CSCE 448/748 – Computational Photography

Homographies and Mosaics

Nima Kalantari

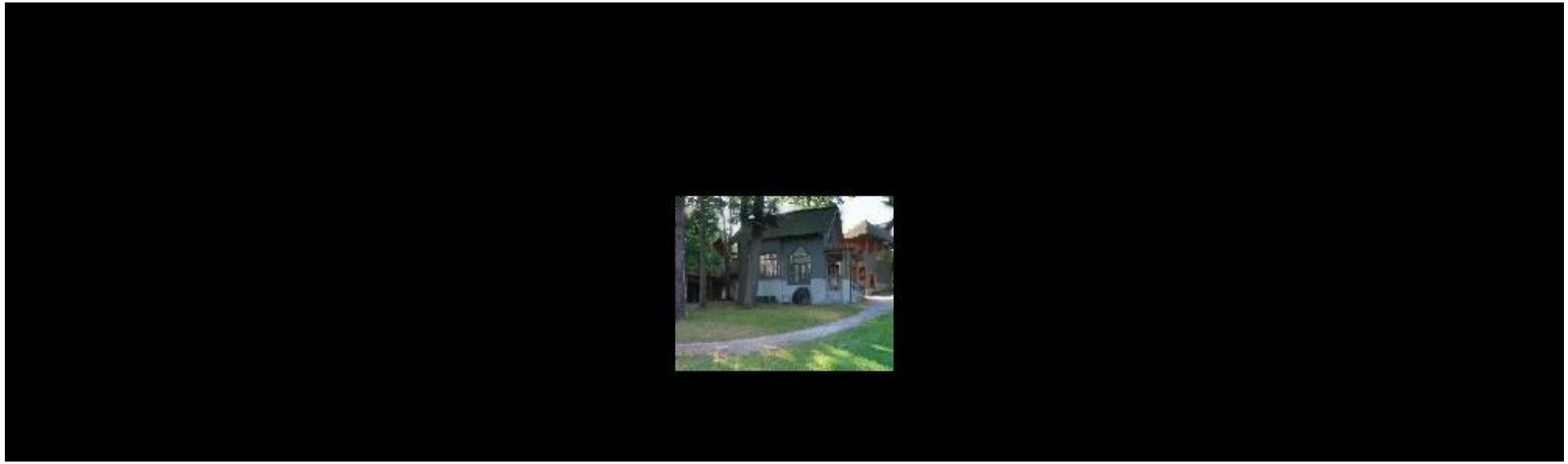
Many slides from Alexei A. Efros, Richard Szeliski, and Steve Seitz

Mosaics: stitching images together

virtual wide-angle camera

Why Mosaic?

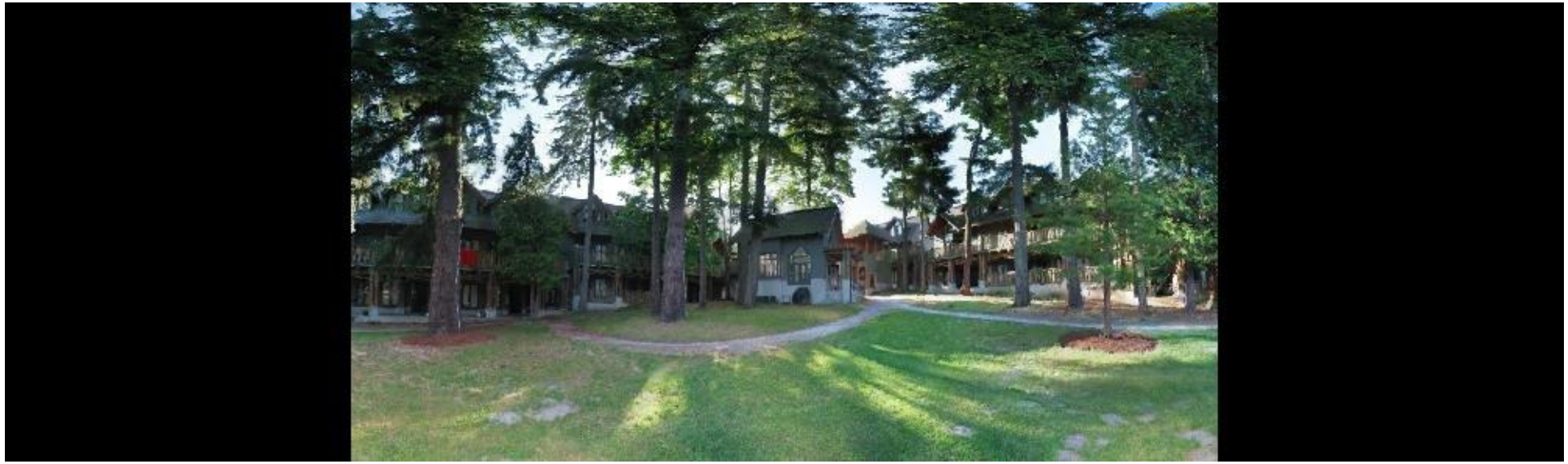
- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°



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?

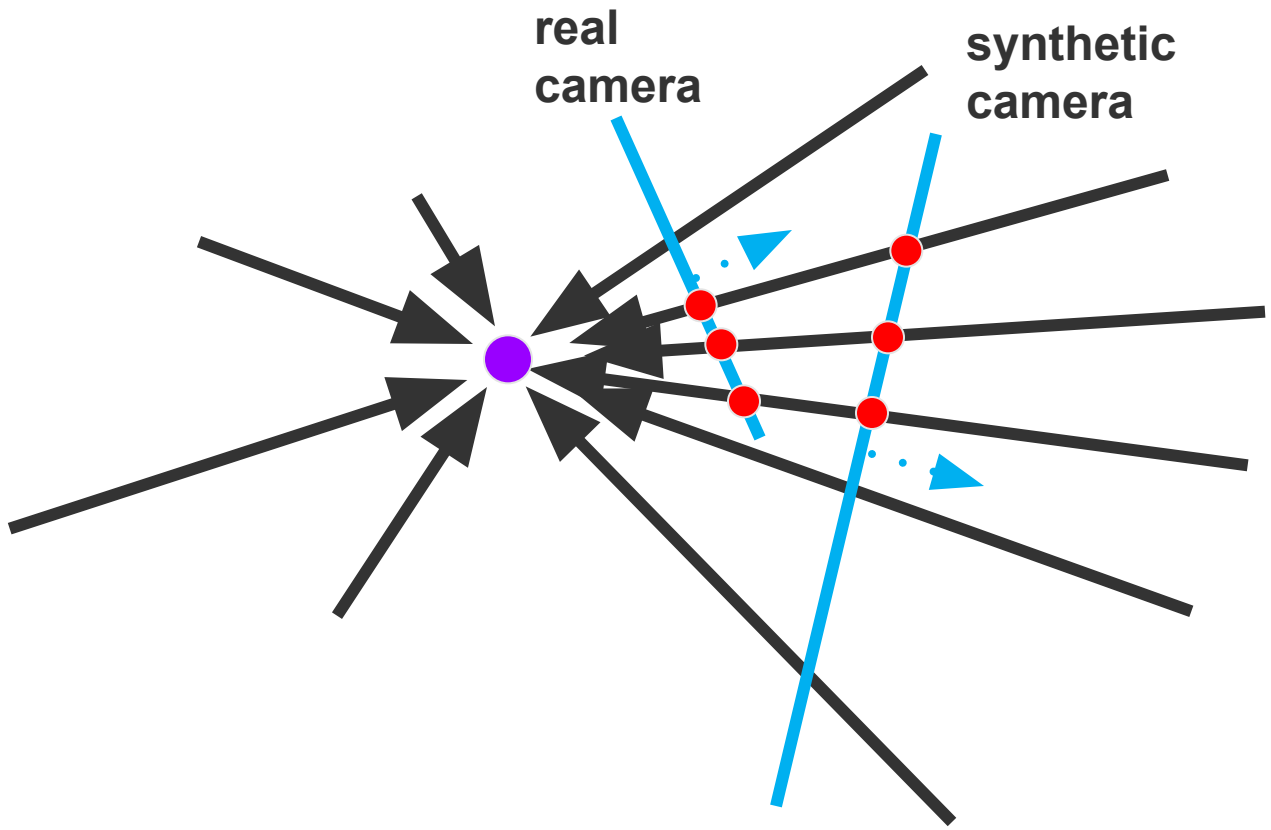
- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Slide from Brown & Lowe

How to do it?

- 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

A pencil of rays contains all views



Can generate any synthetic camera view
as long as it has the same center of projection!

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
  - Mosaic is a *synthetic wide-angle camera*

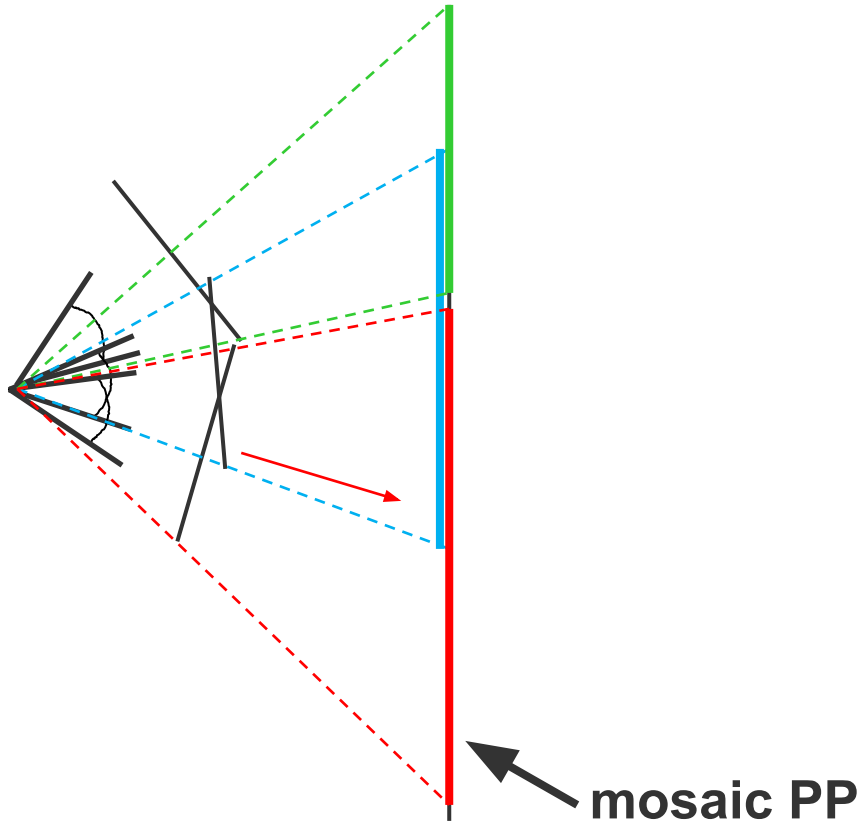


Image reprojection

Basic question

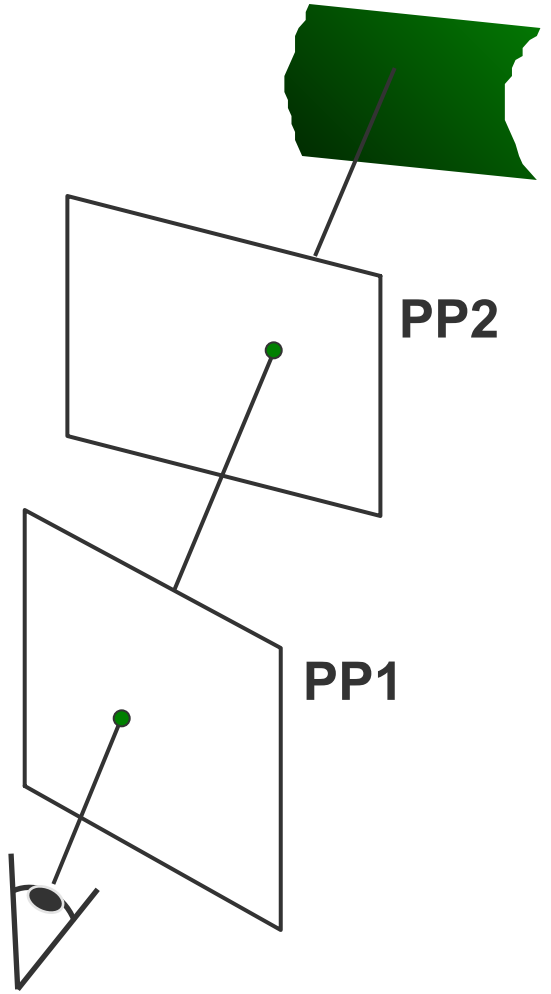
- How to relate two images from the same camera center?
  - how to map a pixel from PP1 to PP2

Answer

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

But don't we need to know the geometry of the two planes with respect to the eye?

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



Aligning images

left on top

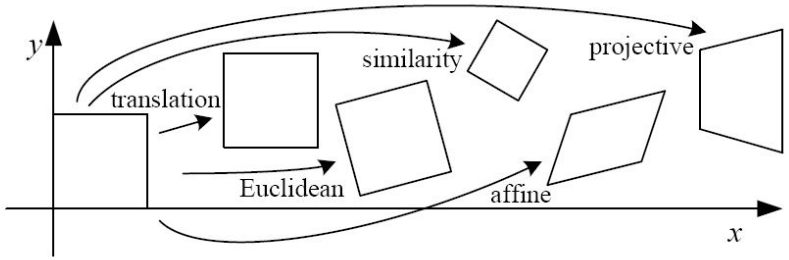
right on top

Translations are not enough to align the images

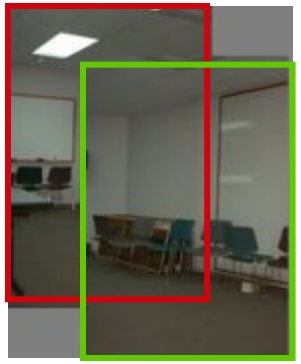
Back to Image Warping

Which t-form is the right one for warping PP1 into PP2?

e.g. translation, Euclidean, affine, projective

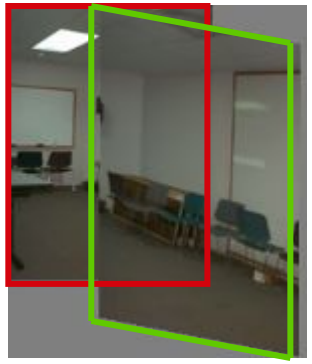


Translation



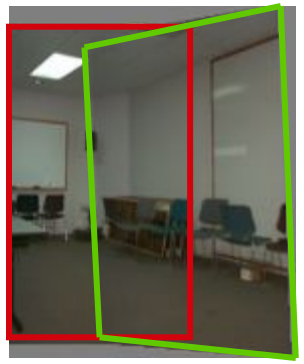
2 unknowns

Affine



6 unknowns

Perspective



8 unknowns

Homography

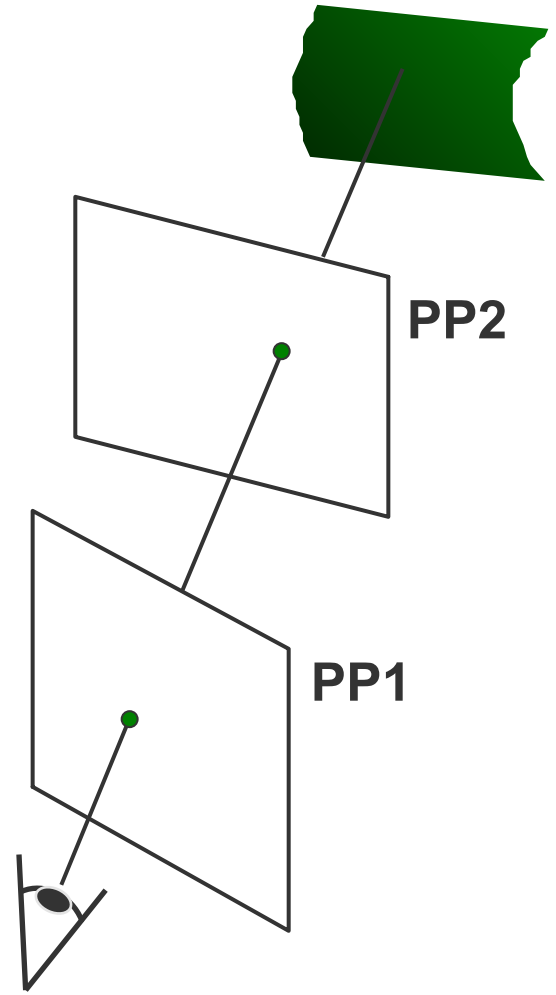
- Projective – mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines aren't
  - but must preserve straight lines
- called Homography

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

P'' H p

To apply a homography H

- Compute p'' = Hp (regular matrix multiply)
- Convert p'' from homogeneous to image coordinates



Mosaics: main steps

- Collect correspondences (manually)
- Solve for homography matrix H

- Warp content from one image frame to the other to combine: say im1 into im2 reference frame

- Overlay im2 content onto the warped im1 content.

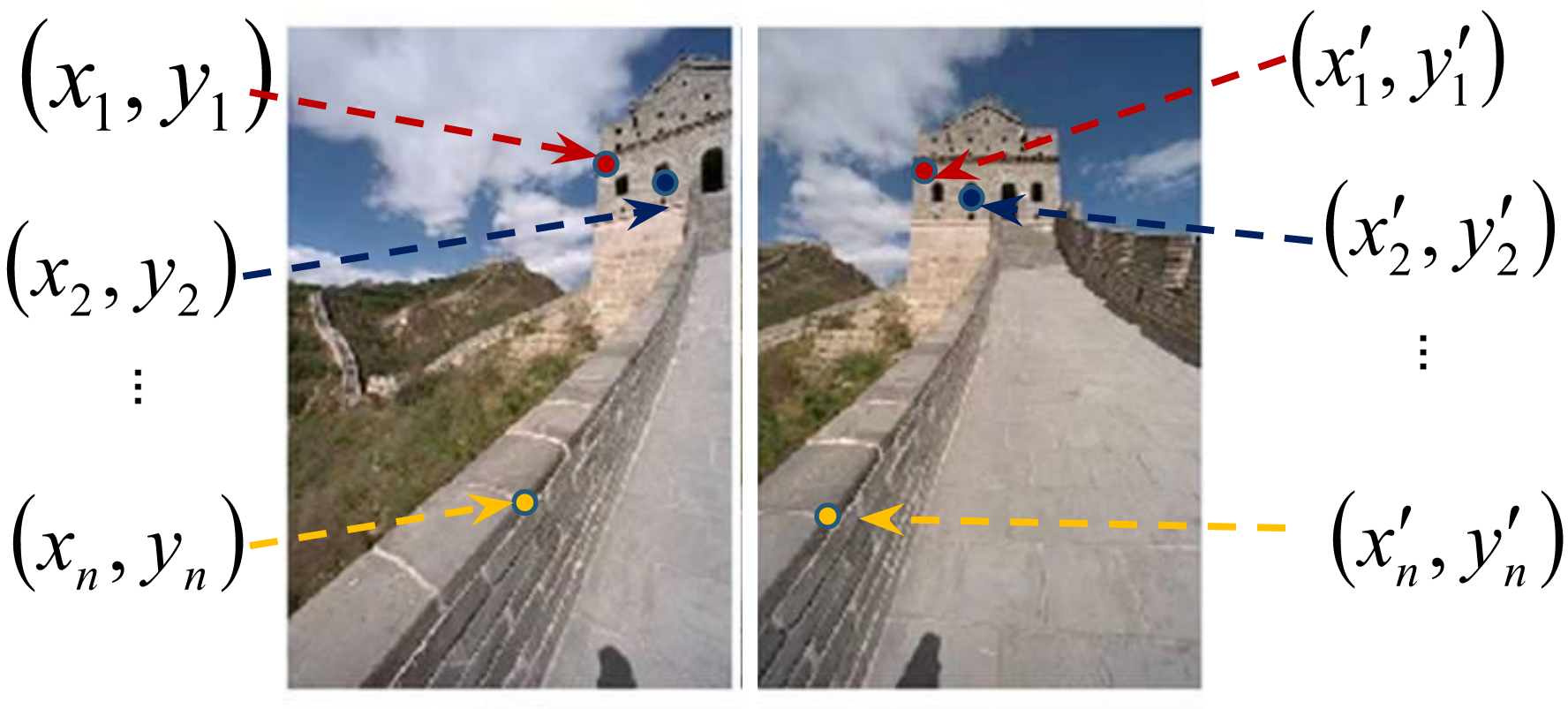
Mosaics: main steps

- **Collect correspondences (manually)**
- Solve for homography matrix H
  - Least squares solution
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1.
  - Compute coordinates in im1's reference frame (via homography) for all points in that range
  - Lookup all colors for all these positions from im1
    - Inverse warp
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny

Mosaics: main steps

- Collect correspondences (manually)
- **Solve for homography matrix H**
  - **Least squares solution**
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1.
  - Compute coordinates in im1's reference frame (via homography) for all points in that range
  - Lookup all colors for all these positions from im1
    - Inverse warp
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny

Homography



To compute the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of H are the unknowns…

Solving for homographies

$$P'' = Hp$$

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

- Can set scale factor *i*=1. So, there are 8 unkowns.
- Set up a system of linear equations:

$$Ah = b$$

- where vector of unknowns h = [a,b,c,d,e,f,g,h]^T
- Need at least 8 eqs, but the more the better…
- Solve for h. If overconstrained, solve using least-squares:

$$\min \|Ah - b\|^2$$

Example system for finding homography

destination coordinates of the four corners of a quadrilateral. Let the correspondence map $(u_k, v_k)^t$ to $(x_k, y_k)^T$ for vertices numbered cyclically $k = 0, 1, 2, 3$. All coordinates are assumed to be real (finite). To compute the forward mapping matrix $\mathbf{M}_{sd}$, assuming that $i = 1$, we have eight equations in the eight unknowns a-h:

$$x_k = \frac{au_k + bv_k + c}{gu_k + hv_k + 1} \quad \Rightarrow \quad u_k a + v_k b + c - u_k x_k g - v_k x_k h = x_k$$

$$y_k = \frac{du_k + ev_k + f}{gu_k + hv_k + 1} \quad \Rightarrow \quad u_k d + v_k e + f - u_k y_k g - v_k y_k h = y_k$$

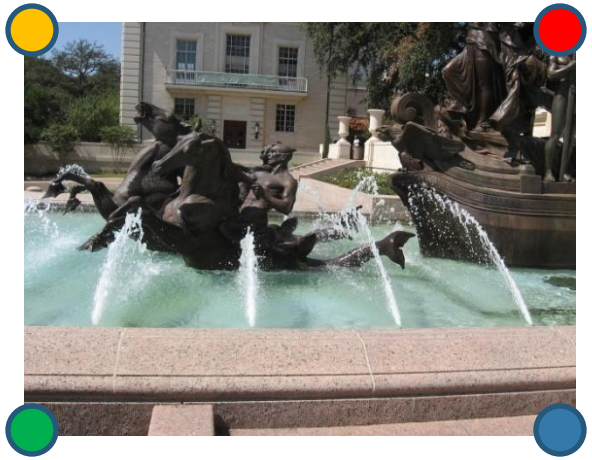
for $k = 0, 1, 2, 3$. This can be rewritten as an 8×8 system:

$$\begin{pmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

Source: Paul Heckbert, ***Fundamentals of Texture Mapping and Image Warping***

Mosaics: main steps

- Collect correspondences (manually)
- Solve for homography matrix H
  - Least squares solution
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny

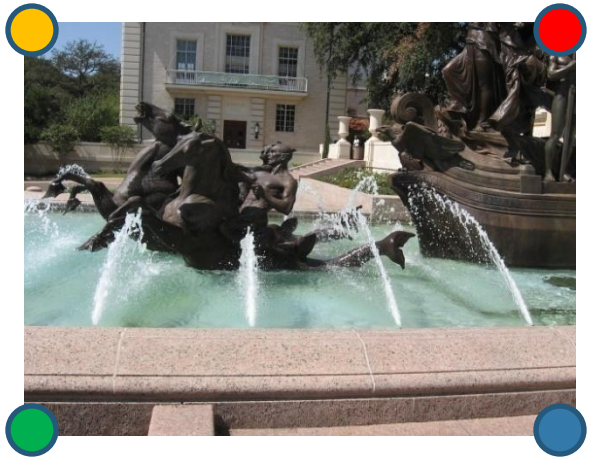


im1

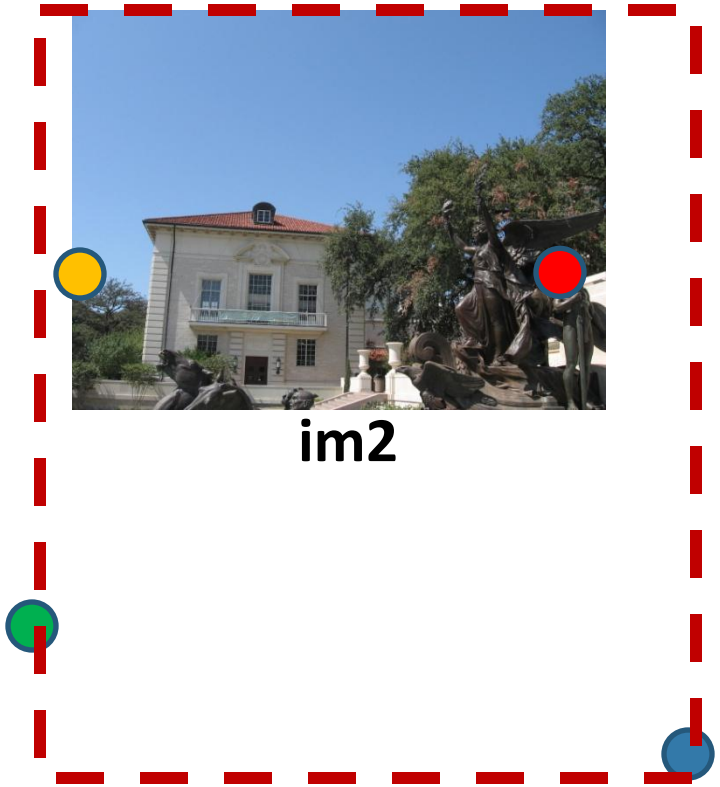
im2

Mosaics: main steps

- Collect correspondences (manually)
- Solve for homography matrix H
  - Least squares solution
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny



im1



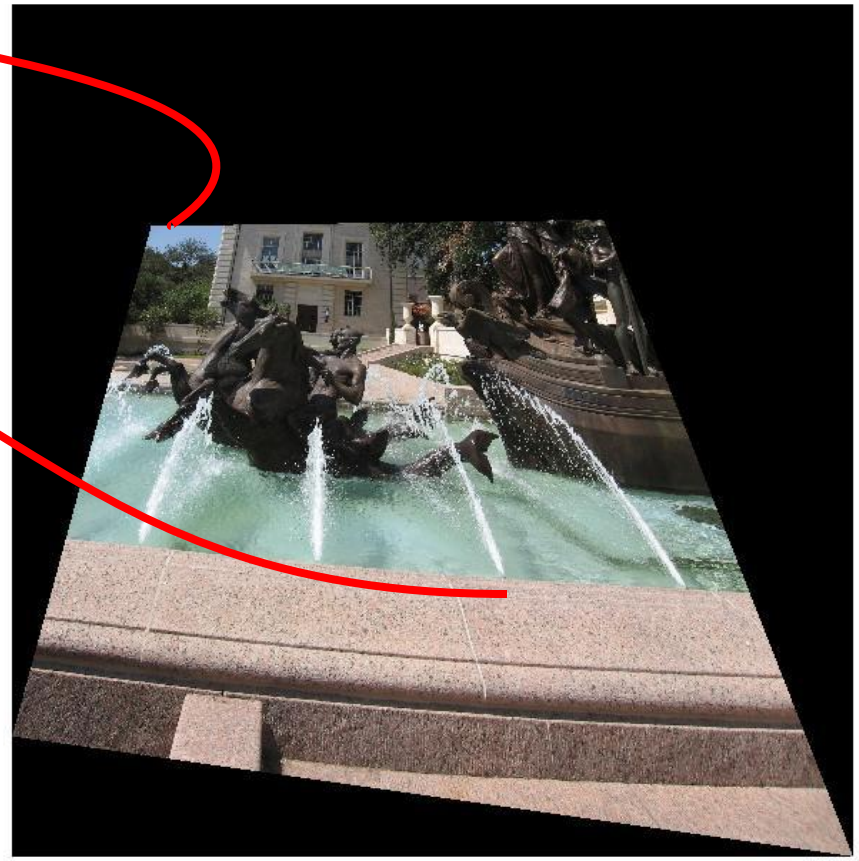
im2

Mosaics: main steps

- Collect correspondences (manually)
- Solve for homography matrix H
  - Least squares solution
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny

im1

im2

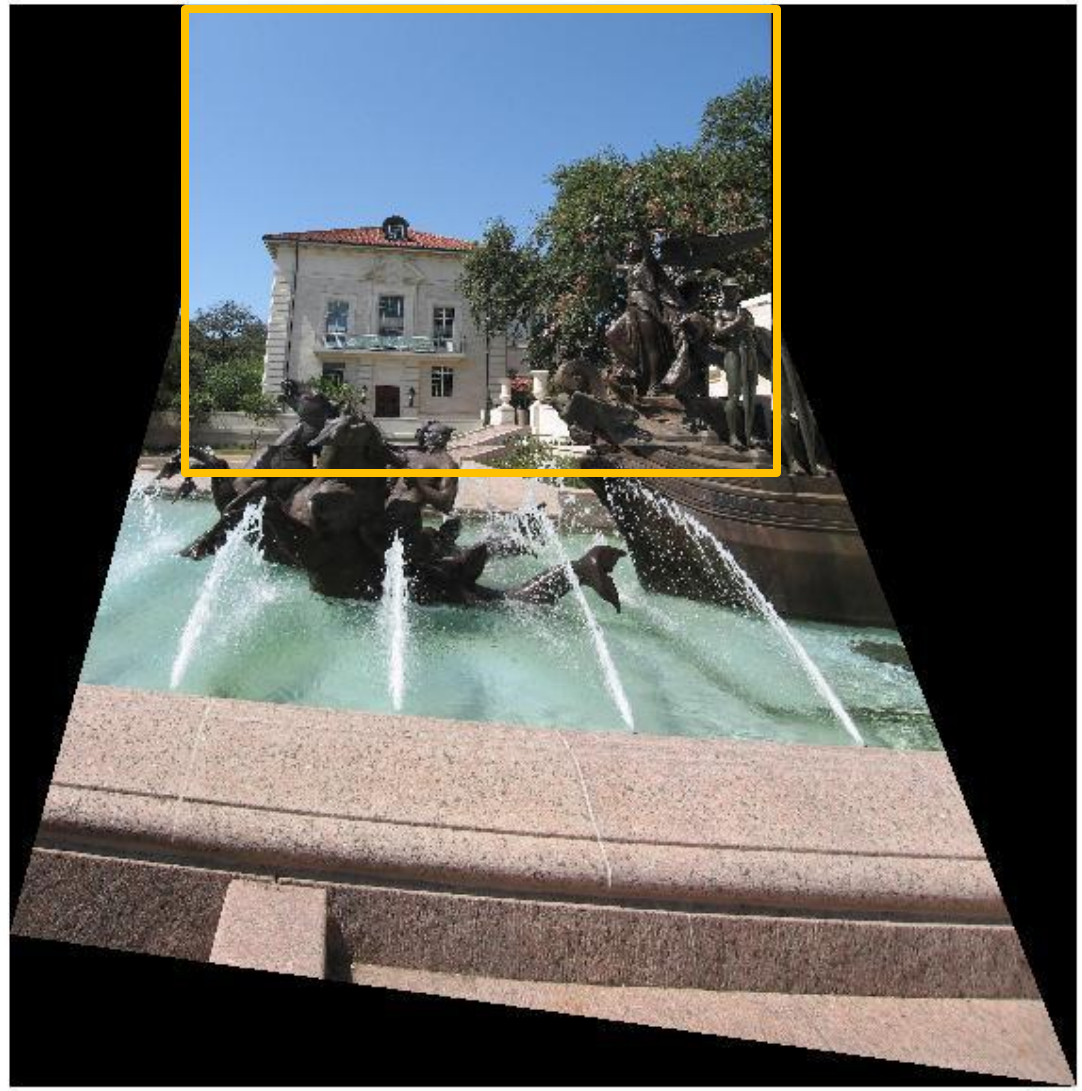


im1 warped into reference frame of im2.

Use interp2 to ask for the colors (possibly interpolated) from im1 at all the positions needed in im2's reference frame.

Mosaics: main steps

- Collect correspondences (manually)
- Solve for homography matrix H
  - Least squares solution
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2's coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1's reference frame (via homography) for all points in that range: H^{-1}
  - Lookup all colors for all these positions from im1
    - Inverse warp
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny



Mosaics: stitching images together

virtual wide-angle camera