Basic Light Field

ShanghaiTech University

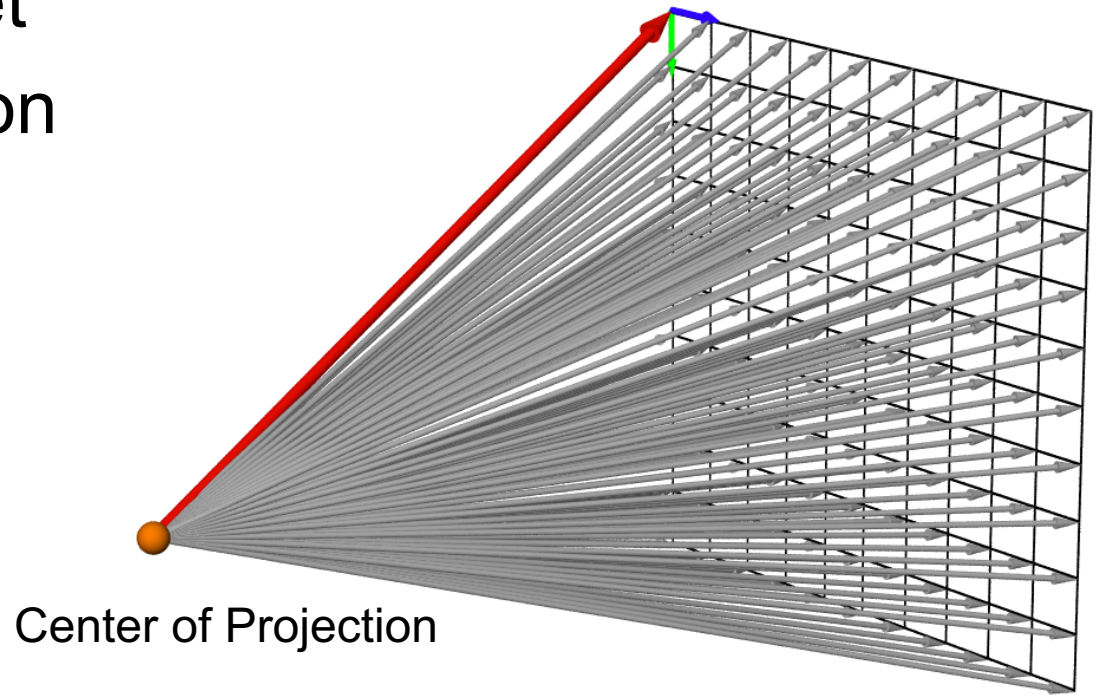
Computational Photography Fall 2020

Presented by Jiakai Zhang

Review: Pinhole Camera

- A Camera captures a set of rays
- A pinhole camera captures a set

of rays passing through a common 3D point

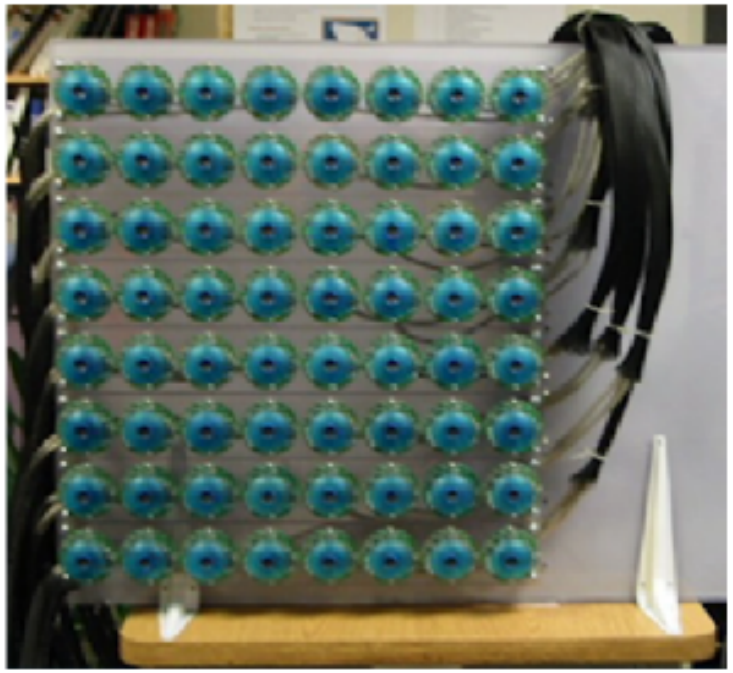


Center of Projection

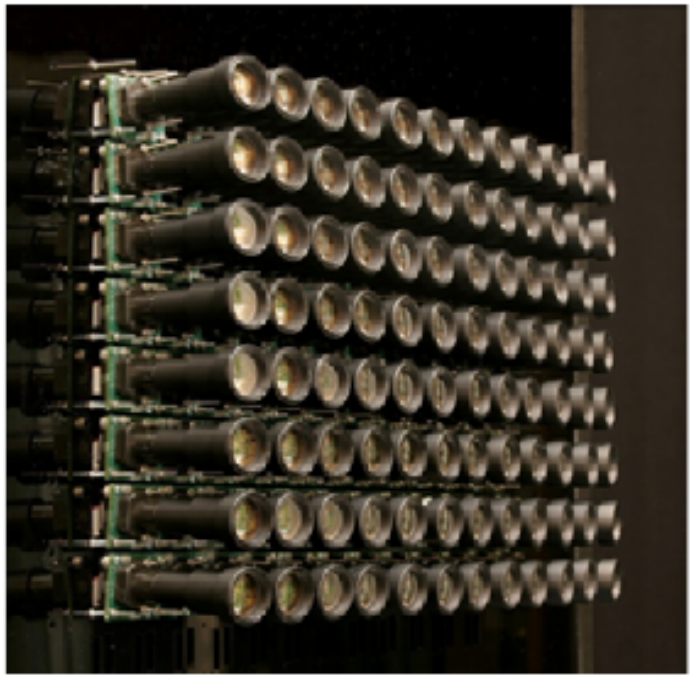
How about a Camera Array?

Capture an array of images

Capture an array of sets of rays



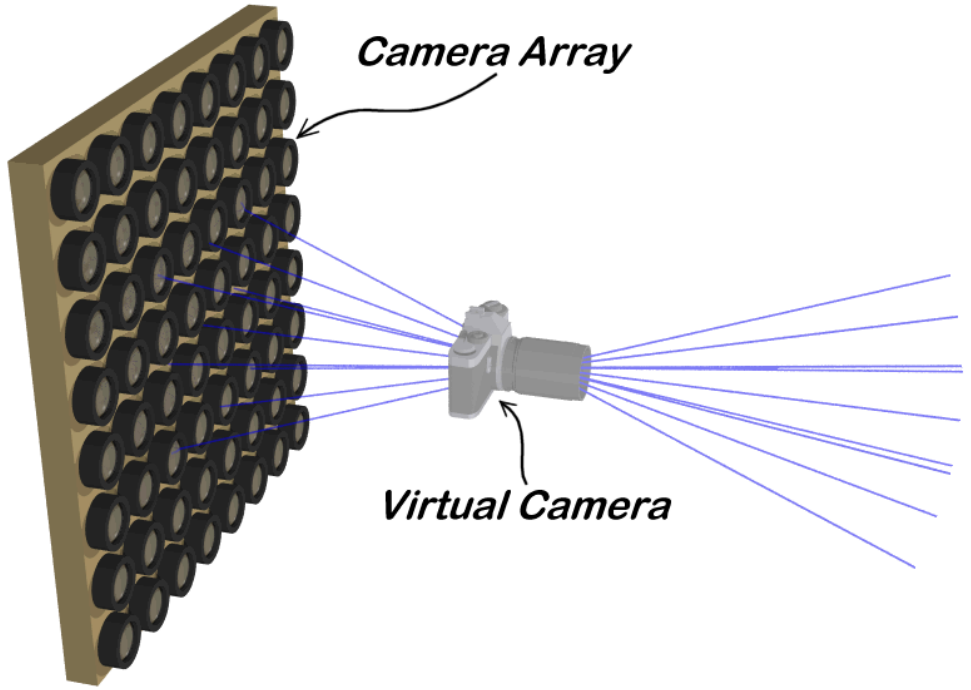
MIT Camera Array



Stanford Camera Array

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

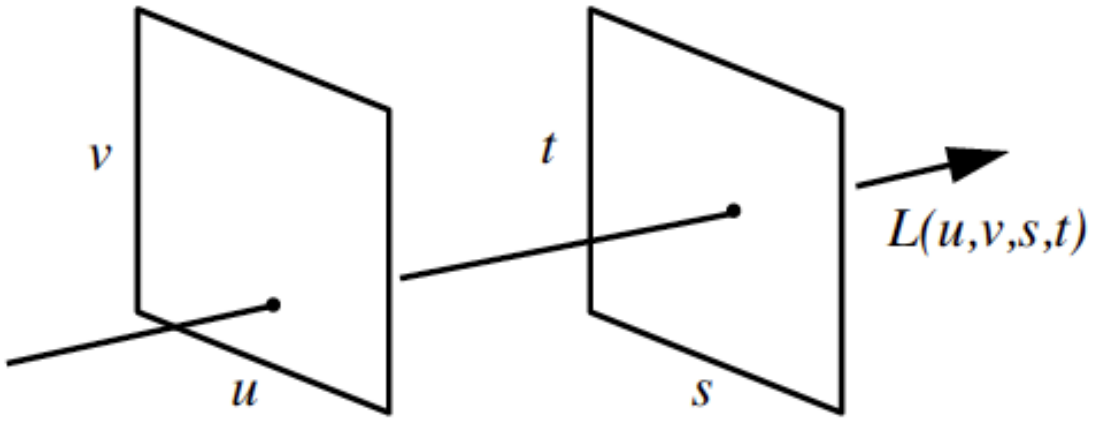


Key Problems

- 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)
  - A point and a direction(angle)(3D + 2D = 5D)
  - Any better Parameterization?

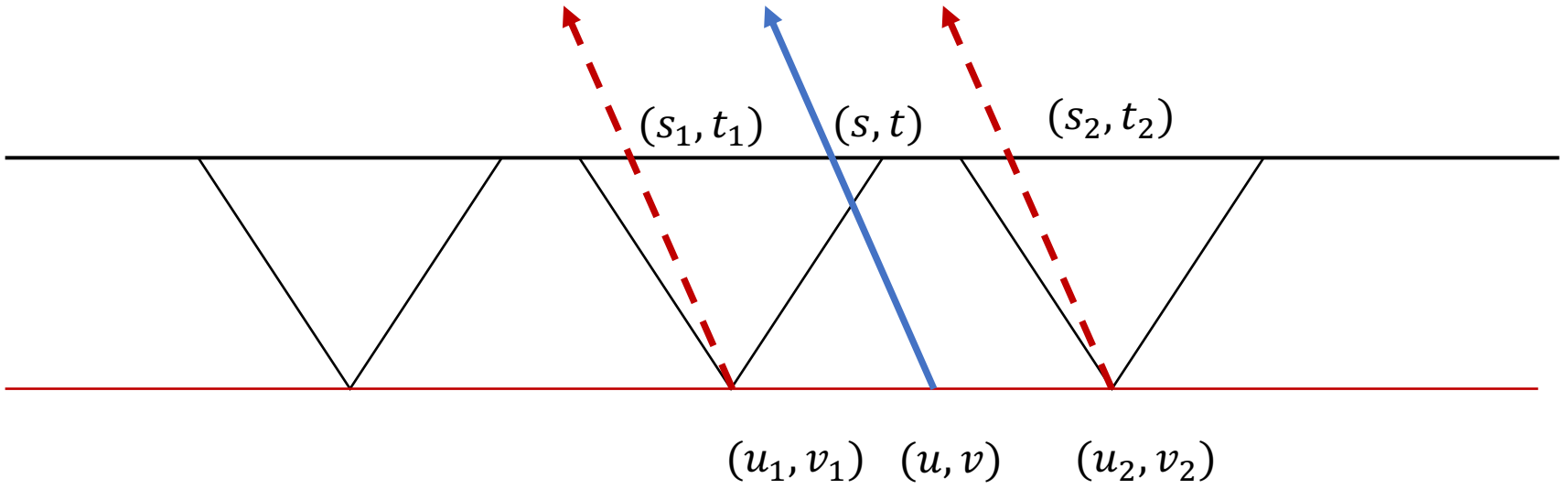
Two Plane Parameterization

- Each ray is parameterized by its two intersection points with two fixed planes.
- In camera array setting:
  - (u, v) can view as camera inde
  - (s, t) can view as pixel index
  - $\Pi_{uv}: z = 0$
  - $\Pi_{st}: z = 1$



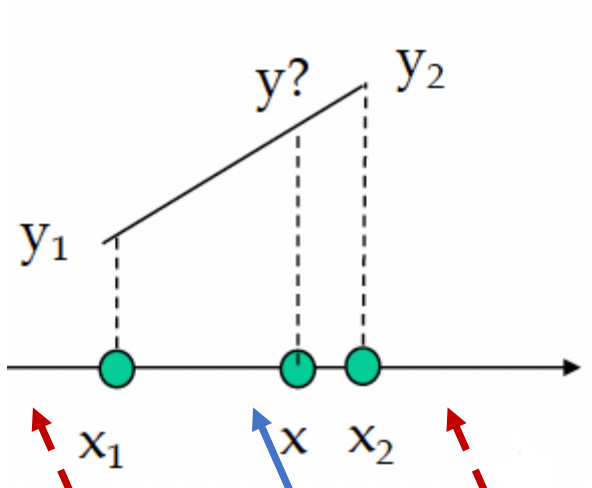
Light Field Rendering

- For each desired ray:
  - Compute intersection with (u,v) and (s,t) planes
  - Blend *closest* rays
  - What does *closest* mean?

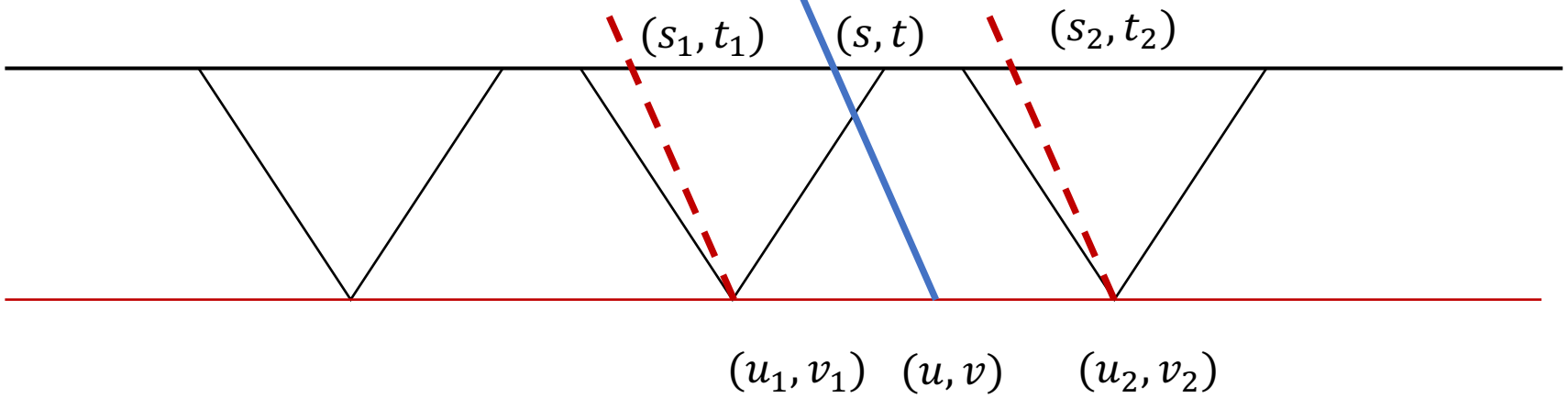


Light Field Rendering

- Blending: Linear Interpolation
- For 1D Function:



What happens to higher dimension?
Bilinear, Tri-linear, Quadra-linear

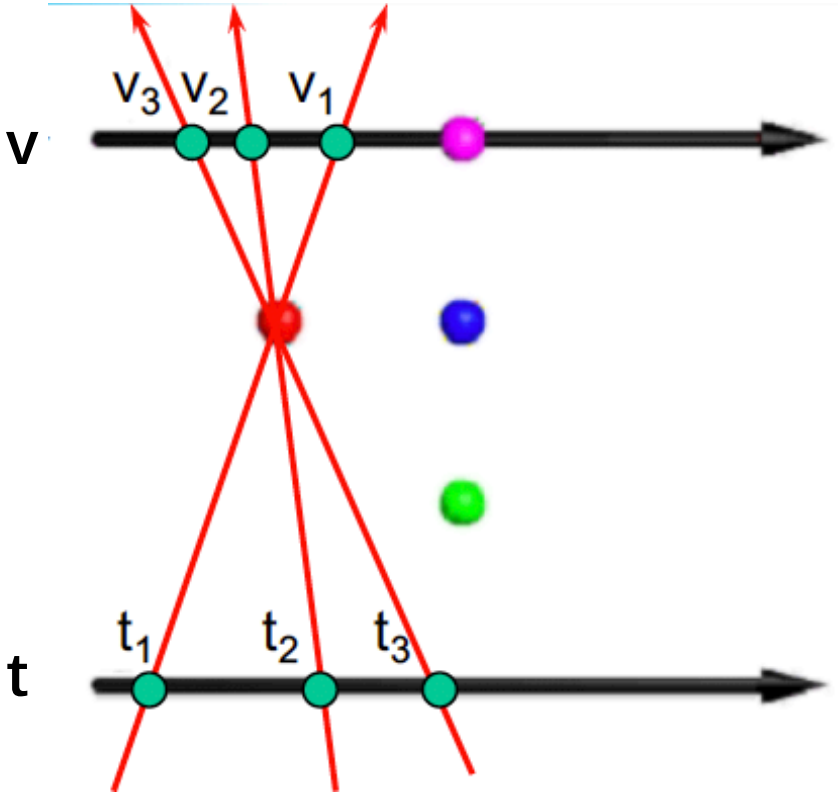


Quadra-linear Interpolation

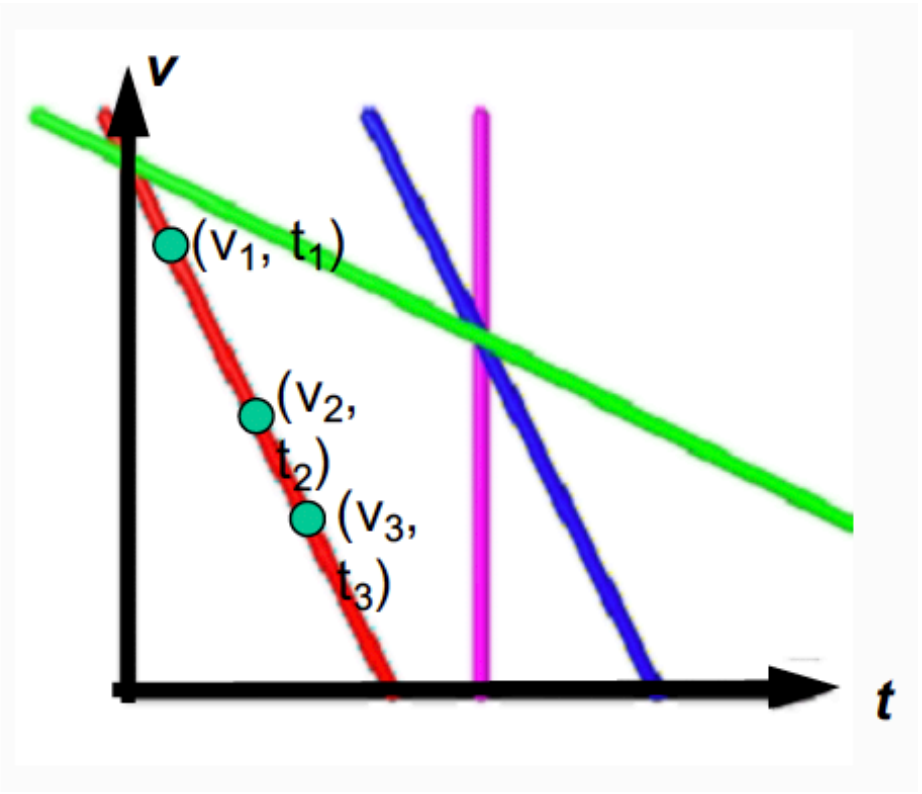
Serious aliasing artifacts

Ray Structure

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

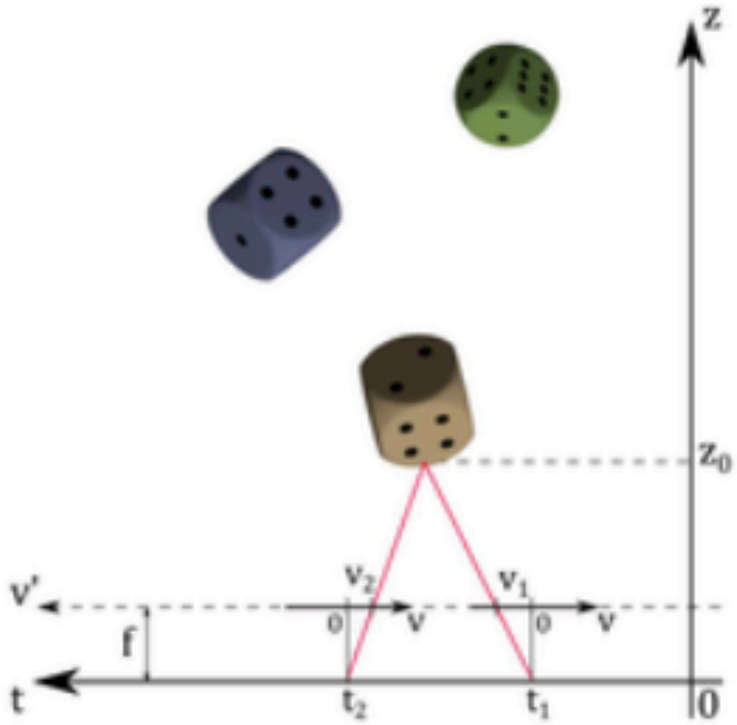


2D Light Field

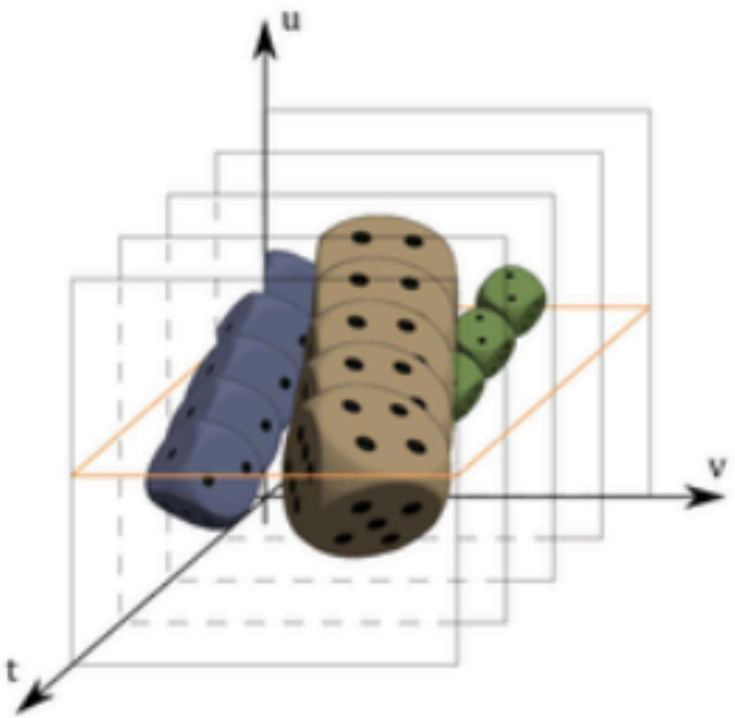


2D Epipolar-plane Image(EPI)

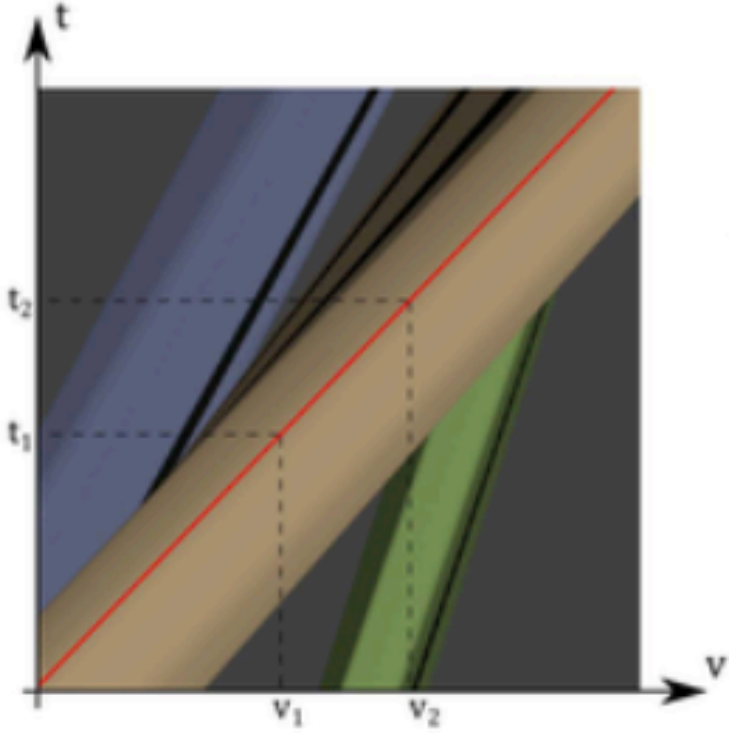
Epipolar Plane Image



t1, t2 is camera position



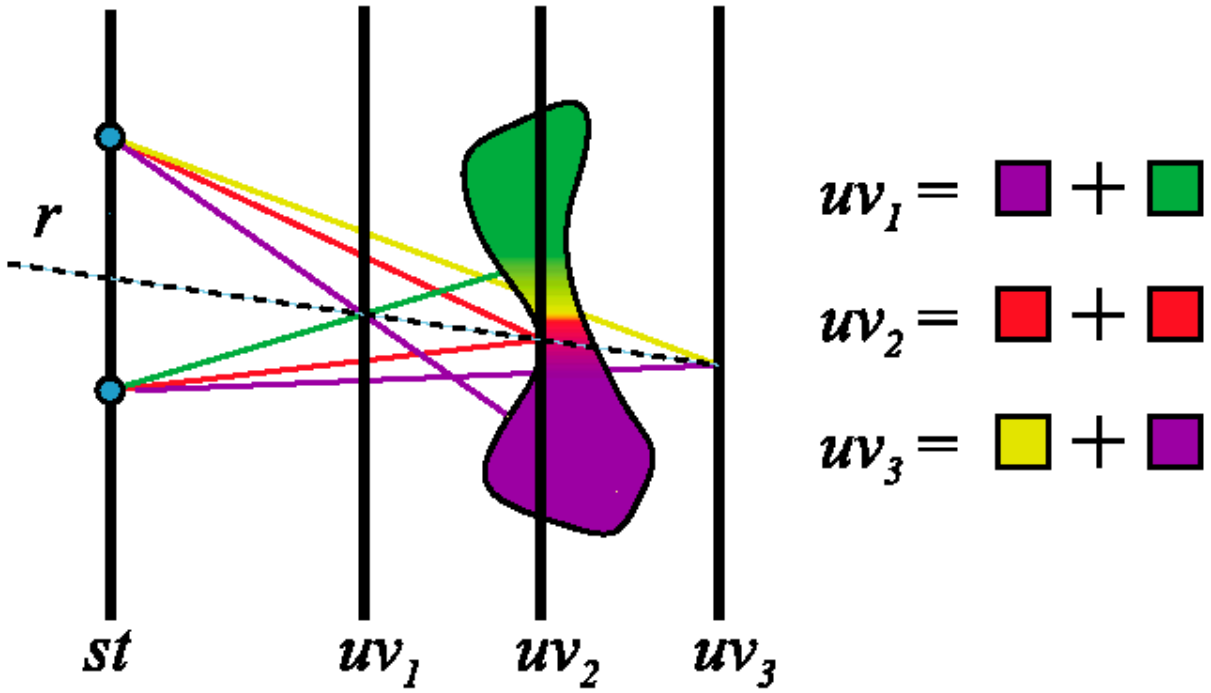
Fix u, yellow plane is epipolar plane image



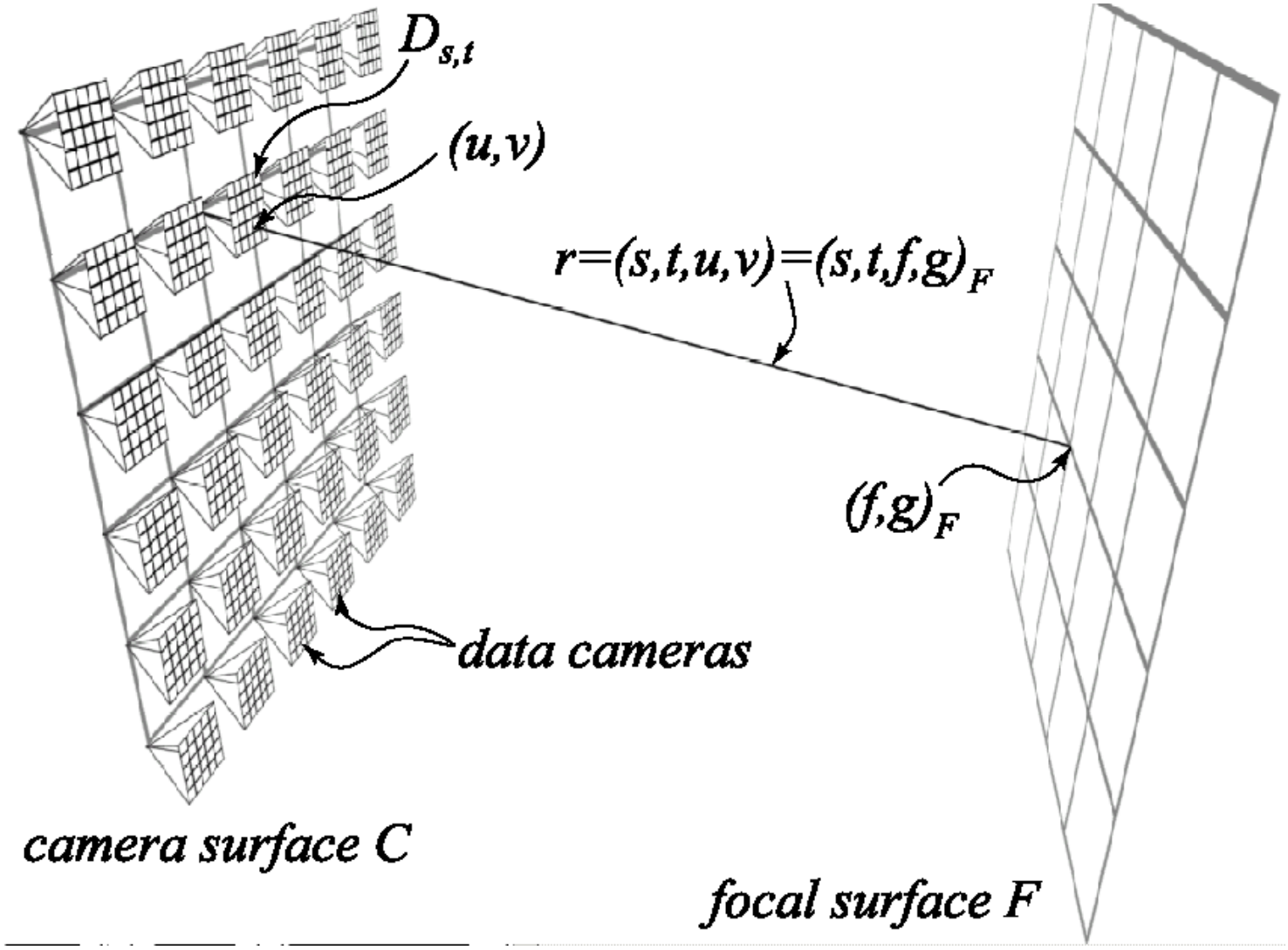
EPI

Better Reconstruction Method

- Assume some geometric proxy
- Danamically Reparametrized Light Field

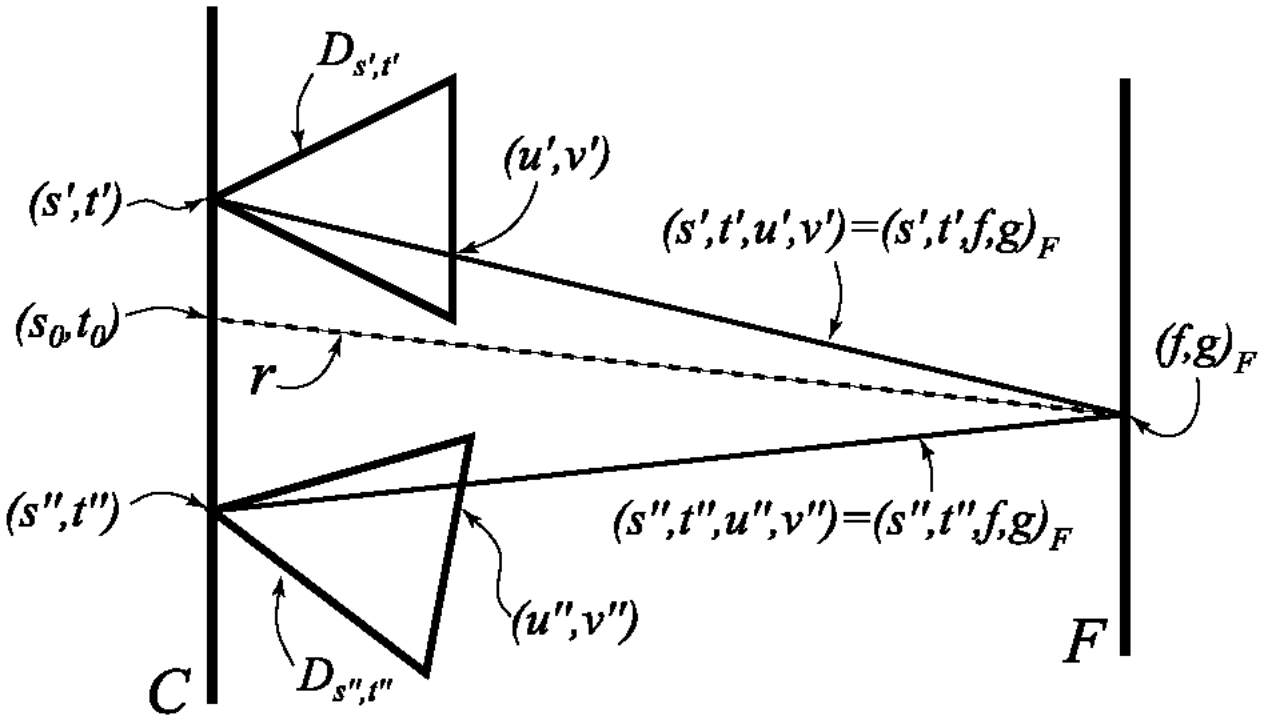


Focal Surface Parameterization



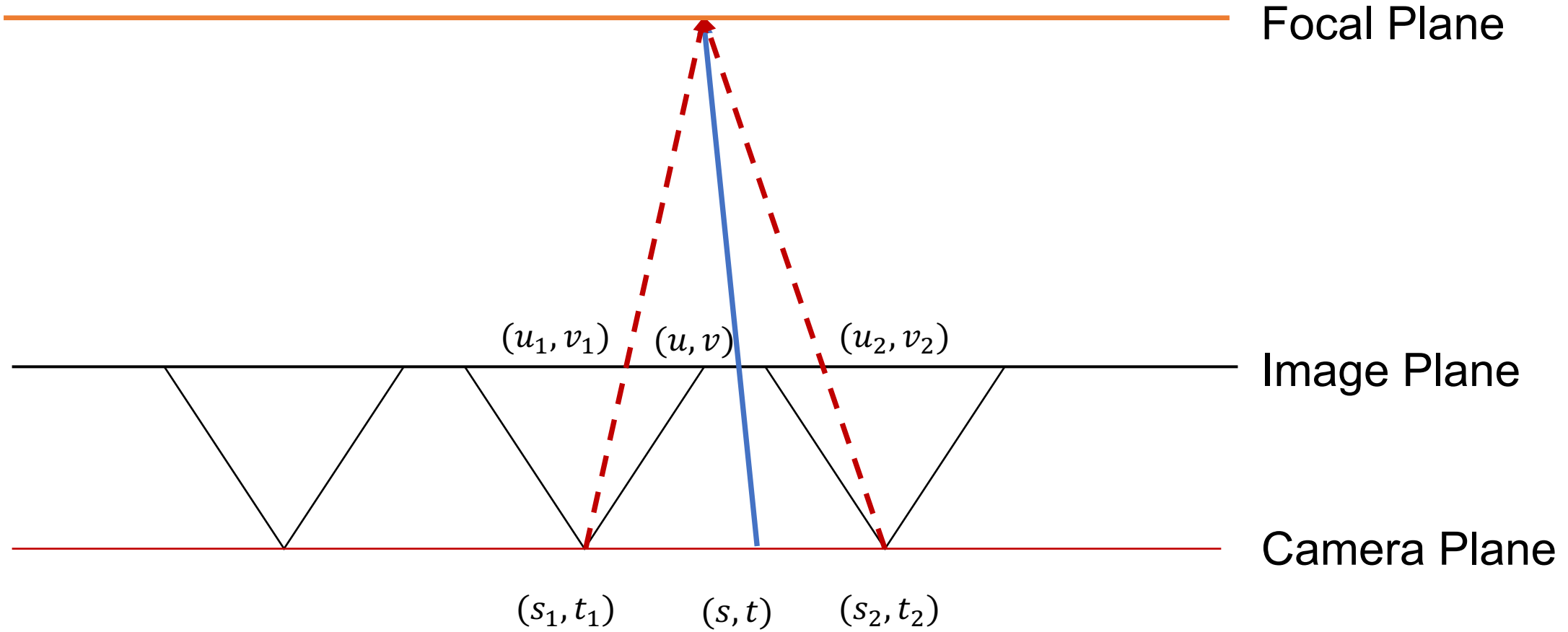
Focal Surface Parameterization

- 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

$$(u_i, v_i) = (u, v) + disparity \times (s_i - s, t_i - t), i = 1,2,3,4$$



Results

Quadra-linear

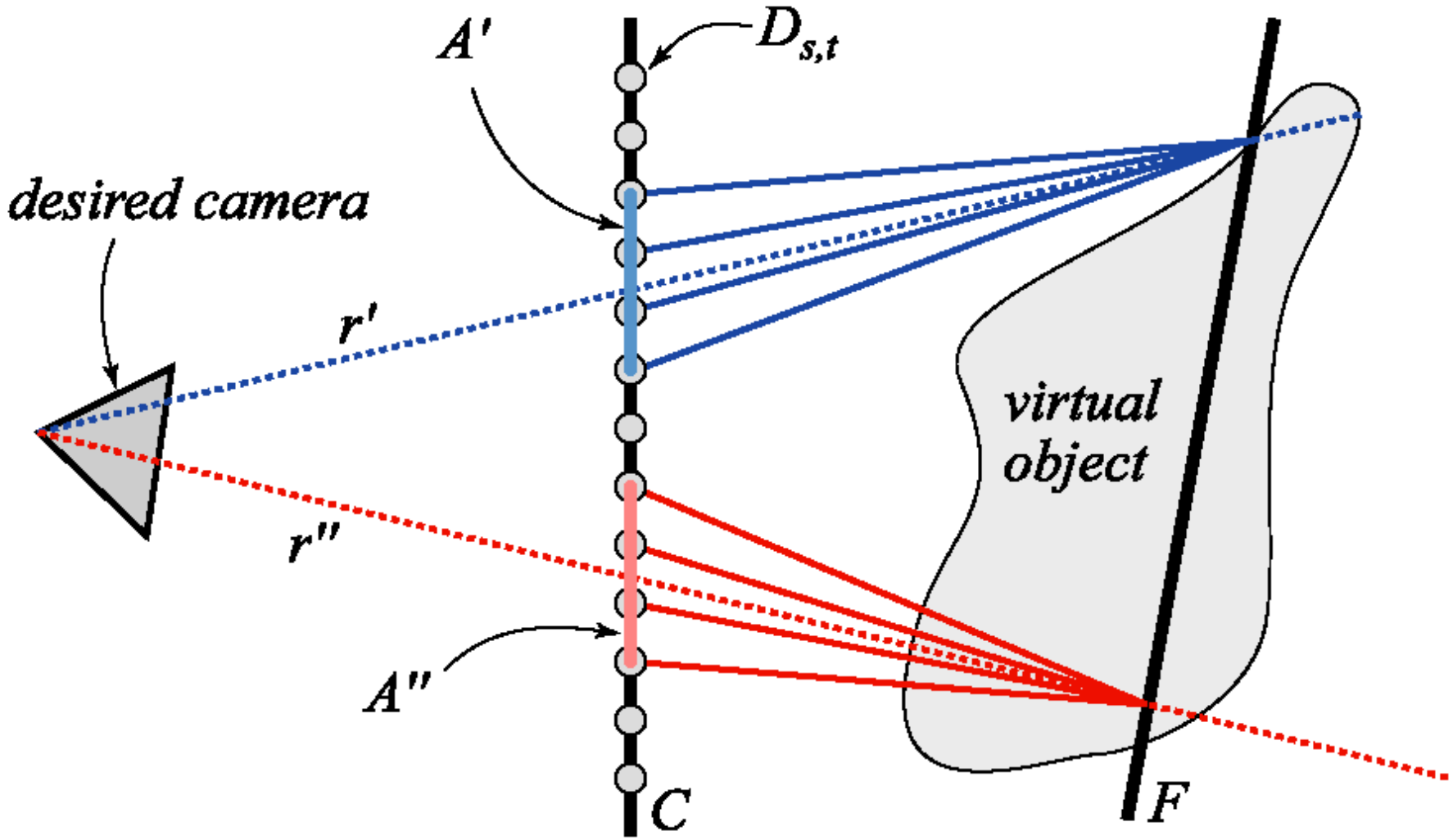
Focus on moniter

These two results are the first part in assignment 1

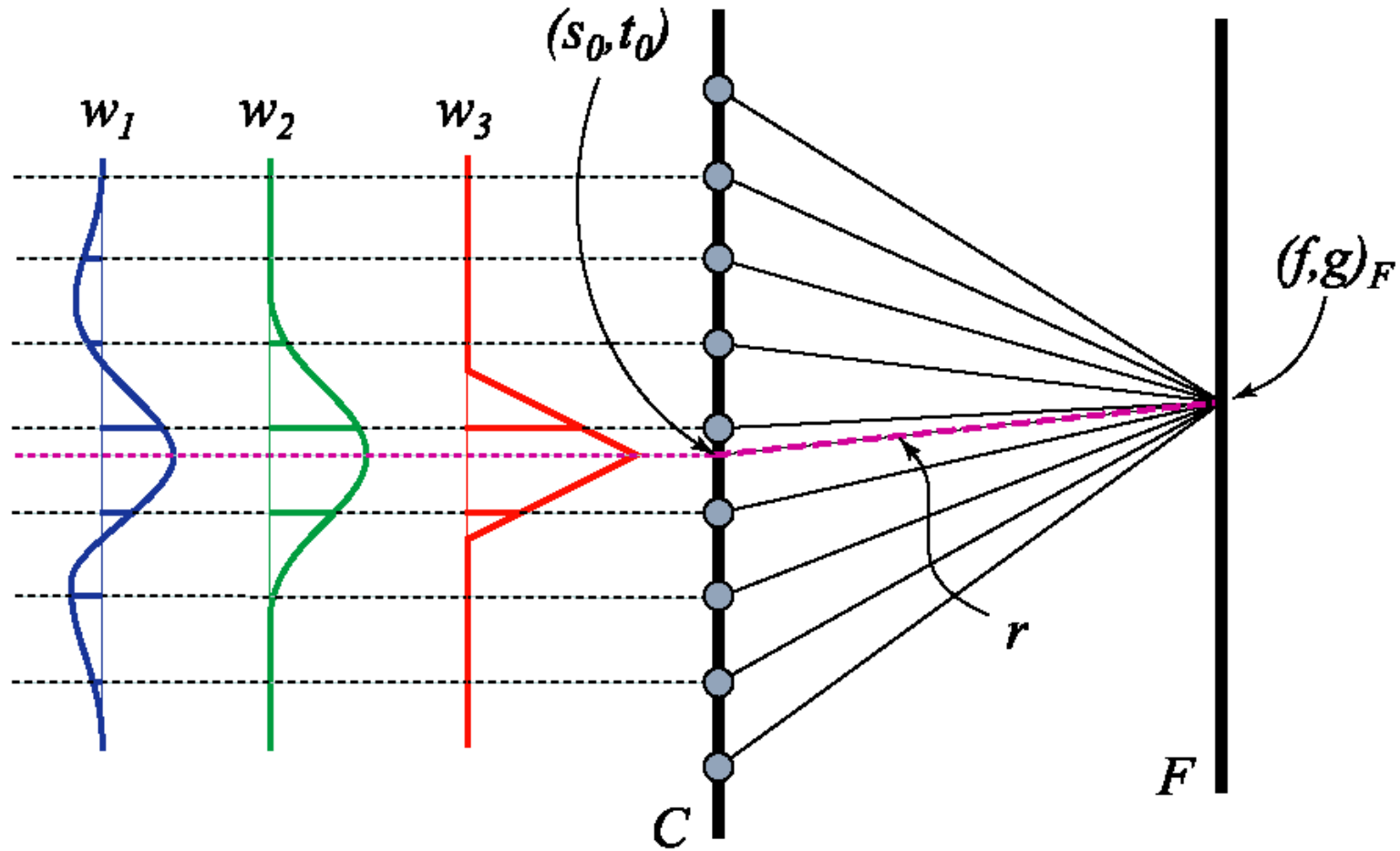
Aperture Filtering

- Simple and easy to implement
- Cause aliasing
- Can we blend more than 4 neighboring rays? 8, 16, 32? Or more?

Variable Aperture



Changing the shape & size of aperture filter

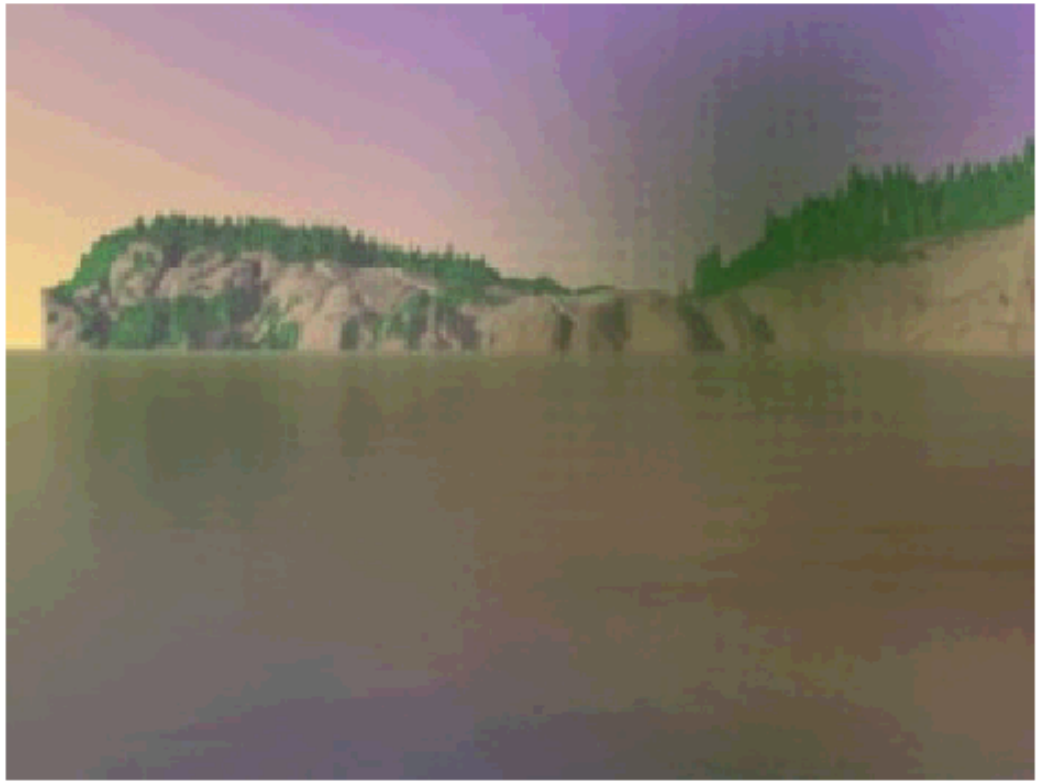


Small Aperture Size

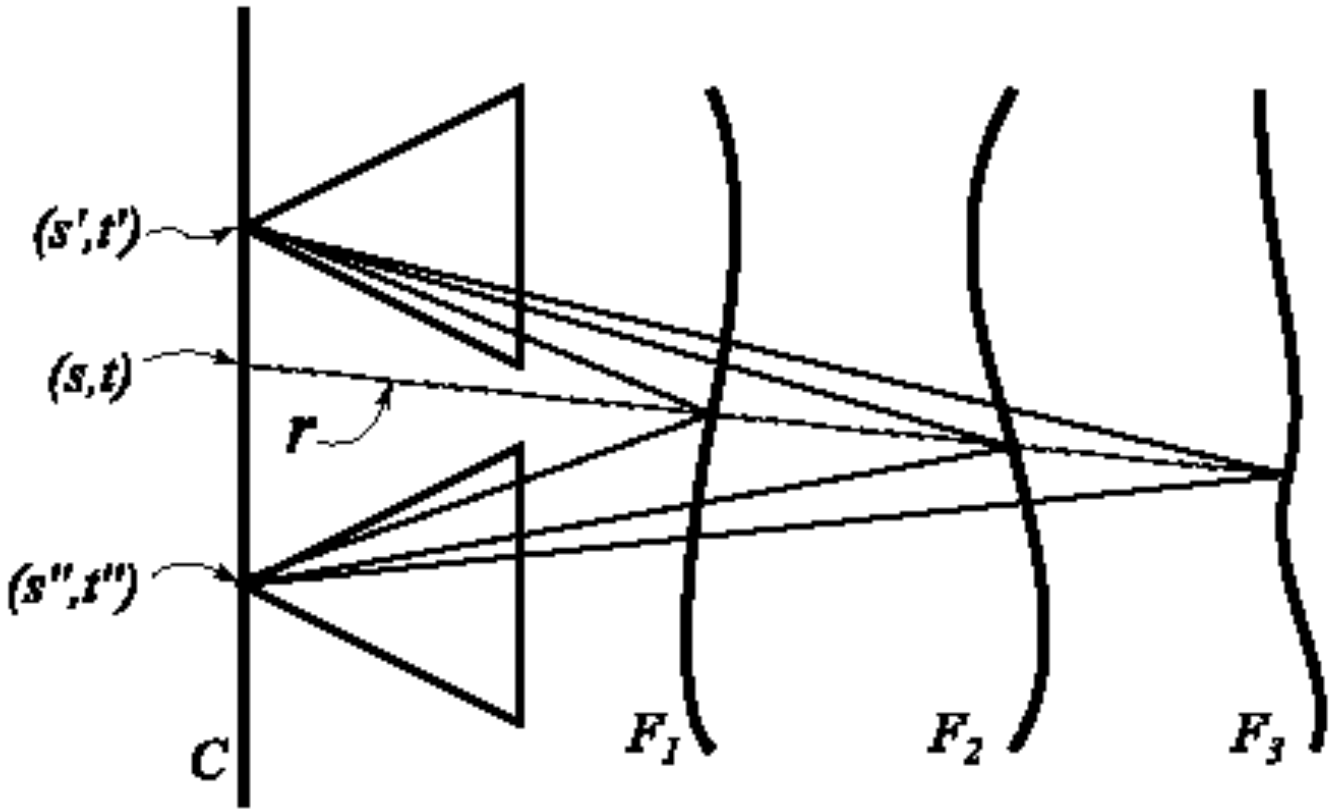
Large Aperture Size

Very Large Size Aperture

- Looking through the object



Variable Focus



Close Focal Surface

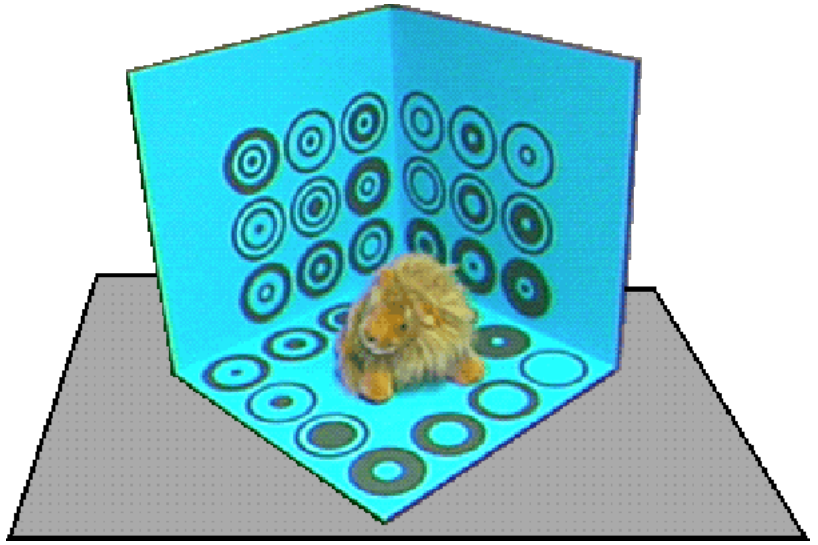
Distant Focal Surface

Issues with Light Field

- It only captures rays within a region (volume)
- It is huge（needs compression)
- If implemented using software, a lot of memory fetches
  - Solution 1: multitexturing
  - Solution 2: using graphics hardware

The Lumigraph

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



Parameterization of the 3D space

- 6 faces of the cube
- Every face use 4D parameterization (s, t, u, v) to record rays
- Discretize 4D Parameterization to M x M x N x N grids (i, j, p, q)
- Recover each rays with:

$$\tilde{L}(s,t,u,v) = \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{p=0}^{N}\sum_{q=0}^{N} x_{i,j,p,q} B_{i,j,p,q}(s,t,u,v)$$

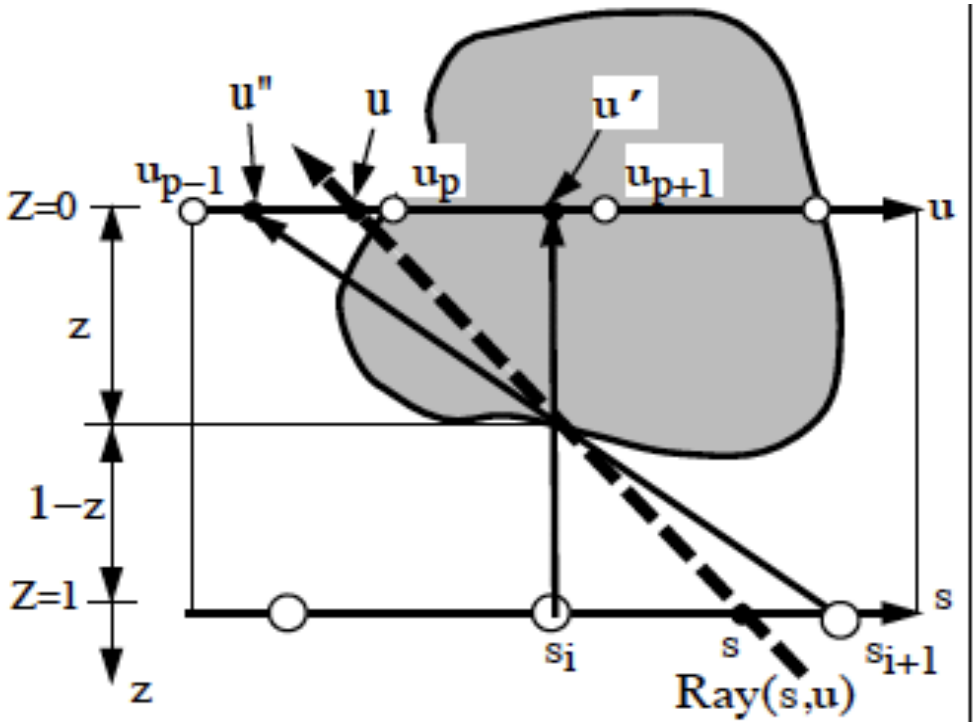
- $x: color, B: basis\ function$

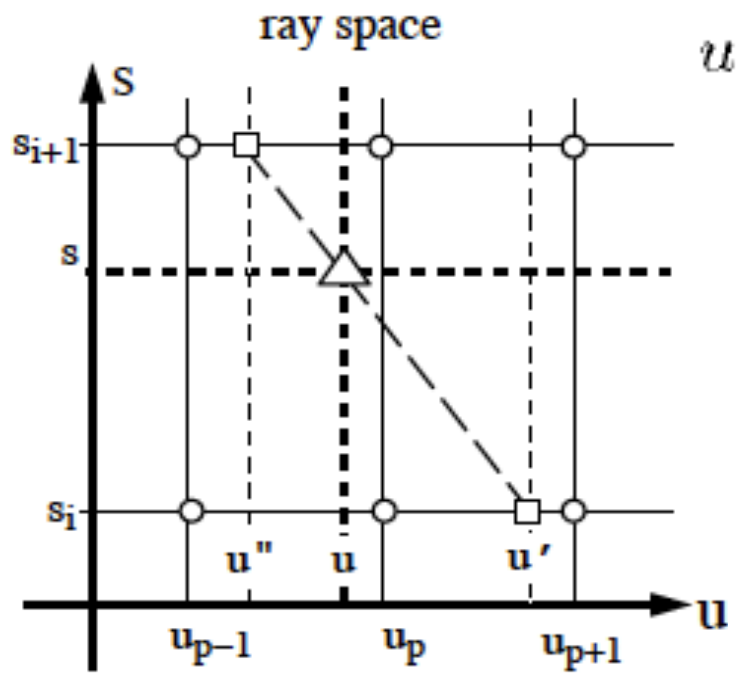
Lumigraph Postprocessing

- 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





$$u' = u + (s - s_i)\frac{z}{1-z}$$

Lumigraph Rendering

Without using geometry

Using approximate geometry

Assignment 1

- Implement a LF renderer
  - Read in an array of images
  - A user interface to control the camera motion
  - Implement the quadralinear interpolation
  - Have another control of focal plane (disparity)
  - Have another control of aperture (aperture filtering)
  - Due: Sep. 30th

Grading Policy

- 15% pre-class assignments
- 20% x 3 Coding assignments
- 25% Final Project