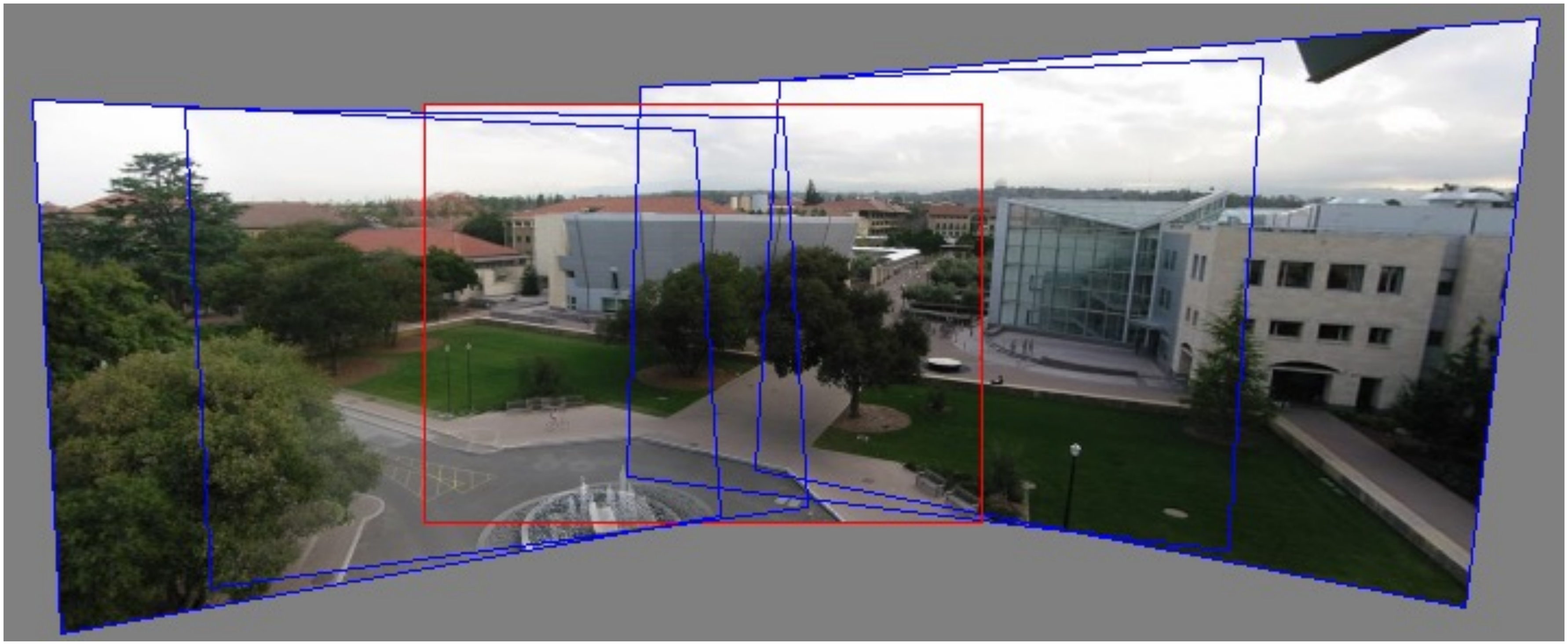
More Mosaic Madness

© Jeffrey Martin (jeffrey-martin.com)

with a lot of slides stolen from Steve Seitz and Rick Szeliski

CS180: Intro to Computer Vision and Comp. Photo
Angjoo Kanazawa and Alexei Efros, UC Berkeley, Fall 2023

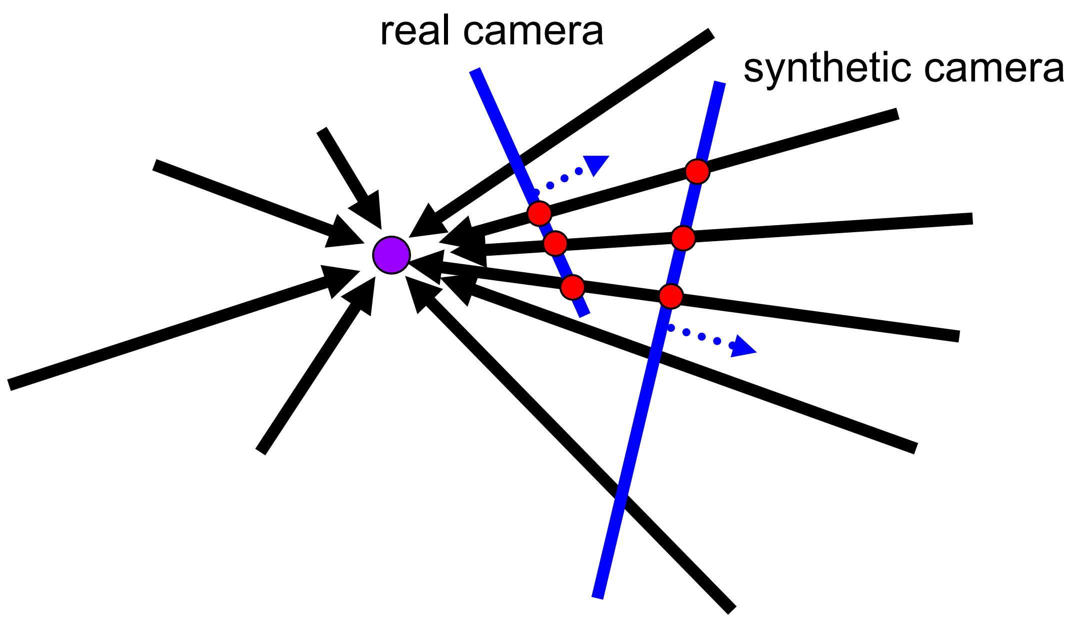
Project 4: Panorama



1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend

Think about this: When is this not true?

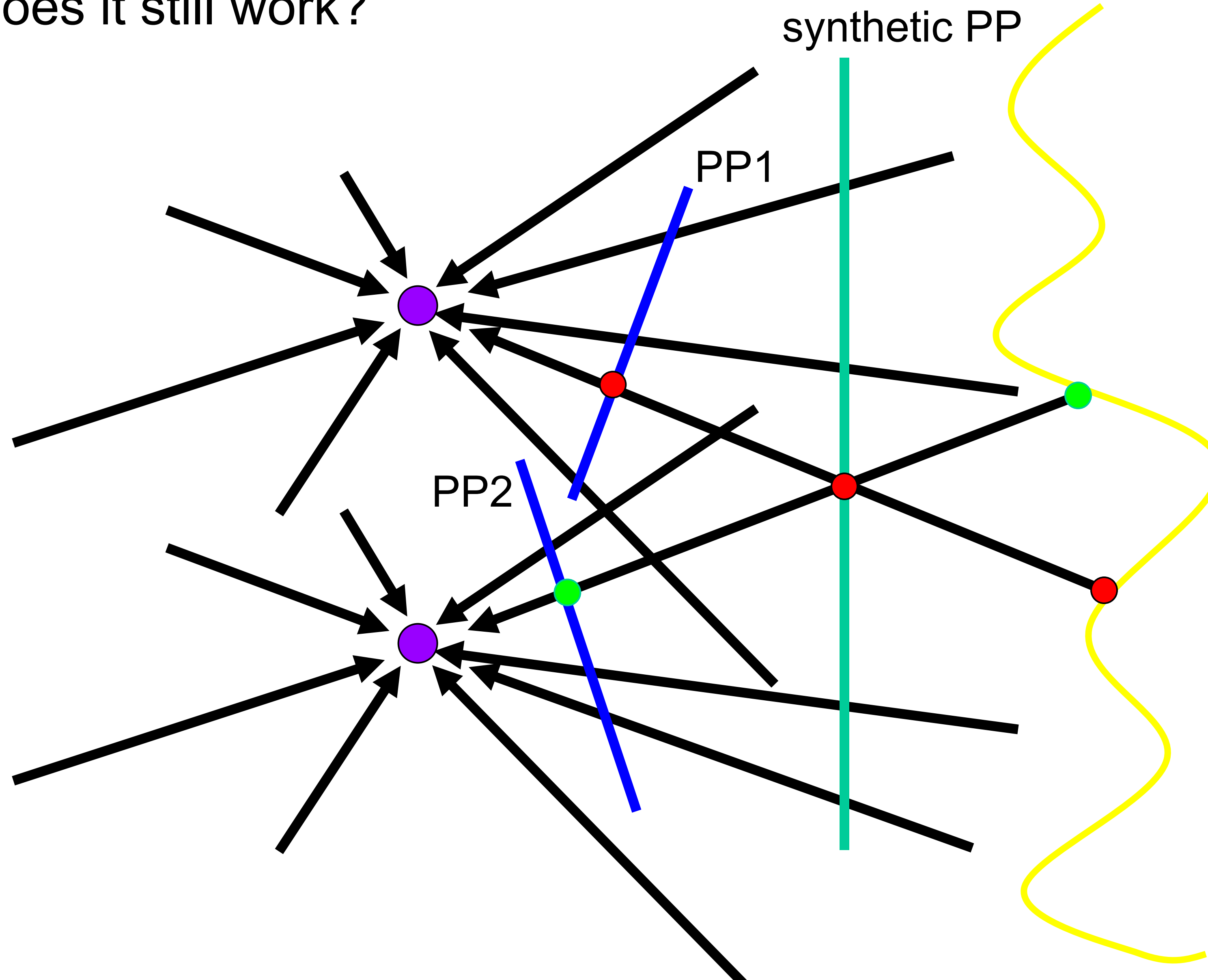
We can generate any synthetic camera view
as long as it has **the same center of projection**!



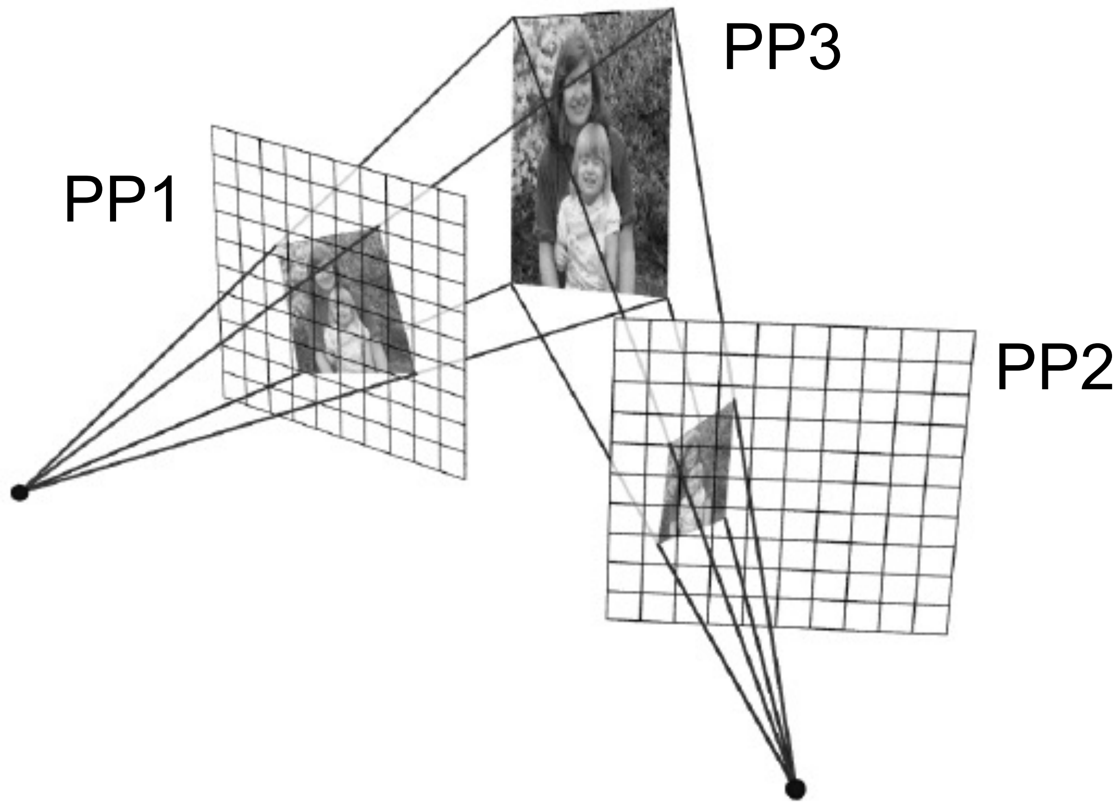
What happens if there are two center of projection?
(you move your head)

changing camera center

Does it still work?



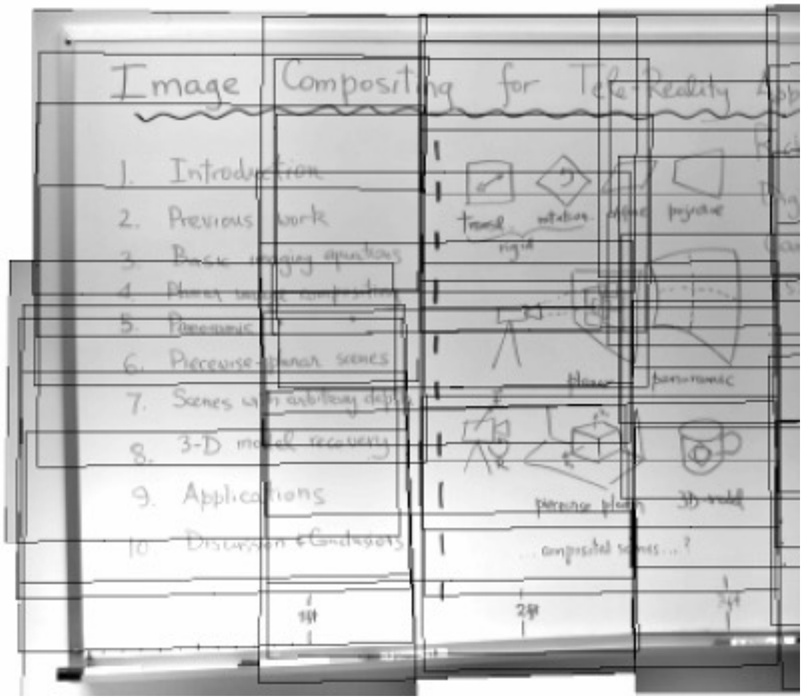
Planar scene (or far away)

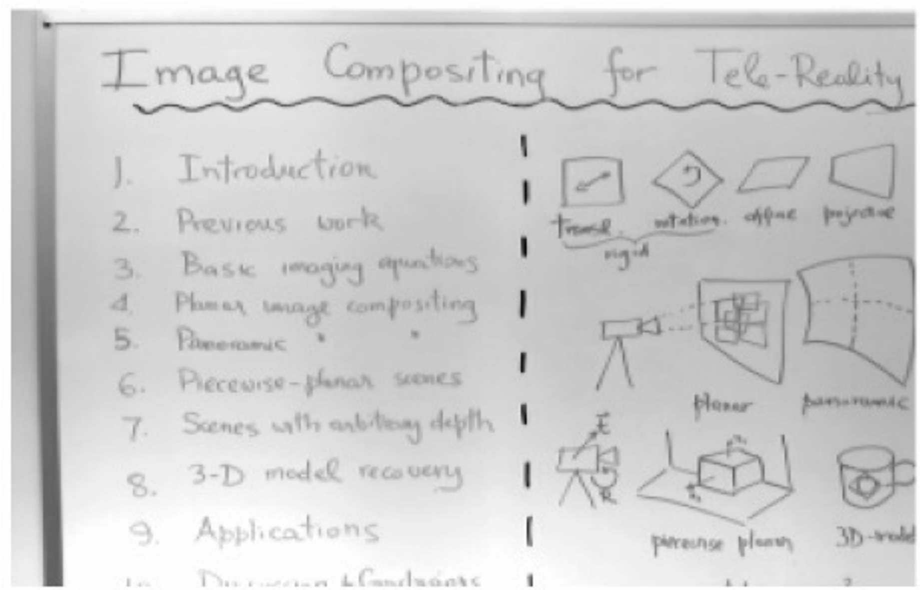


PP3 is a projection plane of both centers of projection, so we are OK!

This is how big aerial photographs are made

Planar mosaic





Julian Beever: Manual Homographies

http://users.skynet.be/J.Beever/pave.htm

Bells and Whistles

Blending and Compositing

- use homographies to combine images or video and images together in an interesting (fun) way. E.g.
  - put fake graffiti on buildings or chalk drawings on the ground
  - replace a road sign with your own poster
  - project a movie onto a building wall
  - etc.

Bells and Whistles

Virtual Camera rorate

- Similar to face morphing, produce a video of virtual camera rotation from a single image
- Can also do it for translation, if looking at a planar object

Other interesting ideas?

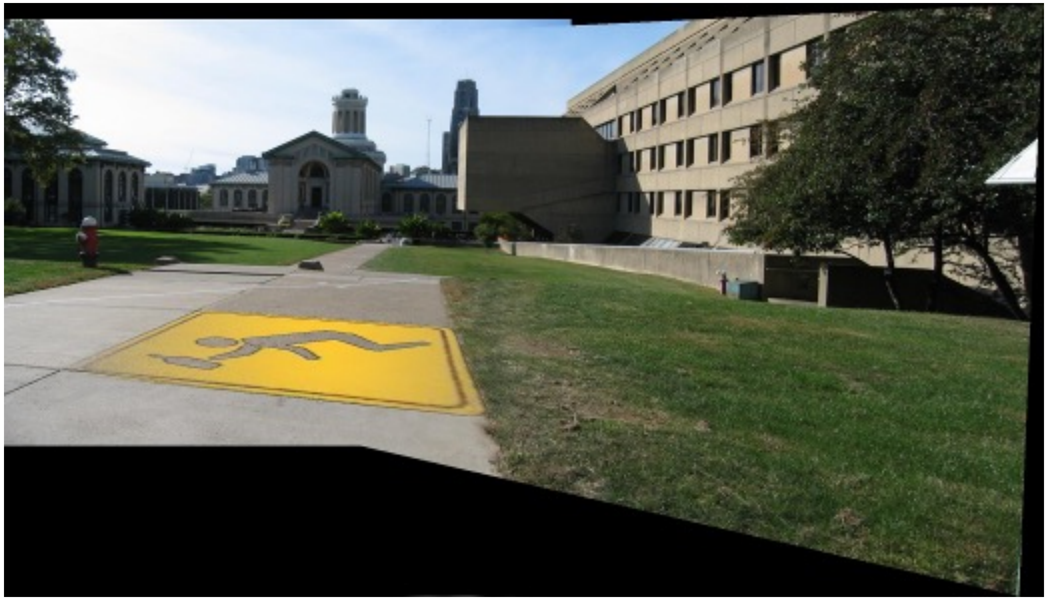
- talk to me

From previous year's classes

Ben Hollis, 2004

Ben Hollis, 2004

Matt Pucevich , 2004



Eunjeong Ryu (E.J), 2004

Bells and Whistles

Capture creative/cool/bizzare panoramas

- Example from UW (by Brett Allen):

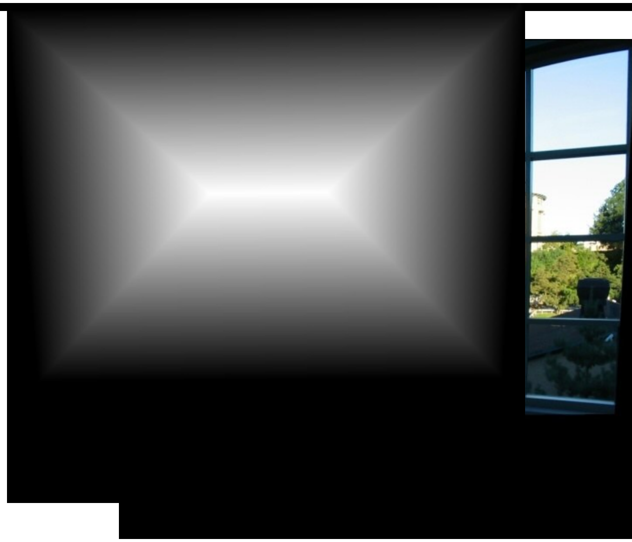
- Ever wondered what is happening inside your fridge while you are not looking?

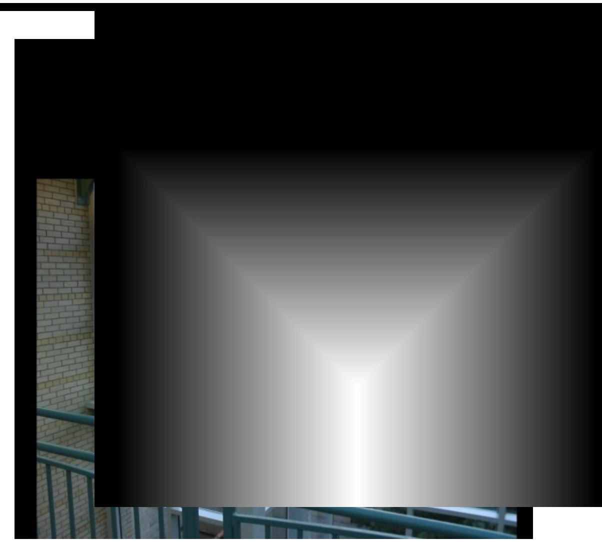
Capture a 360 panorama (quite tricky…)

Setting alpha: simple averaging

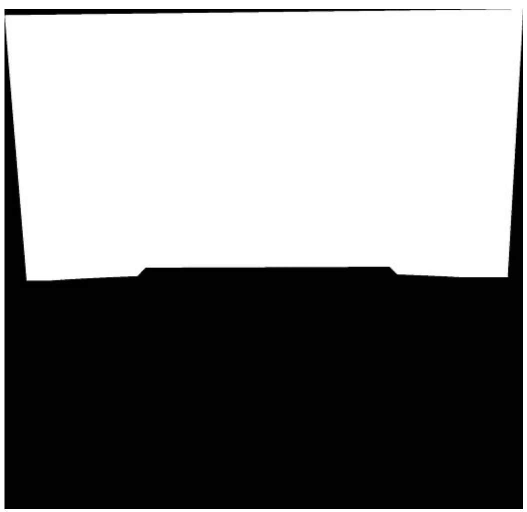
Alpha = .5 in overlap region

Setting alpha: center seam



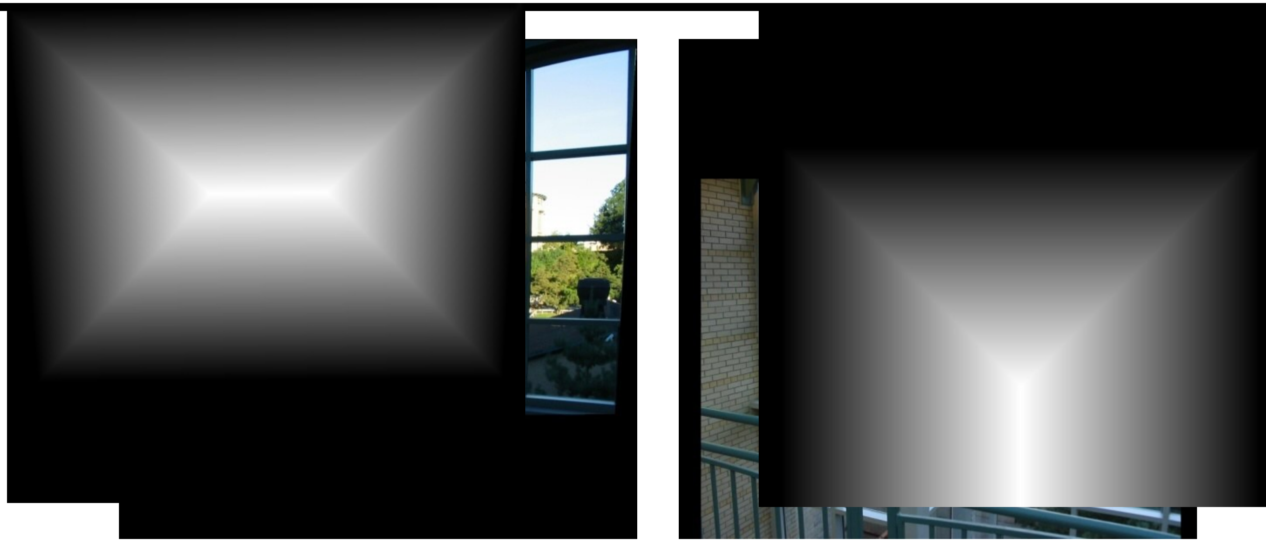


Distance Transform
`bwdist`

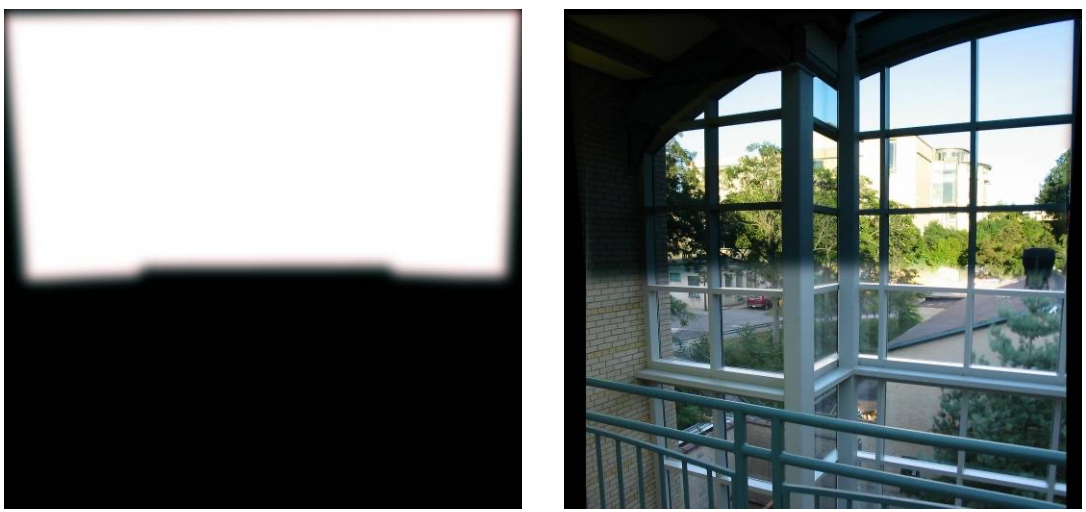


Alpha = logical(dtrans1>dtrans2)

Setting alpha: blurred seam



Distance transform



Alpha = blurred

Simplification: Two-band Blending

Brown & Lowe, 2003

- Only use two bands: high freq. and low freq.
- Blends low freq. smoothly
- Blend high freq. with no smoothing: use binary alpha

2-band “Laplacian Stack” Blending

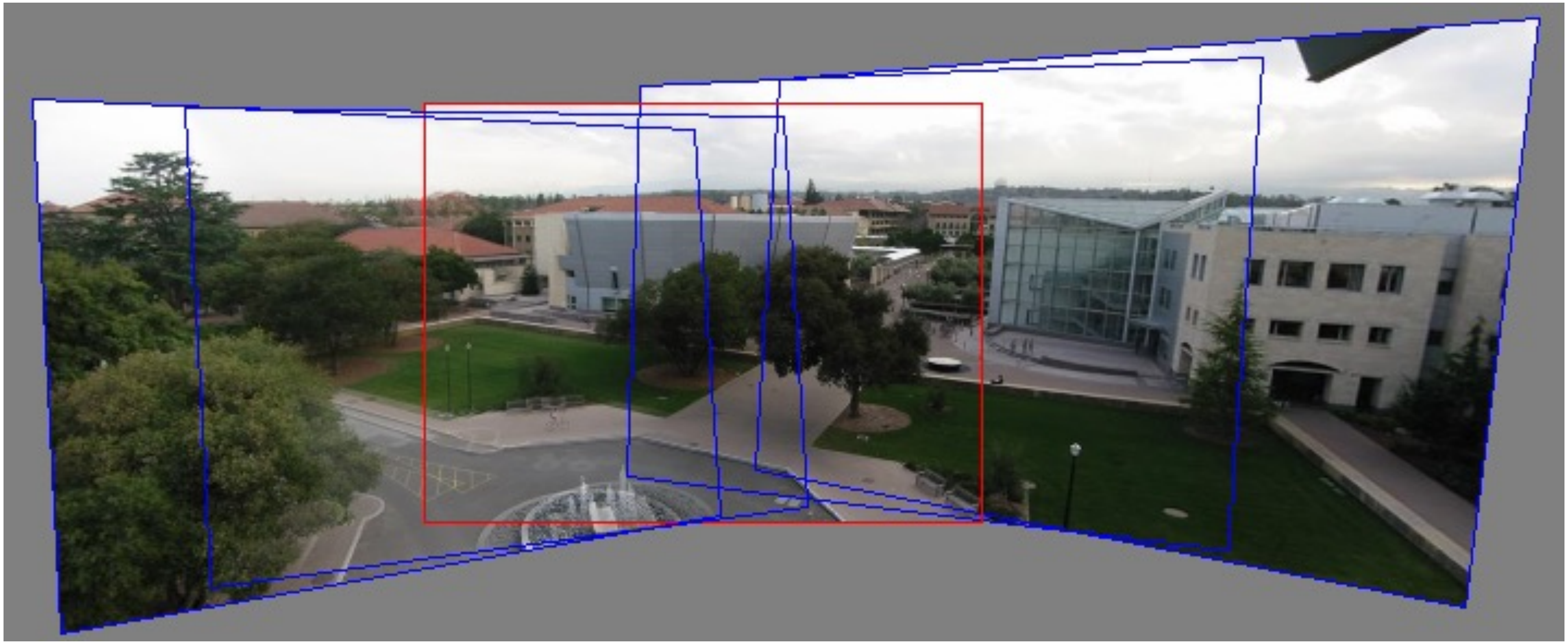
Low frequency (λ > 2 pixels)

High frequency (λ < 2 pixels)

Linear Blending

2-band Blending

Rotational Mosaics

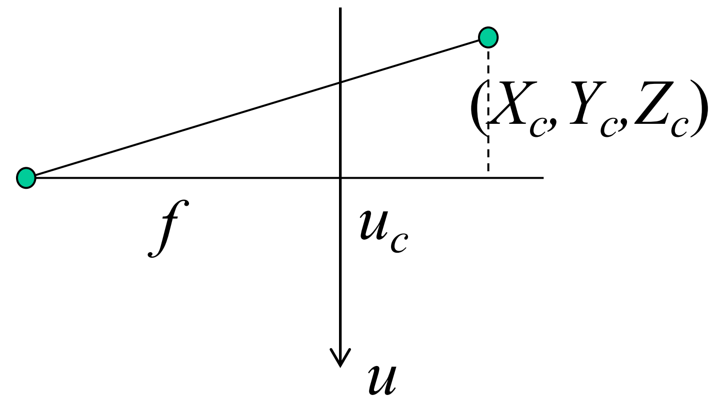


Can we say something more about rotational mosaics?

i.e. can we further constrain our H?

3D → 2D Perspective Projection

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



$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

K

3D Rotation Model

Projection equations

1. Project from image to 3D ray

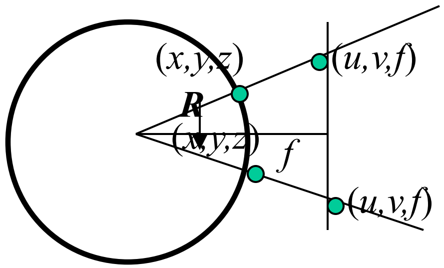
   $(x_0,y_0,z_0) \quad = (u_0\text{-}u_c, v_0\text{-}v_c, f)$

2. Rotate the ray by camera motion

   $(x_1,y_1,z_1) \quad = \boldsymbol{R}_{01}\ (x_0,y_0,z_0)$

3. Project back into new (source) image

   $(u_1,v_1) \qquad = (fx_1/z_1+u_c, fy_1/z_1+v_c)$



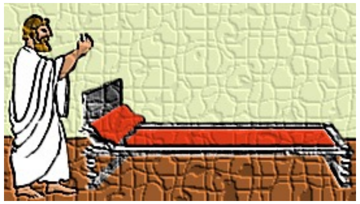
Therefore:

$$\mathbf{H} = \mathbf{K}_0\mathbf{R}_{01}\mathbf{K}_1^{-1}$$

Our homography has only 3,4 or 5 DOF, depending if focal length is known, same, or different.

- This makes image registration much better behaved

Pairwise alignment



Procrustes Algorithm [Golub & VanLoan]

Given two sets of matching points, compute R

$p_i' = \boldsymbol{R}\, p_i$ with 3D rays

$p_i = N(x_i, y_i, z_i) = N(u_i - u_c, v_i - v_c, f)$

$$\min_R ||Rp_i - p_i'||$$

Can be solved in closed form with SVD:

$\boldsymbol{A} = \Sigma_\mathbf{i}\, p_i\, p_i'^T = \Sigma_\mathbf{i}\, p_i\, p_i^T\, \boldsymbol{R}^T = \boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{V}^T = (\boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{U}^T)\,\boldsymbol{R}^T$

$\boldsymbol{V}^T = \boldsymbol{U}^T\boldsymbol{R}^T$

$\boldsymbol{R} = \boldsymbol{V}\,\boldsymbol{U}^T$

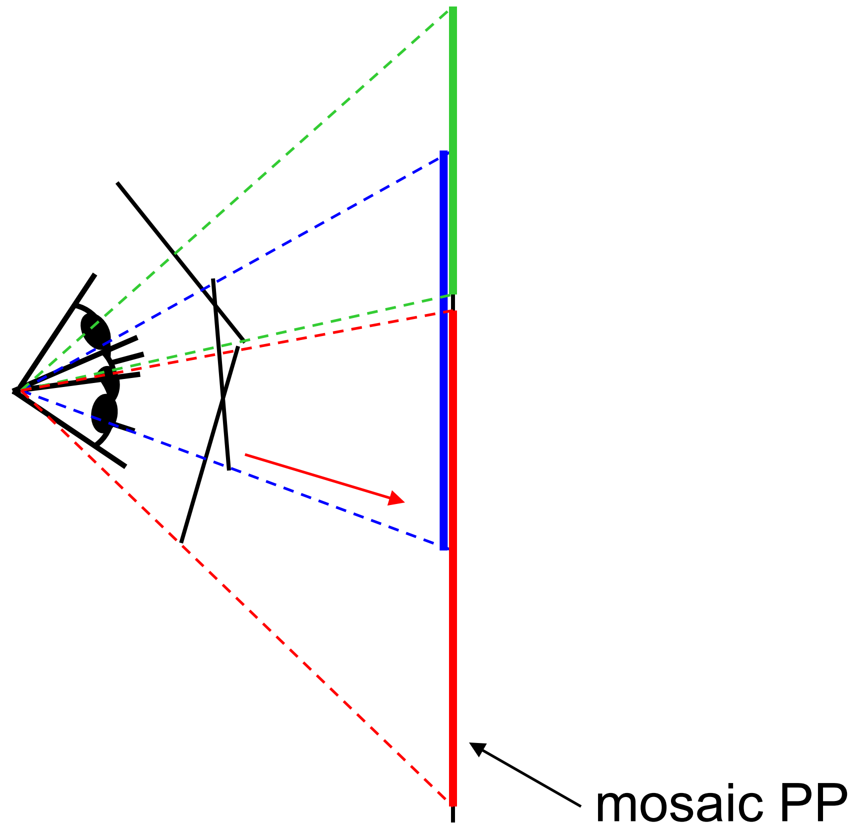
https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

Rotation about vertical axis

What if our camera rotates on a tripod?

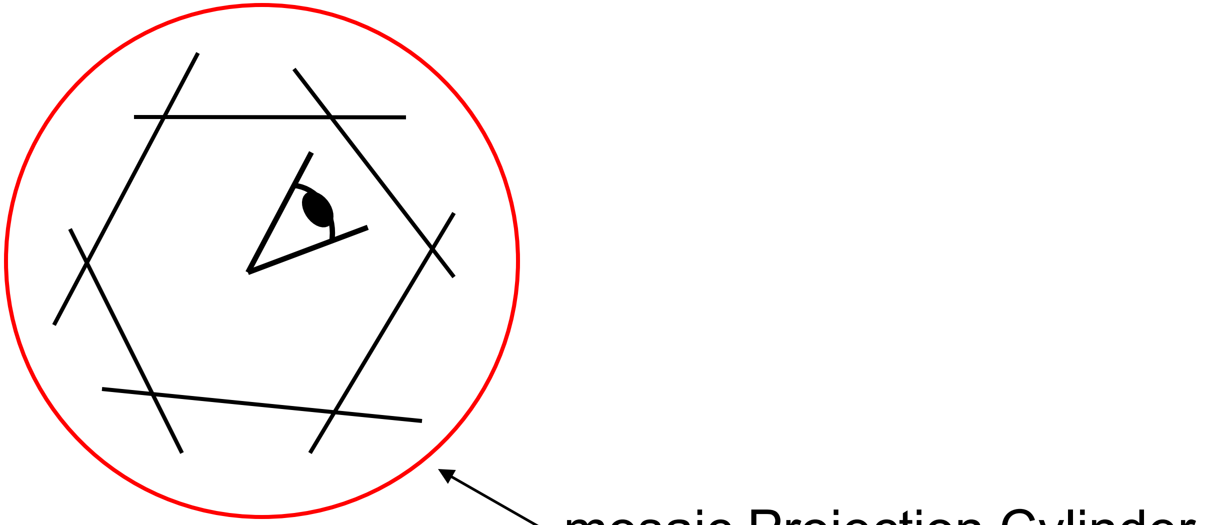
What's the structure of H?

Do we have to project onto a plane?

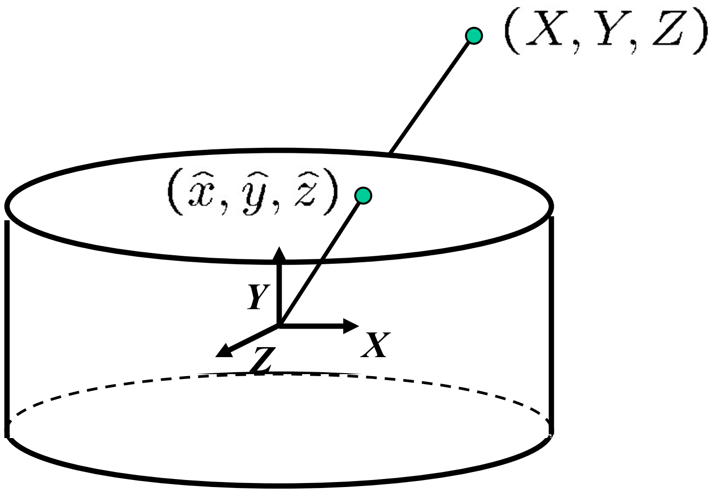


Full Panoramas

What if you want a 360° field of view?



Cylindrical projection



unit cylinder

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

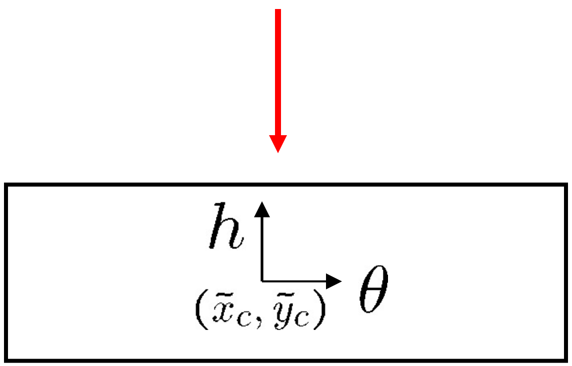
$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$$

- Convert to cylindrical coordinates

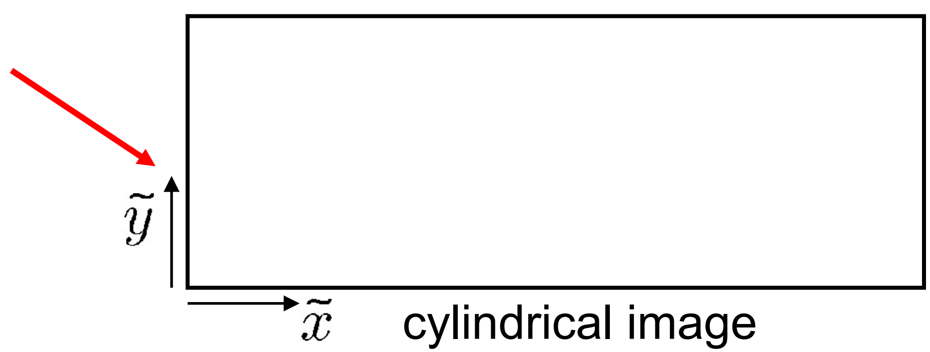
$$(sin\theta, h, cos\theta) = (\hat{x}, \hat{y}, \hat{z})$$

- Convert to cylindrical image coordinates

$$(\tilde{x}, \tilde{y}) = (f\theta, fh) + (\tilde{x}_c, \tilde{y}_c)$$

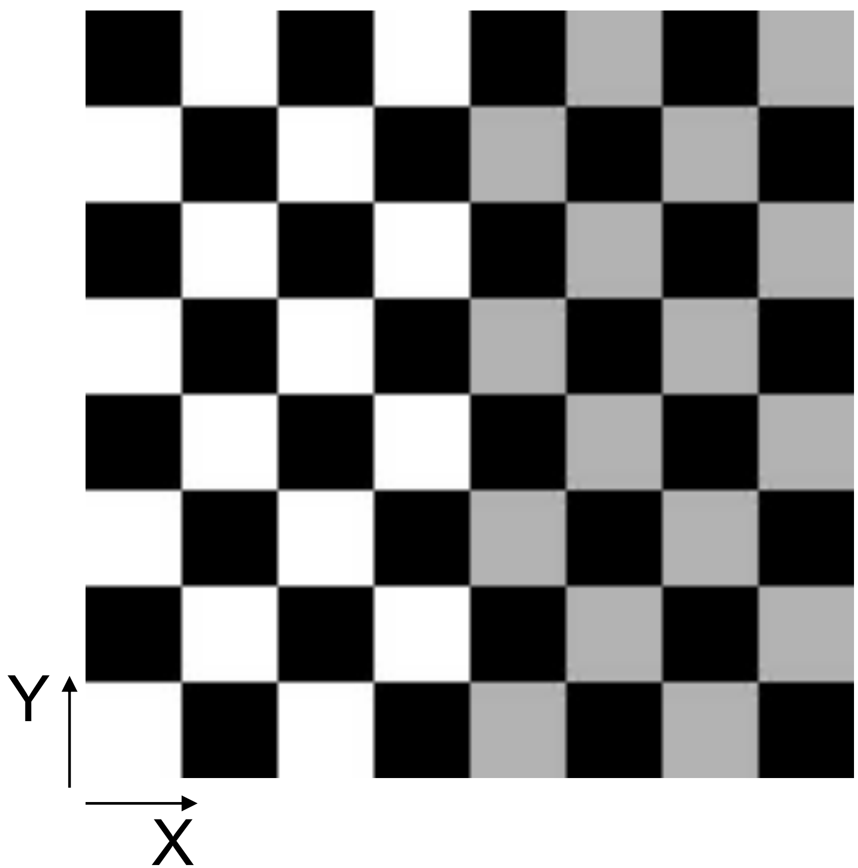


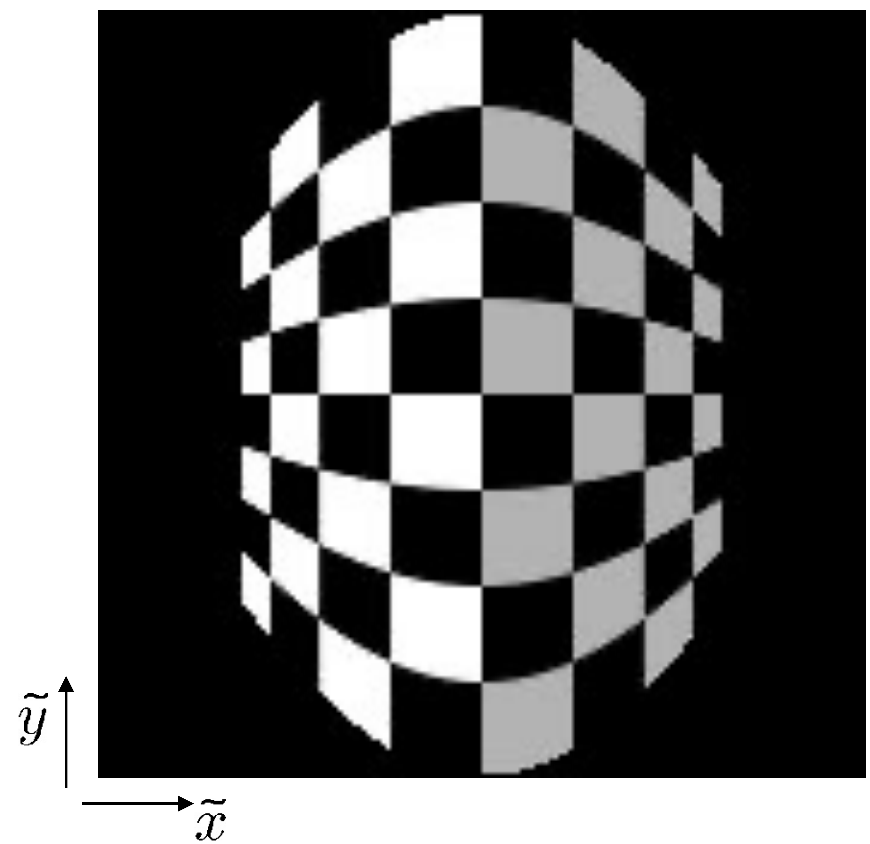
unwrapped cylinder



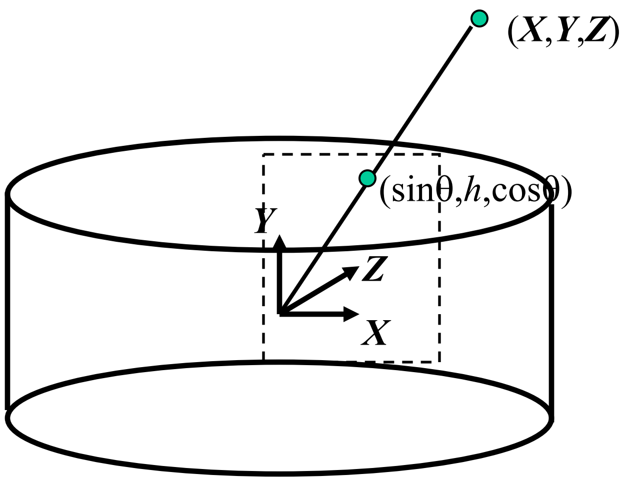
cylindrical image

Cylindrical Projection





Inverse Cylindrical projection



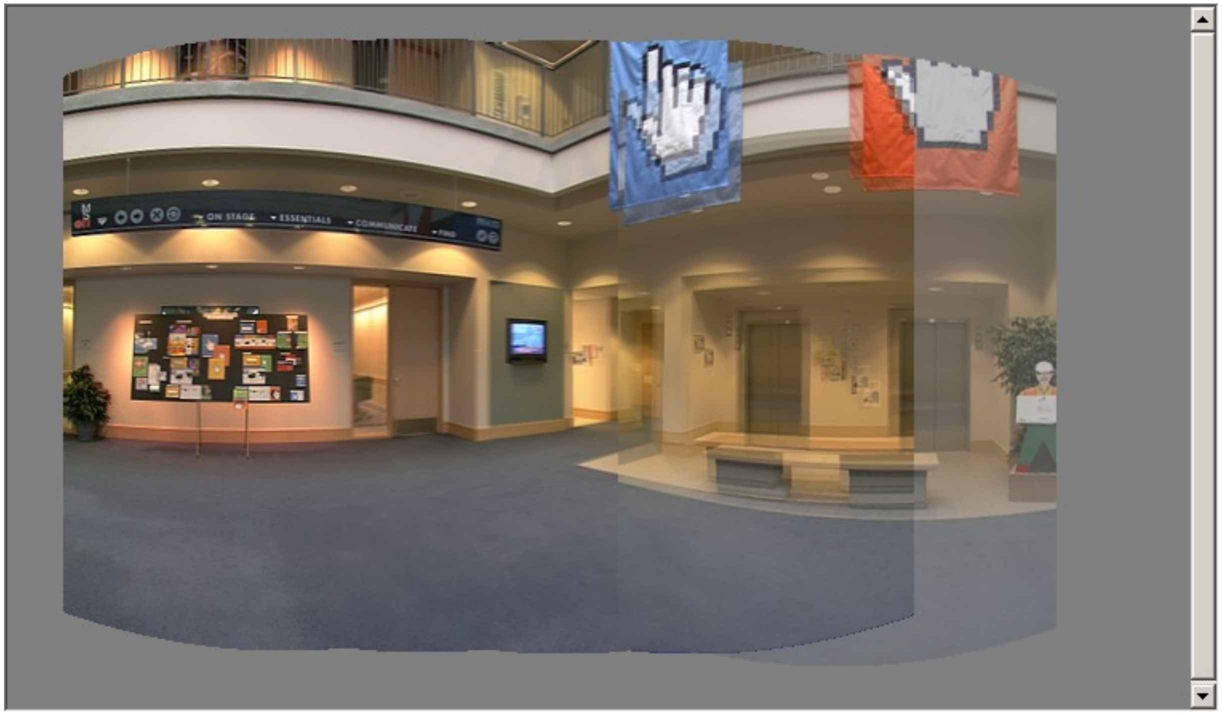
$$
\begin{aligned}
\theta &= (x_{cyl} - x_c)/f \\
h &= (y_{cyl} - y_c)/f \\
\hat{x} &= \sin\theta \\
\hat{y} &= h \\
\hat{z} &= \cos\theta \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c
\end{aligned}
$$

Cylindrical panoramas

Steps

- Reproject each image onto a cylinder
- Blend
- Output the resulting mosaic

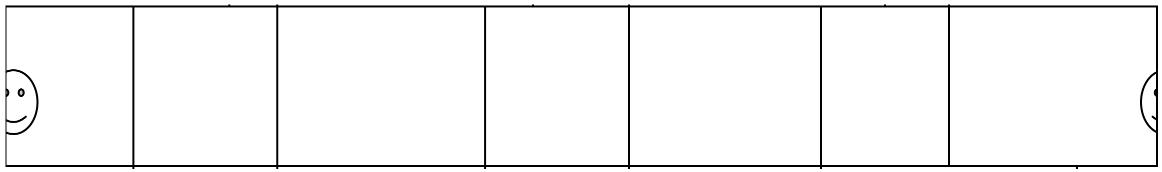
Cylindrical image stitching



What if you don't know the camera rotation?

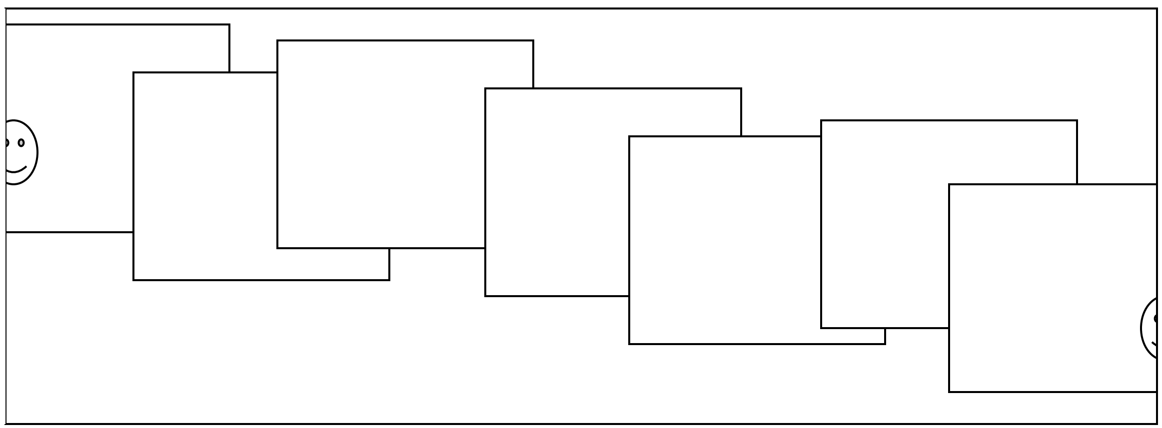
- Solve for the camera rotations
  - Note that a rotation of the camera is a **translation** of the cylinder!

Assembling the panorama



Stitch pairs together, blend, then crop

Problem: Drift



Vertical Error accumulation

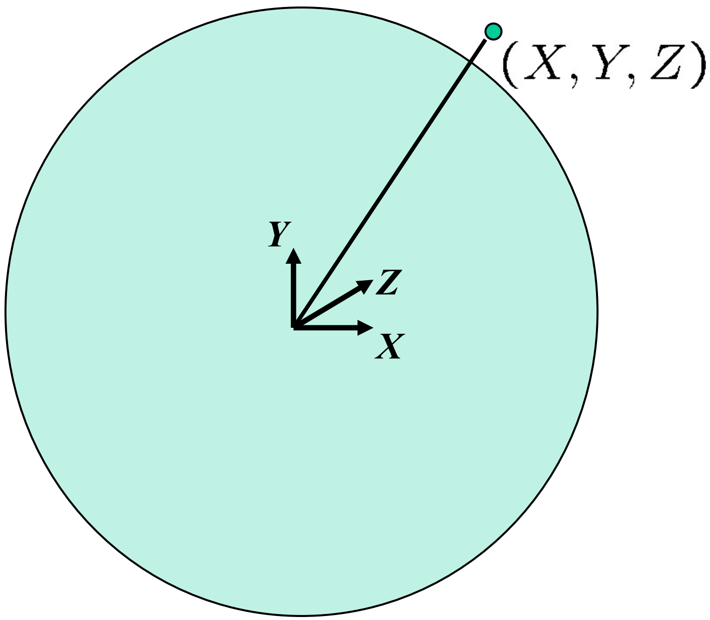
- small (vertical) errors accumulate over time
- apply correction so that sum = 0 (for 360° pan.)

Horizontal Error accumulation

- can reuse first/last image to find the right panorama radius

Full-view (360°) panoramas

Spherical projection



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

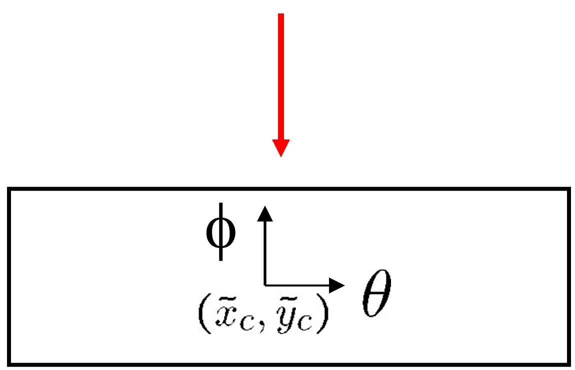
$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}}(X, Y, Z)$$

- Convert to spherical coordinates

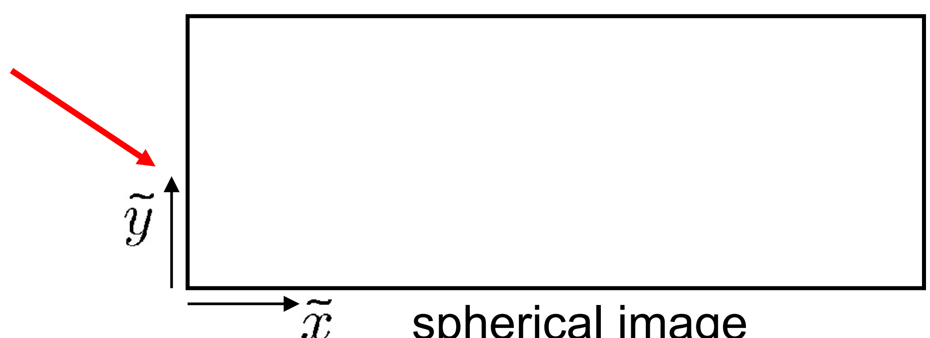
$$(\sin\theta\cos\phi, \sin\phi, \cos\theta\cos\phi) = (\hat{x}, \hat{y}, \hat{z})$$

- Convert to spherical image coordinates

$$(\tilde{x}, \tilde{y}) = (f\theta, fh) + (\tilde{x}_c, \tilde{y}_c)$$

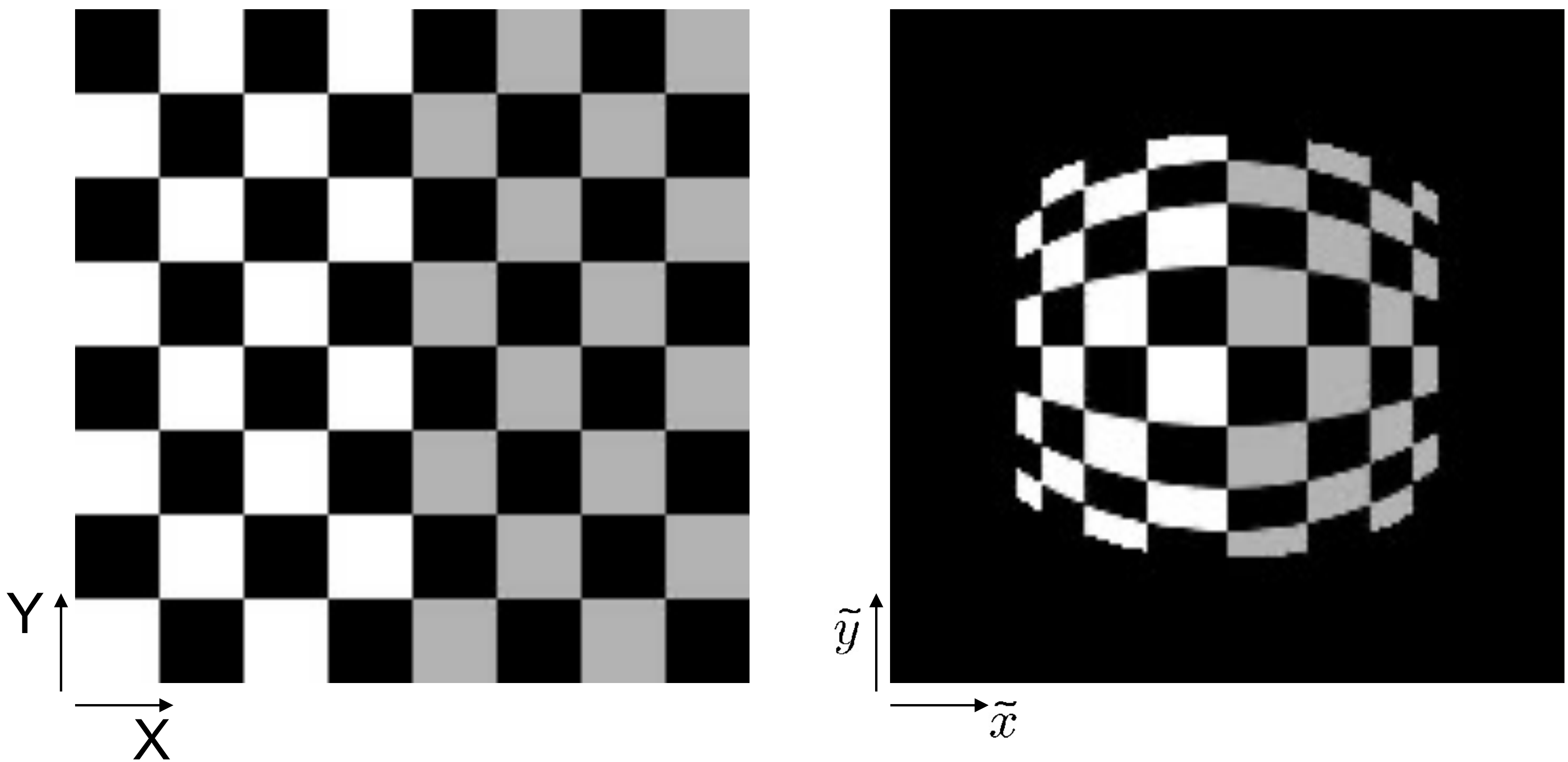


unwrapped sphere

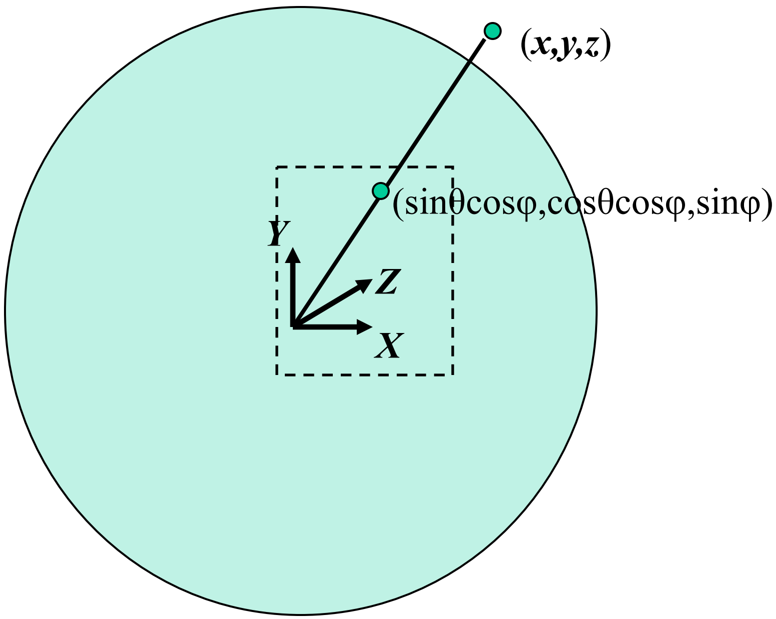


spherical image

Spherical Projection



Inverse Spherical projection



$$\theta = (x_{sph} - x_c)/f$$

$$\varphi = (y_{sph} - y_c)/f$$

$$\hat{x} = \sin\theta \cos\varphi$$

$$\hat{y} = \sin\varphi$$

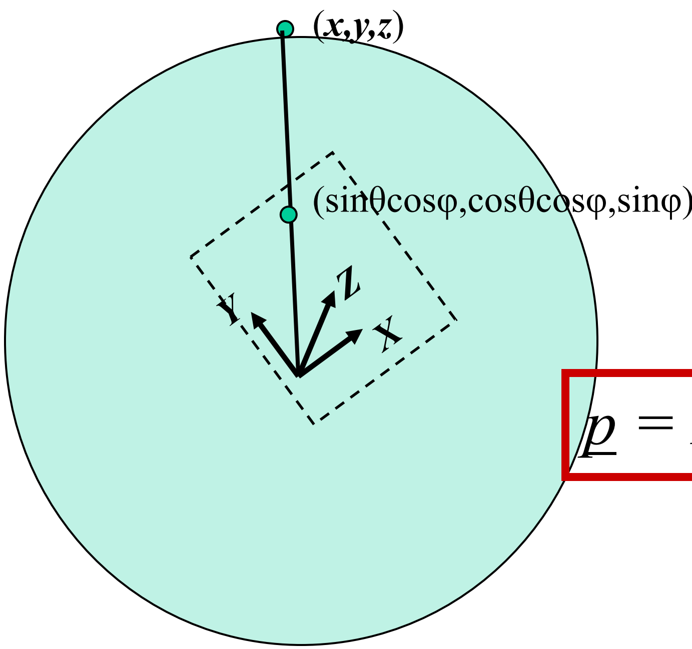
$$\hat{z} = \cos\theta\cos\varphi$$

$$x = f\hat{x}/\hat{z} + x_c$$

$$y = f\hat{y}/\hat{z} + y_c$$

3D rotation

Rotate image before placing on unrolled sphere



$$\underline{p} = \boldsymbol{R}\, p$$

$$
\begin{aligned}
\theta &= (x_{sph} - x_c)/f \\
\varphi &= (y_{sph} - y_c)/f \\
\hat{x} &= \sin\theta \cos\varphi \\
\hat{y} &= \sin\varphi \\
\hat{z} &= \cos\theta \cos\varphi \\
x &= f\underline{\hat{x}}/\underline{\hat{z}} + x_c \\
y &= f\underline{\hat{y}}/\underline{\hat{z}} + y_c
\end{aligned}
$$

Full-view Panorama

Other projections are possible

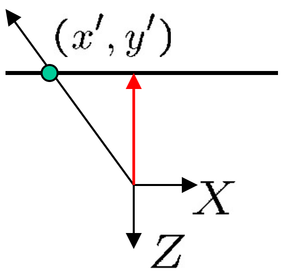
You can stitch on the plane and then warp the resulting panorama

- What's the limitation here?

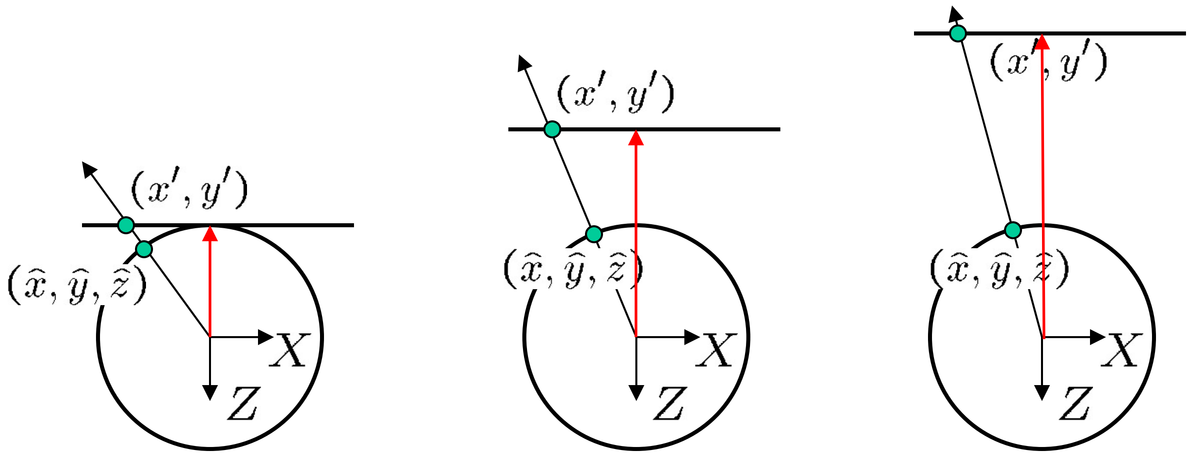
Or, you can use these as stitching surfaces

- But there is a catch…

Cylindrical reprojection



top-down view



Focal length – the dirty secret…

Image 384x300

f = 180 (pixels)

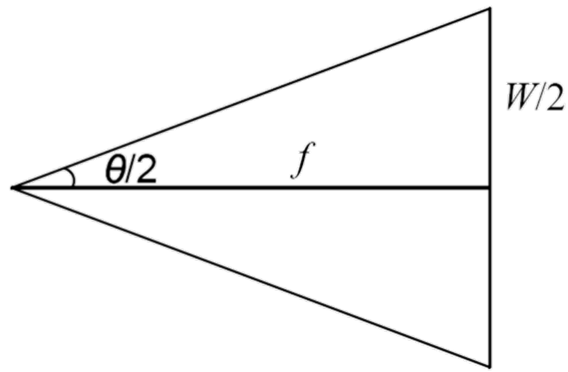
f = 280

f = 380

What's your focal length, buddy?

Focal length is (highly!) camera dependant

- Can get a rough estimate by measuring FOV:

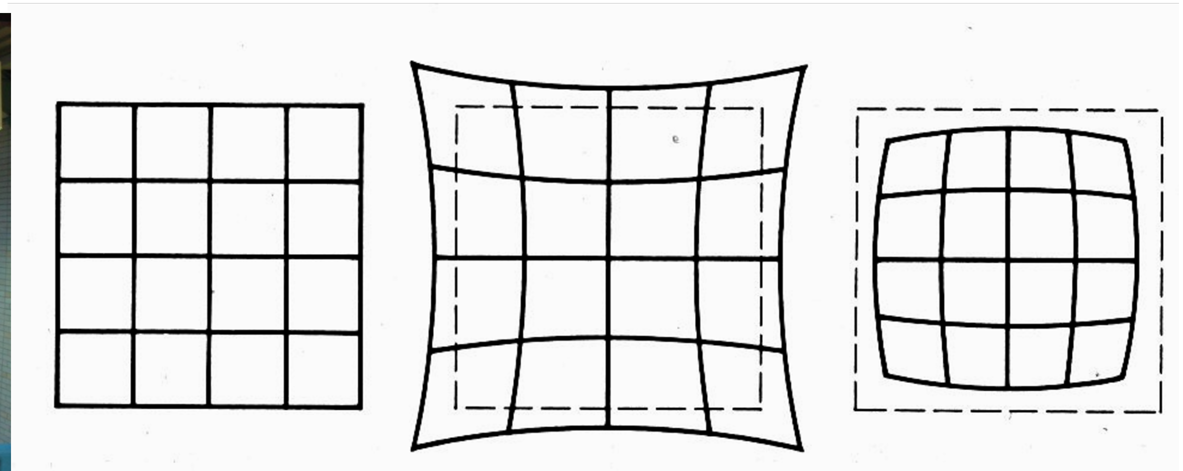


- Can use the EXIF data tag (might not give the right thing)
- Can use several images together and try to find f that would make them match
- Can use a known 3D object and its projection to solve for f
- Etc.

There are other camera parameters too:

- Optical center, non-square pixels, lens distortion, etc.

Distortion



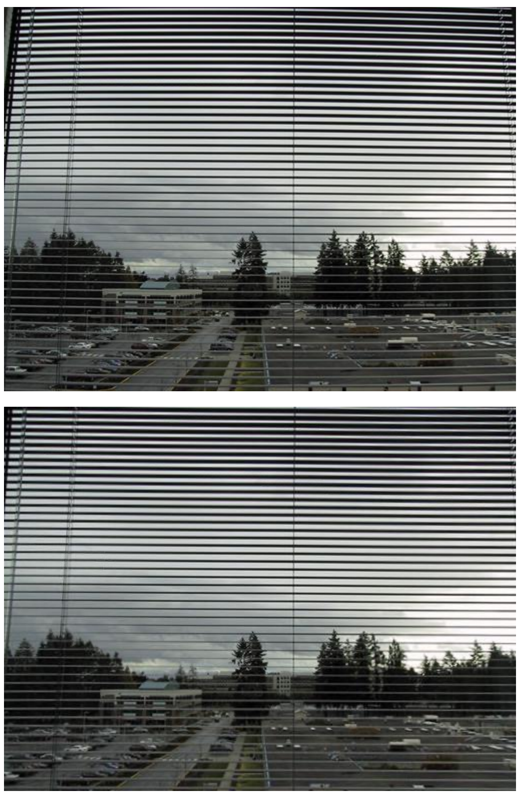
No distortion Pin cushion Barrel

Radial distortion of the image

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

Radial distortion

Correct for “bending” in wide field of view lenses

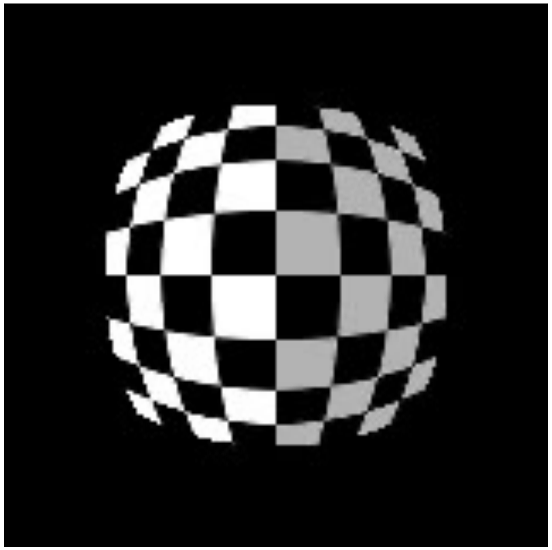


$$
\begin{aligned}
\hat{r}^2 &= \hat{x}^2 + \hat{y}^2 \\
\hat{x}' &= \hat{x}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
\hat{y}' &= \hat{y}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
x &= f\hat{x}'/\hat{z} + x_c \\
y &= f\hat{y}'/\hat{z} + y_c
\end{aligned}
$$

Use this instead of normal projection

Polar Projection

Extreme “bending” in ultra-wide fields of view



$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2$$

$$(\cos\theta\sin\phi, \sin\theta\sin\phi, \cos\phi) = s\,(x, y, z)$$

uations become

$$x' = s\phi\cos\theta = s\frac{x}{r}\tan^{-1}\frac{r}{z},$$
$$y' = s\phi\sin\theta = s\frac{y}{r}\tan^{-1}\frac{r}{z},$$

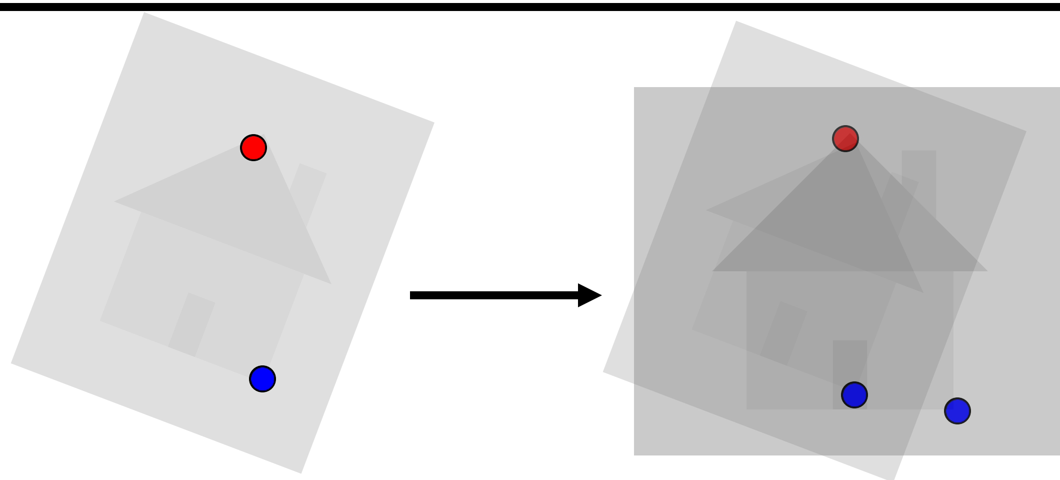
Up till now:

Known correspondences via clicking!

How to make it automatic??

Live Homography…

Image Alignment



How do we align two images automatically?

Two broad approaches:

- Feature-based alignment
  - Find a few matching features in both images
  - compute alignment
- Direct (pixel-based) alignment
  - Search for alignment where most pixels agree

Direct Alignment

The simplest approach is a brute force search (hw1)

- Need to define image matching function
  - SSD, Normalized Correlation, edge matching, etc.
- Search over all parameters within a reasonable range:

e.g. for translation:

```
for tx=x0:step:x1,
  for ty=y0:step:y1,
     compare image1(x,y) to image2(x+tx,y+ty)
  end;
end;
```

Need to pick correct `x0,x1` and `step`

- What happens if `step` is too large?

Direct Alignment (brute force)

What if we want to search for more complicated transformation, e.g. homography?

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

```
for a=a0:astep:a1,
  for b=b0:bstep:b1,
    for c=c0:cstep:c1,
      for d=d0:dstep:d1,
        for e=e0:estep:e1,
          for f=f0:fstep:f1,
             for g=g0:gstep:g1,
                for h=h0:hstep:h1,
     compare image1 to H(image2)
end; end; end; end; end; end; end; end;
```

Problems with brute force

Not realistic

- Search in $O(N^8)$ is problematic
- Not clear how to set starting/stopping value and step

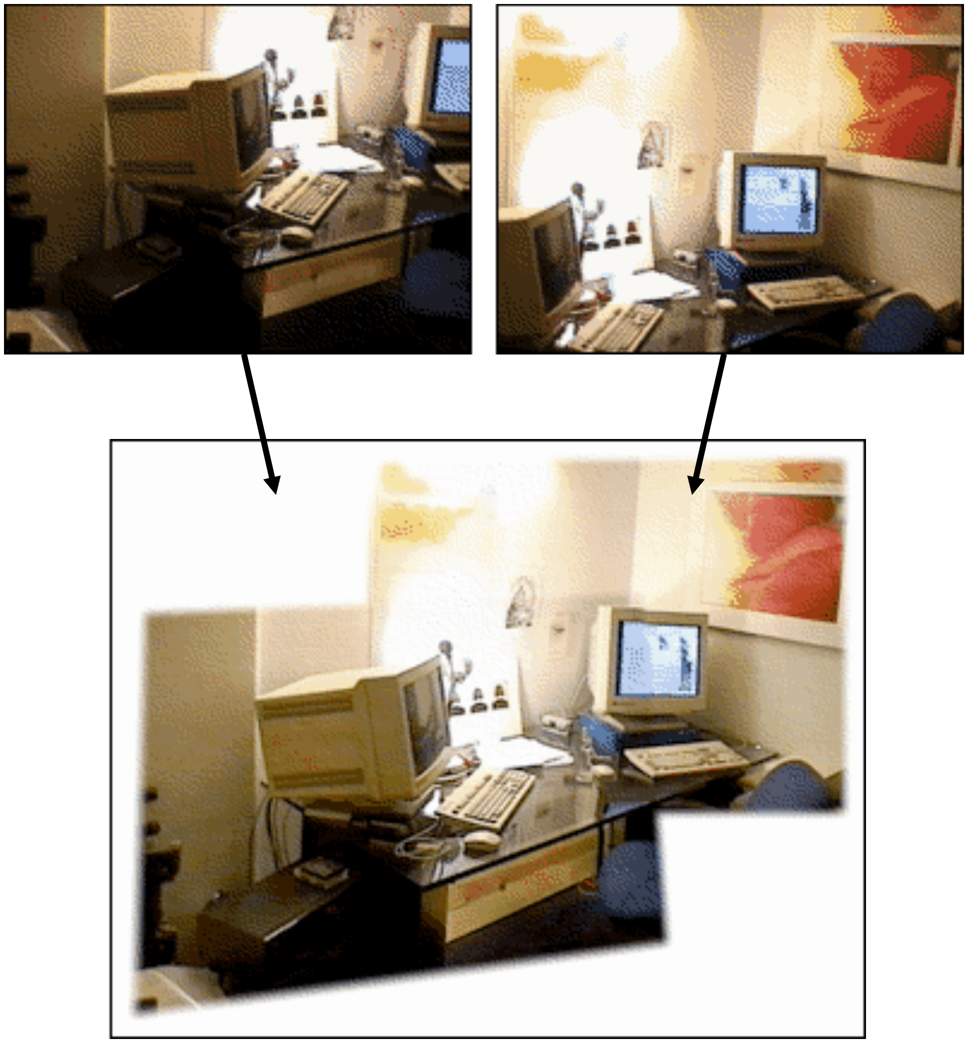
What can we do?

- Use pyramid search to limit starting/stopping/step values

Alternative: gradient decent on the error function

- i.e. how do I tweak my current estimate to make the SSD error go down?
- Can do sub-pixel accuracy
- BIG assumption?
  – Images are already almost aligned (<2 pixels difference!)
  – Can improve with pyramid
- Same tool as in **motion estimation**

Image alignment



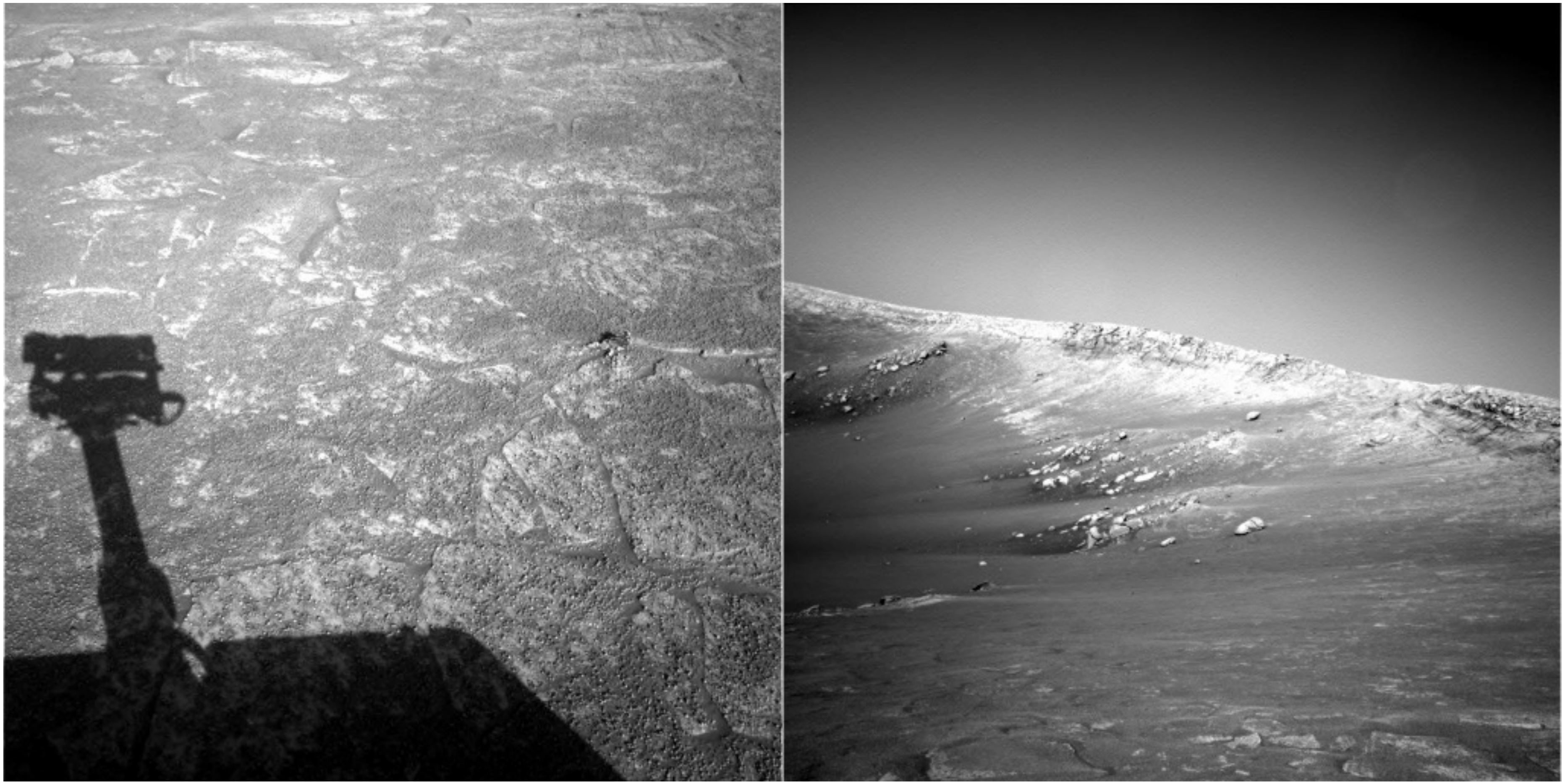
Feature-based alignment

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: as per Project 5, Part I

How do we <u>choose</u> good features automatically?

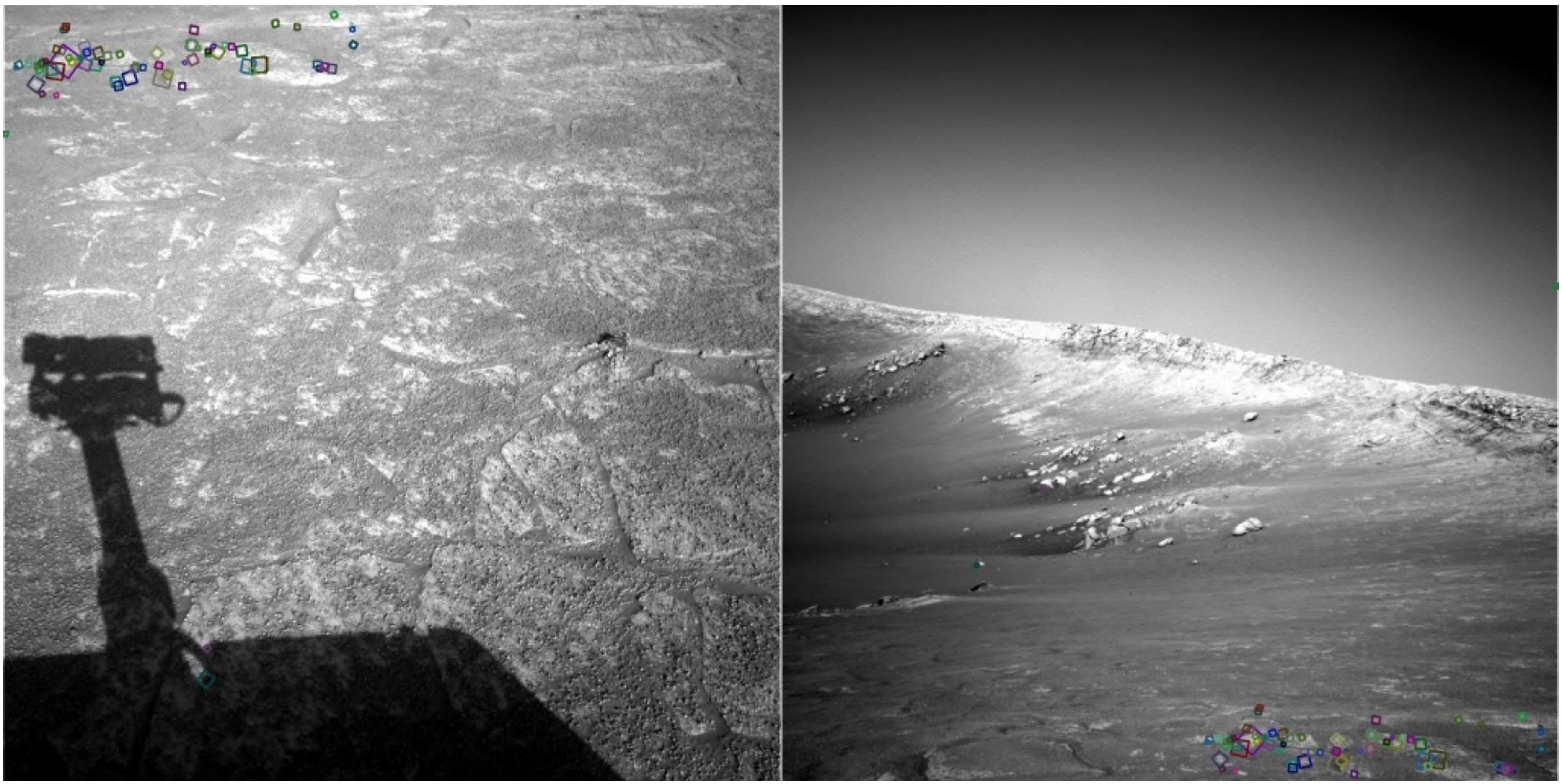
- They must be prominent in both images
- Easy to localize
- Think how you did that by hand in Project #6 Part I
- Corners!

A hard feature matching problem



NASA Mars Rover images

Answer below (look for tiny colored squares…)



NASA Mars Rover images
with SIFT feature matches
Figure by Noah Snavely

Feature Detection

Feature Matching

How do we match the features between the images?

- Need a way to <u>describe</u> a region around each feature
  - e.g. image patch around each feature
- Use successful matches to estimate homography
  - Need to do something to get rid of outliers

Issues:

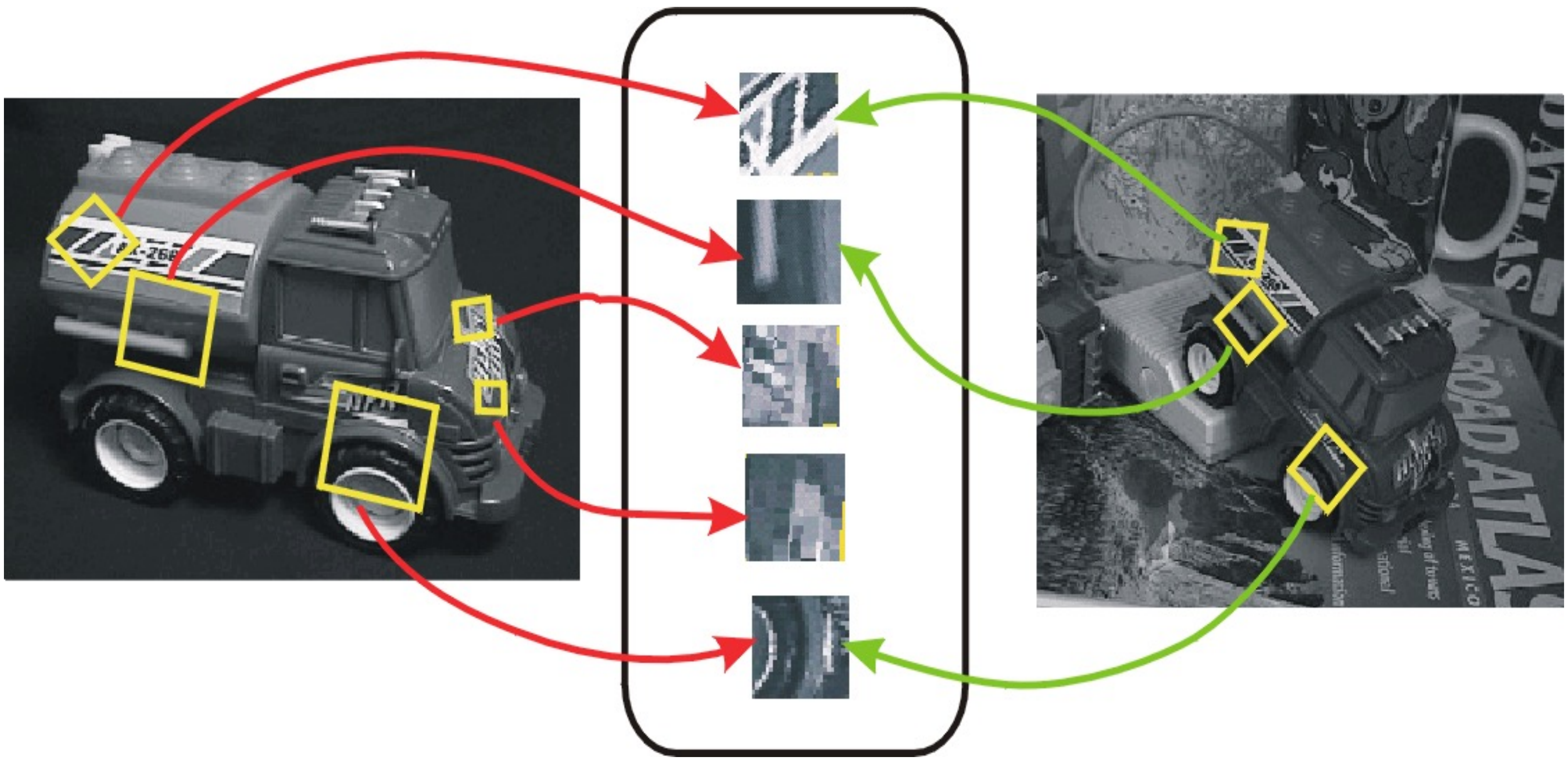
- What if the image patches for several interest points look similar?
  - Make patch size bigger
- What if the image patches for the same feature look different due to scale, rotation, etc.
  - Need an invariant descriptor

Invariant Feature Descriptors

Schmid & Mohr 1997, Lowe 1999, Baumberg 2000, Tuytelaars & Van Gool 2000, Mikolajczyk & Schmid 2001, Brown & Lowe 2002, Matas et. al. 2002, Schaffalitzky & Zisserman 2002

Invariant Local Features

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Features Descriptors

Applications

Feature points are used for:

- Image alignment (homography, fundamental matrix)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other