SAIR

Spatial AI & Robotics Lab

CSE 473/573-A

L11: ALIGNMENT & FITTING

Chen Wang

Spatial AI & Robotics Lab

Department of Computer Science and Engineering

University at Buffalo The State University of New York

Content

- Alignment
  - Homography
  - Interpolation
  - Stitching, Panorama

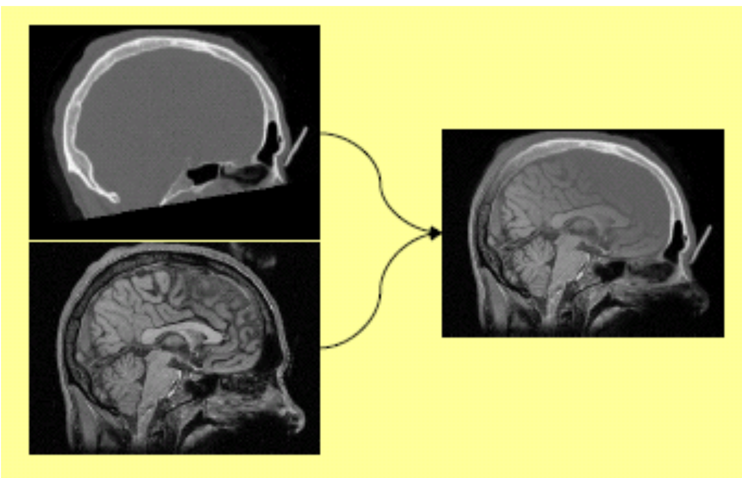
What are Alignment and Fitting?

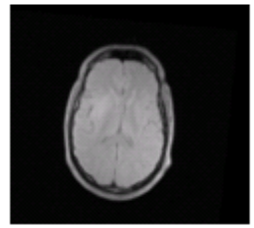
- Alignment
  - Find the parameters of a transformation that best aligns matched points
- Fitting
  - Find the parameters of a model that best fit the data

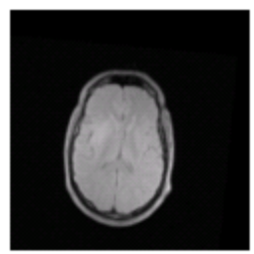
Fitting and Alignment: Methods

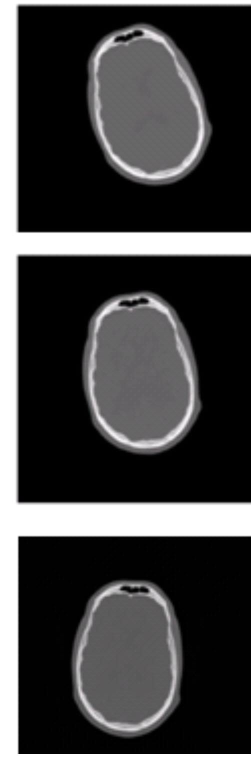
- Hypothesize and test
  - Generalized Hough transform
- General Alignment
  - **Homographies**
  - **Rotational Panoramas**
  - Global Alignment
  - RANSAC
  - Warping
  - Blending
- Global optimization / Search for parameters
  - Least squares fit
  - Robust least squares
  - Other parameter search methods

Motivation: Medical image registration









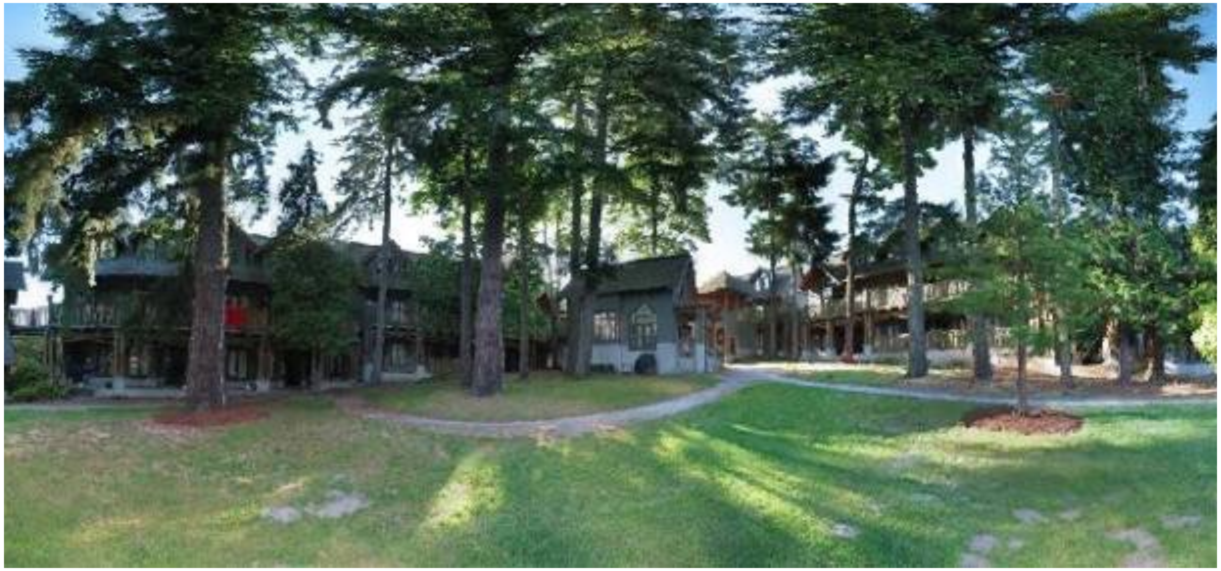
Motivation: Mosaics

- Getting the whole picture
  - Typical camera: 50° x 35°

Brown & Lowe 2003

Motivation: Mosaics

- Getting the whole picture
  - Typical camera: 50° x 35°
  - Human Vision: 176° x 135°



Brown & Lowe 2003

Motivation: Mosaics

- Getting the whole picture
  - Typical camera: 50° x 35°
  - Human Vision: 176° x 135°

Brown & Lowe 2003

Alignment

- Homography
- Rotational Panoramas
- RANSAC (Next Lecture)
- Global alignment
- Warping
- Blending

(a) (b)

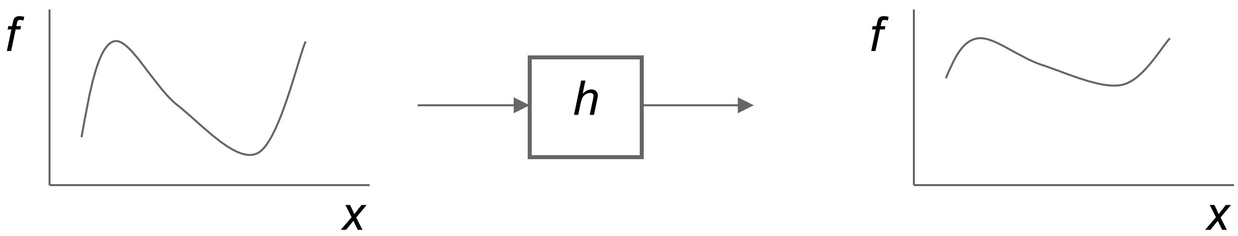
Motion models

- What happens when we take two images with a camera and try to align them?
- translation?
- rotation?
- scale?
- affine?
- perspective?

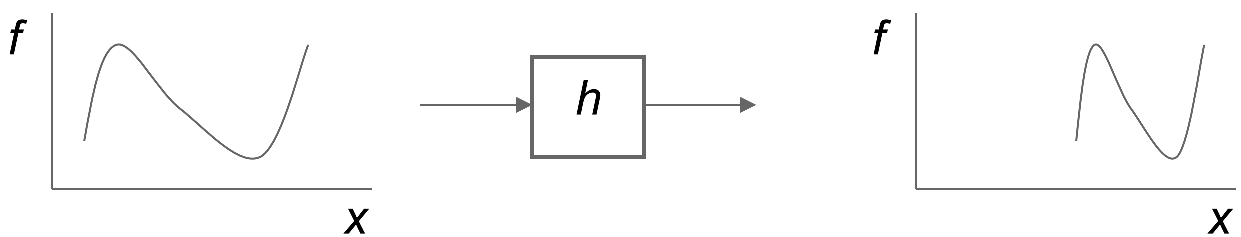
Szeliski

Image Warping (Recap)

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



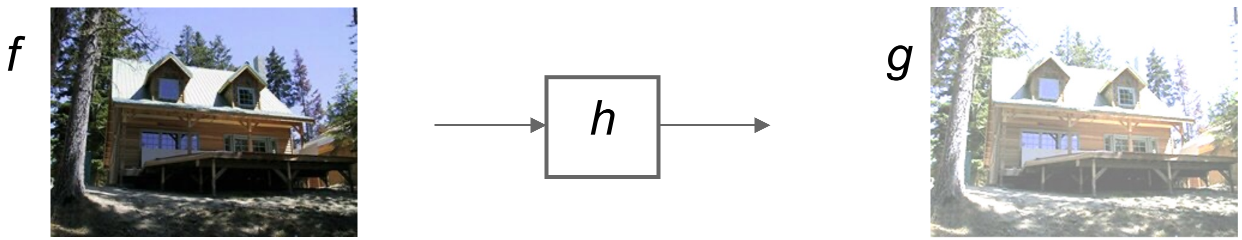
- image warping: change *domain* of image
  - $g(x) = f(h(x))$



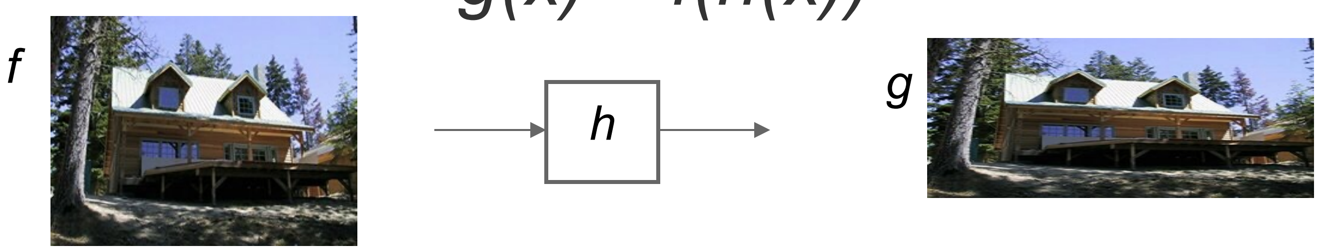
Szeliski

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

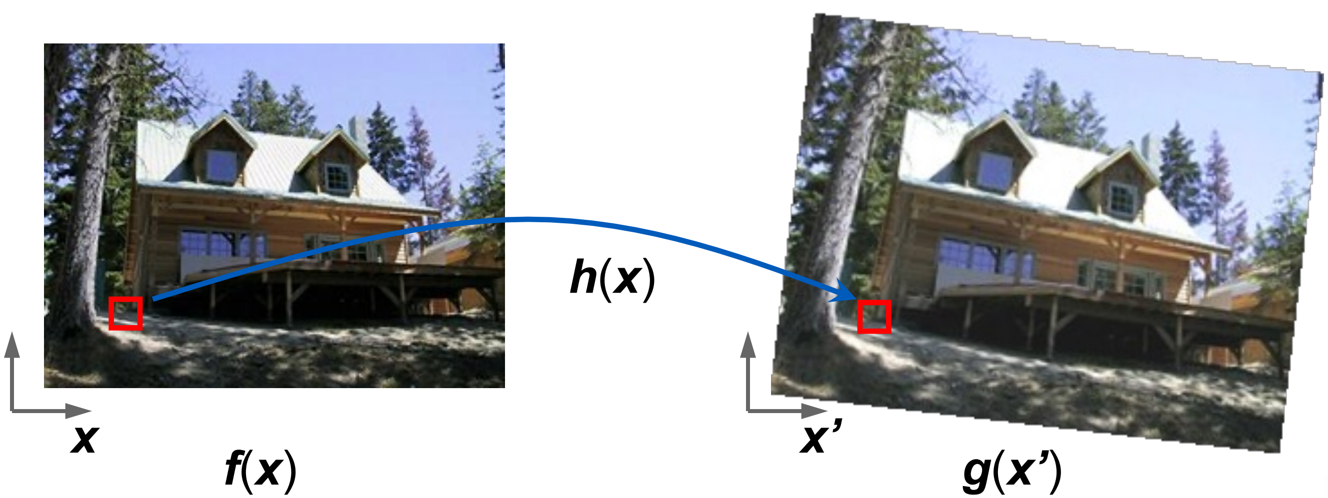
perspective

cylindrical

Szeliski

Image Warping

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

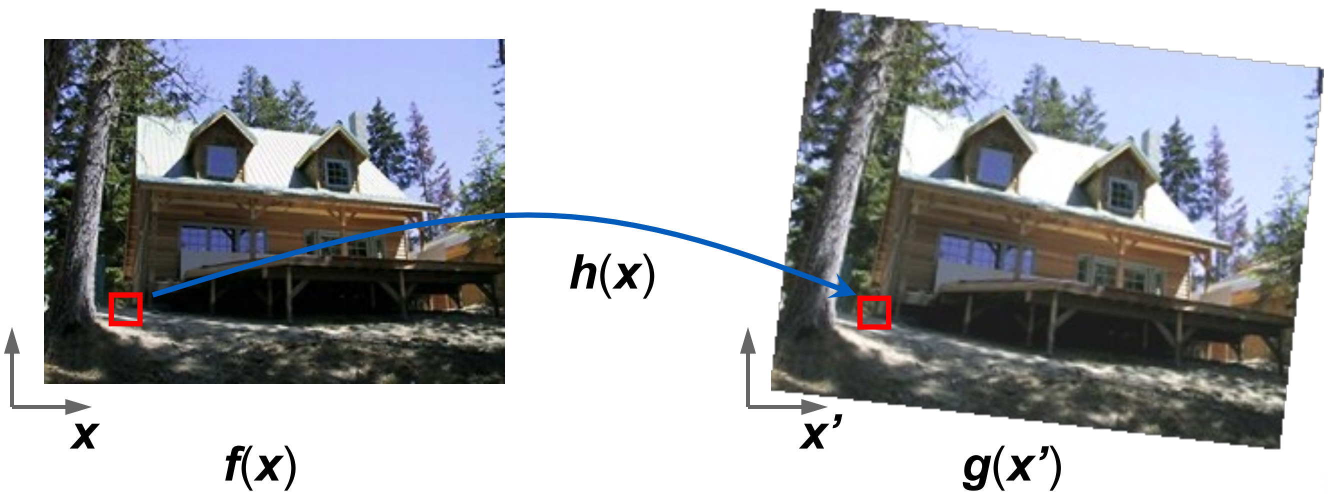


Szeliski

Forward Warping

- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$

  - What if pixel lands "between" two pixels?

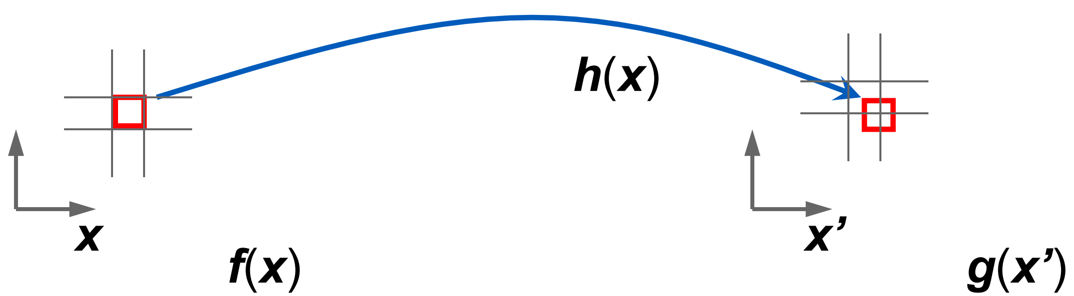


Szeliski

Forward Warping

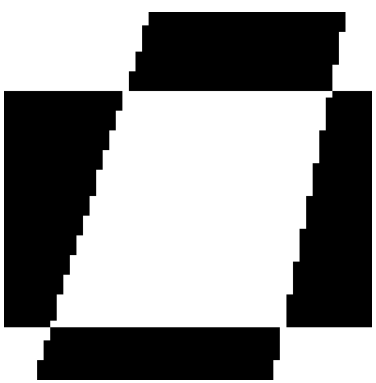
- Send each pixel $\boldsymbol{f}(\boldsymbol{x})$ to its corresponding location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$ in $\boldsymbol{g}(\boldsymbol{x}')$

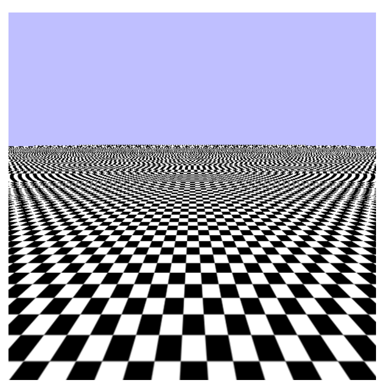
- What if pixel lands "between" two pixels?
- Answer: add "contribution" to several pixels, normalize later



Interpolation

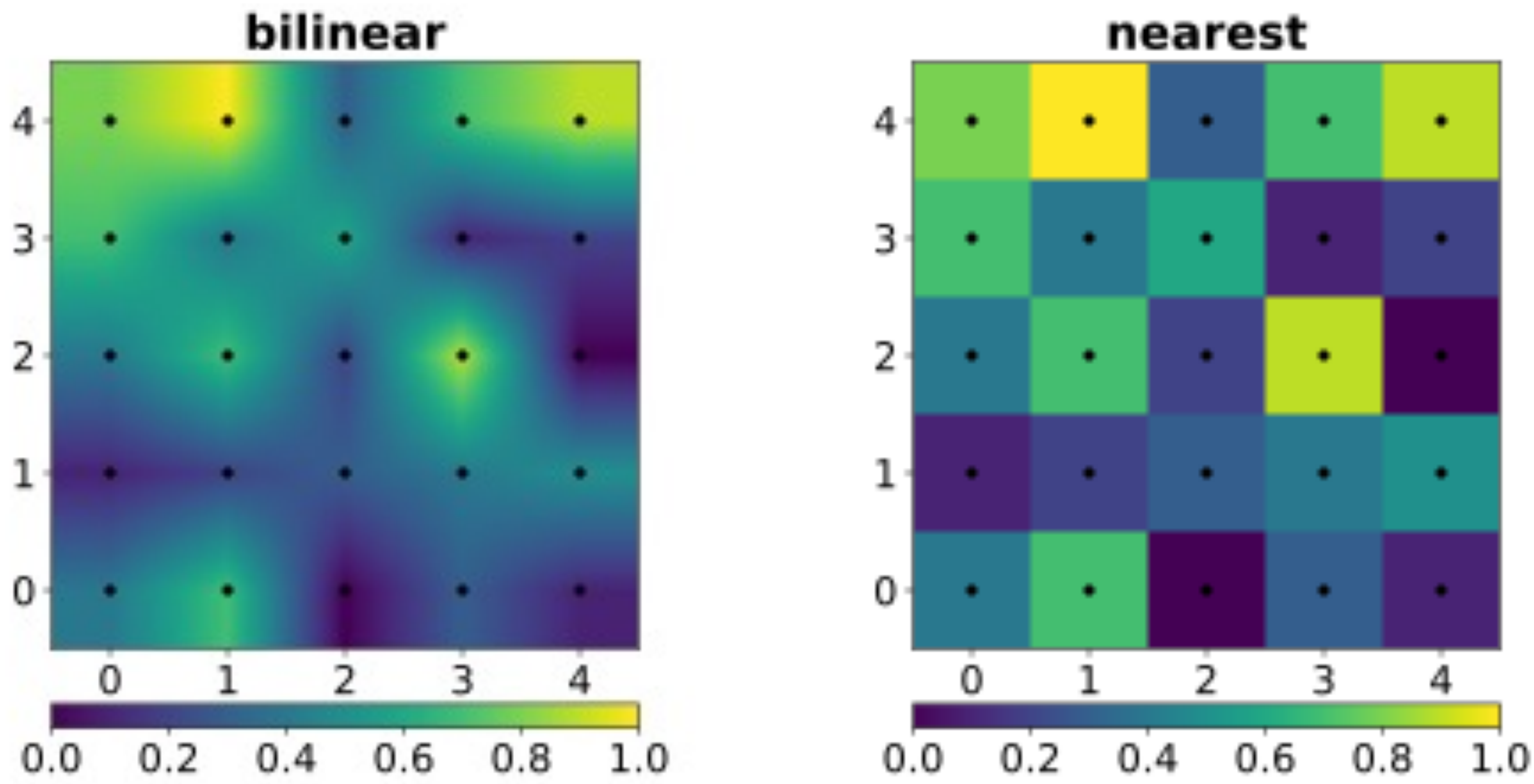
- Possible interpolation filters:
  - **Nearest Neighbor**
  - **Bilinear**
  - **Bicubic**
- Needed to prevent “jaggies” and “texture crawl”





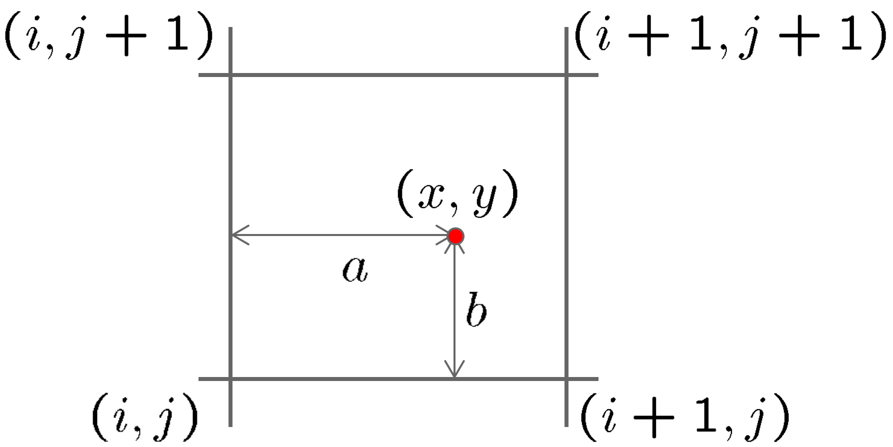
Szeliski

Bilinear interpolation



Bilinear interpolation

Sampling at *f(x,y):*

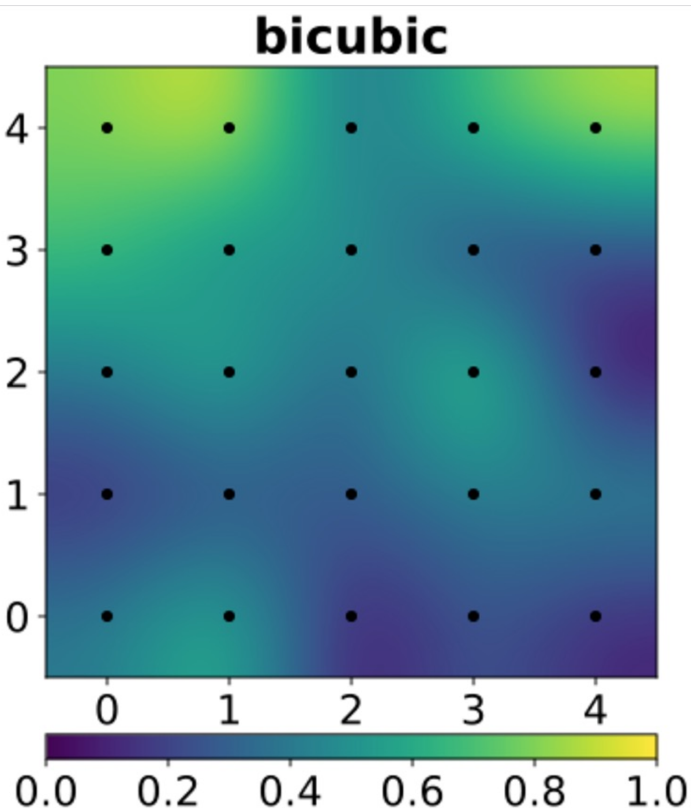


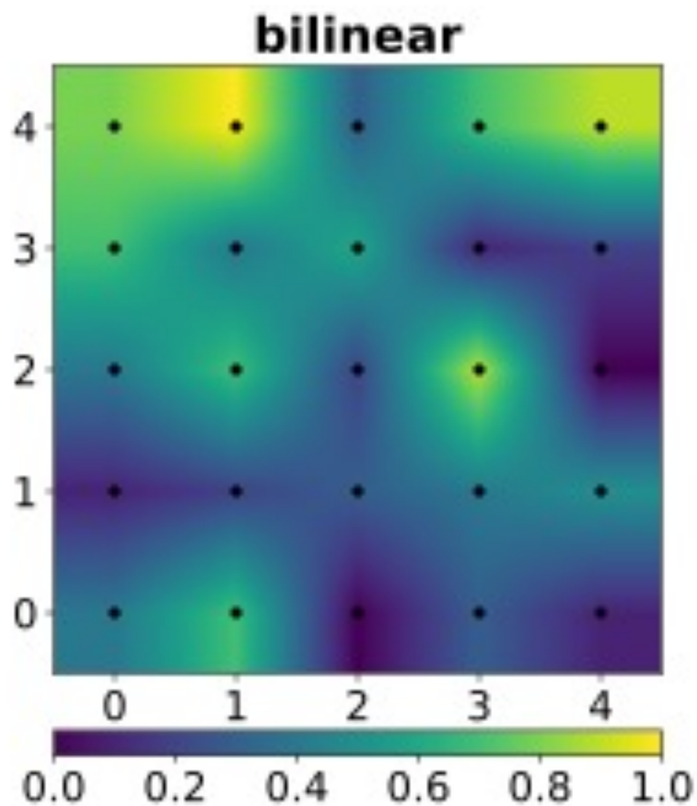
$$
\begin{aligned}
f(x,y) = \ & (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

Reading more

Bicubic interpolation

- Bilinear interpolation processes 2x2 (4 pixels) squares
- Bicubic interpolation processes 4x4 (16 pixels) squares to take image gradients into account.





If the quality is of concern, bicubic would be the best choice.

Bicubic interpolation

$$p(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}x^iy^j.$$

surface $p(x,y)$ on the unit square $[0,1]\times[0,1]$ that is continuous

This requires determining the 16 coefficients.

Consider 4 corners of the unit square. (0, 0) (1, 0) (0, 1) (1, 1)

1. $f(0,0)=p(0,0)=a_{00},$
2. $f(1,0)=p(1,0)=a_{00}+a_{10}+a_{20}+a_{30},$
3. $f(0,1)=p(0,1)=a_{00}+a_{01}+a_{02}+a_{03},$
4. $f(1,1)=p(1,1)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}.$

Bicubic interpolation

We need following derivatives

$$p_x(x,y) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i x^{i-1} y^j,$$

$$p_y(x,y) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} x^i j y^{j-1},$$

$$p_{xy}(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} i x^{i-1} j y^{j-1}.$$

Likewise, eight equations for the derivatives in the x and the y directions:

1. $f_x(0,0) = p_x(0,0) = a_{10},$
2. $f_x(1,0) = p_x(1,0) = a_{10} + 2a_{20} + 3a_{30},$
3. $f_x(0,1) = p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13},$
4. $f_x(1,1) = p_x(1,1) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i,$
5. $f_y(0,0) = p_y(0,0) = a_{01},$
6. $f_y(1,0) = p_y(1,0) = a_{01} + a_{11} + a_{21} + a_{31},$
7. $f_y(0,1) = p_y(0,1) = a_{01} + 2a_{02} + 3a_{03},$
8. $f_y(1,1) = p_y(1,1) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} j.$

And four equations for the xy mixed partial derivative:

1. $f_{xy}(0,0) = p_{xy}(0,0) = a_{11},$
2. $f_{xy}(1,0) = p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31},$
3. $f_{xy}(0,1) = p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13},$
4. $f_{xy}(1,1) = p_{xy}(1,1) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} ij.$

Bicubic interpolation

Grouping the unknown parameters a_{ij} in a vector

$$\alpha = \begin{bmatrix} a_{00} & a_{10} & a_{20} & a_{30} & a_{01} & a_{11} & a_{21} & a_{31} & a_{02} & a_{12} & a_{22} & a_{32} & a_{03} & a_{13} & a_{23} & a_{33} \end{bmatrix}^T$$

and letting

$$x = \begin{bmatrix} f(0,0) & f(1,0) & f(0,1) & f(1,1) & f_x(0,0) & f_x(1,0) & f_x(0,1) & f_x(1,1) & f_y(0,0) & f_y(1,0) & f_y(0,1) & f_y(1,1) & f_{xy}(0,0) & f_{xy}(1,0) & f_{xy}(0,1) & f_{xy}(1,1) \end{bmatrix}^T,$$

the above system of equations can be reformulated into a matrix for the linear equation $A\alpha = x$.

Inverting the matrix gives the more useful linear equation $A^{-1}x = \alpha$, where

$$A^{-1} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\
-3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\
9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\
-6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\
2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
-6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\
4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1
\end{bmatrix},$$

which allows α to be calculated quickly and easily.

2D coordinate transformations

- translation: $\mathbf{x}' = \mathbf{x} + \mathbf{t}$ $\qquad \mathbf{x} = (x, y)$
- rotation: $\mathbf{x}' = \mathbf{R}\,\mathbf{x} + \mathbf{t}$
- similarity: $\mathbf{x}' = s\,\mathbf{R}\,\mathbf{x} + \mathbf{t}$
- affine: $\mathbf{x}' = \mathbf{A}\,\mathbf{x} + \mathbf{t}$
- perspective: $\underline{\mathbf{x}}' \cong \mathbf{H}\,\underline{\mathbf{x}}$ $\qquad \underline{\mathbf{x}} = (x, y, 1)$

  ($\underline{\mathbf{x}}$ is a *homogeneous* coordinate)

- These all form a nested *group* (closed w/ inv.)

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

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

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

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

Rotate

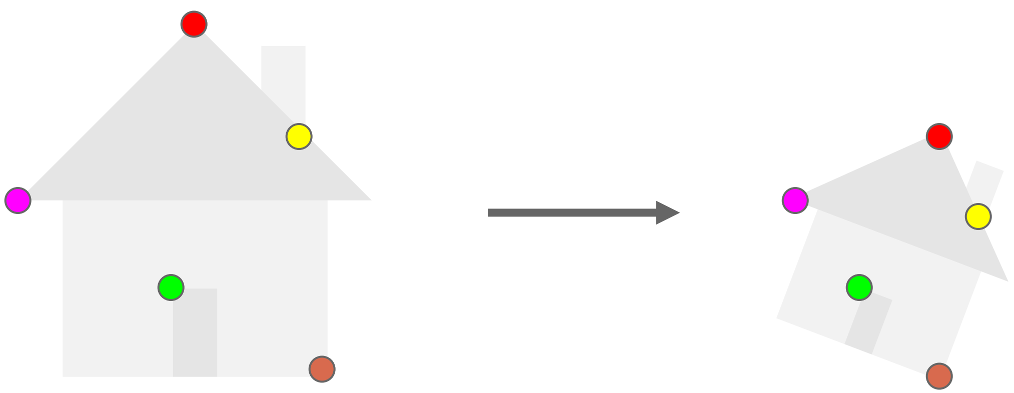
$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} & 0 \\ \boldsymbol{sh_y} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear (Skew)

Source: Alyosha Efros

Image alignment

- Two broad approaches:
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree
  - Feature-based alignment
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment



L. Lazebnik

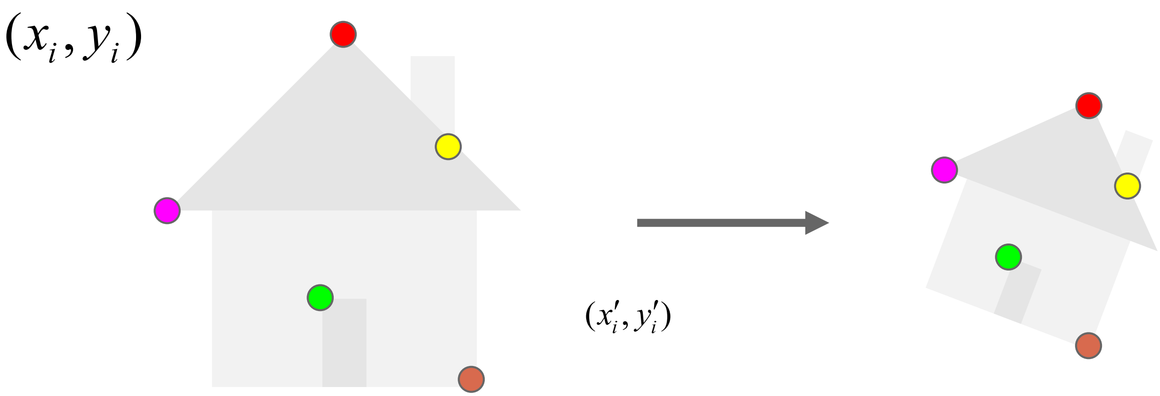
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

Fitting an affine transformation

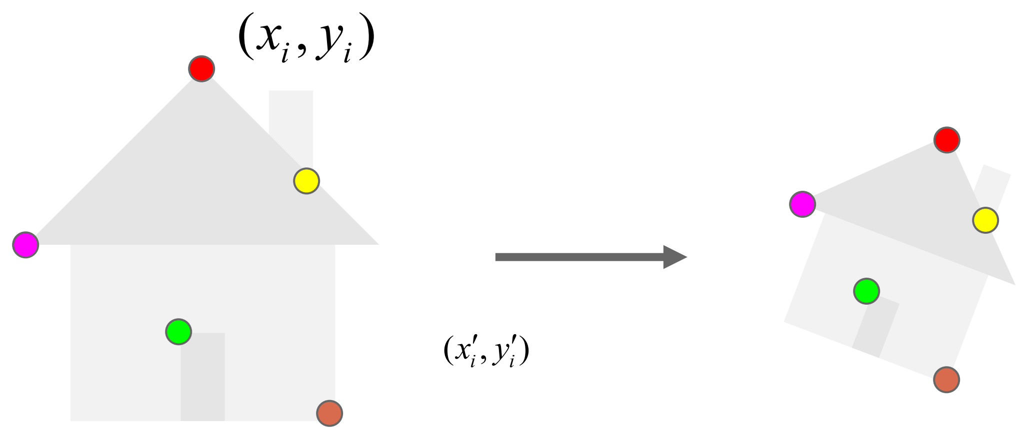
- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

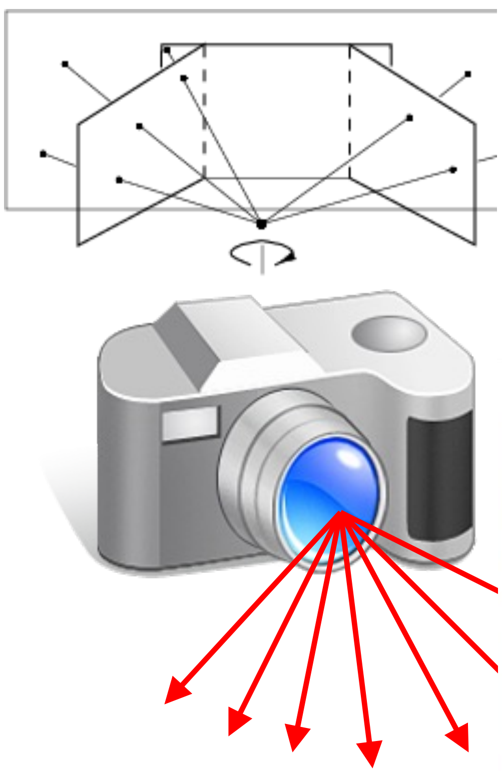
$$\begin{bmatrix} & \\ & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \\ \end{bmatrix}$$

Fitting an affine transformation

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for (x_{new}, y_{new})?

Panoramas



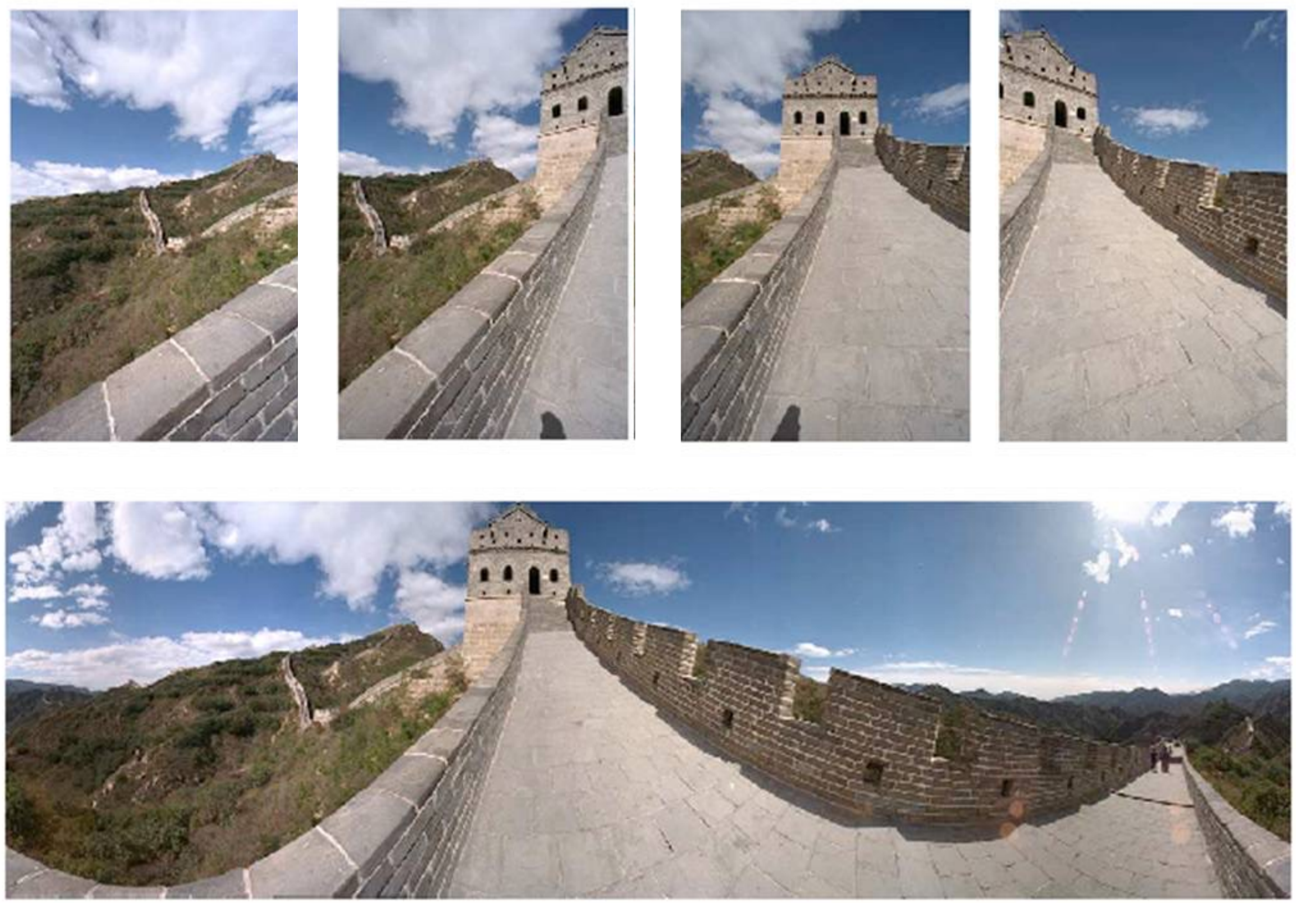


image from S. Seitz

Obtain a wide angle view by combining multiple images.

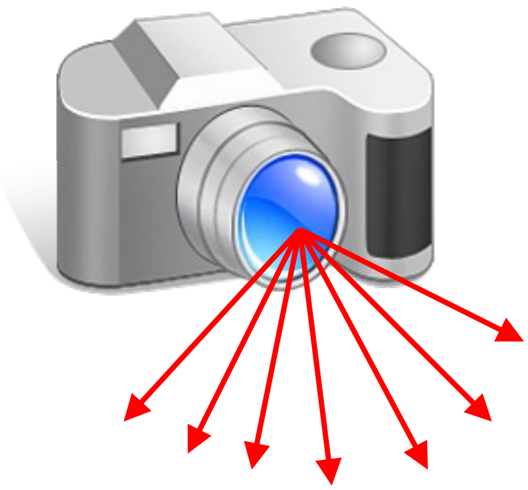
How to stitch together a panorama?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its **optical center.**
  - Compute transformation between 2nd and 1st image
  - Transform the 2nd image to overlap with the 1st
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)
- Why should this work at all?
  - What about the 3D geometry of the scene?
  - Why aren't we using it?

Source: Steve Seitz

Correspondence

- Allows us to map image back to some real space



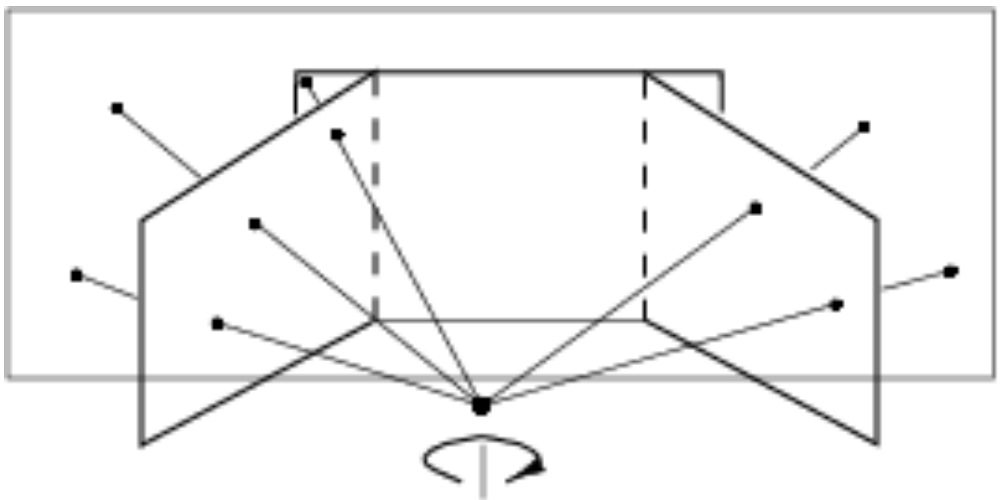
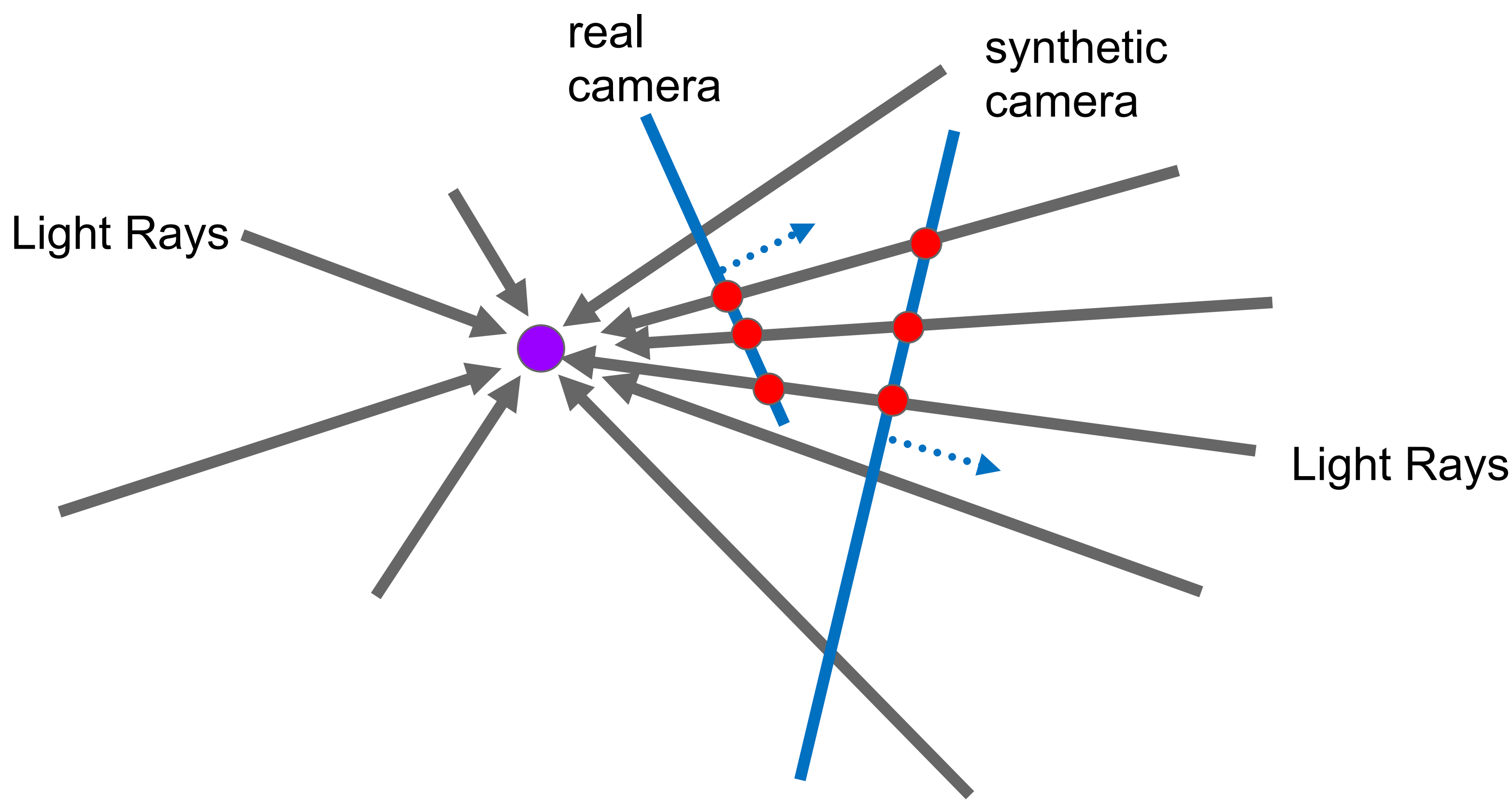


image from S. Seitz

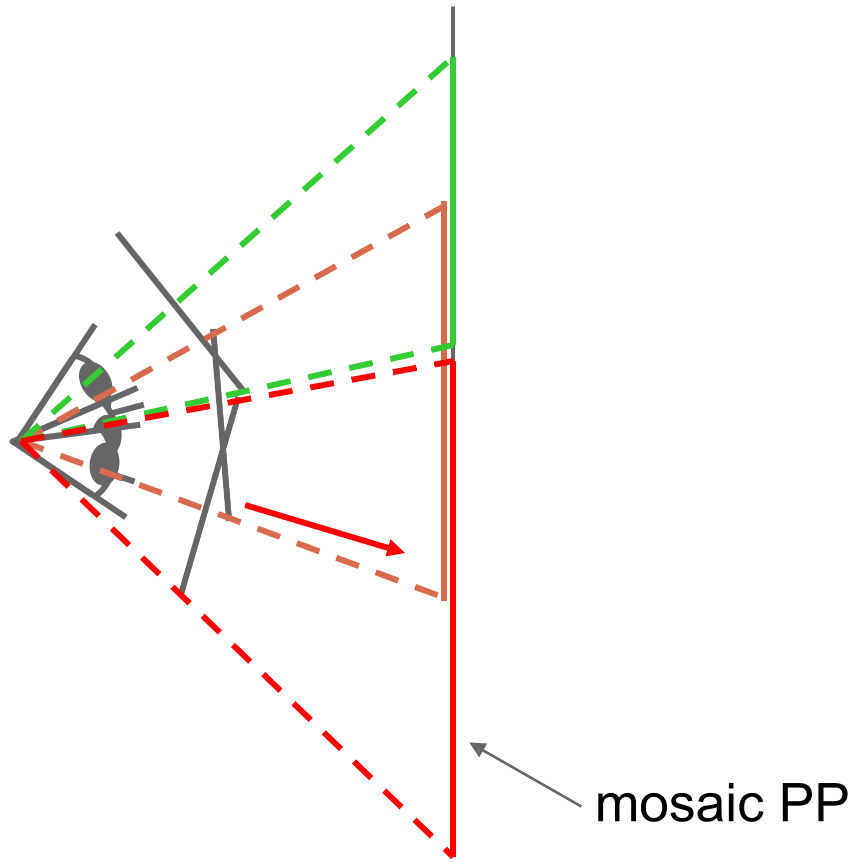
Panoramas: generating synthetic views



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

Source: Alyosha Efros

Image reprojection



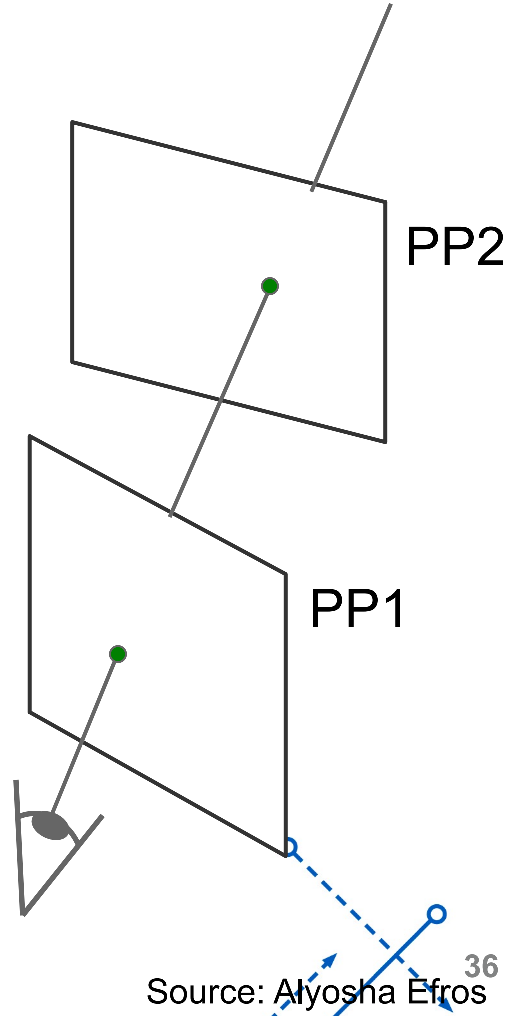
- The mosaic has a natural interpretation in 3D
  - The images are reprojected onto a common plane
  - The mosaic is formed on this plane
  - Mosaic is a *synthetic wide-angle camera*

Source: Steve Seitz

Homography

- How to relate two images from the same camera center?
    - how to map a pixel from PP1 to PP2?
- Take as a 2D **image warp** using projective transform.
- A **projective transform** is a mapping between any two PPs with the same center of projection
    - rectangle should map to arbitrary quadrilateral
    - parallel lines aren't preserved.
    - but straight lines are preserved.
- Called **Homography**

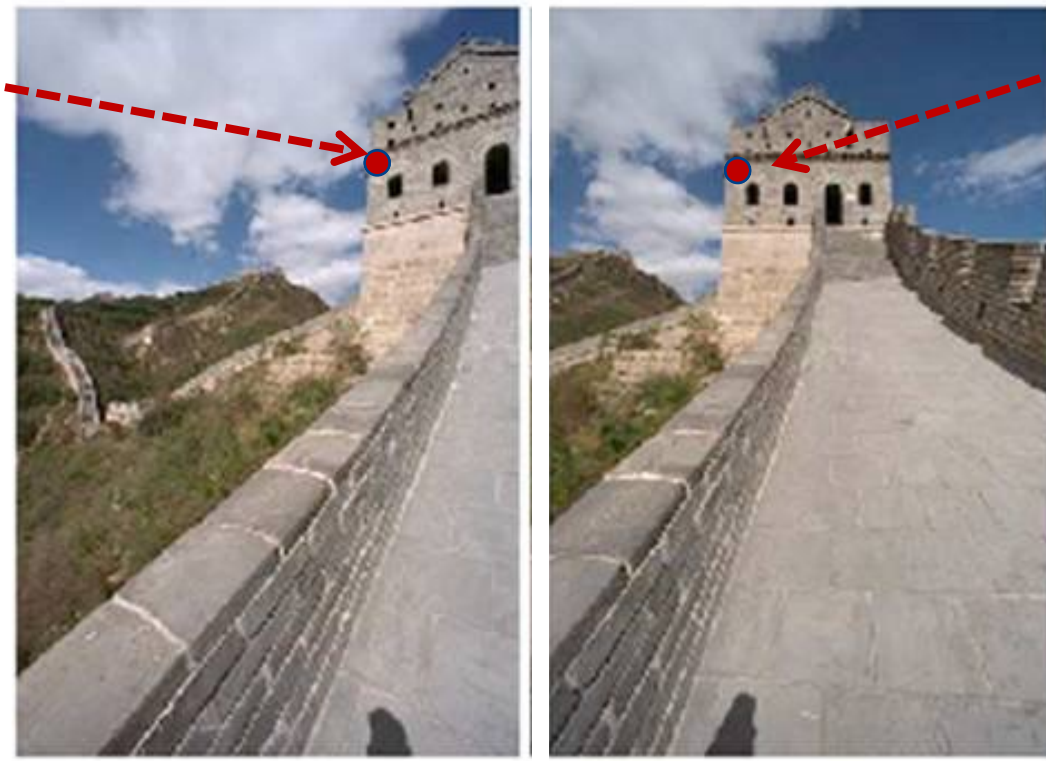
$$\underset{\mathbf{p'}}{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}} = \underset{\mathbf{H}}{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}} \underset{\mathbf{p}}{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}$$



Source: Alyosha Efros

Solving for homographies

$$(x, y)$$



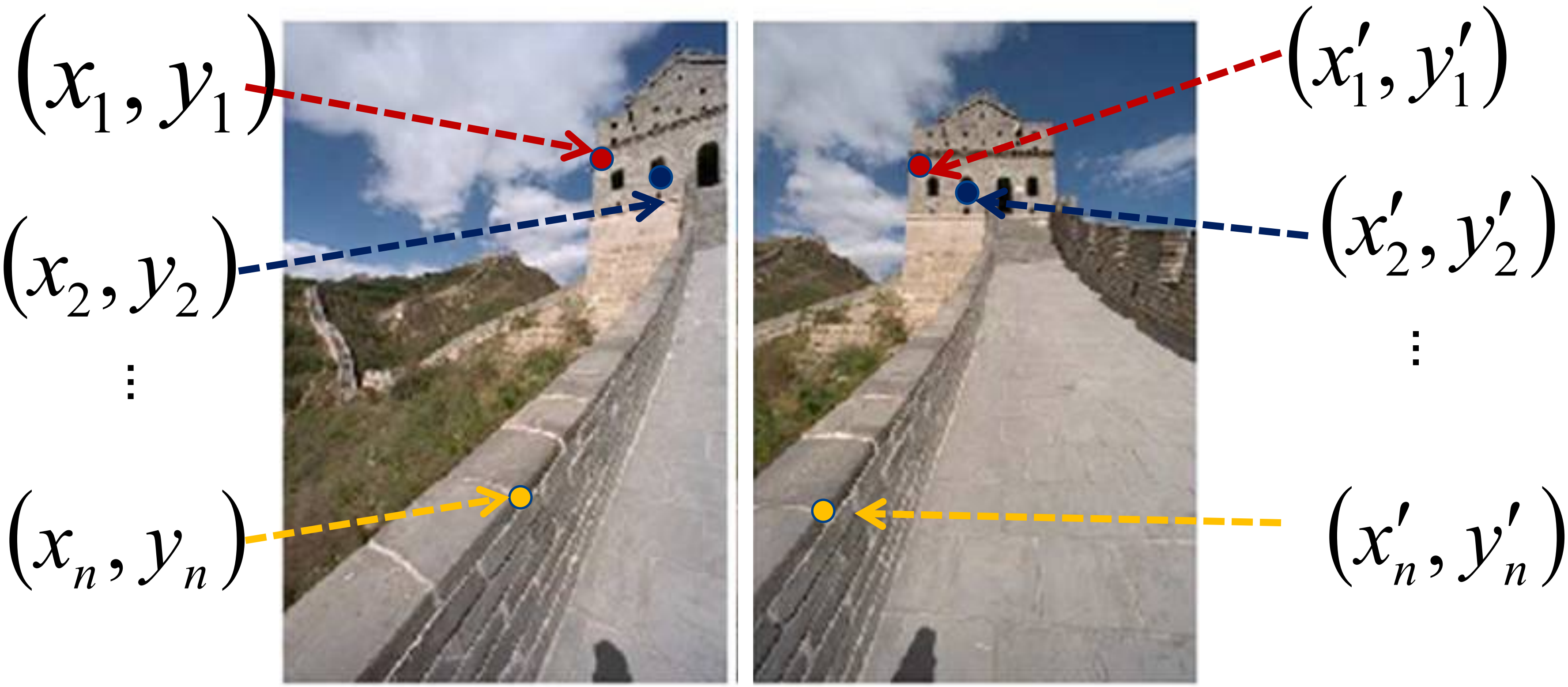
$$\left({}^{wx'}\!/_{w}, {}^{wy'}\!/_{w} \right) = (x', y')$$

To **apply** a given homography **H**

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

$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$

Solving for homographies



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

Solving for homographies

p' = Hp

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

- Can set scale factor *i*=1 or ||H|| = 1. So, there are 8 unknowns.
- 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 equations (4 points), but the more the better…
- Solve for H. If over-constrained, solve using least-squares:

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

$$h = (A^T A)^{-1} A^T b$$

Grauman

SAIR
Spatial AI & Robotics Lab

Proof of least squares

- $F(h) = ||Ah - b||^2 = (Ah - b)^T\ (Ah - b)$
- $F(h) = h^T A^T Ah - h^T A^T b - b^T Ah + b^T b$
- $\frac{\partial}{\partial h} F(h) = 2A^T Ah - A^T b - (b^T A)\hat{}T$
- Setting derivative to 0: $\frac{\partial}{\partial h} F(h)$=0
- $A^T Ah = A^T b$
- $h = (A^T A)^{-1} A^T b$

How to stitch together a panorama?

- 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

Source: Steve Seitz