Digital Photography and Videos

Jingyi Yu
CISC 849
Light Fields, Lumigraph, and Image-based Rendering
Pinhole Camera

- A camera captures a set of rays
- A pinhole camera captures a set of rays passing through a common 3D point
Camera Array and Light Fields

- Digital cameras are cheap
  - Low image quality
  - No sophisticated aperture (depth of view) effects
- Build up an array of cameras
- It captures an array of images
- It captures an array of set of rays
Why is it useful?

• If we want to render a new image
  – We can query each new ray into the light field ray database
  – Interpolate each ray (we will see how)
  – No 3D geometry is needed
Key Problem

• How to represent a ray?
• What coordinate system to use?
• How to represent a line in 3D space
  – Two points on the line (3D + 3D = 6D) Problem?
  – A point and a direction (3D + 2D = 5D) Problem?
  – Any better parameterization?
Two Plane Parameterization

- Each ray is parameterized by its two intersection points with two fixed planes.
- For simplicity, assume the two planes are \( z = 0 \) (st plane) and \( z = 1 \) (uv plane).
- Alternatively, we can view \((s, t)\) as the camera index and \((u, v)\) as the pixel index.
For the moment let's consider just a 2D slice of this 4D ray space. Rendering new pictures amounts to interpolating rays.
Light Field Rendering

• For each desired ray:
  - Compute intersection with \((u,v)\) and \((s,t)\) plane
  - Blend closest ray
  - What does closest mean?

\[
\begin{align*}
u_1 &= u_2 = u \\
v_1 &= v_2 = v \\
s_1 &= \lfloor s \rfloor, \quad s_2 = \lceil s \rceil \\
t_1 &= \lfloor t \rfloor, \quad t_2 = \lceil t \rceil
\end{align*}
\]
Light Field Rendering

• Linear Interpolation

$$\alpha = \frac{x - x_1}{x_2 - x_1}$$

$$y = \alpha y_2 + (1 - \alpha) y_1$$

• What happens to higher dimension?
  Bilinear, tri-linear, quadra-linear

$$u_1 = u_2 = u$$

$$v_1 = v_2 = v$$

$$s_1 = \begin{bmatrix} s \end{bmatrix}, \quad s_2 = \begin{bmatrix} s \end{bmatrix}$$

$$t_1 = \begin{bmatrix} t \end{bmatrix}, \quad t_2 = \begin{bmatrix} t \end{bmatrix}$$
Quadralinear Interpolation

Serious aliasing artifacts

Department of Computer and Information Science
Ray Structure

All the rays in a light field passing through a 3D geometry point

2D Light Field

2D EPI

Department of Computer and Information Science
Why Aliasing?

- We study aliasing in spatial domain
- Next class, we will study frequency domain (take a quick review of Fourier transformation if you can)
Better Reconstruction Method

• Assume some geometric proxy
• Dynamically Reparametrized Light Field

$$uv_1 = \text{紫} + \text{绿}$$
$$uv_2 = \text{红} + \text{红}$$
$$uv_3 = \text{黄} + \text{紫}$$
Focal Surface Parameterization

\[ D_{s,t} \]

\[ (u,v) \]

\[ r=(s,t,u,v)=(s,t,f,g)_F \]

\[ (f,g)_F \]

data cameras

camera surface \( C \)

focal surface \( F \)

Department of Computer and Information Science
Focal Surface Parameterization -2

- Intersect each new ray with the focal plane
- Back trace the ray into the data camera
- Interpolate the corresponding rays
- But need to do ray-plane intersection
- Can we avoid that?
Using

Focal Plane

Image Plane

Camera Plane

\[(u_1, v_1) \quad (u, v) \quad (u_2, v_2)\]

\[(s_1, t_1) \quad (s, t) \quad (s_2, t_2)\]

- Relative parameterization is hardware friendly
  - \([s', t']\) corresponds to a particular texture
  - \([u', v']\) corresponds to the texture coordinate
  - Focal plane can be encoded as the disparity across the light field

\[(u_i, v_i) = (u', v') + \text{disparity} \cdot (s_i - s, t_i - t), i = 1, 2, 3, 4\]
Results

Quadralinear  Focal plane at the monitor

First part in Assignment 1
Aperture Filtering

• Simple and easy to implement
• Cause aliasing (as we will see later in the frequency analysis)
• Can we blend more than 4 neighboring rays? 8? 16? 32? Or more?
• Aperture filtering
Variable Aperture

desired camera

Department of Computer and Information Science
Changing the shape and size of aperture filter
Filtering-2

Camera plane (uv)  Focal plane (st)

Film plane  Aperture
Rendering with memory coherent Ray Tracing
Small aperture size
Large aperture size
Using very large size aperture
Variable focus
Close focal surface
Distant focal surface
Issues with Light Field

• It only captures rays within a region (volume)
• It is huge (and, thus, needs compression)
• If implemented using software, a lot of memory fetches
  – Solution 1: multitexturing
  – Solution 2: using graphics hardware
Light Field Coverage

Department of Computer and Information Science
Multi-Slab Light Fields
Light Field Slabs

Naïve approach misses rays.

Solution:
Extending the light fields!

\[ l / \tan(\theta) \]
Light Field Slabs

$\theta = 45^\circ$

Department of Computer and Information Science
Hardware-based Implementation

• LF is an array of images
  – Can be very efficiently stored as textures
  – 32x32x128x128 DXT1 compression
  – Each LF is stored in a 1024x1024x16 volume
  – 6 slabs stored in a 1024x1024x128 volume => 64 MB

• Avoid branching by calculating Light Field slices in volume
  – Choose the largest direction component for each ray
  – Avoid unnecessary texture fetch
GPU-based Implementation

• Two pass PS2.0 algorithm
  – First pass: render geometry, calculate and write \((s, t, u, v)\) coordinates into the back buffer
  – Second pass:
    • Use \((s, t, u, v)\) to query the light fields from the volume texture
    • Four bilinear texture fetches and then interpolates results in shader

• Can adopt current technology

Department of Computer and Information Science
Light Field Compression

- Based on vector quantization
- Preprocessing: build a representative codebook of 4D tiles
- Each tile in lightfield represented by index
- Example: 2x2x2x2 tiles, 16 bit index = 24:1 compression
Lightfield Compression by Vector Quantization

• Advantages:
  – Fast at runtime
  – Reasonable compression

• Disadvantages:
  – Preprocessing slow
  – Manually-selected codebook size
  – Does not take advantage of structure
3D Imaging System

- The light field captures all (or almost all) necessary rays
- If we project the ray onto some image plane, we can do 3D imaging
- Autostereoscopic display
- 3D TV
Autostereoscopic Light Field

![Diagram of autostereoscopic light field](image)

- **Image**
  - Principal point
  - Lenslet

- **Left Eye**
- **Right Eye**
  - Lens Array
  - C
  - F
  - Object
3D TV

- 16 camera Horizontal Array
- Auto-Stereoscopic Display
- Efficient coding, transmission

Department of Computer and Information Science
Real-time Rendering

Department of Computer and Information Science
The Lumigraph

- Capture: move camera by hand
- Camera intrinsics assumed calibrated
- Camera pose recovered from markers
Lumigraph Postprocessing

• Obtain rough geometric model
  – Chrome keying (blue screen) to extract silhouettes
  – Octree-based space carving
• Resample images to two-plane parameterization
Lumigraph Rendering

- Use rough depth information to improve rendering quality
Lumigraph Rendering

- Use rough depth information to improve rendering quality
Lumigraph Rendering

Without using geometry

Using approximate geometry

Department of Computer and Information Science
Unstructured Lumigraph Rendering

• Further enhancement of lumigraphs: do not use two-plane parameterization
• Store original pictures: no resampling
• Hand-held camera, moved around an environment
Unstructured Lumigraph Rendering

• To reconstruct views, assign penalty to each original ray
  - Distance to desired ray, using approximate geometry
  - Resolution
  - Feather near edges of image

• Construct “camera blending field”

• Render using texture mapping
Unstructured Lumigraph Rendering

Blending field

Rendering

Department of Computer and Information Science
Other Lightfield Acquisition Devices

- Spherical motion of camera around an object
- Samples space of directions uniformly
- Second arm to move light source – measure BRDF
Other Lightfield Acquisition Devices

- Acquire an entire light field at once
- Video rates
- Integrated MPEG2 compression for each camera

(Bennett Wilburn, Michal Smulski, Mark Horowitz)
Assignment 1

• Implement a LF renderer
  – Read in an array of images (I will provide the link to the images on my computer)
  – Store the images as an array of array of pixels (of RGB)
  – A user interface to control the camera motion (forward, backward, left, right)
  – Implement the quadrilinear interpolation
  – Have another control of focal plane (disparity)
  – Have another control of aperture
  – Due March 12