Lecture 22: Image-based Rendering

Fall 2004
Kavita Bala
Computer Science
Cornell University

Announcements
- In-class exam next week Nov 18th
  - Will post last year’s exam on CMS
- HW 3
  - First, make it work
  - Then optimize
  - Use results reported as guide

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

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

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

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
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 \( P \)

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 = \( P \)

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

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
- Synthetic systems: impostors

QuickTime VR

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

Quicktime VR

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

- Input: set of images (panoramic)
- Output: images from new viewpoint
  - Removes constraint on new viewpoint position

- How?
  - Reconstruct the plenoptic function from the images
  - Assumes depth/disparity information

Pixel Reprojection

- Goal: Want image at new viewpoint
- Reproject points from input images

Pixel Reprojection

- Direction D
  \[ D = C + x \mathbf{i} + y \mathbf{j} + d \mathbf{k} \]
  \[(x, y) = \text{pixel}\]
- C = camera center
- d = distance of image plane from C
- C, d are known

Reprojection

\[ P = C_0 + t_0 D_0(x, y) \]
\[ C_0 + t_0 D_0 = C_1 + t_1 D_1 \]
\[ t_1 D_1 = (C_0 - C_1) + t_0 D_0 \]
\[ t_1, D_1 \text{ defines the reprojected pixel} \]
**Pixel Reprojection**

- $D_1 = C_1 + x_1 i + y_1 j + d_1 k$
- Solve for $x_1, y_1$ and $t_1$

**Problems with Reprojection**

- Aliasing: pixels do not project to pixel centers
  - Solution: Splatting
- Multiple pixels project to same pixel in new view
  - Solution: z-buffer

**Reprojection Example**

**How to compute depth/disparity?**

- Assumption: disparity is known
- Correspondences specified by user
- Recover point (depth/disparity)
Computing depth/disparity

- \( P = C_0 + t_0 D_0(x,y) \)
- \( C_0 + t_0 D_0 = C_1 + t_1 D_1 \)
- \( C_0, C_1, D_0, D_1 \) are known
- Solve for \( t_0 \) and \( t_1 \)

Epipolar geometry

- Specifying correspondence: tedious
- Disparity/depth recovery using epipolar geometry
- Ray corresponds to epipolar line in \( C_1 \)'s image plane

Epipolar geometry

- Different depths correspond to different points on epipolar line

Epipolar geometry

- We don't know depth, but we know the ray
- Given color at pixel \((x, y)\) search along epipolar line for pixel of same color
- Find match, recover depth/disparity
- Problem?

Demo

- Cylindrical epipolar geometry

Plenoptic Issues

- Hard to get accurate depth/disparity
  - View-dependence
- From new viewpoints have holes to fill
  - Interpolation blurs
Lumigraph / Light field
- Idea: capture many photographs from different views
- No depth information
- Render image from new viewpoint using existing images
  - Have to lie outside object

What is an image?
- Image = rays going through one point
- Usually restricted to viewing frustum, but can also be panoramic

What is an object?
- Outgoing radiance field of an object
- Radiance varies at points on surface
  - 2D function (position on surface)
- Radiance varies in all directions
  - 2D function

What is an object?
- All possible images of an object
- We don’t really need the object

Replace object by images
- Object is only defined by its radiance field
- Images capture all information about object
- New viewpoint: look up appropriate images
Questions

• How to capture the input images?
• How to store the images efficiently for retrieval?
• How to render new images?

Rays: 2 plane parameterization

\[ (\theta, \phi) \]
\[ (x, y, z) \]
\[ (s, t) \]
\[ (u, v) \]

Lumigraph organization

• Hold \((s, t)\) constant: an image

Lumigraph organization

• Hold \((s, t)\) constant: an image

Lumigraph organization

• Is this an image?

LightField/Lumigraph Idea

• Move camera carefully in \((s, t)\) plane
• Each image is a 2D slice of 4D function
• Hold \((s, t)\) constant and get an image
Fish LightField

- Images are a database of rays
  - store in 4D array
- Demo (1,2)

Lumigraph - rendering

- Look for closest \((s,t,u,v)\) tuplet

Lumigraph - rendering

- Interpolation of 16 values

Lumigraph organization

- Higher resolution near object
  - captures texture
- Lower resolution far away
  - captures direction

LightField/Lumigraph Pros/Cons

- Pros
  - No depth information at all
  - Interactive performance
- Cons
  - Lots of images!!! (w/ compression 100s MB)
  - Specialized hardware to compute images
  - Constrained to lie outside the object
  - Works for small objects
  - Blurry results