

Lecture 21: Point-based Rendering

Fall 2004
Kavita Bala
Computer Science
Cornell University

Announcements

- In-class exam next week Nov 18th.
- Regrade requests in writing
 - Will regrade whole assignment

© Kavita Bala, Computer Science, Cornell University

Complexity

- Lighting: many lights, environment maps
 - Global illumination, shadows
- Materials: BRDFs, textures
- Geometry: Level-of-detail, point-based representations
- All: impostors, image-based rendering

© Kavita Bala, Computer Science, Cornell University

Motivation

- Scene complexity is increasing
- Scanning is producing large point datasets
- Procedural model generation (trees, plants)



4 million pts.
[Levoy et al. 2000]

© Kavita Bala, Computer Science, Cornell University

Motivation

- Creating meshes from scanned datasets
 - Hard
 - Not robust
- Projected triangles too small
 - Many triangles per pixel
 - Setup and rasterization useless

© Kavita Bala, Computer Science, Cornell University

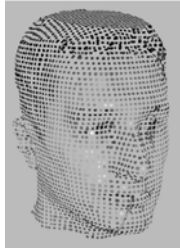
Insight

- Use points as a rendering primitive
- Avoid creating meshes
 - Connectivity information
 - More robust
 - Compact
 - Matches data sets better...

© Kavita Bala, Computer Science, Cornell University

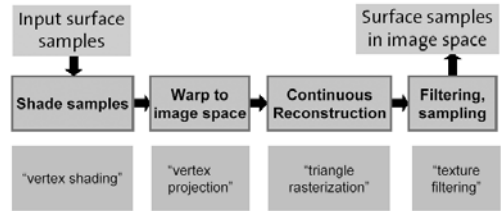
Point-Based Representation

- Point cloud represents
 - 3D geometry of surface
 - Surface reflectance
 - Diffuse, color
- No connectivity
- No texture information



© Kavita Bala, Computer Science, Cornell University

Rendering Pipeline



© Kavita Bala, Computer Science, Cornell University

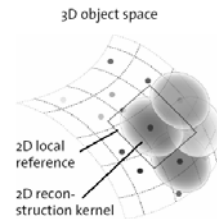
Rendering points

- Map a point to image plane
- What do we do with holes?
- Filter kernels (Gaussian)
 - Merge nearby points to reconstruct pixel

© Kavita Bala, Computer Science, Cornell University

Surface Splatting

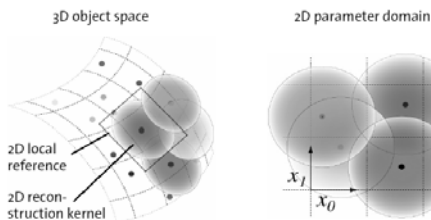
- Surface samples are specified in local reference frame with respect to normal
 - Sphere or disk representation



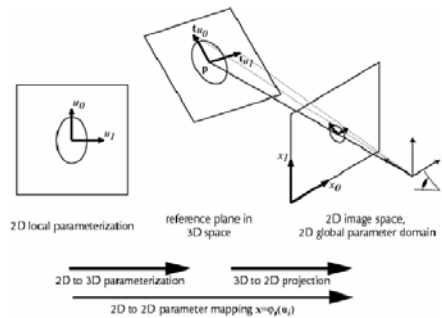
© Kavita Bala, Computer Science, Cornell University

Splat on image plane

- Warp to image space
 - 2D to 2D projective mapping

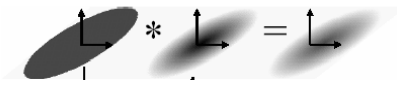


© Kavita Bala, Computer Science, Cornell University



© Kavita Bala, Computer Science, Cornell University

Combining multiple points



- Weighted sum of kernels in image space
 - Normalize weights of kernels

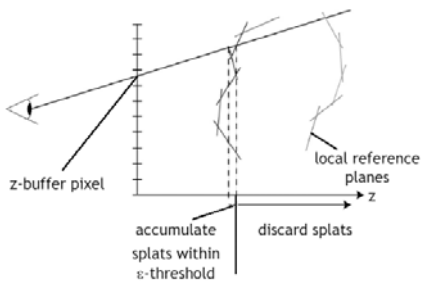
© Kavita Bala, Computer Science, Cornell University

Algorithm

- For each point
 - Shade point
 - Splat = projected reconstruction filter kernel
 - Rasterize and accumulate splat
- For each output pixel
 - normalize

© Kavita Bala, Computer Science, Cornell University

Z-buffer



© Kavita Bala, Computer Science, Cornell University

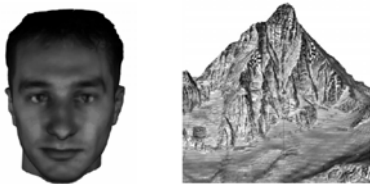
3-pass GPU Algorithm

- Pass 1: Depth image with depth offset epsilon away from viewpoint
 - Do z-buffer tests
- Pass 2: Draw colored splats with additive blending. Accumulate
 - Colors of visible splats in color channels
 - Visible footprint in alpha channels
- 3rd pass: Normalize color channels (divide by alpha channel)

© Kavita Bala, Computer Science, Cornell University

Results

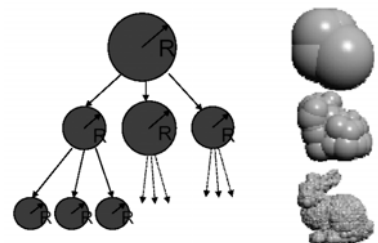
- Scanned head: 429k points
- Matterhorn: 4,787k points
- On GPUs: 3M points/sec



© Kavita Bala, Computer Science, Cornell University

LODs with points

- Hierarchical data structure
- Q-splat [SIGGRAPH 2000]



© Kavita Bala, Computer Science, Cornell University

Construction

- Each vertex of original mesh is leaf sphere (such that adjacent vertices overlap)
- Construct top down
- Store sphere center, radius, normal
 - All quantized for compactness

© Kavita Bala, Computer Science, Cornell University

Hierarchical Traversal

- recursive rendering

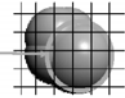
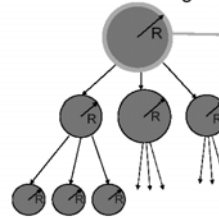


image size > 1 pixel

→ traverse children

© Kavita Bala, Computer Science, Cornell University

- recursive rendering

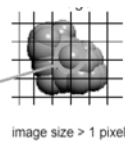
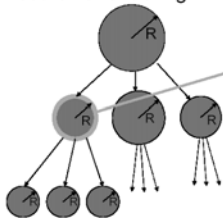


image size > 1 pixel

→ traverse children

© Kavita Bala, Computer Science, Cornell University

- recursive rendering

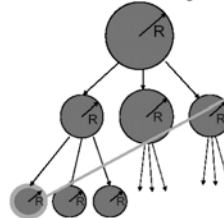


image size < 1 pixel

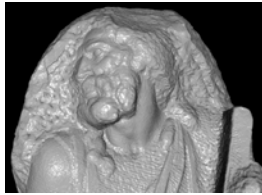
→ render disk

© Kavita Bala, Computer Science, Cornell University

Results



130k splats, 132 ms



1M splats, 722 ms

© Kavita Bala, Computer Science, Cornell University

Other point-based work

- Anti-aliasing of points/textures
- Hybrid rendering: polygons and points
- Point editing and animation
- Expensive shading with points: open question

© Kavita Bala, Computer Science, Cornell University

Complexity

- Lighting: many lights, environment maps
 - Global illumination, shadows
- Materials: BRDFs, textures
- Geometry: Level-of-detail, point-based representations
- All: image-based rendering

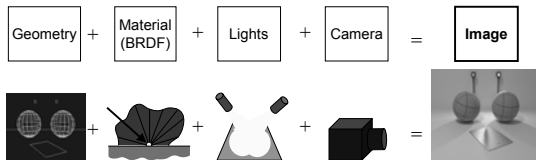
© Kavita Bala, Computer Science, Cornell University

Scene Complexity



© Kavita Bala, Computer Science, Cornell University

Computer Graphics



© Kavita Bala, Computer Science, Cornell University

Why is image generation slow?

- Requires labor-intensive modeling: geometry and BRDF
 - Hard
 - Tedious
 - Error-prone
- Rendering time long
 - Global illumination
 - Proportional to complexity



© Kavita Bala, Computer Science, Cornell University

One Approach: Texture Mapping

- Use textures to create the effect of complex geometry and lighting conditions
 - displacement mapping
 - change position of surface
 - bump mapping
 - change normal
 - reflection/environment mapping

© Kavita Bala, Computer Science, Cornell University

Bump Mapping



© Kavita Bala, Computer Science, Cornell University

Reflection Mapping



(Terminator II - 1991)

© Kavita Bala, Computer Science, Cornell University

Texture Mapping not enough!

- How do we create textures?
 - Model BRDFs and colors
- To what geometry should we apply textures? How?
 - Model geometry
 - But, simple models
 - flat textures, don't look good
 - Complex models
 - time consuming, tedious, hard to map

© Kavita Bala, Computer Science, Cornell University

Idea

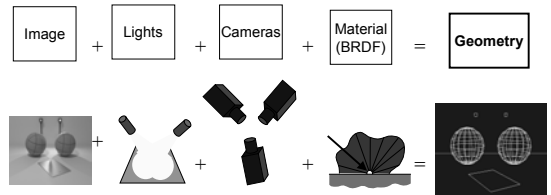
- Can we use photographs?



- **Photographs capture**
 - High geometric complexity
 - High lighting and material (BRDF) complexity
- How do we use them?

© Kavita Bala, Computer Science, Cornell University

Machine Vision

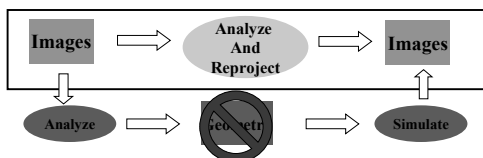


- **Given images, find geometry of scene**
- **Problem: very hard inverse problem**
 - too many unknowns

© Kavita Bala, Computer Science, Cornell University

Image-based Approaches

- Combine vision and graphics
- Given images and *some* geometry
 - Render new images from existing images
 - New idea: Image is input *and* rendering primitive
 - No (or very little) geometry recovery



© Kavita Bala, Computer Science, Cornell University

Pros

- Promising approach to handle complexity
- Benefits:
 - No labor-intensive modeling
 - Captures high geometric/material complexity
 - Rendering time constant: proportional to image size, independent of scene complexity

© Kavita Bala, Computer Science, Cornell University

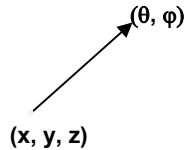
Outline

- Theory
- Image-based Rendering
- Image-based Modeling
 - Façade

© Kavita Bala, Computer Science, Cornell University

The Plenoptic Function

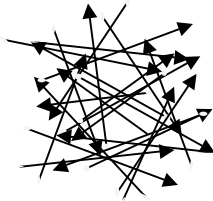
- $P(x, y, z, \theta, \phi)$: radiance over all points in space and in all directions
 - 5D function: theoretical concept
- Why do we care? Rendering computes P



© Kavita Bala, Computer Science, Cornell University

Plenoptic function

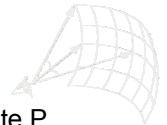
- Radiance value for all possible rays = P



© Kavita Bala, Computer Science, Cornell University


Images are subset of P

- Think of an image in a new way!!!
- Image = radiance for each ray in image
 - = radiance through a collection of rays
 - = subset of plenoptic function P
- 1 Input image = subset of P
- Several input images approximate P
- All possible images = P



© Kavita Bala, Computer Science, Cornell University

IBR idea

- Idea: Replace scene by images
- Output: new viewpoint
 - Look up plenoptic fn.  look up input images
- What are the assumptions?
 - Existing scene
 - Static scene
 - Fixed lighting

© Kavita Bala, Computer Science, Cornell University

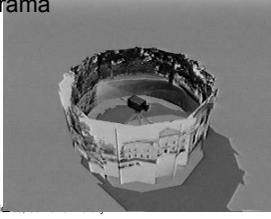
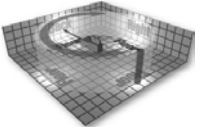
Approaches

- Systems that have no depth
 - Quicktime VR
 - Plenoptic Modeling
 - Lightfields/Lumigraphs
 - Image-based visual hulls
- Systems that have full geometry
 - Surface Lightfields
- Systems that have partial geometry: Image-Based Modeling
 - Façade

© Kavita Bala, Computer Science, Cornell University

QuickTime VR

- Fixed viewpoint + full range of viewing directions (360°)
- Panoramic images:
 - Stitch image to form panorama
 - Can look around panorama



© Kavita Bala, Computer Science, Cornell University

Quicktime VR

- Demo
- Pros
 - Simple, fast, effective
- Cons
 - Camera position is confined to predefined observer positions
 - Distortion when user deviates from position

© Kavita Bala, Computer Science, Cornell University