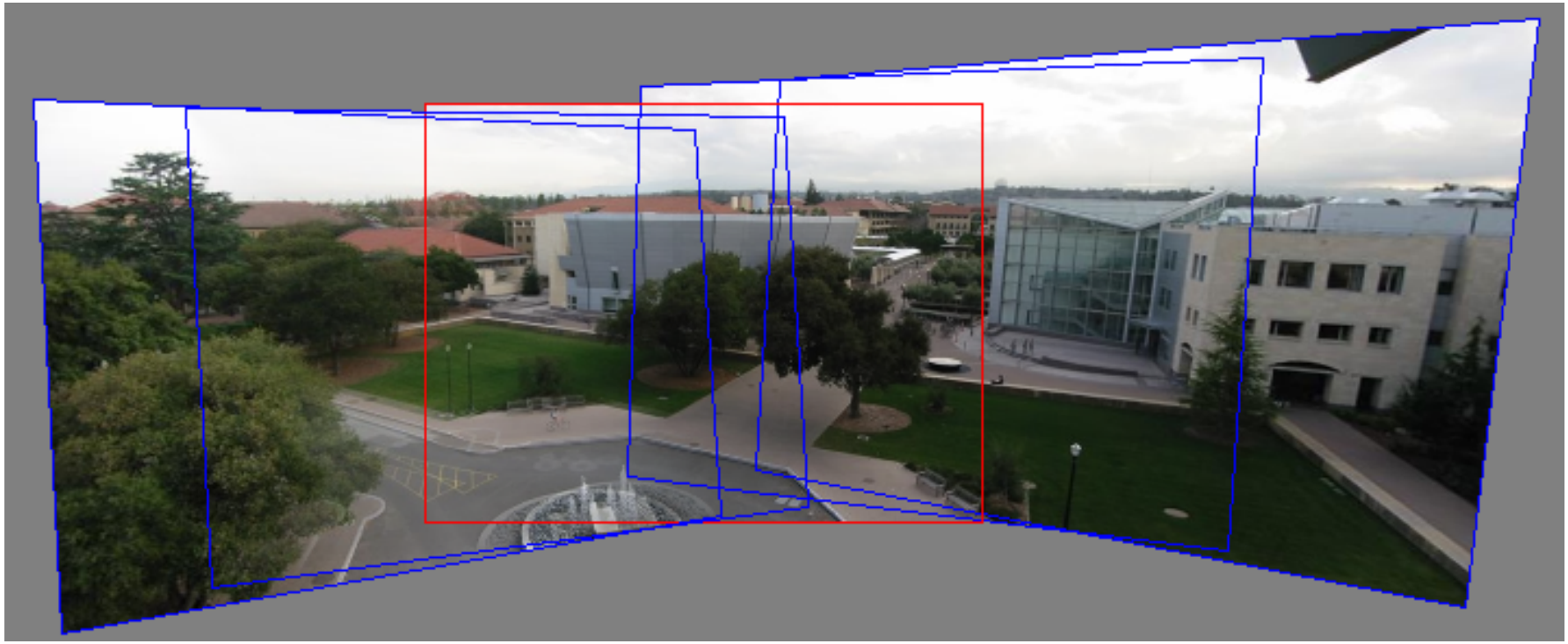
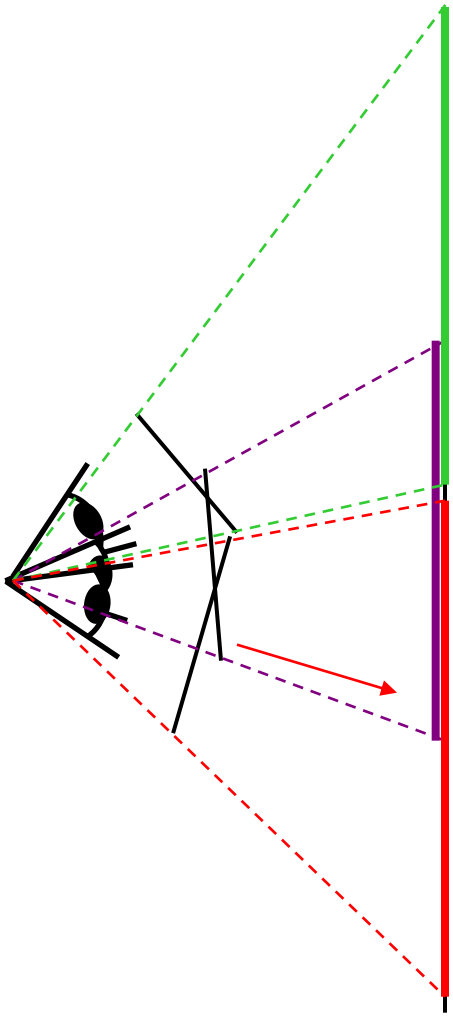
Panoramas

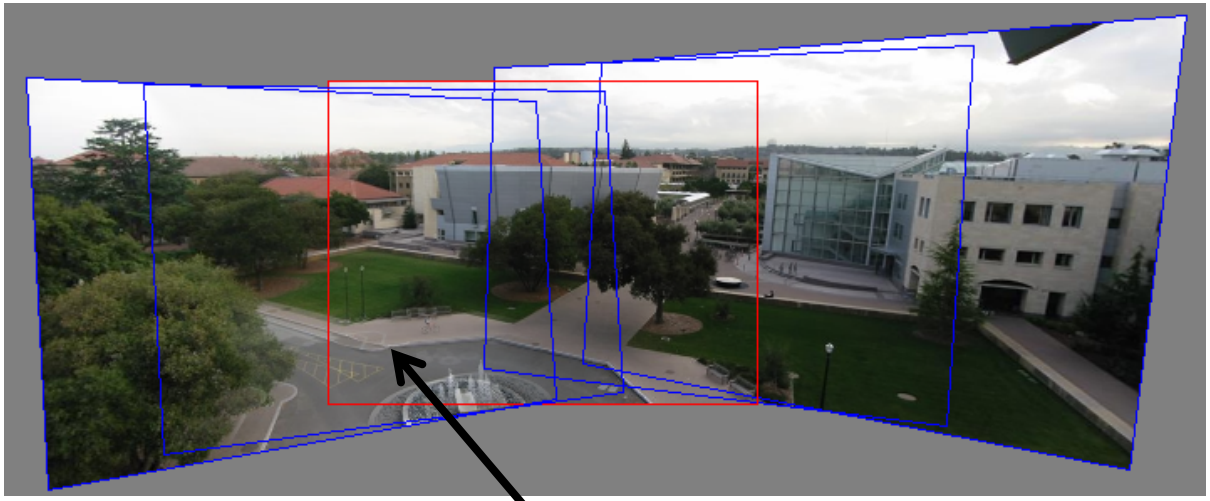


We can’t use homographies to create a 360 degree panorama.

Homographies project images onto a common plane.



mosaic **Projection Plane**



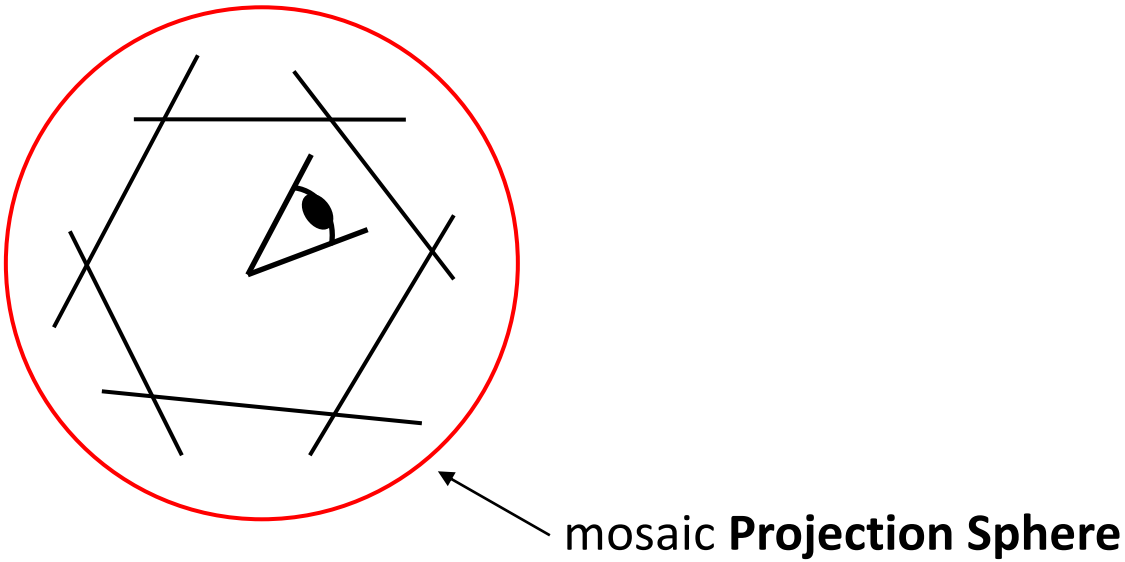
each image is warped
with a homography $\mathbf{H}$

We'll see what this homography means in terms of cameras and projection later.

First -- Can't create a 360 panorama this way…

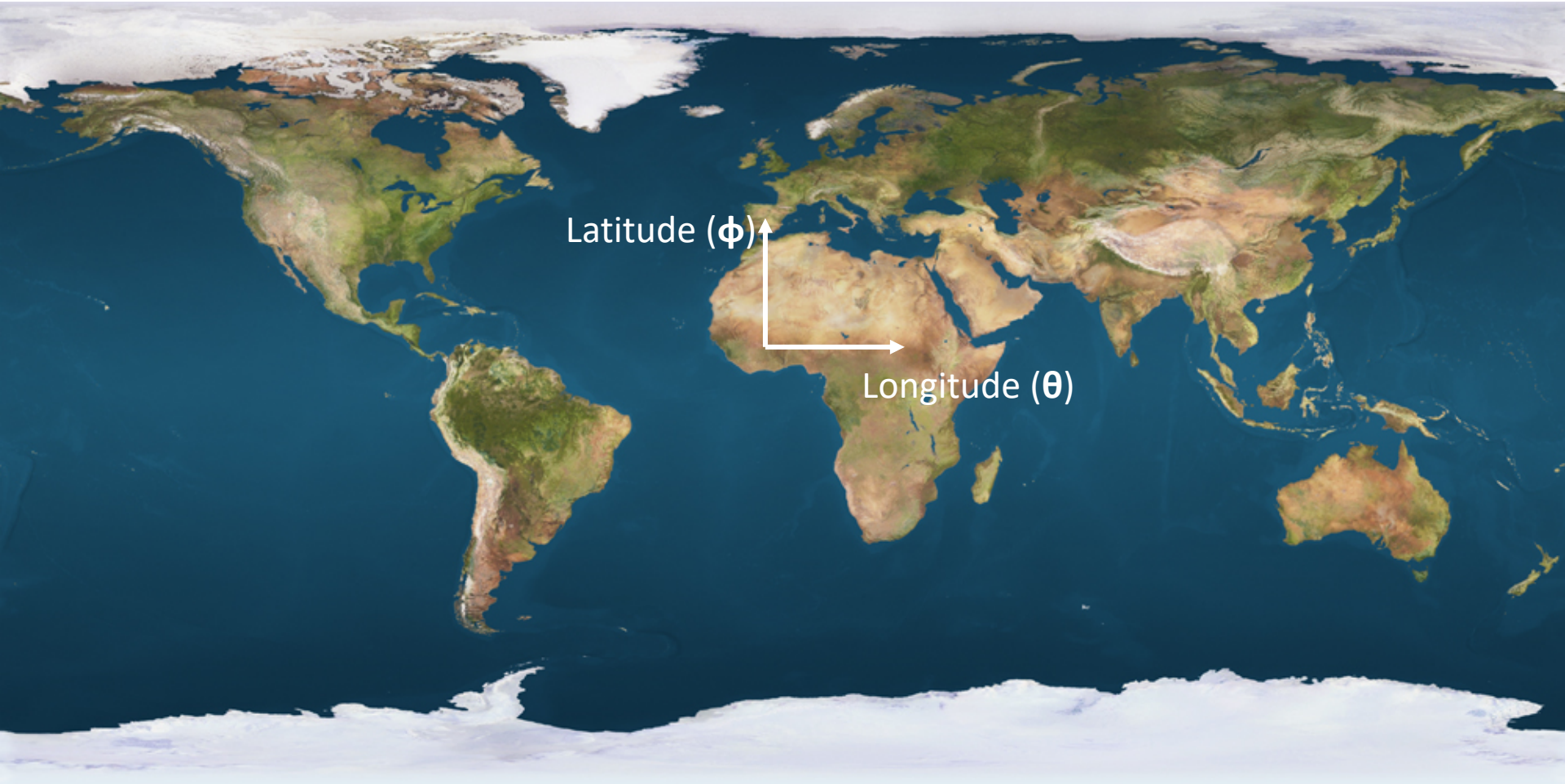
Panoramas

- What if you want a 360° field of view?

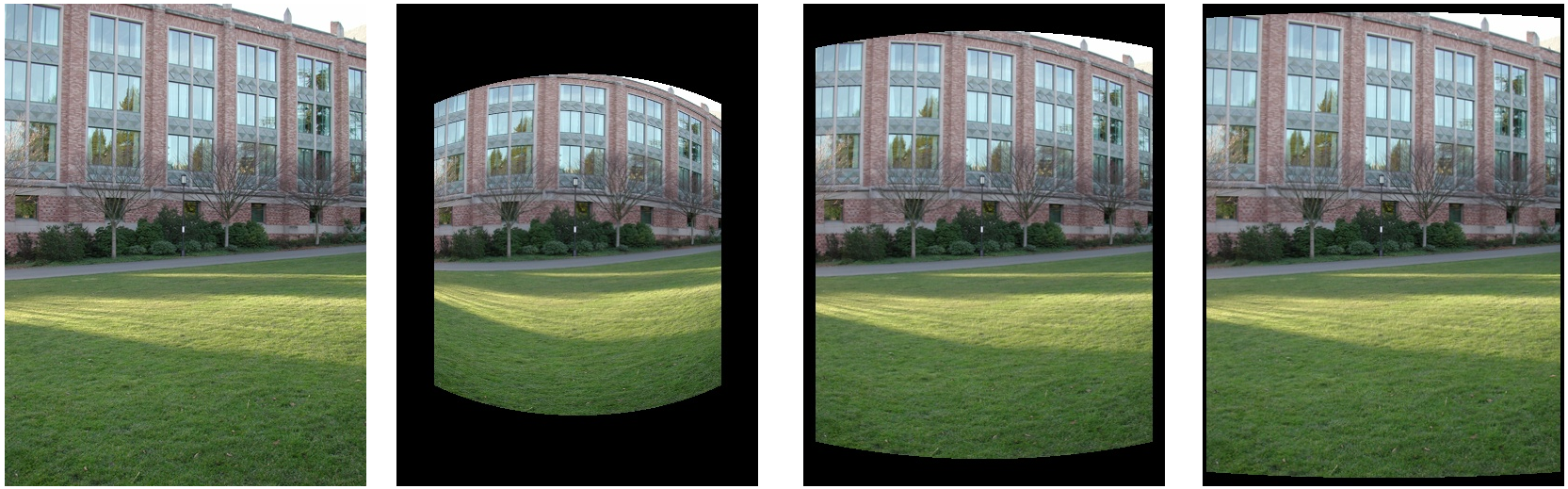


Unwrapping a sphere

Credit: JHT's Planetary Pixel Emporium



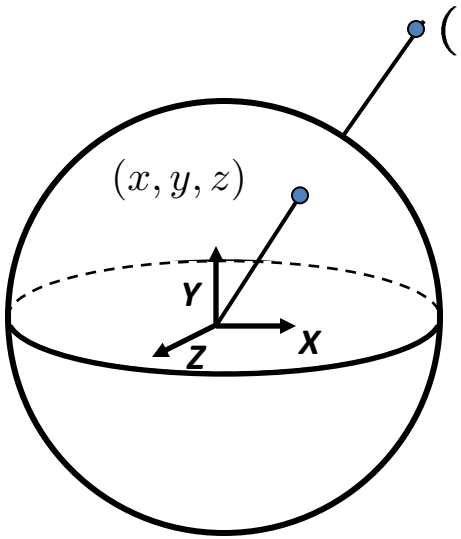
Spherical reprojection



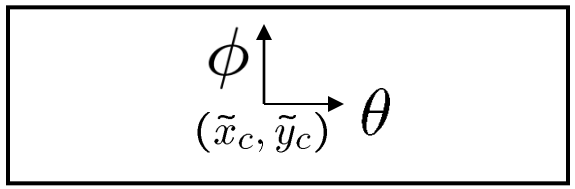
input **f = 200 (pixels)** **f = 400** **f = 800**

- Map image to spherical coordinates
  - need to know the focal length

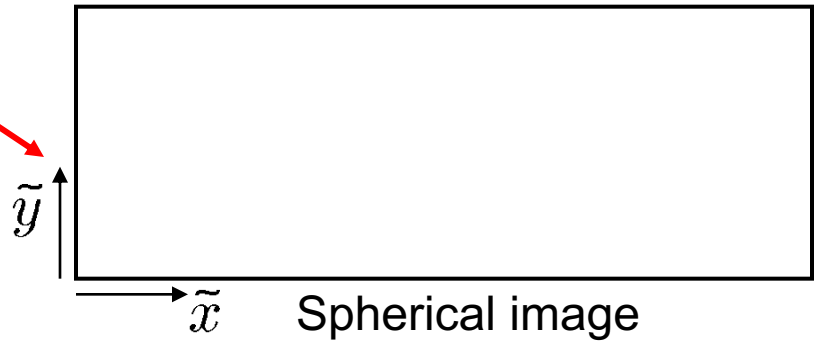
Spherical projection: Forward process



unit sphere



unwrapped sphere



Spherical image

- Map 3D point (X,Y,Z) onto sphere

$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Y^2+Z^2}}(X, Y, Z)$$

- Convert to spherical coordinates

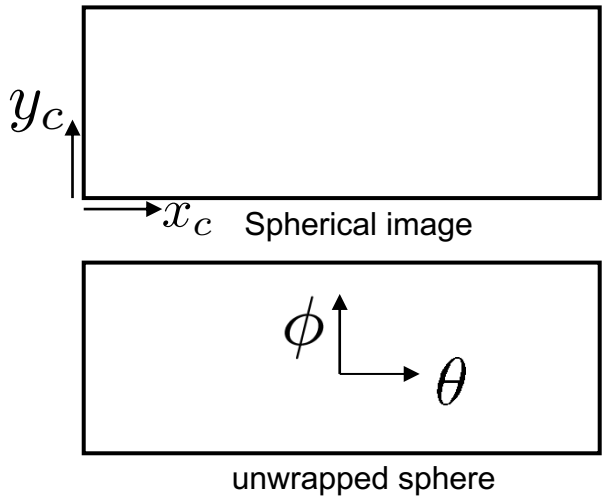
$$(sin\theta cos\phi, sin\phi, cos\theta cos\phi) = (\hat{x}, \hat{y}, \hat{z})$$

- Convert to spherical image coordinates

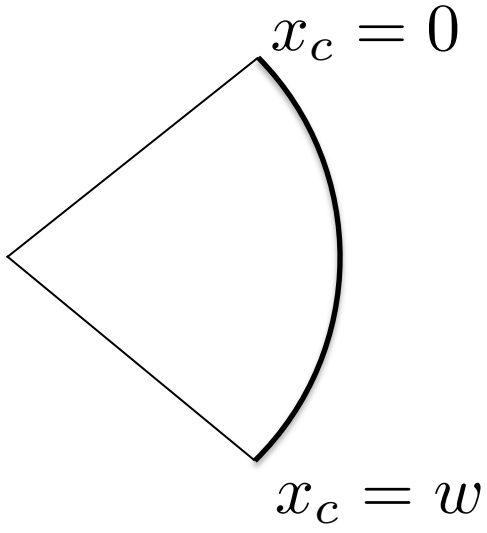
$$(\tilde{x}, \tilde{y}) = (s\theta, s\phi) + (\tilde{x}_c, \tilde{y}_c)$$

  - s defines size of the final image
    - » Usually set s = camera focal length

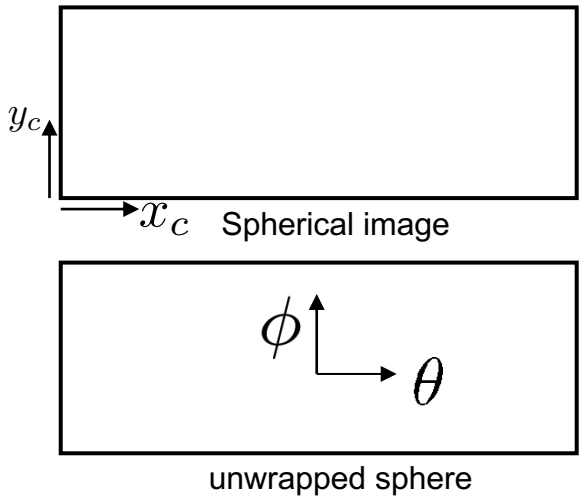
Spherical projection: Inverse Process



- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

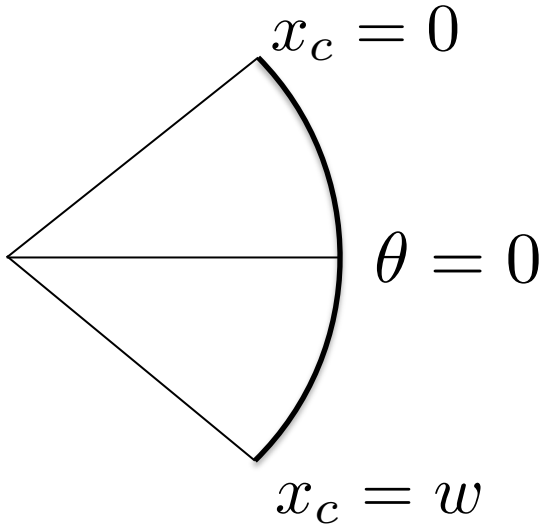


Spherical projection: Inverse Process

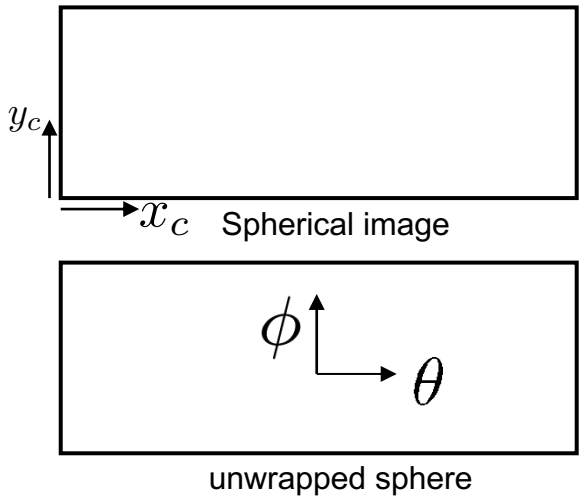


- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

$$\theta = ??(x_c)$$

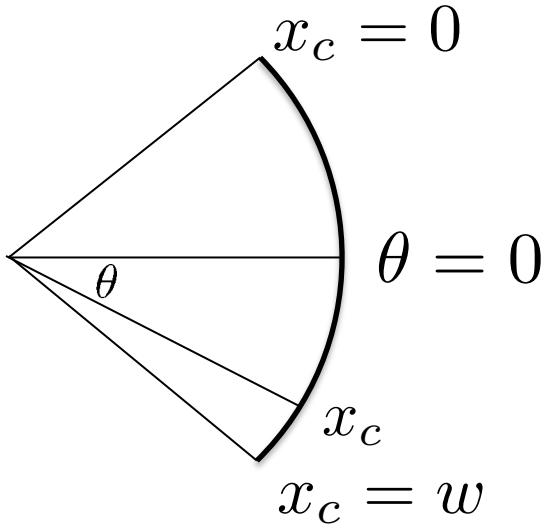


Spherical projection: Inverse Process

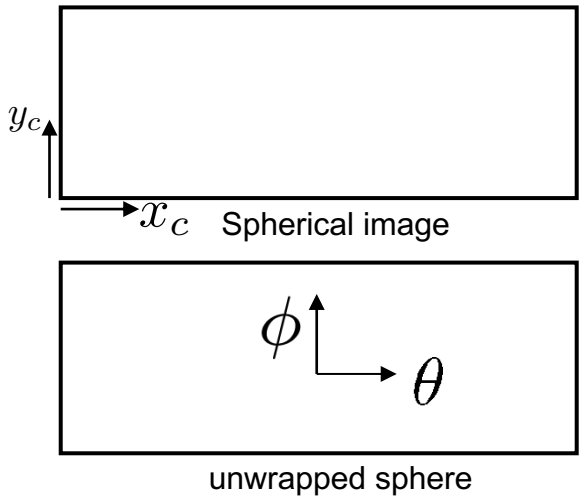


- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

$$\theta = ??(x_c)$$

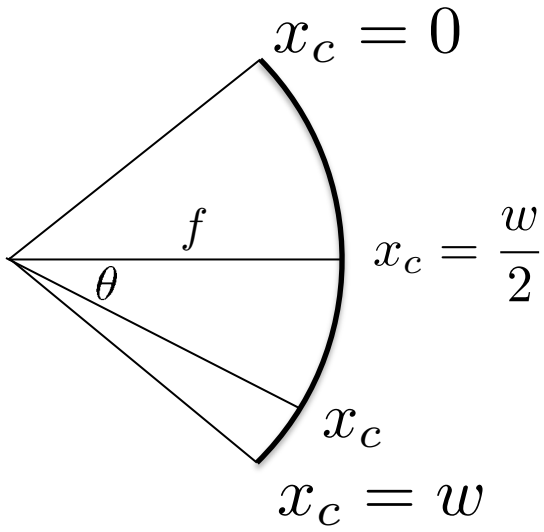


Spherical projection: Inverse Process



- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

$$\theta = ??(x_c)$$



angle

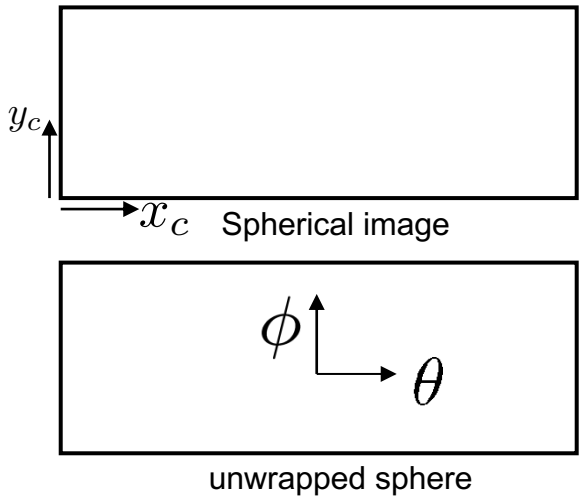
$$\frac{\theta}{2\pi} = \frac{x_c - \frac{w}{2}}{2\pi f}$$

arc

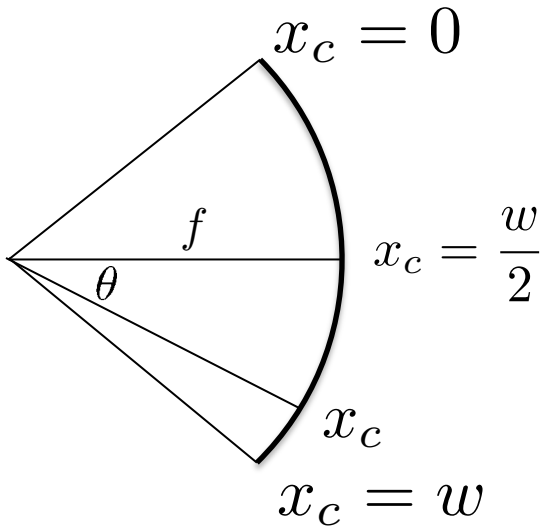
Circumf.

circle

Spherical projection: Inverse Process



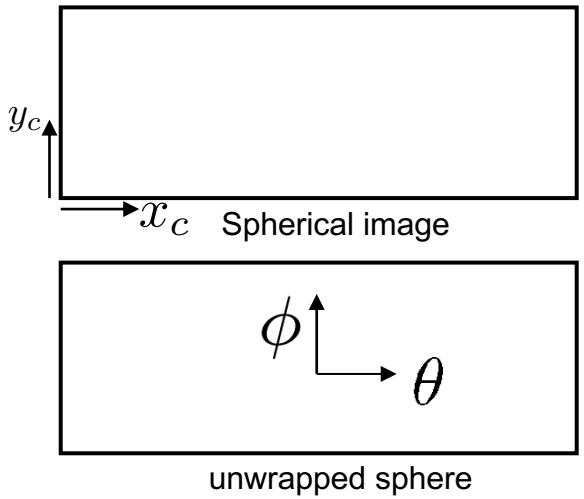
- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:



$$\frac{\theta}{2\pi} = \frac{x_c - \frac{w}{2}}{2\pi f}$$

$$\theta = \frac{1}{f}\left(x_c - \frac{w}{2}\right)$$

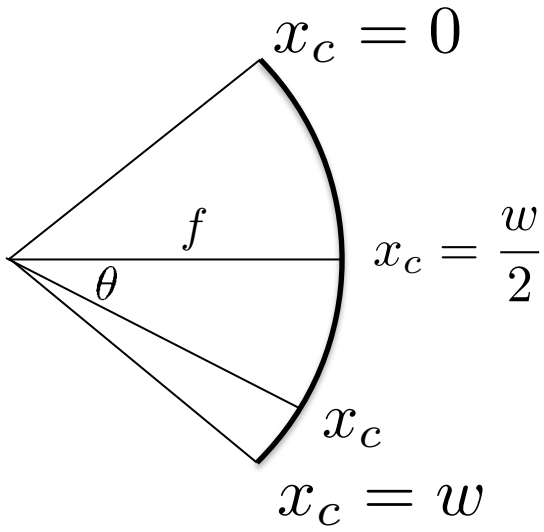
Spherical projection: Inverse Process



- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

$$\theta = \frac{1}{f}(x_c - \frac{w}{2})$$

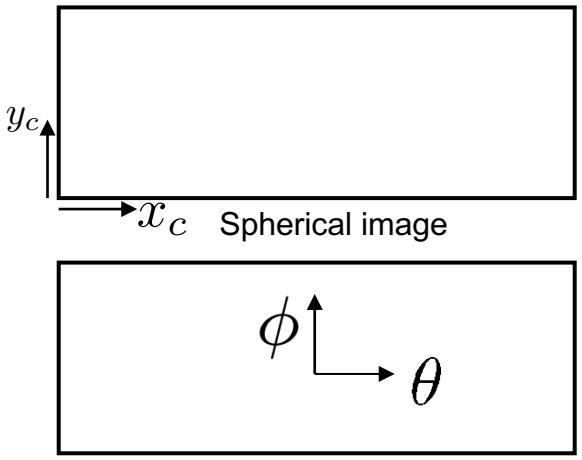
$$\phi = \frac{1}{f}(y_c - \frac{h}{2})$$

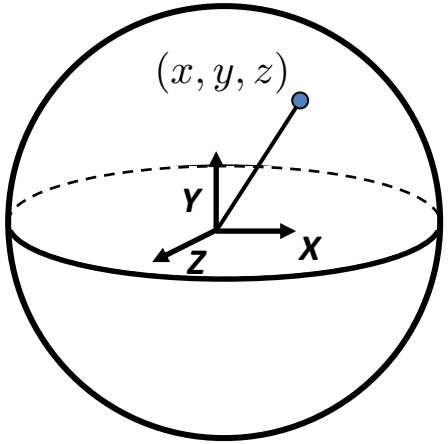


$$\frac{\theta}{2\pi} = \frac{x_c - \frac{w}{2}}{2\pi f}$$

$$\theta = \frac{1}{f}\left(x_c - \frac{w}{2}\right)$$

Spherical projection: Inverse Process





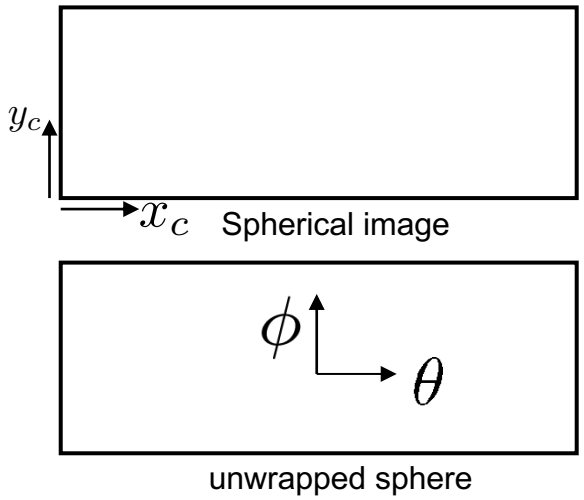
- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

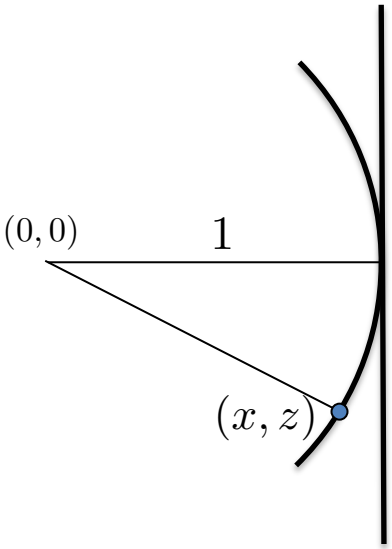
$$\theta = \frac{1}{f}(x_c - \frac{w}{2})$$
$$\phi = \frac{1}{f}(y_c - \frac{h}{2})$$

- Convert angles to (x,y,z) coordinates on a unit sphere:

$$x = \sin\theta\cos\phi$$
$$y = \sin\phi$$
$$z = \cos\theta\cos\phi$$

Spherical projection: Inverse Process





- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

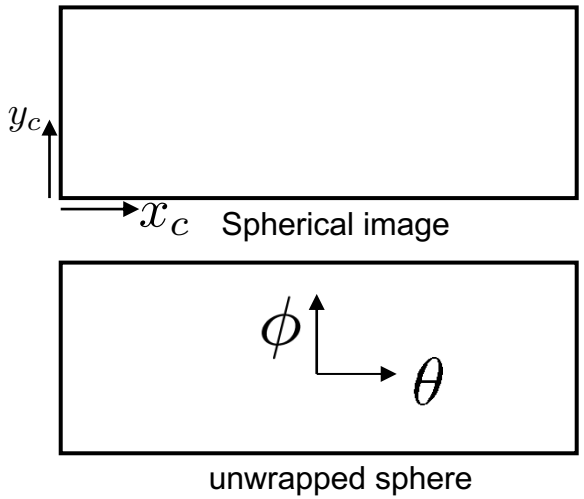
$$\theta = \frac{1}{f}(x_c - \frac{w}{2})$$
$$\phi = \frac{1}{f}(y_c - \frac{h}{2})$$

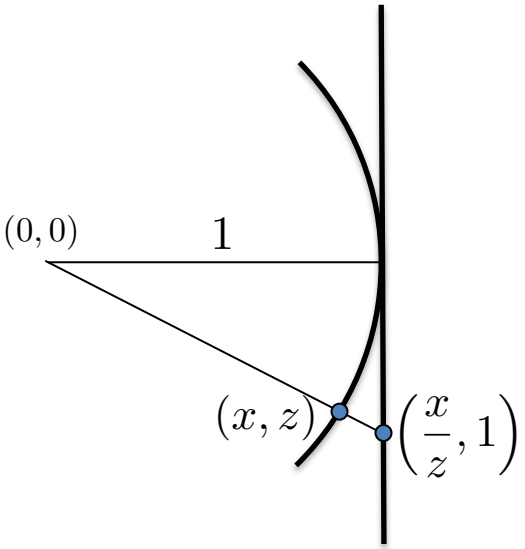
- Convert angles to (x,y,z) coordinates on a unit sphere:

$$x = \sin\theta\cos\phi$$
$$y = \sin\phi$$
$$z = \cos\theta\cos\phi$$

- Project (x,y,z) coordinates onto the image plane at Z=1 by dividing by z:

Spherical projection: Inverse Process





- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

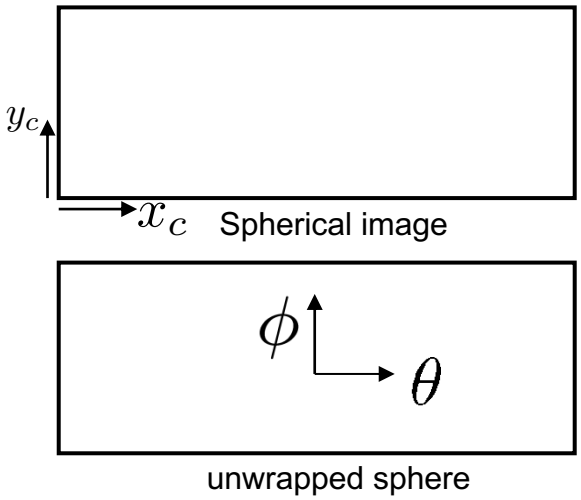
$$\theta = \frac{1}{f}(x_c - \frac{w}{2})$$
$$\phi = \frac{1}{f}(y_c - \frac{h}{2})$$

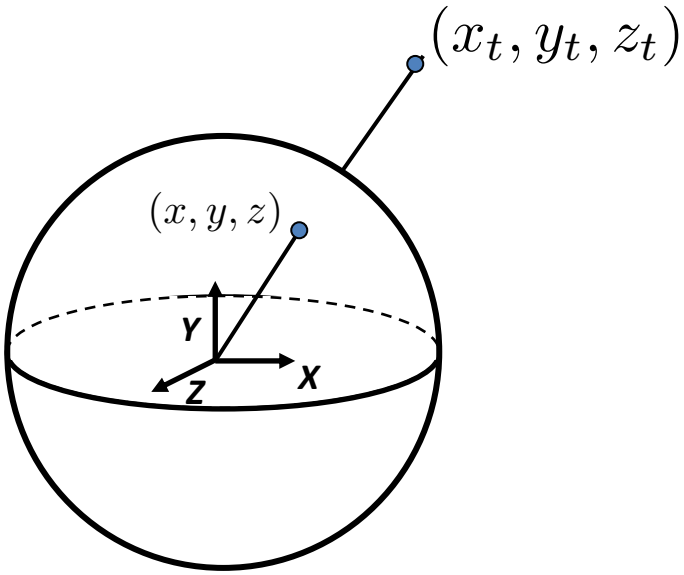
- Convert angles to (x,y,z) coordinates on a unit sphere:

$$x = \sin\theta\cos\phi$$
$$y = \sin\phi$$
$$z = \cos\theta\cos\phi$$

- Project (x,y,z) coordinates onto the image plane at Z=1 by dividing by z: $(x_t, y_t, z_t) = \left(\frac{x}{z}, \frac{y}{z}, 1\right)$

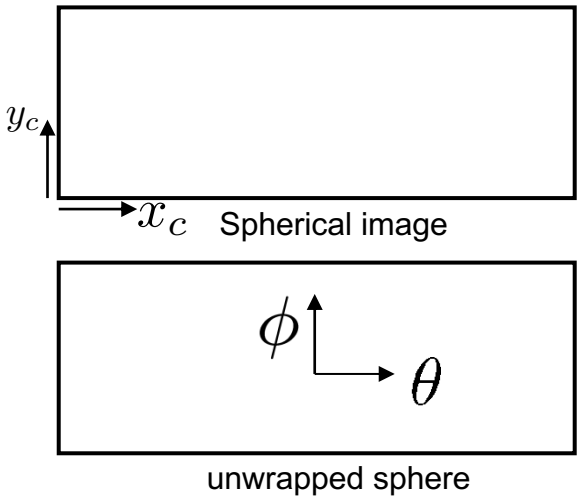
Spherical projection: Inverse Process

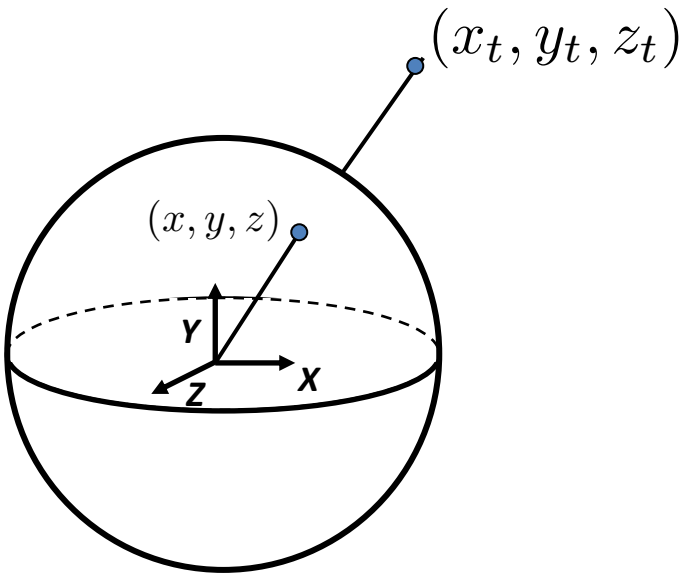




- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image: $\theta = \frac{1}{f}(x_c - \frac{w}{2})$ $\phi = \frac{1}{f}(y_c - \frac{h}{2})$
- Convert angles to (x,y,z) coordinates on a unit sphere: $x = \sin\theta\cos\phi$ $y = \sin\phi$ $z = \cos\theta\cos\phi$
- Project (x,y,z) coordinates onto the image plane at Z=1 by dividing by z: $(x_t, y_t, z_t) = \left(\frac{x}{z}, \frac{y}{z}, 1\right)$

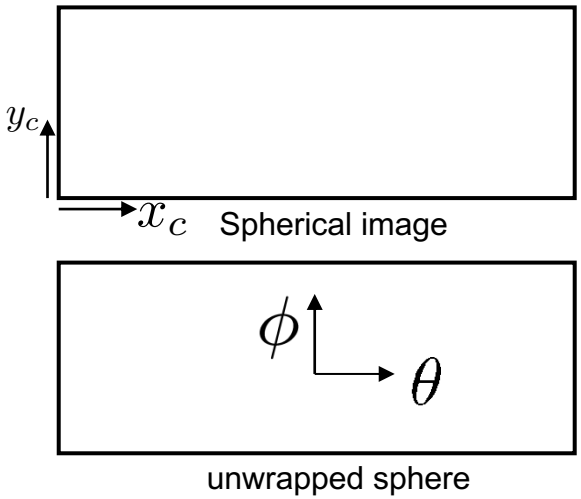
Spherical projection: Inverse Process

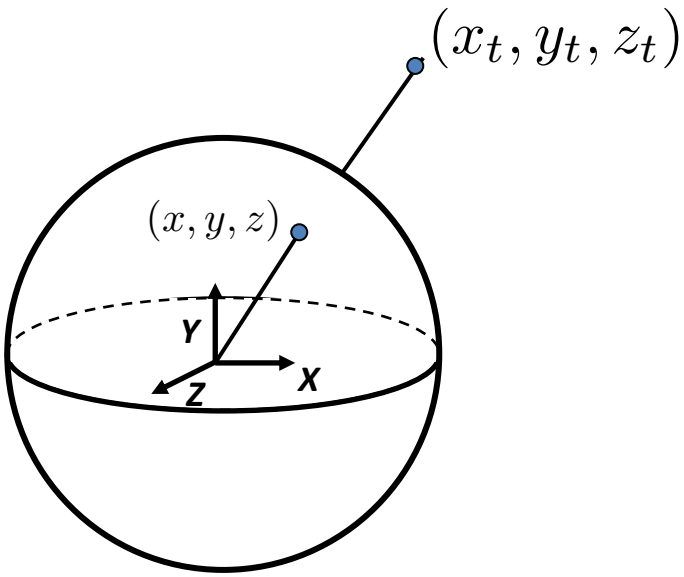




- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image: $\theta = \frac{1}{f}(x_c - \frac{w}{2})$ $\phi = \frac{1}{f}(y_c - \frac{h}{2})$
- Convert angles to (x,y,z) coordinates on a unit sphere: $x = \sin\theta\cos\phi$ $y = \sin\phi$ $z = \cos\theta\cos\phi$
- Project (x,y,z) coordinates onto the image plane at Z=1 by dividing by z: $(x_t, y_t, z_t) = \left(\frac{x}{z}, \frac{y}{z}, 1\right)$
- Convert from image plane to original pixel coordinates:

Spherical projection: Inverse Process





- Begin with spherical image coordinates (x_c, y_c)
- Convert to angles with (0,0) at center of image:

$$\theta = \frac{1}{f}(x_c - \frac{w}{2})$$
$$\phi = \frac{1}{f}(y_c - \frac{h}{2})$$

- Convert angles to (x,y,z) coordinates on a unit sphere:

$$x = \sin\theta\cos\phi$$
$$y = \sin\phi$$
$$z = \cos\theta\cos\phi$$

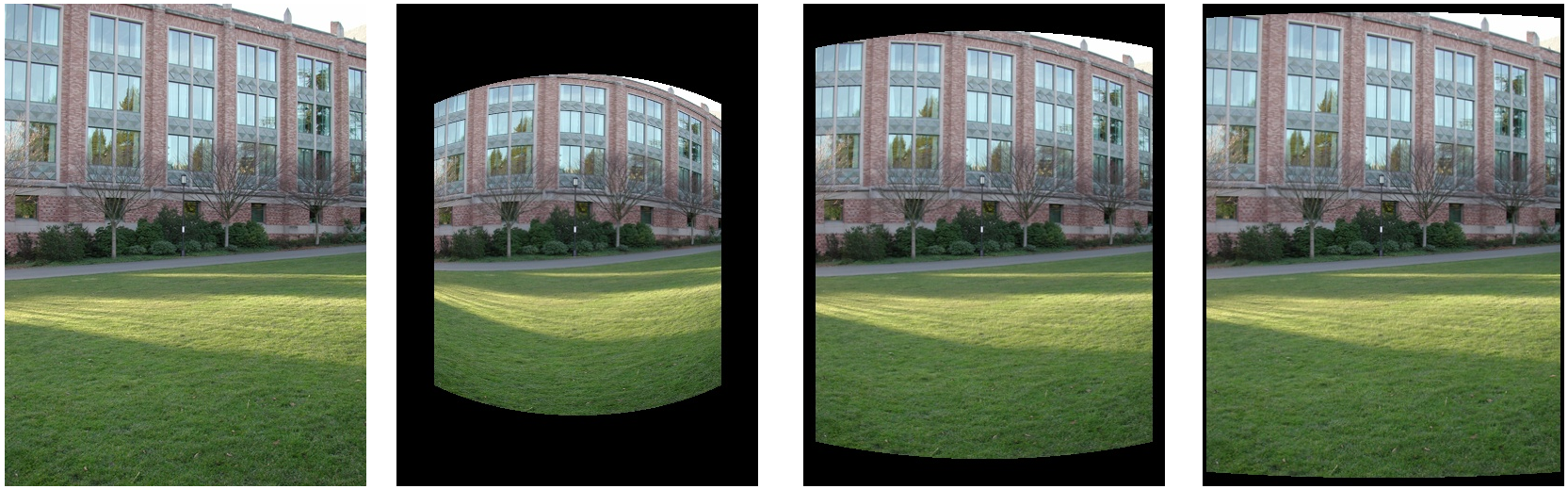
- Project (x,y,z) coordinates onto the image plane at Z=1 by dividing by z:

$$(x_t, y_t, z_t) = \left(\frac{x}{z}, \frac{y}{z}, 1\right)$$

- Convert from image plane to original pixel coordinates:

$$x_n = \frac{w}{2} + f * x_t \qquad y_n = \frac{h}{2} + f * y_t$$

Spherical reprojection



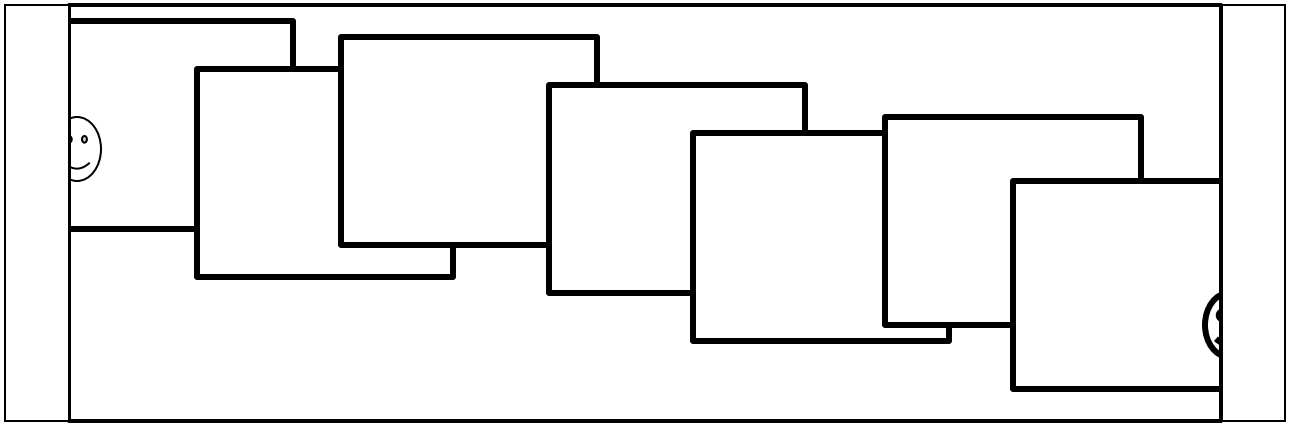
input **f = 200 (pixels)** **f = 400** **f = 800**

- Map image to spherical coordinates
  - need to know the focal length

Aligning spherical images

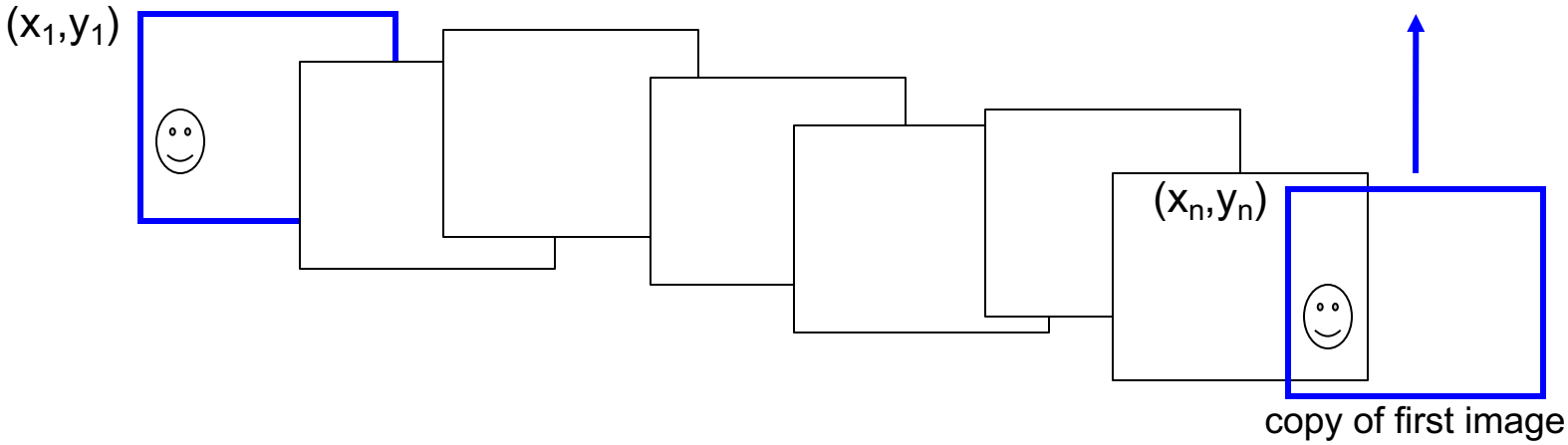
- Suppose we rotate the camera by θ about the vertical axis
  - How does this change the spherical image?

Problem: Drift



- Error accumulation
  - small errors accumulate over time

Problem: Drift



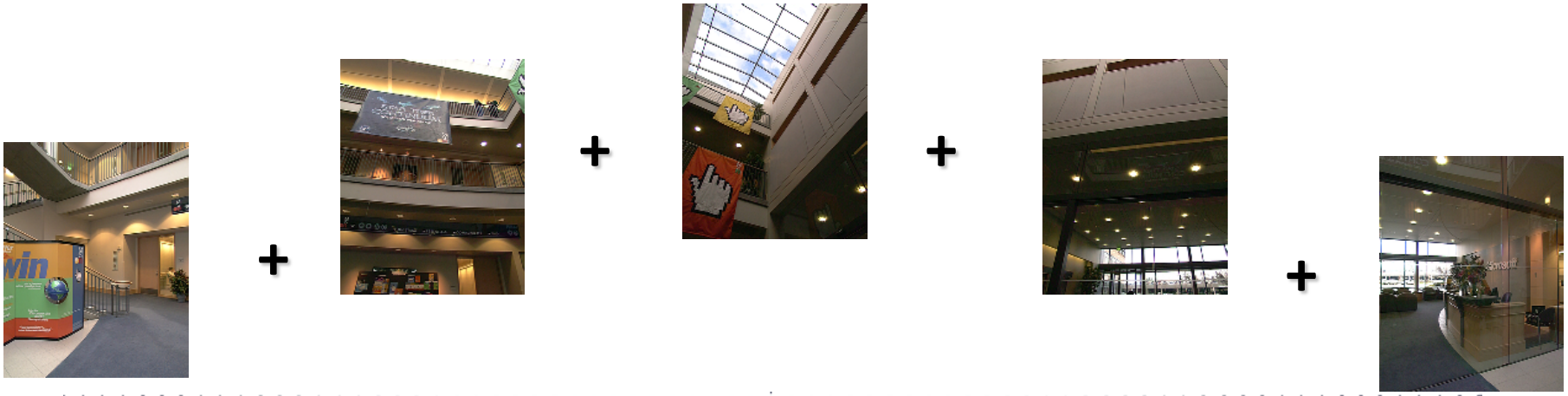
- Solution
  - add another copy of first image at the end
  - this gives a constraint: $y_n = y_1$
  - there are a bunch of ways to solve this problem
    - add displacement of $(y_1 - y_n)/(n-1)$ to each image after the first
    - **apply an affine warp: y' = y + ax [you will implement this for P3]**
    - run a big optimization problem, incorporating this constraint
      - best solution, but more complicated
      - known as "bundle adjustment"

Project 3

- Take pictures on a tripod (or handheld)
- Warp to spherical coordinates (optional if using homographies to align images)
- Extract features
- Align neighboring pairs using RANSAC
- Write out list of neighboring translations
- Correct for drift
- Read in warped images and blend them
- Crop the result and import into a viewer

- Roughly based on Autostitch
  - By Matthew Brown and David Lowe
  - http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Spherical panoramas



Microsoft Lobby: http://www.acm.org/pubs/citations/proceedings/graph/258734/p251-szeliski

Different projections are possible

Cube-map

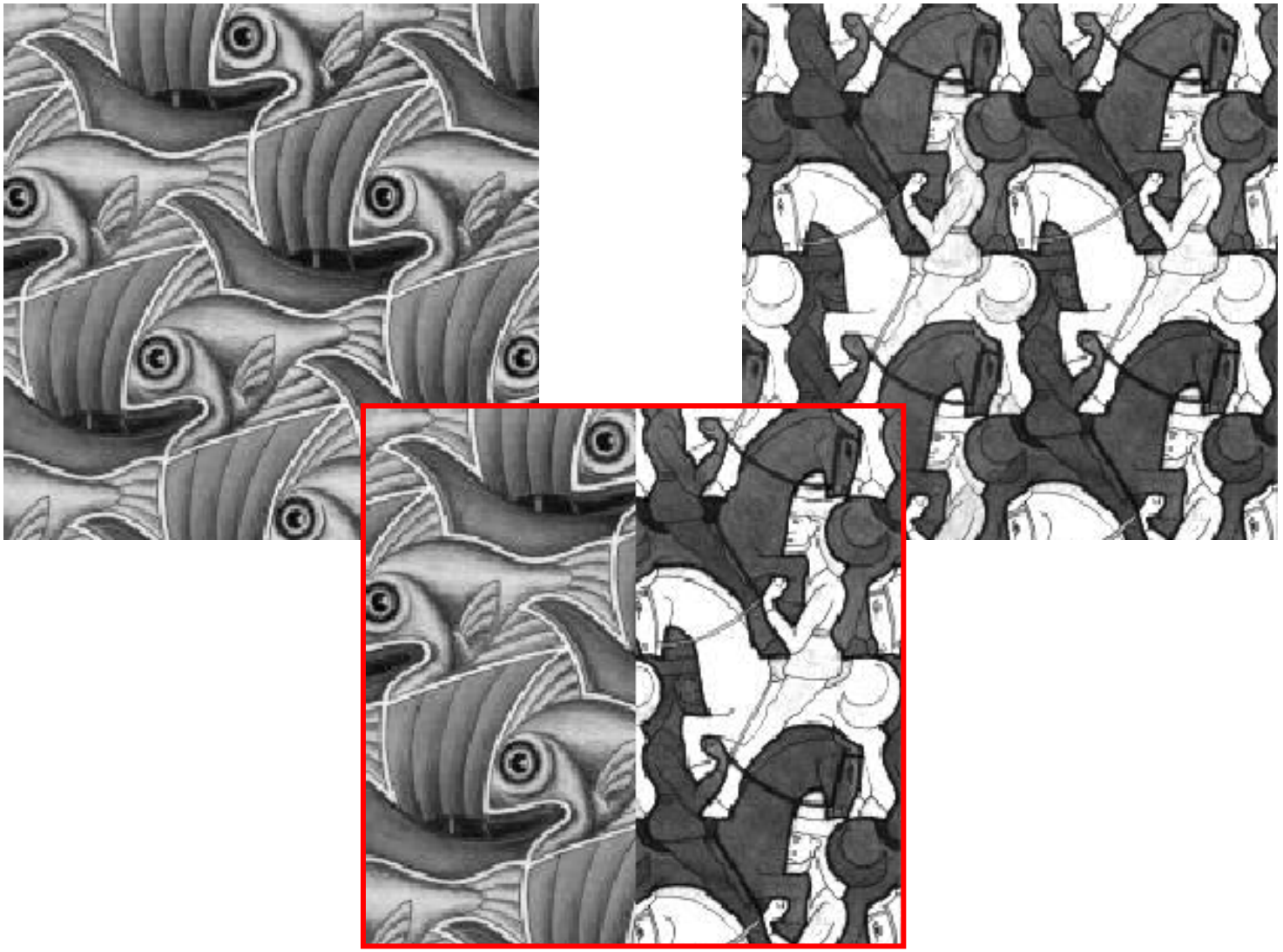
Blending

- We’ve aligned the images – now what?

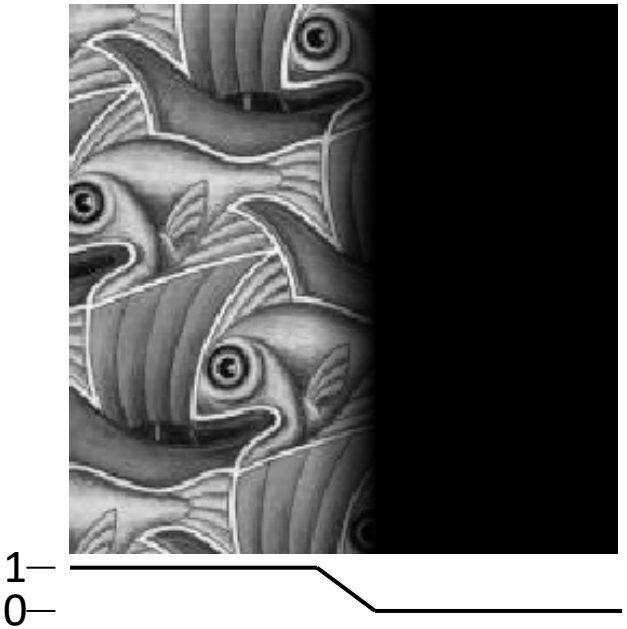
Blending

- Want to seamlessly blend them together

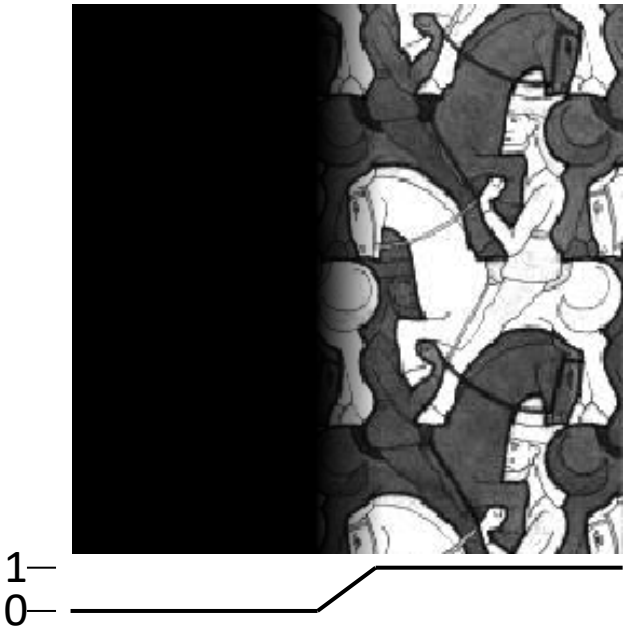
Image Blending



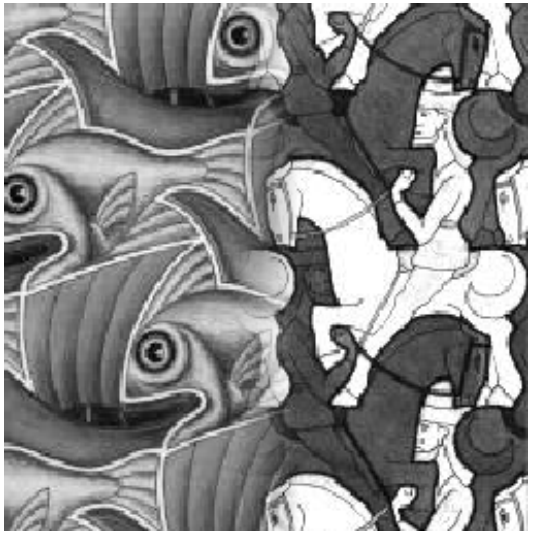
Feathering



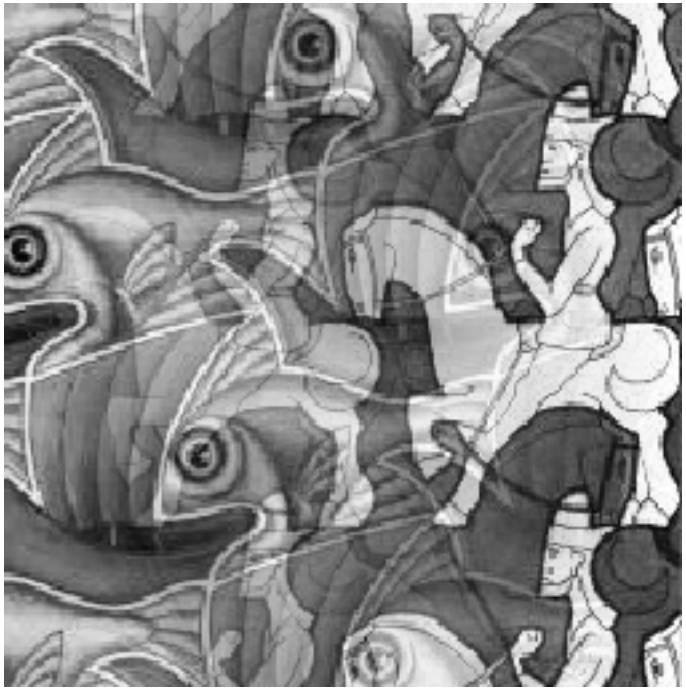
+

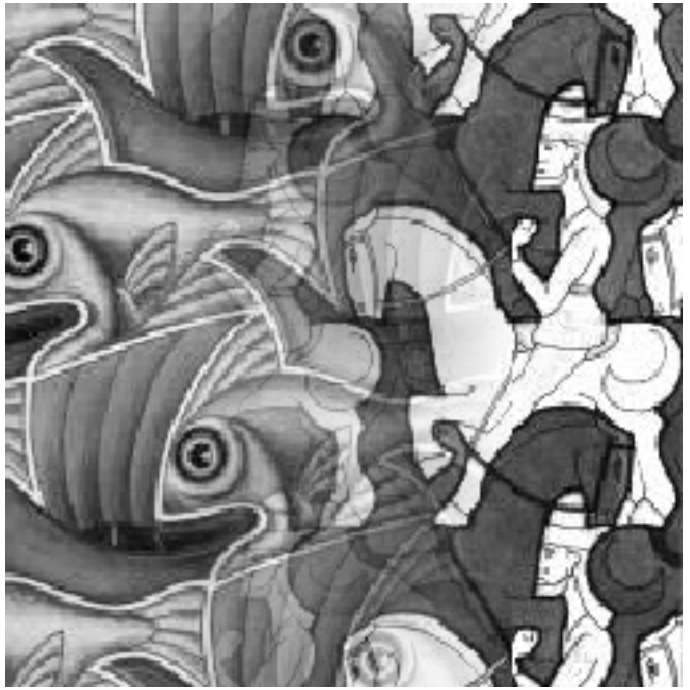


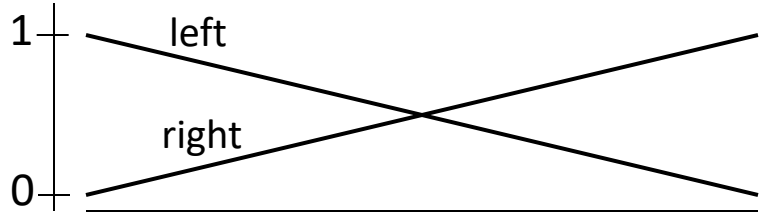
=

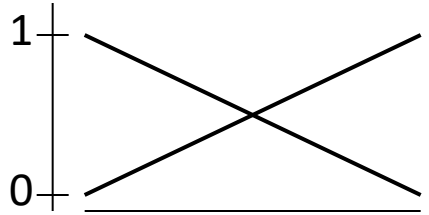


Effect of window size

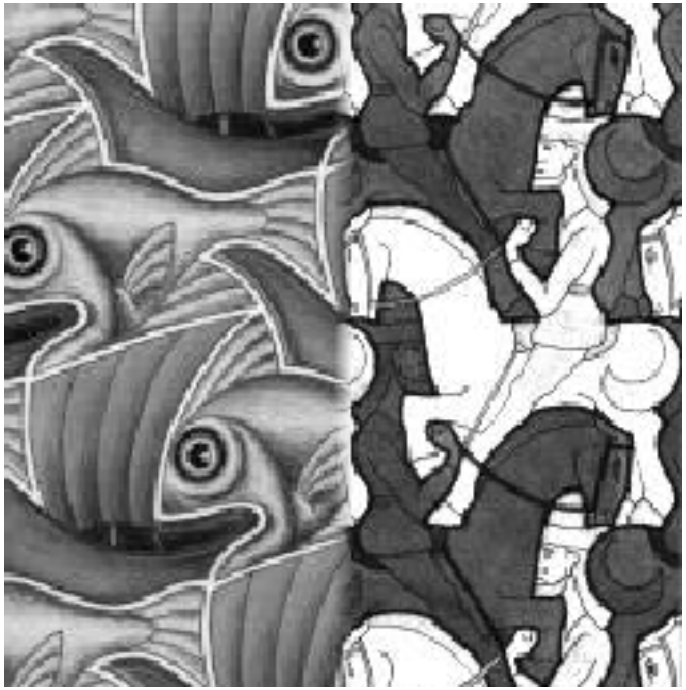


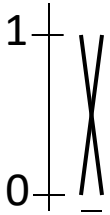


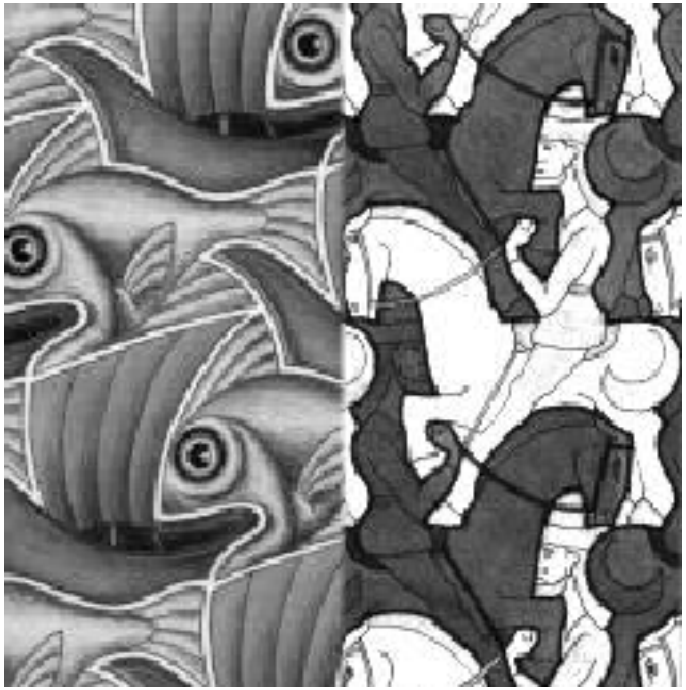




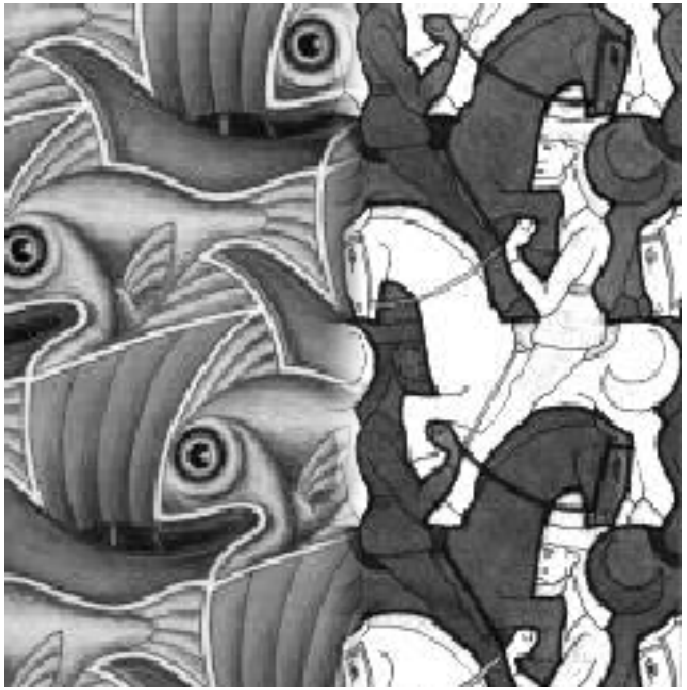
Effect of window size

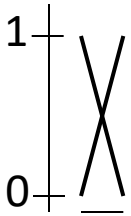






Good window size

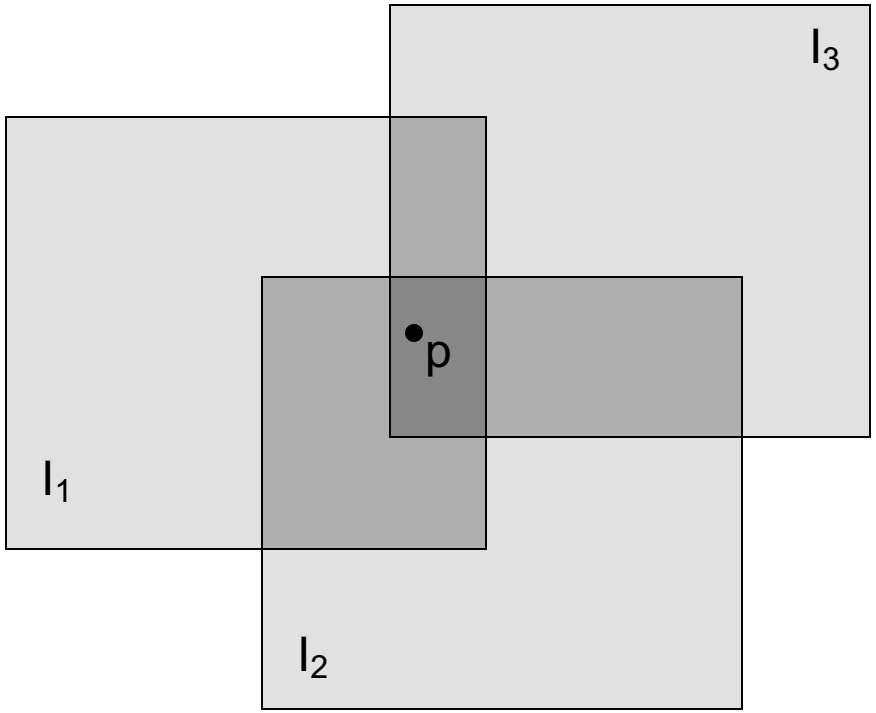




Best window: smooth but not ghosted

- Doesn’t always work...
- Fancier blending tricks exist.

With more images: Alpha Blending



Encoding blend weights: I(x,y) = (αR, αG, αB, α)

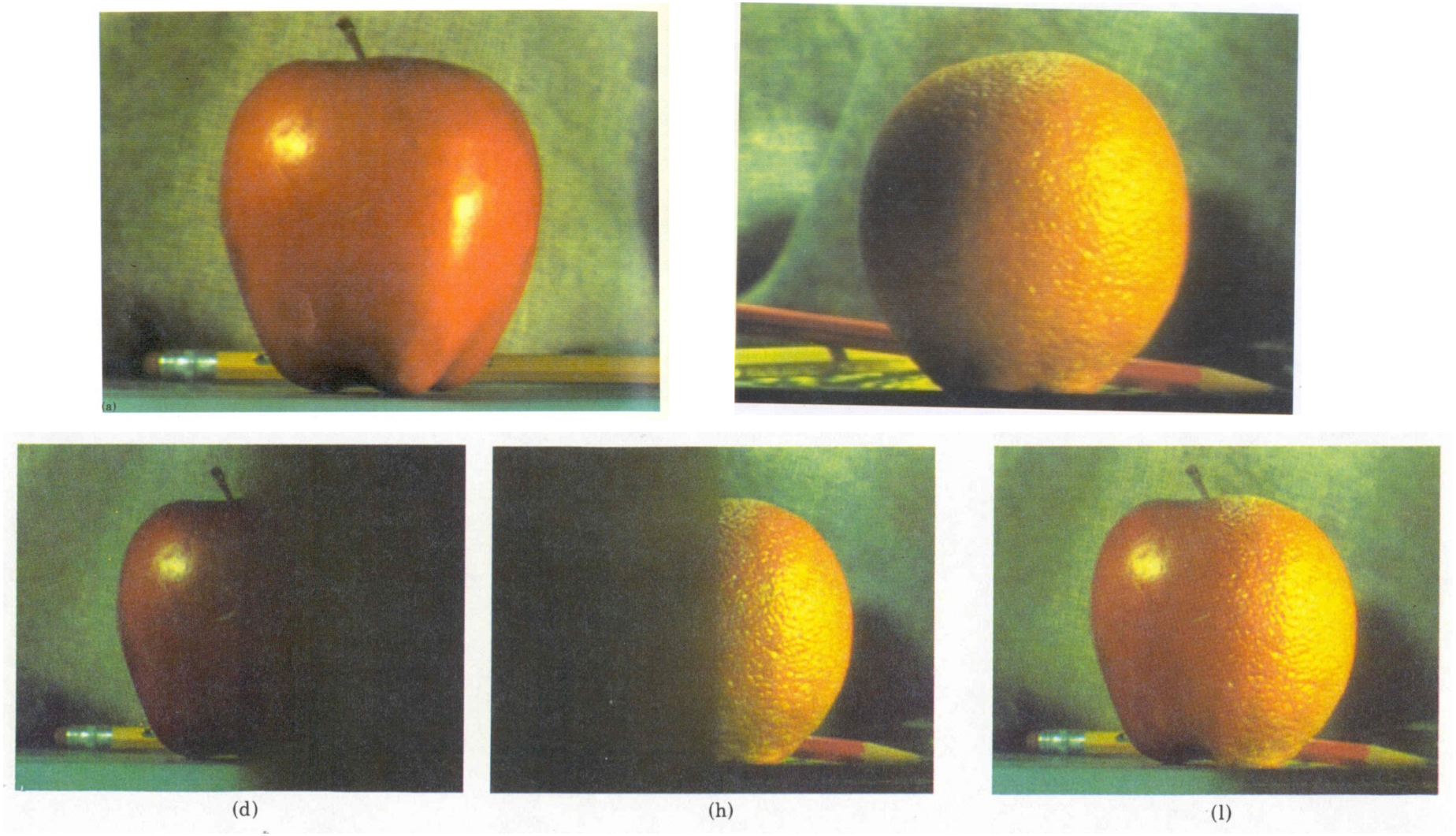
color at p = $\frac{(\alpha_1 R_1,\ \alpha_1 G_1,\ \alpha_1 B_1) + (\alpha_2 R_2,\ \alpha_2 G_2,\ \alpha_2 B_2) + (\alpha_3 R_3,\ \alpha_3 G_3,\ \alpha_3 B_3)}{\alpha_1 + \alpha_2 + \alpha_3}$

Implement this in two steps:

1. accumulate: add up the (α premultiplied) RGBα values at each pixel
2. normalize: divide each pixel's accumulated RGB by its α value

   Q: what if α = 0?

Pyramid blending



Create a Laplacian pyramid, blend each level

- Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.