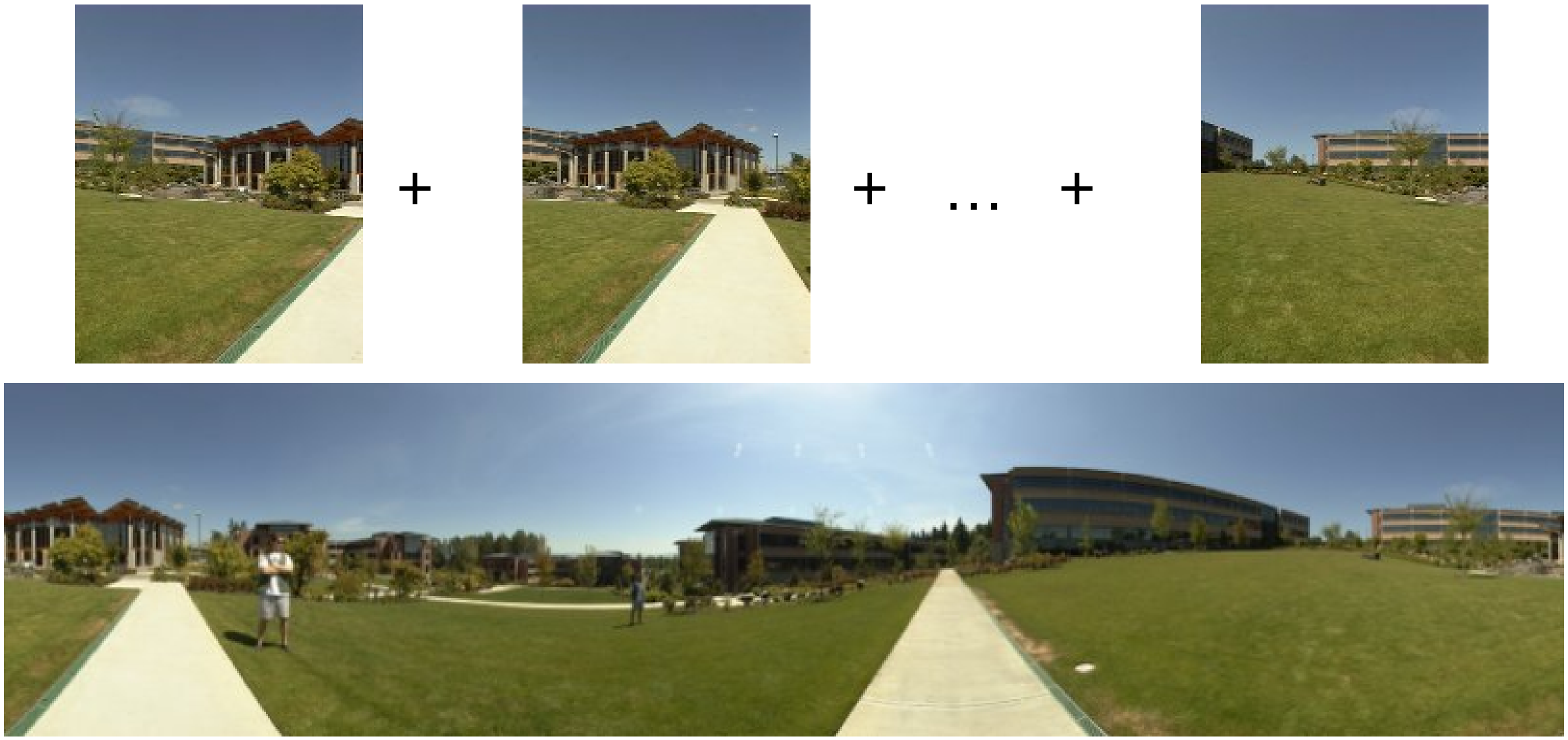
Mosaics

VR Seattle: http://www.vrseattle.com/
Full screen panoramas (cubic): http://www.panoramas.dk/
Mars: http://www.panoramas.dk/fullscreen3/f2_mars97.html

Today's Readings

- Szeliski and Shum paper (sections 1 and 2, skim the rest)
  - http://www.cs.washington.edu/education/courses/455/08wi/readings/szeliskiShum97.pdf

Image Mosaics



Goal

- Stitch together several images into a seamless composite

How to do it?

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
- Shift the second image to overlap with the first
- Blend the two together to create a mosaic
- If there are more images, repeat

Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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**

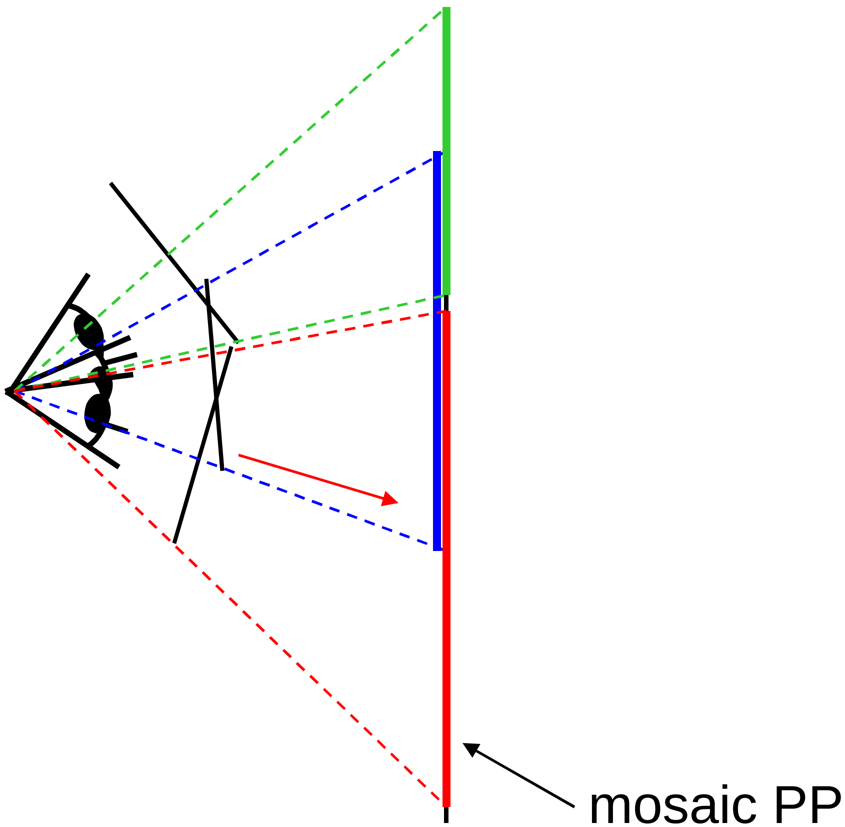
- By Matthew Brown and David Lowe
- http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Aligning images

How to account for warping?

- Translations are not enough to align the images
- Photoshop demo

Image reprojection



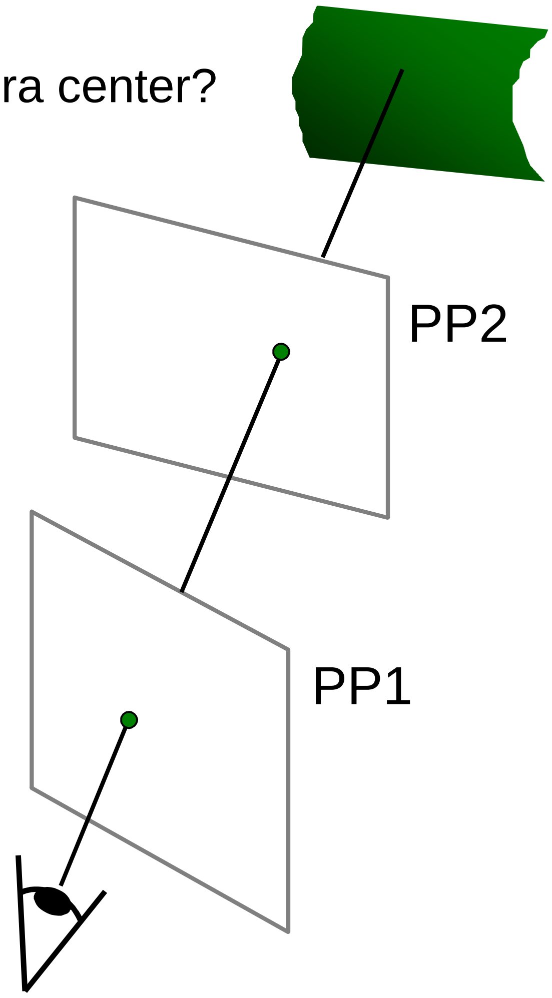
The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane

Image reprojection

Basic question

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

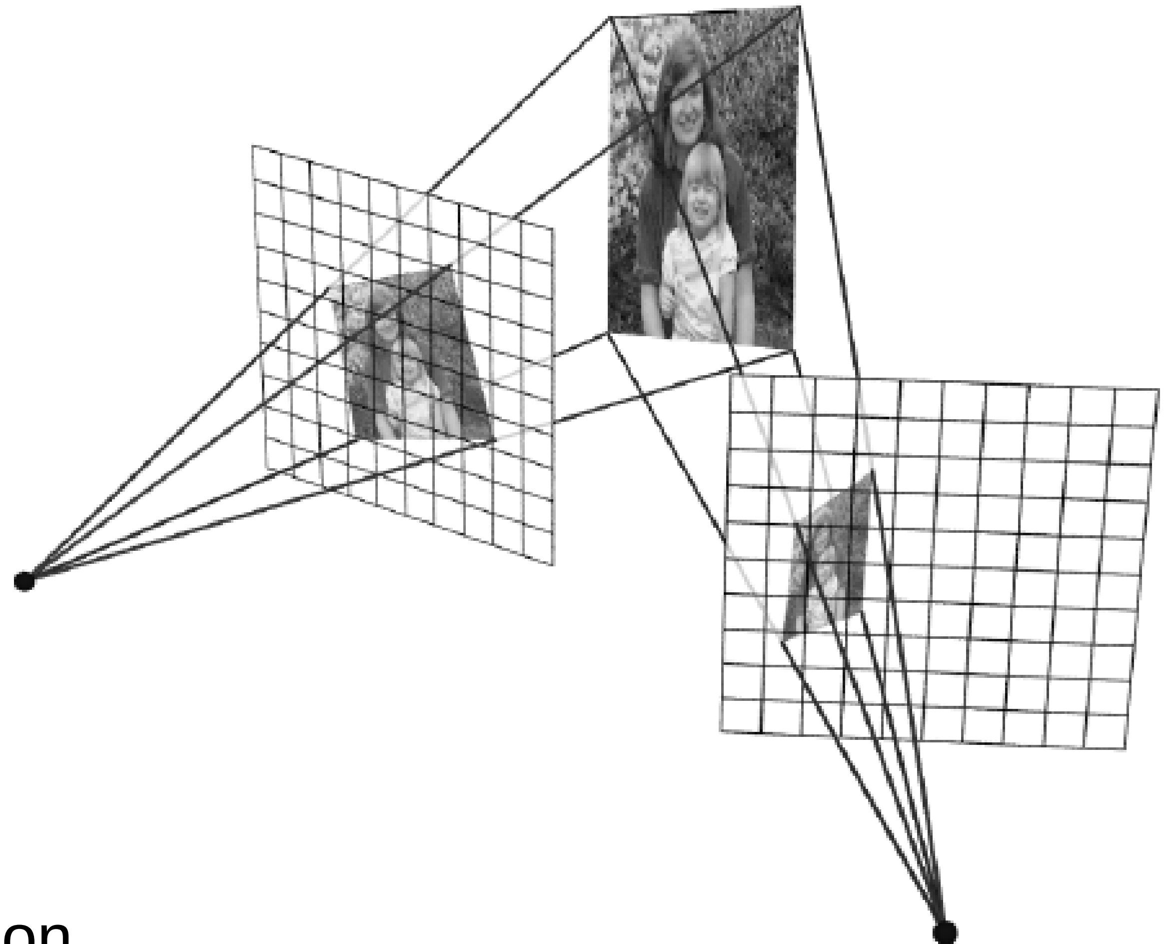


Answer

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

Don't need to know what's in the scene!

Image reprojection



Observation

- Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another

Homographies

Perspective projection of a plane

- Lots of names for this:
  - **homography**, texture-map, colineation, planar projective map
- Modeled as a 2D warp using homogeneous coordinates

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{p'} \qquad \mathbf{H} \qquad \mathbf{p}$$

To apply a homography **H**

- Compute **p'** = **Hp** (regular matrix multiply)
- Convert **p'** from homogeneous to image coordinates
  - divide by w (third) coordinate

Image warping with homographies

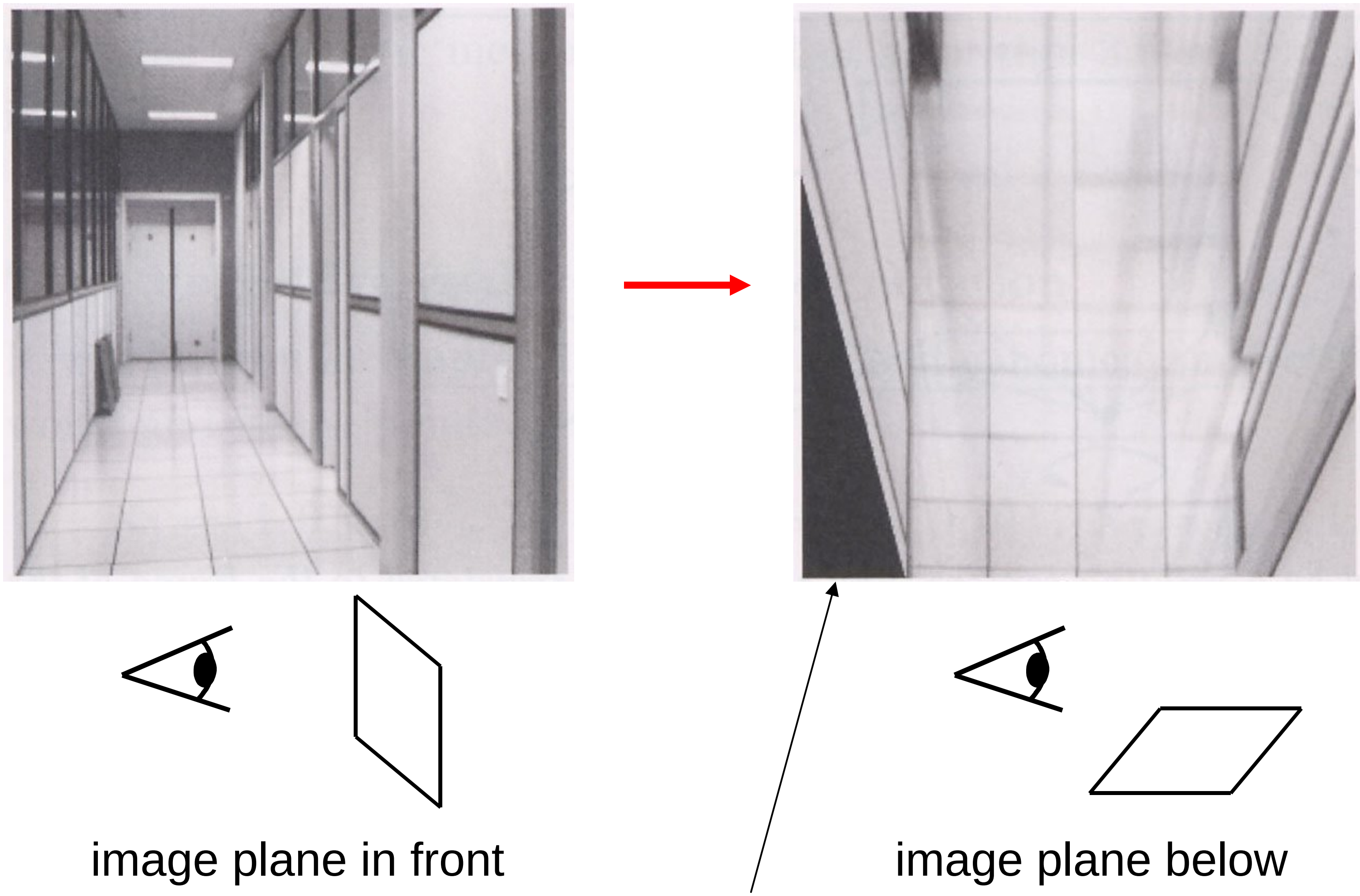


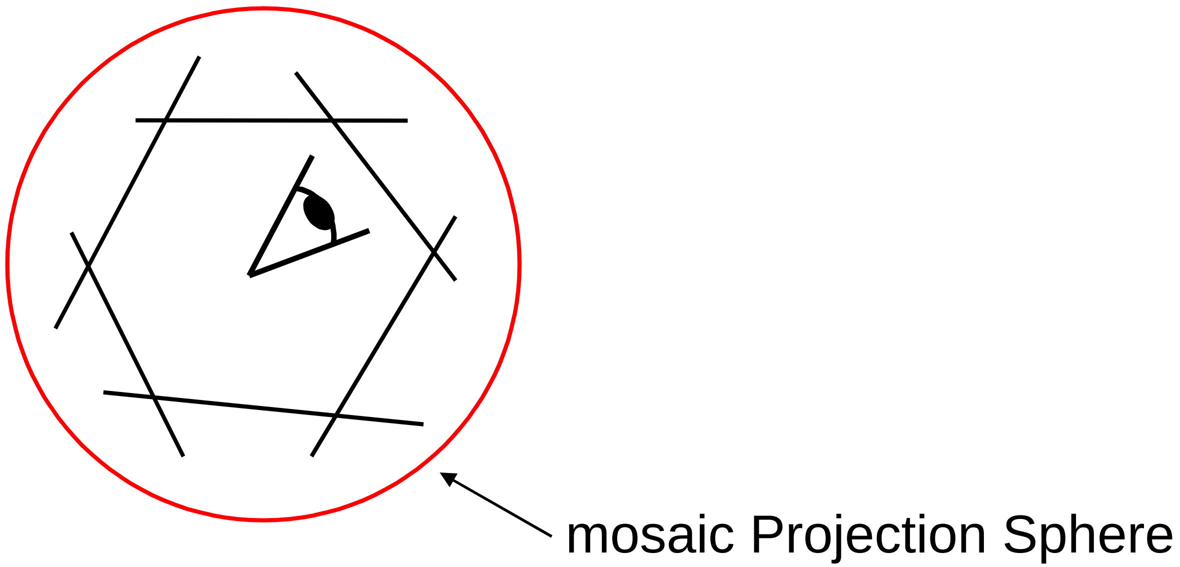
image plane in front

image plane below

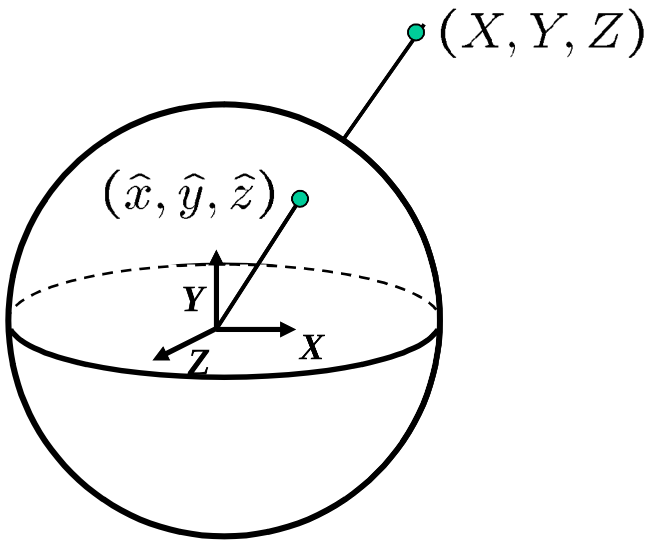
black area
where no pixel
maps to

Panoramas

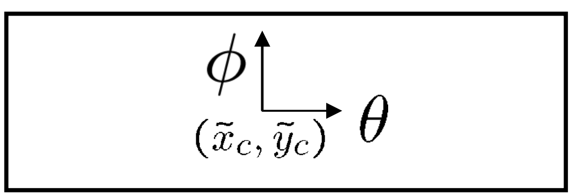
What if you want a 360° field of view?



Spherical projection



unit sphere



unwrapped 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

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

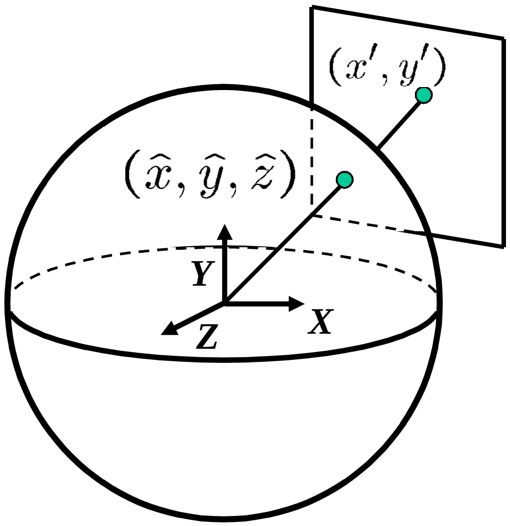
- Convert to spherical image coordinates

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

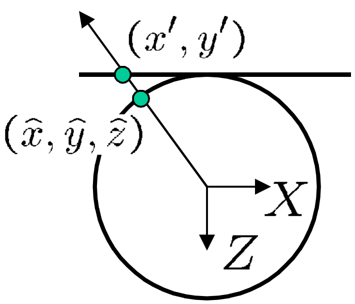
  – s defines size of the final image
    » often convenient to set s = camera focal length

Spherical image

Spherical reprojection



side view

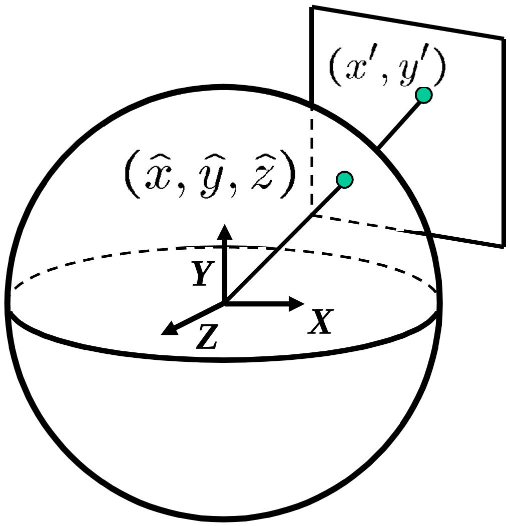


top-down view

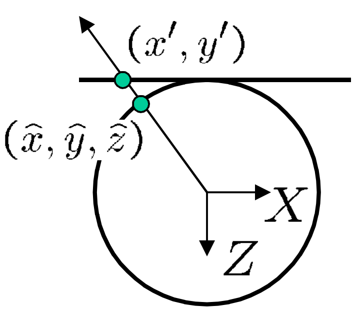
How to map sphere onto a flat image?

- $(\hat{x}, \hat{y}, \hat{z})$ to (x', y')

Spherical reprojection



side view



top-down view

How to map sphere onto a flat image?

- $(\hat{x}, \hat{y}, \hat{z})$ to (x', y')
  - Use image projection matrix!
    - or use the version of projection that properly accounts for radial distortion, as discussed in projection slides. This is what you'll do for project 2.

Correcting radial distortion

from Helmut Dersch

Modeling distortion

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

$$x'_n = \widehat{x}/\widehat{z}$$
$$y'_n = \widehat{y}/\widehat{z}$$

Apply radial distortion

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

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

Spherical reprojection

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

Map image to spherical coordinates

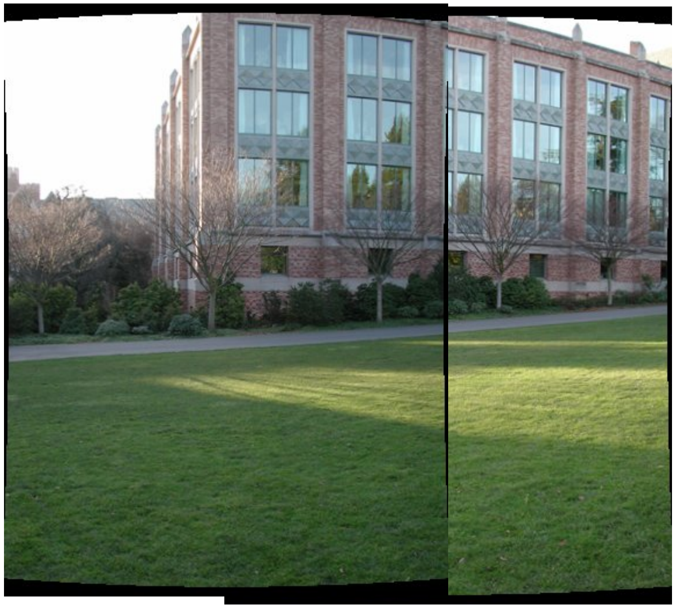
- need to know the focal length

Aligning spherical images

Suppose we rotate the camera by θ about the vertical axis

- How does this change the spherical image?

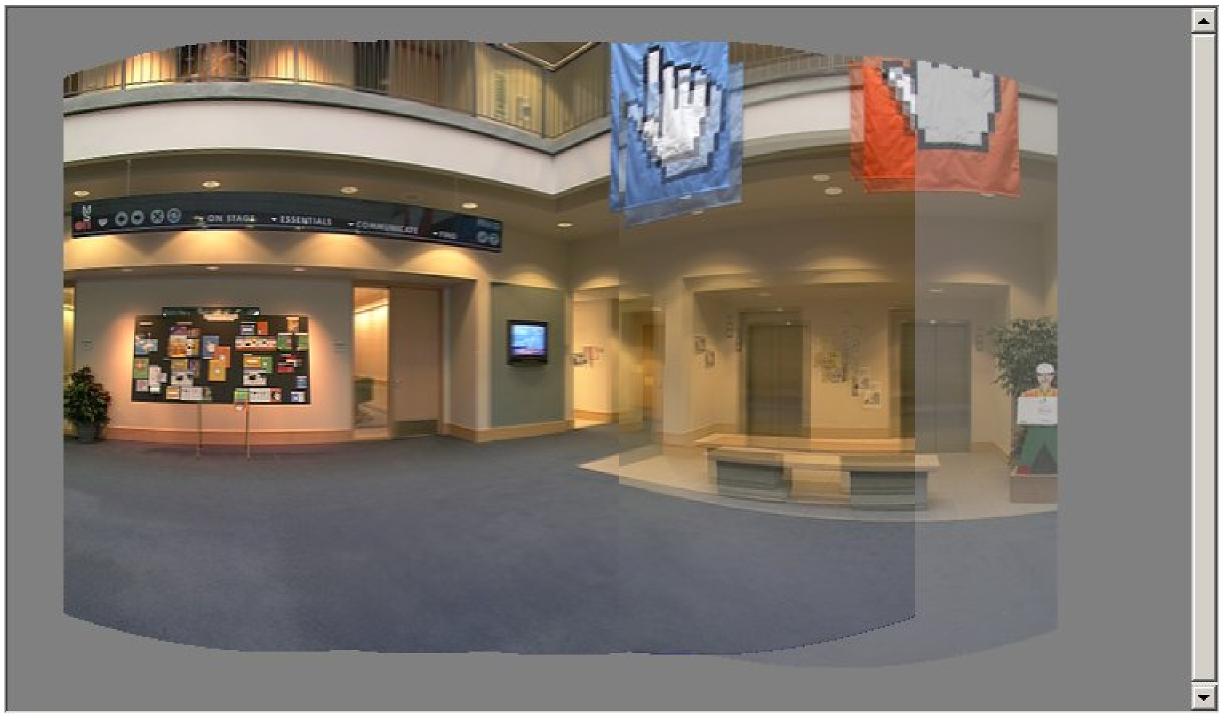
Aligning spherical images



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

Spherical image stitching



What if you don't know the camera rotation?

- Solve for the camera rotations
  - Note that a pan (rotation) of the camera is a **translation** of the sphere!
  - Use feature matching to solve for translations of spherical-warped images

Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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

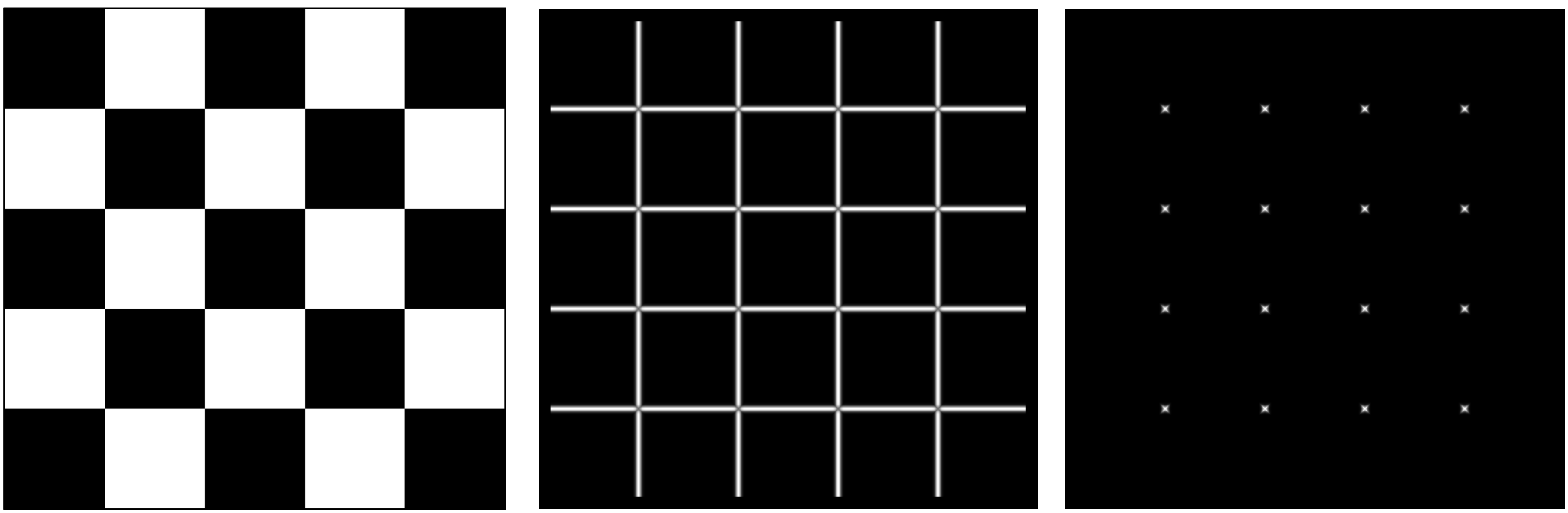
Roughly based on **Autostitch**

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

Feature detection summary

Here's what you do

- Compute the gradient at each point in the image
- Create the ***H*** matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response (λ_- > threshold)
- Choose those points where λ_- is a local maximum as features



I λ_+ λ_-

The Harris operator

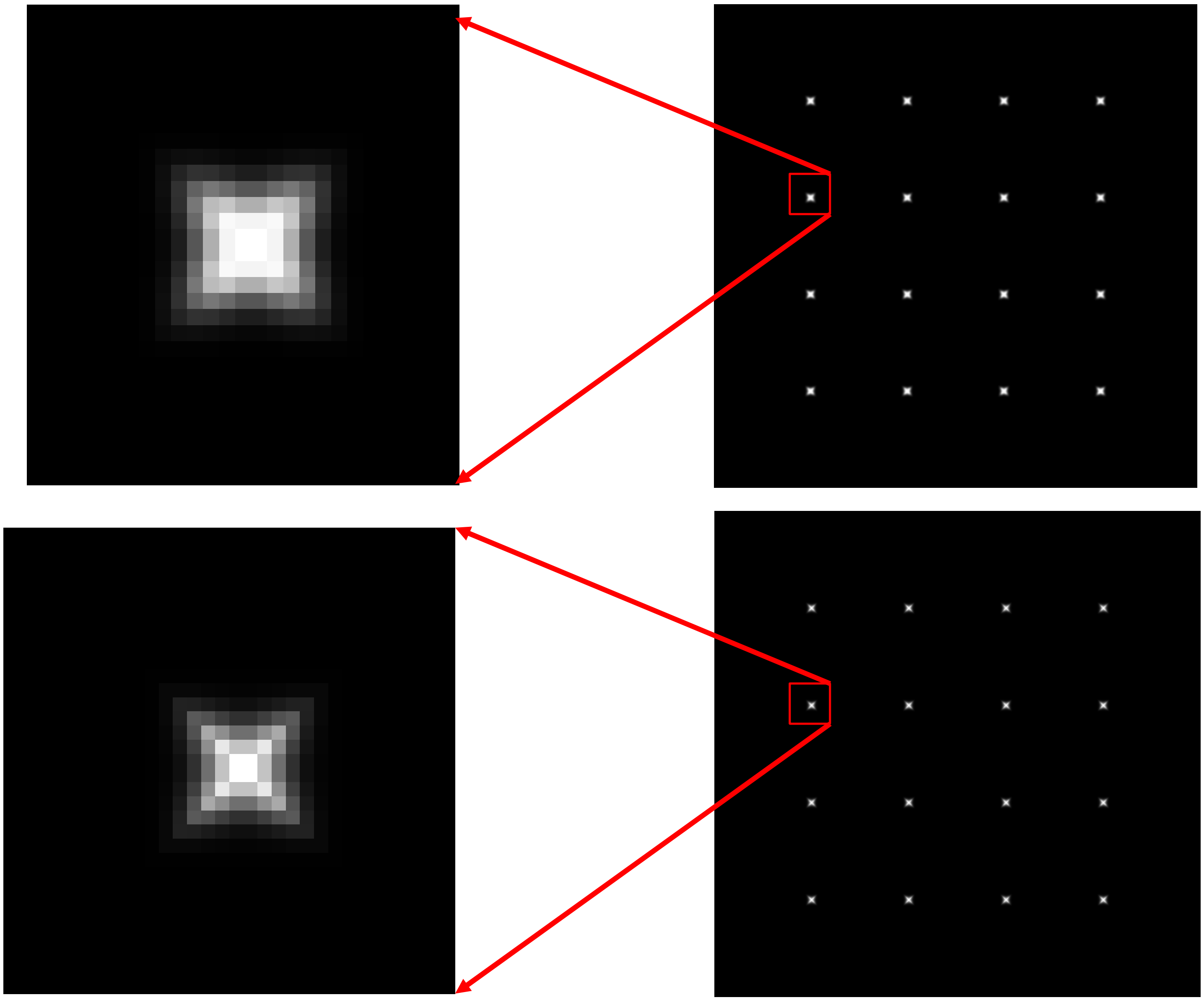
λ_- is a variant of the "Harris operator" for feature detection

$$f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

$$= \frac{determinant(H)}{trace(H)}$$

- The *trace* is the sum of the diagonals, i.e., $trace(H) = h_{11} + h_{22}$
- Very similar to λ_- but less expensive (no square root)
- Called the "Harris Corner Detector" or "Harris Operator"
- Lots of other detectors, this is one of the most popular

The Harris operator



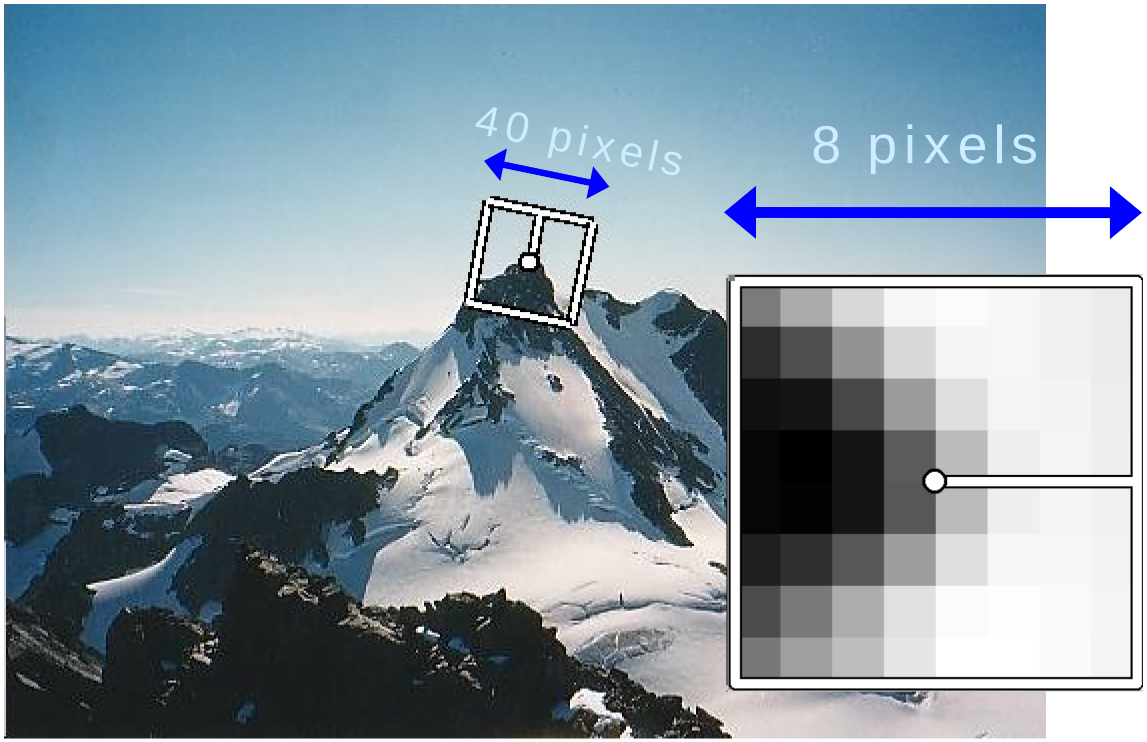
Harris operator

λ_-

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window



Adapted from slide by Matthew Brown

Feature matching

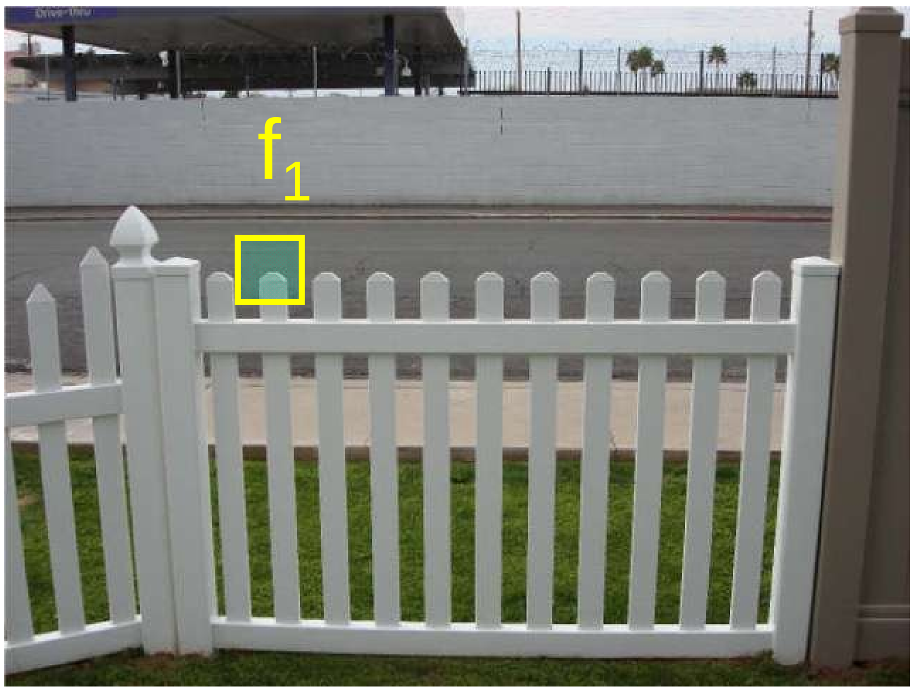
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

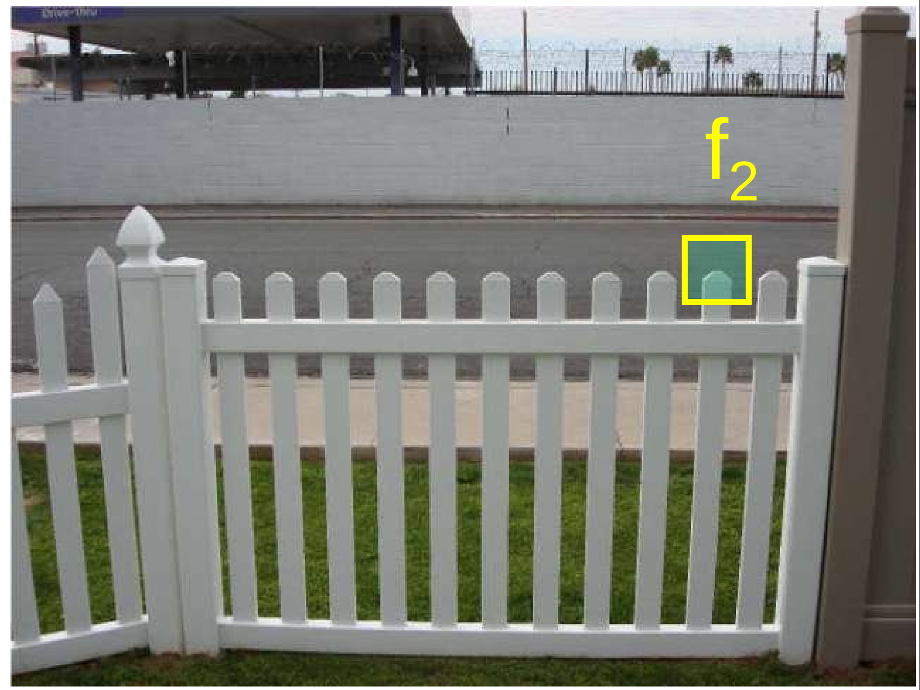
Feature distance

How to define the difference between two features f_1, f_2?

- Simple approach is SSD(f_1, f_2)
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches



I_1

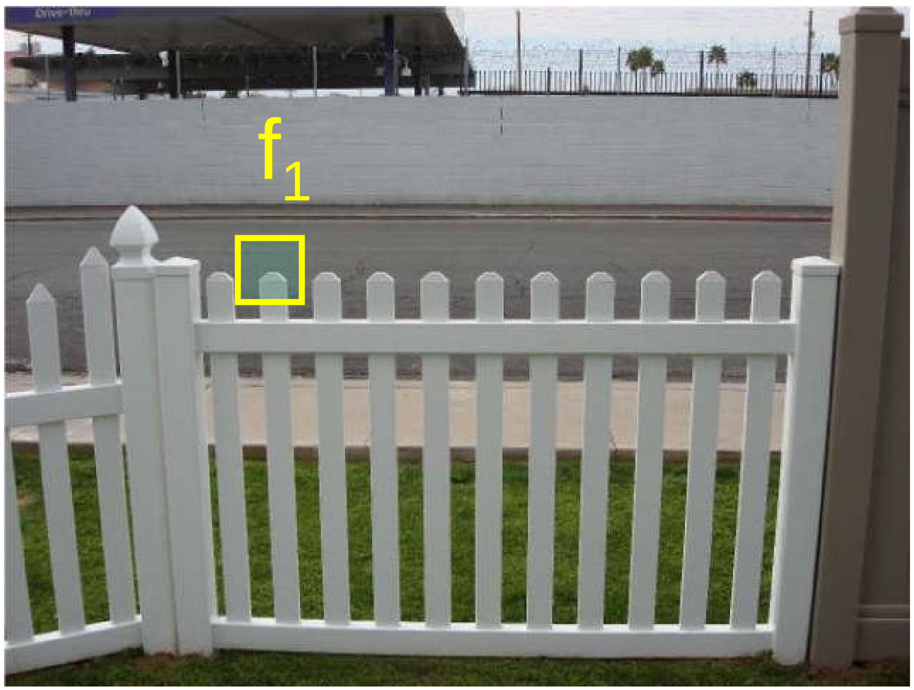


I_2

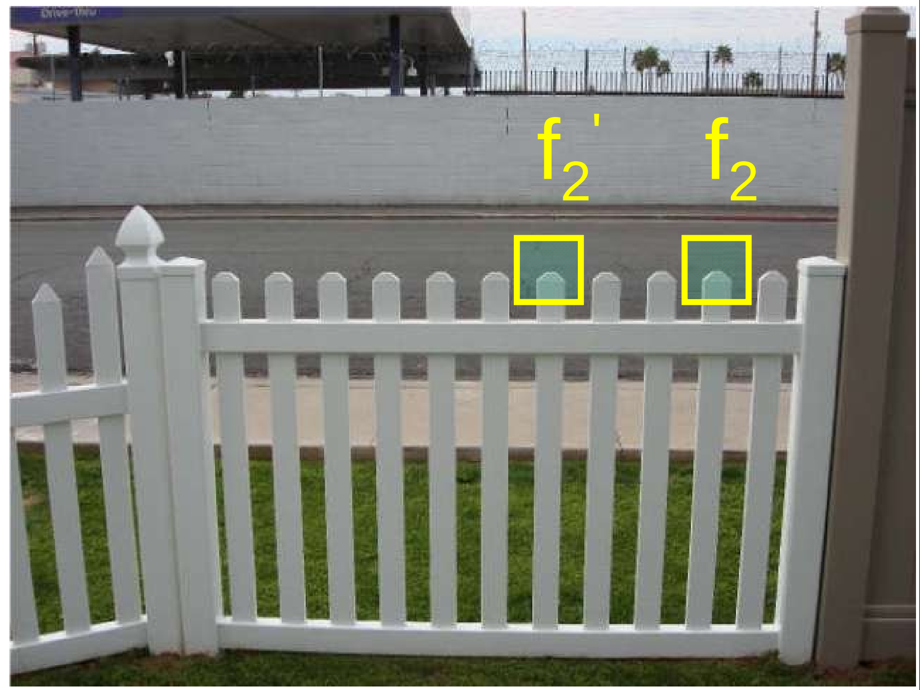
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = $SSD(f_1, f_2) / SSD(f_1, f_2')$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives small values for ambiguous matches



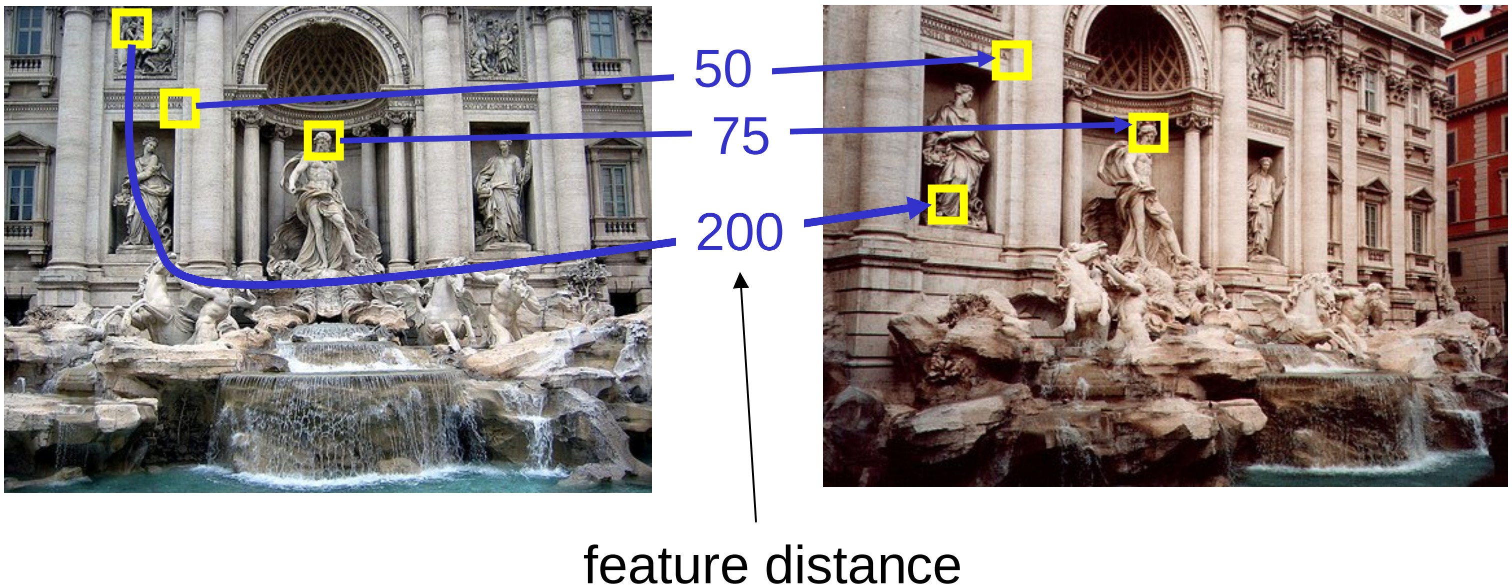
I_1



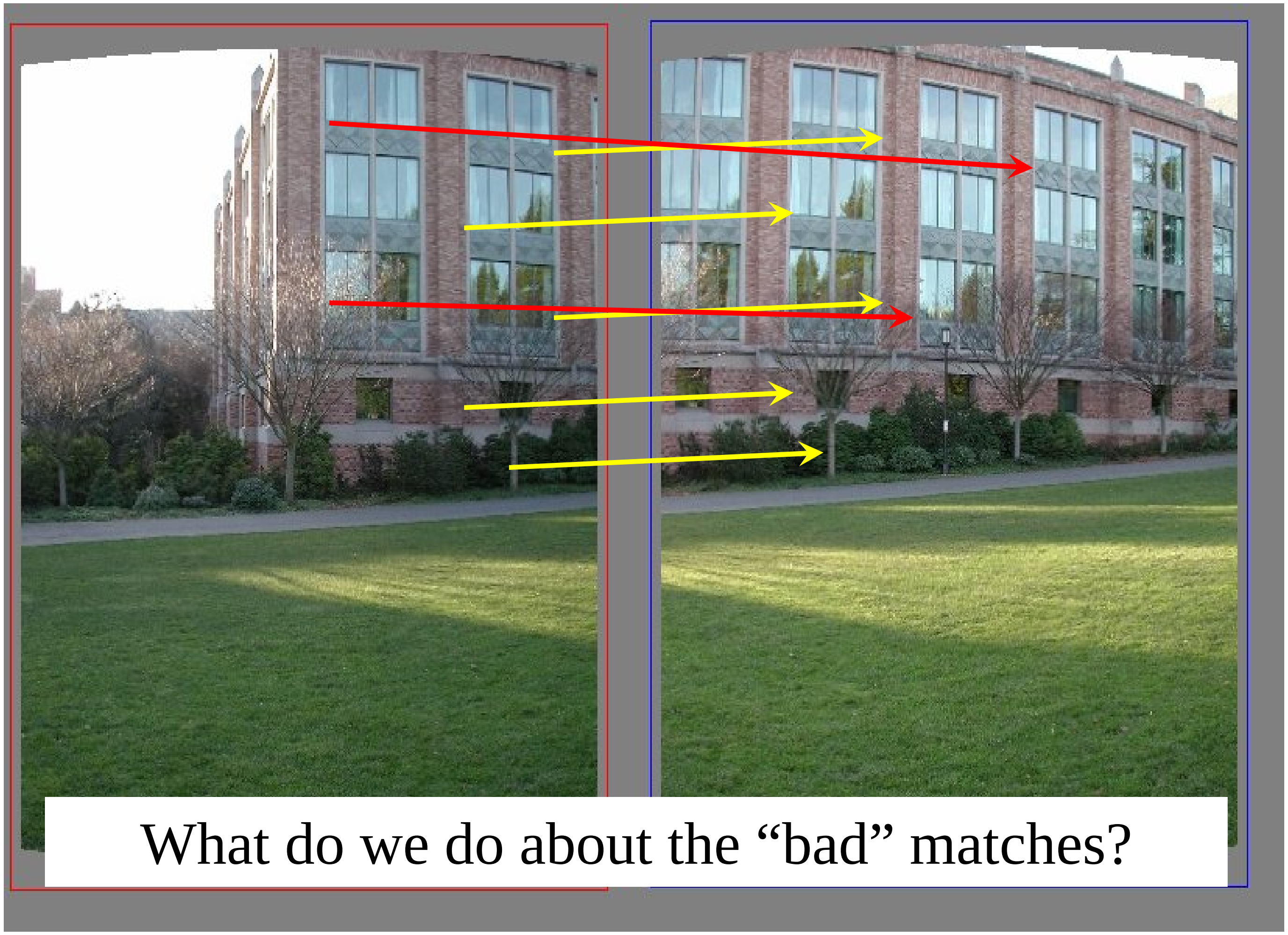
I_2

Evaluating the results

How can we measure the performance of a feature matcher?



Computing image translations



What do we do about the "bad" matches?

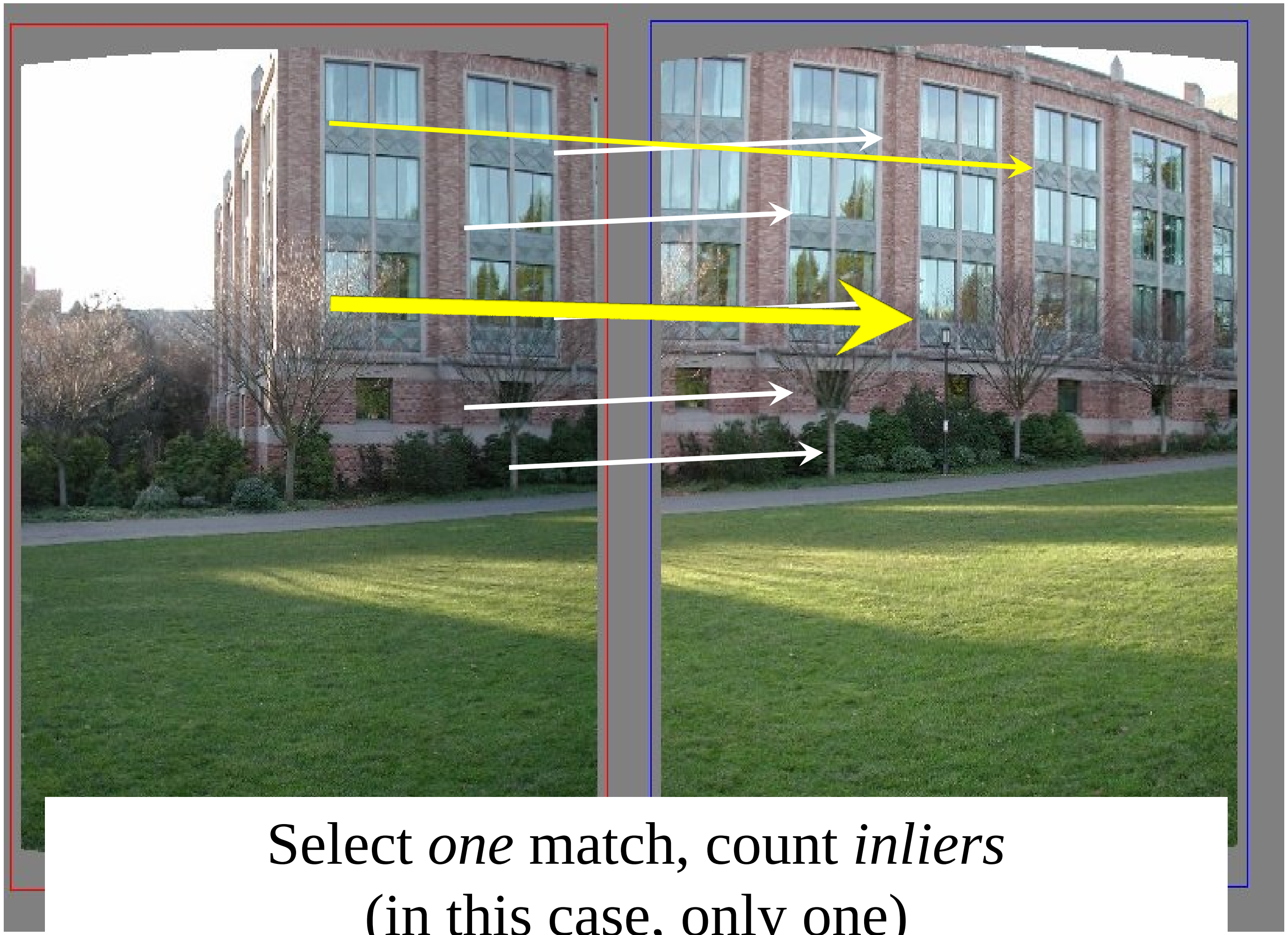
Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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**

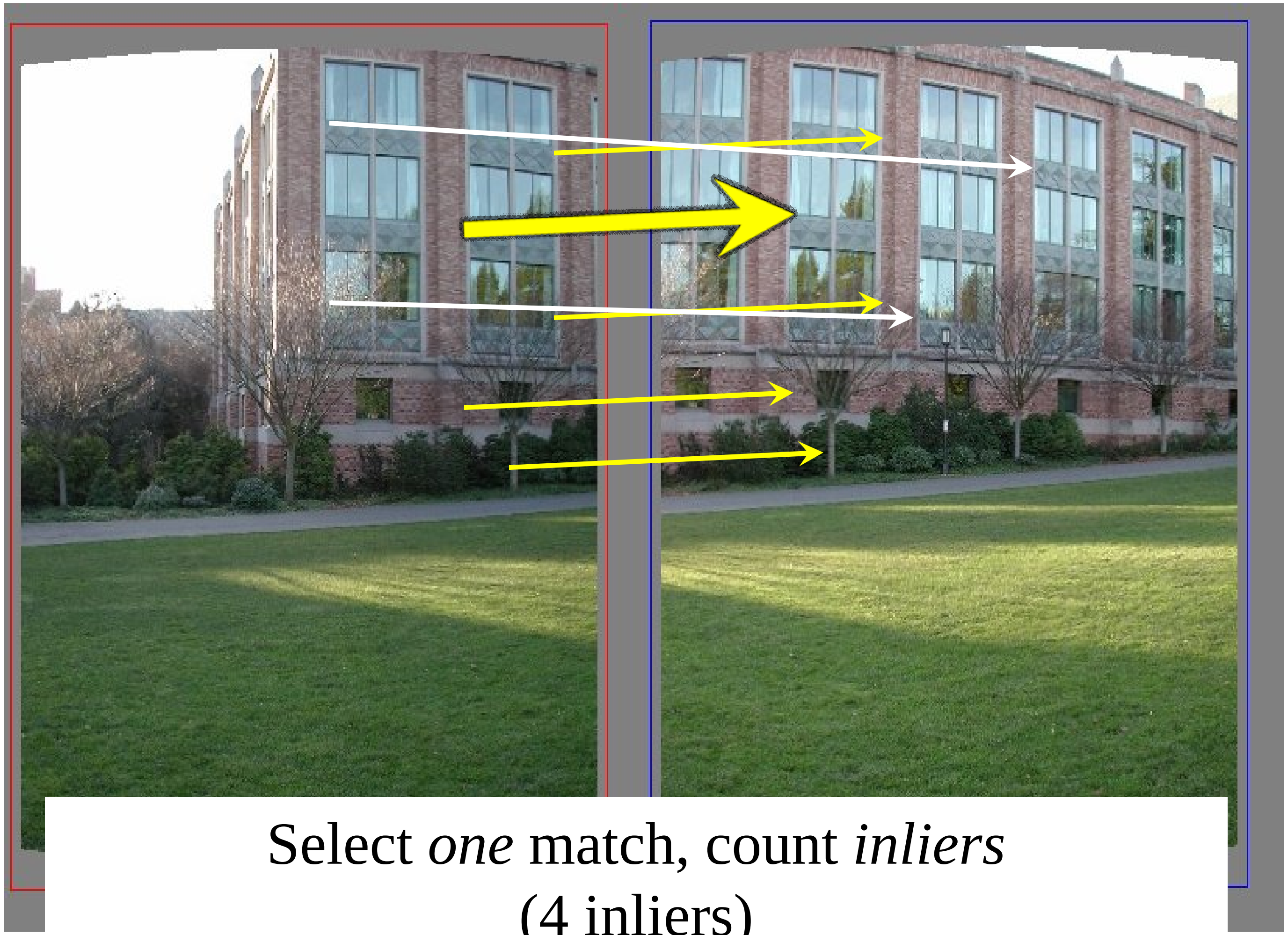
- By Matthew Brown and David Lowe
- http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

RAndom SAmple Consensus



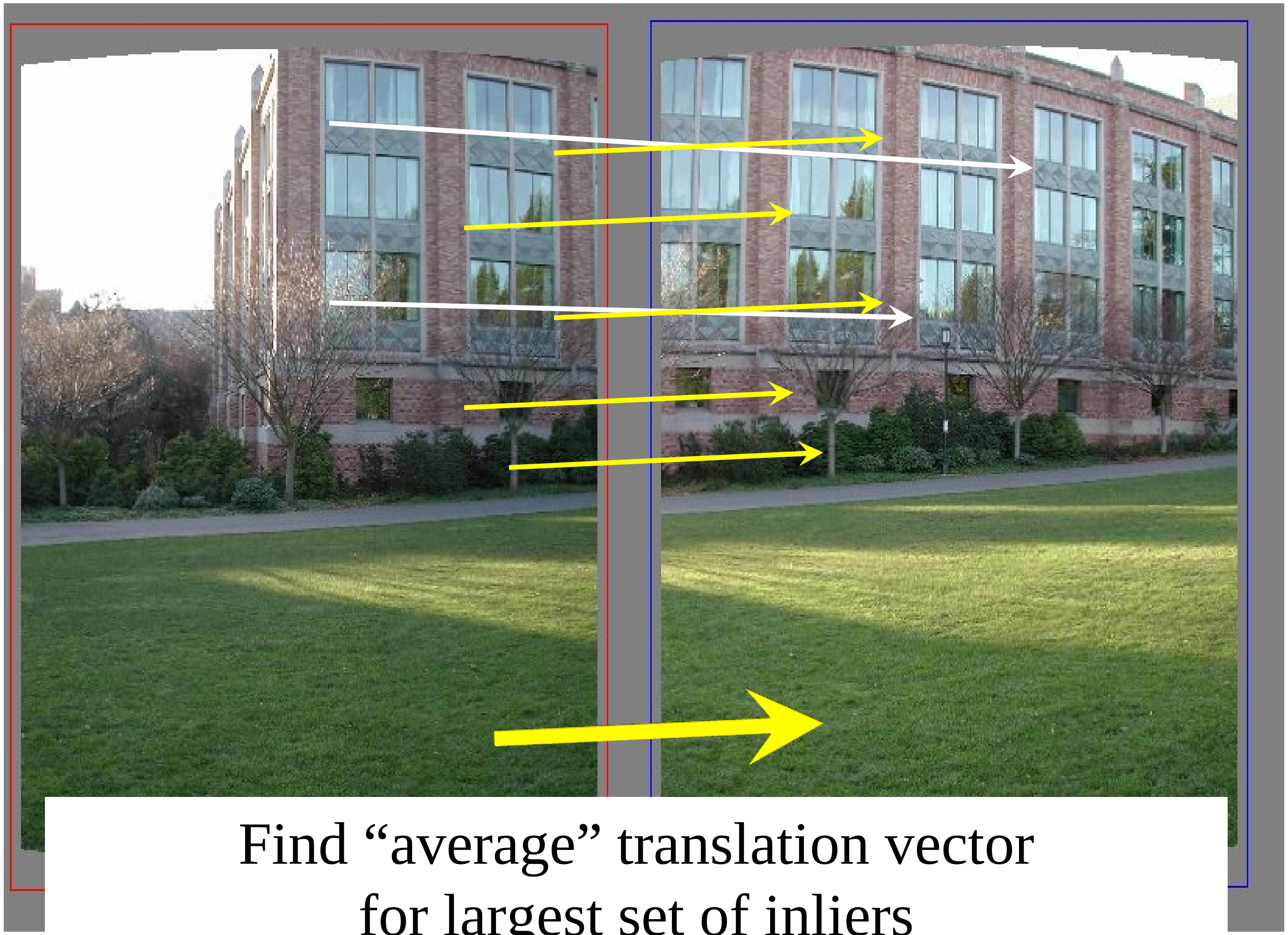
Select *one* match, count *inliers*
(in this case, only one)

RAndom SAmple Consensus



Select *one* match, count *inliers*
(4 inliers)

Least squares fit



Find "average" translation vector
for largest set of inliers

RANSAC

Same basic approach works for any transformation

- Translation, rotation, homographies, etc.
- Very useful tool

General version

- Randomly choose a set of K correspondences
  - Typically K is the minimum size that lets you fit a model
- Fit a model (e.g., homography) to those correspondences
- Count the number of inliers that “approximately” fit the model
  - Need a threshold on the error
- Repeat as many times as you can
- Choose the model that has the largest set of inliers
- Refine the model by doing a least squares fit using ALL of the inliers

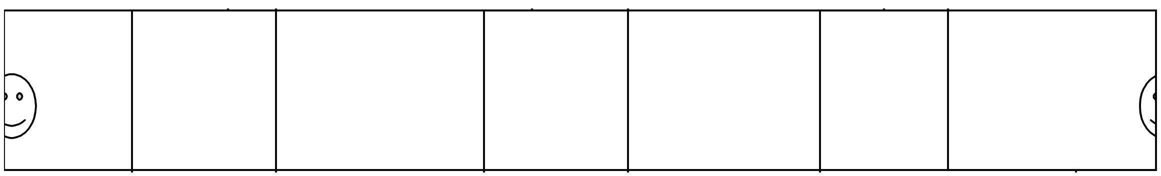
Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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**

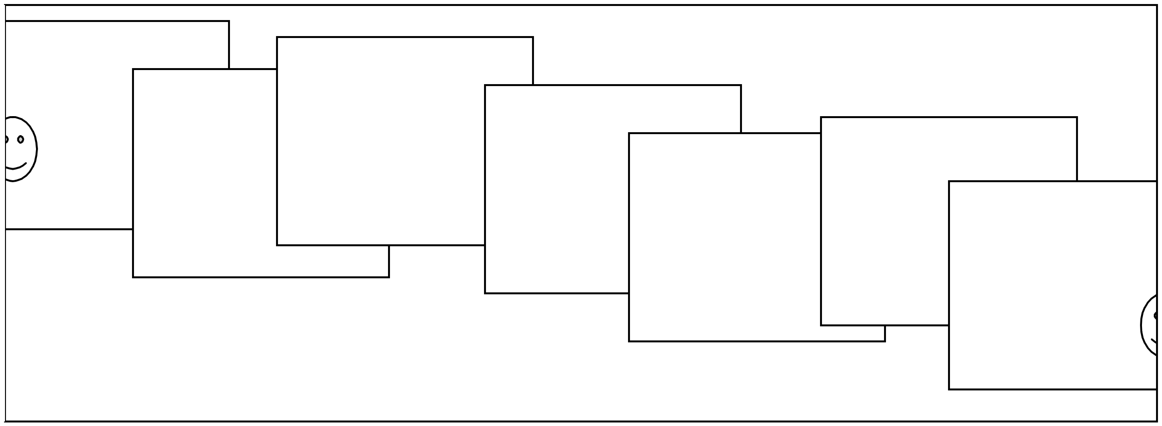
- By Matthew Brown and David Lowe
- http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Assembling the panorama



Stitch pairs together, blend, then crop

Problem: Drift



Error accumulation

- small errors accumulate over time

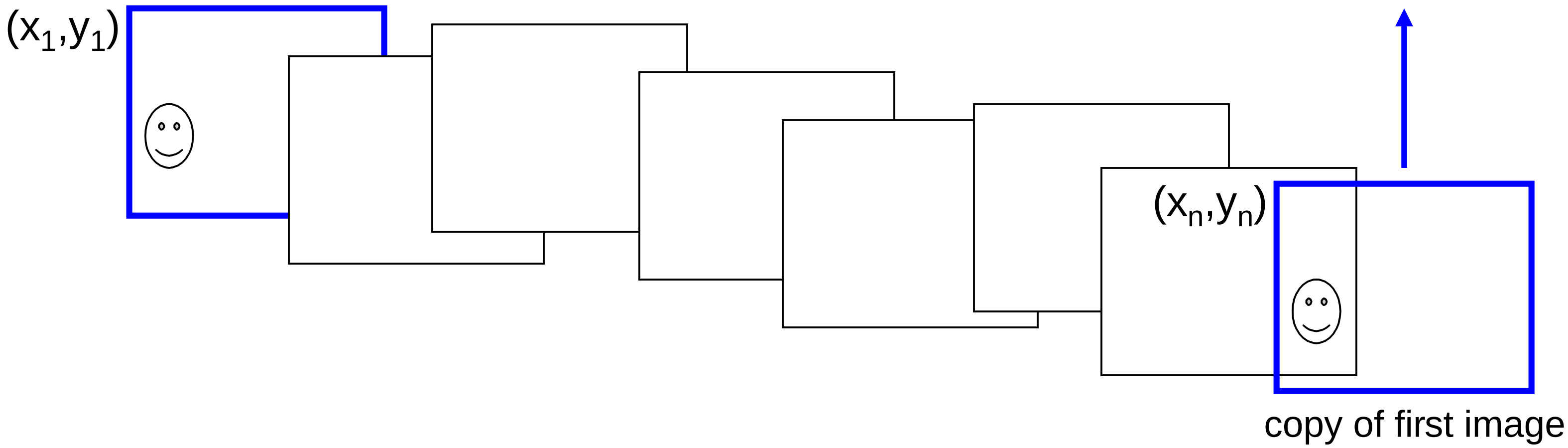
Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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

Problem: Drift



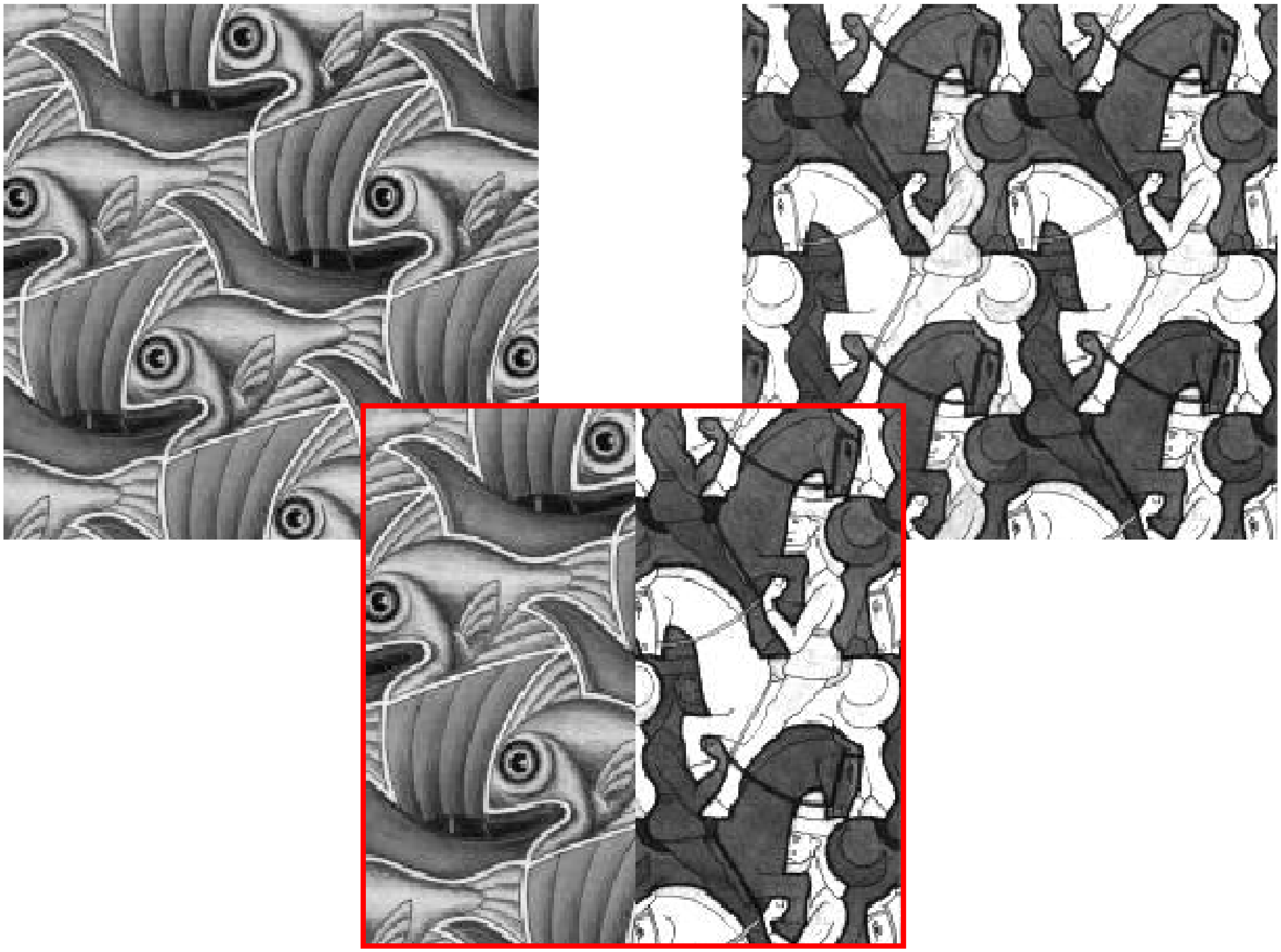
Solution

- 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
  - compute a global warp: $y' = y + ax$
  - run a big optimization problem, incorporating this constraint
    - » best solution, but more complicated
    - » known as “bundle adjustment”

Full-view Panorama

Different projections are possible

Image Blending



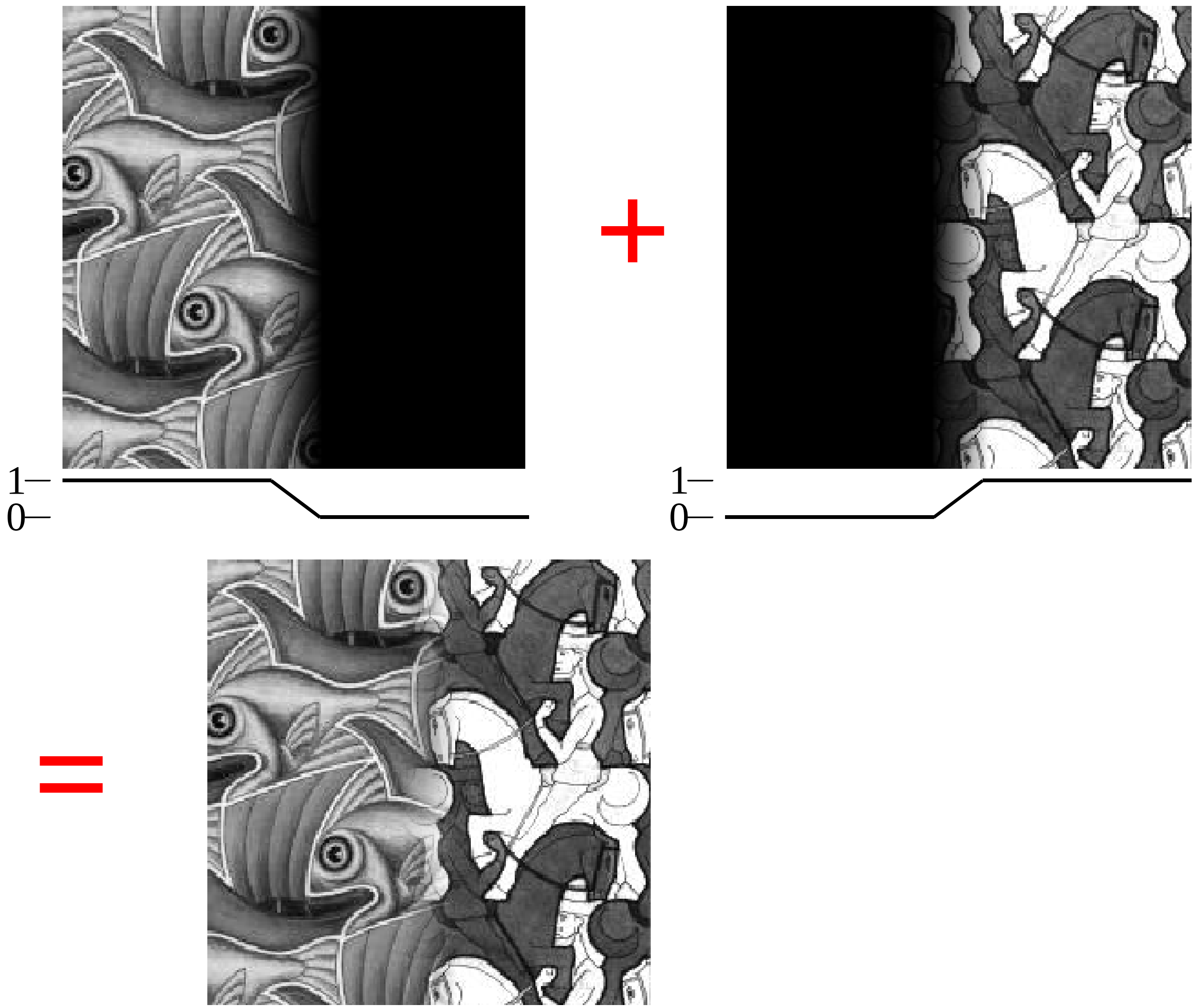
Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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

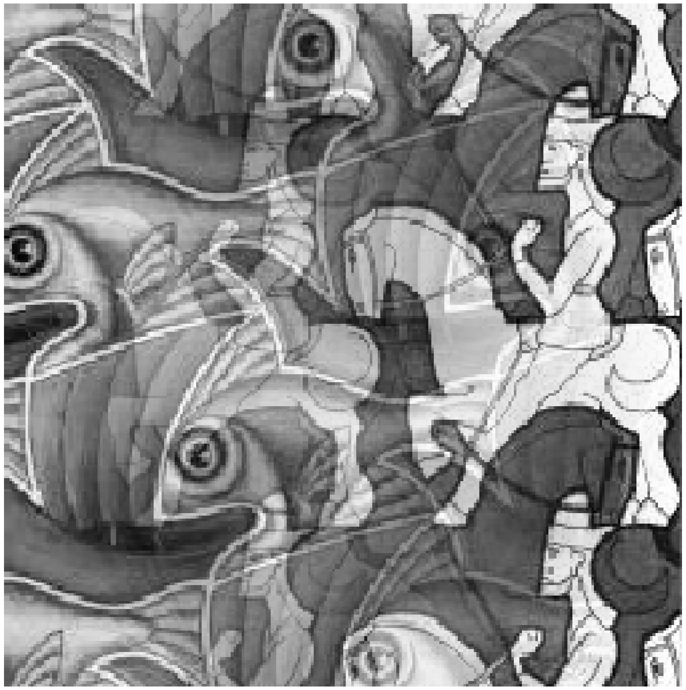
Roughly based on **Autostitch**

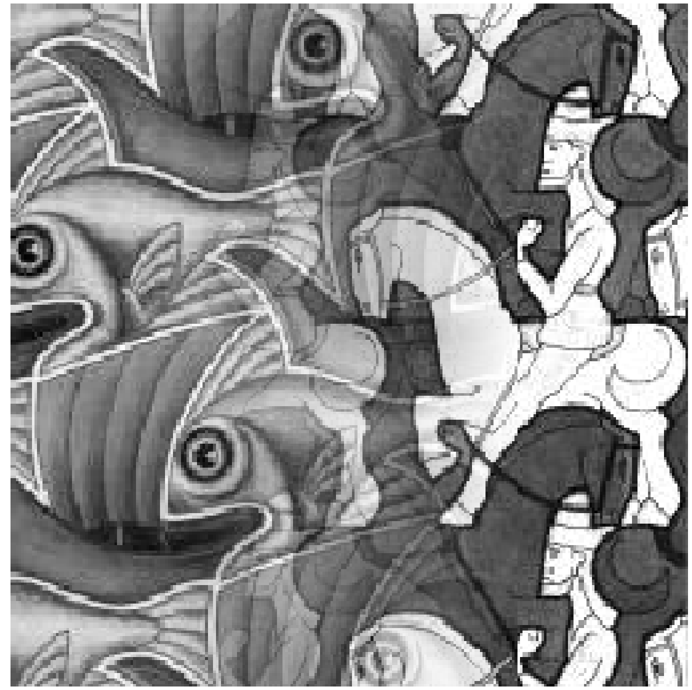
- By Matthew Brown and David Lowe
- http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

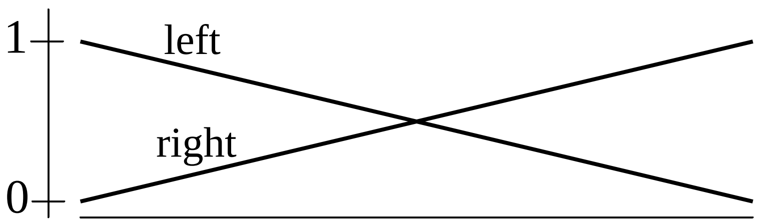
Feathering

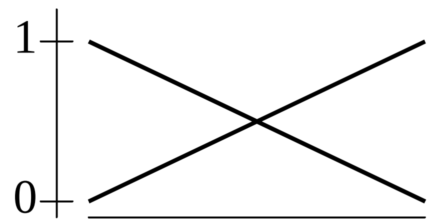


Effect of window size

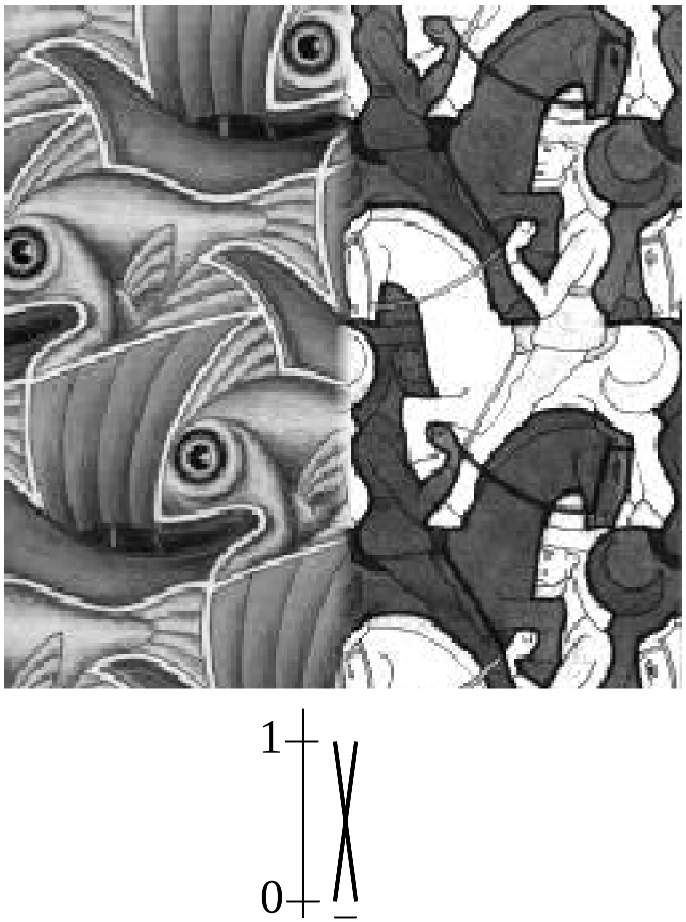


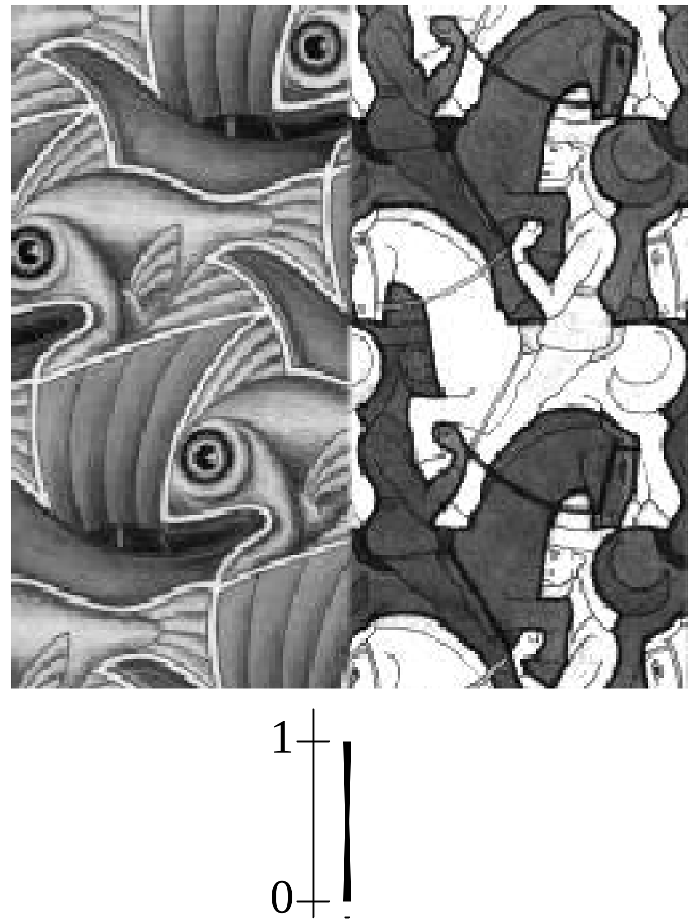




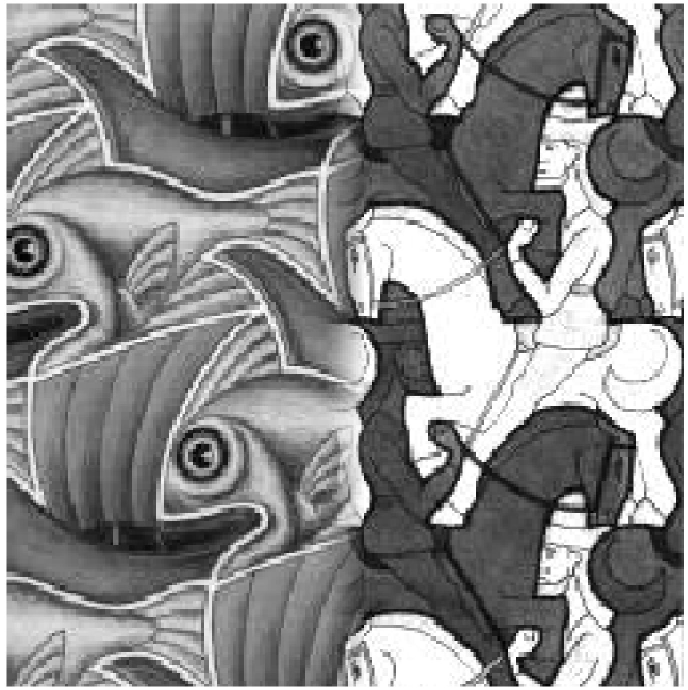


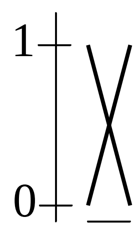
Effect of window size





Good window size

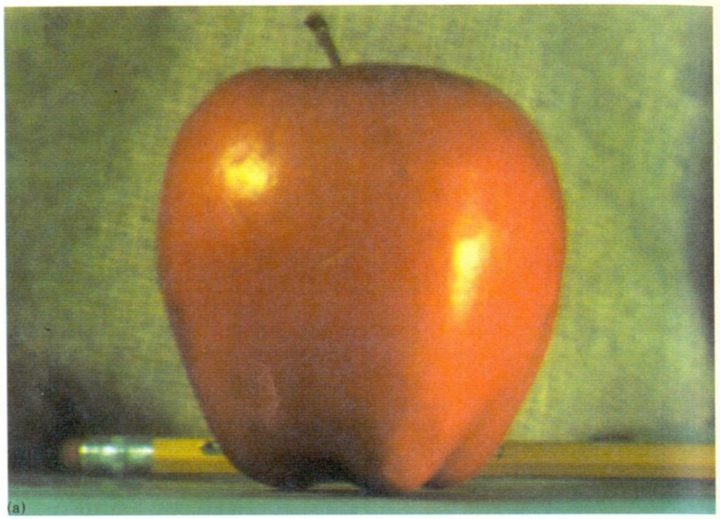


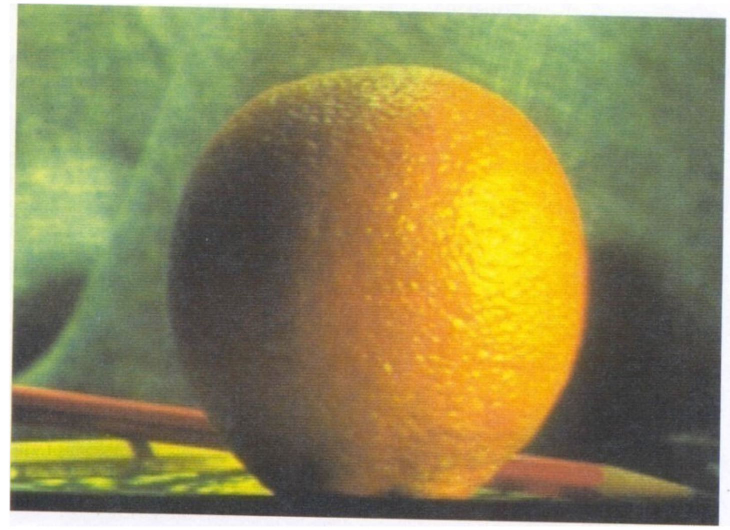


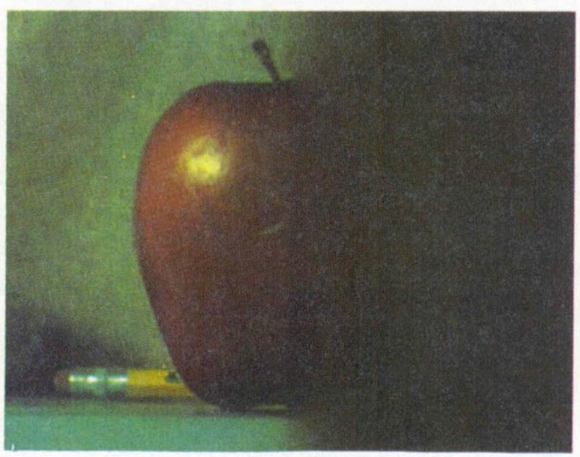
"Optimal" window: smooth but not ghosted

- Doesn't always work...

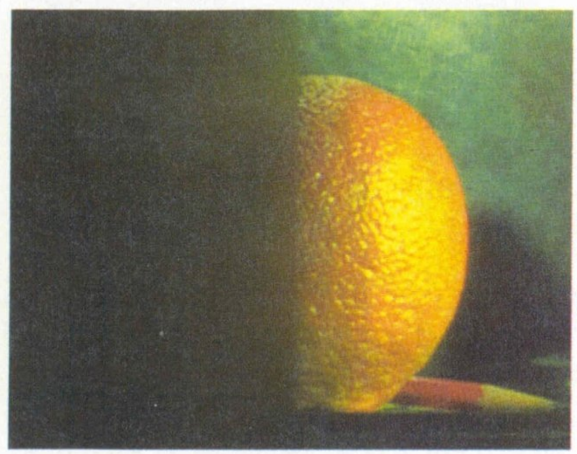
Pyramid blending



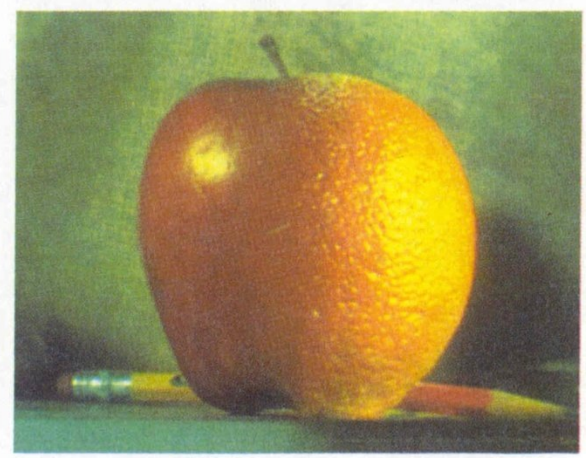




(d)



(h)

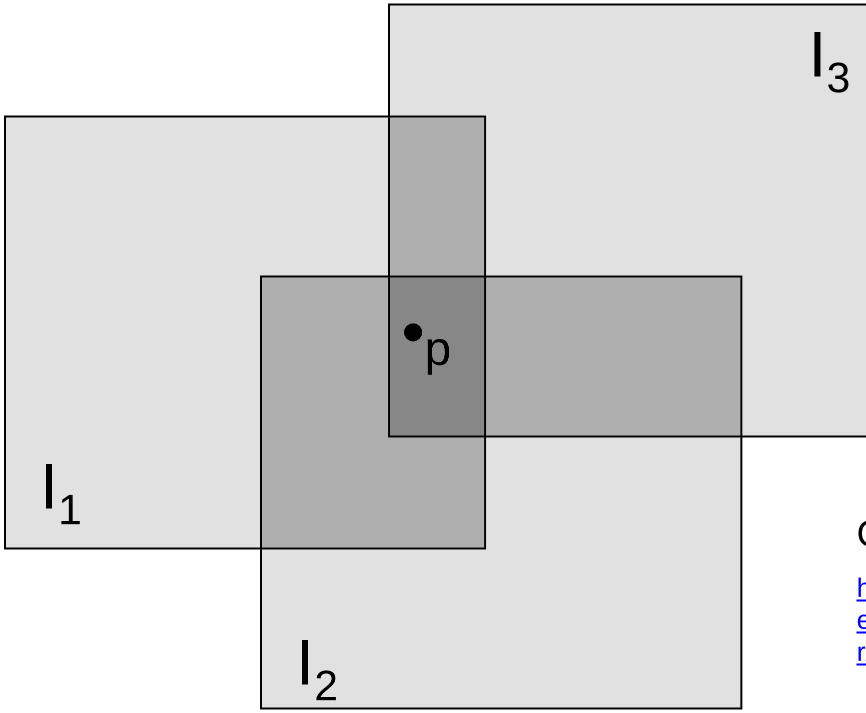


(l)

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.

Alpha Blending



Optional: see Blinn (CGA, 1994) for details:

http://ieeexplore.ieee.org/iel1/38/7531/00310740.pdf?isNumber=7531&prod=JNL&arnumber=310740&arSt=83&ared=87&arAuthor=Blinn%2C+J.F.

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

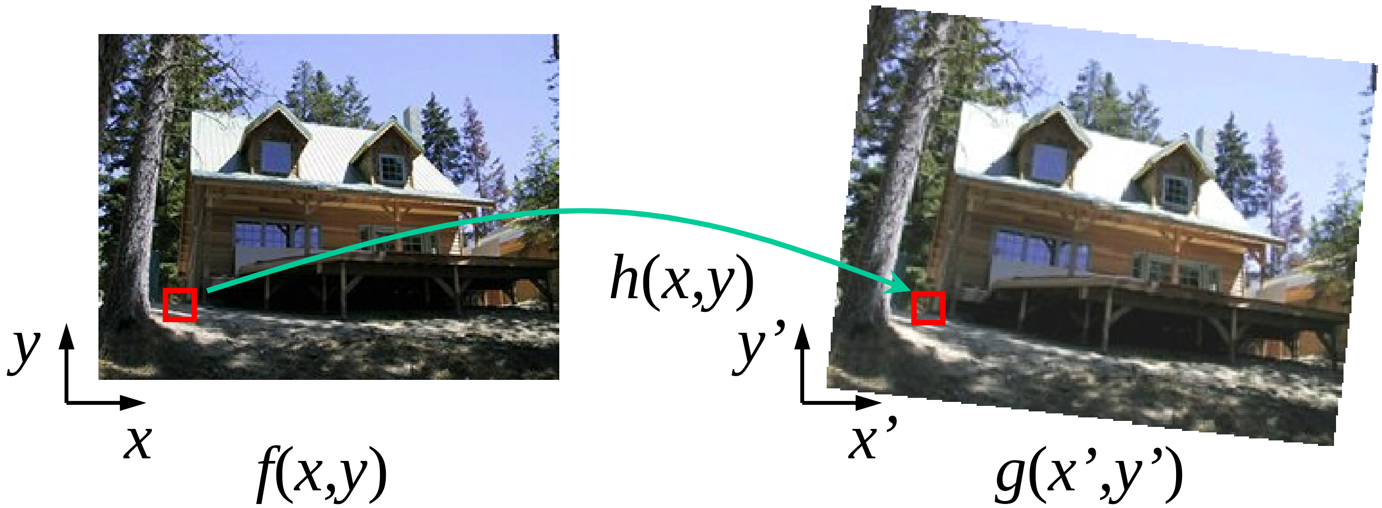
   Q: what if $\alpha = 0$?

Poisson Image Editing

For more info: Perez et al, SIGGRAPH 2003

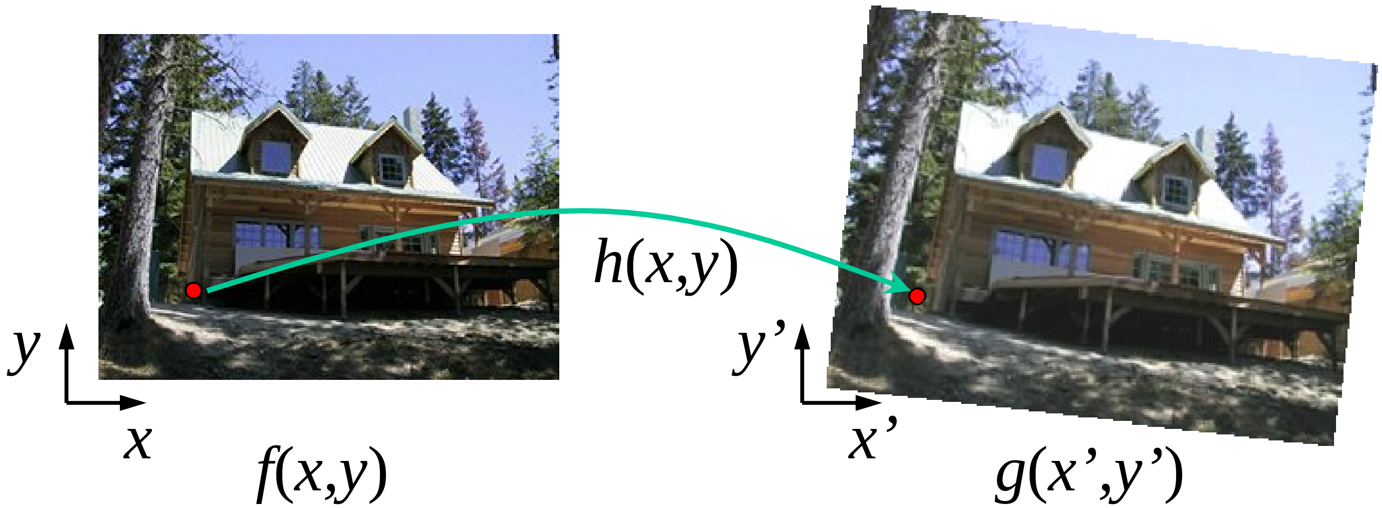
- http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf

Image warping



Given a coordinate transform *(x',y')* = *h*(*x,y*) and a source image *f*(*x,y*), how do we compute a transformed image *g(x',y')* = *f*(*h*(*x,y*))?

Forward warping

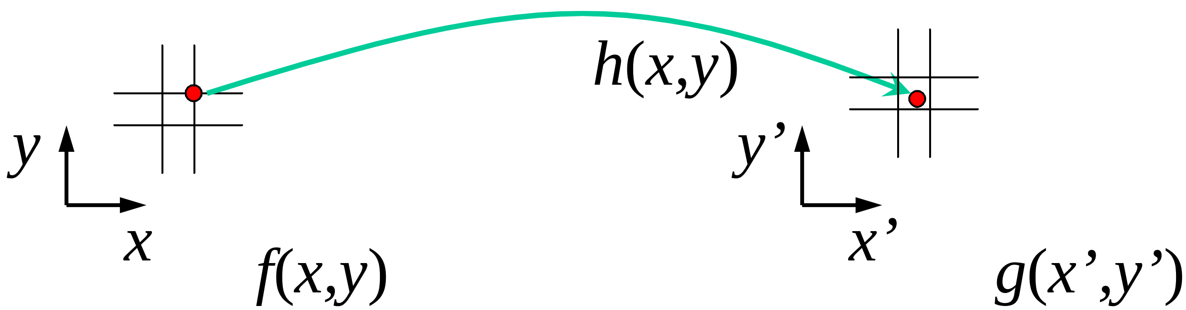


Send each pixel *f*(*x*,*y*) to its corresponding location

(*x'*,*y'*) = *h*(*x*,*y*) in the second image

Q: what if pixel lands "between" two pixels?

Forward warping



Send each pixel *f*(*x,y*) to its corresponding location

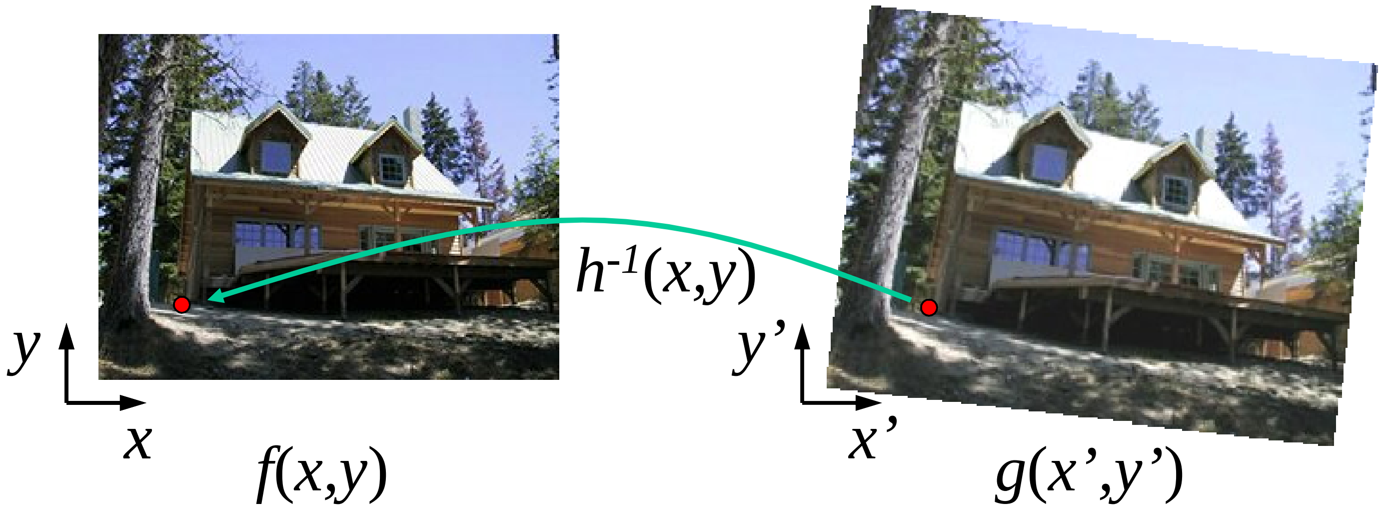
(*x',y')* = *h*(*x,y*) in the second image

Q: what if pixel lands "between" two pixels?

A: distribute color among neighboring pixels (x',y')

- Known as "splatting"

Inverse warping

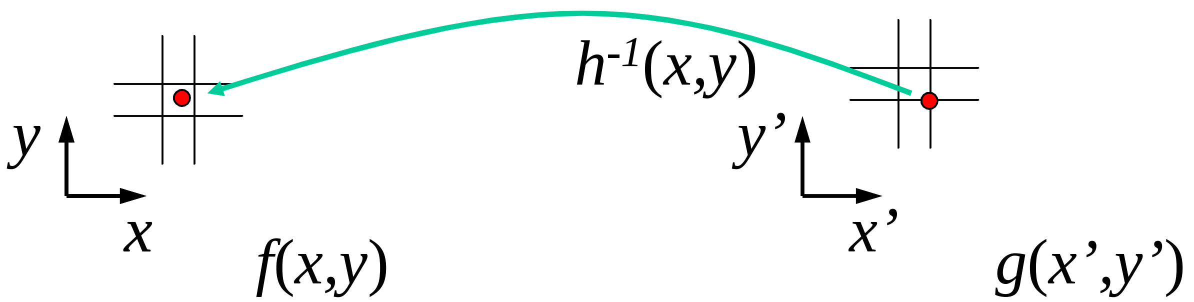


Get each pixel $g(x',y')$ from its corresponding location

$(x,y) = h^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

Inverse warping



Get each pixel $g(x',y')$ from its corresponding location

$(x,y) = h^{-1}(x',y')$ in the first image

Q: what if pixel comes from "between" two pixels?

A: *resample* color value

- We discussed resampling techniques before
  - nearest neighbor, bilinear, Gaussian, bicubic

Forward vs. inverse warping

Q: which is better?

A: usually inverse—eliminates holes

- however, it requires an invertible warp function—not always possible...

Project 2

1. Take pictures on a tripod (or handheld)
2. Warp to spherical coordinates
3. Extract features
4. Align neighboring pairs using 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

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
  - http://earthobservatory.nasa.gov/Newsroom/BlueMarble/
  - Click for images…