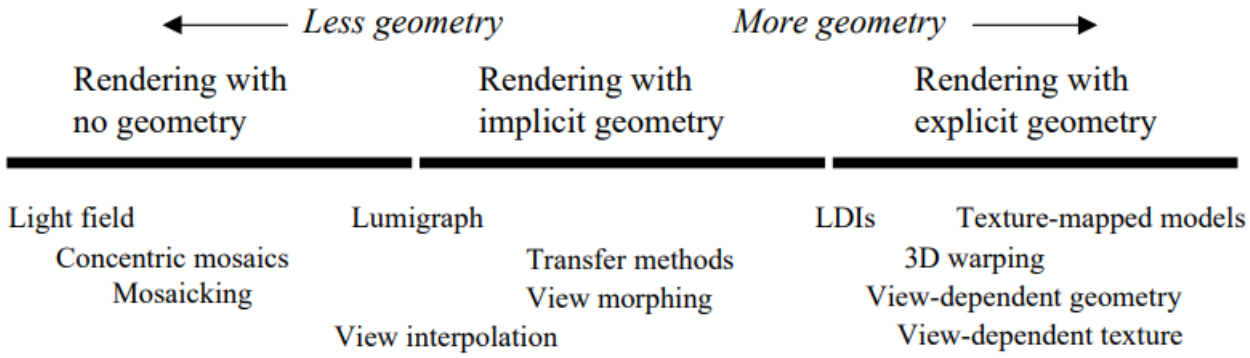
Image-Based Rendering

Unlike traditional 3D computer graphics in which 3D geometry of the scene is known, image-based rendering techniques render novel views directly from input images. Previous image-based rendering techniques can be classified into three categories according to how much geometric information is used: rendering without geometry, rendering with implicit geometry (i.e., correspondence), and rendering with explicit geometry (either with approximate or accurate geometry).



Categories in IBR

Previous work on image-based rendering (IBR) reveals a continuum of image-based representations based on the tradeoff between how many input images are needed and how much is known about the scene geometry

Rendering with No Geometry

Plenoptic Modeling

The original 7D plenoptic function is defined as the intensity of light rays passing through the camera center at every location $(V_x, V_y, V_z$ at every possible angle (θ, ϕ), for every wavelength λ, at every time t, i.e.,

$$P_7 = P(V_x, V_y, V_z, \theta, \phi, \lambda, t)$$

By dropping out two variables, time t (therefore static environment) and light wavelength λ (hence fixed lighting condition), we have the plenoptic modeling with the 5D complete plenoptic function,

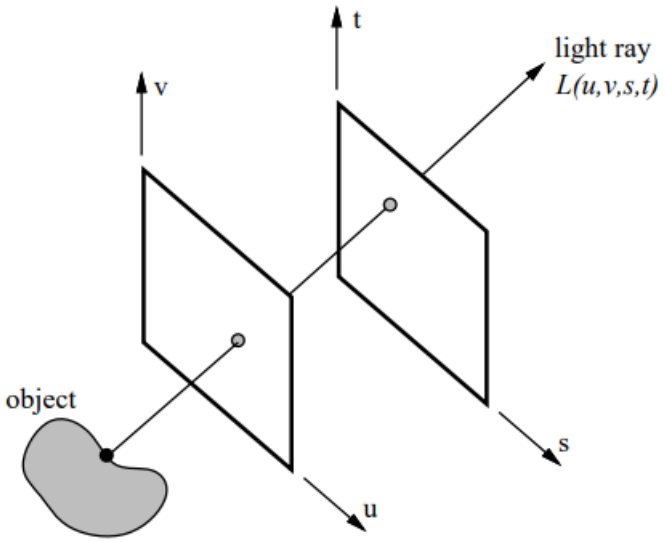
$$P_5 = P(V_x, V_y, V_z, \theta, \phi)$$

The simplest plenoptic function is a 2D panorama when the viewpoint is fixed

$$P_2 = P(\theta, \phi)$$

And a regular image(with a limited field of view) can be regarded as an incomplete plenoptic sample at a fixed viewpoint.

Light Field Rendering



Representation of a light field

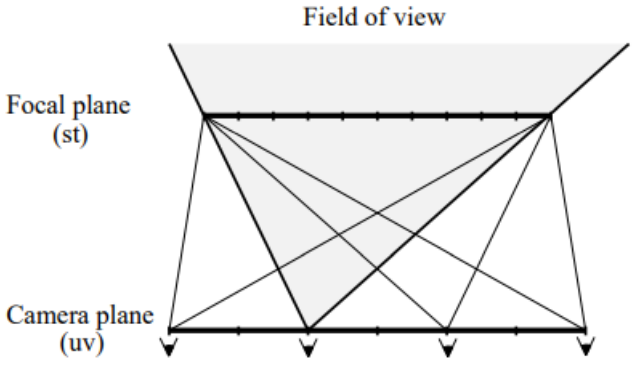
5D representation may be reduced to 4D in free space (regions free of occluders). This is a consequence of the fact that the radiance does not change along a line unless blocked. 4D light fields may be interpreted as functions on the space of oriented lines

$$P_4 = P(u, v, s, t)$$

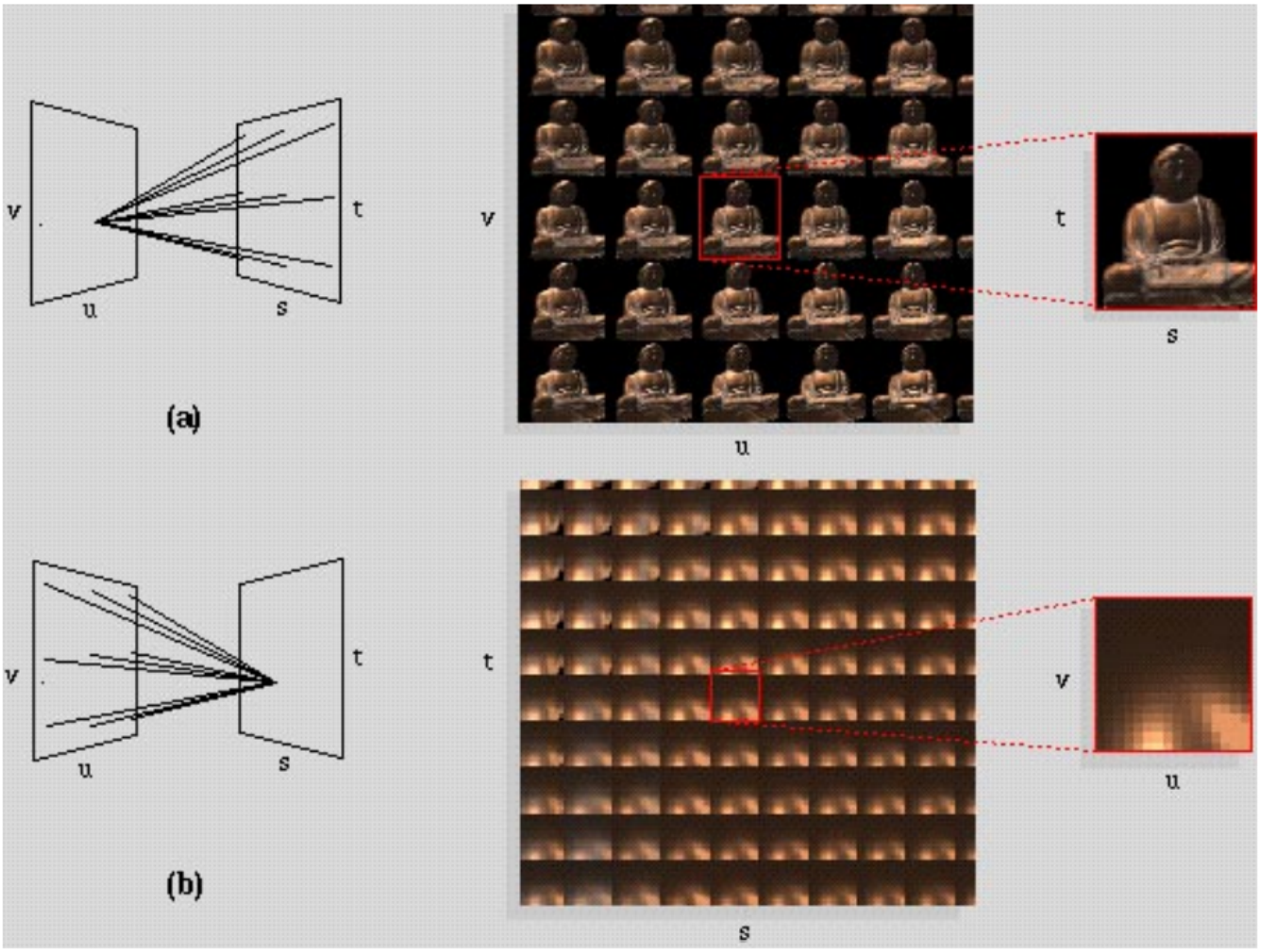
Creation of Light Fiels

From Rendered Images

For a virtual environment, a light slab is easily generated simply by rendering a 2D array of images. Each image represents a slice of the 4D light slab at a fixed uv value and is formed by placing the center of projection of the virtual camera at the sample location on the uv plane. The only issue is that the xy samples of each image must correspond exactly with the st samples. This is easily done by performing a sheared perspective projection



similar to that used to generate a stereo pair of images. The picture below shows the resulting 4D light field, which can be visualized either as a uv array of st images or as an st array of uv images.

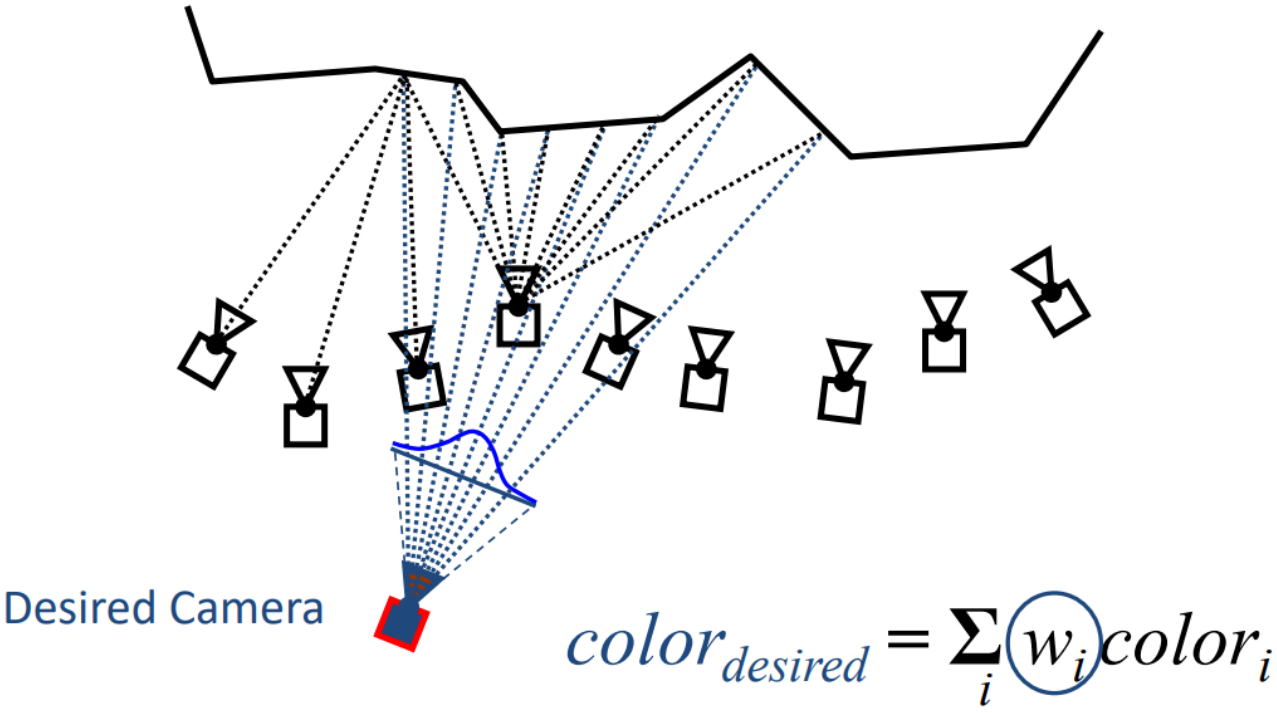


From digitized images

Digitizing the imagery required to build a light field of a physical scene is a formidable engineering problem. The number of images required is large (hundreds or thousands), so the process must be automated or at least computer-assisted. Moreover, the lighting must be controlled to insure a static light field, yet flexible enough to properly illuminate the scene, all the while staying clear of the camera to avoid unwanted shadows. Finally, real optical systems impose constraints on angle of view, focal distance, depth of field, and aperture, all of which must be managed.

Motorized camera stages, multi-camera arrays and light field cameras could be used to create a light field.

Unstructured Lumigraph



Unstructured lumigraph directly renders views from an unstructured collection of input images. The input to our algorithm is a collection of source images along with their associated camera pose estimates.

It works by evaluating a "*camera blending field*" at a set of vertices in the desired image plane and interpolating this field over the whole image. This blending field describes how each source camera is weighted to reconstruct a given pixel; the calculation of this field should be based on the stated goals, and includes factors related to ray angular difference, estimates of undersampling, and field of view.

The blending filed is constructed explicitly using penalties and we sample and interpolate over desired image.

Rendering with Implicit Geometry

There is a class of techniques that relies on positional correspondences across a small number of images to render new views. This class has the term implicit to express the fact that geometry is not directly available; 3D information is computed only using the usual projection calculations.

View Interpolation

From two input images, given dense optical flow between them, view interpolation method can reconstruct arbitrary viewpoints. This method works well when two input views are close by, so that visibility ambiguity does not pose a serious problem. Otherwise, flow fields have to be constrained so as to prevent foldovers. In addition, when two views are far apart, the overlapping parts of two images become too small.

View Morphing

From two input images, Seitz and Dyer's view morphing technique reconstructs any viewpoint on the line linking two optical centers of the original cameras. Intermediate views are exactly linear combinations of two views only if the camera motion associated with the intermediate views are perpendicular to the camera viewing direction.

Rendering with Explicit Geometry

In this class of techniques, the representation has direct 3D information encoded in it, either in the form of depth along known lines-of-sight, or 3D coordinates.

Depth-based Rendering

Forward Mapping

Forward mapping requires an image of the scene and its corresponding depth information. The technique scans the source image (from the original camera) pixel by pixel, and copies the value to the appropriate place in the destination image (for the virtual camera). The coordinate transformation is done with the use of the depth maps.

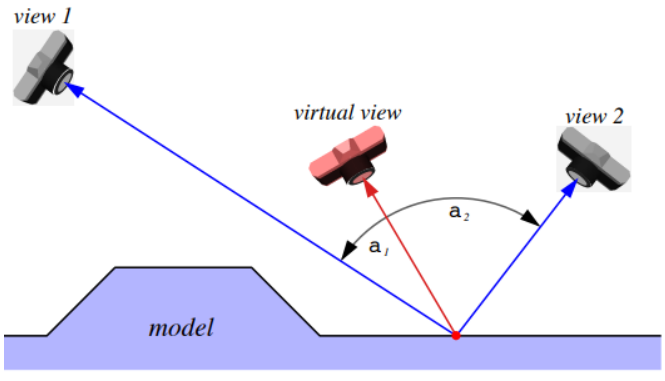
Virtual views may contain gaps due to occlusions, perspective changes (due to magnification, target discretization).

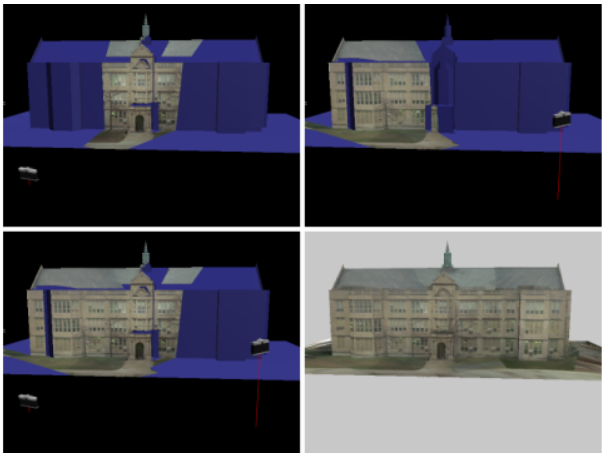
Backward Mapping

Backward mapping requires dense virtual depth maps for multiple views, including one for the virtual camera. The technique goes through the destination image pixel by pixel, and samples the correct pixel from the source image. In this way, every pixel in the destination image gets set to some value.

View-Dependent Texture Mapping

1. Determine visible cameras for each surface element
2. Blend textures (images) depending on distance between original camera and novel viewpoint





References

Light Field Rendering(acm.org)
Unstructured Lumigraph Rendering (harvard.edu)
An overview of free viewpoint Depth-Image-Based Rendering (DIBR) (apsipa.org)
A review on image-based rendering - ScienceDirect
A Review of Image-based Rendering Techniques