

CS5670: Computer Vision

Panoramas



[Space Shuttle Discovery Flight Deck, Gigapan](#)

Announcements

- Midterm exam due by the start of class today
- Project 3: Autostitch (Panorama Stitching)
 - Released today, March 5
 - Demo at end of class
 - Due on Friday, March 15, by 8pm
 - Artifact due on Monday March 8 by 8pm
 - To be done in groups of 2
 - If you need help finding a team member, let us know

First: finishing up camera projection

Recap: Coordinate frames

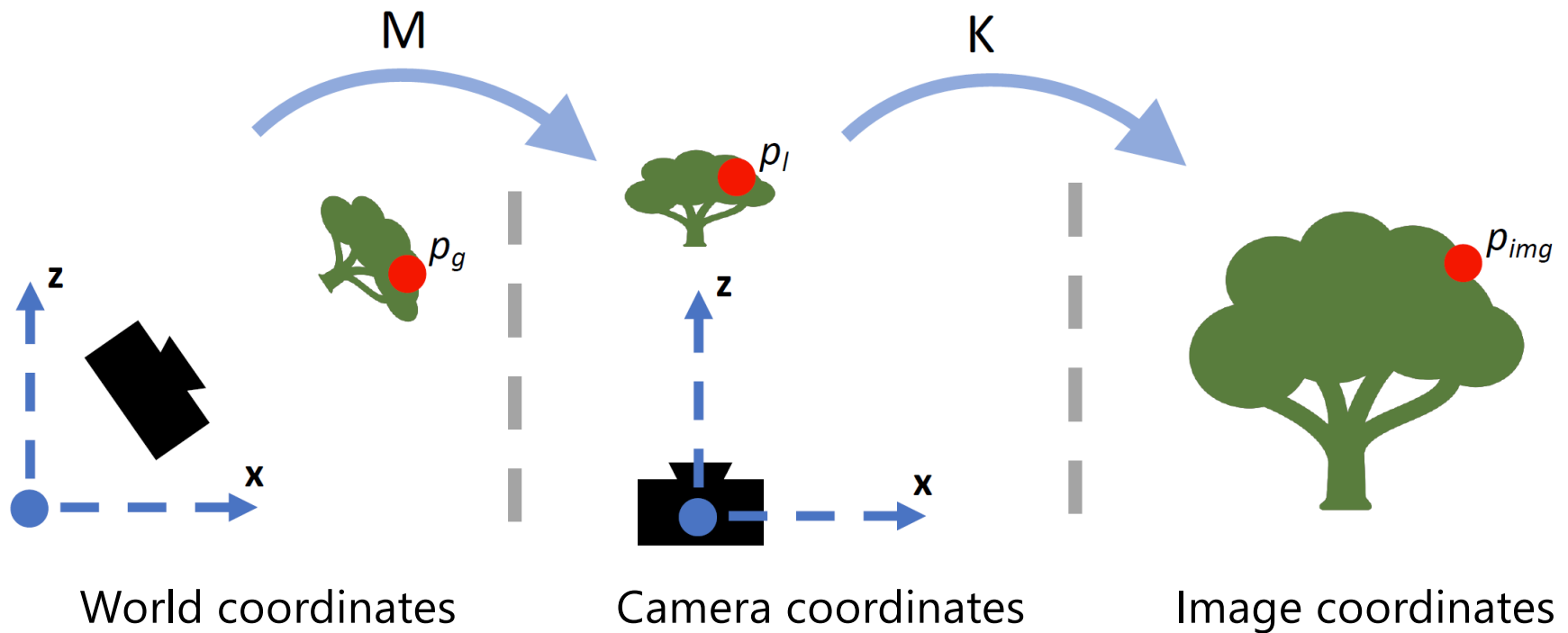
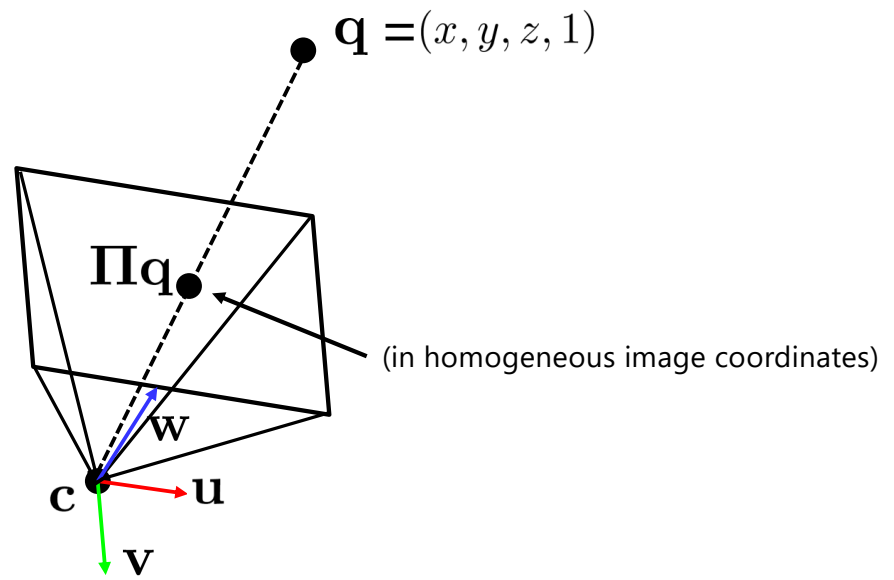
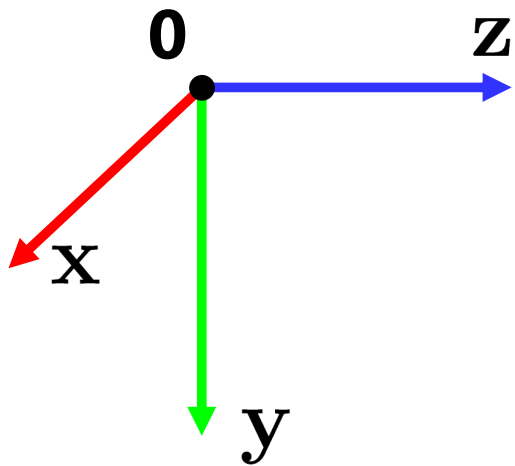


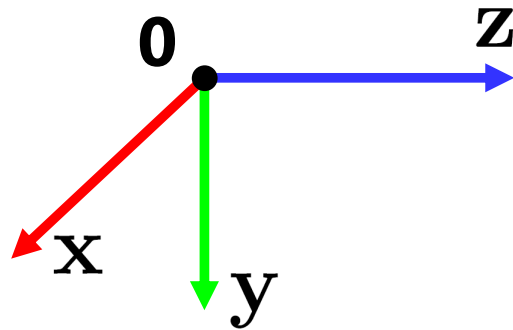
Figure credit: Peter Hedman

Projection matrix

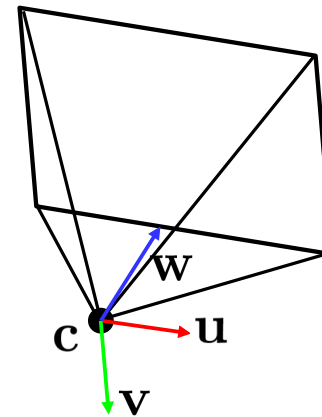


Extrinsics

- How do we get the camera to “canonical form”?
 - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards

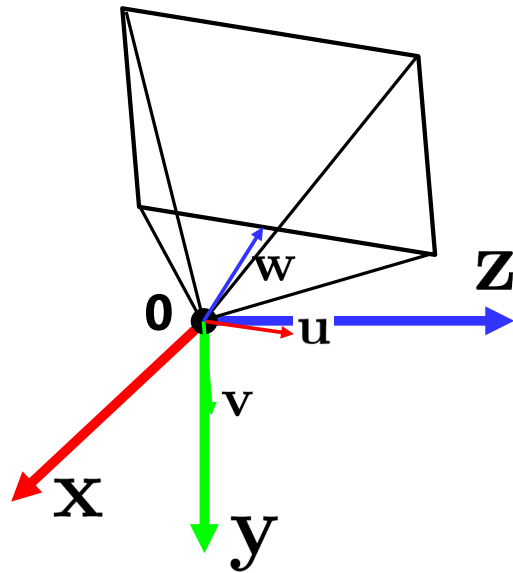


Step 1: Translate by $-c$



Extrinsics

- How do we get the camera to “canonical form”?
 - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



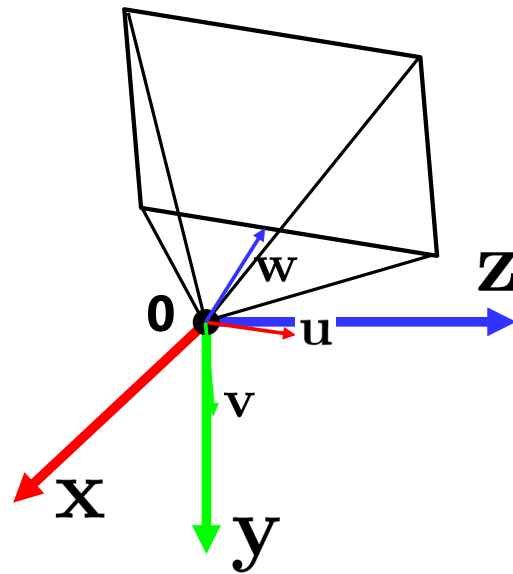
Step 1: Translate by $-\mathbf{c}$

How do we represent translation as a matrix multiplication?

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_{3 \times 3} & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Extrinsics

- How do we get the camera to “canonical form”?
 - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



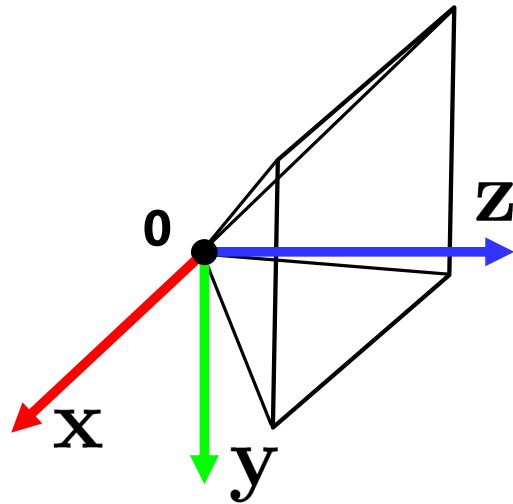
Step 1: Translate by $-\mathbf{c}$
Step 2: Rotate by \mathbf{R}

$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$

3x3 rotation matrix

Extrinsics

- How do we get the camera to “canonical form”?
 - Canonical form: Center of projection at the origin, x-axis points right, y-axis points down, z-axis points forwards



Step 1: Translate by $-\mathbf{c}$

Step 2: Rotate by \mathbf{R}

$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

(with extra row/column of $[0 \ 0 \ 0 \ 1]$)

Perspective projection

$$\underbrace{\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

\mathbf{K}
(intrinsic) (converts from 3D rays in camera coordinate system to pixel coordinates)

in general, $\mathbf{K} = \begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$ (upper triangular matrix)

α : **aspect ratio** (1 unless pixels are not square)

s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

(c_x, c_y) : **principal point** ((w/2,h/2) unless optical axis doesn't intersect projection plane at image center)

Typical intrinsics matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- **2D affine transform** corresponding to a scale by f (focal length) and a translation by (c_x, c_y) (principal point)
- Maps 3D rays to 2D pixels

Focal length

- Can think of as “zoom”



24mm



50mm



200mm

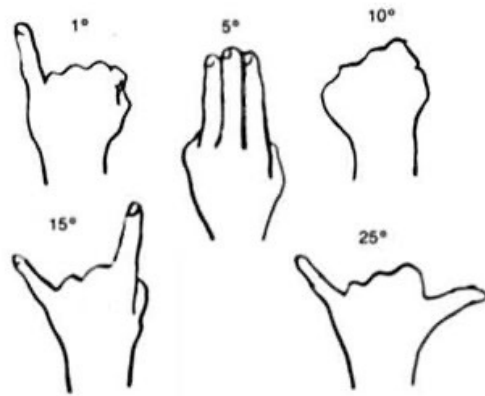


800mm









- Also related to *field of view*

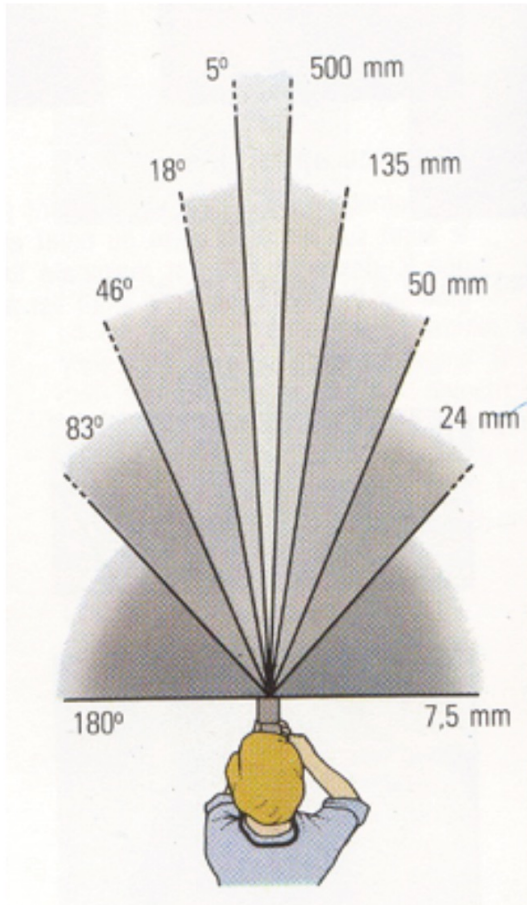
Field of view



APS-C Crop Body Measurement Table

Lens	After 1.62 Multiplier	APS-C Sensor (1.62 lens multiplier) Canon 60D, 7D, 70D, T3i, T4i	Hand Positions
18mm	29.16mm	Three hands wide at full arms length.	
28mm	45.36mm	Slightly less than two hands wide at full arms length.	
35mm	56.7mm	One hand + width of one fist at full arms length.	
50mm	81mm	One hand wide + width of thumb at full arms length.	
55mm	89.1mm	Slightly less than one hand wide at full arms length.	
85mm	137.7mm	Inside edge of thumb to tip of forefinger wide with hand in "L" shape, thumb up.	

Focal length in practice



24mm



50mm

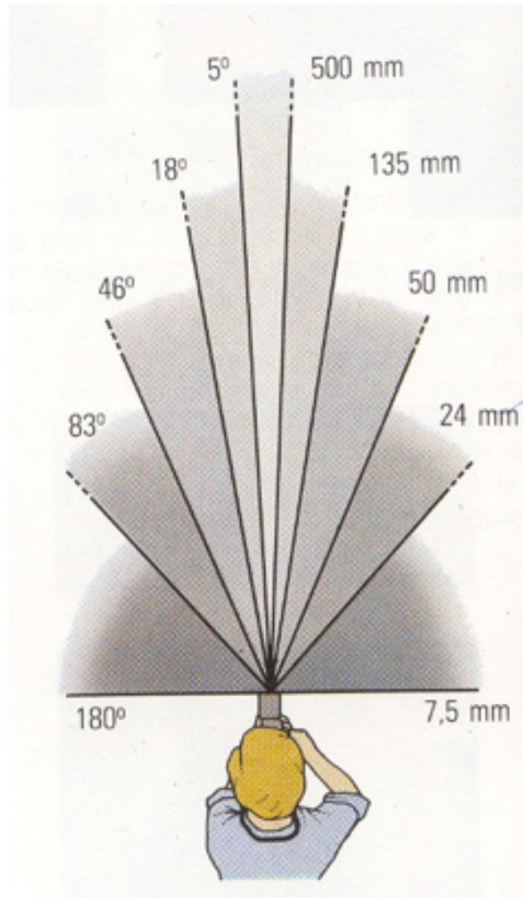


135mm

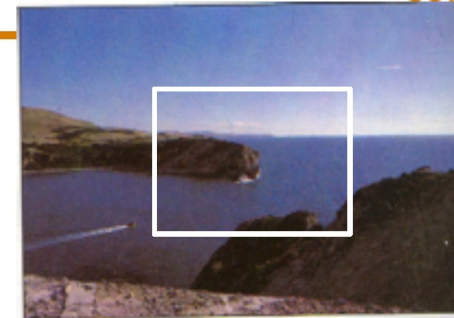


Fredo Durand

Focal length = cropping



24mm



50mm

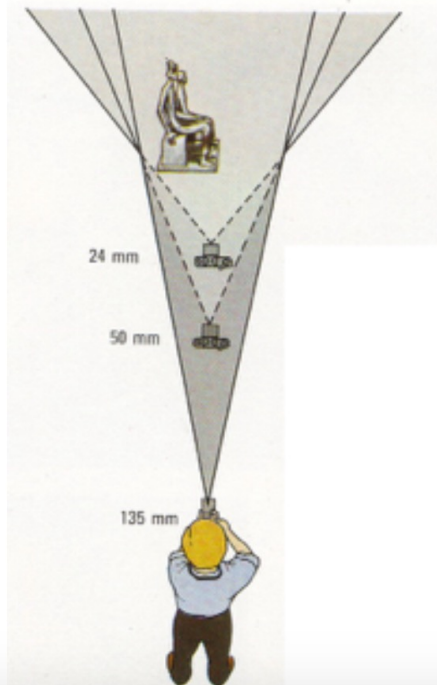


135mm



Focal length vs. viewpoint

- Telephoto makes it easier to select background (a small change in viewpoint is a big change in background).



Grand-angulaire 24 mm



Normal 50 mm



Longue focale 135 mm

Fredo Durand



Fredo Durand



Wide angle



Standard



Telephoto



<http://petapixel.com/2013/01/11/how-focal-length-affects-your-subjects-apparent-weight-as-seen-with-a-cat/>

Projection matrix

$$\mathbf{\Pi} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

The \mathbf{K} matrix converts 3D rays in the camera's coordinate system to 2D image points in image (pixel) coordinates.

This part converts 3D points in world coordinates to 3D rays in the camera's coordinate system. There are 6 parameters represented (3 for position/translation, 3 for rotation).

Projection matrix

$$\mathbf{\Pi} = \mathbf{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3 \times 3} & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{translation}}$$

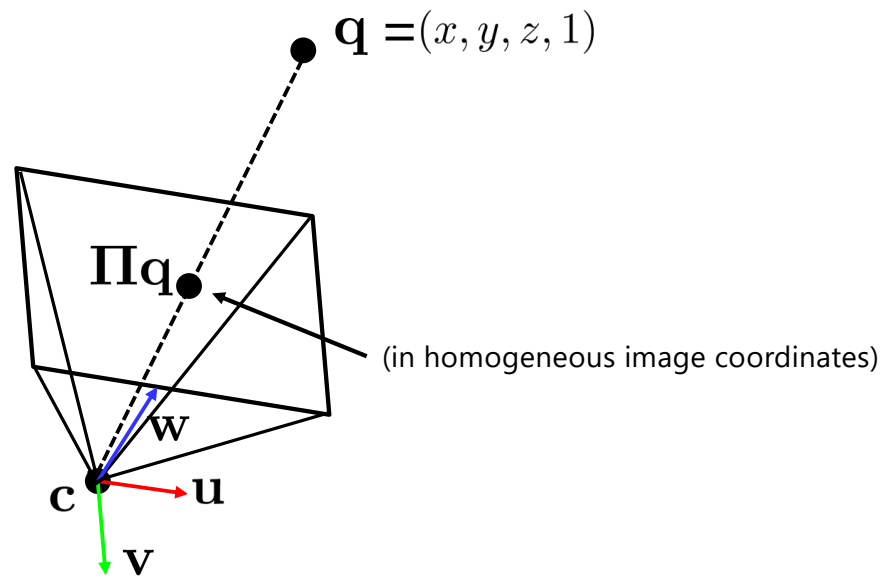
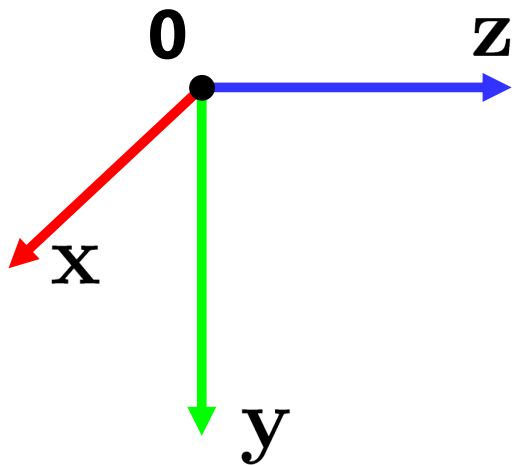
$$\left[\mathbf{R} \mid \underbrace{-\mathbf{R}\mathbf{c}} \right]$$



(\mathbf{t} in book's notation)

$$\mathbf{\Pi} = \mathbf{K} \left[\mathbf{R} \mid -\mathbf{R}\mathbf{c} \right]$$

Projection matrix



Questions?

Perspective distortion

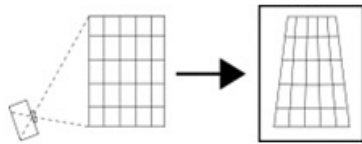
- Problem for architectural photography: converging verticals



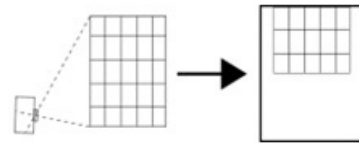
Source: F. Durand

Perspective distortion

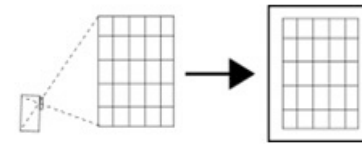
- Problem for architectural photography: converging verticals



Tilting the camera upwards results in converging verticals

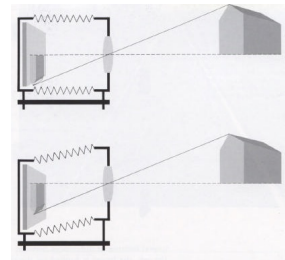
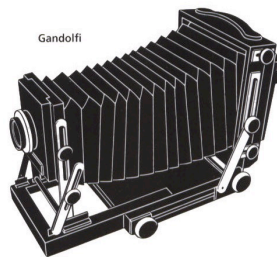


Keeping the camera level, with an ordinary lens, captures only the bottom portion of the building



Shifting the lens upwards results in a picture of the entire subject

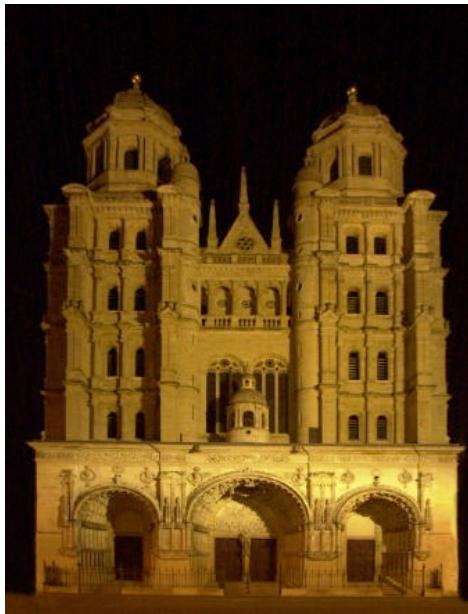
- Solution: view camera (lens shifted w.r.t. film)



http://en.wikipedia.org/wiki/Perspective_correction_lens

Perspective distortion

- Problem for architectural photography: converging verticals
- Result:



Source: F. Durand

Perspective distortion

- What does a sphere project to?

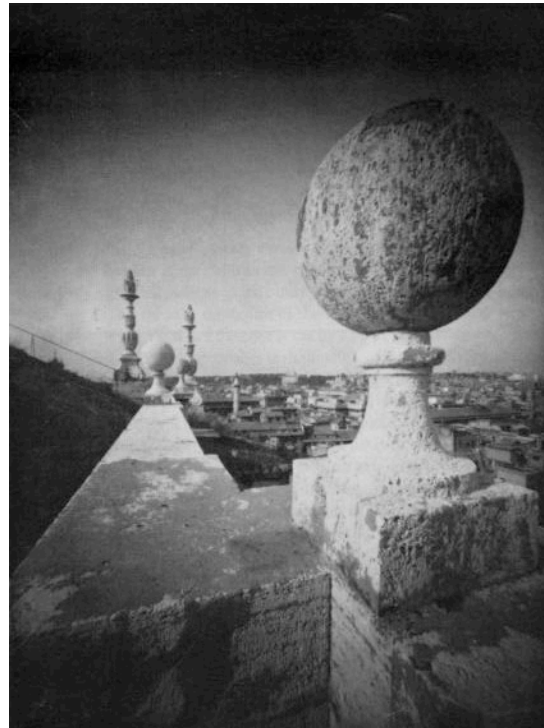
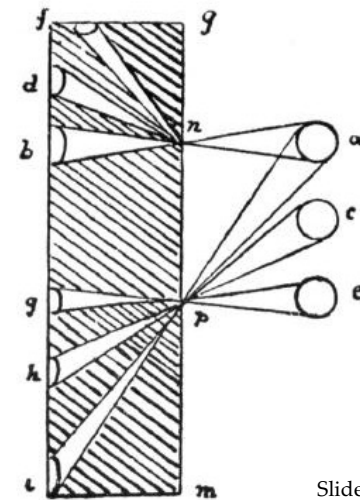
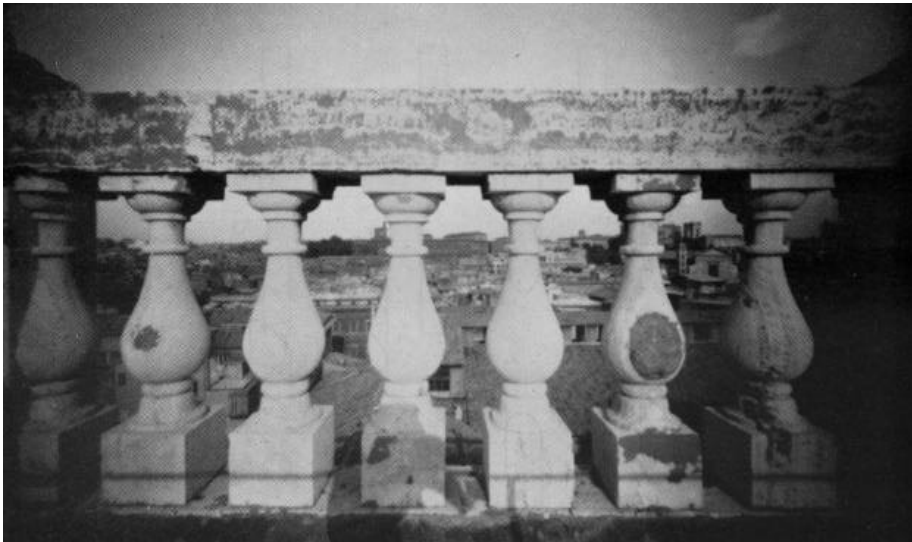


Image source: F. Durand

Perspective distortion

- The exterior columns appear bigger
- The distortion is not due to lens flaws
- Problem pointed out by Da Vinci



Slide by F. Durand

Perspective distortion: People





<https://aaronhertzmam.com/2022/02/28/how-does-perspective-work.html>

Distortion-Free Wide-Angle Portraits on Camera Phones



(a) A wide-angle photo with distortions on subjects' faces.

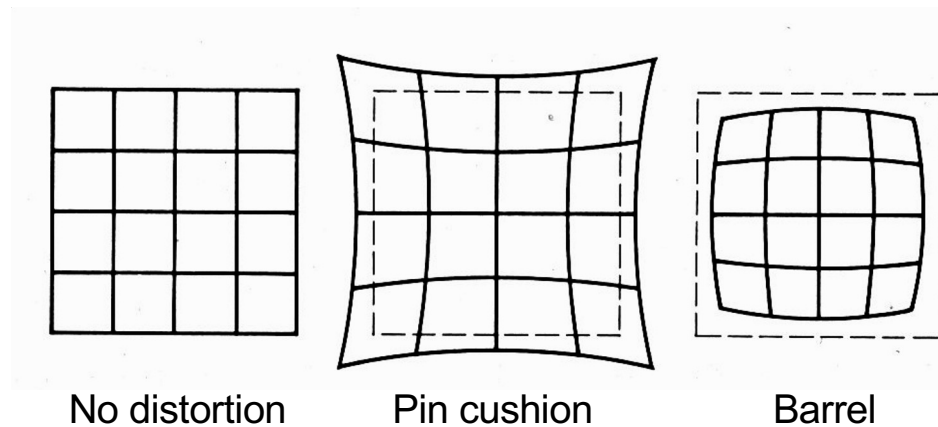


(b) Distortion-free photo by our method.

YiChang Shih, Wei-Sheng Lai, and Chia-Kai Liang, Distortion-Free Wide-Angle Portraits on Camera Phones, SIGGRAPH 2019

https://people.csail.mit.edu/yichangshih/wide_angle_portrait/

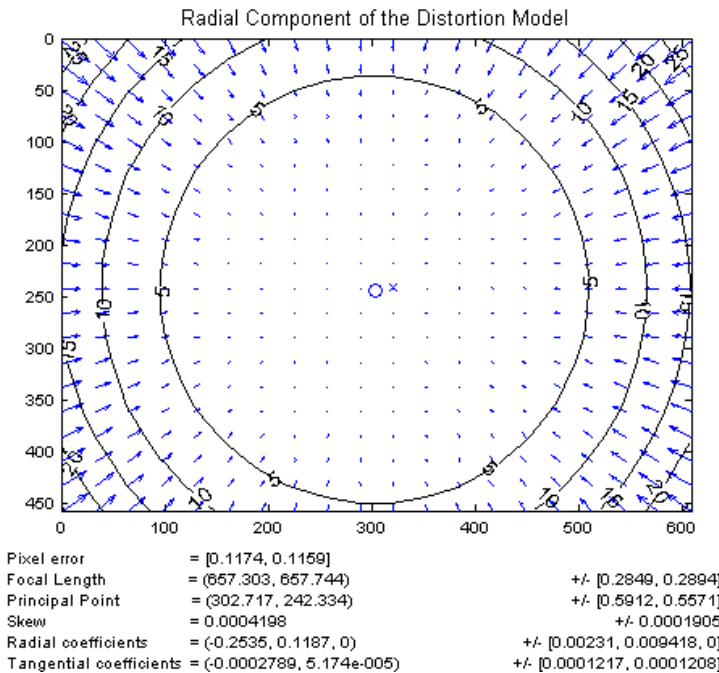
Lens or radial distortion



- Radial distortion of the image
 - Caused by imperfect lenses
 - Deviations are most noticeable for rays that pass through the edge of the lens



Radial distortion



- Arrows show motion of projected points relative to an ideal (distortion-free lens)

[Image credit: J. Bouquet http://www.vision.caltech.edu/bouquetj/calib_doc/htmls/example.html]

Correcting radial distortion



from [Helmut Dersch](#)

Modeling distortion: projection model

Project $(\hat{x}, \hat{y}, \hat{z})$
to "normalized"
image coordinates

$$x'_n = \hat{x} / \hat{z}$$

$$y'_n = \hat{y} / \hat{z}$$

Apply radial distortion

$$r^2 = x'^2_n + y'^2_n$$

$$x'_d = x'_n (1 + \kappa_1 r^2 + \kappa_2 r^4)$$

$$y'_d = y'_n (1 + \kappa_1 r^2 + \kappa_2 r^4)$$

Apply focal length
translate image center

$$x' = f x'_d + x_c$$

$$y' = f y'_d + y_c$$

- To model lens distortion
 - Use above projection operation instead of standard projection matrix multiplication

Other types of projection

- Lots of intriguing variants...
- (I'll just mention a few fun ones)

360 degree field of view...



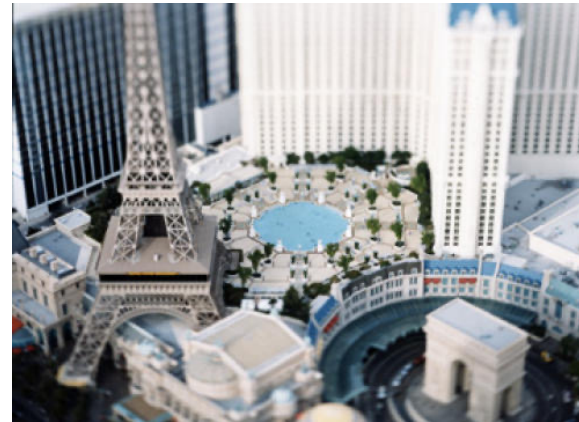
Ricoh Theta

- Basic approach
 - Take a photo of a parabolic mirror with an orthographic lens (Nayar)
 - Or buy one a lens from a variety of omnicam manufacturers...
 - See <http://www.cis.upenn.edu/~kostas/omni.html>
 - Or buy a 360-degree consumer camera

Tilt-shift

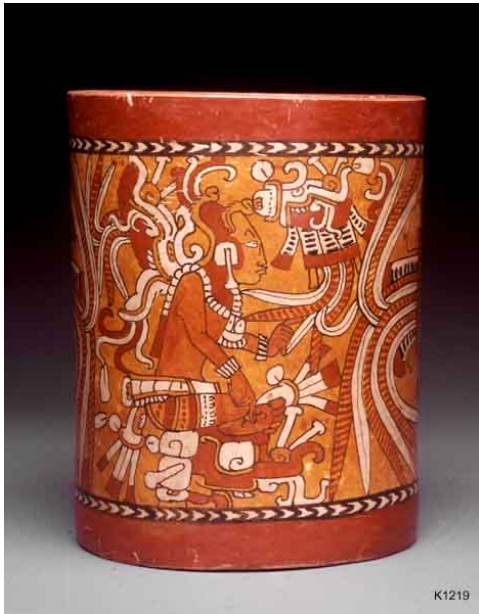


http://www.northlight-images.co.uk/article_pages/tilt_and_shift_ts-e.html



Tilt-shift images from [Olivo Barbieri](#)
and Photoshop [imitations](#)

Rotating sensor (or object)



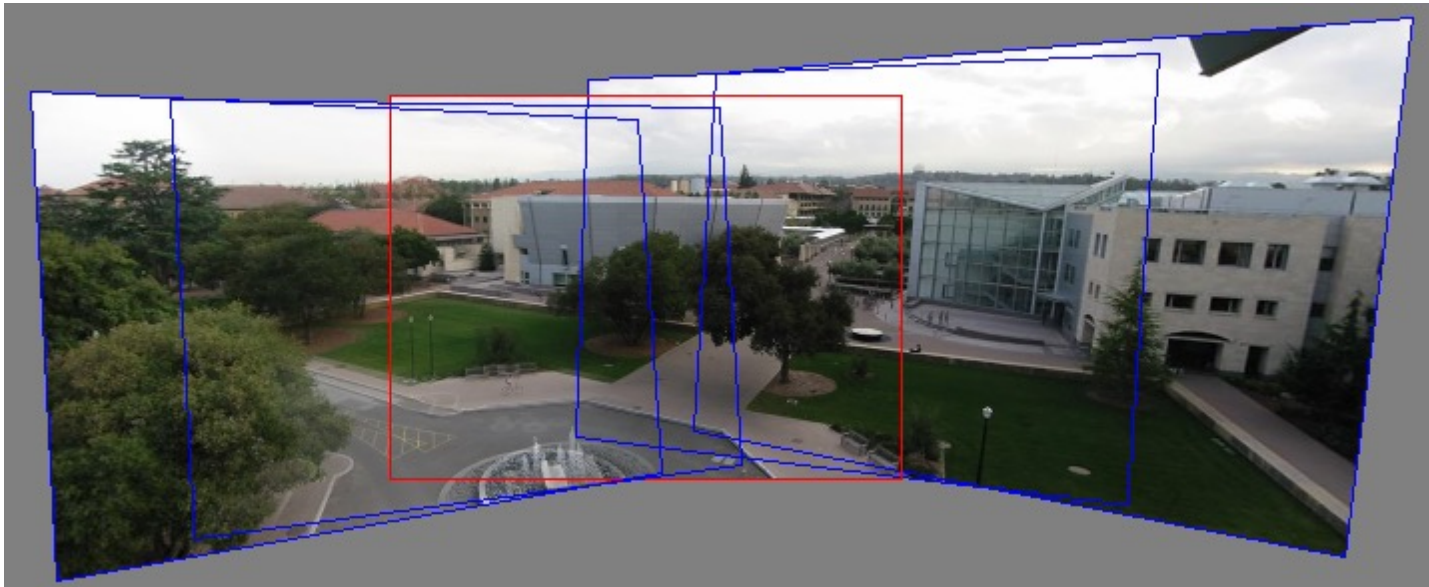
Rollout Photographs © Justin Kerr

<http://research.famsi.org/kerrmaya.html>

Also known as “cyclographs”, “peripheral images”

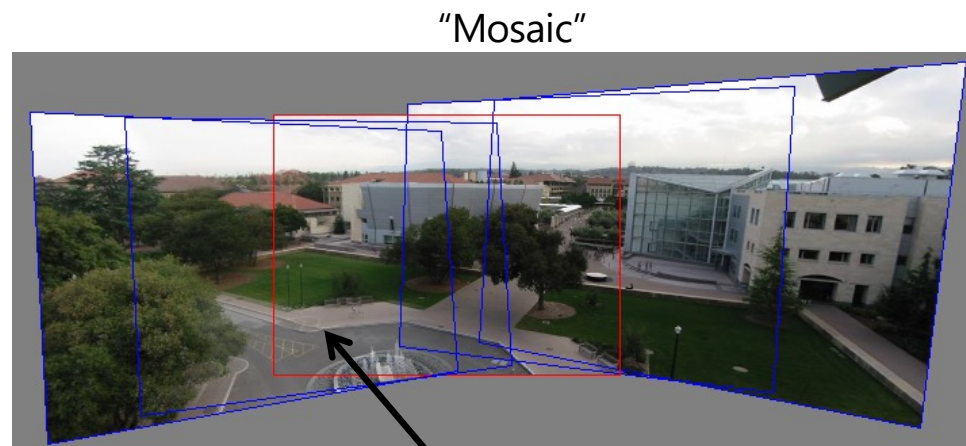
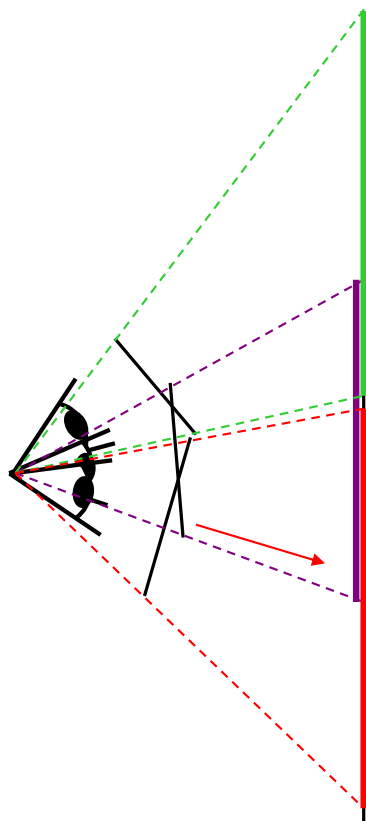
Questions?

Back to panoramas



Can we use homographies to create a 360 degree panorama?

Idea: project images onto a common plane



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

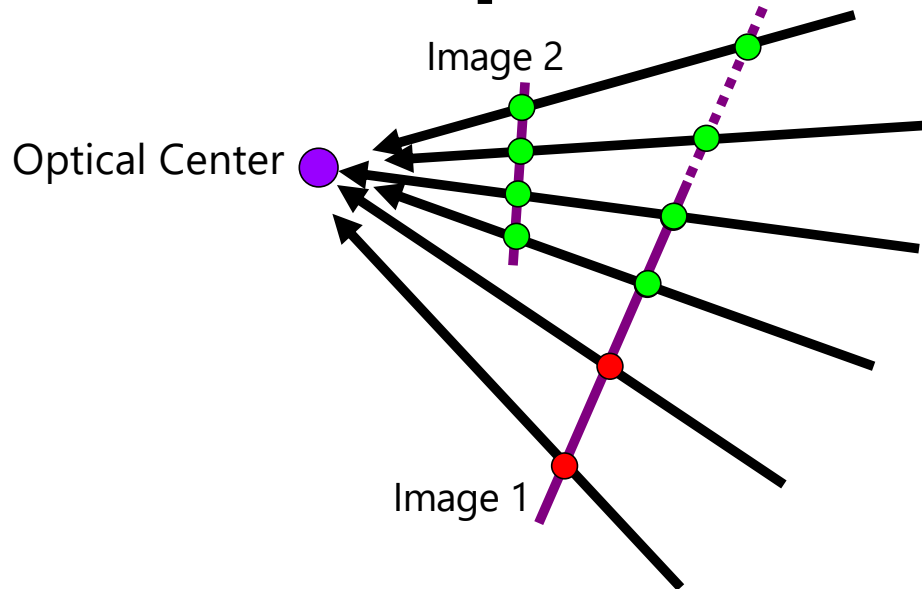
We'll see what this homography means next
Can't create a 360 panorama this way... we'll fix this shortly

mosaic projection plane

Creating a panorama

- Basic Procedure
 - Take a sequence of images from the same position
 - Rotate the camera about its optical center
 - Compute transformation between second image and first
 - Transform the second image to overlap with the first
 - Blend the two together to create a mosaic
 - If there are more images, repeat

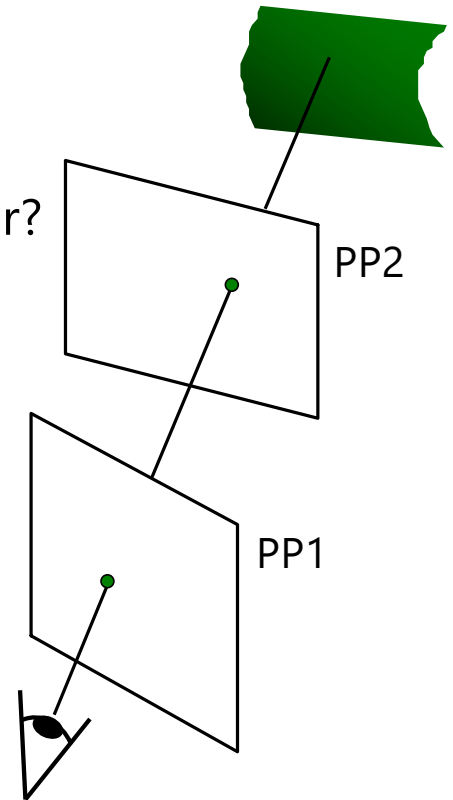
Geometric interpretation of mosaics



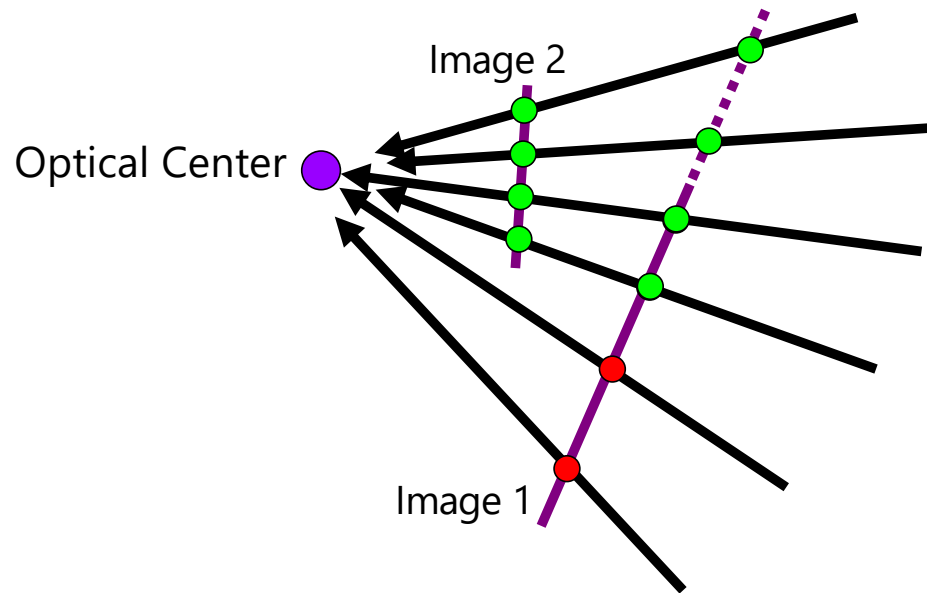
- If we capture all 360° of rays, we can create a 360° panorama
- The basic operation is *projecting* an image from one plane to another
- The projective transformation is scene-INDEPENDENT

Image reprojection

- Basic question
 - How to relate two images from the same camera center?
 - how to map a pixel from PP1 to PP2
- Answer
 - Cast a ray through each pixel in PP1
 - Draw the pixel where that ray intersects PP2



What is the transformation?



How do we map points in image 2 into image 1?

	image 1	image 2
intrinsics	\mathbf{K}_1	\mathbf{K}_2
extrinsics (rotation only)	$\mathbf{R}_1 = \mathbf{I}_{3 \times 3}$	\mathbf{R}_2

Step 1: Convert pixels in image 2 to rays in camera 2's coordinate system.

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

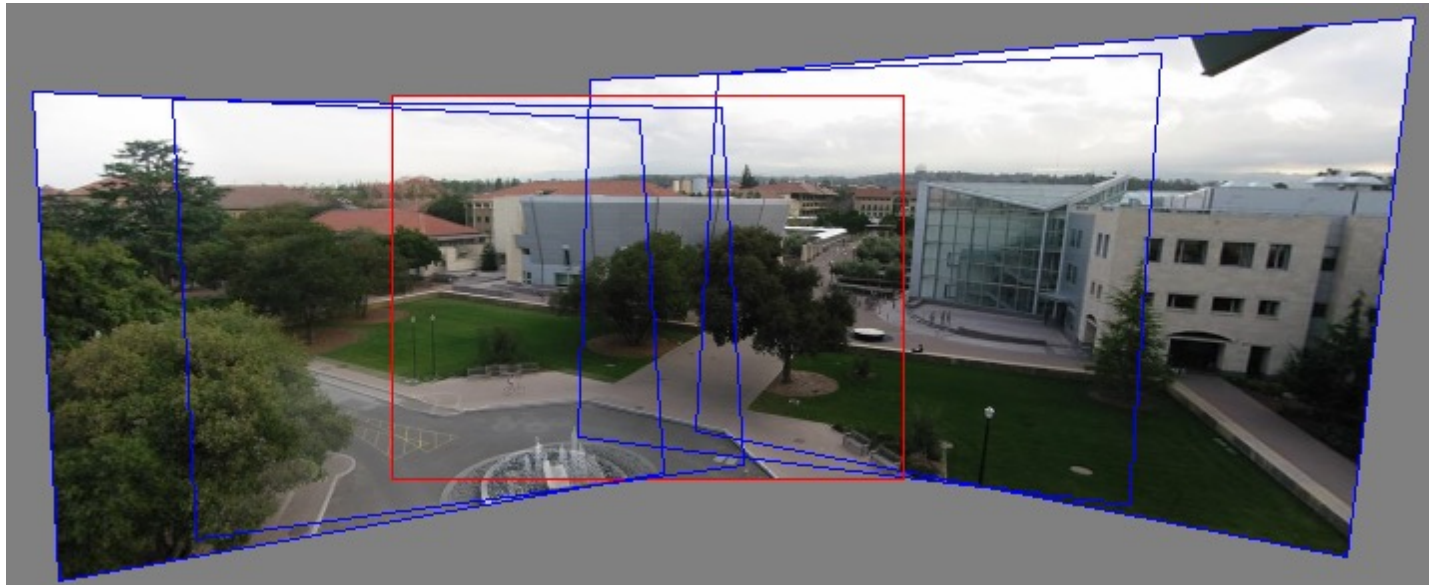
Step 2: Convert rays in camera 2's coordinates to rays in camera 1's coordinates.

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathbf{R}_2^T \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

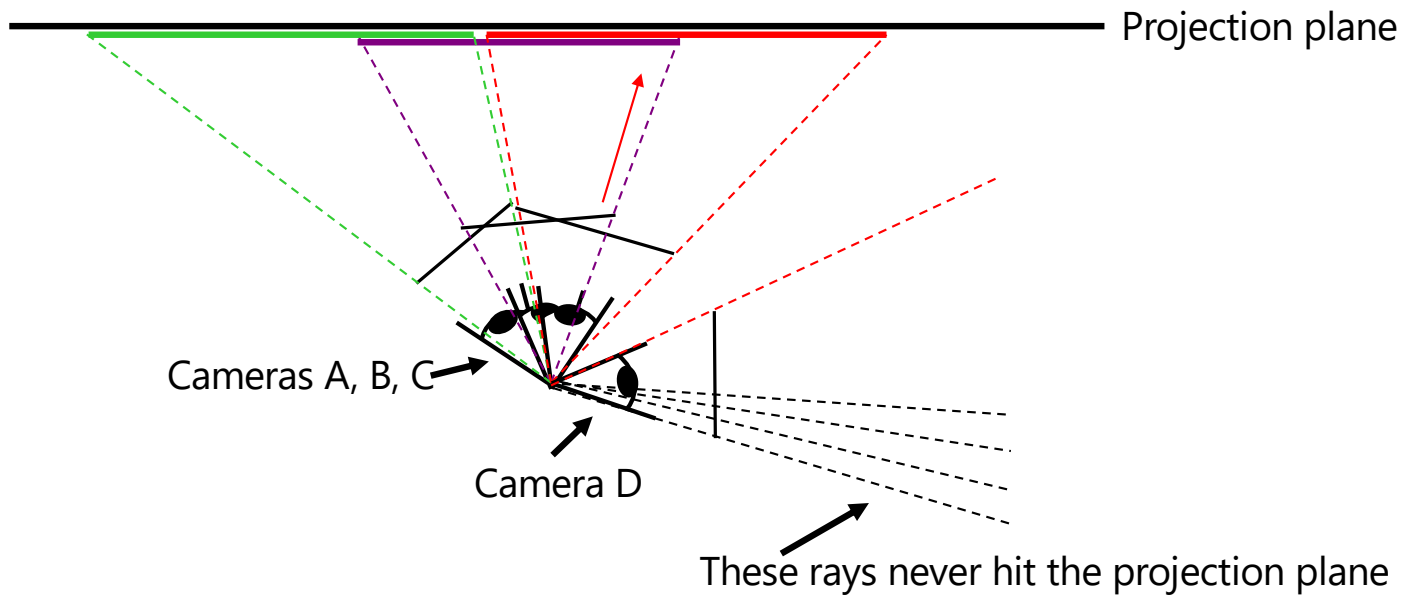
Step 3: Convert rays in camera 1's coordinates to pixels in image 1's coordinates.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \underbrace{\mathbf{K}_1 \mathbf{R}_2^T \mathbf{K}_2^{-1}}_{\substack{\uparrow \\ \text{3x3 homography}}} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

**Can we use homography to create a 360
panorama?**

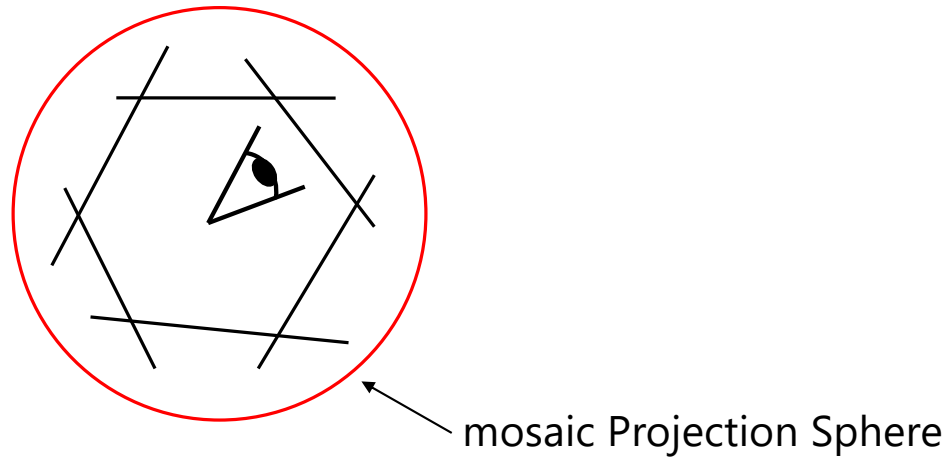


Answer: No

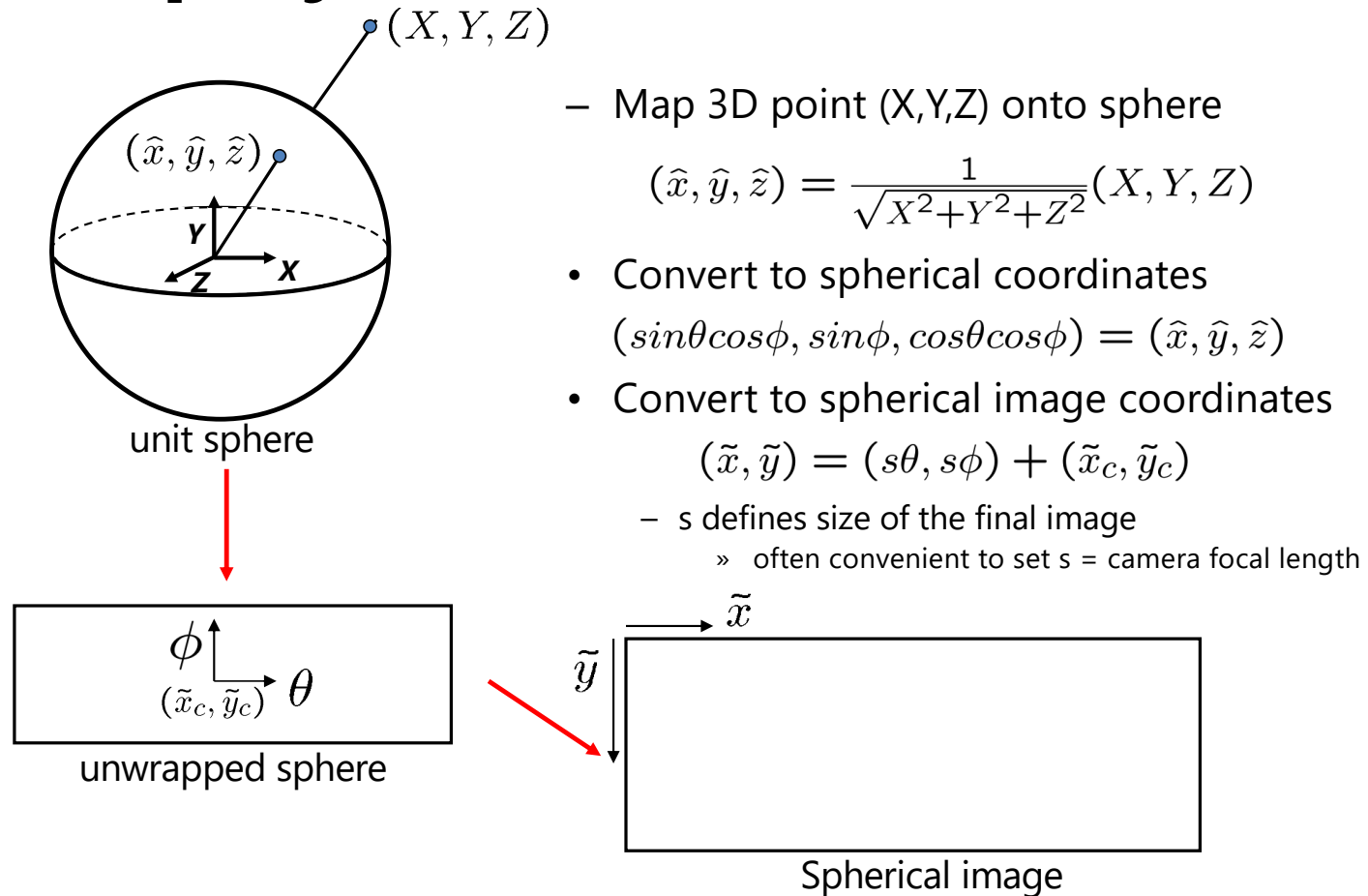


Panoramas

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



Spherical projection



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

- Convert to spherical image coordinates

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

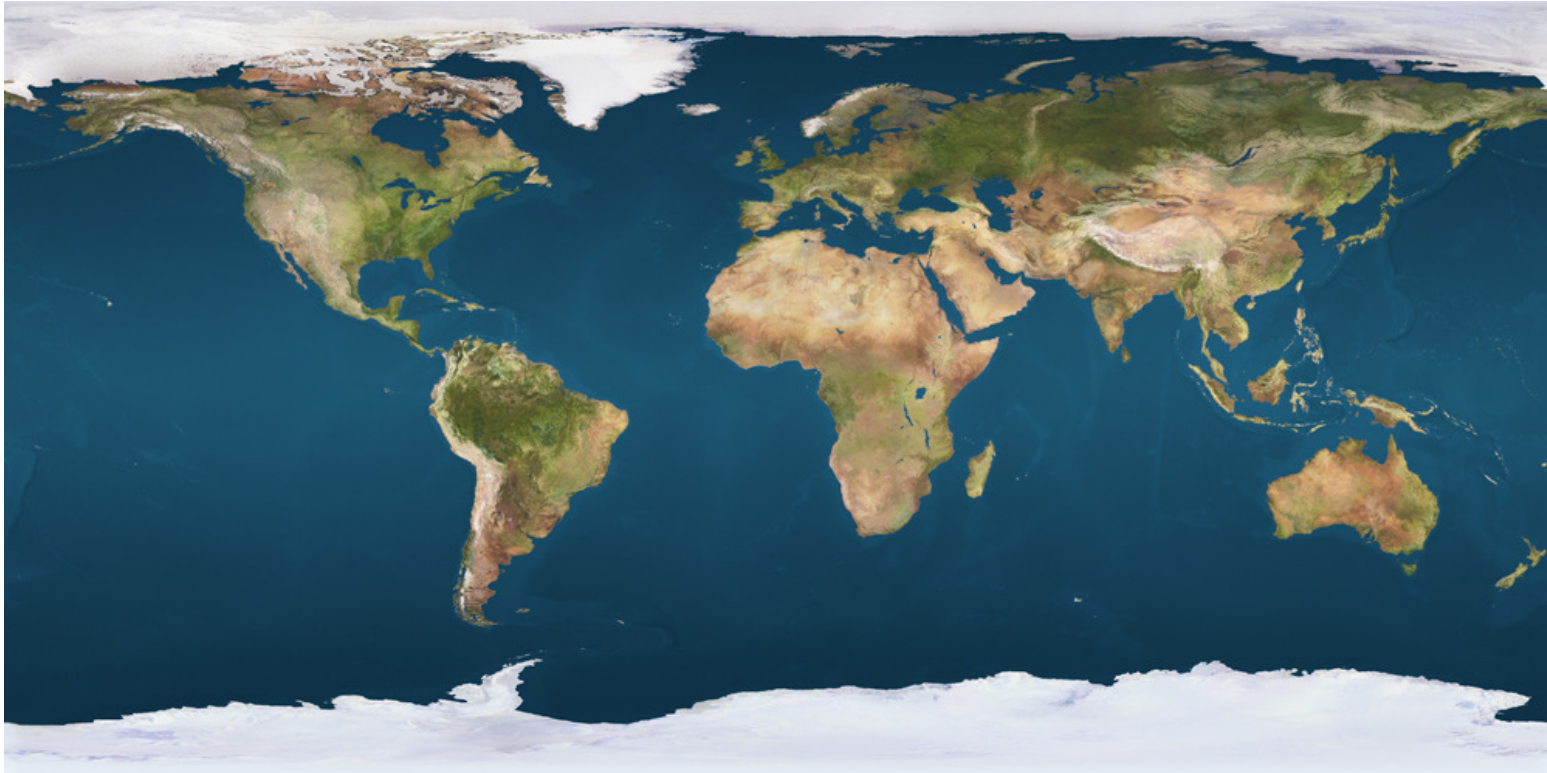
- s defines size of the final image

» often convenient to set $s =$ camera focal length

Unwrapping a sphere



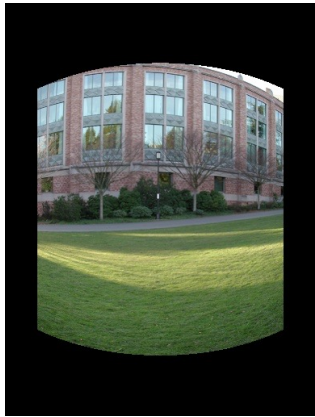
Credit: JHT's Planetary Pixel Emporium



Spherical reprojection



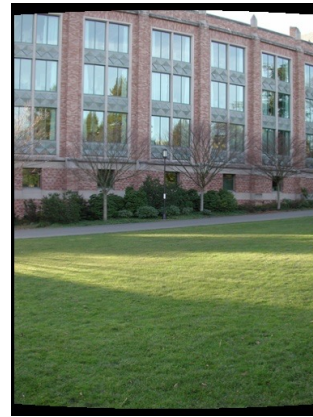
input



$f = 200$ (pixels)



$f = 400$



$f = 800$

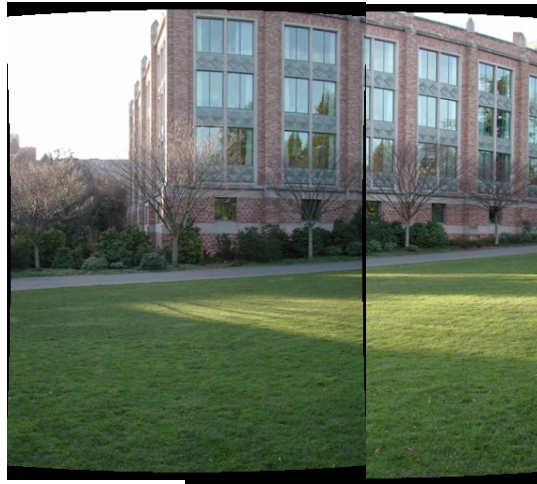
- Map image to spherical coordinates
 - need to know the focal length

Aligning spherical images



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

Aligning spherical images



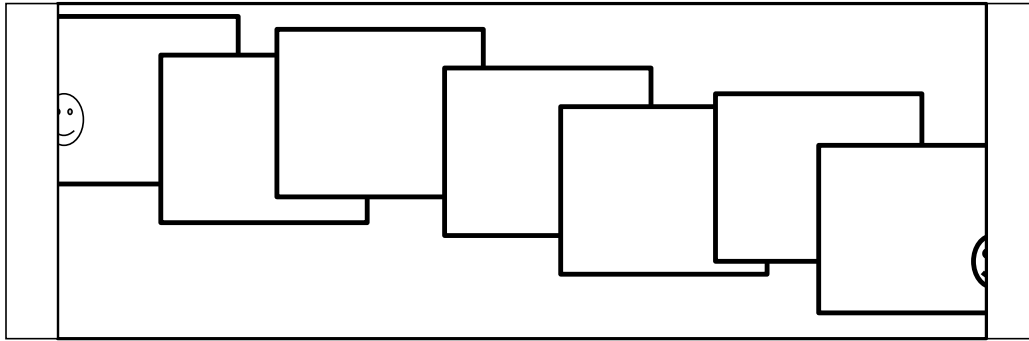
- Suppose we rotate the camera by θ about the vertical axis
 - How does this change the spherical image?
 - Translation by θ
 - This means that we can align spherical images by translation

Assembling the panorama



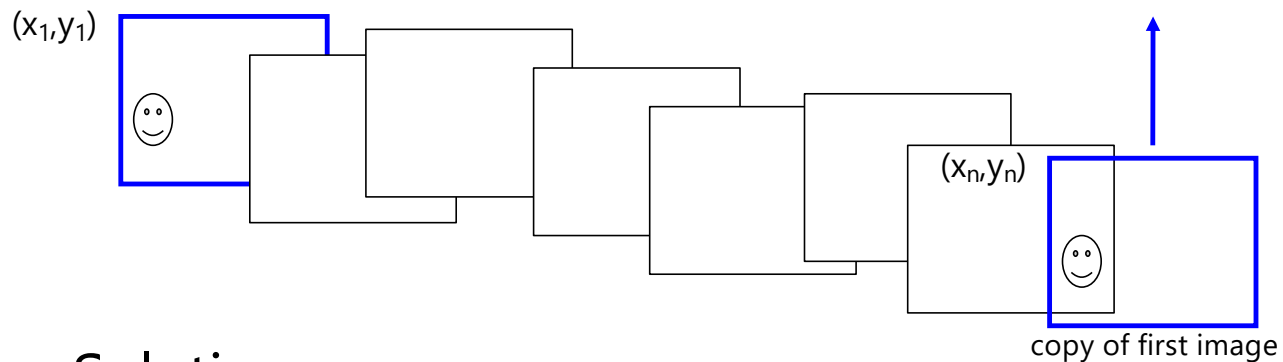
- Stitch pairs together, blend, then crop

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: $\mathbf{y}' = \mathbf{y} + \mathbf{ax}$ [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

1. Take pictures on a tripod (or handheld)
 2. Warp to spherical coordinates (not needed if using homographies to align images)
 3. Extract features
 4. Align neighboring pairs using feature matching + RANSAC
 5. Write out list of neighboring translations
 6. Correct for drift
 7. Read in warped images and blend them
 8. 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?

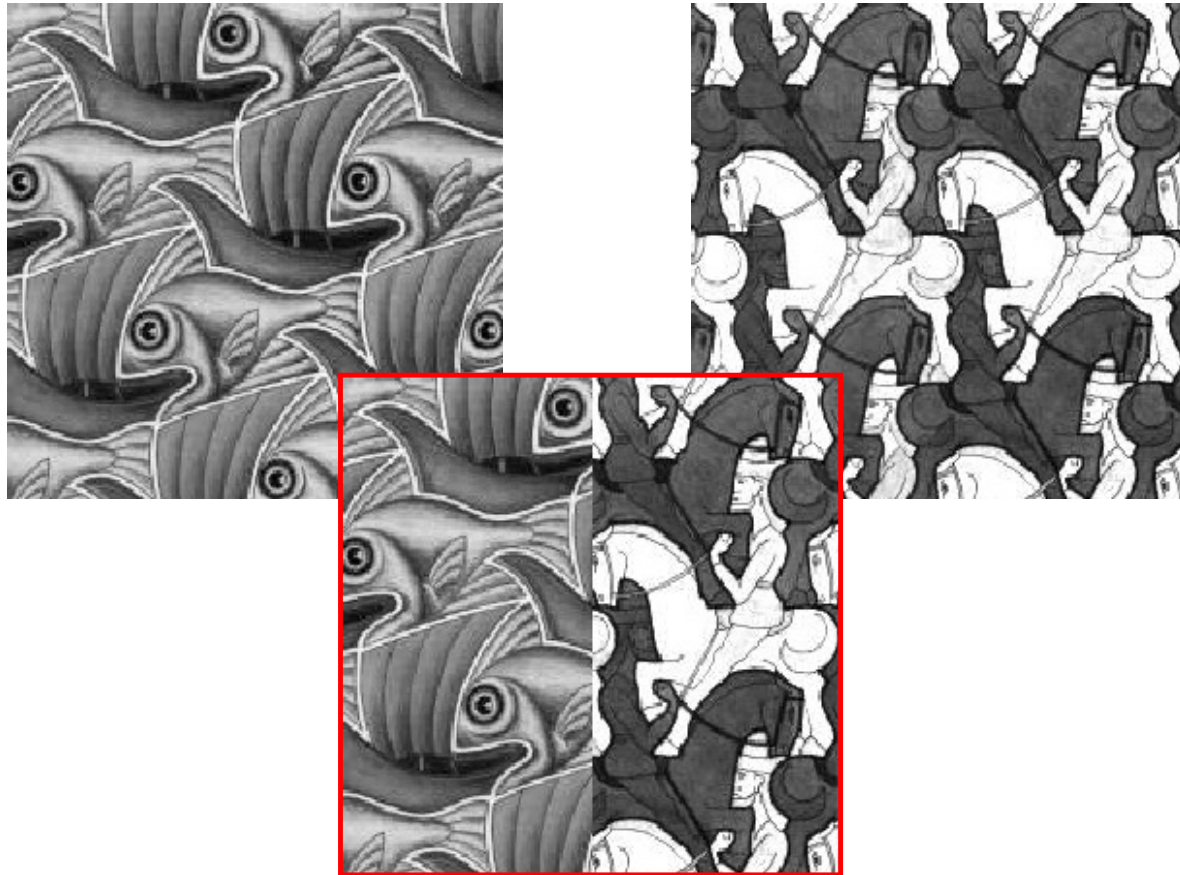


Blending

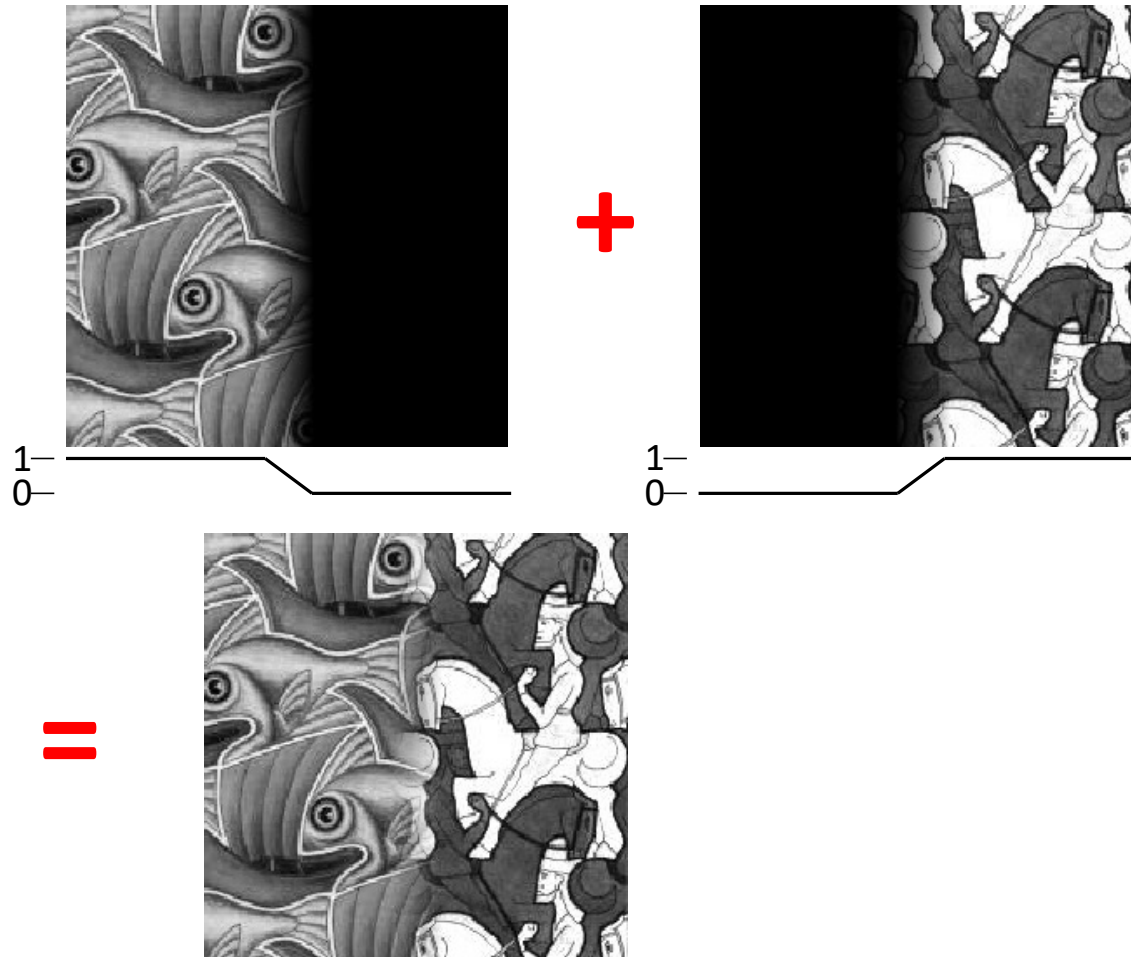
- Want to seamlessly blend them together



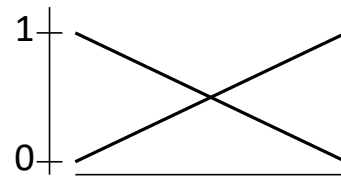
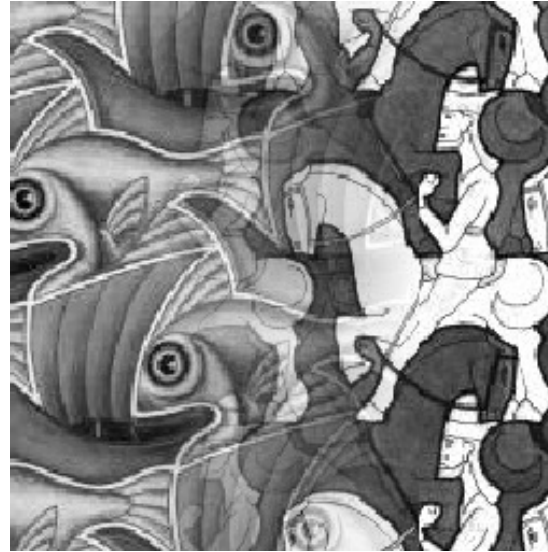
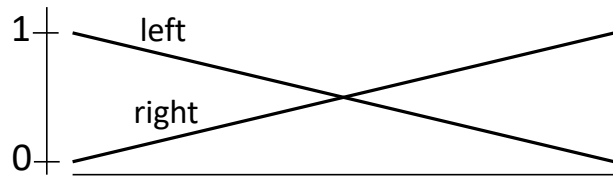
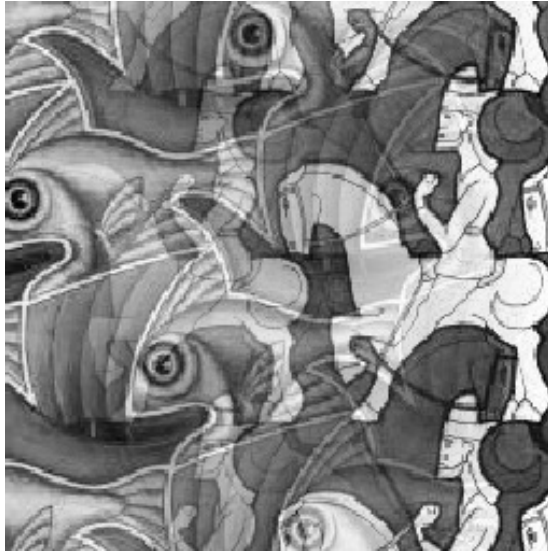
Image Blending



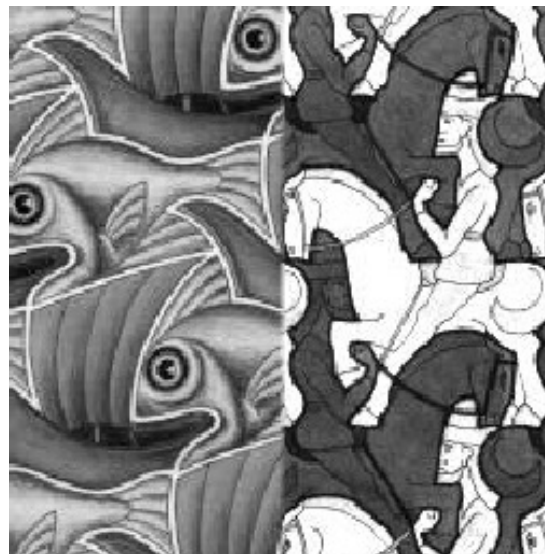
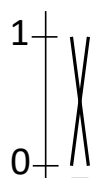
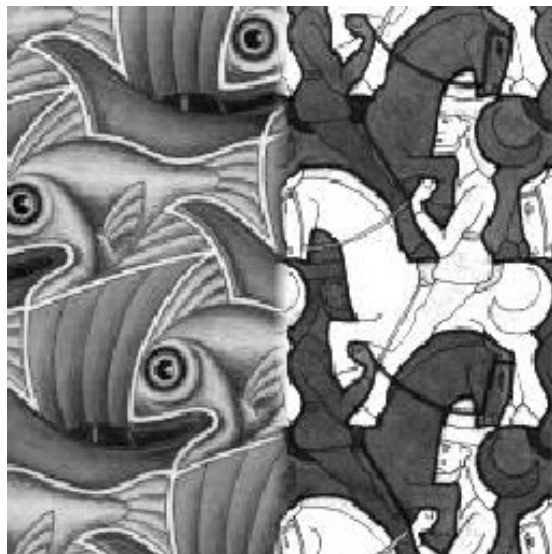
Feathering



Effect of window size



Effect of window size



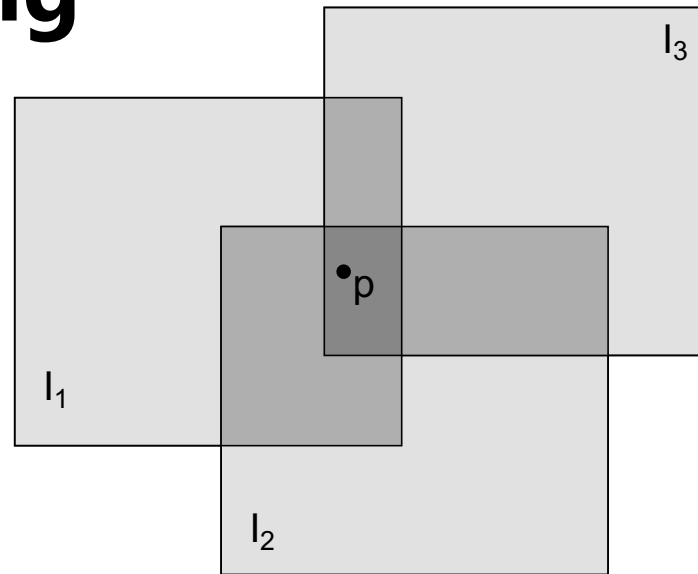
Good window size



“Optimal” window: smooth but not ghosted

- Doesn't always work...

Alpha Blending



see Blinn (CGA, 1994) for details:

[Compositing, Part 1: Theory](#)

Encoding blend weights: $I(x,y) = (\alpha R, \alpha G, \alpha B, \alpha)$

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) $RGB\alpha$ values at each pixel
2. normalize: divide each pixel's accumulated RGB by its α value

Q: what if $\alpha = 0$?

Poisson Image Editing



For more info: [Perez et al, SIGGRAPH 2003](#)

Some panorama examples



"Before SIGGRAPH Deadline" Photo credit: Doug Zongker

Some panorama examples

- Every image on Google Streetview



Magic: ghost removal



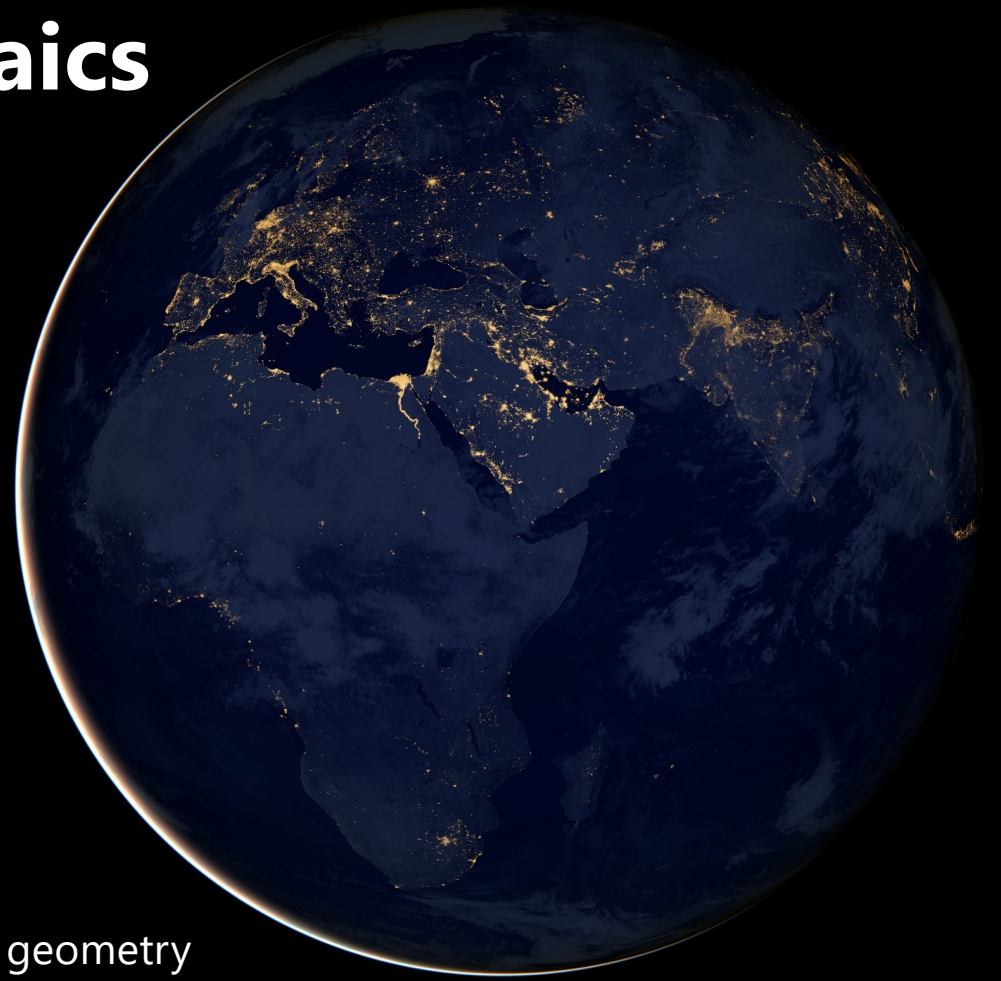
M. Uyttendaele, A. Eden, and R. Szeliski.
Eliminating ghosting and exposure artifacts in image mosaics.
ICCV 2001

Magic: ghost removal

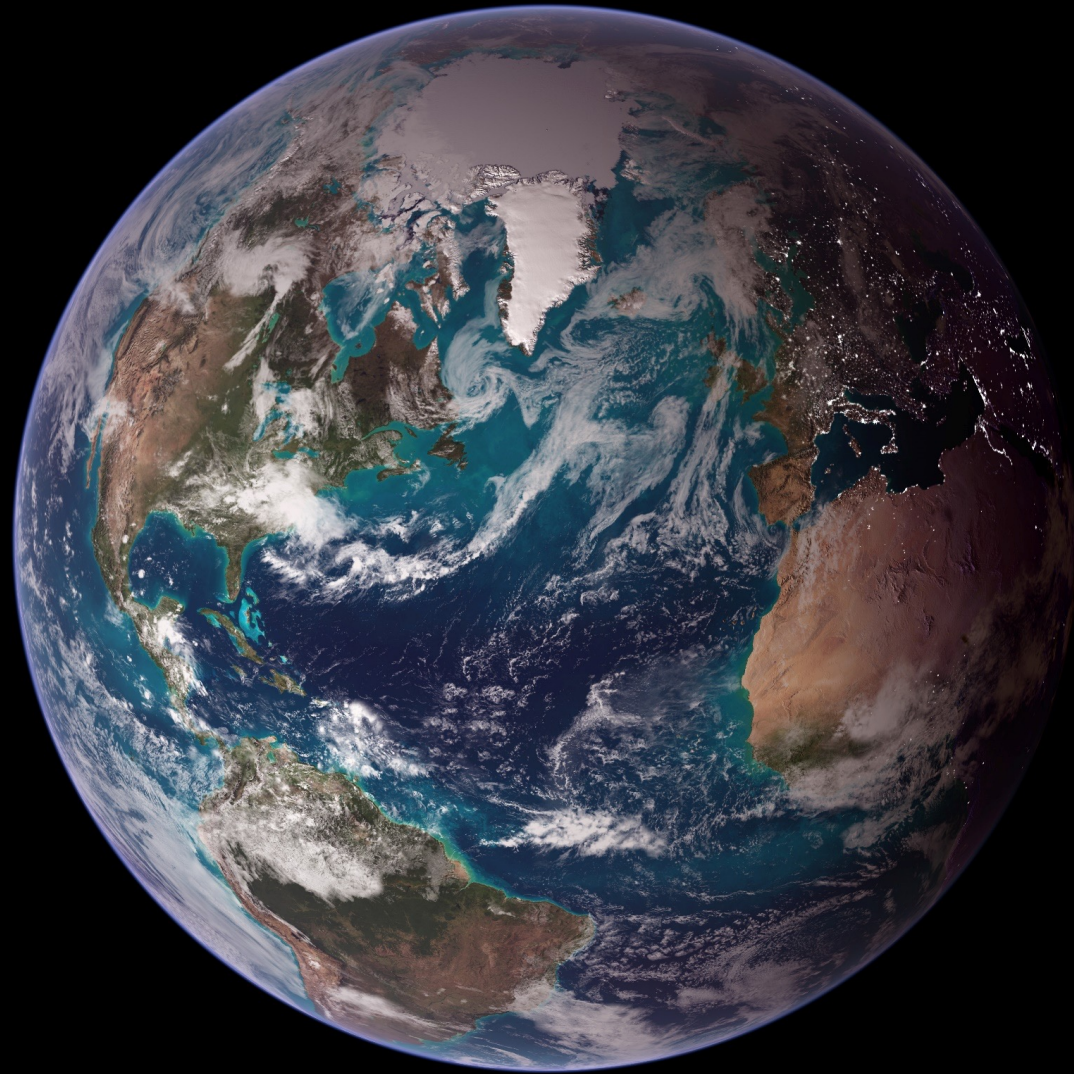


M. Uyttendaele, A. Eden, and R. Szeliski.
Eliminating ghosting and exposure artifacts in image mosaics.
ICCV 2001

Other types of mosaics



- Can mosaic onto *any* surface if you know the geometry
 - See NASA's [Visible Earth project](#) for some stunning earth mosaics





https://www.nasa.gov/centers/wallops/news/frozen_sos.html

Science on a Sphere



[https://news.vcu.edu/article/Science On a Sphere now at VCU offers a world of possibilities](https://news.vcu.edu/article/Science%20On%20a%20Sphere%20now%20at%20VCU%20offers%20a%20world%20of%20possibilities)

Questions?

Project 3 Demo