Unit 7
Image Alignment and Stitching

Reading: Szeliski's book
Sec. 6.1
Chapter 9: Image Stitching

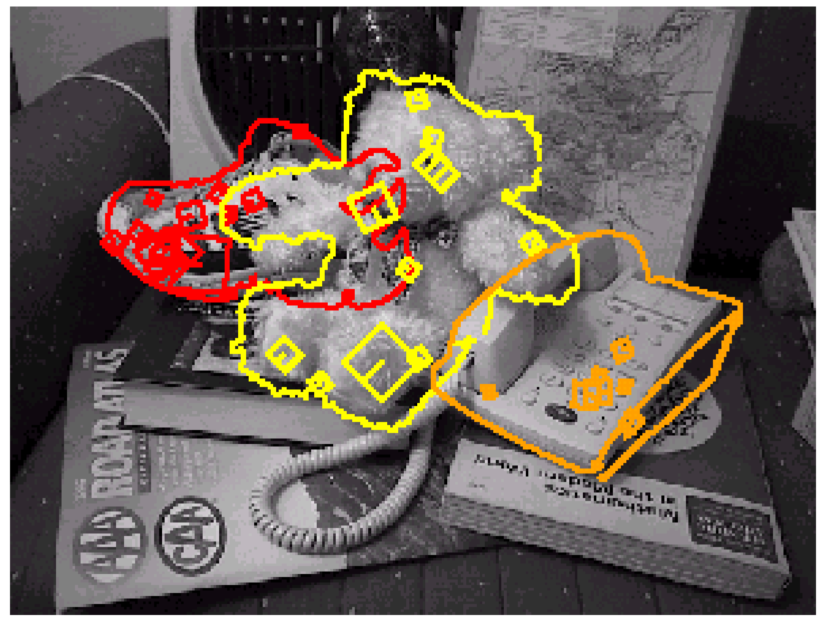
CS 6550

Image Alignment and Stitching

- Homographies
- Rotational Panoramas
- RANSAC
- Global alignment
- Warping
- Blending

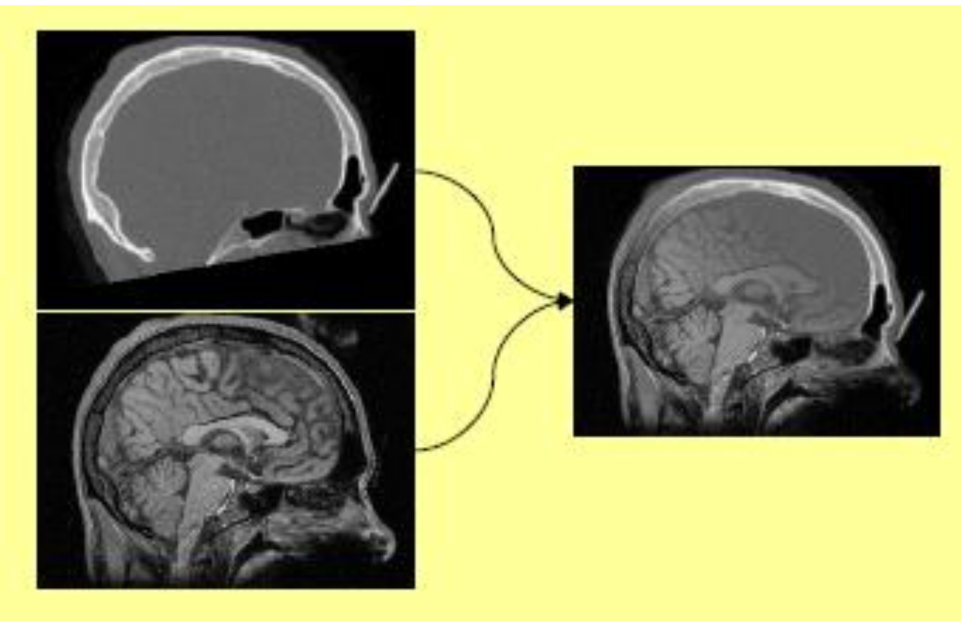
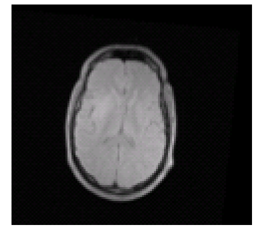
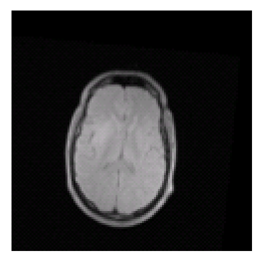
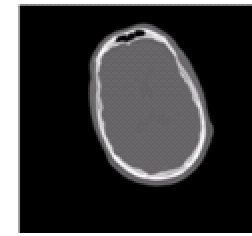
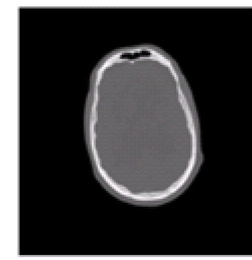
(a) (b) (c)

Motivation: Recognition



Figures from David Lowe

Motivation: medical image registration


Motivation: Mosaics

- Getting the whole picture
  - Consumer camera: 50˚ x 35˚

Slide from Brown & Lowe 2003

Motivation: Mosaics

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

Slide from Brown & Lowe 2003

Motivation: Mosaics

- Getting the whole picture
  - Consumer camera: 50˚ x 35˚
  - Human Vision: 176˚ x 135˚

- Panoramic Mosaic = up to 360 x 180°

Slide from Brown & Lowe 2003

Motion models

- What happens when we take two images with a camera and try to align them?
- translation?
- rotation?
- scale?
- affine?
- perspective?
- … see interactive demo (VideoMosaic)

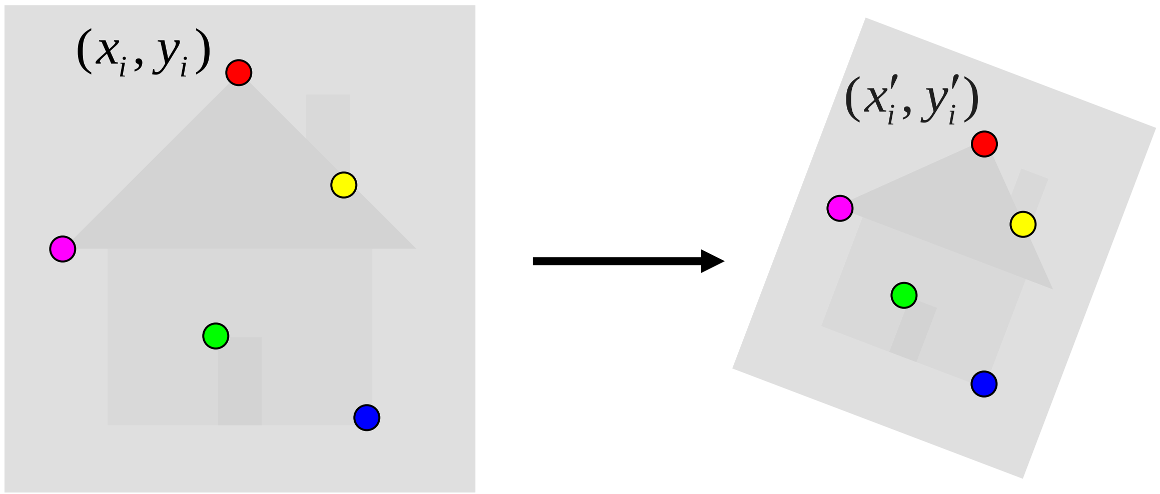
Fitting an affine transformation

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?

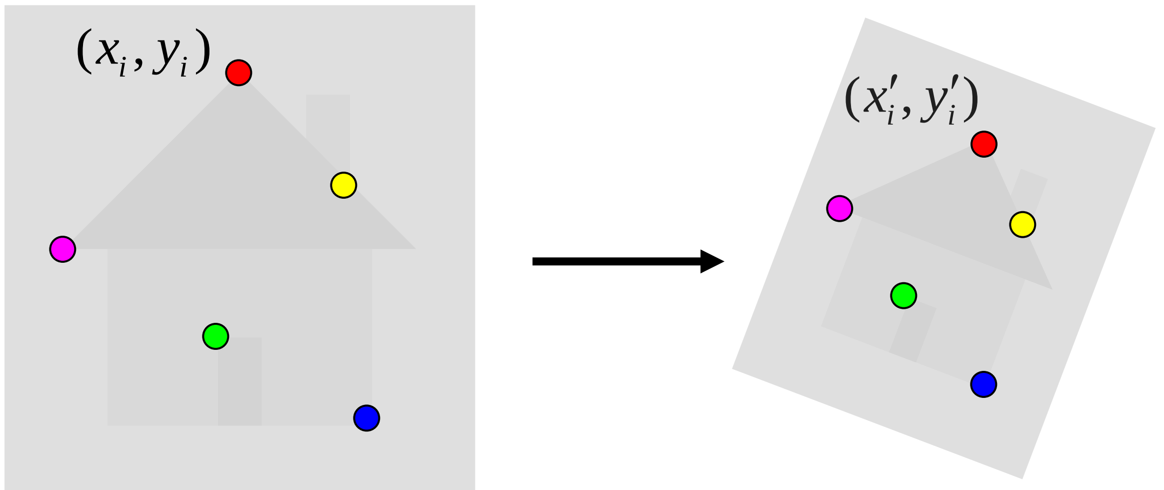


$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Grauman

Fitting an affine transformation

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



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

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

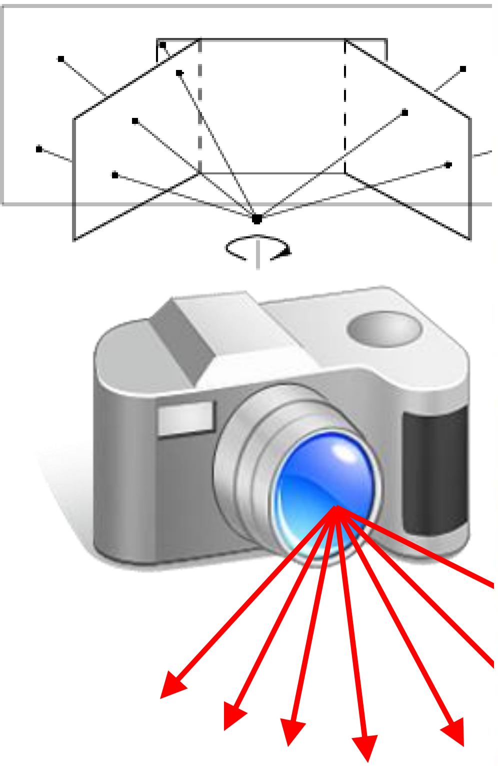
Fitting an affine transformation

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

- 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}) ?

Grauman

Panoramas



. . .

image from S. Seitz

Obtain a wider angle view by combining multiple images.

Grauman

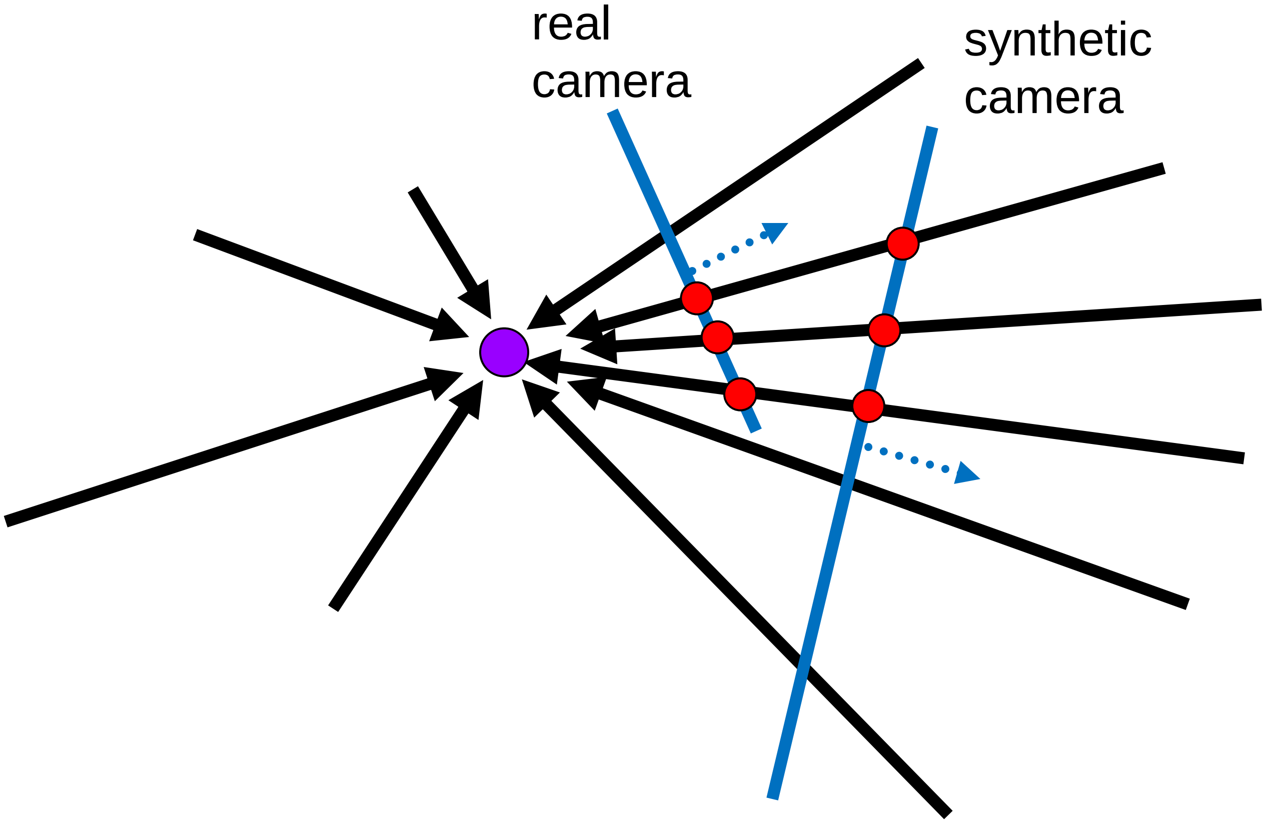
How to stitch together a panorama?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between 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)

- …but **wait**, why should this work at all?
  - What about the 3D geometry of the scene?
  - Why aren’t we using it?

Source: Steve Seitz

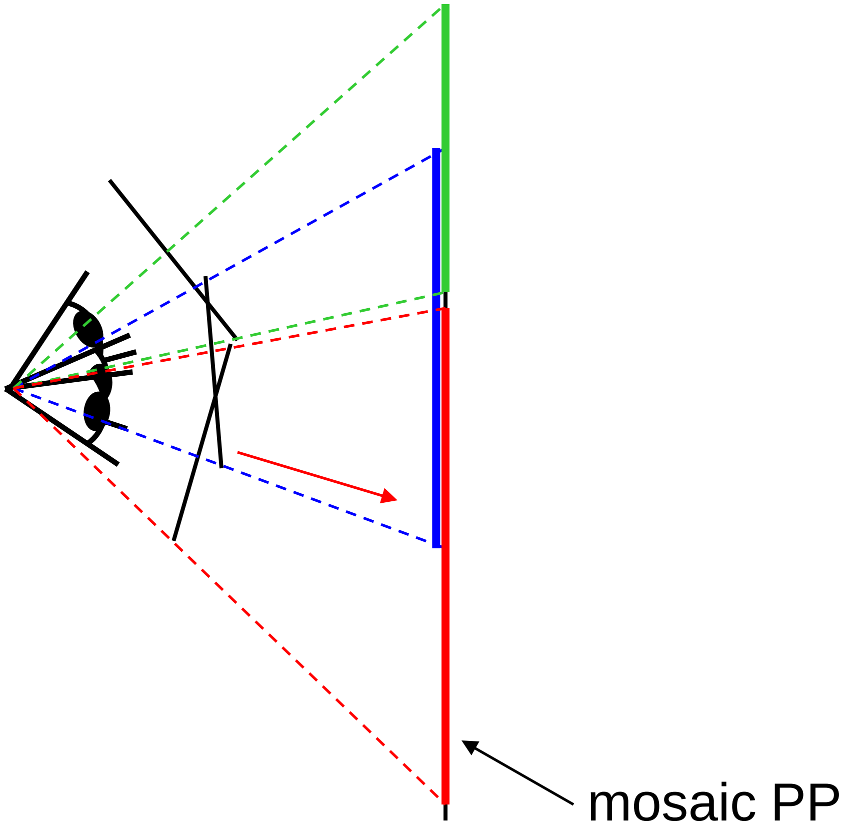
Panoramas: generating synthetic views



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?

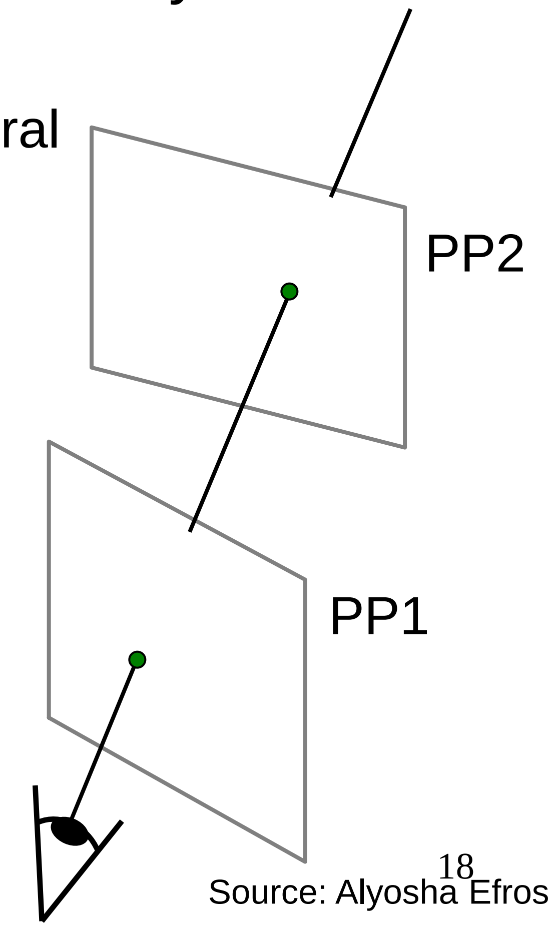
Think of it as a 2D **image warp** from one image to another.

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
- but must preserve straight lines

called **Homography**

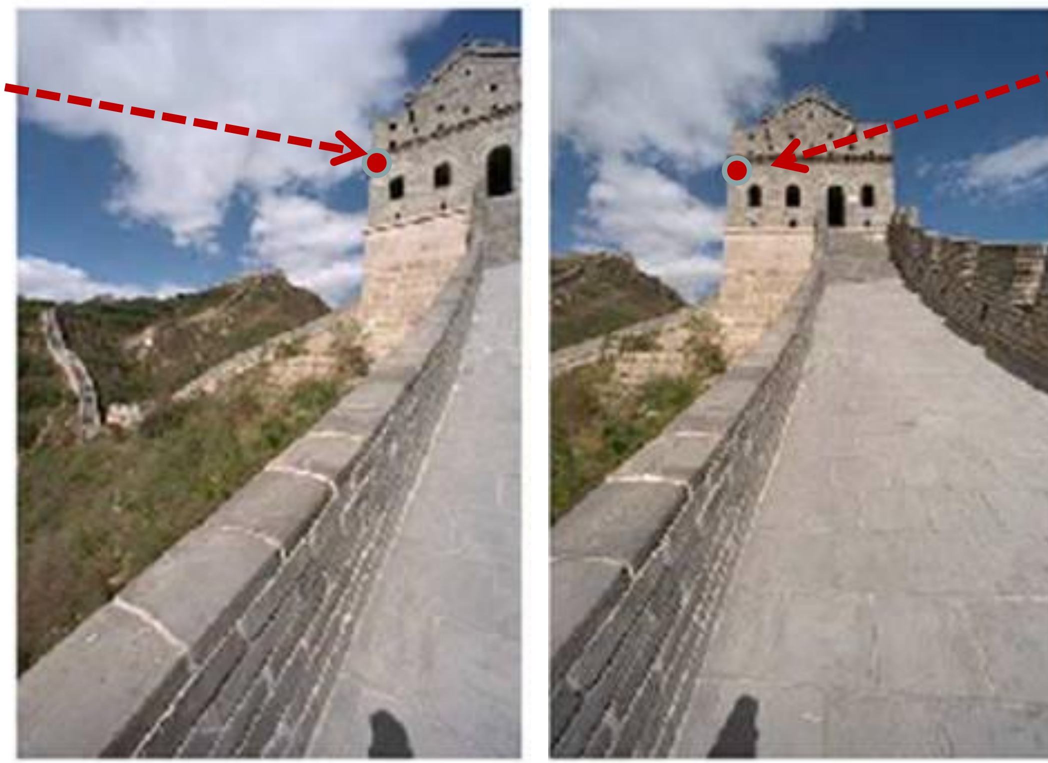
$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p'}} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$



Source: Alyosha Efros

Homography

$$(x, y)$$



$$\left(\frac{wx'}{w}, \frac{wy'}{w} \right) = (x', y')$$

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

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

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

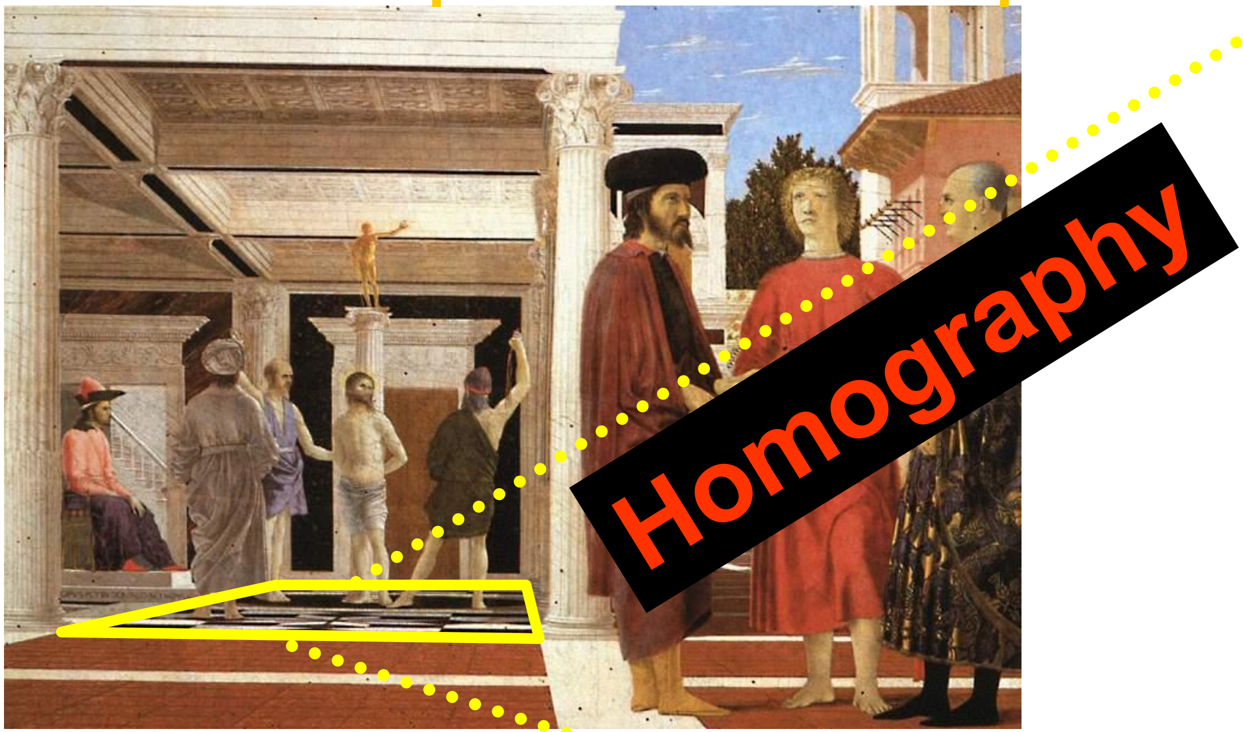
Recap: How to stitch together a panorama?

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between 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

Analysing patterns and shapes

What is the shape of the b/w floor pattern?



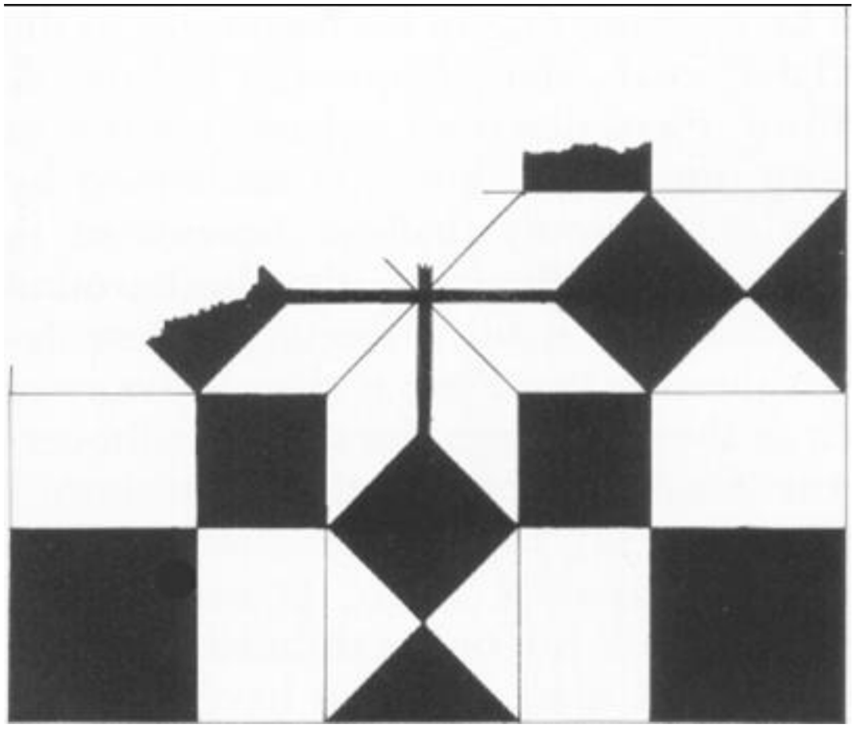
The floor (enlarged)

Automatically rectified floor

Slide from Criminisi

Analysing patterns and shapes

Automatic rectification



From Martin Kemp *The Science of Art* *(manual reconstruction)*

Analysing patterns and shapes

What is the (complicated) shape of the floor pattern?

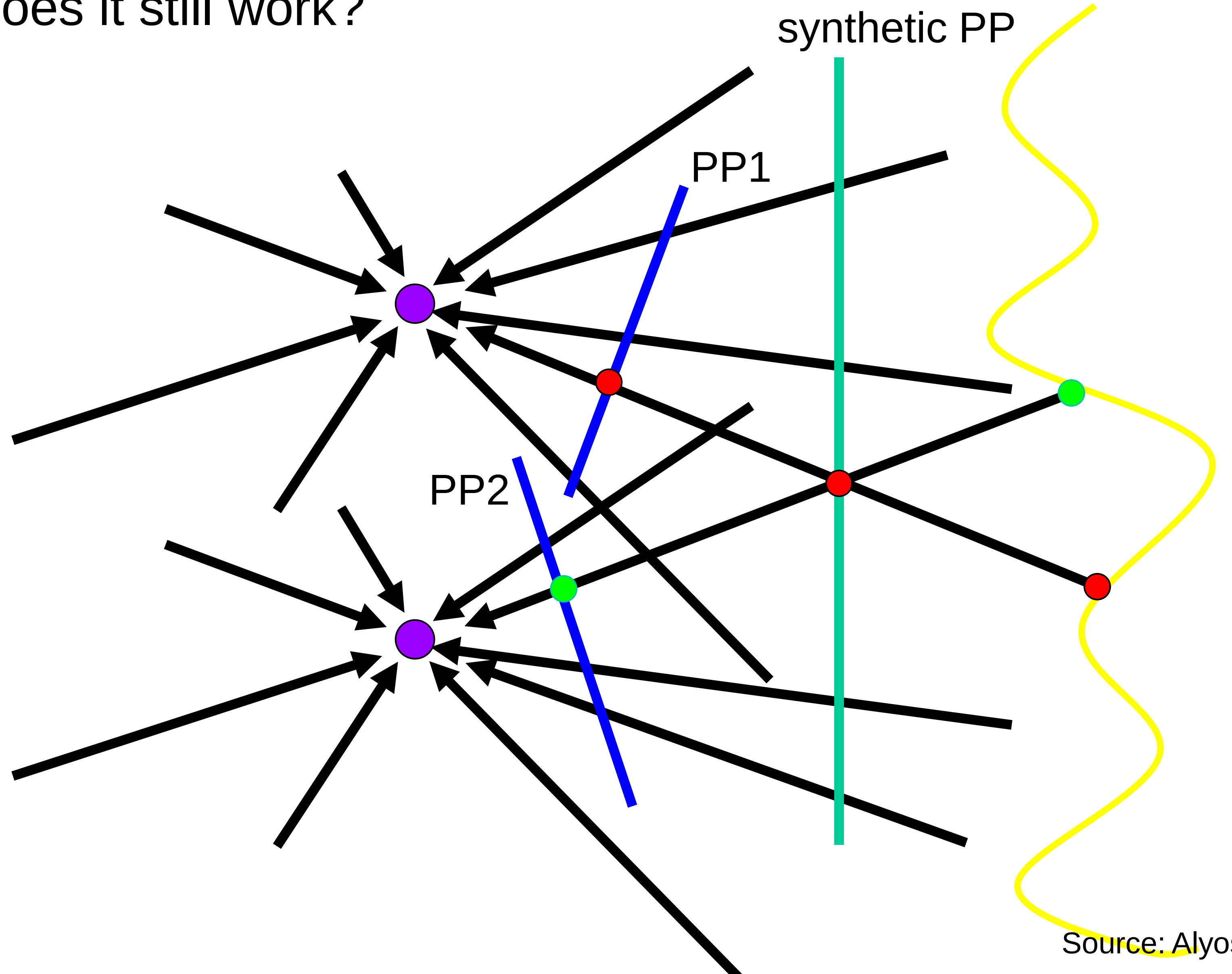
Automatically rectified floor

St. Lucy Altarpiece, **D. Veneziano**

Slide from Criminisi

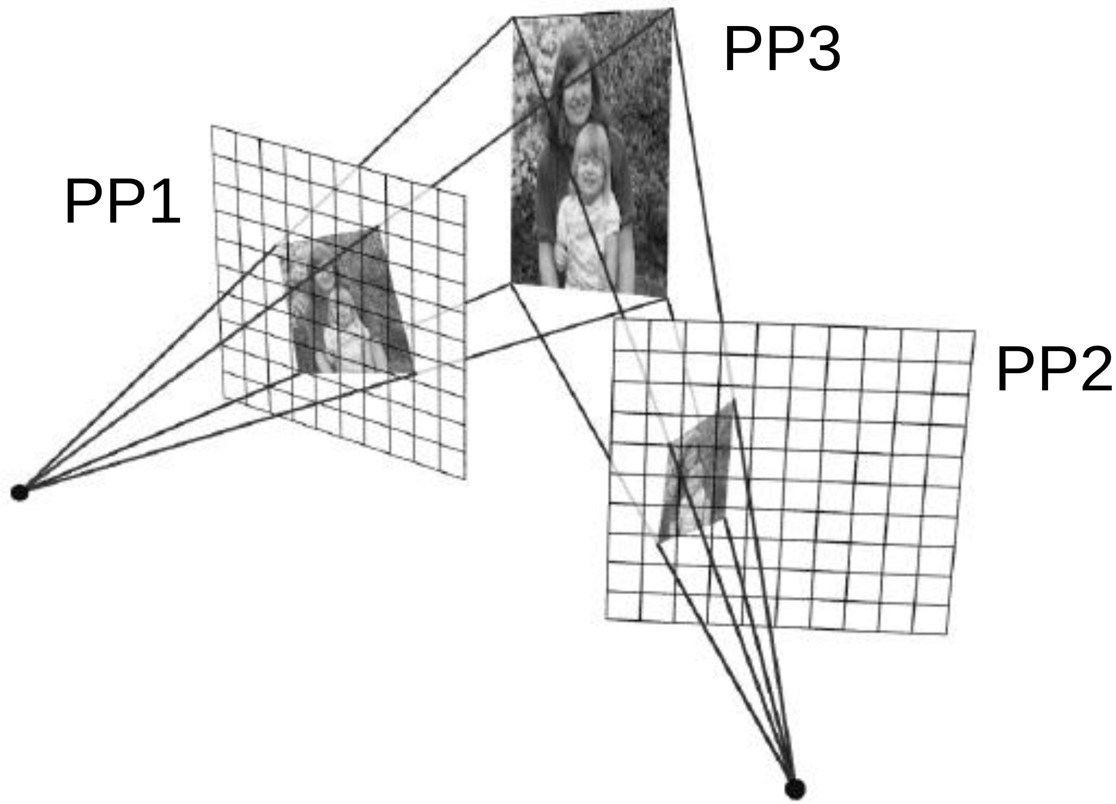
changing camera center

Does it still work?



Source: Alyosha Efros

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

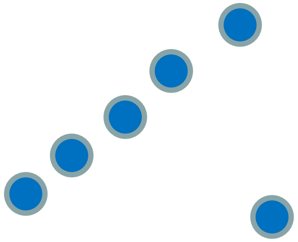
Source: Alyosha Efros


Map
Satellite
Hybrid
200 ft
100 m
©2007 Google - Terms of Use

Outliers

- **Outliers** can hurt the quality of our parameter estimates, e.g.,
  - an erroneous pair of matching points from two images
  - an edge point that is noise, or doesn't belong to the line we are fitting.

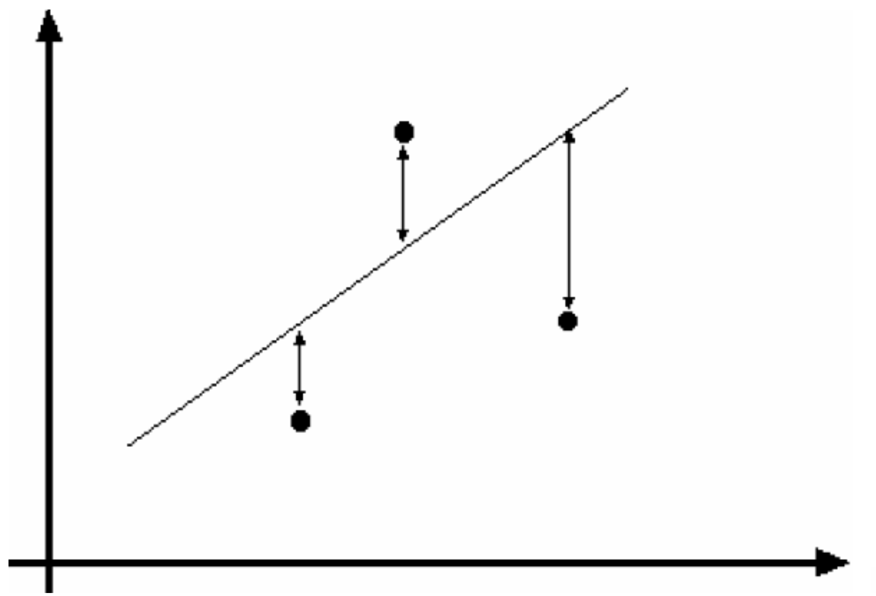




Grauman

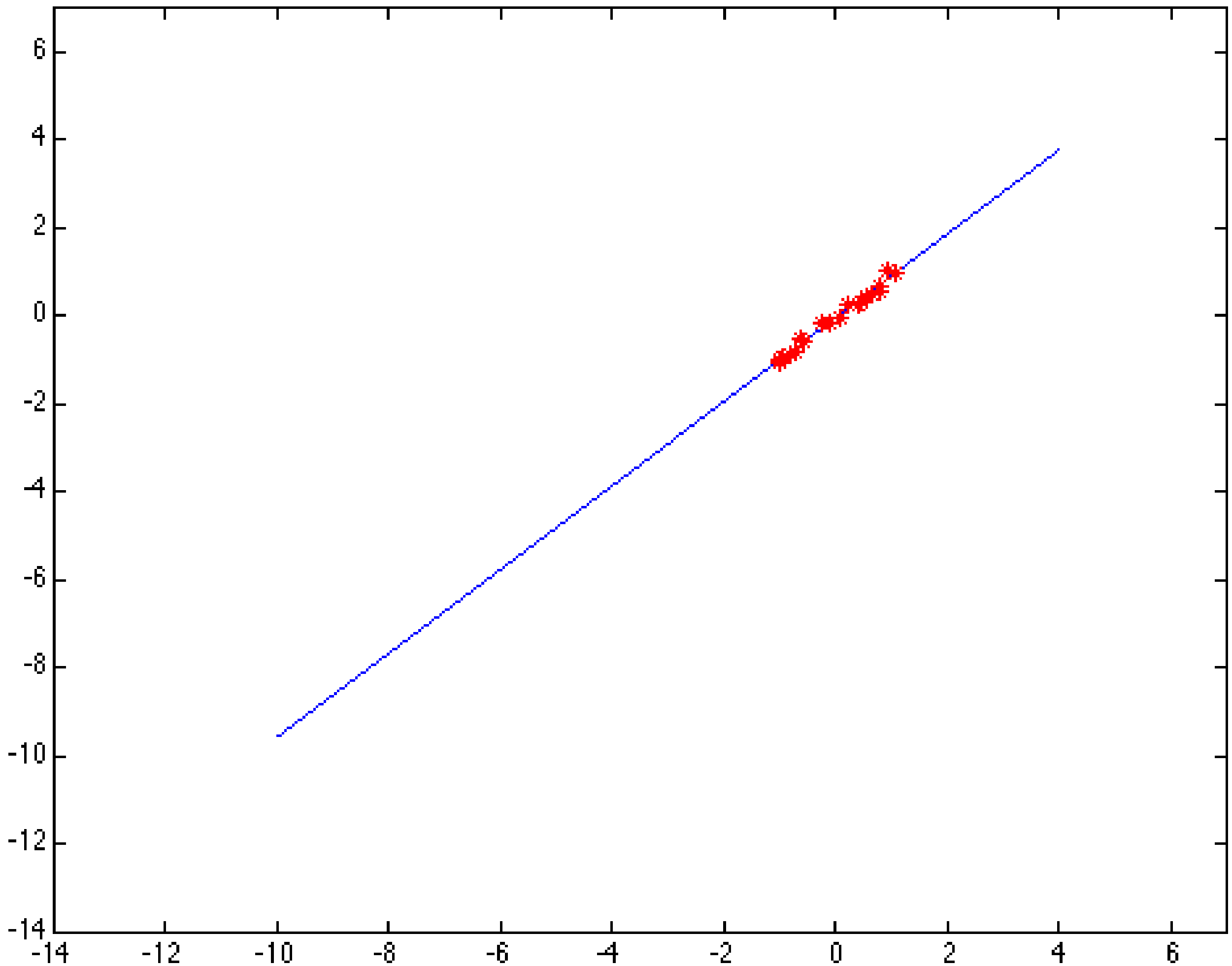
Example: least squares line fitting

- Assuming all the points that belong to a particular line are known

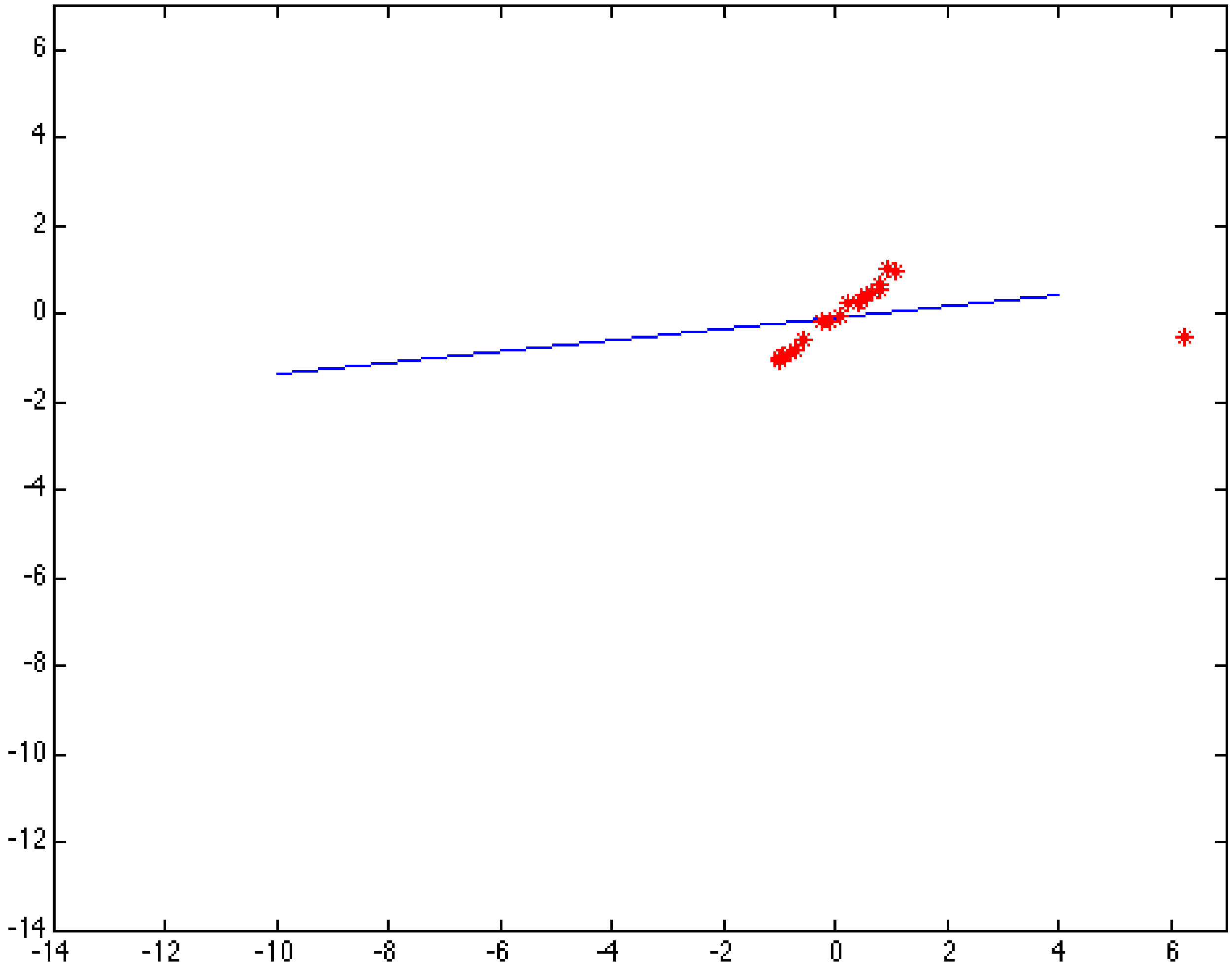


Grauman

Outliers affect least squares fit



Outliers affect least squares fit



RANSAC

- RANdom Sample Consensus
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

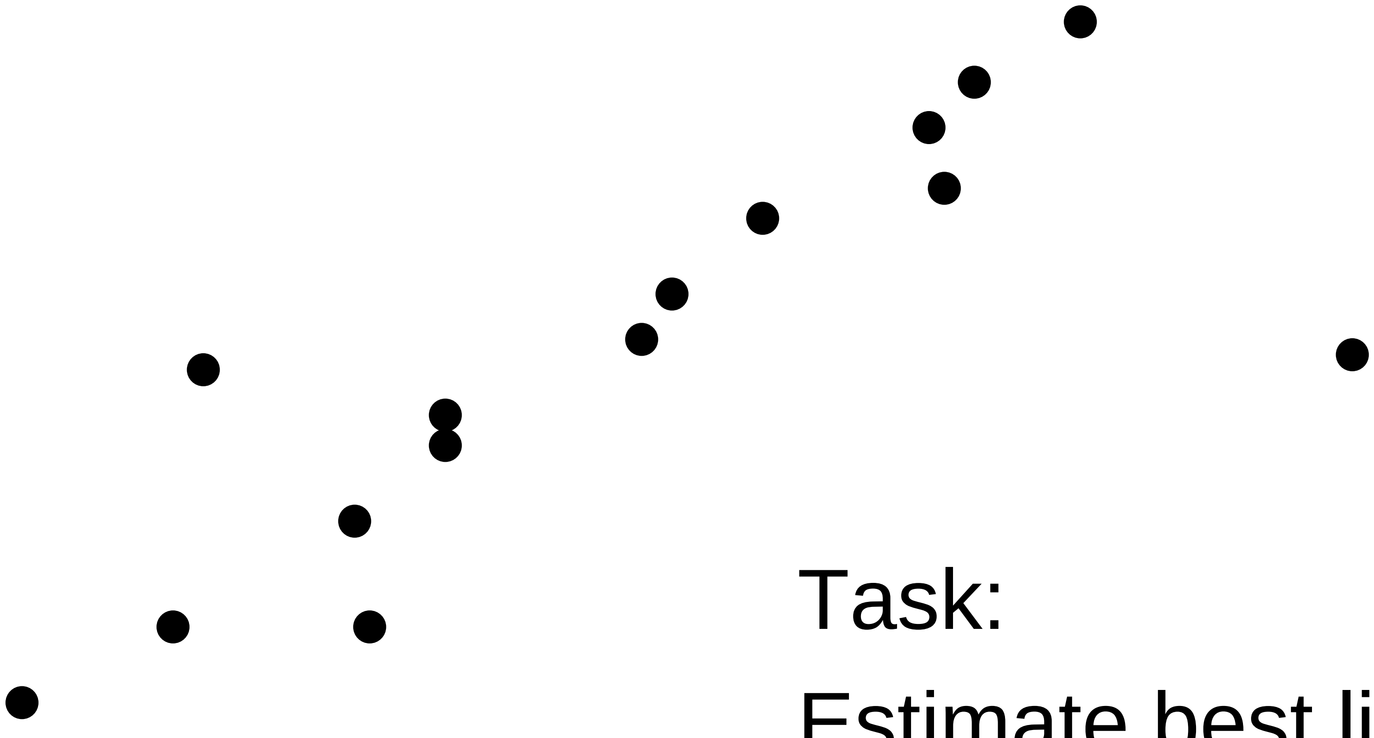
RANSAC

- RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

RANSAC Line Fitting Example

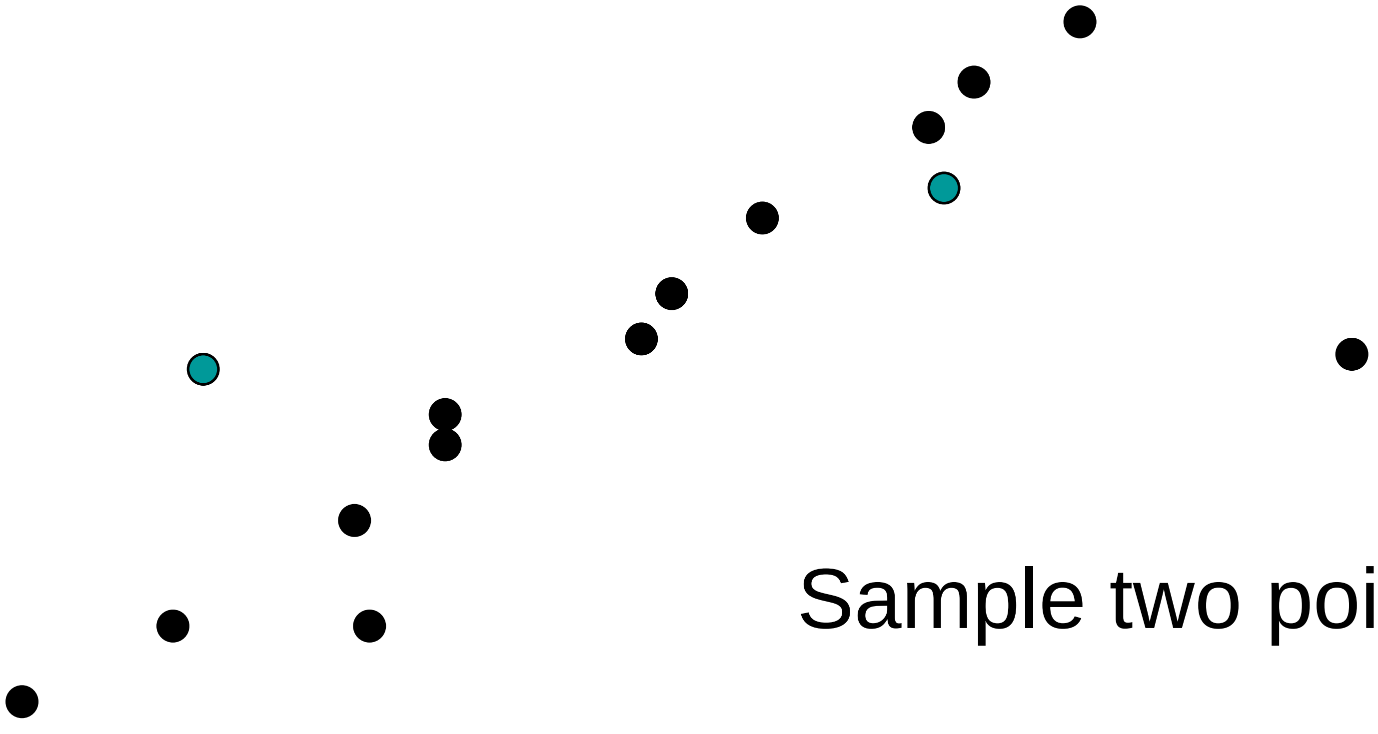


Task:

Estimate best line

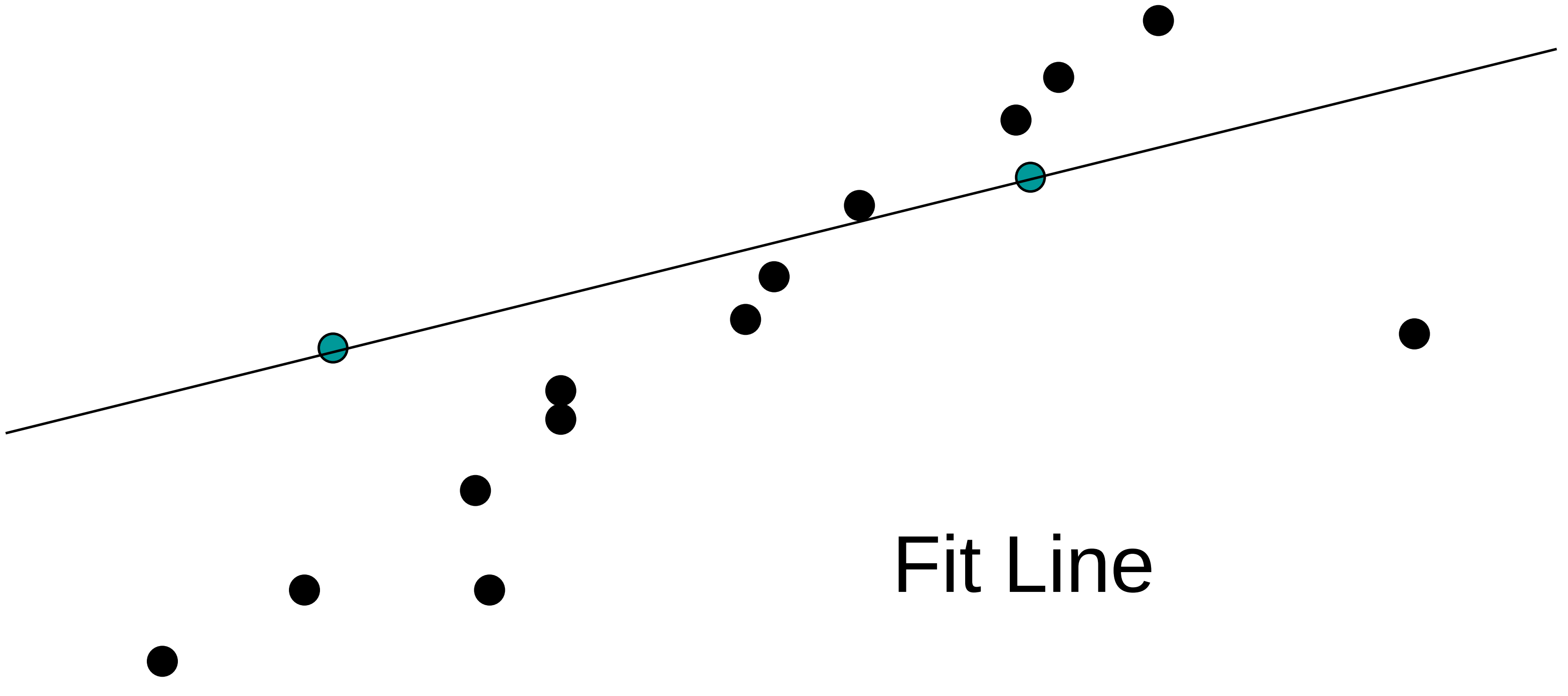
Slide credit: Jinxiang Chai, CMU

RANSAC Line Fitting Example

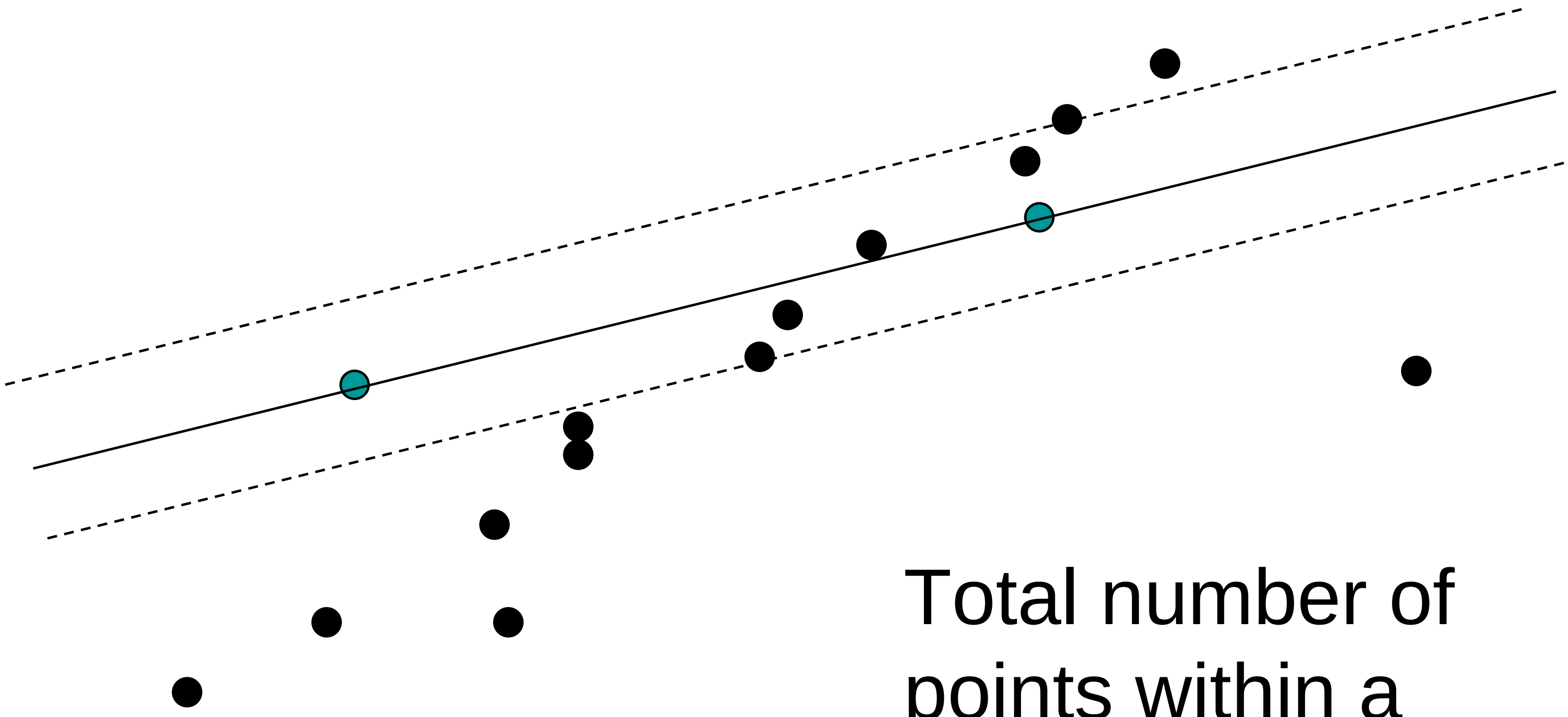


Sample two points

RANSAC Line Fitting Example



RANSAC Line Fitting Example

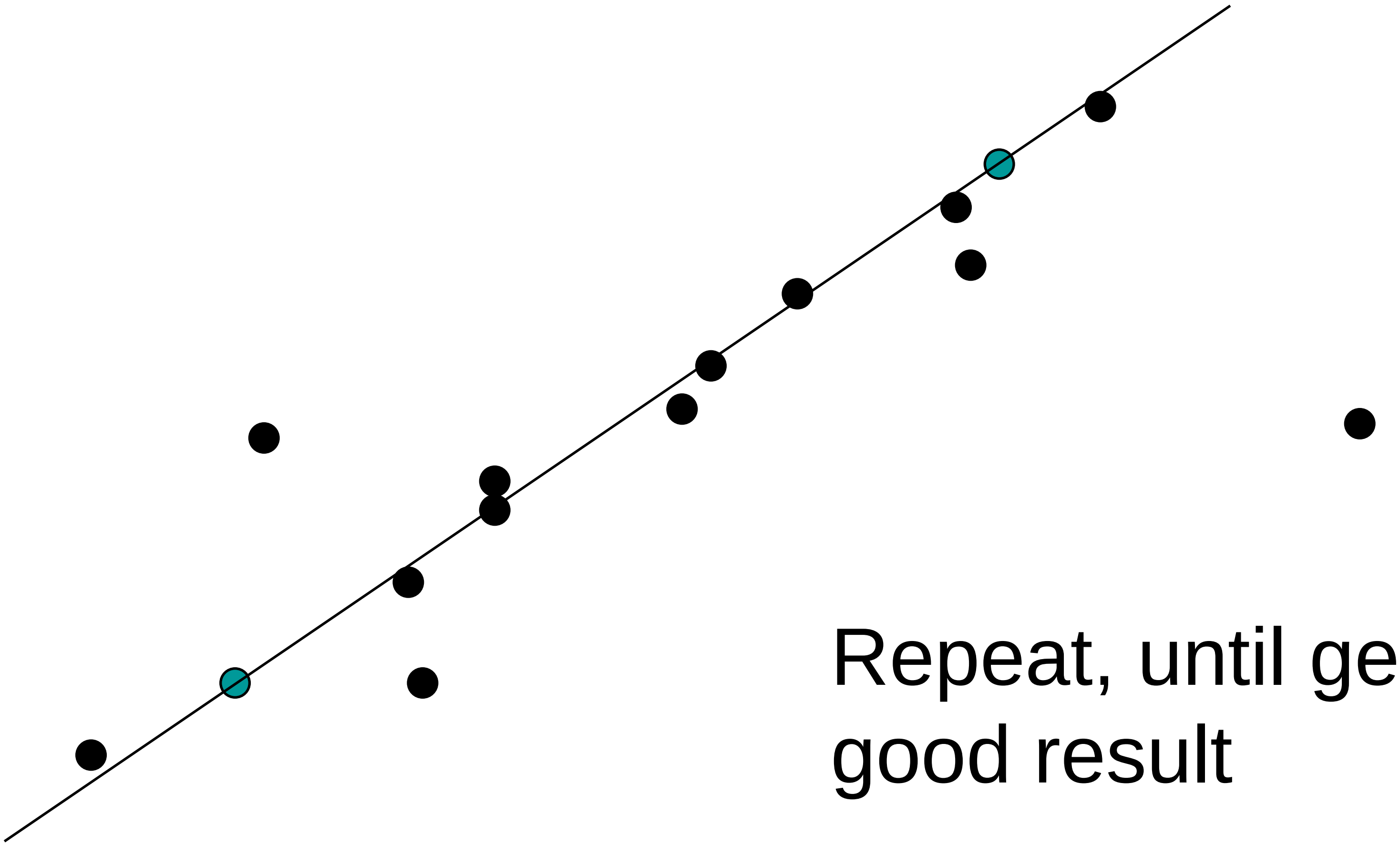


Total number of points within a threshold of line.

RANSAC Line Fitting Example

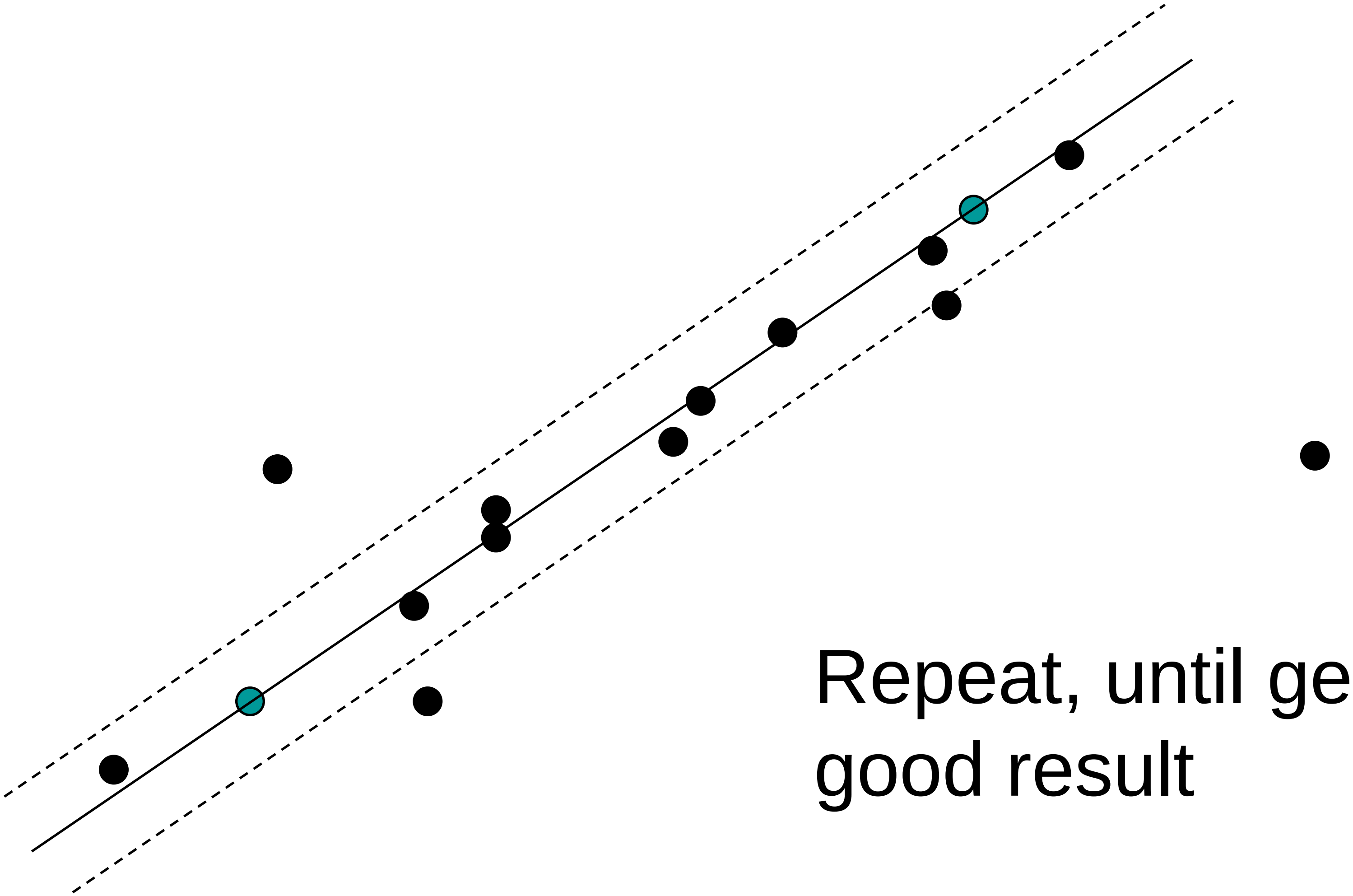
Repeat, until get a good result

RANSAC Line Fitting Example



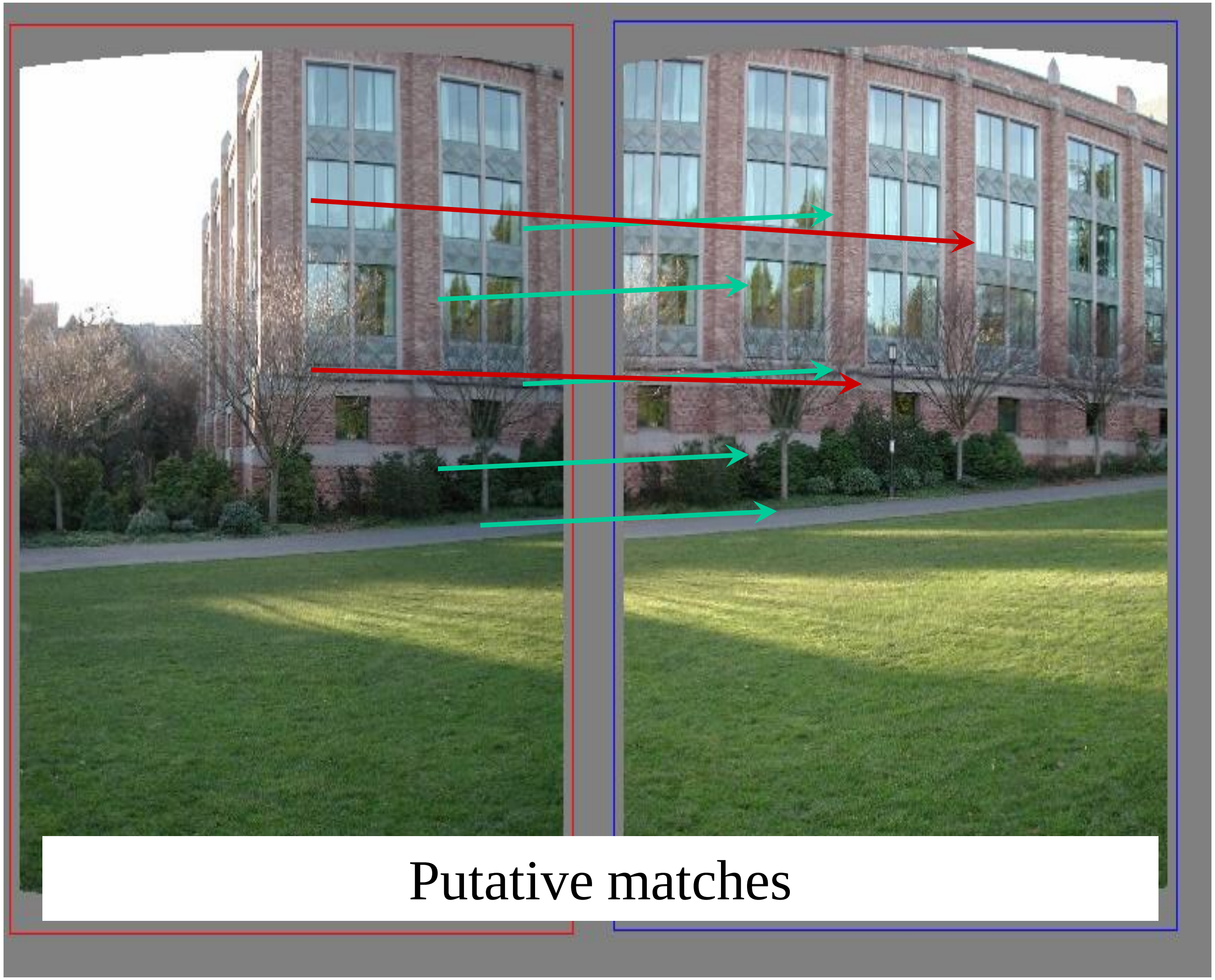
Repeat, until get a good result

RANSAC Line Fitting Example



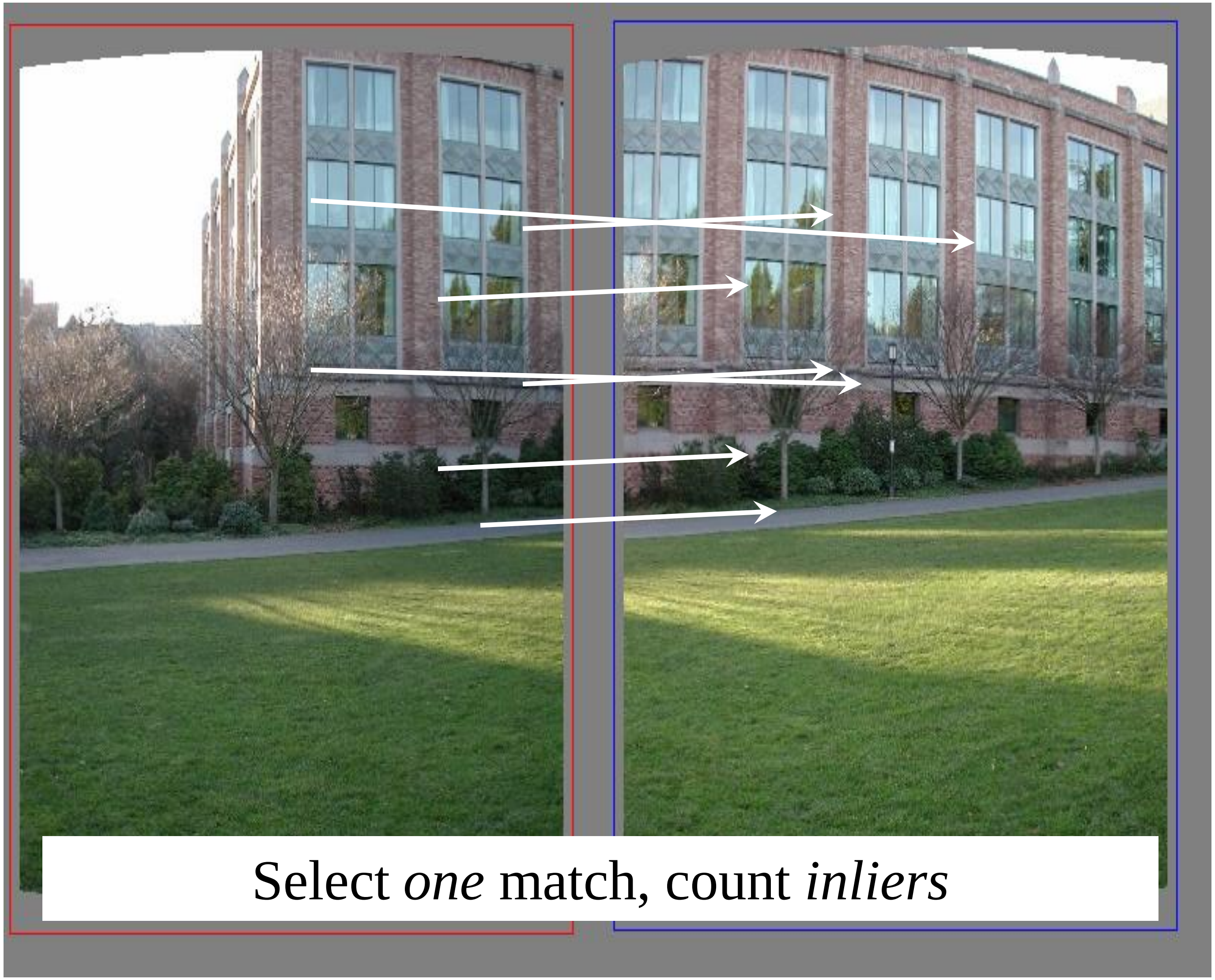
Repeat, until get a good result

RANSAC example: Translation



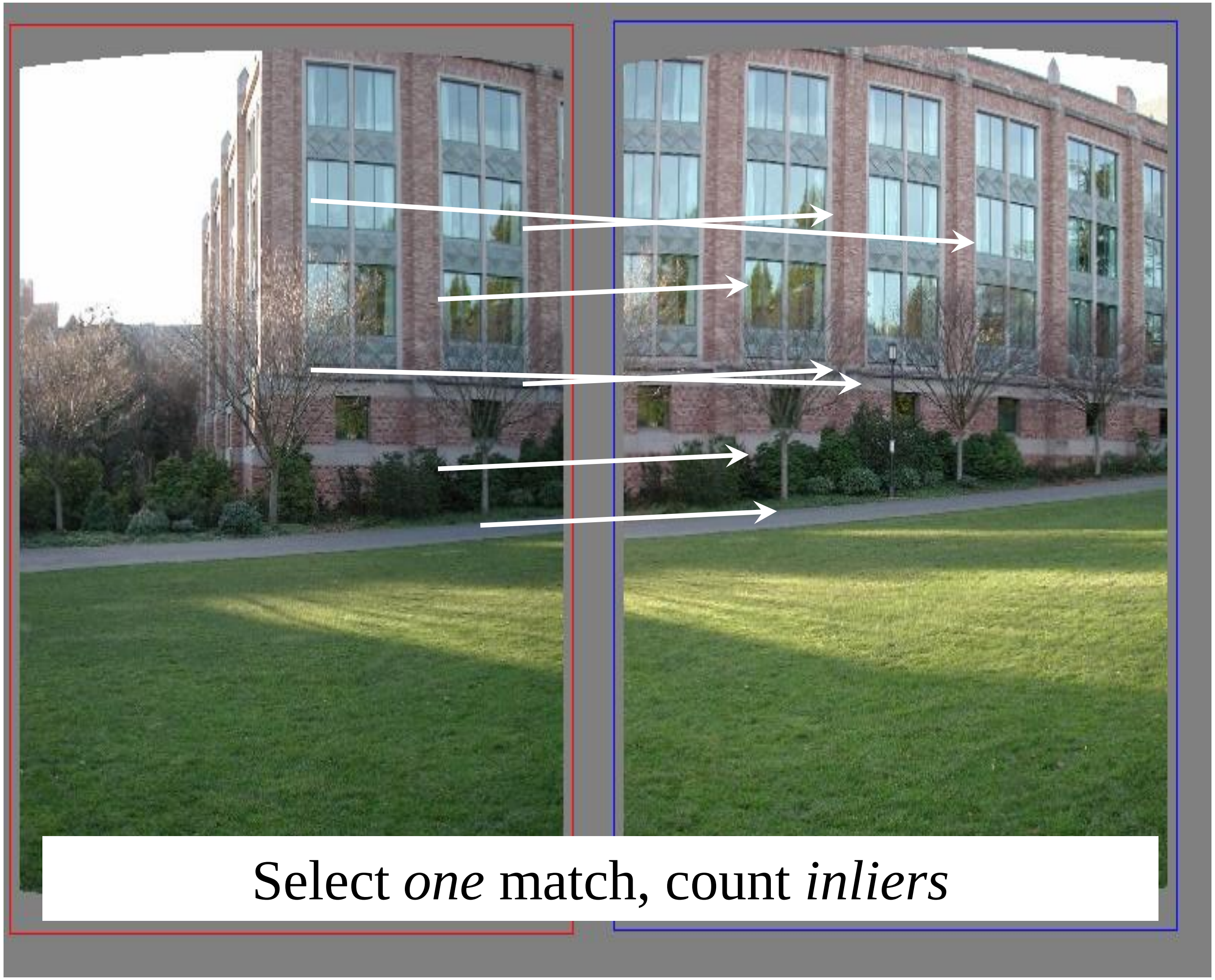
Source: Rick Szeliski

RANSAC example: Translation



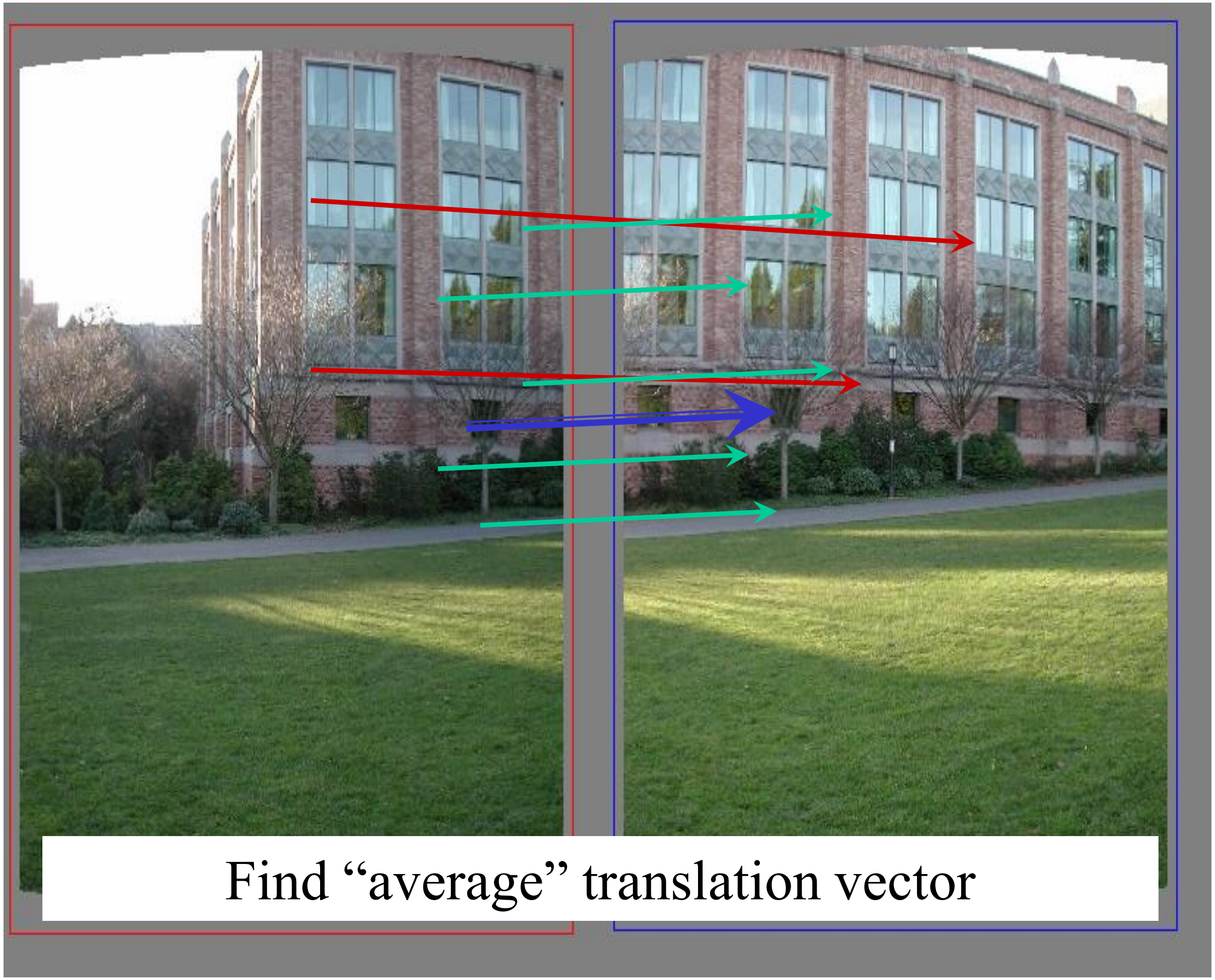
Select *one* match, count *inliers*

RANSAC example: Translation



Select *one* match, count *inliers*

RANSAC example: Translation



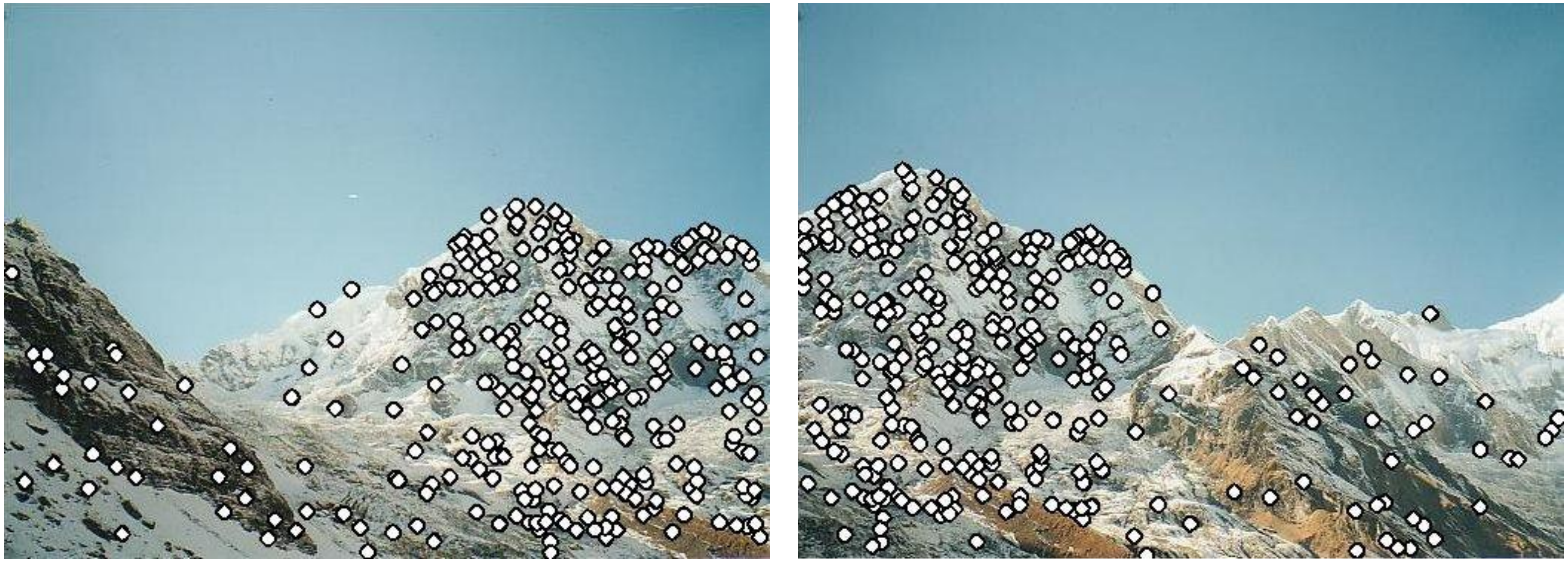
Find “average” translation vector

Feature-based alignment outline

Source: L. Lazebnik

Feature-based alignment outline



- Extract features

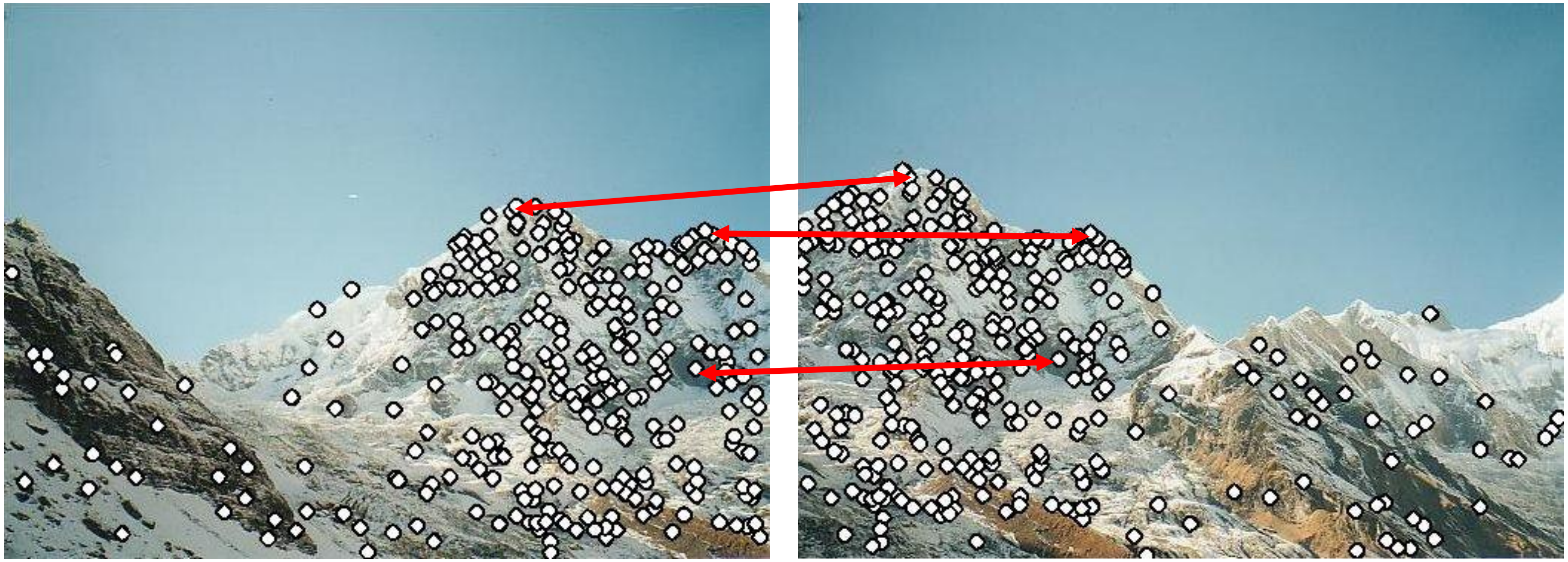
Source: L. Lazebnik

Feature-based alignment outline

- Extract features
- Compute *putative matches*

Source: L. Lazebnik

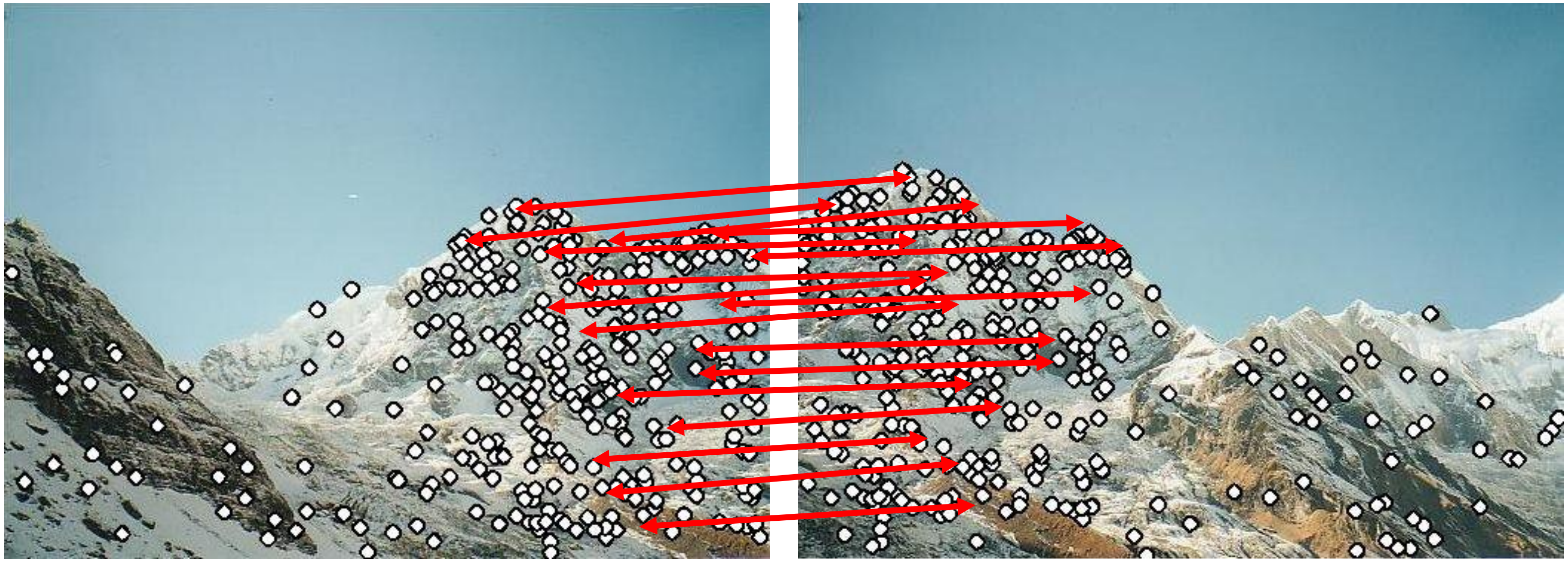
Feature-based alignment outline



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

Source: L. Lazebnik

Feature-based alignment outline



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

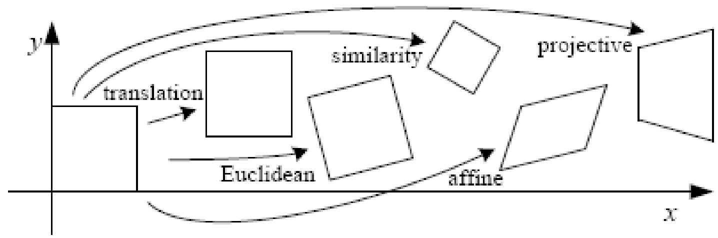
Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

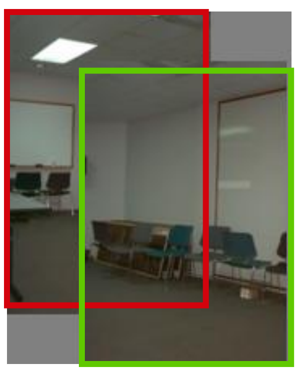
Source: L. Lazebnik

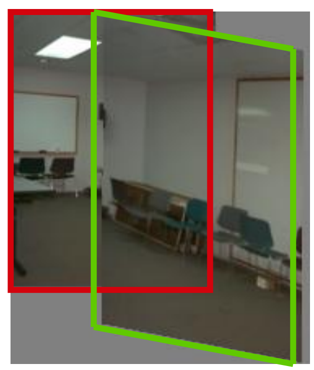
Towards large-scale mosaics…

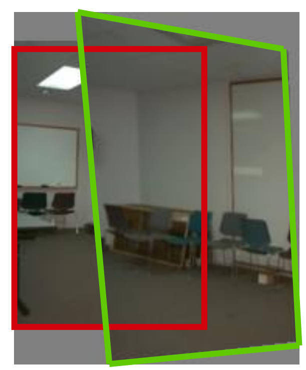
Motion models

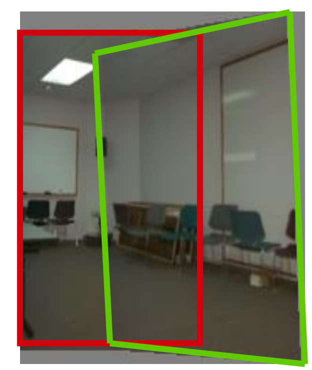


Translation | **Affine** | **Perspective** | **3D rotation**







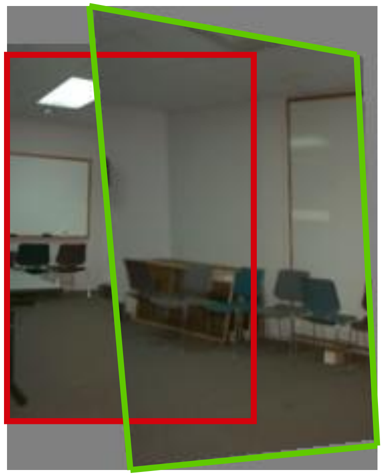


2 unknowns | **6 unknowns** | **8 unknowns** | **3 unknowns**

Szeliski

Plane perspective mosaics

- 8-parameter homographies
- Limitations:
  - local minima
  - slow convergence
  - difficult to control interactively

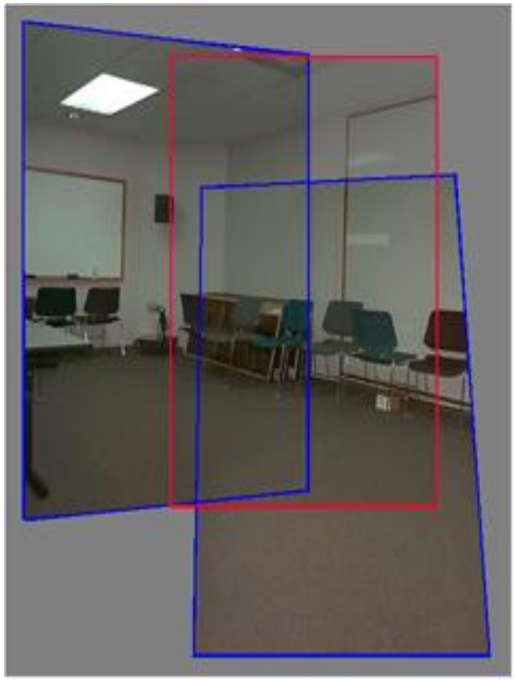


Szeliski

Rotational mosaics

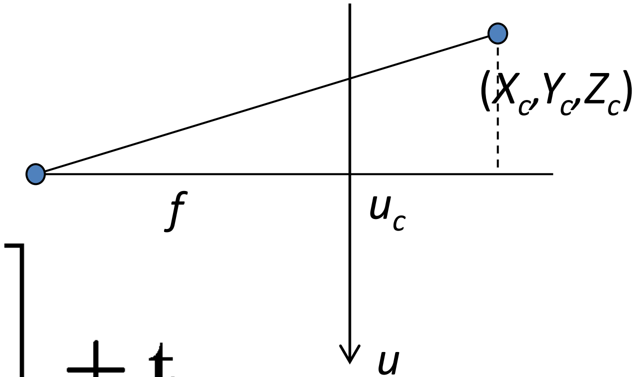
– Directly optimize rotation and focal length

– Advantages:
  - ability to build full-view panoramas
  - easier to control interactively
  - more stable and accurate estimates



Szeliski

3D → 2D Perspective Projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

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

Rotational mosaic

- Projection equations

1. Project from image to 3D ray

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

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

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

Establishing correspondences

1. 'Direct' method:
   - Use generalization of affine motion model [Szeliski & Shum '97]
2. Feature-based method
   - Extract features, match, find consistent *inliers* [Lowe ICCV'99; Schmid ICCV'98, Brown&Lowe ICCV'2003]
   - Compute $\boldsymbol{R}$ from correspondences (absolute orientation)

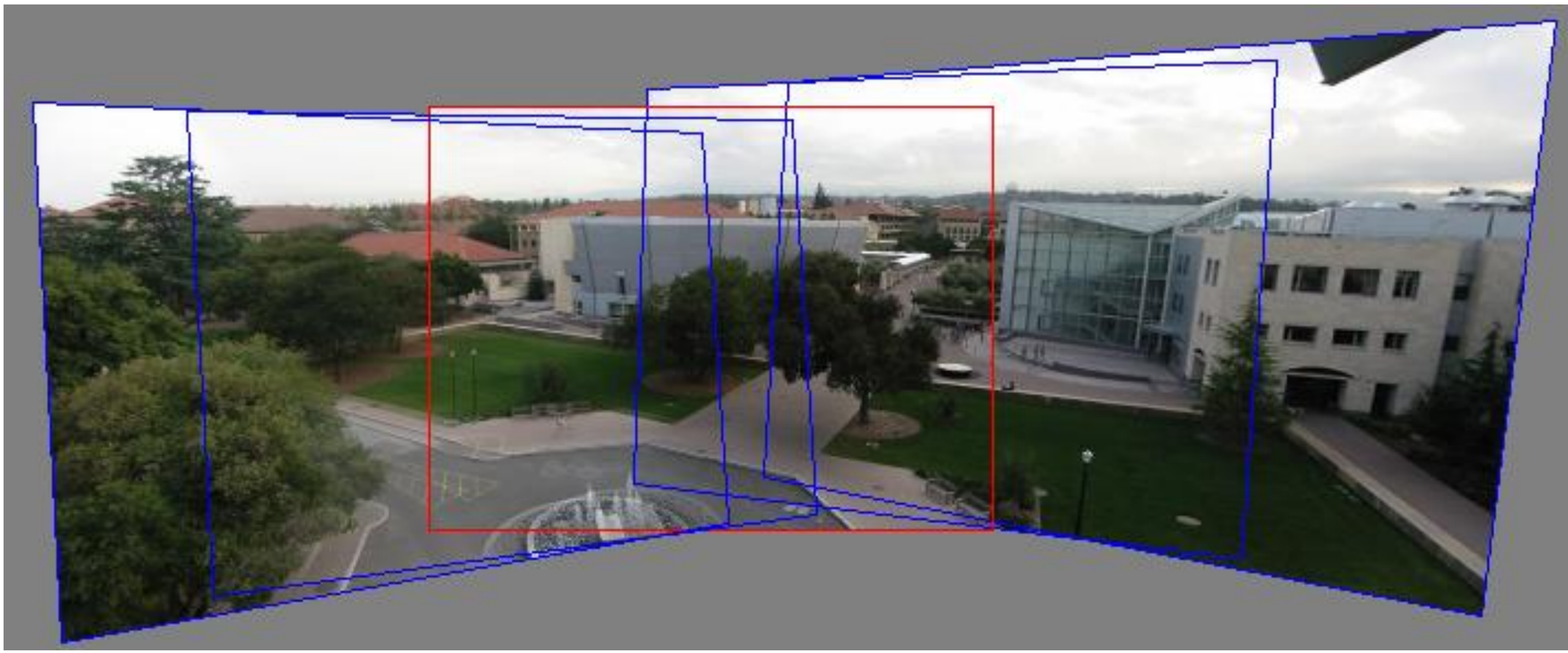
Absolute orientation

[Arun *et al.*, PAMI 1987] [Horn *et al.*, JOSA A 1988]
Procrustes Algorithm [Golub & VanLoan]

Given two sets of matching points, compute R

- $p_i' = \boldsymbol{R}\, p_i$ 3D rays
- $\boldsymbol{A} = \Sigma_{\mathbf{i}}\, p_i\, p_i'^T = \Sigma_{\mathbf{i}}\, p_i\, p_i^T\, \boldsymbol{R}^T = \boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{V}^T = (\boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{U}^T)\,\boldsymbol{R}^T$
- $\boldsymbol{V}^T = \boldsymbol{U}^T \boldsymbol{R}^T$
- $\boldsymbol{R} = \boldsymbol{V}\,\boldsymbol{U}^T$

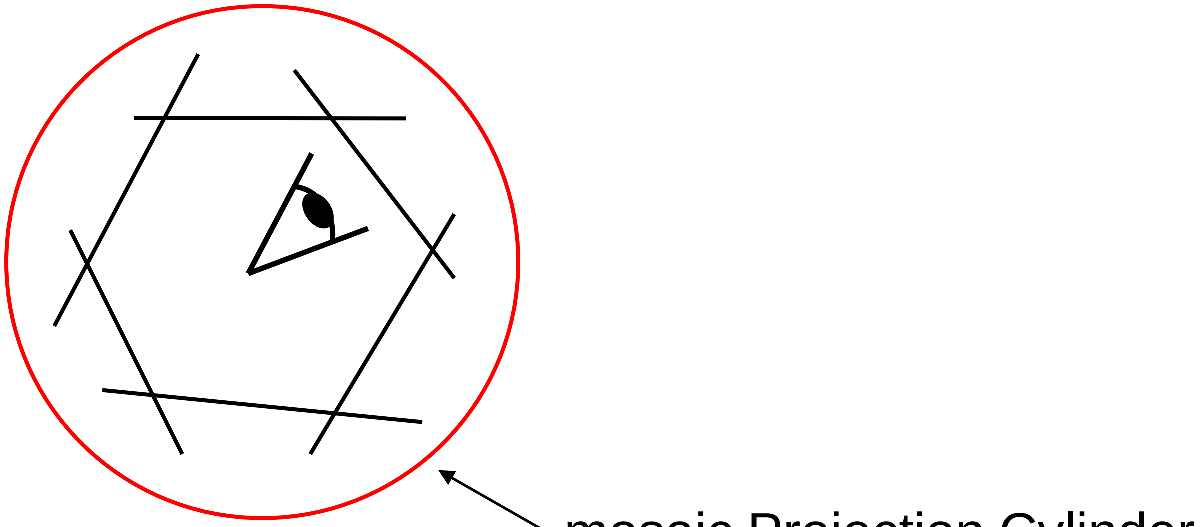
Stitching demo



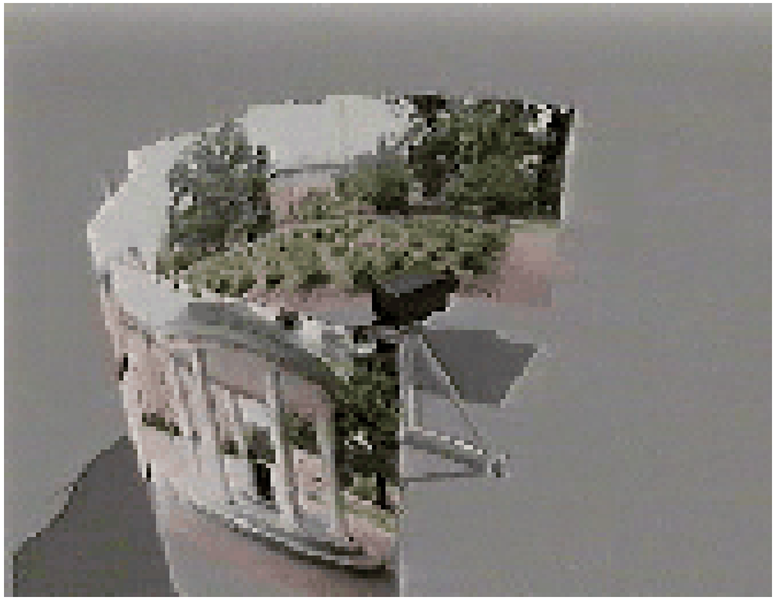
Szeliski

Panoramas

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



Cylindrical panoramas



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

Cylindrical Panoramas

- Map image to cylindrical or spherical coordinates
  - need *known* focal length

Image 384x300

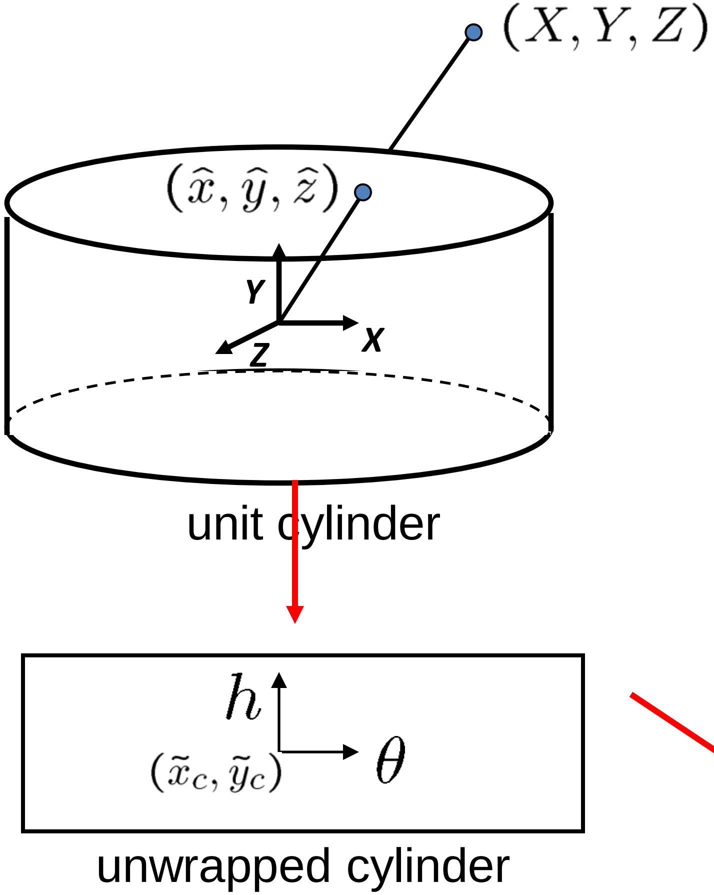
f = 180 (pixels)

f = 280

f = 380

Szeliski

Cylindrical projection



– Map 3D point (X,Y,Z) onto cylinder

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

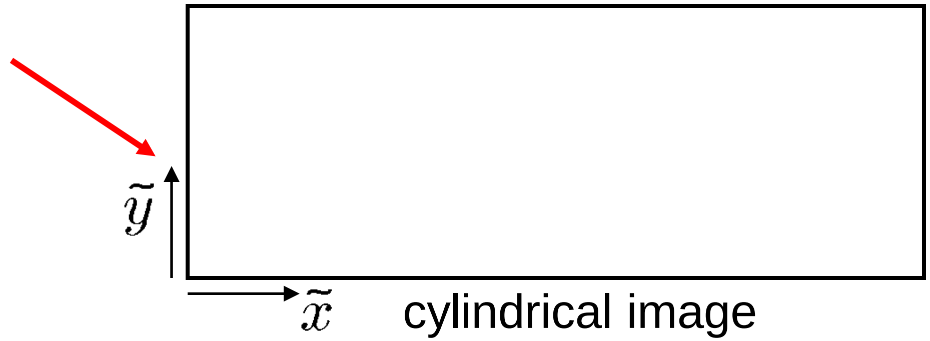
- Convert to cylindrical coordinates

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

- Convert to cylindrical image coordinates

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

  – s defines size of the final image

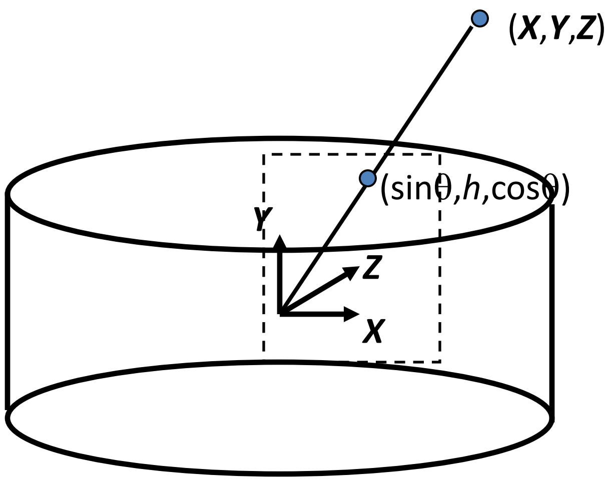


cylindrical image

Szeliski

Cylindrical warping

- Given focal length *f* and image center (x_c, y_c)

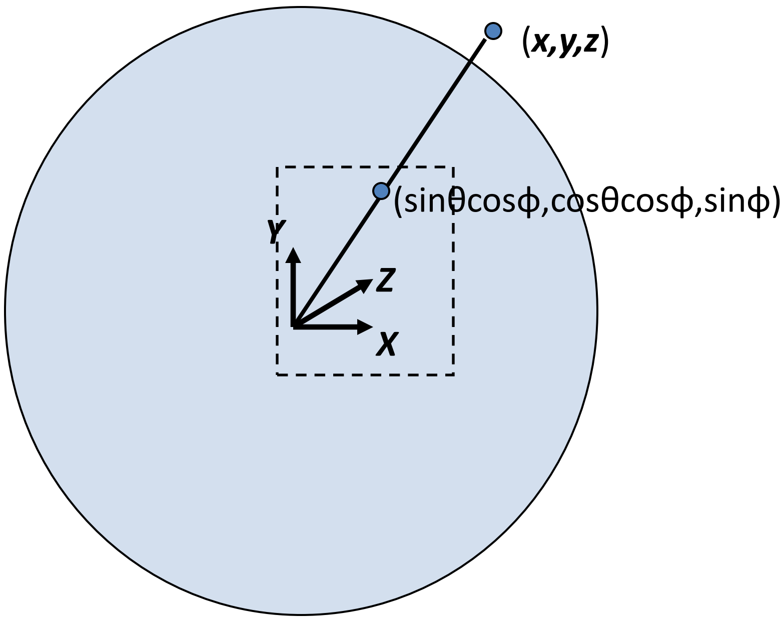


$$
\begin{aligned}
\theta &= (x_{cyl} - x_c)/f \\
h &= (y_{cyl} - y_c)/f \\
\hat{x} &= \sin\theta \\
\hat{y} &= h \\
\hat{z} &= \cos\theta \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c
\end{aligned}
$$

Szeliski

Spherical warping

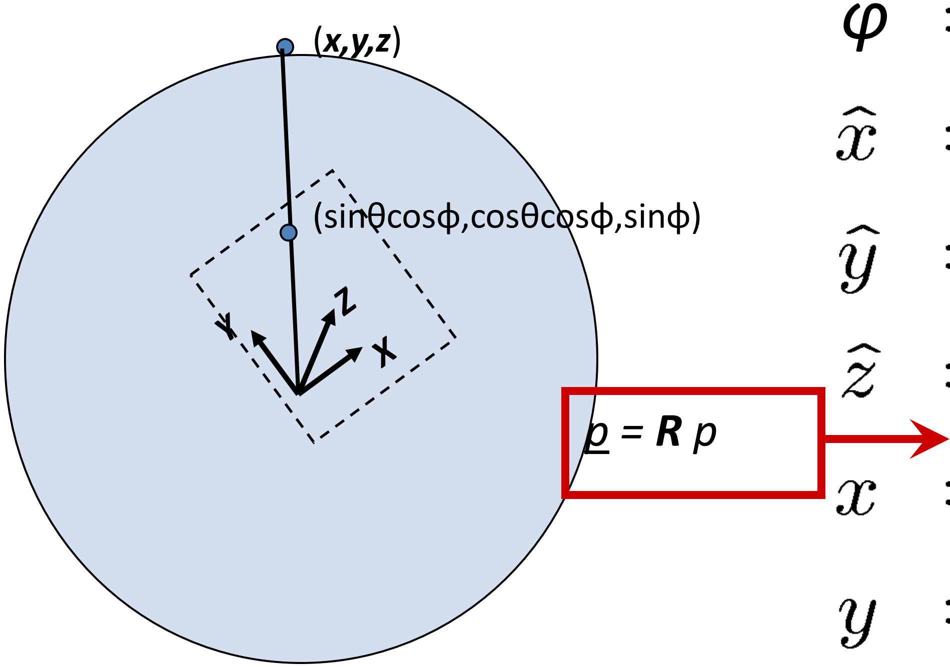
- Given focal length *f* and image center (x_c, y_c)



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

3D rotation

•Rotate image before placing on unrolled sphere



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

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

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

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

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

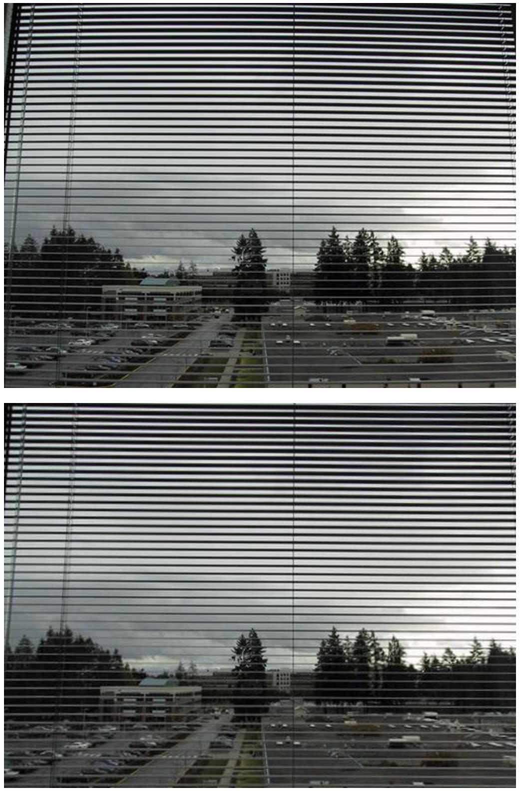
$$x = f\hat{x}/\hat{z} + x_c$$

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

Szeliski

Radial distortion

- Correct for "bending" in wide field of view lenses



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

Szeliski

Fisheye lens

- 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

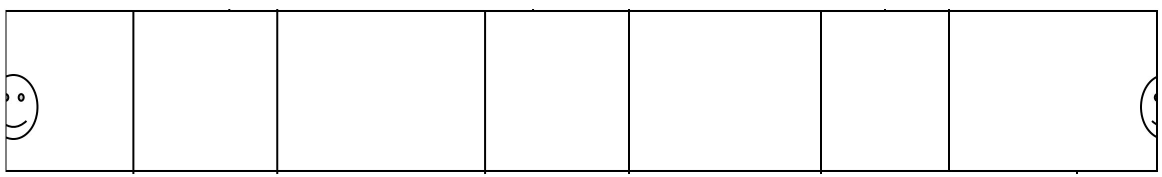
$$x' = s\phi\cos\theta = s\frac{x}{r}\tan^{-1}\frac{r}{z},$$

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

Image Stitching

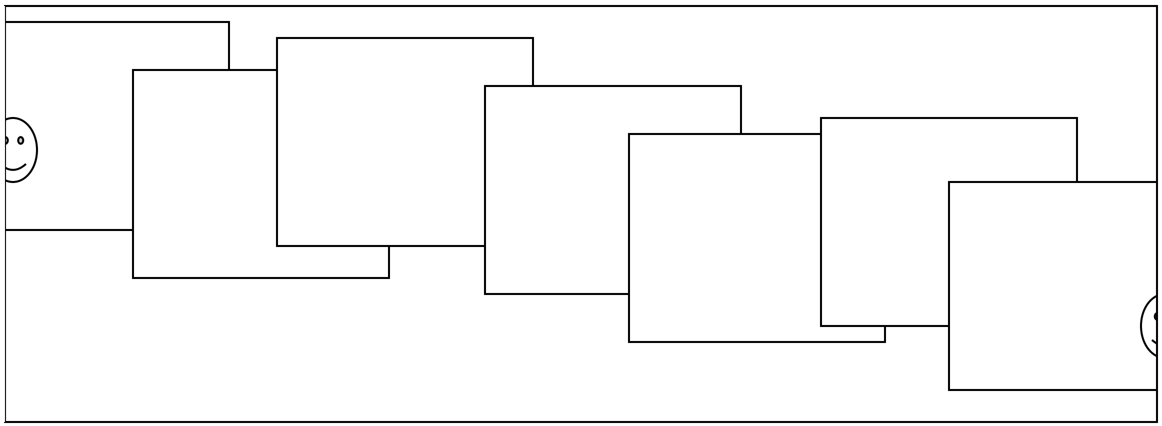
1. Align the images over each other
   – camera pan ↔ translation on cylinder
2. Blend the images together

Assembling the panorama



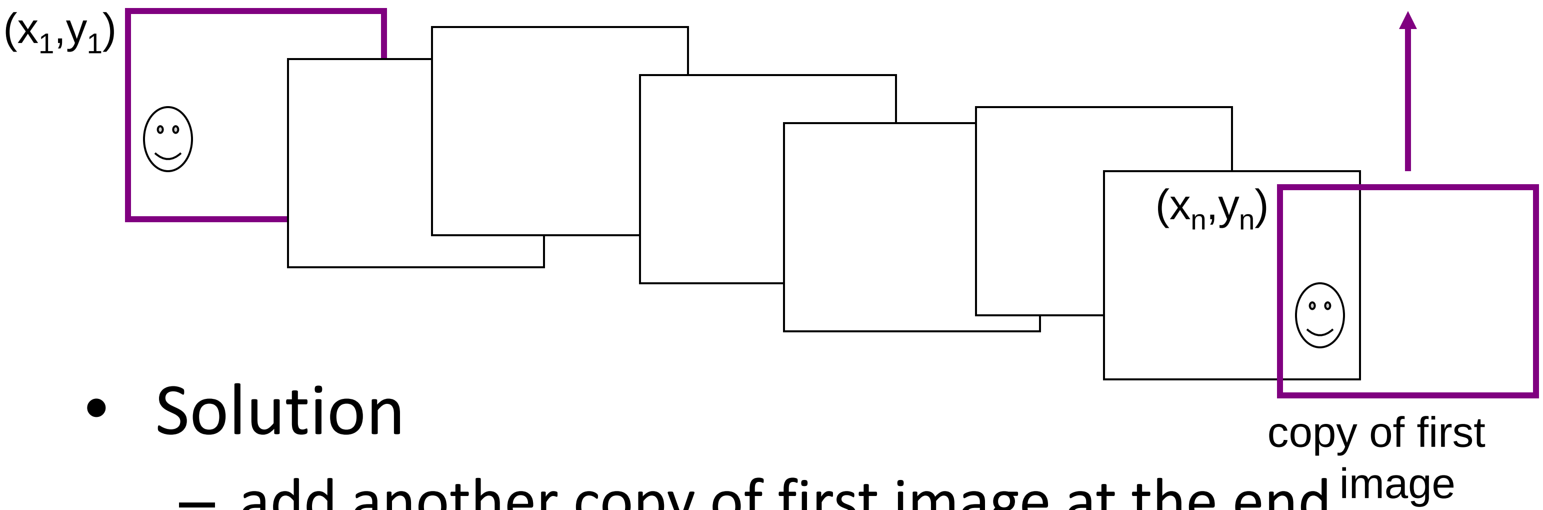
- Stitch pairs together, blend, then crop

Problem: Drift



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

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 (360° spherical) panoramas

Full-view Panorama

Szeliski

Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use direct alignment (patch centers) or feature based
- *Infer* overlaps based on previous matches (incremental)
- Optionally *discover* which images overlap other images using feature selection (RANSAC)

Bundle adjustment formulations

Confidence / uncertainty of point i in image j

All pairs optimization:

$$E_{\text{all-pairs-2D}} = \sum_i \sum_{jk} c_{ij} c_{ik} \|\tilde{\boldsymbol{x}}_{ik}(\hat{\boldsymbol{x}}_{ij}; \boldsymbol{R}_j, f_j, \boldsymbol{R}_k, f_k) - \hat{\boldsymbol{x}}_{ik}\|^2, \quad (9.29)$$

Map 2D point i in image j to 2D point in image k

Full bundle adjustment, using 3-D point positions $\{\boldsymbol{x}_i\}$

$$E_{\text{BA-2D}} = \sum_i \sum_j c_{ij} \|\tilde{\boldsymbol{x}}_{ij}(\boldsymbol{x}_i; \boldsymbol{R}_j, f_j) - \hat{\boldsymbol{x}}_{ij}\|^2, \quad (9.30)$$

Map 3D point i in to 2D point in image i

Bundle adjustment using 3-D ray:

$$E_{\text{BA-3D}} = \sum_i \sum_j c_{ij} \|\tilde{\boldsymbol{x}}_i(\hat{\boldsymbol{x}}_{ij}; \boldsymbol{R}_j, f_j) - \boldsymbol{x}_i\|^2, \quad (9.31)$$

3-D ray from point i

All-pairs 3-D ray formulation:

$$E_{\text{all-pairs-3D}} = \sum_i \sum_{jk} c_{ij} c_{ik} \|\tilde{\boldsymbol{x}}_i(\hat{\boldsymbol{x}}_{ij}; \boldsymbol{R}_j, f_j) - \tilde{\boldsymbol{x}}_i(\hat{\boldsymbol{x}}_{ik}; \boldsymbol{R}_k, f_k)\|^2. \quad (9.32)$$

3-D ray from points i and j

Projected point

$$\tilde{\boldsymbol{x}}_{ij} \sim \boldsymbol{K}_j \boldsymbol{R}_j \boldsymbol{x}_i \ \text{ and } \ \boldsymbol{x}_i \sim \boldsymbol{R}_j^{-1} \boldsymbol{K}_j^{-1} \tilde{\boldsymbol{x}}_{ij},$$

3-D ray from point

Summary

- Image alignment is very essential in many computer vision applications
- Selection of appropriate image transformations is very important and usually depends on the application domains.
- Image mosaicing/stitching is very common in many applications, such as panoramas, medical imaging, etc.