Image-based Rendering and Interpolation

New Images From Old

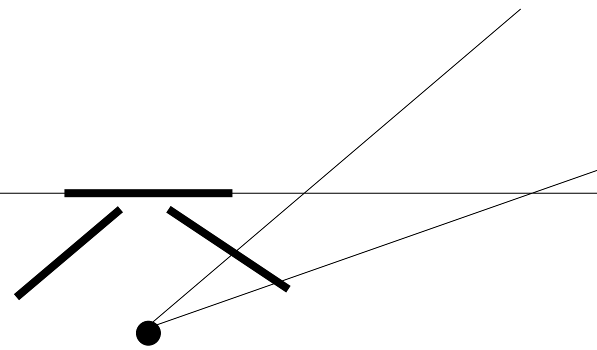
- Transfer Rays of Light from One Image to Another
  - Rectification and mosaicing
  - Light Fields
- Interpolate missing pixels
  - Interpolate in the image
  - Interpolate light rays

Light-Field Rendering

- Sample the set of light rays in the world.
- Then generate an image by selecting the right rays.
- Mosaicing: simpler, just sample rays through one focal point.
- If one has all rays then camera can also move.

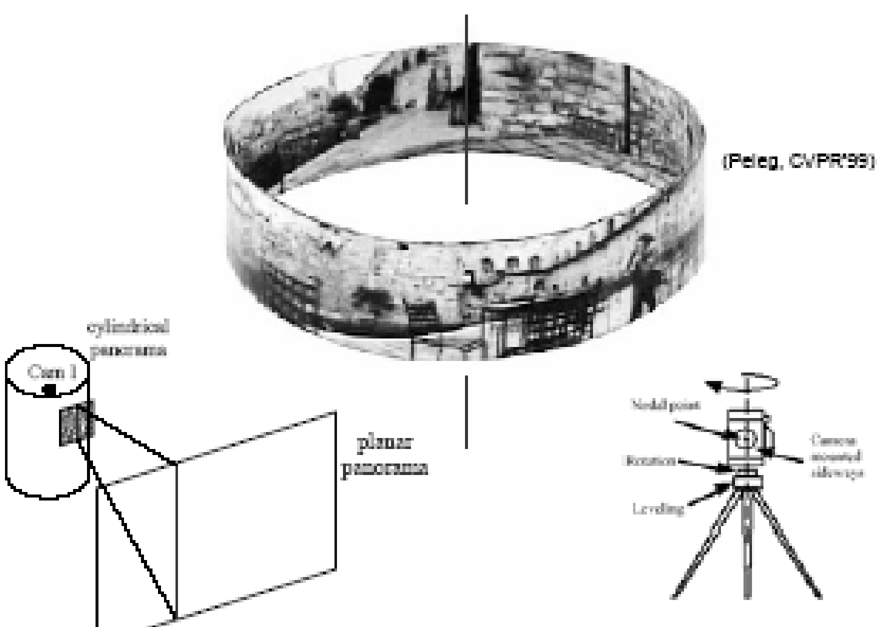
Mosaics

- Take multiple images and construct one big image.
  - Represented as image, cylinder or sphere.
- Allows panning and zooming.
  - Simplest kind of motion.



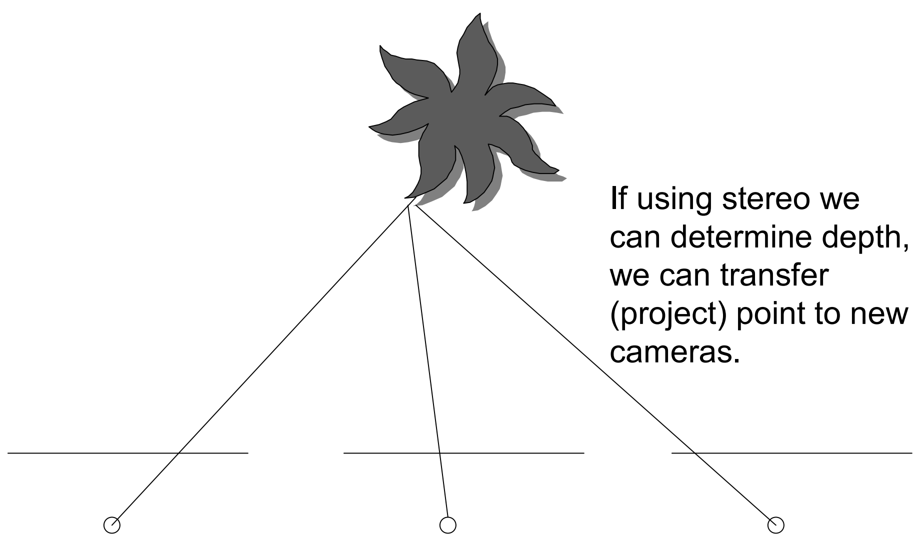
- Fixed focal point.
- Correspondence needed to align images.
- Image rectification

Cylindrical Panoramas for 360° Viewing



Cameras with different focal points

- Stereo Transfer



If using stereo we can determine depth, we can transfer (project) point to new cameras.

Stereo Transfer

- Note that we can just interpolate disparity to determine where a point will appear in a new image, providing focal points are all collinear.
  - Given Z and f, d and T are linearly related.

$$Z = f\frac{T}{d}$$

Linear Interpolation

- Given a function defined at two points, f(0), f(1), we want to find values for intermediate points, eg., f(x), 0 < x < 1.
- Can take weighted average:

f(x) = (1-x)*f(0) + x*f(1) = f(0) + x(f(1)-f(0))

- This is equation for line with slope f(1)-f(0).

Bilinear Interpolation – 4 points

- Given values at (0,0), (1,0), (0,1), (1,1) find value at (x,y).
- Linearly interpolate (x,0), (x,1), then interpolate (x,y).
- Or, find (0,y) and (1,y) and interpolate.
- These produce same results.

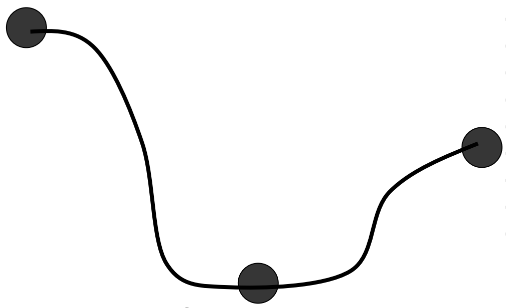
If we interpolate to get f(x,0) = (1-x)f(0,0)+xf(1,0), f(x,1) = (1-x)f(0,1) + xf(1,1). Then f(x,y) = ((1-x)f(0,0)+xf(1,0))(1-y) + (f(x,1) = (1-x)f(0,1) + xf(1,1))y.

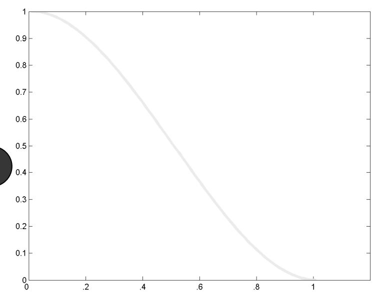
If we interpolate to get f(0,y) = (1-y)f(0,0) + yf(0,1), f(1,y) = (1-y)f(1,0) + y(f(1,1). Then

f(x,y) = ((1-y)f(0,0) + yf(0,1))(1-x) + ((1-y)f(1,0)+y(f(1,1)))x

These are the same.

Cubic Interpolation





Instead of weighting by distance, d, weight by:

$1 - 3d^2 + 2|d|^3$

- Smooth
- Symmetric

Suppose 0<= x <= 1, and a function f is defined on f(0), f(1). We want to define it for f(x) so that f(x) is smooth.
If we do this by averaging neighbors, we have:
f(x) = g(x)*f(0) + g(1-x)f(1). Then we want a function g that is smooth, and in which g(0) = 1 and g(1) = 0, and in which g is symmetric so that g(x) + g(1-x) = 1.

With linear interpolation g(x) = 1-x. This fits the second two criteria, but this g is not smooth. There is a discontinuity at f(0), since we suddenly switch between averaging f(0) and f(1) and averaging f(0) and f(-1)

So instead, we want f(x) near f(0) to be based mostly on the value of f(0), and only to gradually average in f(1) as we get closer to it.

A nice function that does this is 1 – 3*d*d +2*|d*d*d|

Note that g(1-x) = 1 – 3*(1-x)(1-x) +2(1-x)(1-x)(1-x)

= 1 – 3 + 6x –3x*x + 2 – 6x + 6x*x – 2x*x*x = 3x*x – 2*x*x*x

= 1 – (1 – 3x*x + 2x*x*x)

Also, we can see that when x -> 0, g(x) -> 1 – 3x*x + 2*x*x*x -> 1, and that g(1-x) similarly goes to 0. This means that (g(x)f(0) + g(1-x)f(1))-f(0) / x -> 0, which shows that the tangent at f(0) on the right side of the curve is 0. Similarly, the tangent on the other side is also zero, so two interpolating curves meet at x=0 with the same tangent, ie., smoothly.

Application: Image Resizing

- When we enlarge an image, we need values for the new pixels.
- Common methods:
  - Nearest neighbor
  - Bilinear interpolation
  - Bicubic interpolation




Nearest
~.007 secs


Bilinear
~.43 secs


Bicubic
~ .75 secs