CS6670: Computer Vision

Noah Snavely

Lecture 6: Image transformations and alignment

Announcement

- New TA! Adarsh Kowdle

- Office hours: M 11-12, Ward Laboratory 112

Announcements

- Project 1 out, due Thursday, 9/24, by 11:59pm
- Quiz on Thursday, first 10 minutes of class
- Next week: guest lecturer, Prof. Pedro Felzenszwalb, U. Chicago

Announcements

- Project 2 will be released on Tuesday

- You can work in groups of two
  - Send me your groups by Friday evening

Readings

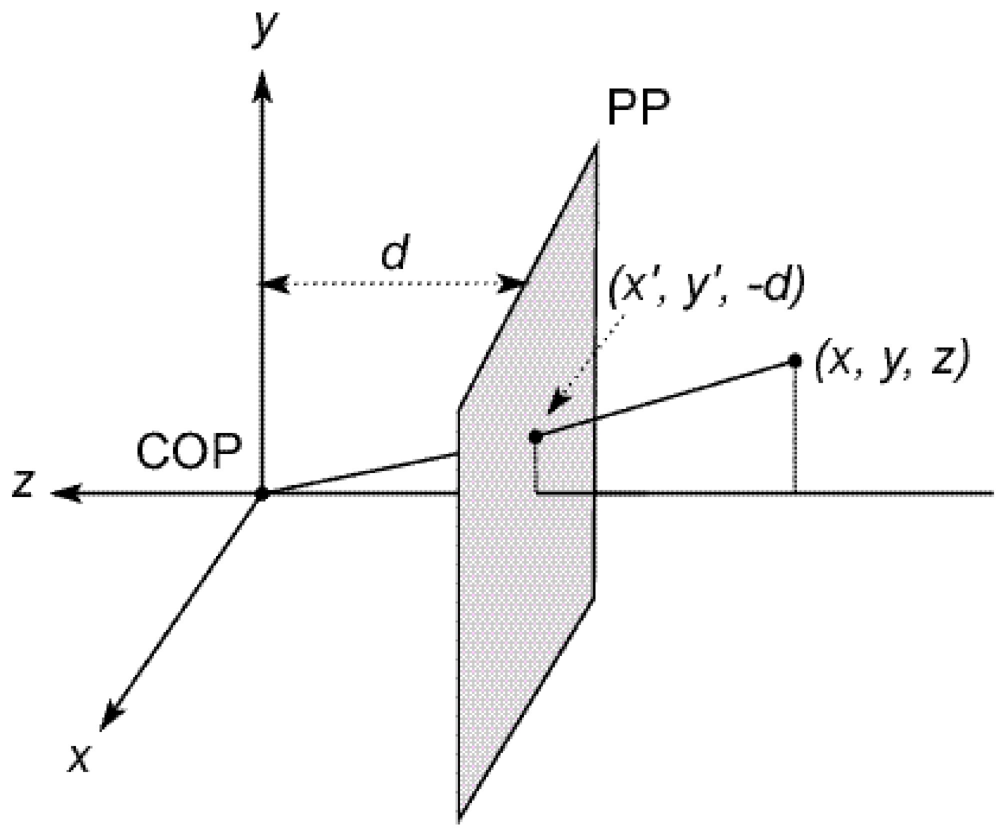
- Szeliski Chapter 3.5 (image warping), 9.1 (motion models)

Announcements

- A total of 3 late days will be allowed for projects

Project 1 questions

Last time: projection



$$(x, y, z) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z},\ -d) \rightarrow (-d\frac{x}{z},\ -d\frac{y}{z})$$

Perspective projection

Projection is a matrix multiply using homogeneous coordinates:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

divide by third coordinate

Equivalent to:

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \Rightarrow (-d\frac{x}{z}, \quad -d\frac{y}{z})$$

Perspective projection

$$\underbrace{\begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\mathbf{K} \\ \text{(intrinsics)}}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(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)

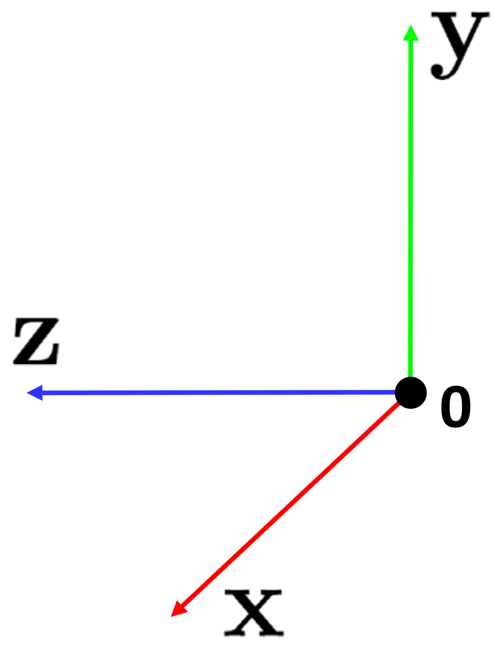
s : **skew** (0 unless pixels are shaped like rhombi/parallelograms)

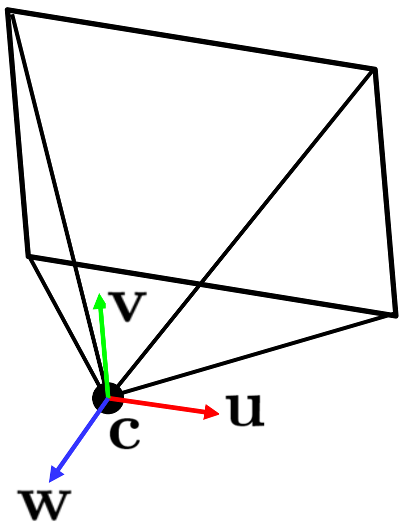
(c_x, c_y) : **principal point** ((0,0) unless optical axis doesn’t intersect projection plane at origin)

Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)

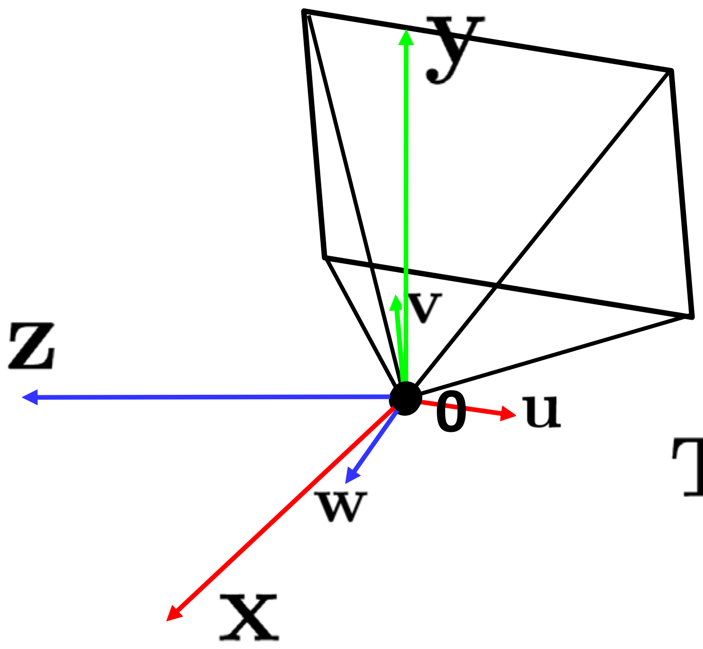
Step 1: Translate by -**c**





Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



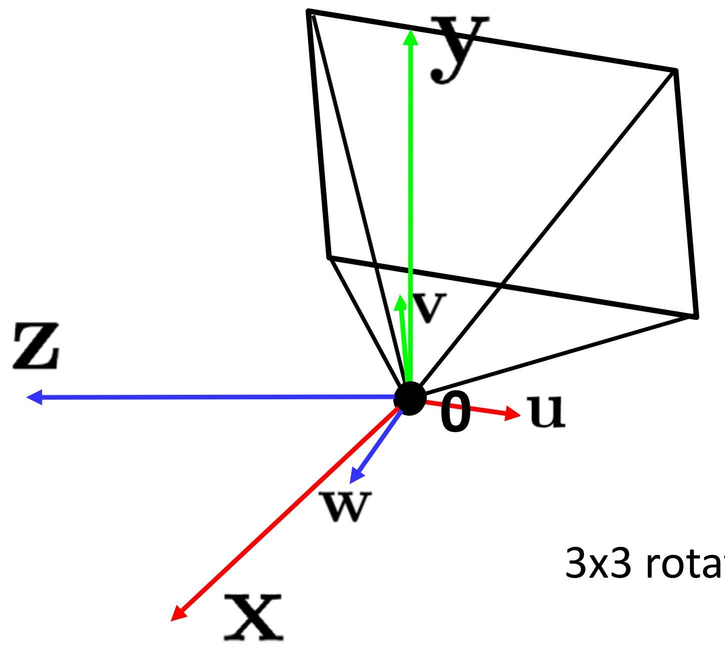
Step 1: Translate by -**c**

How do we represent translation as a matrix multiplication?

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

Extrinsics

- How do we get the camera to “canonical form”?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

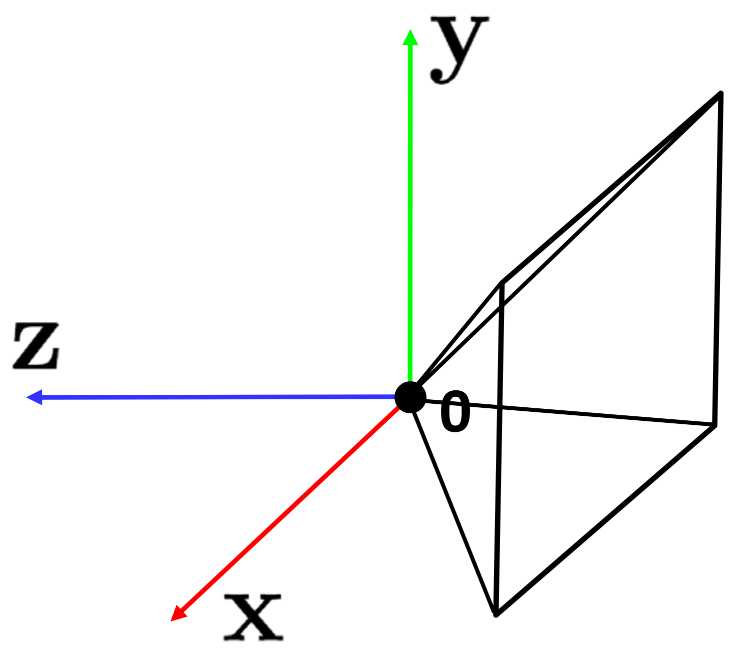
Step 2: Rotate by **R**

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

3x3 rotation matrix

Extrinsics

- How do we get the camera to "canonical form"?
  - (Center of projection at the origin, x-axis points right, y-axis points up, z-axis points backwards)



Step 1: Translate by -**c**

Step 2: Rotate by **R**

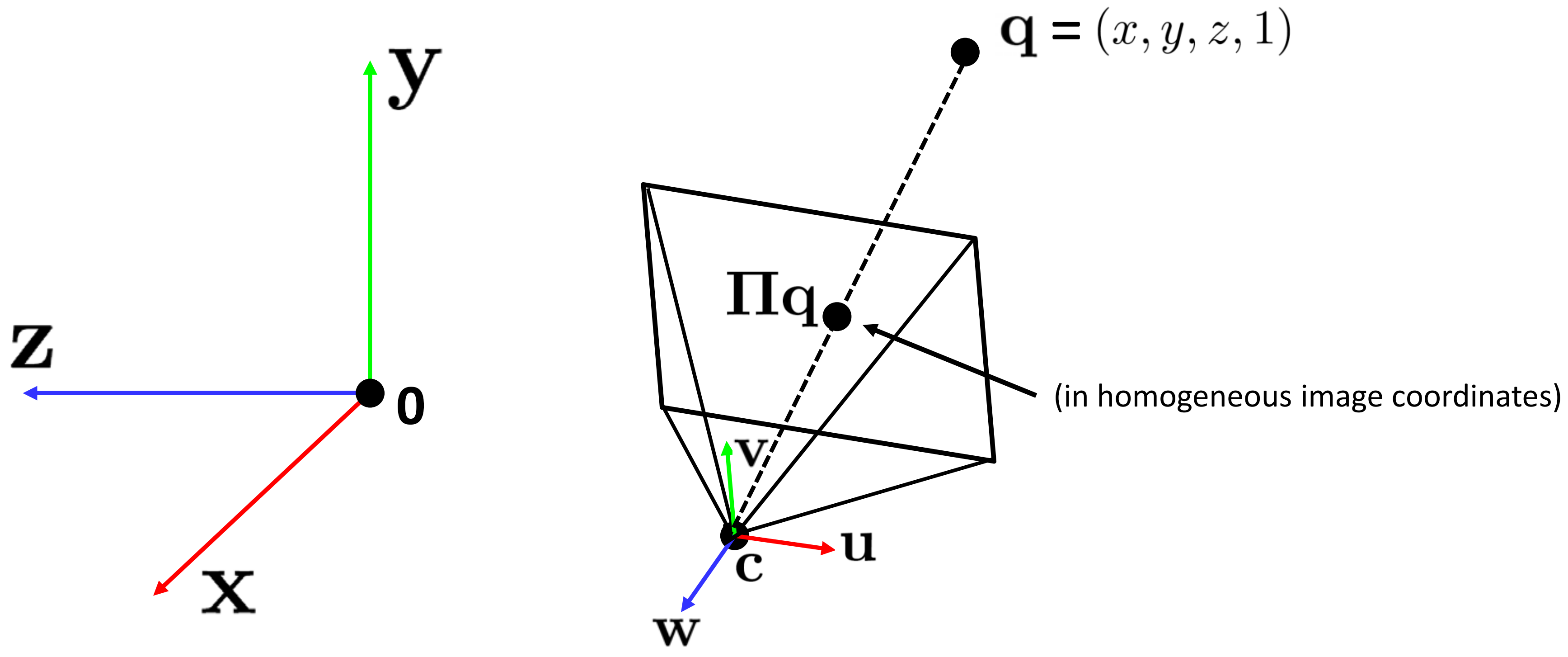
$$\mathbf{R} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix}$$

Projection matrix

$$\mathbf{\Pi} = \underset{\text{intrinsics}}{\mathbf{K}} \underbrace{\underset{\text{projection}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}} \underset{\text{rotation}}{\begin{bmatrix} & \mathbf{R} & & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{\text{translation}}{\begin{bmatrix} & \mathbf{I}_{3\times 3} & & -\mathbf{c} \\ 0 & 0 & 0 & 1 \end{bmatrix}}}_{\left[\ \mathbf{R}\ \middle|\ \underbrace{-\mathbf{Rc}}_{(\textbf{t} \text{ in book's notation})}\ \right]}$$

$$\mathbf{\Pi} = \mathbf{K} \left[\ \mathbf{R}\ \middle|\ -\mathbf{Rc}\ \right]$$

Projection matrix



Perspective distortion

- What does a sphere project to?

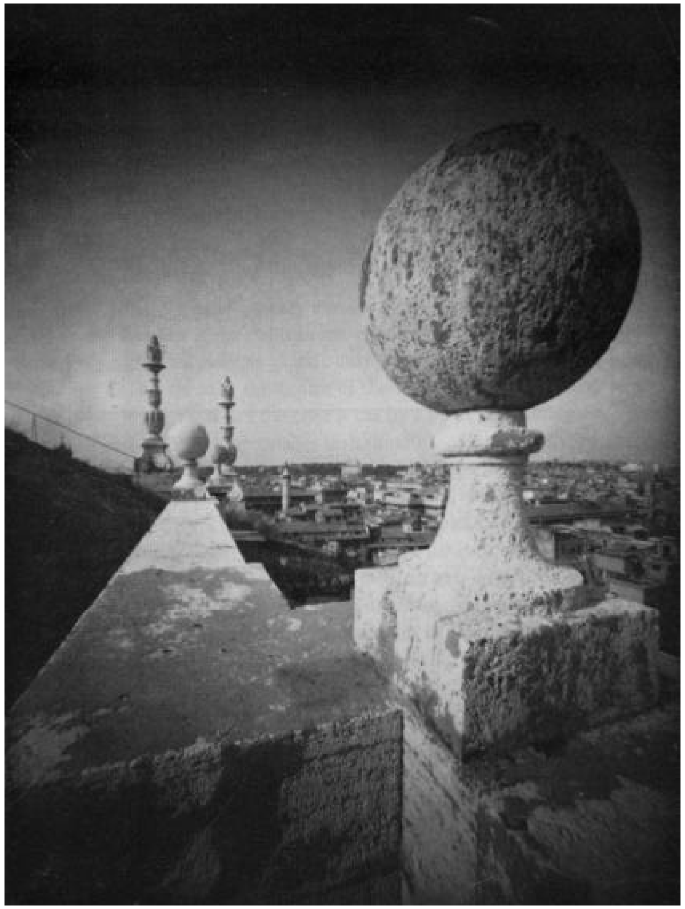
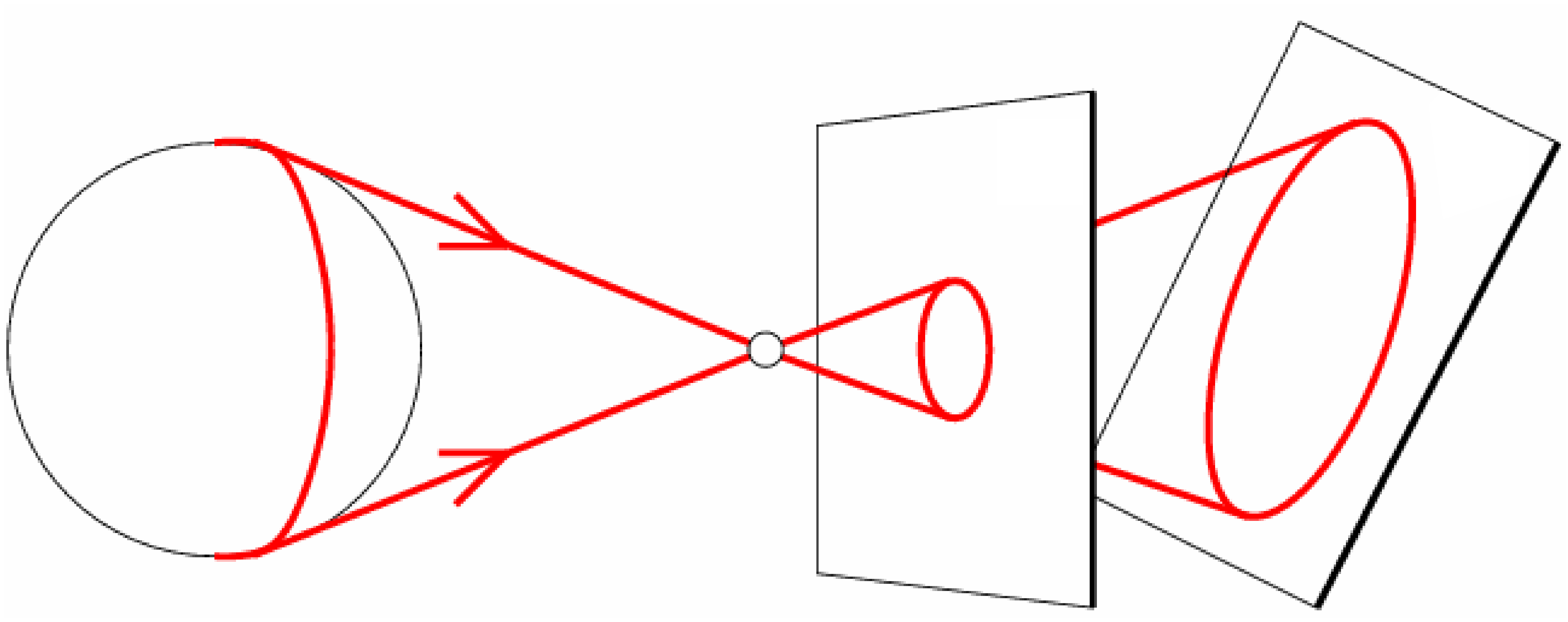


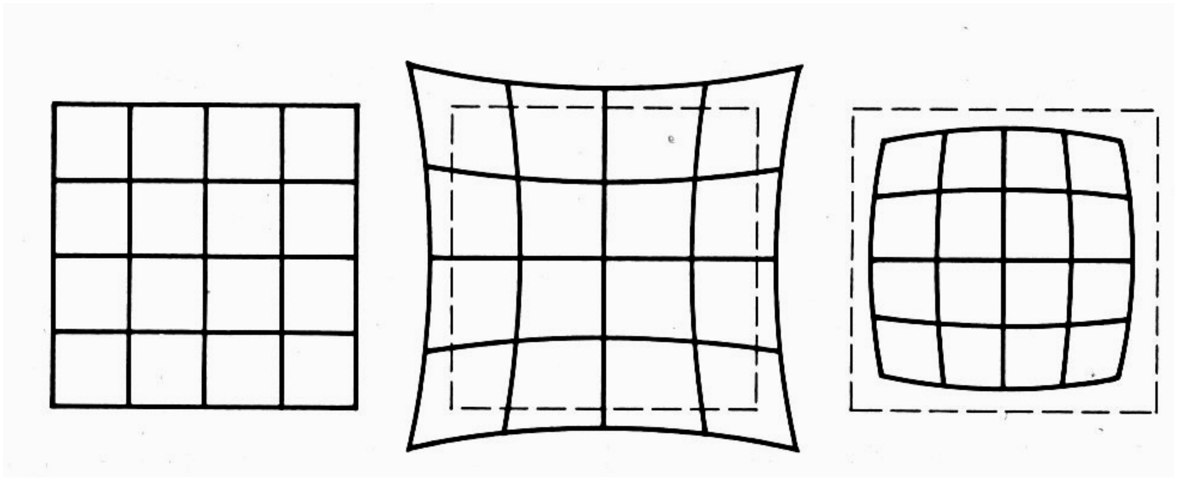
Image source: F. Durand

Perspective distortion

- What does a sphere project to?



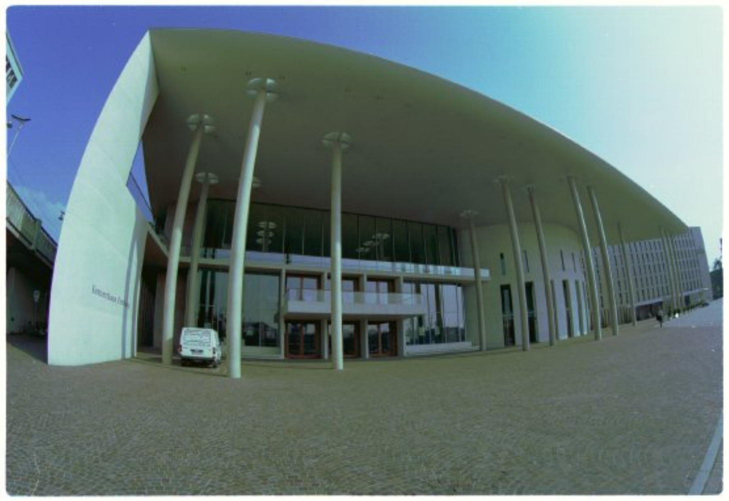
Distortion

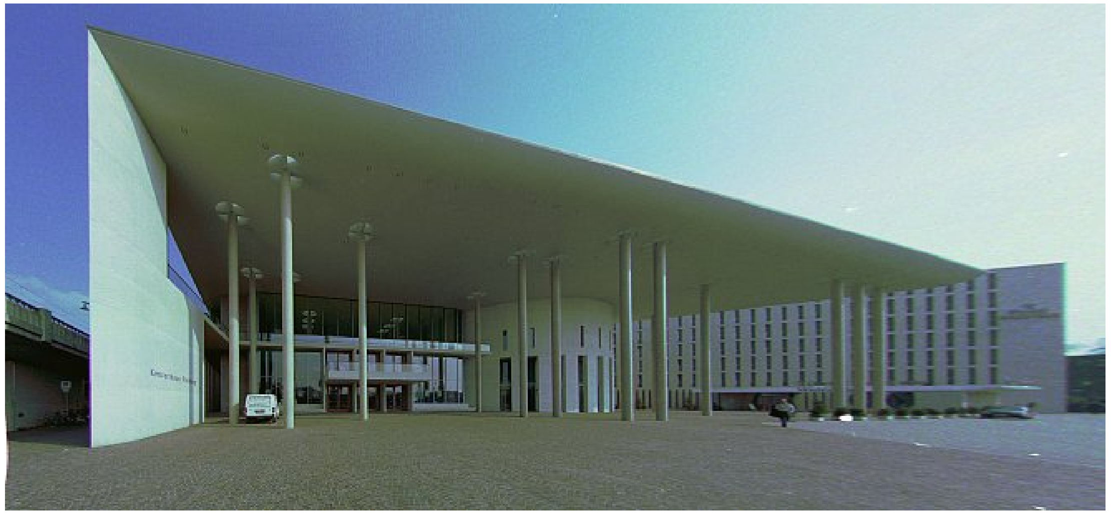


No distortion Pin cushion Barrel

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

Correcting radial distortion





from Helmut Dersch

Modeling distortion

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

$$\begin{aligned} x'_n &= \hat{x}/\hat{z} \\ y'_n &= \hat{y}/\hat{z} \end{aligned}$$

Apply radial distortion

$$\begin{aligned} r^2 &= {x'_n}^2 + {y'_n}^2 \\ x'_d &= x'_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \\ y'_d &= y'_n(1 + \kappa_1 r^2 + \kappa_2 r^4) \end{aligned}$$

Apply focal length translate image center

$$\begin{aligned} x' &= f x'_d + x_c \\ y' &= f y'_d + y_c \end{aligned}$$

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

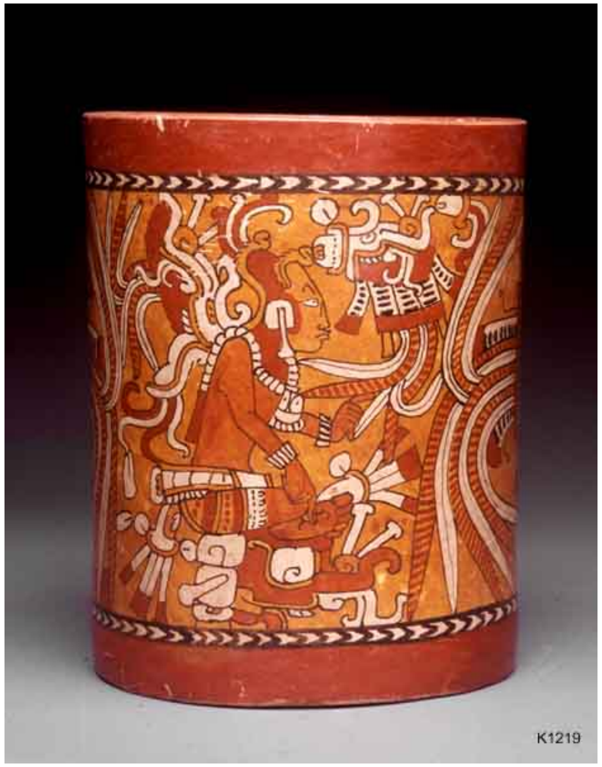
Other types of projection

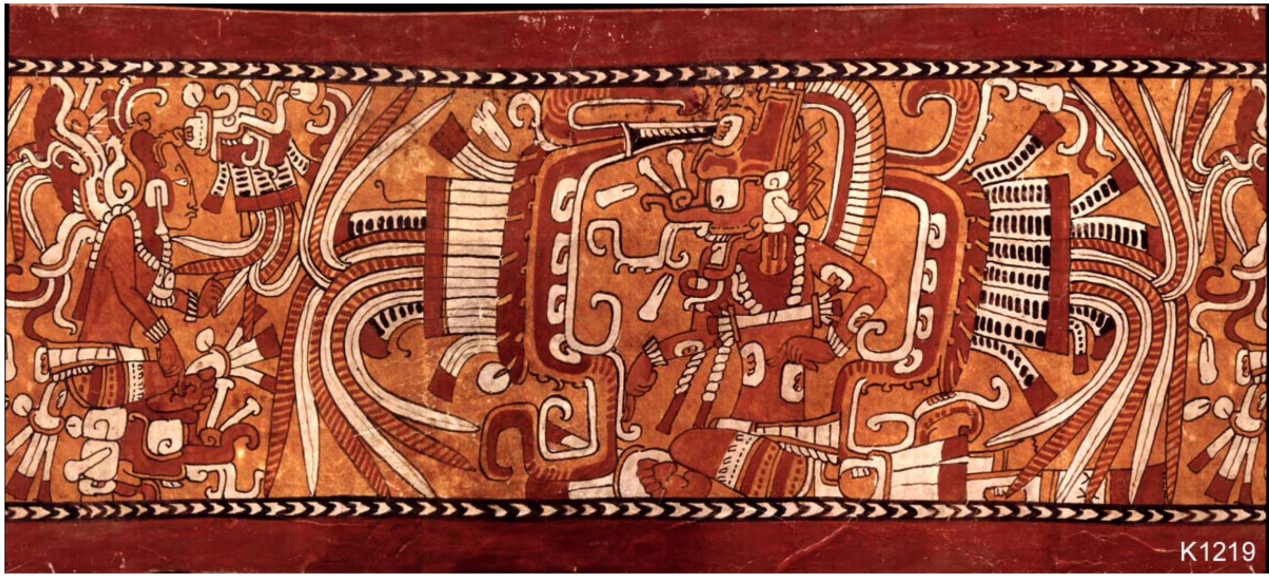
- Lots of intriguing variants…
- (I’ll just mention a few fun ones)

360 degree field of view…

- 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

Rotating sensor (or object)

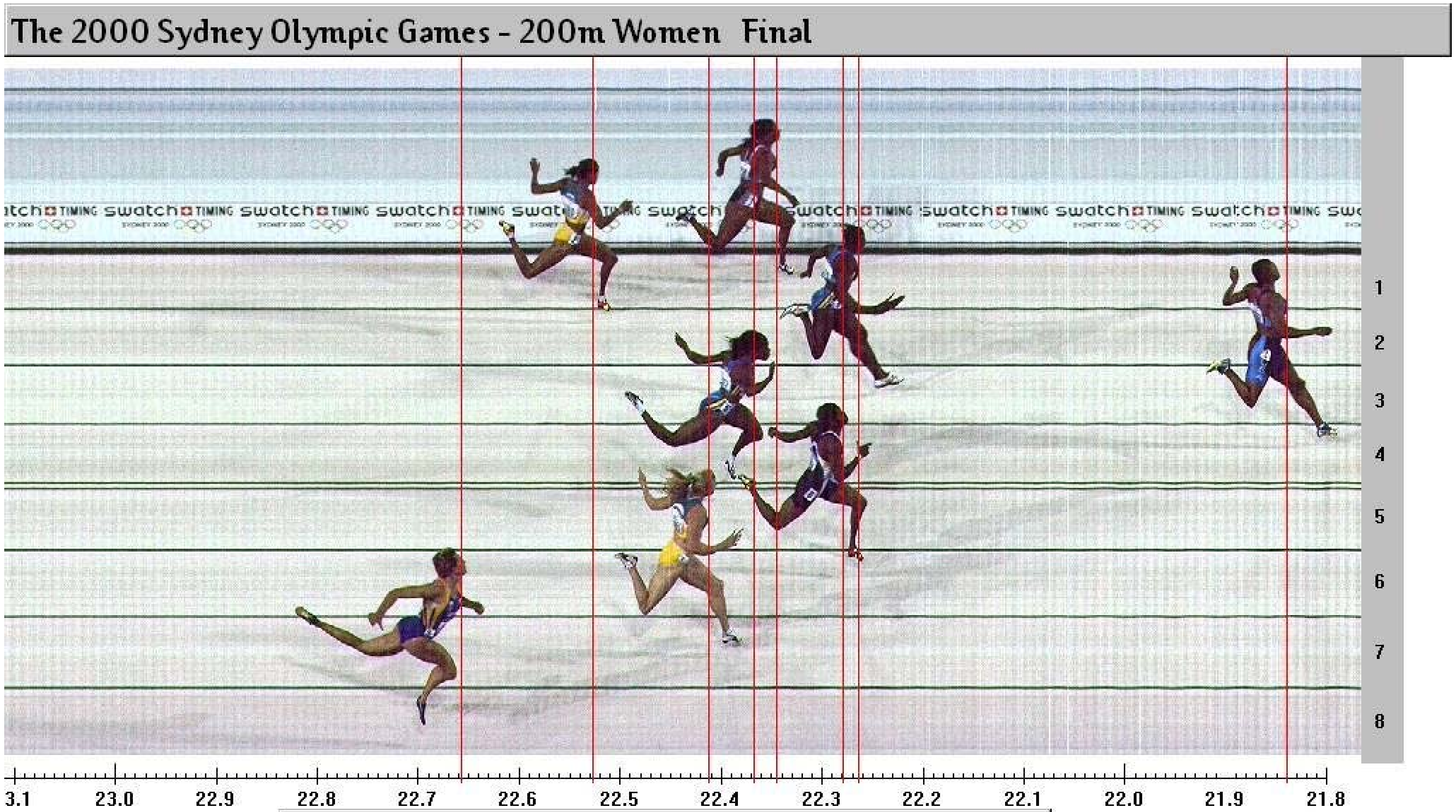


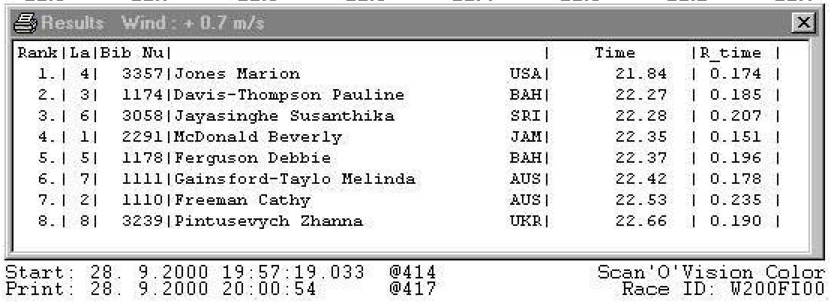


Rollout Photographs © Justin Kerr
http://research.famsi.org/kerrmaya.html

Also known as “cyclographs”, “peripheral images”

Photofinish





```
Rank|La|Bib Nu|                              |      Time    |R_time |
  1.| 4|  3357|Jones Marion              USA|      21.84   | 0.174 |
  2.| 3|  1174|Davis-Thompson Pauline    BAH|      22.27   | 0.185 |
  3.| 6|  3058|Jayasinghe Susanthika     SRI|      22.28   | 0.207 |
  4.| 1|  2291|McDonald Beverly          JAM|      22.35   | 0.151 |
  5.| 5|  1178|Ferguson Debbie           BAH|      22.37   | 0.196 |
  6.| 7|  1111|Gainsford-Taylo Melinda   AUS|      22.42   | 0.178 |
  7.| 2|  1110|Freeman Cathy             AUS|      22.53   | 0.235 |
  8.| 8|  3239|Pintusevych Zhanna        UKR|      22.66   | 0.190 |

Start:  28.  9.2000 19:57:19.033  @414            Scan'O'Vision Color
Print:  28.  9.2000 20:00:54      @417              Race ID:  W200FI00
```

TM © SOCOG 1996

Questions?

Today: Image transformations and alignment

Full screen panoramas (cubic): http://www.panoramas.dk/
Mars: http://www.panoramas.dk/fullscreen3/f2_mars97.html
2003 New Years Eve: http://www.panoramas.dk/fullscreen3/f1.html

Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°

Slide from Brown & Lowe

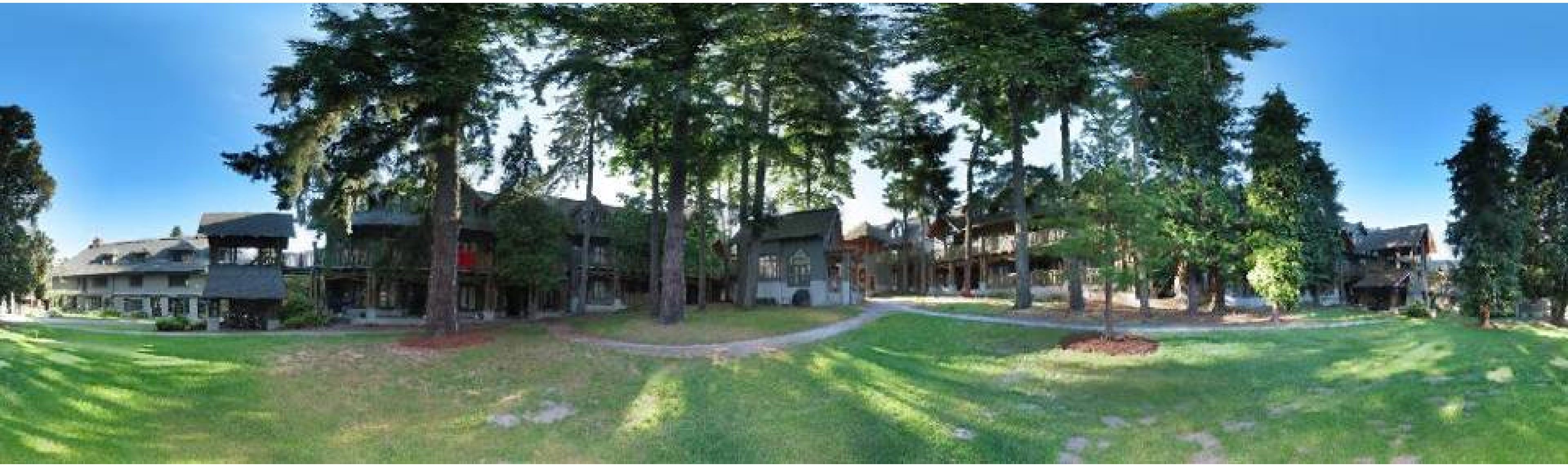
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°

Slide from Brown & Lowe

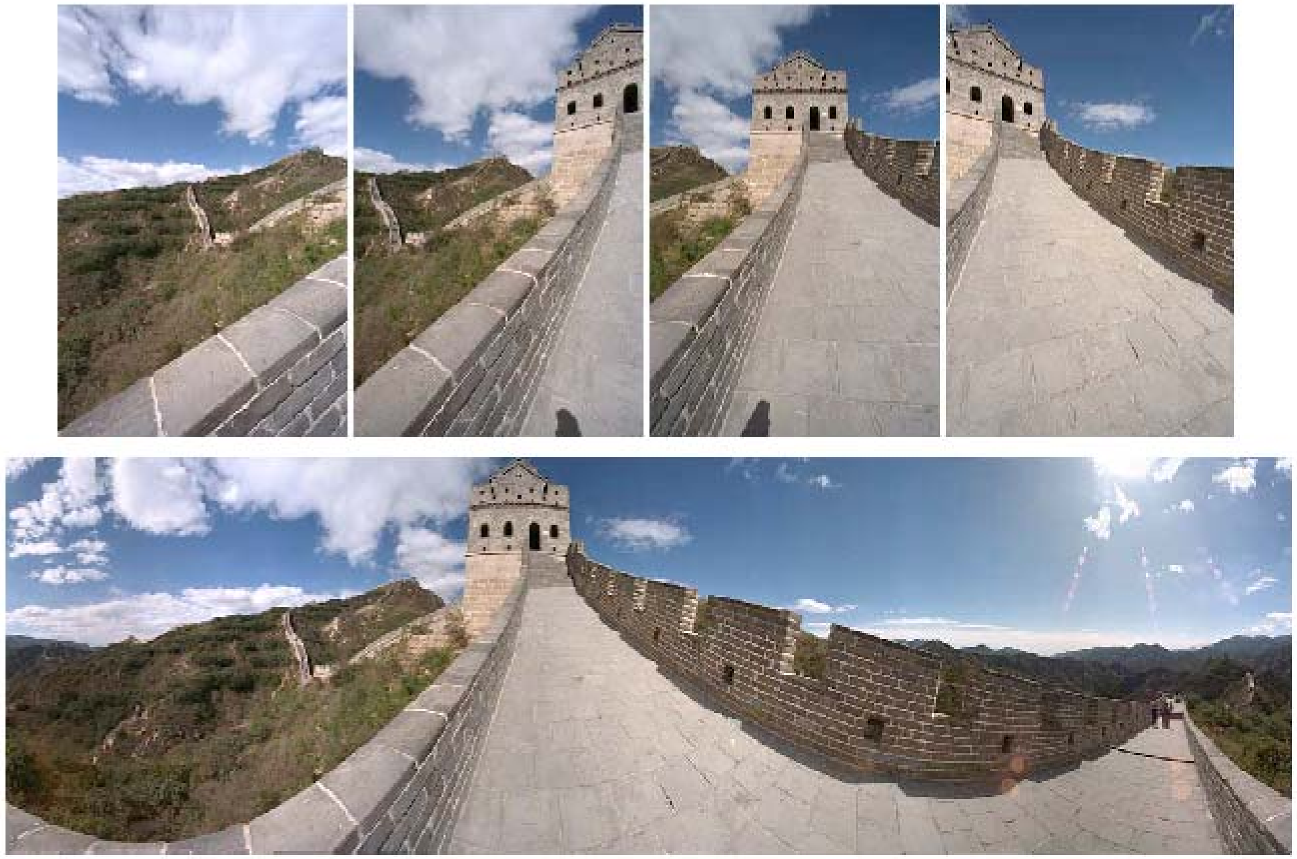
Why Mosaic?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°



Slide from Brown & Lowe

Mosaics: stitching images together

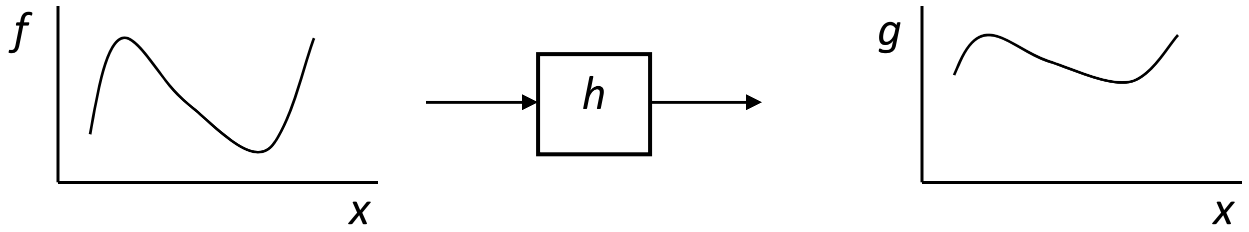


Readings

- Szeliski:
  - Chapter 3.5: Image warping
  - Chapter 5.1: Feature-based alignment
  - Chapter 8.1: Motion models

Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$

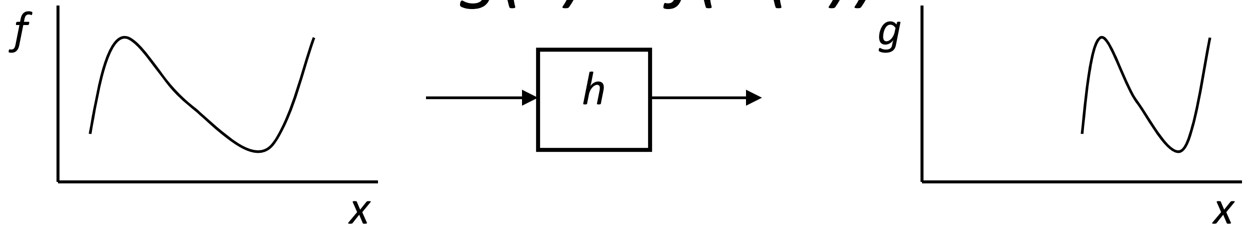
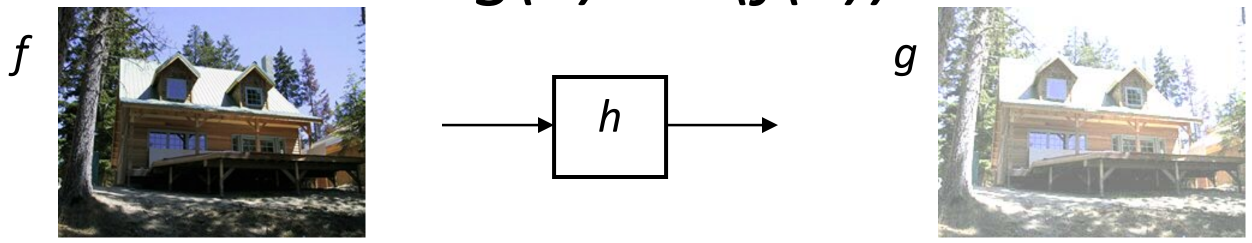
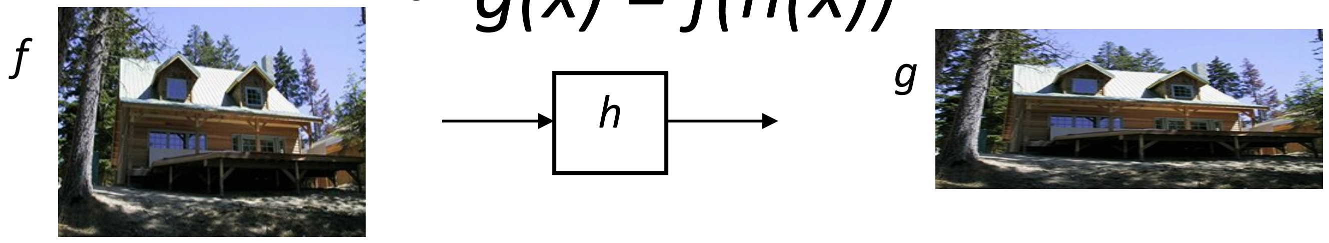


Image Warping

- image filtering: change *range* of image
  - $g(x) = h(f(x))$



- image warping: change *domain* of image
  - $g(x) = f(h(x))$



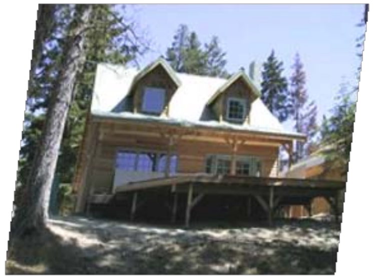
Parametric (global) warping

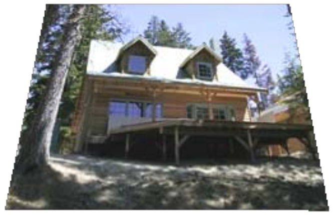
- Examples of parametric warps:

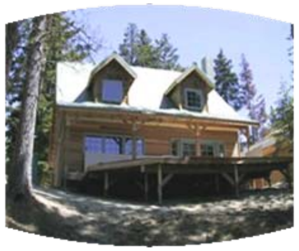
translation

rotation

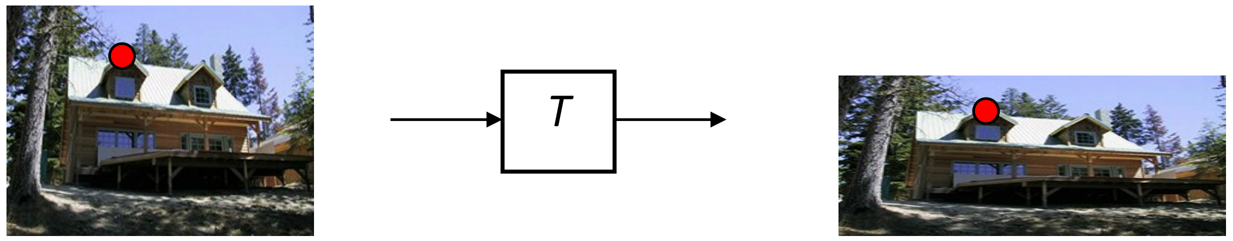
aspect


affine


perspective


cylindrical

Parametric (global) warping



p = (x,y) **p’** = (x’,y’)

- Transformation T is a coordinate-changing machine:

  p’ = *T*(p)

- What does it mean that *T* is global?
  - Is the same for any point p
  - can be described by just a few numbers (parameters)
- Let’s consider *linear* xforms (can be represented by a 2D matrix):

$$\mathbf{p}' = \mathbf{T}\mathbf{p} \qquad \left[\begin{array}{c} x' \\ y' \end{array}\right] = \mathbf{T}\left[\begin{array}{c} x \\ y \end{array}\right]$$

Common linear transformations

- Uniform scaling by *s*:

(0,0)

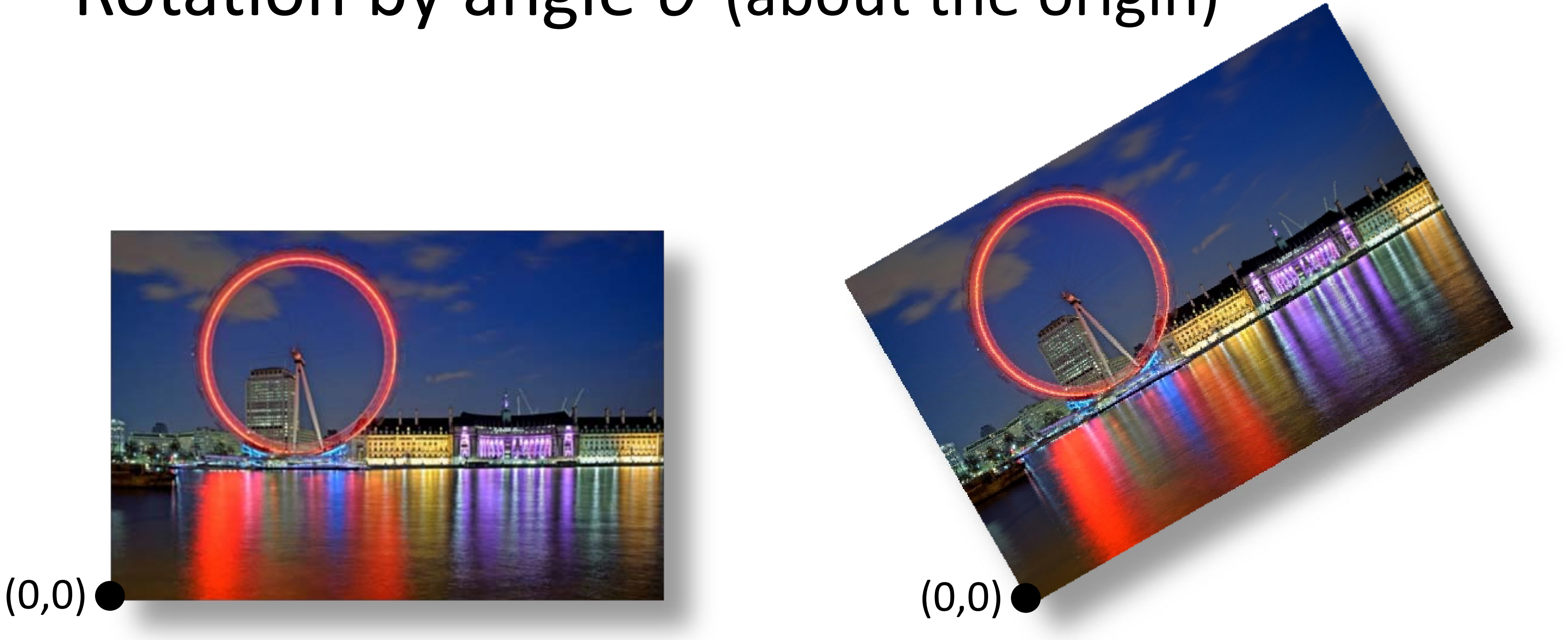
(0,0)

$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?

Common linear transformations

- Rotation by angle θ (about the origin)



$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

What is the inverse?

For rotations:

$$\mathbf{R}^{-1} = \mathbf{R}^T$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D mirror about Y axis?

$$\begin{aligned} x' &= -x \\ y' &= y \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

2D mirror over (0,0)?

$$\begin{aligned} x' &= -x \\ y' &= -y \end{aligned} \qquad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$\begin{aligned} x' &= x + t_x \\ y' &= y + t_y \end{aligned}$$

NO!

$$\boxed{\text{Translation is not a linear operation on 2D coordinates}}$$

All 2D Linear Transformations

- Linear transformations are combinations of …
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Properties of linear transformations:
  - Origin maps to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Translation

- Solution: homogeneous coordinates to the rescue

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix}$$

Affine transformations

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

any transformation with last row [0 0 1] we call an *affine* transformation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$$

Basic affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

2D *in-plane* rotation

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

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

intrinsics matrix

Affine Transformations

- Affine transformations are combinations of …
  - Linear transformations, and
  - Translations

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Properties of affine transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines remain parallel
  - Ratios are preserved
  - Closed under composition

Projective Transformations

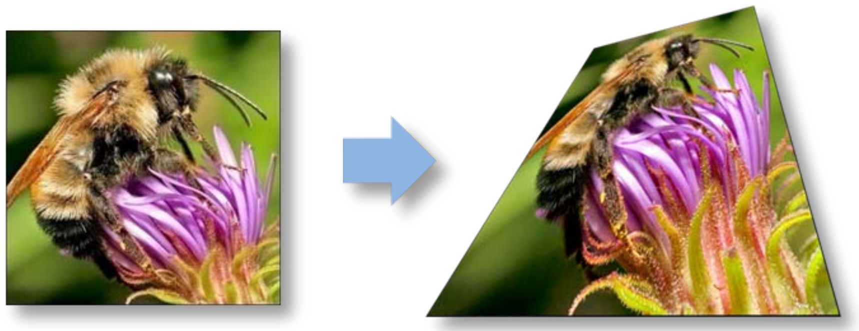
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- Projective transformations …
  - Affine transformations, and
  - Projective warps
- Properties of projective transformations:
  - Origin does not necessarily map to origin
  - Lines map to lines
  - Parallel lines do not necessarily remain parallel
  - Ratios are not preserved
  - Closed under composition

Projective Transformations

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)



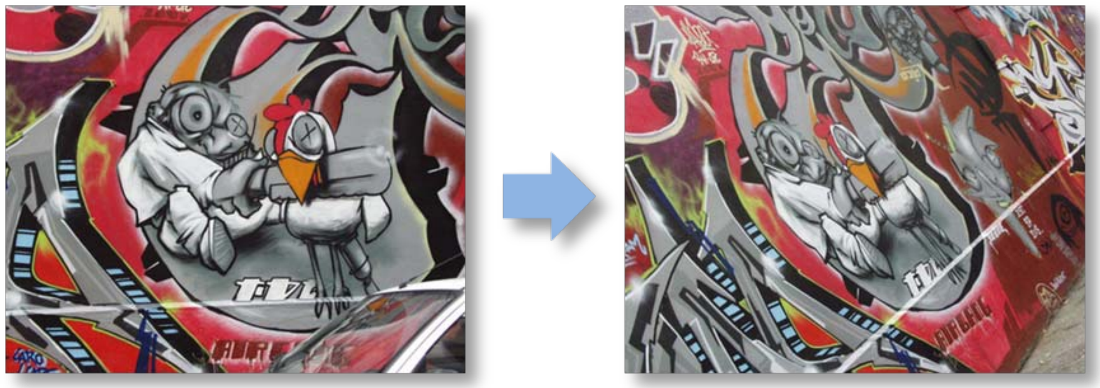
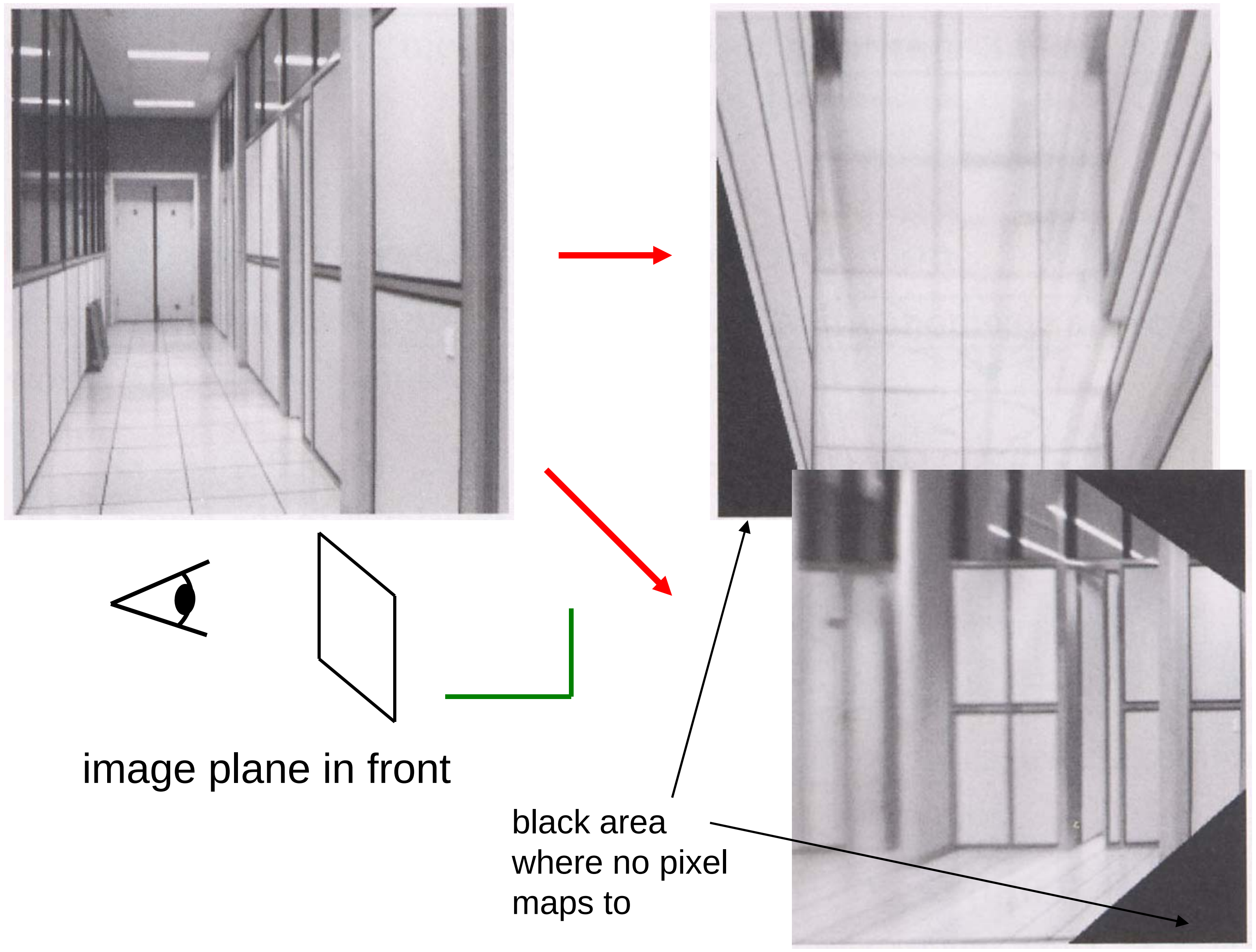
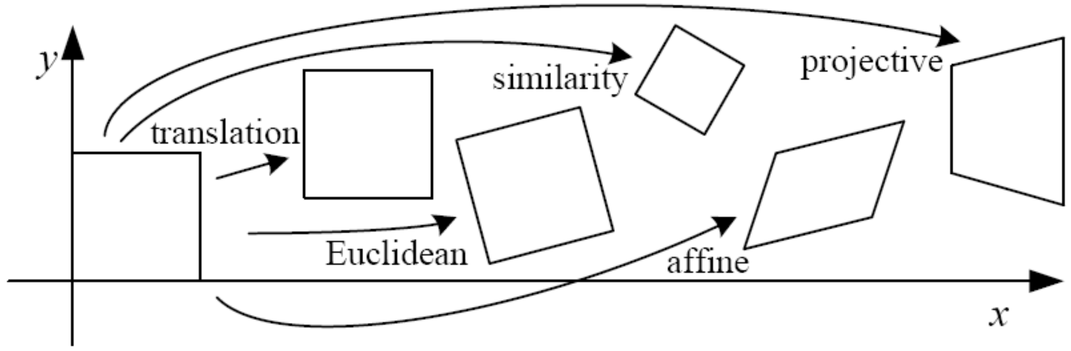


Image warping with homographies



2D image transformations



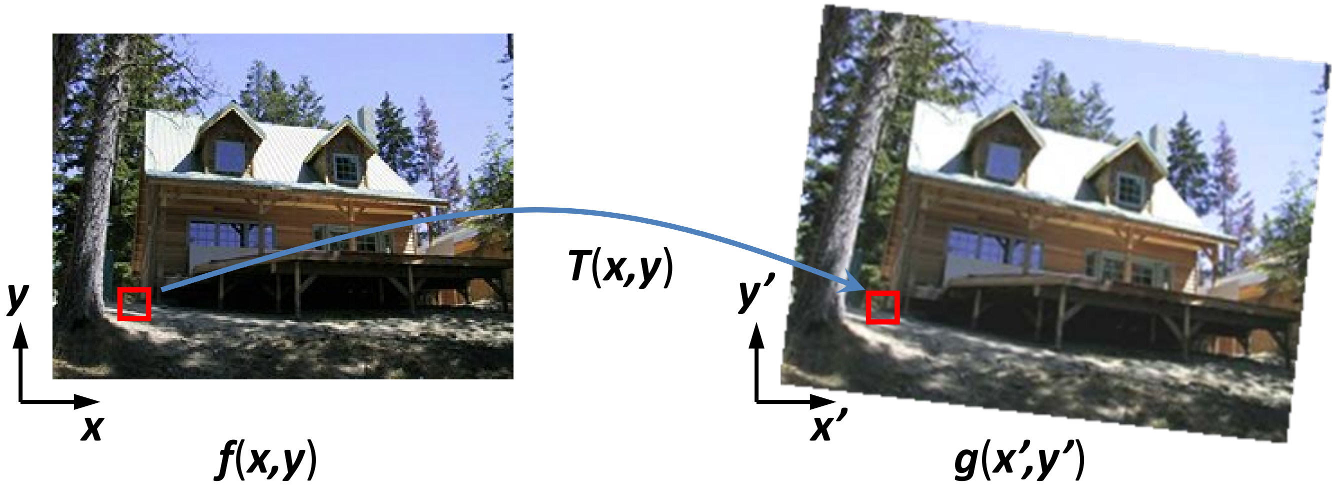
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	□
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	◇
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	◇
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	▱
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	⏢

These transformations are a nested set of groups

- Closed under composition and inverse is a member

Image Warping

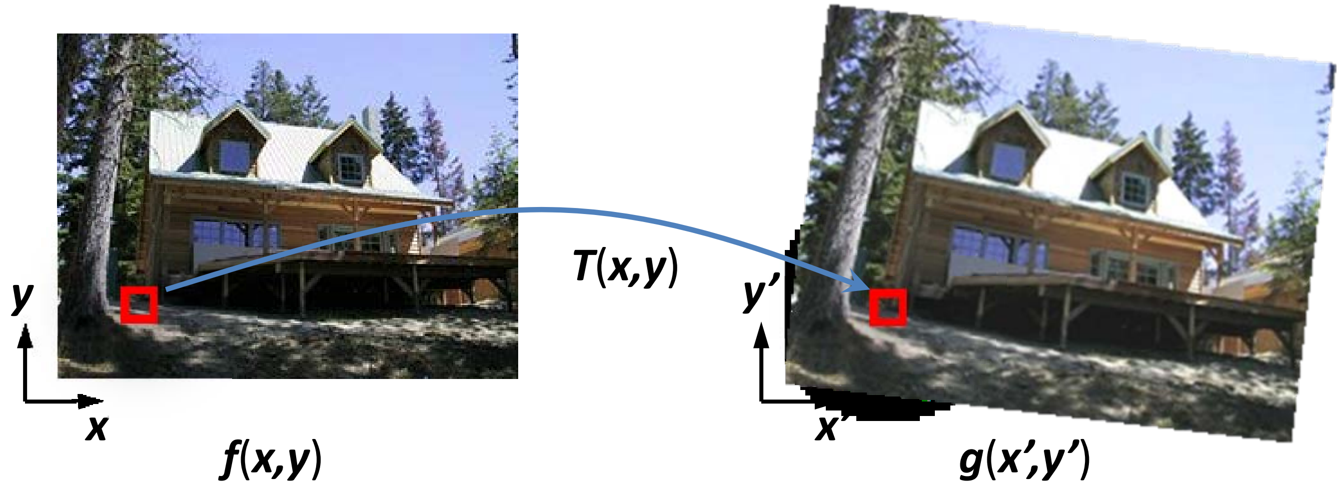
- Given a coordinate xform $(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{T}(\boldsymbol{x},\boldsymbol{y})$ and a source image $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$, how do we compute an xformed image $\boldsymbol{g}(\boldsymbol{x'},\boldsymbol{y'}) = \boldsymbol{f}(\boldsymbol{T}(\boldsymbol{x},\boldsymbol{y}))$?



f(*x*,*y*)   *g*(*x'*,*y'*)

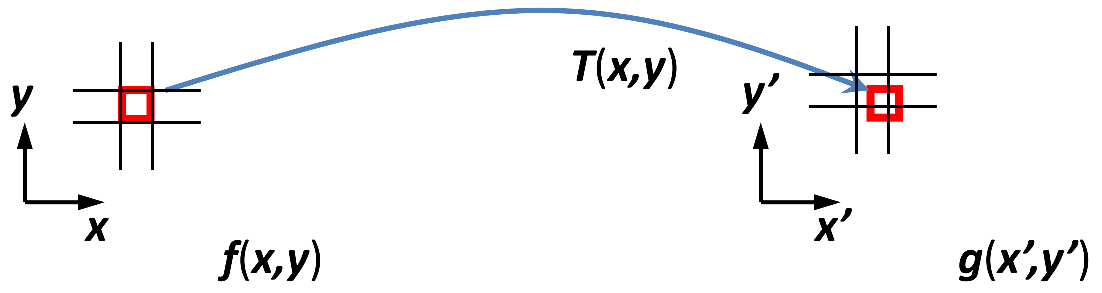
Forward Warping

- Send each pixel ***f***(***x***) to its corresponding location (***x'***,***y'***) = ***T***(***x***,***y***) in ***g***(***x'***,***y'***)
  - What if pixel lands "between" two pixels?



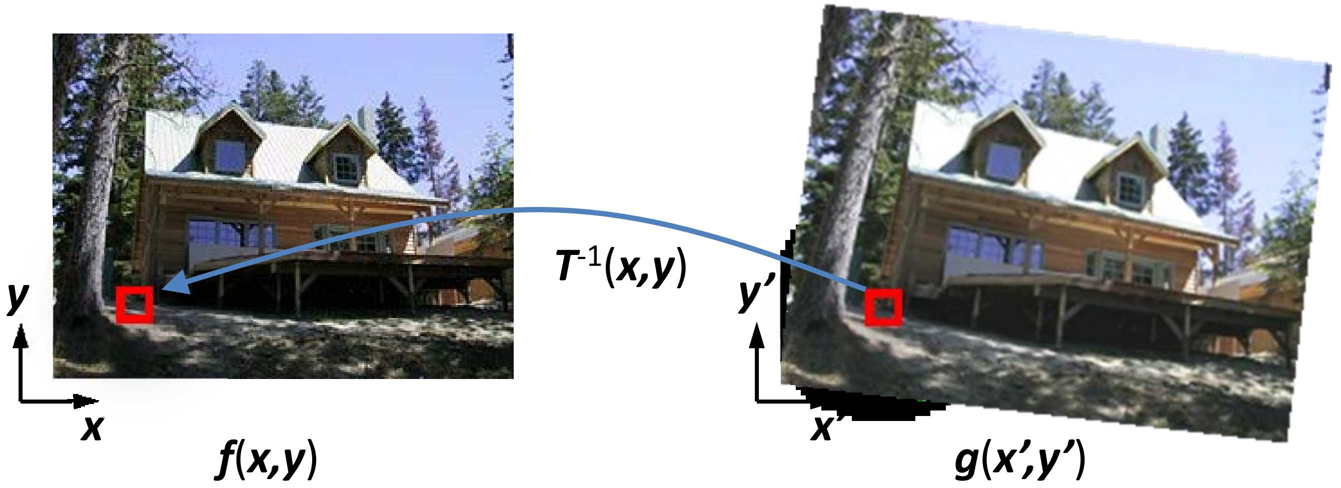
Forward Warping

- Send each pixel ***f***(***x,y***) to its corresponding location ***x'*** = ***h***(***x,y***) in ***g***(***x',y'***)
  - What if pixel lands "between" two pixels?
  - Answer: add "contribution" to several pixels, normalize later (*splatting*)
  - Can still result in holes



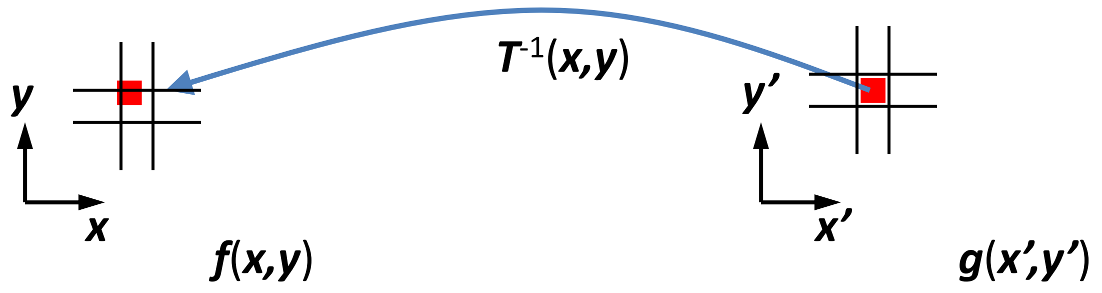
Inverse Warping

- Get each pixel ***g***(***x'***,***y'***) from its corresponding location (***x***,***y***) = ***T***$^{-1}$(***x***,***y***) in ***f***(***x***,***y***)
  - Requires taking the inverse of the transform
  - What if pixel comes from "between" two pixels?



Inverse Warping

- Get each pixel $\boldsymbol{g}(\boldsymbol{x'})$ from its corresponding location $\boldsymbol{x'} = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{f}(\boldsymbol{x})$
  - What if pixel comes from "between" two pixels?
  - Answer: *resample* color value from *interpolated* (*prefiltered*) source image



Interpolation

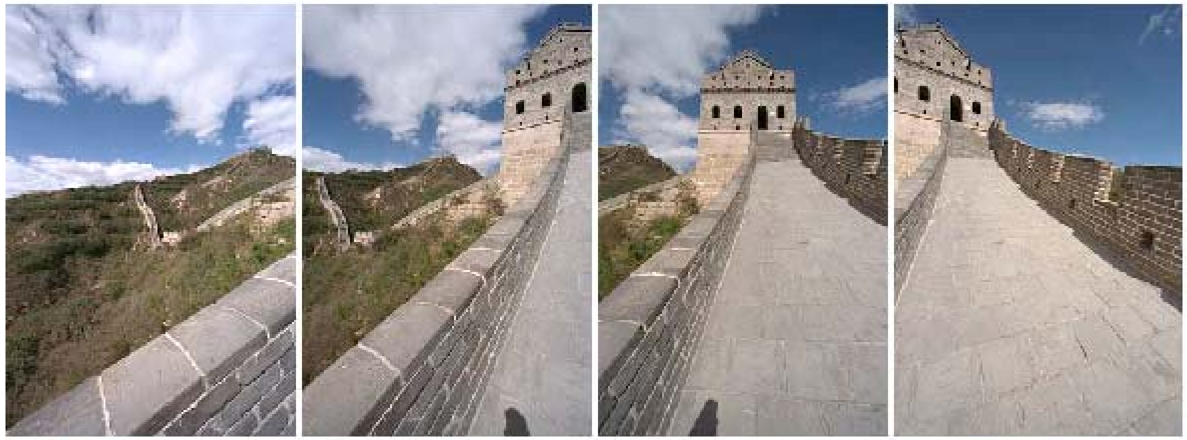
- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
  - sinc
- Needed to prevent “jaggies” and “texture crawl”

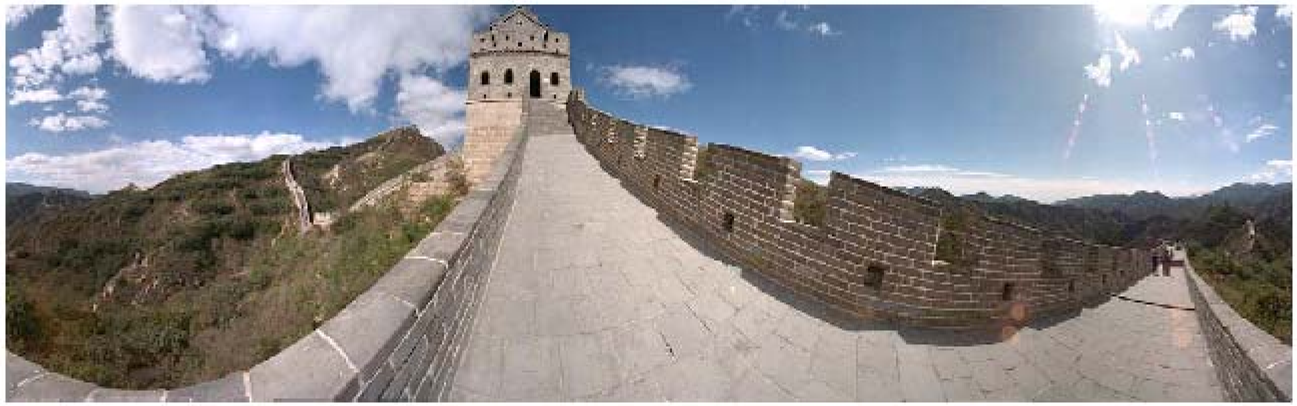
  (with prefiltering)

Questions?

- 3-minute break

Back to mosaics



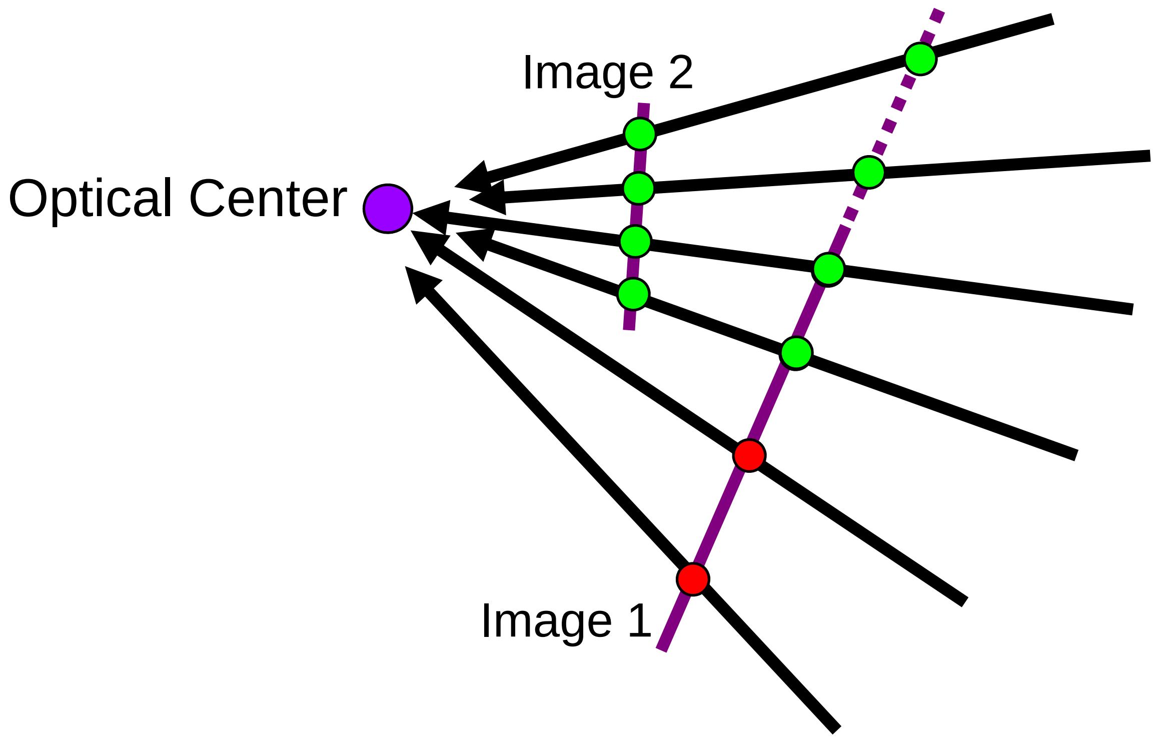


- How to we align the images?

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



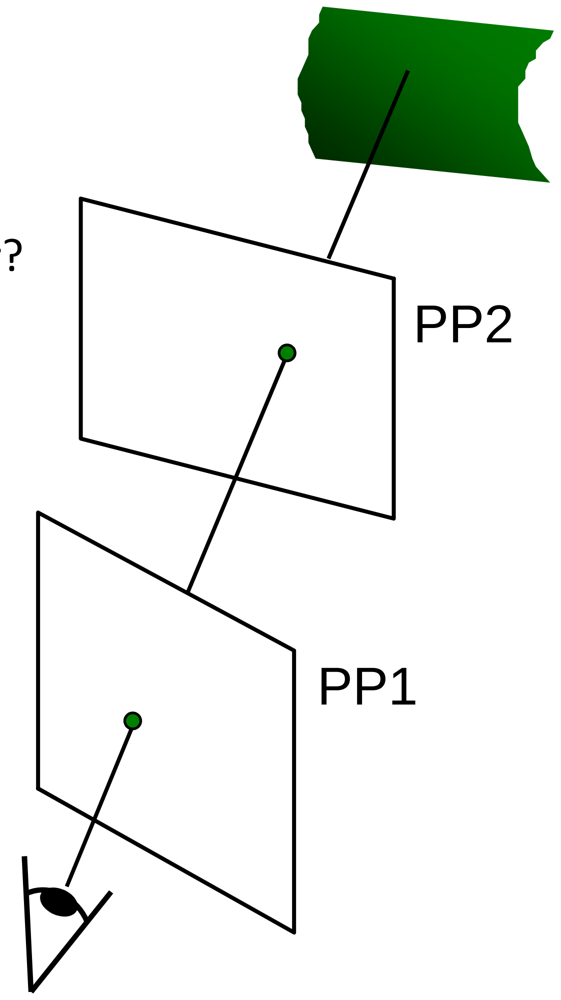
- If we capture all 360º of rays, we can create a 360º panorama
- The basic operation is *projecting* an image from one plane to another
- The projective transformation is scene-INDEPENDENT
  - This depends on all the images having the same optical center

Image reprojection

- Basic question
  - How to relate two images from the same camera center?
    - how to map a pixel from PP1 to PP2

Answer

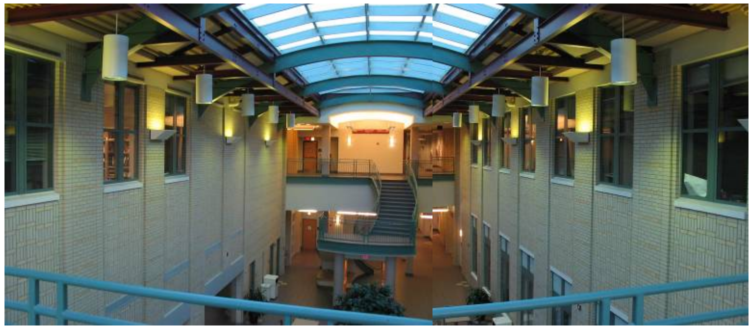
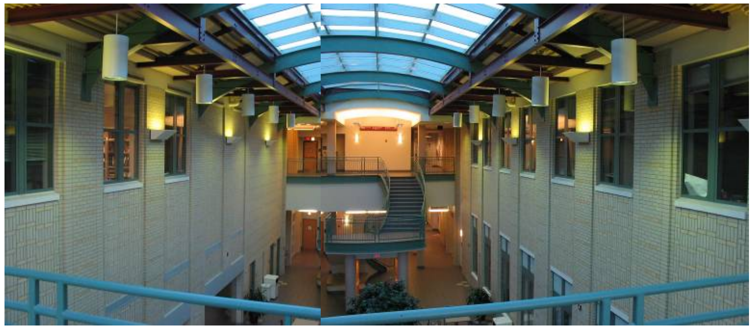
- Cast a ray through each pixel in PP1
- Draw the pixel where that ray intersects PP2



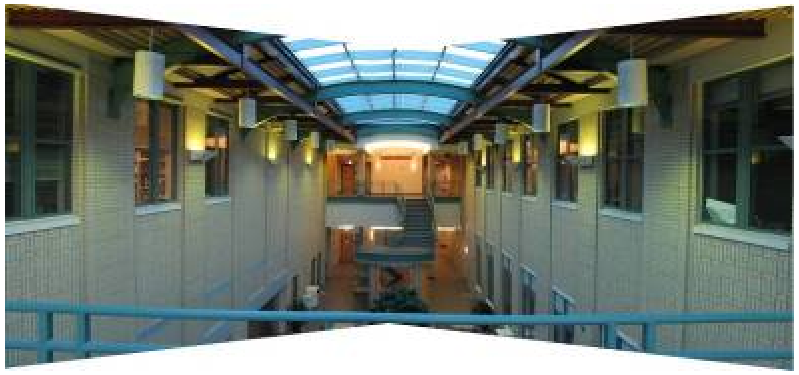
What is the transformation?

left on top

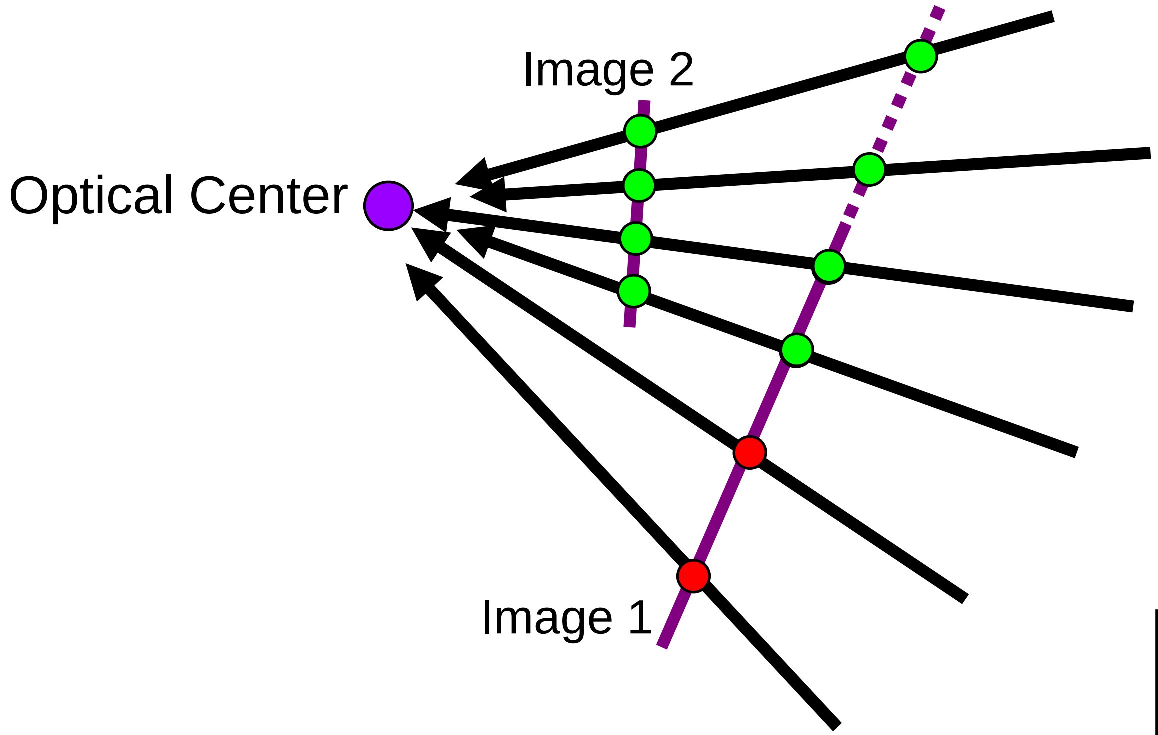
right on top

Translations are not enough to align the images



What is the transformation?



$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \mathbf{K}_2^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

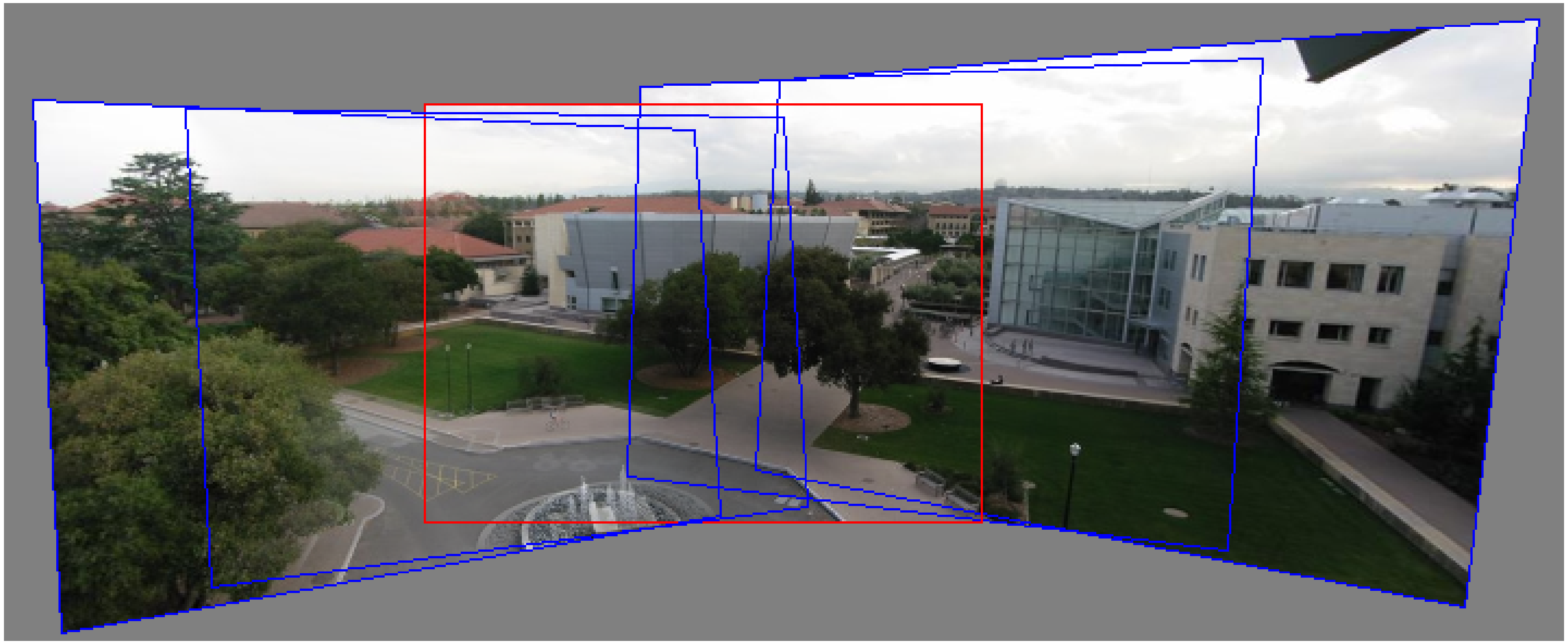
$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \mathbf{R}_2^T \mathbf{K}_2^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

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

How do we map image 2 onto image 1's projection plane?

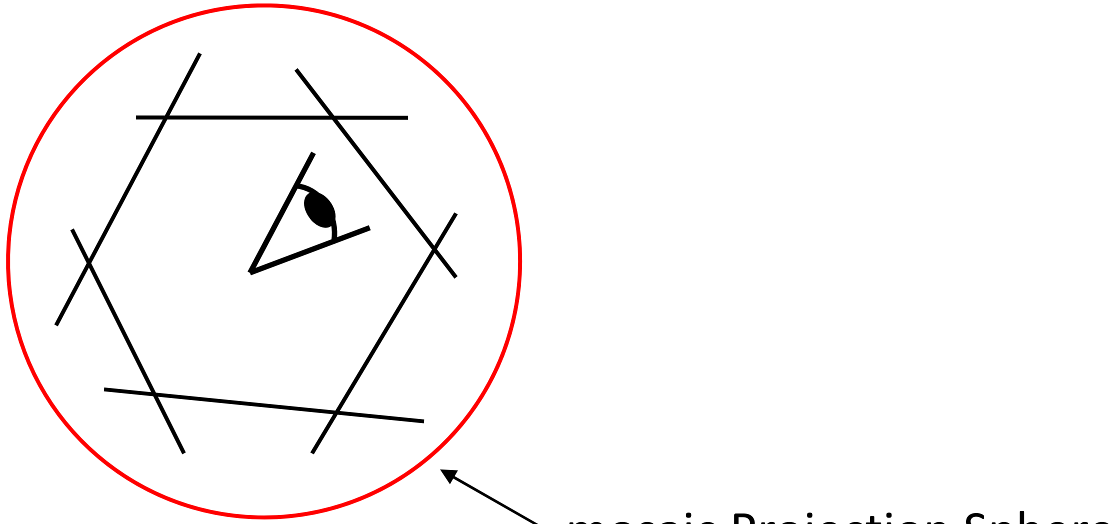
image 1	image 2
$\mathbf{K}_1$	$\mathbf{K}_2$
$\mathbf{R}_1 = \mathbf{I}_{3\times 3}$	$\mathbf{R}_2$

Can we use homography to create a 360 panorama?

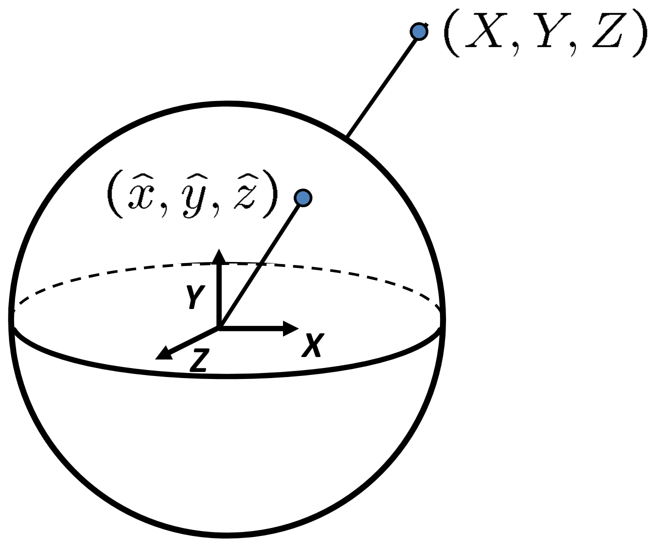


Panoramas

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



Spherical projection



unit sphere

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

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

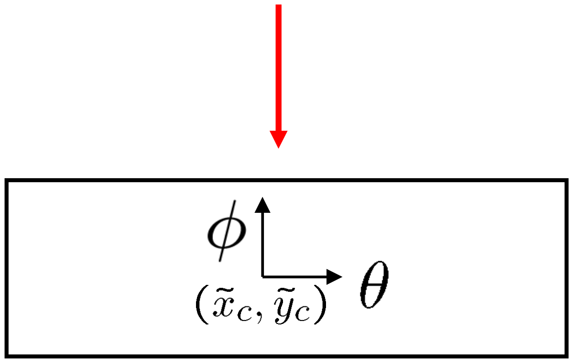
- Convert to spherical coordinates

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

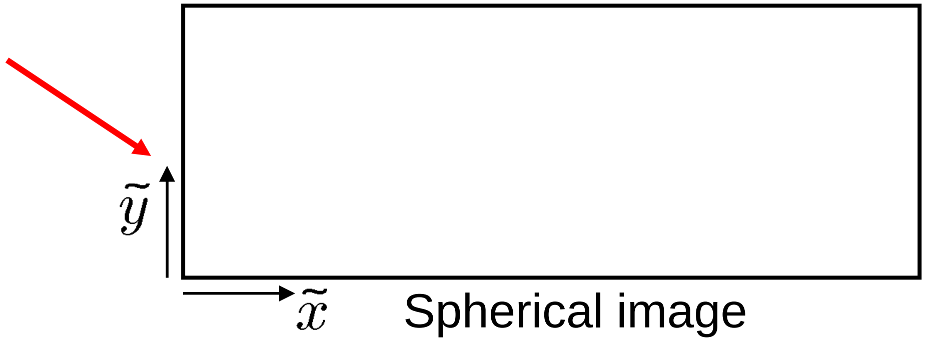
- Convert to spherical image coordinates

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

  - s defines size of the final image
    - » often convenient to set s = camera focal length

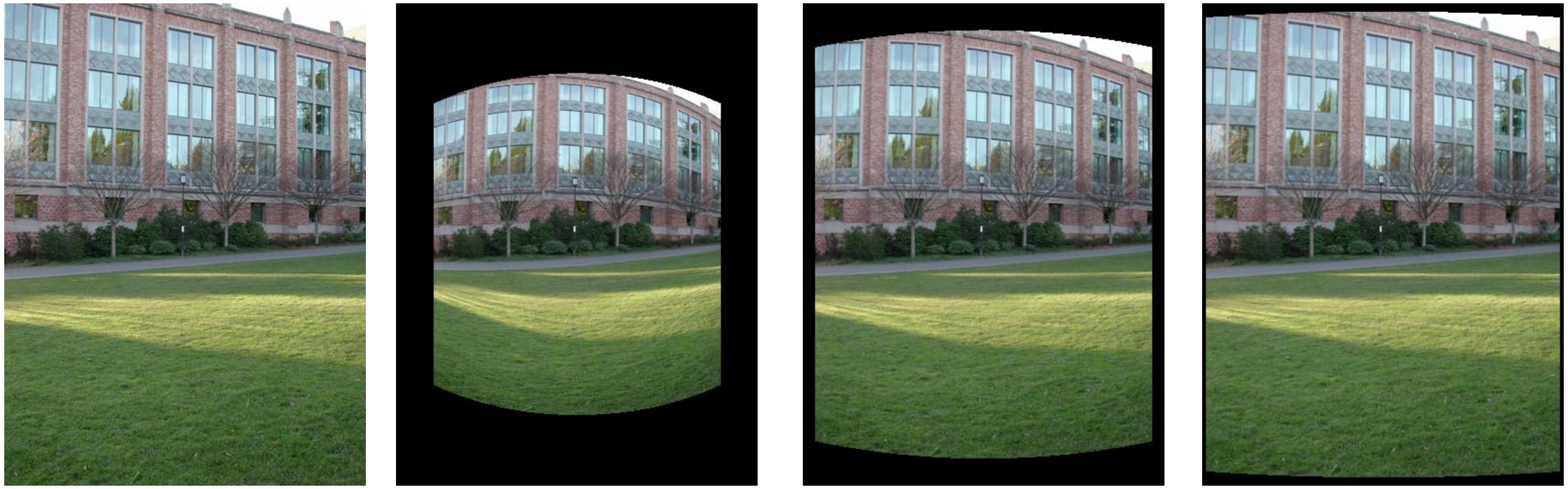


unwrapped sphere



Spherical image

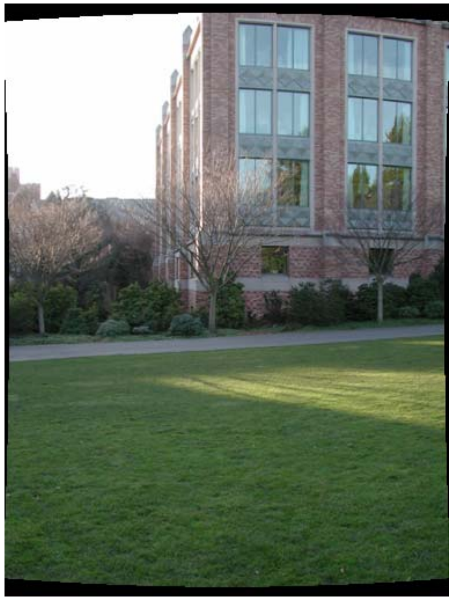
Spherical reprojection



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

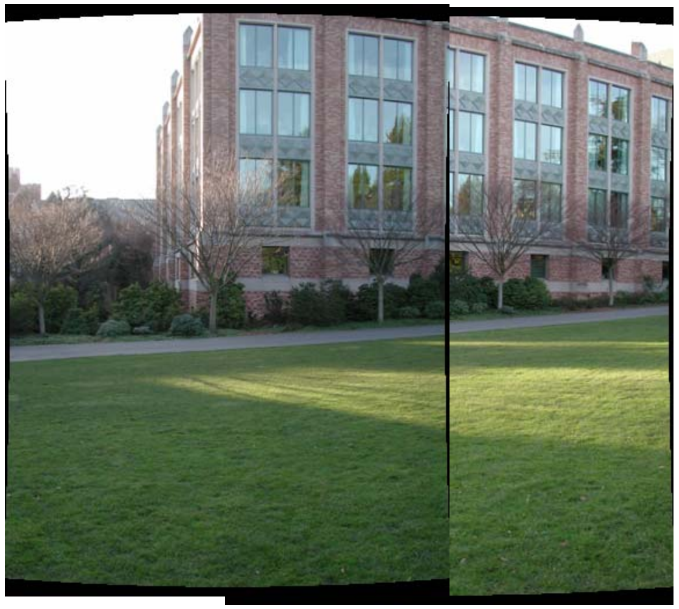
- Map image to spherical coordinates
  - need to know the focal length

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

Questions?