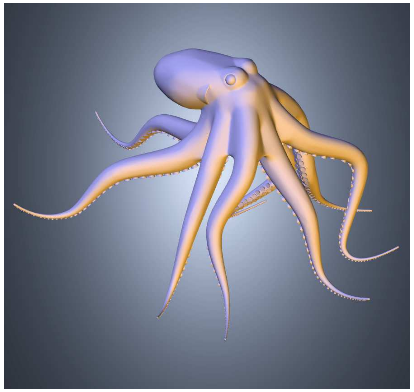
Curvature-based Viewpoint Selection





Curvature

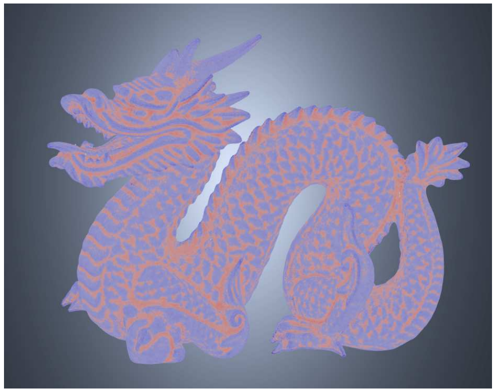
Saliency-based Viewpoint Selection



Saliency

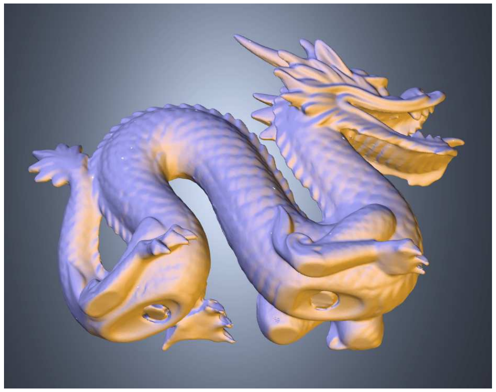
Viewpoint Selection Results

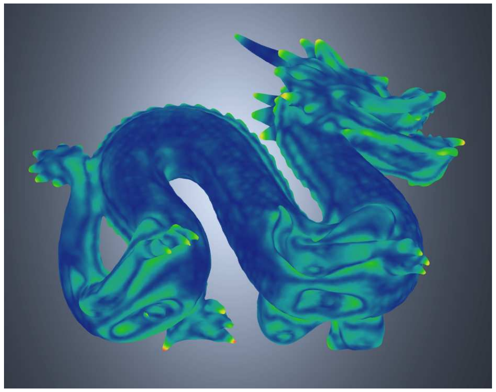
Curvature-based
Viewpoint Selection



Curvature

Saliency-based
Viewpoint Selection





Saliency

Conclusions & Future Work

- CPU-GPU coupling offers a promising platform for high-performance computing and visualization
- Real-time illumination models for graphics are getting to be fairly sophisticated and visually realistic
- Need further research on comprehension-friendly visual representations/abstractions
- Quantify improvement in task performance for specific applications by such representations
- Further research in visual computing will involve tightly integrating realism, modeling, and perception

Some Final Thoughts …

*The computer is a window into a virtual world that **looks** real, **feels** real, and **acts** real*

Ivan Sutherland (1968)

*The computer scientist is a **toolsmith***

Fred Brooks (1977)

Acknowledgements

David Dao
Youngmin Kim
Chang Ha Lee
Derek Juba

Joseph Ja'Ja'
Dianne O'Leary
David Jacobs

National Science Foundation

Questions?