Image Based Rendering Representations

CS 319
Advanced Topics in Computer Graphics
John C. Hart
Accurate IBR

- Is there a “photograph” that gives us all the information we need to view a scene correctly from any viewpoint?
  - What dimension is the “image?”
  - How can we represent the “image?”

- Answers
  - Light Field
  - Lumigraph
  - Layered Depth Image
How Much Light is Really in a Scene?

- Light transported throughout scene along rays
  - Anchor
    - Any point in 3-D space
    - 3 coordinates
  - Direction
    - Any 3-D unit vector
    - 2 angles
    - Total of 5 dimensions
- Radiance remains constant along ray
  - Removes one dimension
  - Total of 4 dimensions
Representing All of the Light in a Scene

• View scene through a window
• All visible light from scene must have passed through window
• Window light is 4-D
  – 2 coordinates where ray intersects window pane
  – 2 angles for ray direction
• Use a double-paned window
  – 2 coordinates \((u,v)\) where ray intersects first pane
  – 2 coordinates \((s,t)\) where ray intersects second pane
Light Field v. Lumigraph

- Light Field Rendering
  - Levoy & Hanrahan, S96
- Lumigraph
  - Gortler et al., S96
- Consider \((u, v)\) the image plane and \((s, t)\) the viewpoint plane
- Remember depth of field?
- Photographs from a bunch of different viewpoints
- Reconstructed photographs of scene are 2-D slices of 4-D light field
Ray Tracing and Light Fields

• Rendering into a light field
  – Cast rays between all pairs of points in panes
  – Store resulting radiance at \((u,v,s,t)\)

• Rendering from a light field
  – Cast rays through pixels into light field
  – Compute two ray-plane intersections to find \((u,v,s,t)\)
  – Interpolate u,v and s,t to find radiance between samples
  – Plot radiance in pixel
Antialiasing and Light Fields

- Light field aliases
  - jagged edges
  - jumping between discretized images when animated
- Correct sampling uses depth of field from distribution ray tracing
- Circle of confusion equals distance between camera positions
Results
Image Based Rendering - Big Issues

- **Representation**
  - 3D Implicit
    - multi-frame mosaics and local linearisations
    - frame-frame transfer
    - light fields, etc.
  - 3D Explicit
    - meshes of polygons, splines, etc.
    - assemblies of primitives

- **Recovery**
  - implicit
  - specialised cameras
  - software mosaicing
  - sampling issues
  - Explicit
    - relations between views;
    - between appearance and shape
  - Both
    - correspondence: manual vs automatic
Implicit example: Quicktime VR

- Construct a mosaic that can be queried to provide various camera views at various points.
- Issues:
  - recovering the mosaics
    - specialised hardware
    - correlation based mosaicing
  - structuring the representation for fast rendering
  - geometry of views
  - incremental view relations

Figures from “QuickTime VR – An Image-Based Approach to Virtual Environment Navigation”, Shenchang Eric Chen, SIGGRAPH 95
Figures from “QuickTime VR – An Image-Based Approach to Virtual Environment Navigation”, Shenchang Eric Chen, SIGGRAPH 95
Figures from “QuickTime VR – An Image-Based Approach to Virtual Environment Navigation”, Shenchang Eric Chen, SIGGRAPH 95
Image Warping

- Warping allows us to replace geometric detail with textures
  - Textures created from photographs
  - Mapped to coarse planar model
- Warping problems
  - Warping incorrect for non-planes
  - Depth warping creates “holes”
- Image warping alone not enough to correctly reconstruct arbitrary scene
Layered Depth Images

- Shade et al. S98
- Replace z-buffer with depth-sorted list of all objects intersected by the ray
  - Compare to Roth’s CSG
  - Compare to Catmull’s A-buffer
- Three-dimensional “solid” image
  - Compare \((x,y,z)\) to \((u,v,s,t)\)
Fast LDI Display

- Reconstruct new view of an LDI by warping each depth pixel individually
- Prevents holes from occlusion
- Location of depth pixel in new image
  - scale \textbf{depth} by depth pixel z value
  - add result to \textbf{start}
  - divide by homogeneous coordinate
- Location of start for next pixel found by adding a constant vector
- Need to also compute splat footprint
  - Area of screen onto which the LDI sample projects

\[
\begin{bmatrix}
  x_1 w \\
  y_1 w \\
  z_1 w \\
  w
\end{bmatrix} = M_1 M_0^{-1} \begin{bmatrix}
  x_0 \\
  y_0 \\
  z_0 \\
  1
\end{bmatrix}
\]

\[
= M_1 M_0^{-1} \begin{bmatrix}
  x_0 \\
  y_0 \\
  0 \\
  1
\end{bmatrix} + z_0 M_1 M_0^{-1} \begin{bmatrix}
  0 \\
  0 \\
  1 \\
  0
\end{bmatrix}
\]

\[
= \text{start} + z_0 \times \text{depth}
\]

\[
M_1 M_0^{-1} \begin{bmatrix}
  x_0 + 1 \\
  y_0 \\
  0 \\
  1
\end{bmatrix}
\]

\[
= M_1 M_0^{-1} \begin{bmatrix}
  x_0 \\
  y_0 \\
  0 \\
  1
\end{bmatrix} + M_1 M_0^{-1} \begin{bmatrix}
  1 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\]

\[
= \text{start} + \text{xincr}
\]