CSC 589
Lecture 22 Image Alignment and least square methods

Bei Xiao

American University

April 13

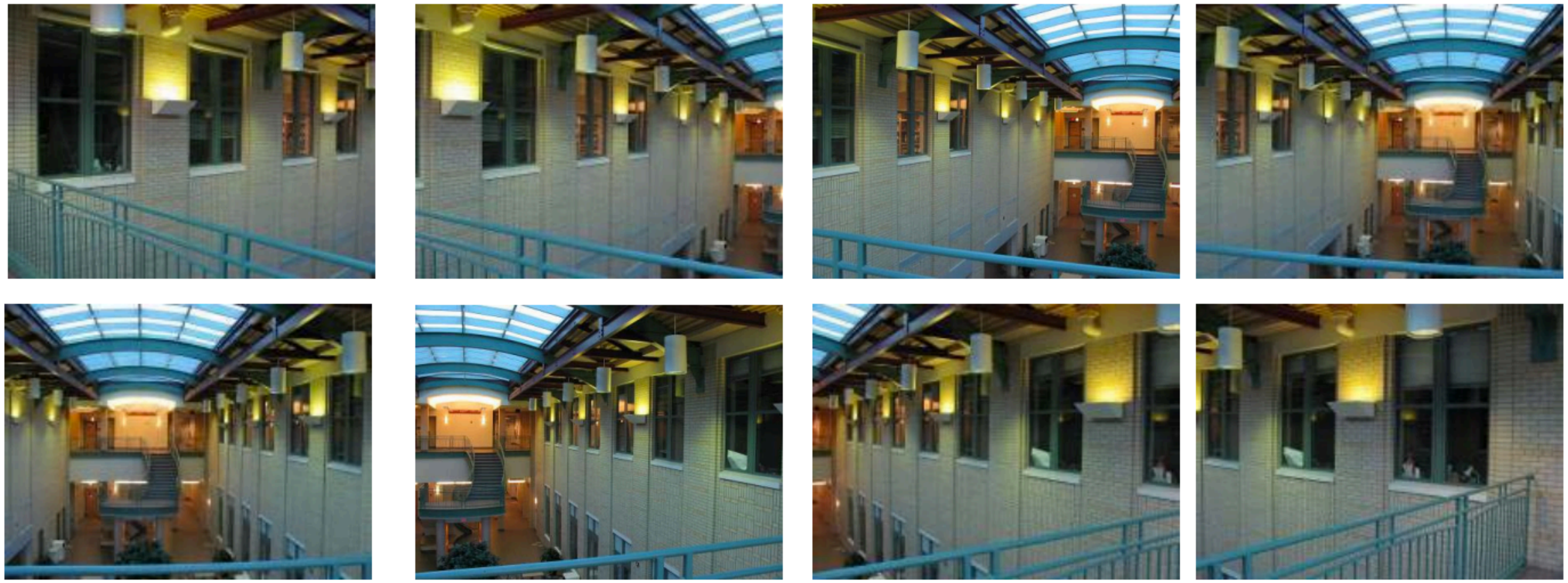
Final project

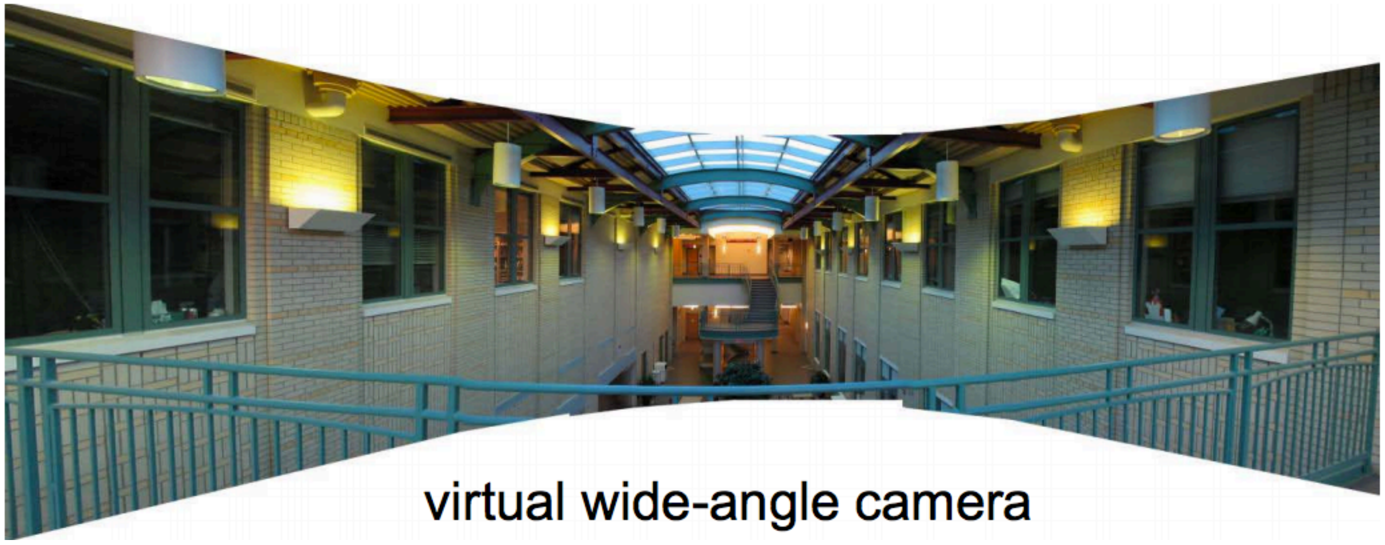
- Due May 4th.
- The final websites will be shared among students!!
- Presentations will be recorded as a 5 mins you-tube video! We will learn how to do this.
- Cite the papers you used and the methods you implemented!! Do not copy codes form the internet!!! If you must copy one or two lines, cite the resources. Plagiarism is easy to detect these days!!
- Team members clearly specify who contributed what in your final report.

Reading

- Appendix A.2, 6.1

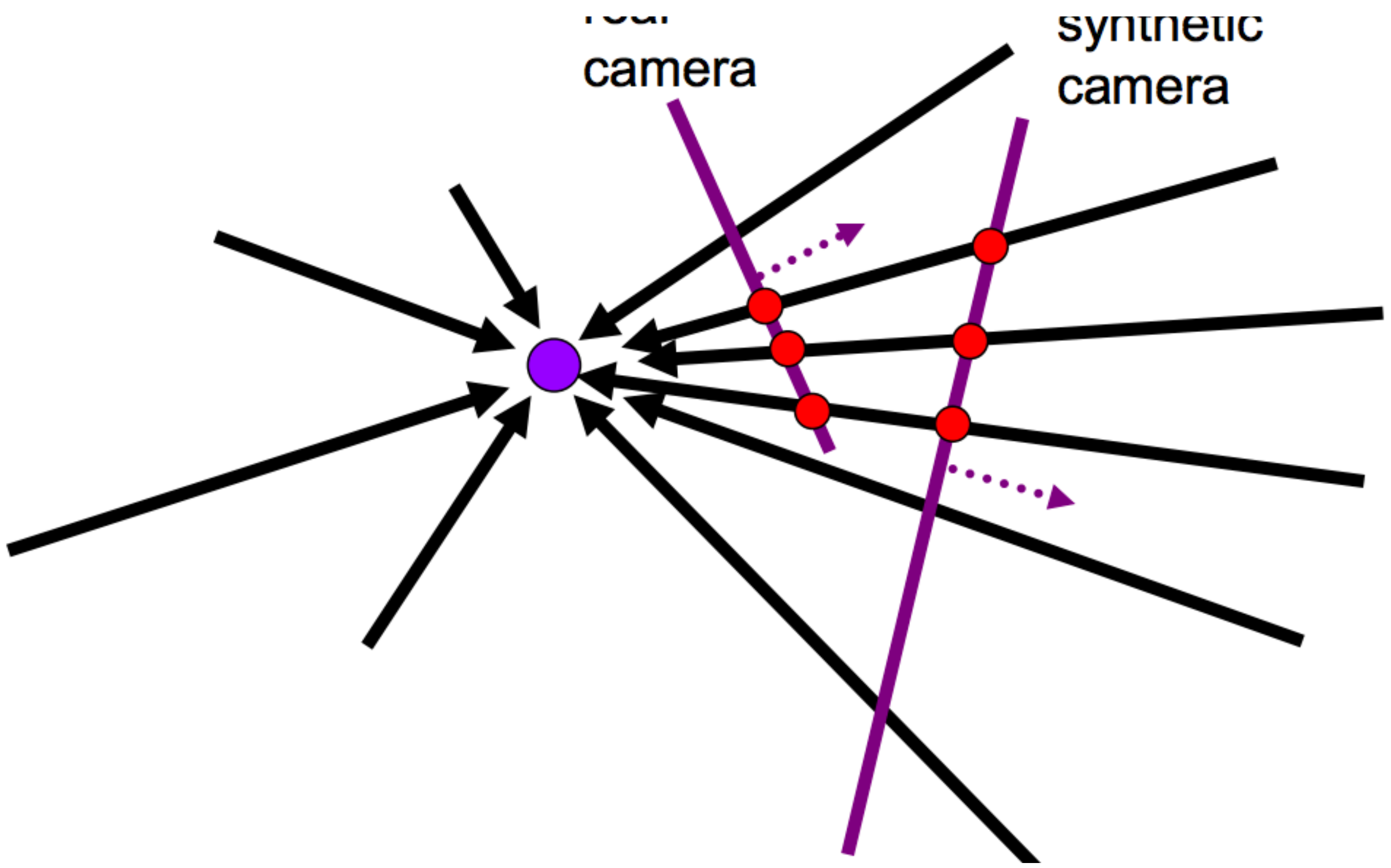
Mosaics: stitching images together





virtual wide-angle camera

A pencil of rays contains all views



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

How to do it?

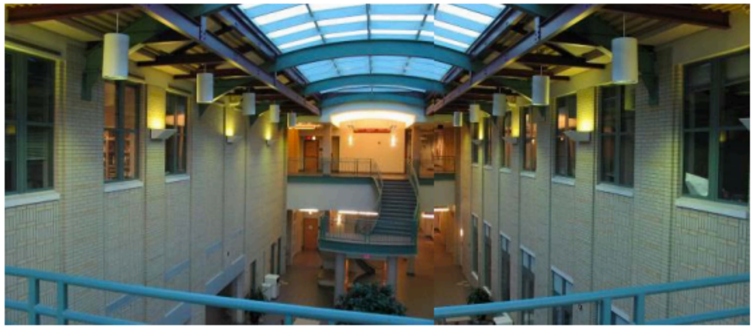
Basic Procedure:

- Take a sequence of images from the same position
- Rotate the camera about its optical center
- Compute transformation between second image and first
- Transform the second image overlap the first
- Blend the two together to create a mosaic
- If there are more images, repeat

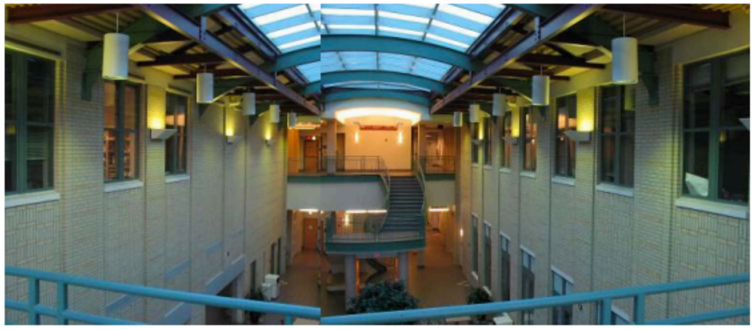
- But wait, why should this work at all?
  - What about 3D geometry of the scenes?
  - Why aren’t we use it at all?

Aligning images

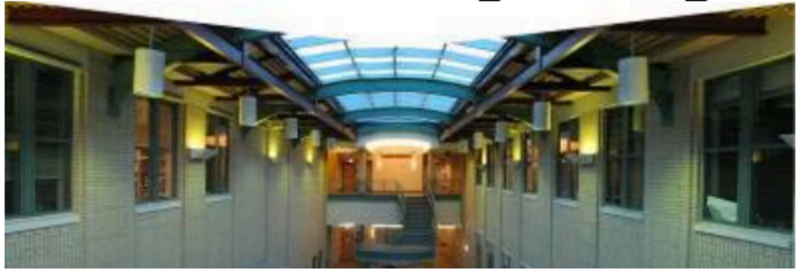
left on top



right on top

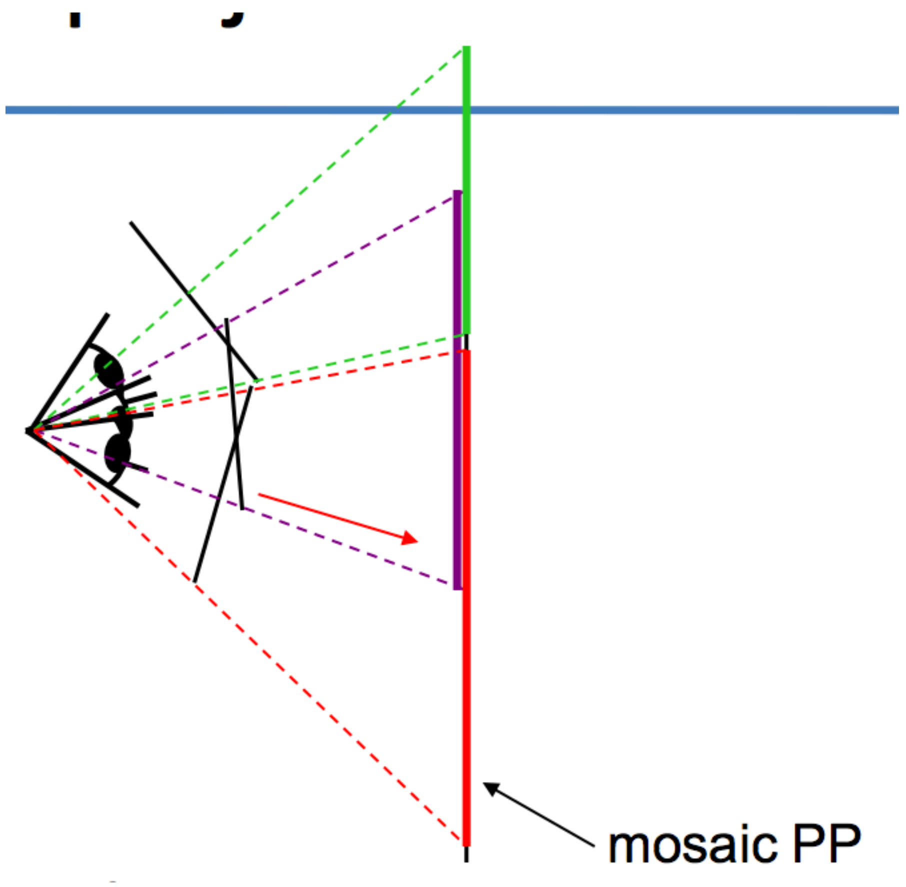


Translations are not enough to align the images



http://www.inf.ethz.ch/personal/pomarc/courses/CompPhoto/cpv03.pdf

Image reprojection



The mosaic has a natural Interpretation in 3D

- The images are repojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a synthetic wide-angel camera

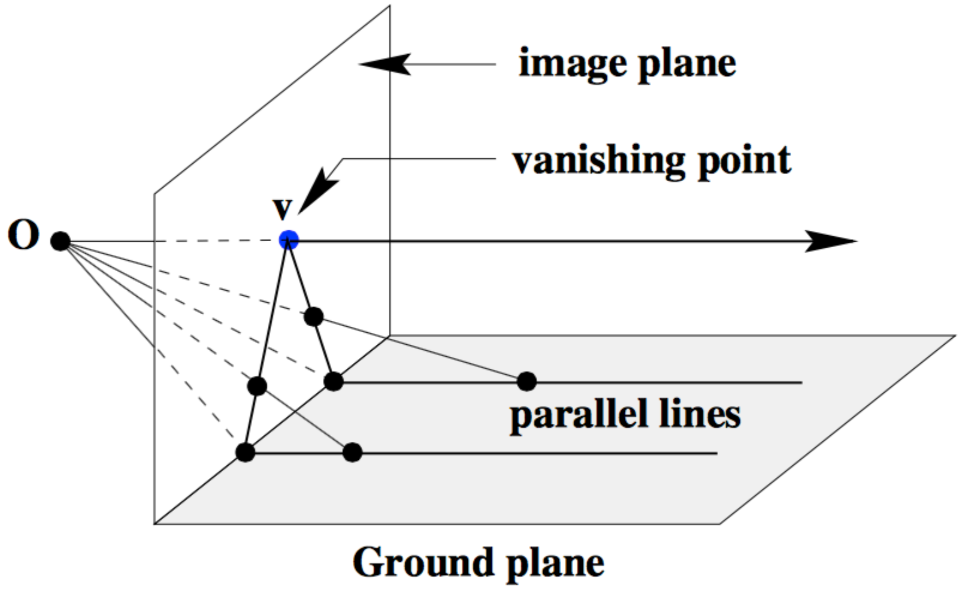
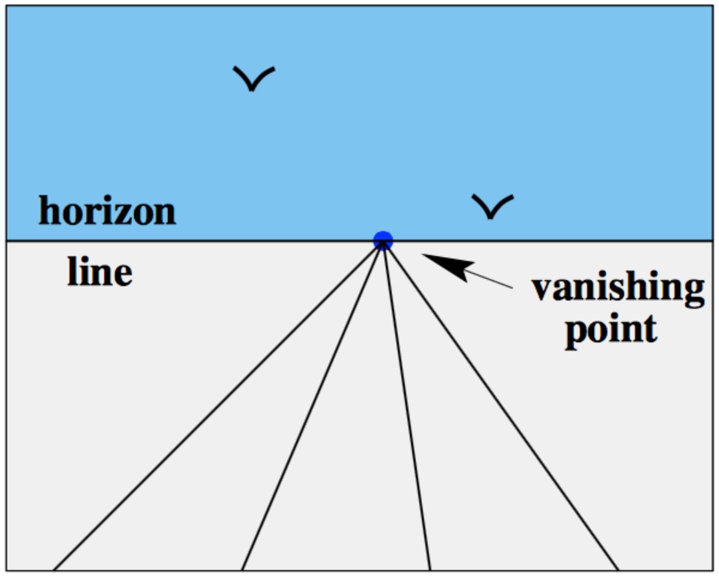


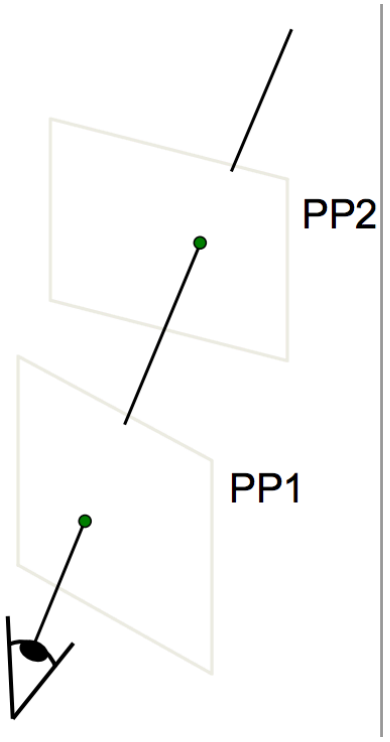
image plane
vanishing point
v
O
parallel lines
Ground plane


horizon
line
vanishing
point

Homography

- A projective-mapping between any two PPs with the same center of projection

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



Properties of projective transformations:

Origin does not necessarily map to origin

Lines map to lines

Parallel lines do not necessarily remain parallel

Ratios are not preserved

Closed under composition

Common problems in vision

Fitting: find the parameters of a model that best fit the data

Alignment: find the parameters of the transformation that best align matched points

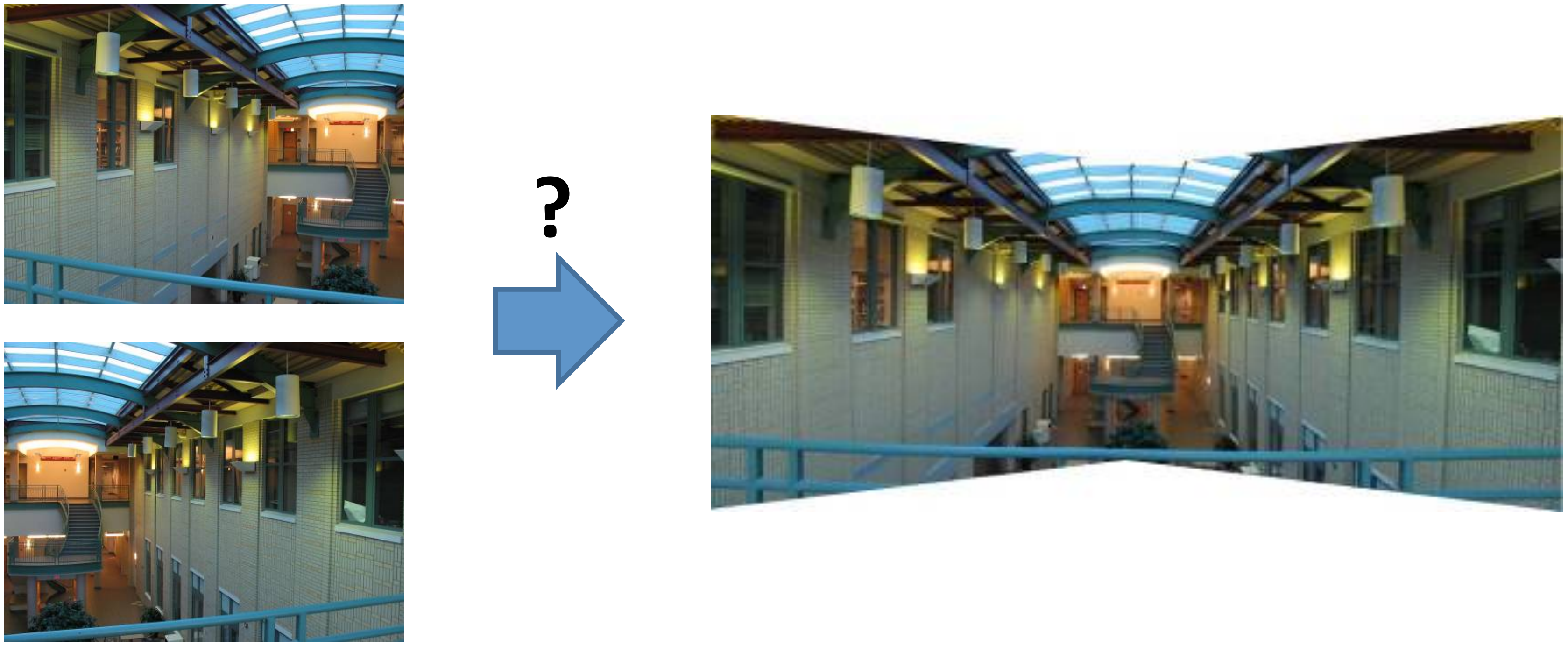
Alignment

- Alignment: find parameters of model that maps one set of points to another

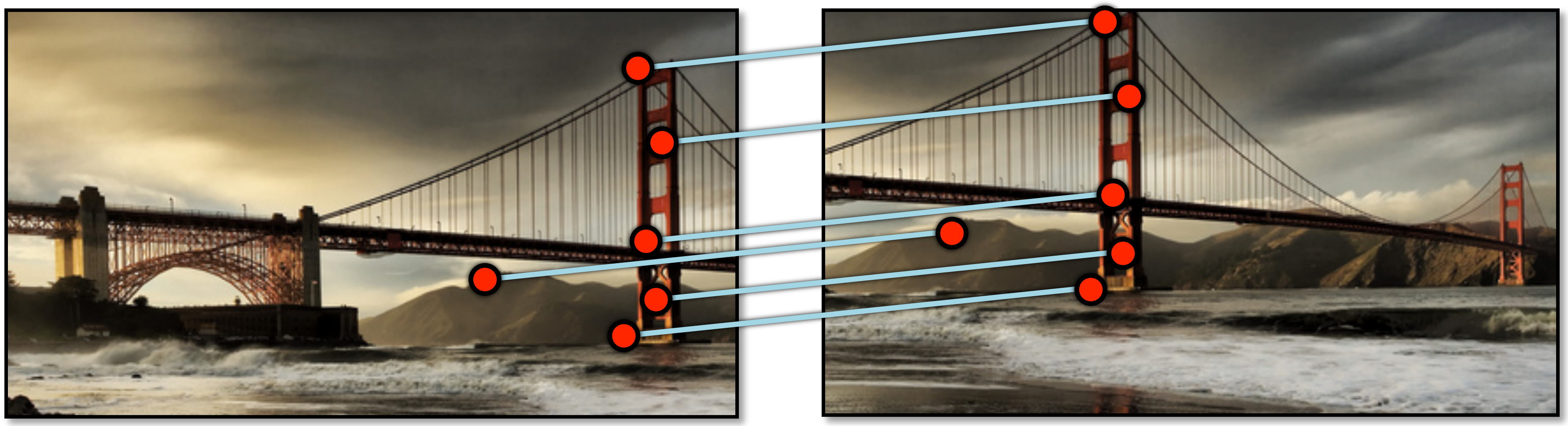
- Typically want to solve for a global transformation that accounts for *most* true correspondences

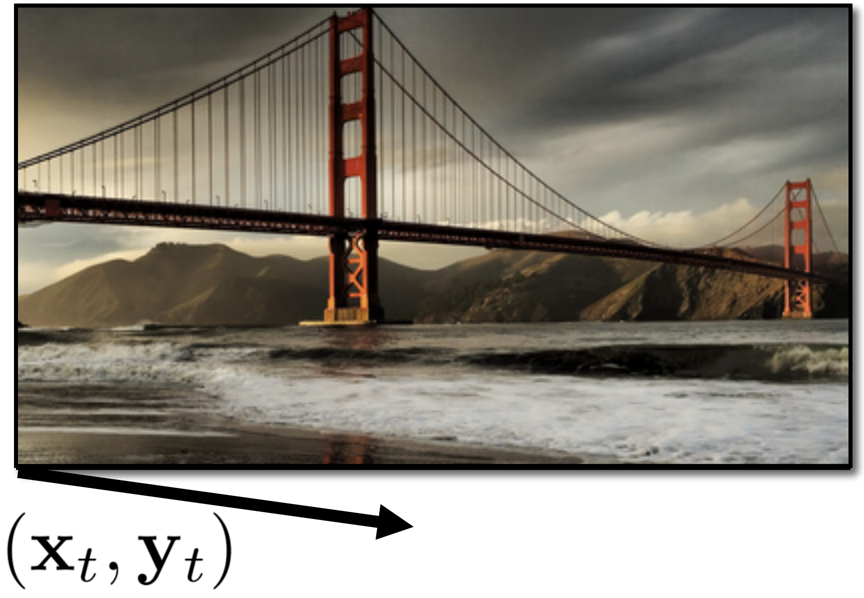
- Difficulties
  - Noise (typically 1-3 pixels)
  - Outliers (often 50%)

Computing transformations



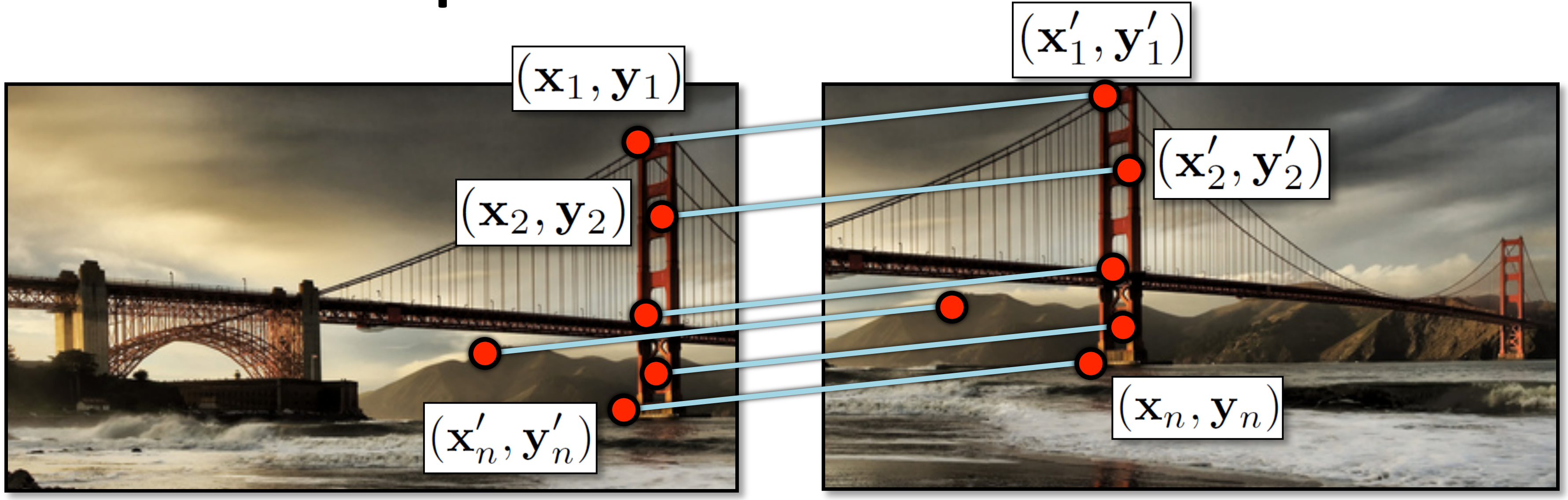
Simple case: translations





How do we solve for $(\mathbf{x}_t, \mathbf{y}_t)$**?**

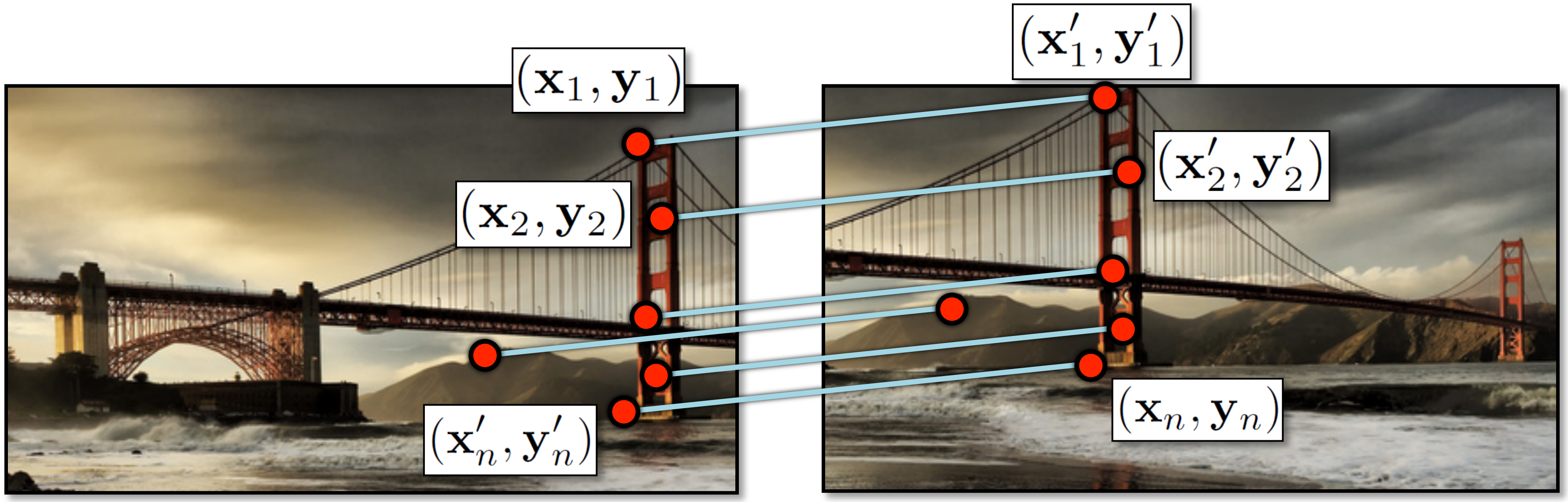
Simple case: translations


$(\mathbf{x}_1, \mathbf{y}_1)$
$(\mathbf{x}_2, \mathbf{y}_2)$
$(\mathbf{x}'_n, \mathbf{y}'_n)$
$(\mathbf{x}'_1, \mathbf{y}'_1)$
$(\mathbf{x}'_2, \mathbf{y}'_2)$
$(\mathbf{x}_n, \mathbf{y}_n)$

Displacement of match *i* = $(\mathbf{x}'_i - \mathbf{x}_i, \mathbf{y}'_i - \mathbf{y}_i)$

$$(\mathbf{x}_t, \mathbf{y}_t) = \left(\frac{1}{n} \sum_{i=1}^{n} \mathbf{x}'_i - \mathbf{x}_i, \frac{1}{n} \sum_{i=1}^{n} \mathbf{y}'_i - \mathbf{y}_i \right)$$

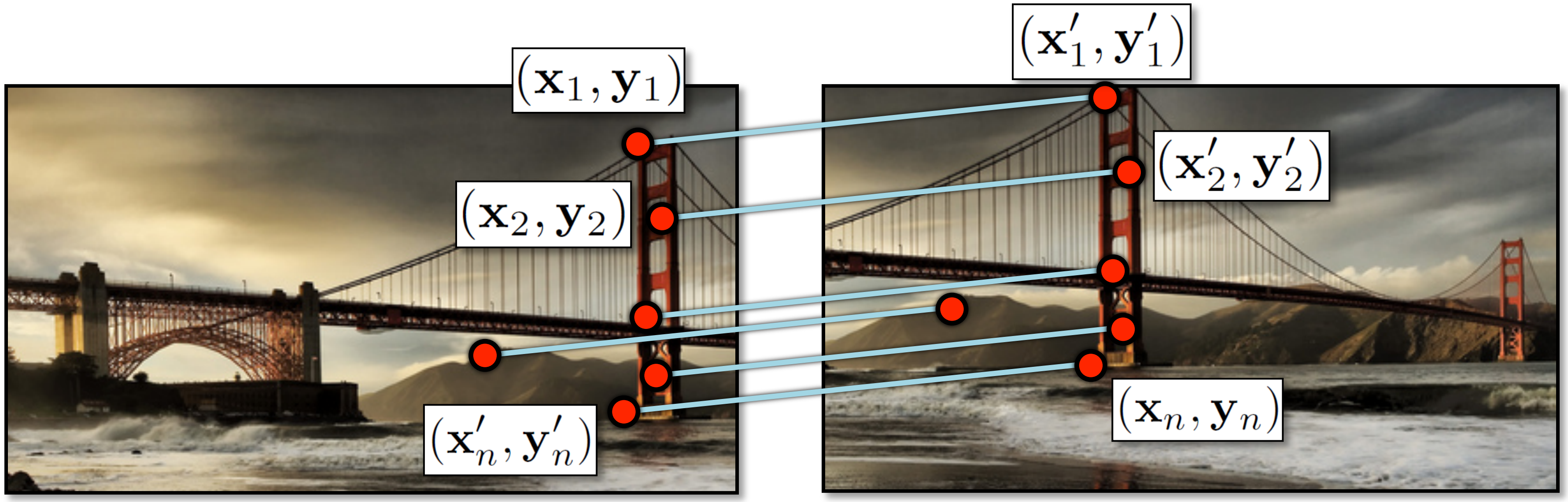
Another view


$(\mathbf{x}_1, \mathbf{y}_1)$
$(\mathbf{x}_2, \mathbf{y}_2)$
$(\mathbf{x}'_n, \mathbf{y}'_n)$
$(\mathbf{x}'_1, \mathbf{y}'_1)$
$(\mathbf{x}'_2, \mathbf{y}'_2)$
$(\mathbf{x}_n, \mathbf{y}_n)$

$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$
$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

Another view



$(\mathbf{x}_1, \mathbf{y}_1)$
$(\mathbf{x}_2, \mathbf{y}_2)$
$(\mathbf{x}'_n, \mathbf{y}'_n)$
$(\mathbf{x}'_1, \mathbf{y}'_1)$
$(\mathbf{x}'_2, \mathbf{y}'_2)$
$(\mathbf{x}_n, \mathbf{y}_n)$

$$\mathbf{x}_i + \mathbf{x_t} = \mathbf{x}'_i$$

$$\mathbf{y}_i + \mathbf{y_t} = \mathbf{y}'_i$$

- Problem: more equations than unknowns
  - "Overdetermined" system of equations
  - We will find the *least squares* solution

Least squares formulation

- For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$\begin{aligned} \mathbf{x}_i + \mathbf{x_t} &= \mathbf{x}_i' \\ \mathbf{y}_i + \mathbf{y_t} &= \mathbf{y}_i' \end{aligned}$$

- we define the *residuals* as

$$\begin{aligned} r_{\mathbf{x}_i}(\mathbf{x}_t) &= (\mathbf{x}_i + \mathbf{x}_t) - \mathbf{x}_i' \\ r_{\mathbf{y}_i}(\mathbf{y}_t) &= (\mathbf{y}_i + \mathbf{y}_t) - \mathbf{y}_i' \end{aligned}$$

Least squares formulation

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^{n} \left(r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2 \right)$$

- "Least squares" solution
- For translations, is equal to mean displacement

Least squares formulation

- Can also write as a matrix equation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ & \vdots & \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\underset{2n \times 2}{\mathbf{A}} \quad \underset{2 \times 1}{\mathbf{t}} = \underset{2n \times 1}{\mathbf{b}}$$

Matrix Product mini-review

- http://en.wikipedia.org/wiki/Matrix_multiplication

- System of linear equations

  http://en.wikipedia.org/wiki/System_of_linear_equations

Matrix manipulations with Numpy

- http://www.python-course.eu/matrix_arithmetic.php
- https://jameshensman.wordpress.com/2010/06/14/multiple-matrix-multiplication-in-numpy/

Least squares

$$\mathbf{A}\mathbf{t} = \mathbf{b}$$

- Find **t** that minimizes

$$||\mathbf{A}\mathbf{t} - \mathbf{b}||^2$$

Least squares: find t to minimize

$$||\mathbf{At} - \mathbf{b}||^2$$

$$t^T(A^TA)t - 2t^T(A^Tb) + ||b||^2$$

- To solve, form the *normal equations*
  - Differentiate and equate to 0 to minimize

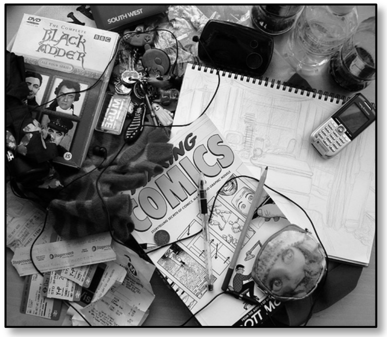
$$\mathbf{A}^{\mathrm{T}}\mathbf{At} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{t} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

MAKING
COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
FROM THE AUTHOR OF UNDERSTANDING COMICS
SCOTT McCLOUD



- How many unknowns?
- How many equations per match?
- How many matches do we need?

Affine transformations

- Residuals:

$$\begin{aligned} r_{x_i}(a,b,c,d,e,f) &= (ax_i + by_i + c) - x'_i \\ r_{y_i}(a,b,c,d,e,f) &= (dx_i + ey_i + f) - y'_i \end{aligned}$$

- Cost function:

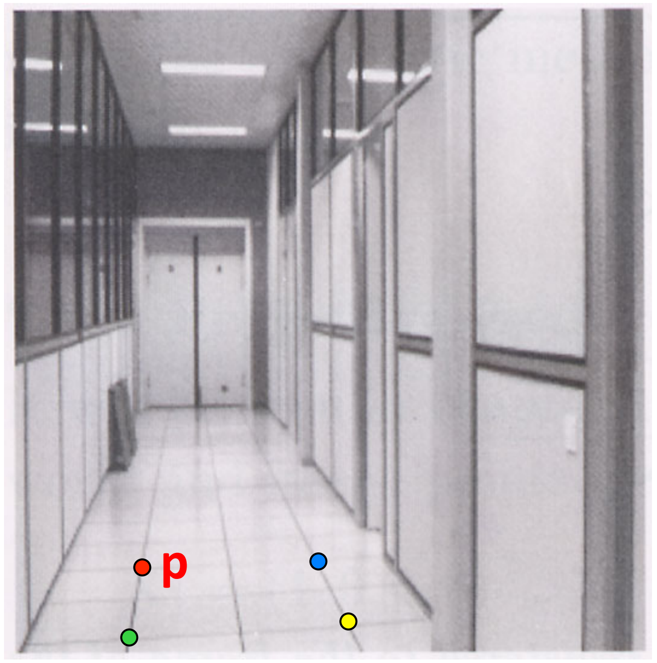
$$C(a,b,c,d,e,f) = \sum_{i=1}^{n} \left(r_{x_i}(a,b,c,d,e,f)^2 + r_{y_i}(a,b,c,d,e,f)^2 \right)$$

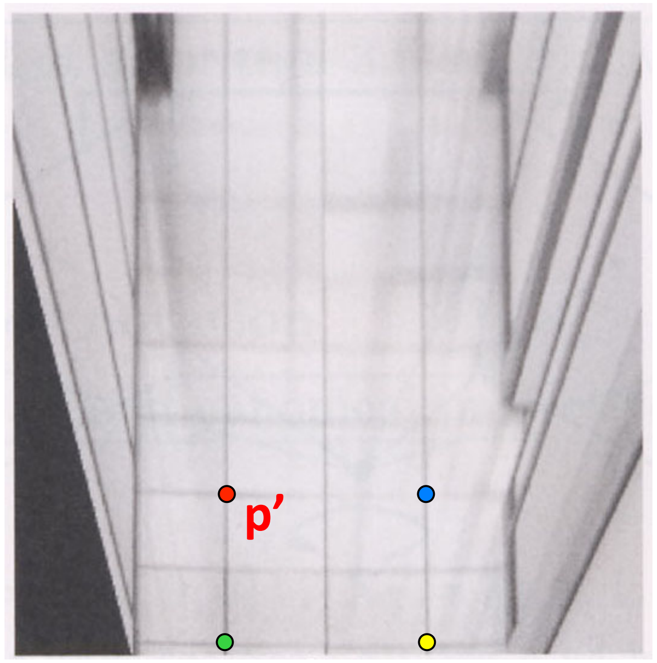
Affine transformations

- Matrix form

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 6}} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}}_{\substack{\mathbf{t} \\ 6 \times 1}} = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}}_{\substack{\mathbf{b} \\ 2n \times 1}}$$

Homographies





To unwarp (rectify) an image

- solve for homography **H** given **p** and **p’**
- solve equations of the form: **p’** = **Hp**

Alternate formulation for homographies

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where the length of the vector [h_{00} h_{01} … h_{22}] is 1

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

Linear or non-linear?

Solving for homographies

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

Solving for homographies

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

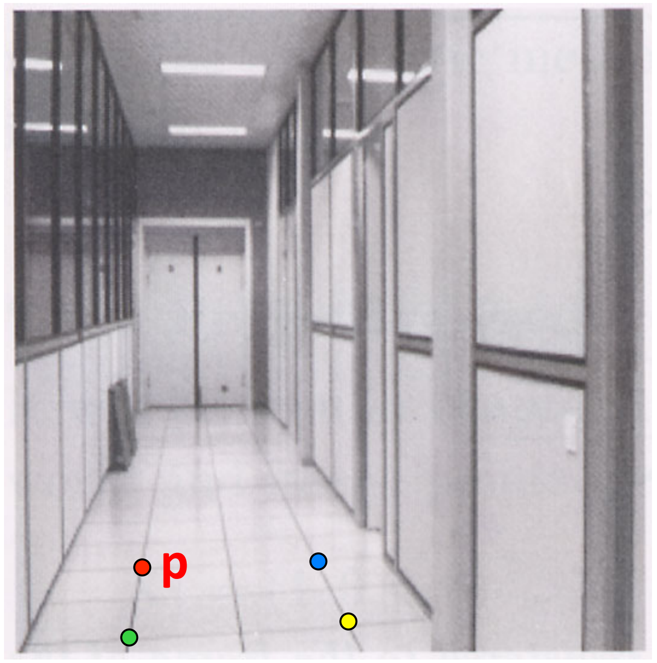
Solving for homographies

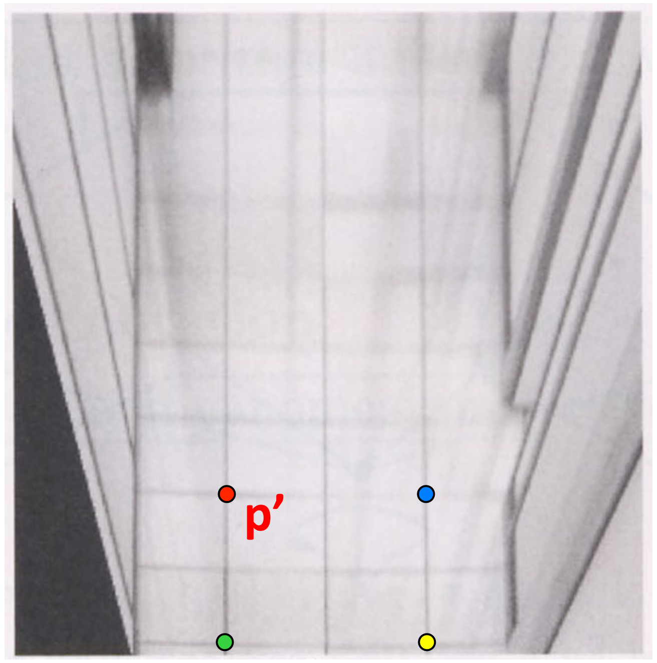
$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

Defines a least squares problem: $\text{minimize } \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$

Homographies





To unwarp (rectify) an image

- solve for homography **H** given **p** and **p’**
- solve equations of the form: **p’** = **Hp**
  - linear in unknowns: coefficients of **H**
  - H is defined up to an arbitrary scale factor
  - how many points are necessary to solve for **H**?

Solving for homographies

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

Defines a least squares problem: minimize $\|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

- Since $\mathbf{h}$ is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Recap: Two Common Optimization Problems

Problem statement

$$\text{minimize } \|\mathbf{Ax} - \mathbf{b}\|^2$$

least squares solution to $\mathbf{Ax} = \mathbf{b}$

Solution

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ (matlab)

Problem statement

$$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \text{ s.t. } \mathbf{x}^T\mathbf{x} = 1$$

non - trivial lsq solution to $\mathbf{Ax} = 0$

Solution

$$[\mathbf{v}, \lambda] = \text{eig}(\mathbf{A}^T\mathbf{A})$$

$$\lambda_1 < \lambda_{2..n} : \mathbf{x} = \mathbf{v}_1$$

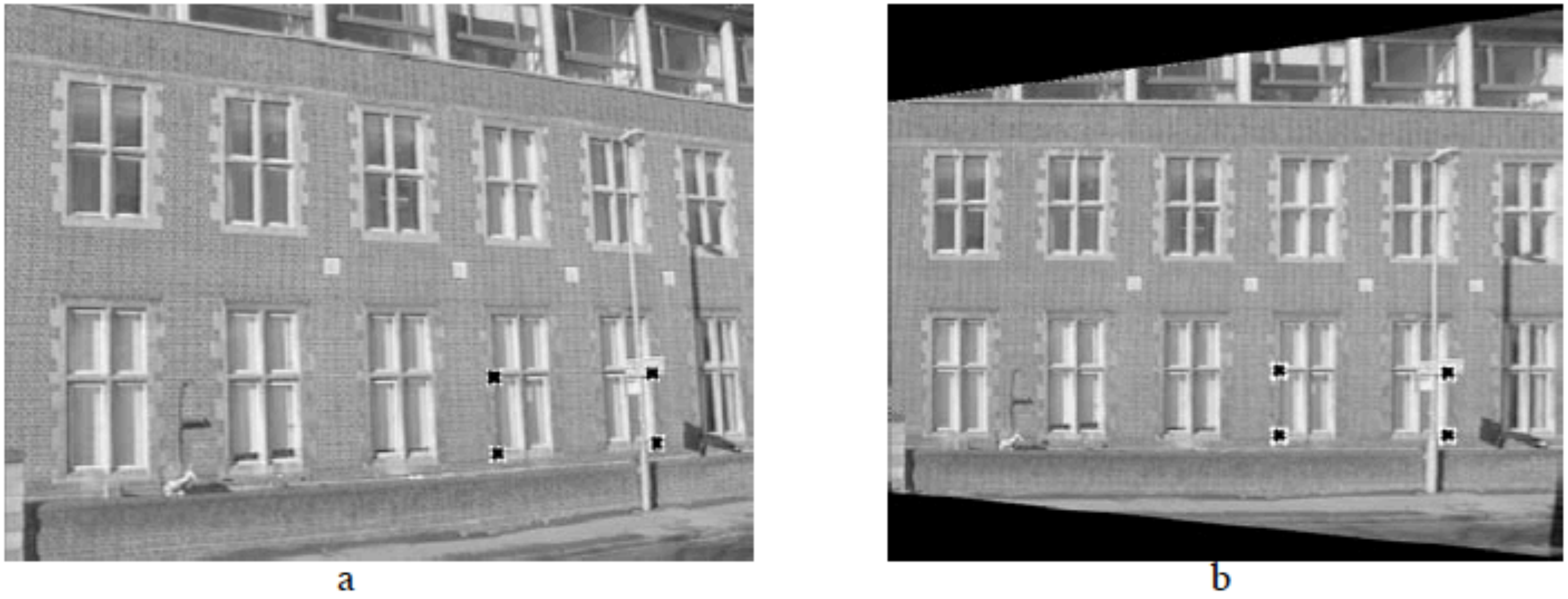
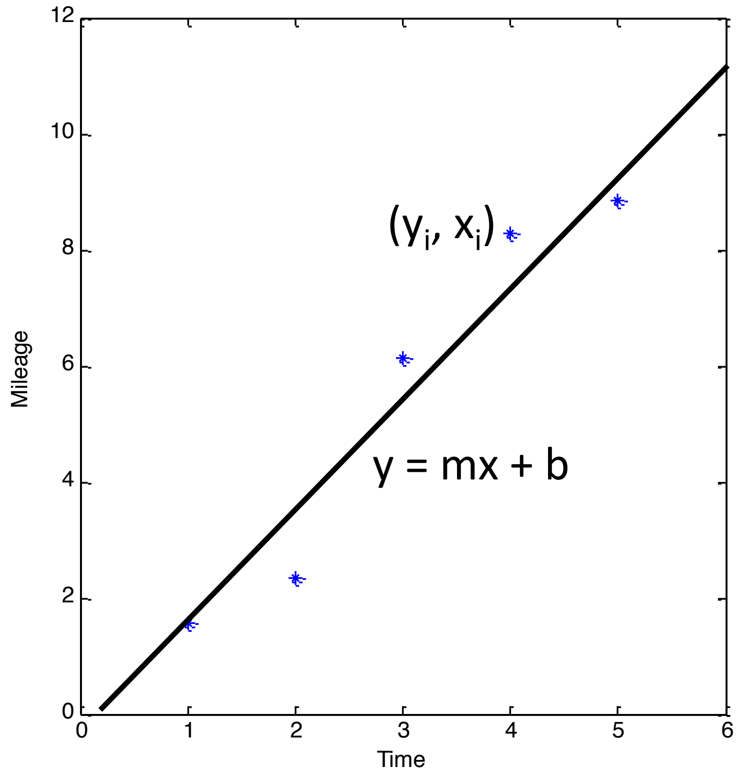
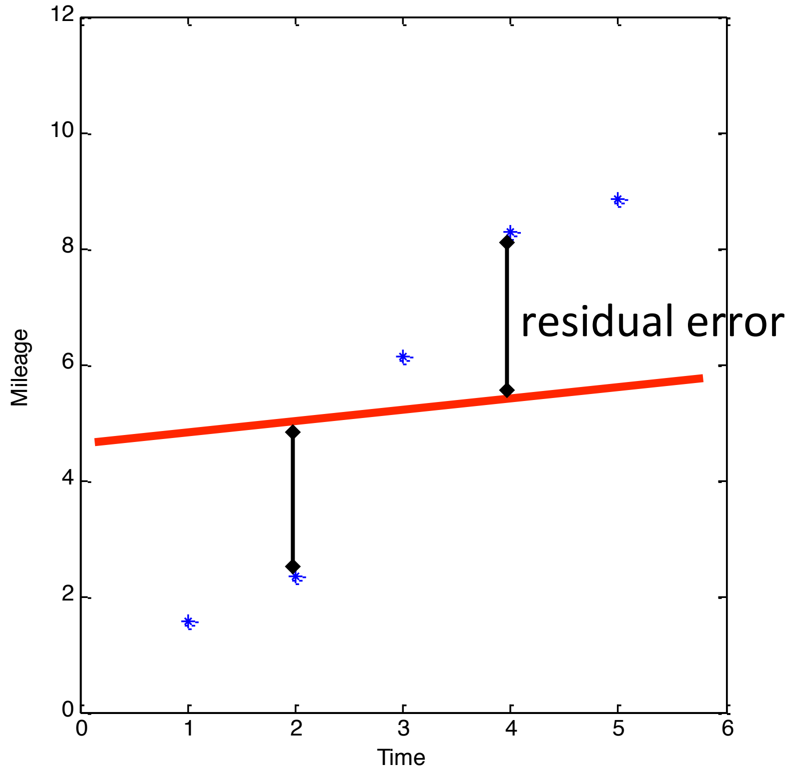


Fig. 2.4. **Removing perspective distortion.** *(a) The original image with perspective distortion – the lines of the windows clearly converge at a finite point. (b) Synthesized frontal orthogonal view of the front wall. The image (a) of the wall is related via a projective transformation to the true geometry of the wall. The inverse transformation is computed by mapping the four imaged window corners to corners of an appropriately sized rectangle. The four point correspondences determine the transformation. The transformation is then applied to the whole image. Note that sections of the image of the ground are subject to a further projective distortion. This can also be removed by a projective transformation.*

Least squares: linear regression



Linear regression



$$\mathrm{Cost}(m,b) = \sum_{i=1}^{n} |y_i - (mx_i + b)|^2$$

Linear regression

$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

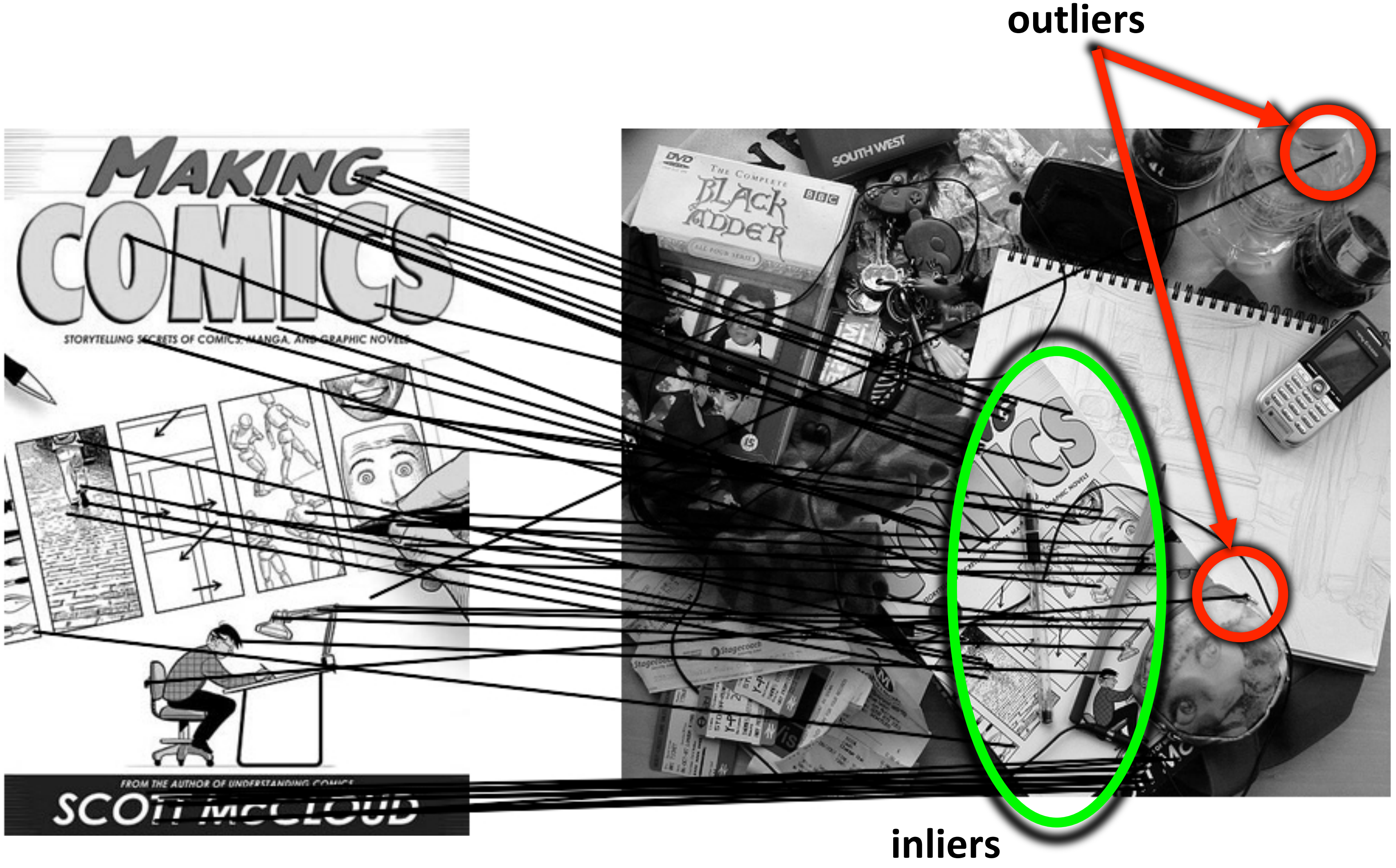
Image Alignment Algorithm

Given images A and B

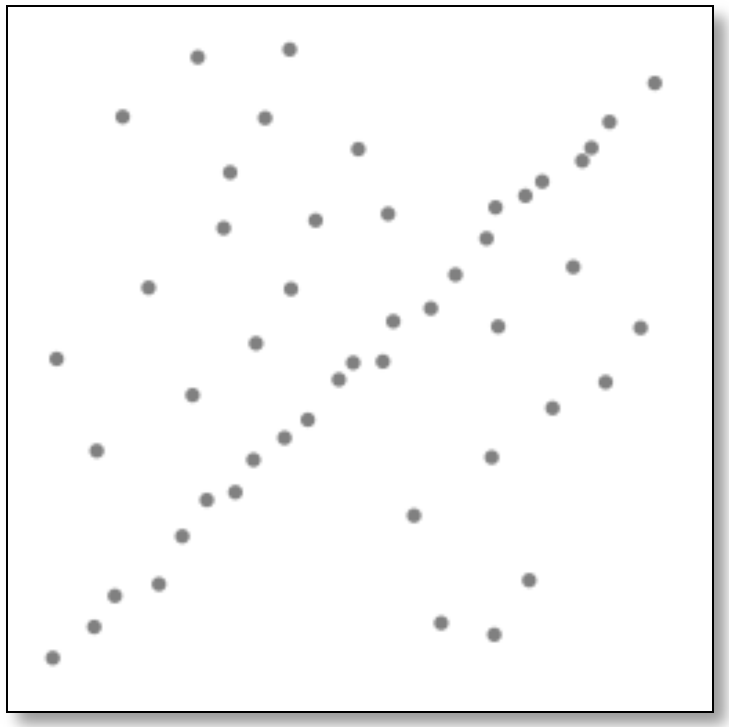
1. Compute image features for A and B
2. Match features between A and B
3. Compute homography between A and B using least squares on set of matches

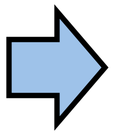
What could go wrong?

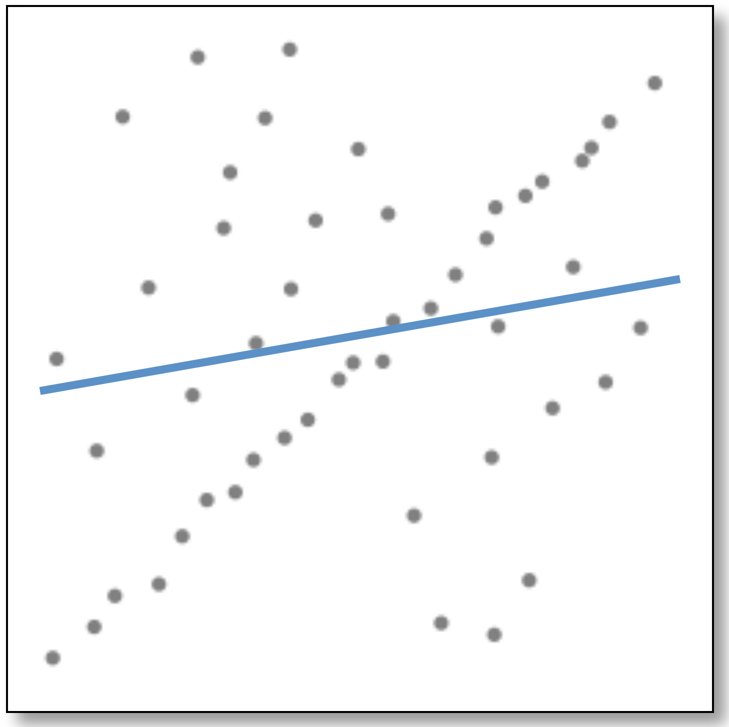
Outliers


outliers
inliers
MAKING COMICS
STORYTELLING SECRETS OF COMICS, MANGA, AND GRAPHIC NOVELS
FROM THE AUTHOR OF UNDERSTANDING COMICS
SCOTT McCLOUD
The Complete BLACK ADDER
SOUTH WEST

Robustness


Problem: Fit a line to these datapoints




Least squares fit

What can we do?

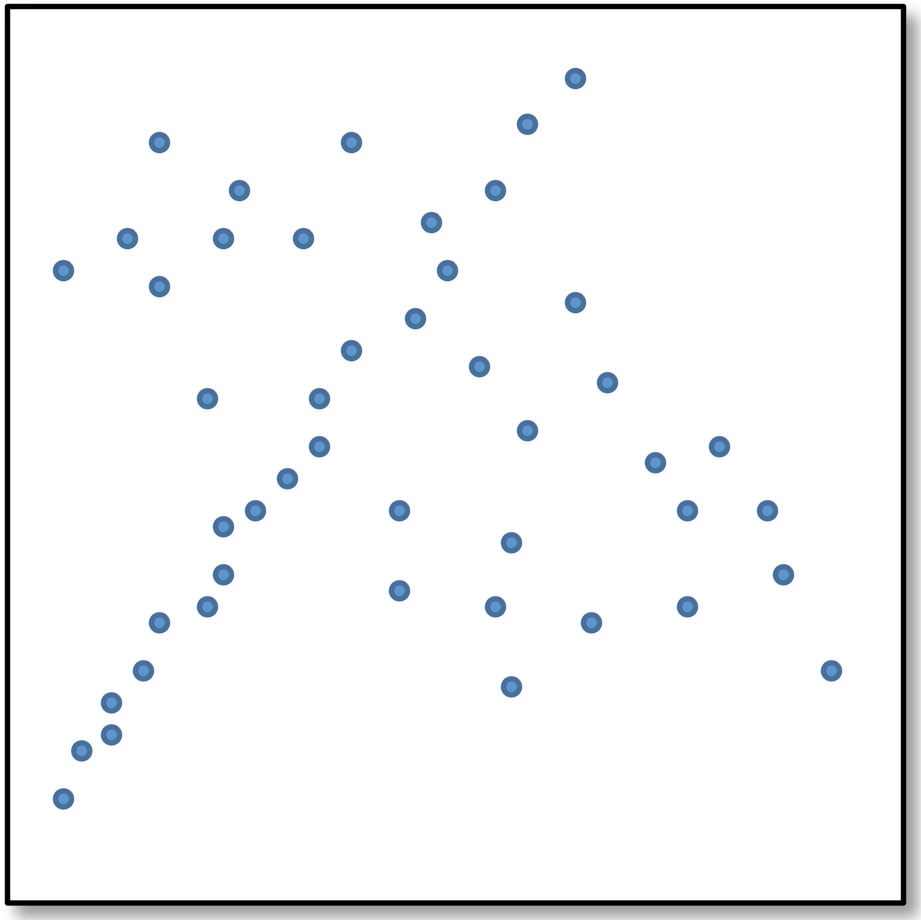
- Suggestions?

Idea

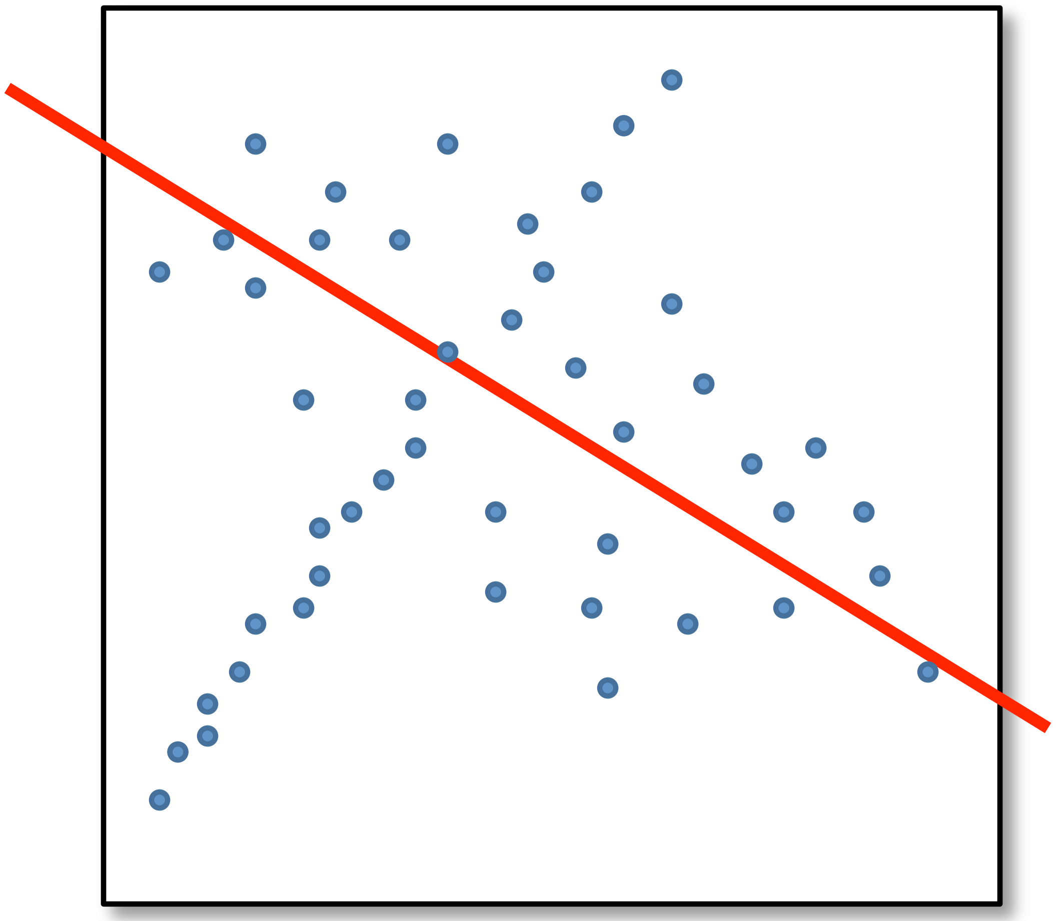
- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

Counting inliers

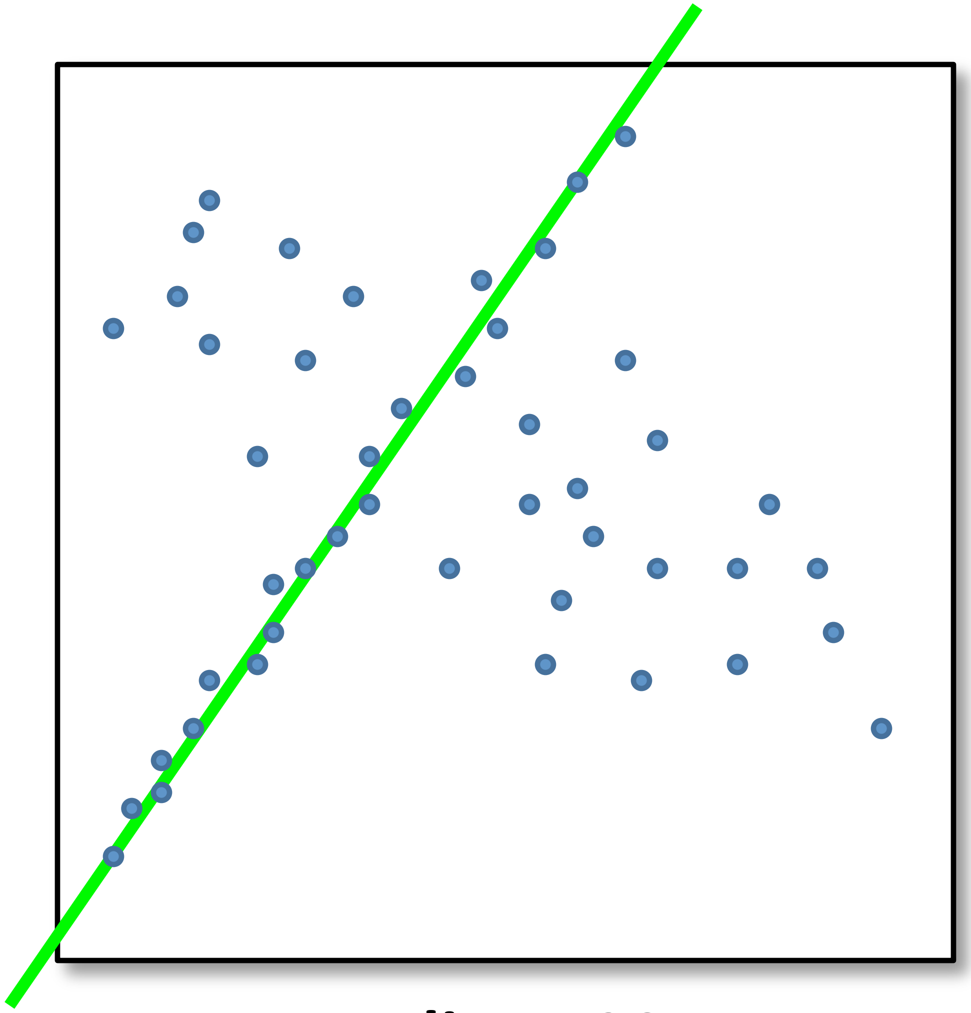


Counting inliers



Inliers: 3

Counting inliers



Inliers: 20