4. Image Matching

Hough transform, LLS, RANSAC, Image warping and Stitching

Matlab Functions

conv: 1D convolution

conv2: 2D convolution

filter2: filter2(F,X), apply filter F to image X

Matlab Functions

[Fx, Fy] =gradient(F), always central

[magnitude, direction]=imgradient(I,method)– for images,there are several options:

Method	Description
'sobel'	Sobel gradient operator. The gradient of a pixel is a weighted sum of pixels in the 3-by-3 neighborhood. For gradients in the vertical (y) direction, the weights are: `[ 1 2 1; 0 0 0; -1 -2 -1 ]` In the x direction, the weights are transposed.
'prewitt'	Prewitt gradient operator. The gradient of a pixel is a weighted sum of pixels in the 3-by-3 neighborhood. For gradients in the vertical (y) direction, the weights are: `[ 1 1 1; 0 0 0; -1 -1 -1 ]` In the x direction, the weights are transposed.
'central'	Central difference gradient. The gradient of a pixel is a weighted difference of neighboring pixels. In the y direction, `dI/dy = (I(y+1) - I(y-1))/2`.
'intermediate'	Intermediate difference gradient. The gradient of a pixel is the difference between an adjacent pixel and the current pixel. In the y direction, `dI/dy = I(y+1) - I(y)`.
'roberts'	Roberts gradient operator. The gradient of a pixel is the difference between diagonally adjacent pixels. For gradients in one direction, the weights are: `[ 1 0; 0 -1 ]` In the orthogonal direction, the weights are flipped along the vertical axis.

Matlab Functions

xcorr2(a,b): cross correlation (use for template matching)

xcorr2(a): auto-correlation (use for image analysis)

detectHarrisFeatures

detectSIFTFeatures

Hough Transform

Detecting lines in images

Basic idea: Line y = ax+b

Two points (x,y) and(z,k) define a line in the x-y plane

These two points give rise to two different lines in a-b plane.

In (a-b) space these lines will intersect in a point (a’,b’) where a’ is the rise and b’ the intersect of the line defined by (x,y) and (z,k) in x-y plane.

The fact is that all points on the line defined by (x,y) and (z,k) in x-y plane will parameterize lines that intersect in (a’,b’) in a-b plane.

Points that lie on a line will form a “cluster of crossings” in the a-b plane.

Hough Transform

Detecting lines in images

Basic idea: Line $y = ax+b$

Two points (x,y) and(z,k) define a line in the x-y plane

These two points give rise to two different lines in a-b plane.

In (a-b) space these lines will intersect in a point (a’,b’) where a’ is the rise and b’ the intersect of the line defined by (x,y) and (z,k) in x-y plane.

The fact is that all points on the line defined by (x,y) and (z,k) in x-y plane will parameterize lines that intersect in (a’,b’) in a-b plane.

Points that lie on a line will form a “cluster of crossings” in the a-b plane.

Hough Transform

Detecting lines in images

Basic idea: Line $y = ax+b$

Two points (x,y) and(z,k) define a line in the x-y plane

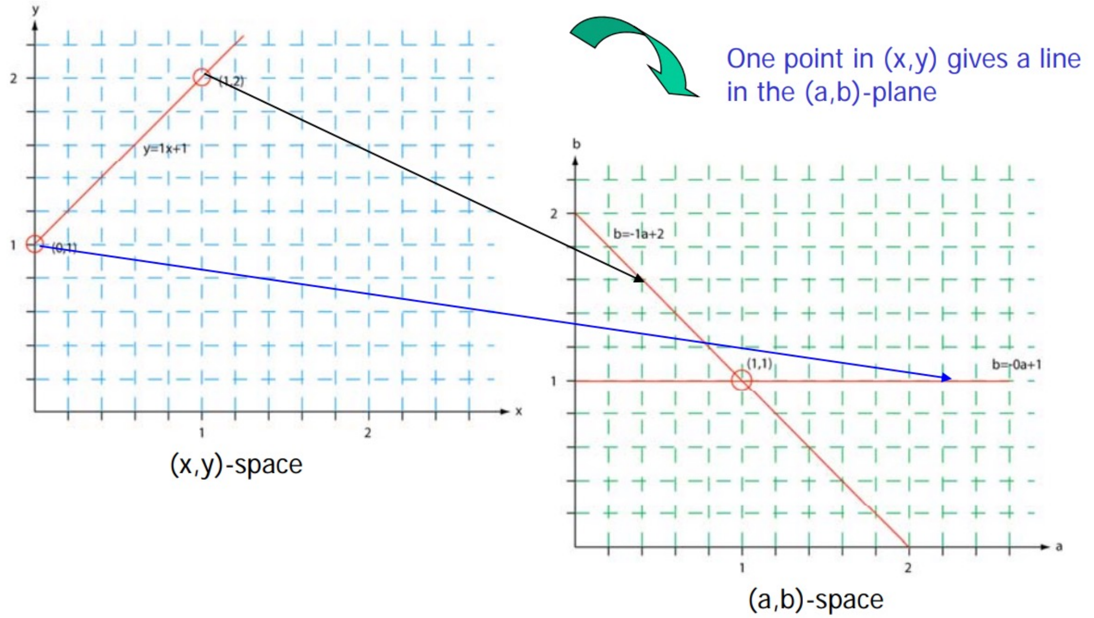
These two points give rise to two different lines in a-b plane.

In (a-b) space these lines will intersect in a point (a’,b’) where a’ is the rise and b’ the intersect of the line defined by (x,y) and (z,k) in x-y plane.

The fact is that all points on the line defined by (x,y) and (z,k) in x-y plane will parameterize lines that intersect in (a’,b’) in a-b plane.

Points that lie on a line will form a “cluster of crossings” in the a-b plane.

Hough Transform

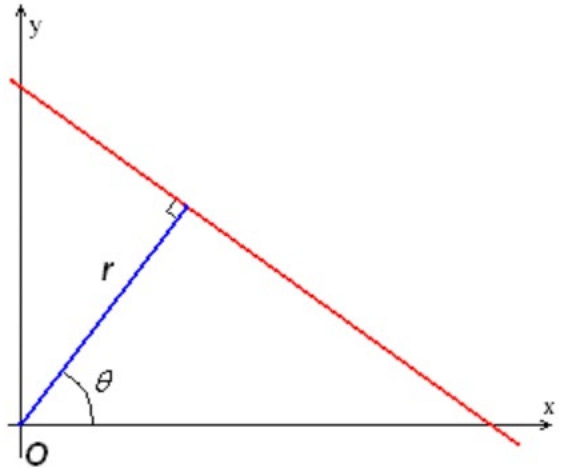


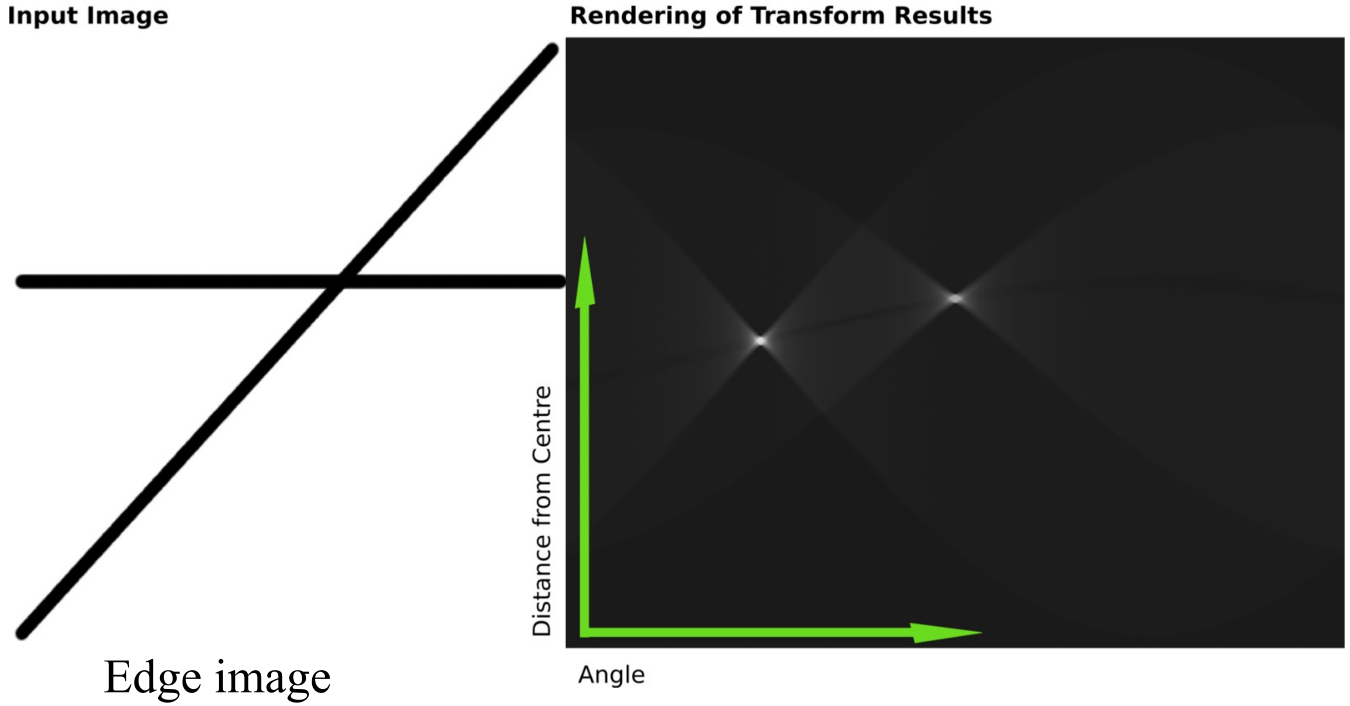
Collinear points in x-y, intersectling lines in a-b

Hough Transform

Line parametrisation

$$r = x\cos\theta + y\sin\theta,$$





Edge image

Least-square fitting

1. Equation of a line: y = mx +c
2. Consider n points

$$mx_1 + c = y_1$$
$$\vdots$$
$$mx_n + c = y_n$$

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

$$\mathbf{A}p = Y$$

$$\mathbf{A}p = Y$$

$$\mathbf{A}^{\mathbf{T}}\mathbf{A}p = \mathbf{A}^{\mathbf{T}}Y$$

$$p = (\mathbf{A}^{\mathbf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathbf{T}}Y$$

$$\min \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

Least-square fitting

1. Equation of a line: y = mx +c
2. Consider n points

$$mx_1 + c = y_1$$
$$\vdots$$
$$mx_n + c = y_n$$

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

$$\mathbf{A}p = Y$$

$$\mathbf{A}p = Y$$

$$\mathbf{A}^{\mathbf{T}}\mathbf{A}p = \mathbf{A}^{\mathbf{T}}Y$$

$$p = (\mathbf{A}^{\mathbf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathbf{T}}Y$$

$$\min \sum_{i=1}^{n} (y_i - mx_i - c)^2$$

Estimating transformations from Image features

- If point correspondences (*x,y*)<-->(*x',y'*) are known
- *a'*s can be determined by least squares fit

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

$$a_7 x' x + a_8 x' y + x' = a_1 x + a_2 y + a_3$$

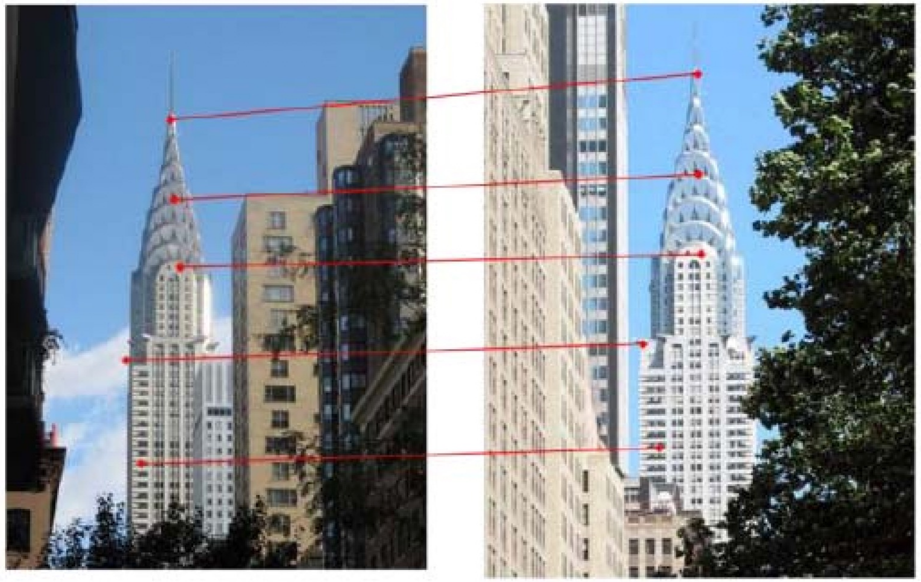
$$a_7 y' x + a_8 y' y + y' = a_7 x + a_8 y + a_6$$

$$x' = a_1 x + a_2 y + a_3 - a_7 x' x - a_8 x' y$$

$$y' = a_7 x + a_8 y + a_6 - a_7 y' x - a_8 y' y$$

$$a_1 x + a_2 y + a_3 - a_7 x' x - a_8 x' y = x'$$

$$a_7 x + a_8 y + a_6 - a_7 y' x - a_8 y' y = y'$$



Two rows for each point *i*

$$\begin{bmatrix} & & & & \vdots & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

$$Aa = \mathbf{x}'$$

$$a = (A^T A)^{-1} A^T \mathbf{x}'$$

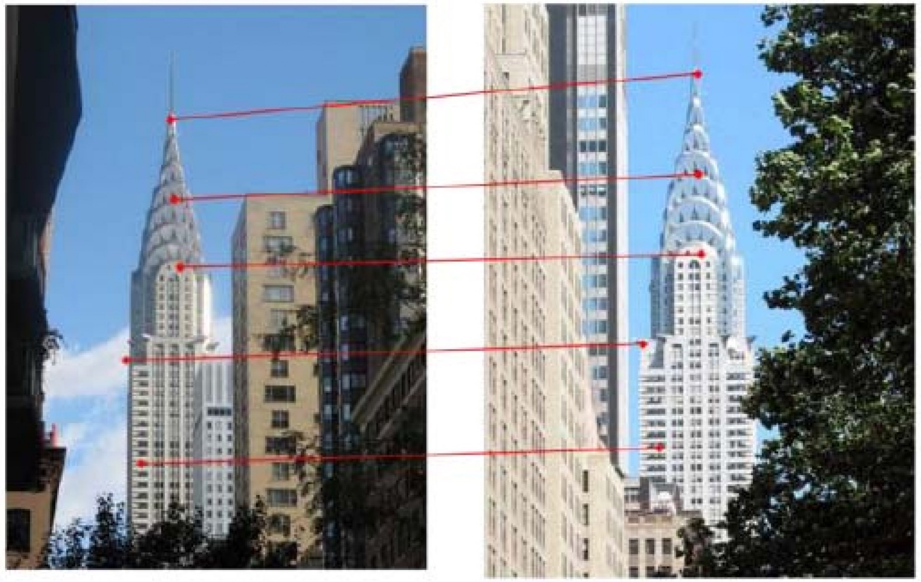
Estimating transformations from Image features

- If point correspondences (*x,y*)<-->(*x’,y’*) are known
- *a*’s can be determined by least squares fit

$$x' = \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1}$$

$$y' = \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1}$$

$$a_7 x'x + a_8 x'y + x' = a_1 x + a_2 y + a_3$$
$$a_7 y'x + a_8 y'y + y' = a_7 x + a_8 y + a_6$$
$$x' = a_1 x + a_2 y + a_3 - a_7 x'x - a_8 x'y$$
$$y' = a_7 x + a_8 y + a_6 - a_7 y'x - a_8 y'y$$
$$a_1 x + a_2 y + a_3 - a_7 x'x - a_8 x'y = x'$$
$$a_7 x + a_8 y + a_6 - a_7 y'x - a_8 y'y = y'$$



Two rows for each point *i*

$$\begin{bmatrix} & & & & \vdots & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -x_i y_i' & -y_i y_i' \\ & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} \vdots \\ x_i' \\ y_i' \\ \vdots \end{bmatrix}$$

$$Aa = \mathbf{x}'$$

$$a = (A^T A)^{-1} A^T \mathbf{x}'$$

Problem with Linear Least Squares (LLS) fitting

1. Noisy data and outliers

Solution:

a) Voting. Check all possible combinations. Cycle through features, cast votes for model parameters. Bad idea due to high computation cost.

b) RANdom SAmple Consensus (RANSAC)

RANSAC

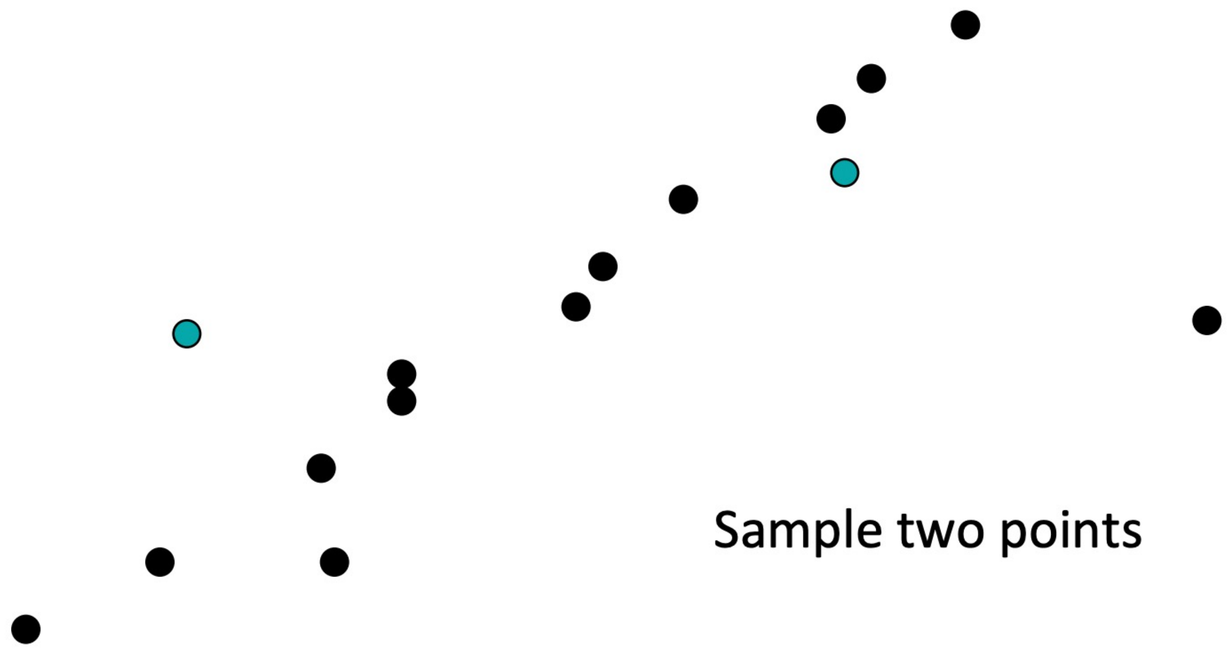
Look for ‘only’ inliers.

1. Randomly select a seed group to estimate transformation
2. Compute transformation
3. Compute inliers
4. If inliers are large, recompute transformation

Keep transformation with most inliers

RANSAC: line fitting example

Estimate the best line

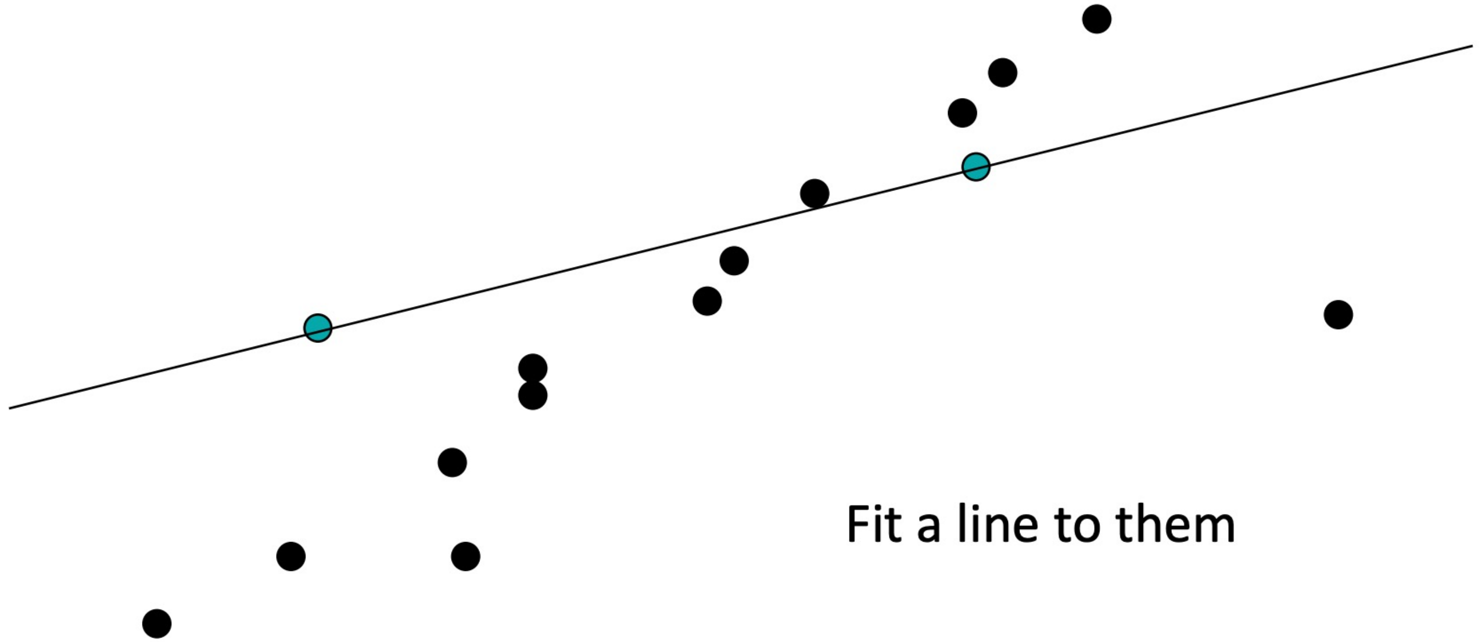


Sample two points

Slide credit: Jinxiang Chai

RANSAC: line fitting example

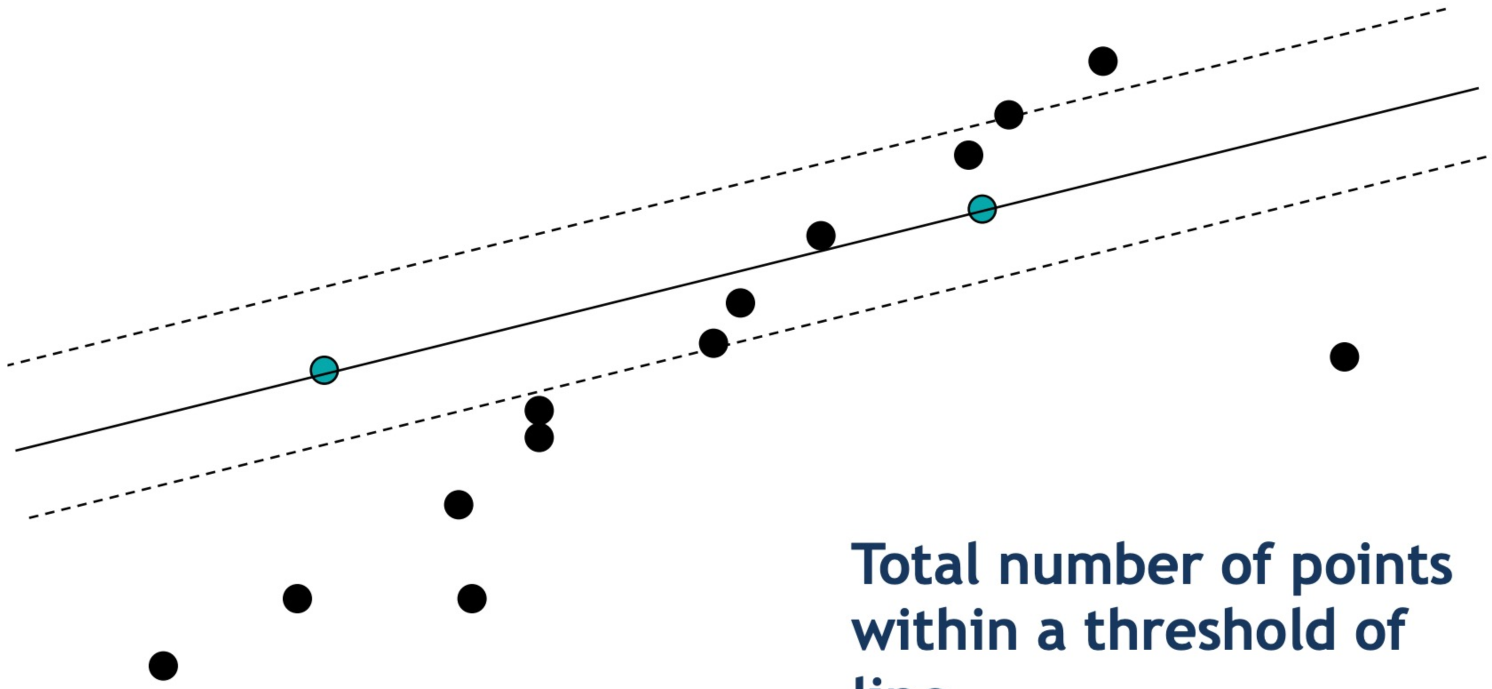
Estimate the best line



Slide credit: Jinxiang Chai

RANSAC: line fitting example

Estimate the best line

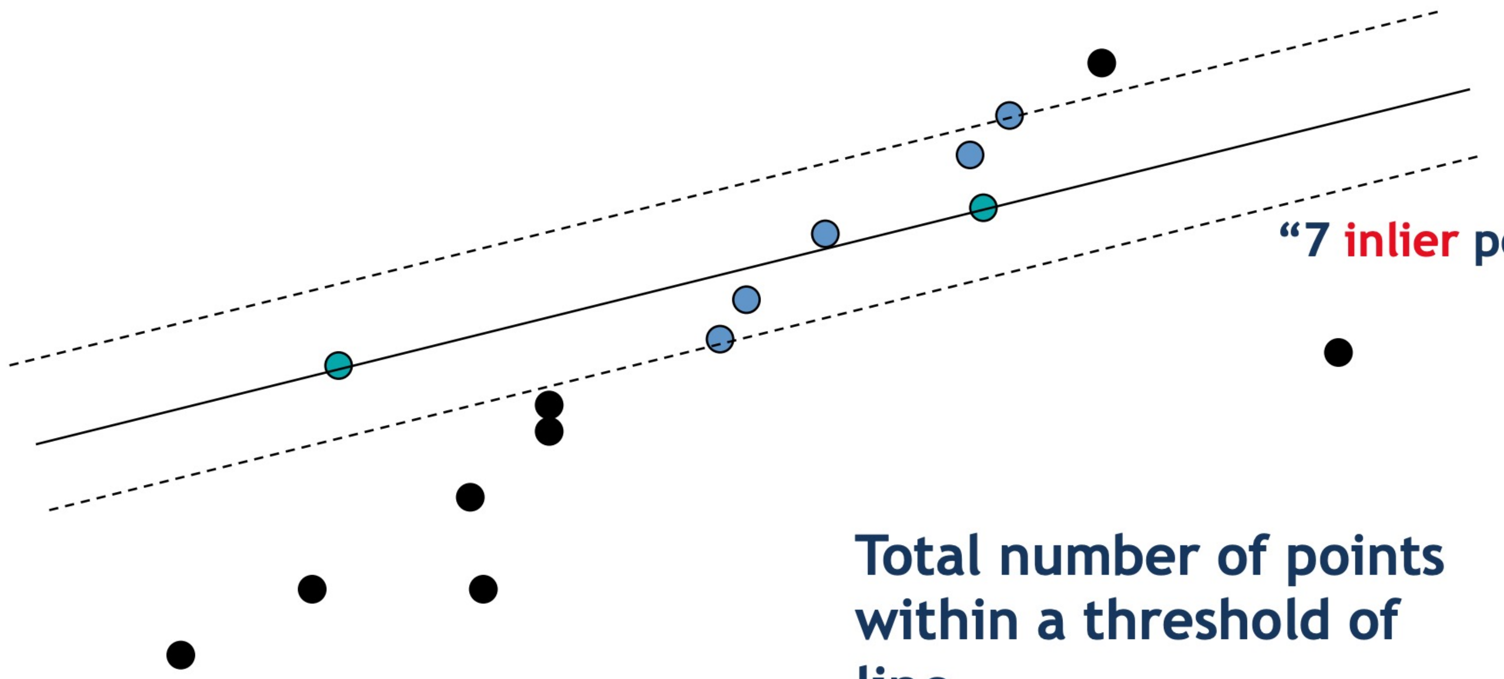


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC: line fitting example

Estimate the best line

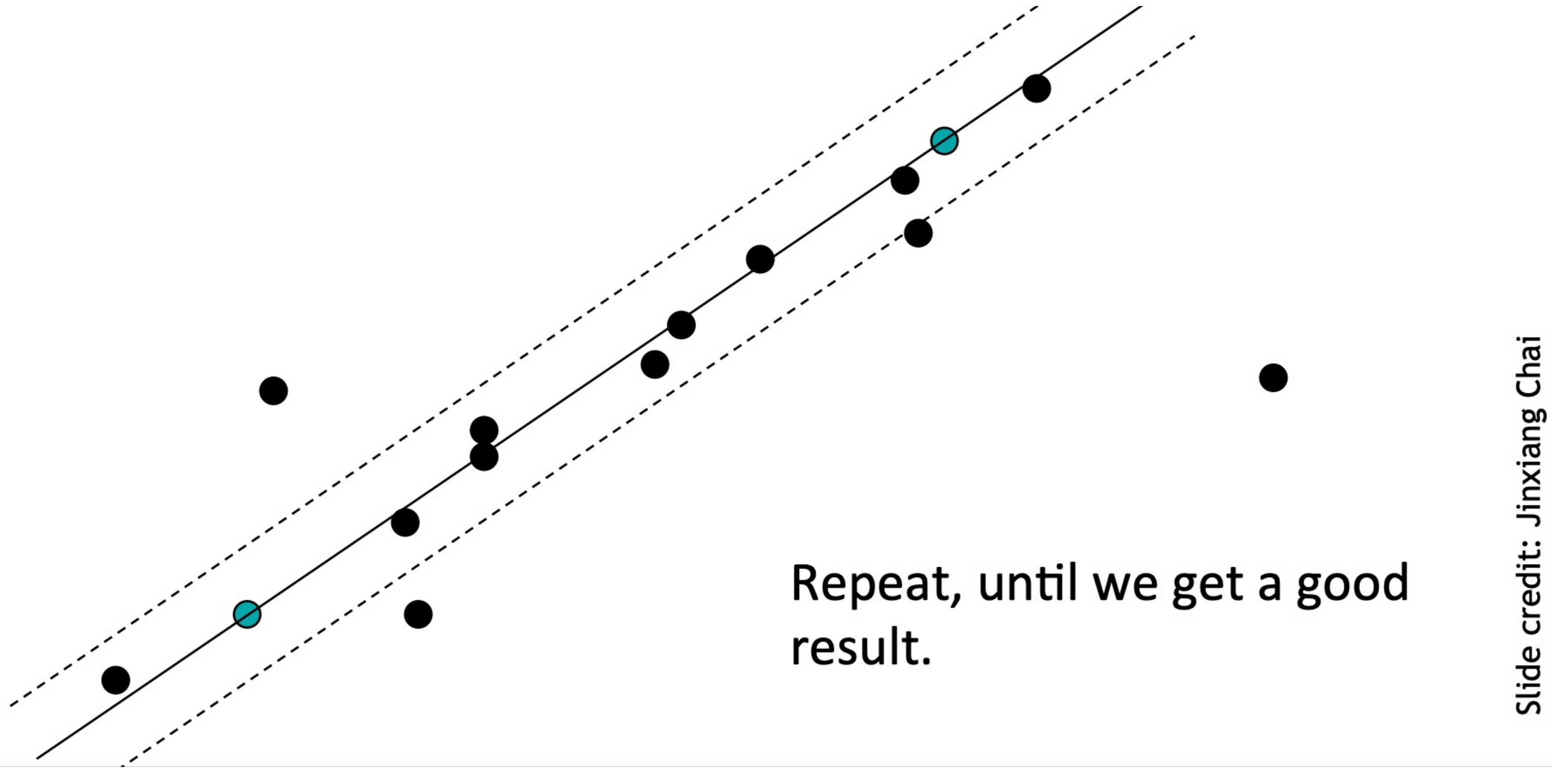


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC: line fitting example

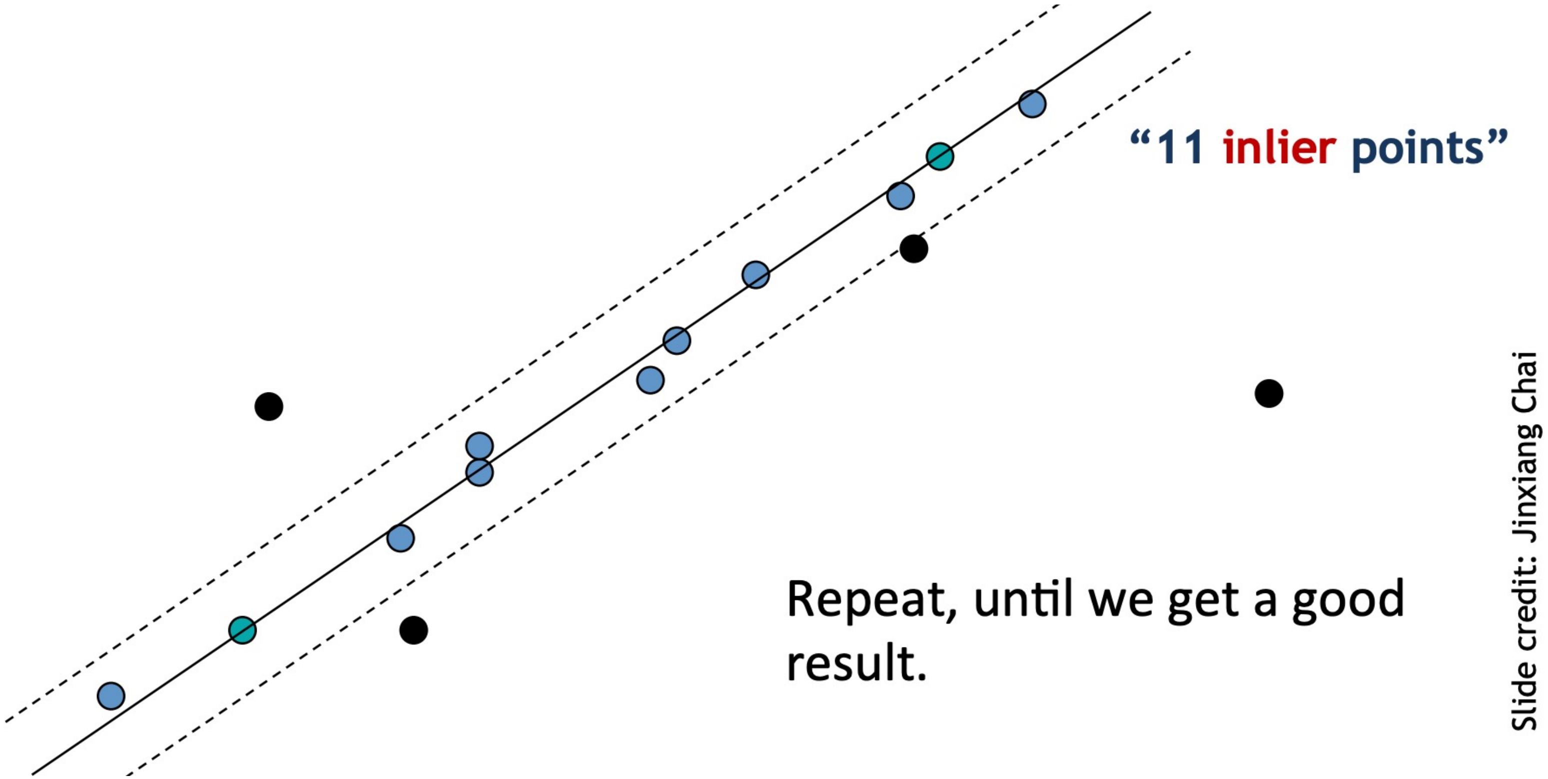
Estimate the best line



Repeat, until we get a good result.

RANSAC: line fitting example

Estimate the best line



RANSAC: line-fitting example

After RANSAC, use all inliers to improve transformation. This may change the transformation, so perform another reclassification of inliers-outliers.

RANSAC is fast but it can accommodate relatively small number of outliers.

Voting is efficient but computationally expensive.

RANSAC: line-fitting example

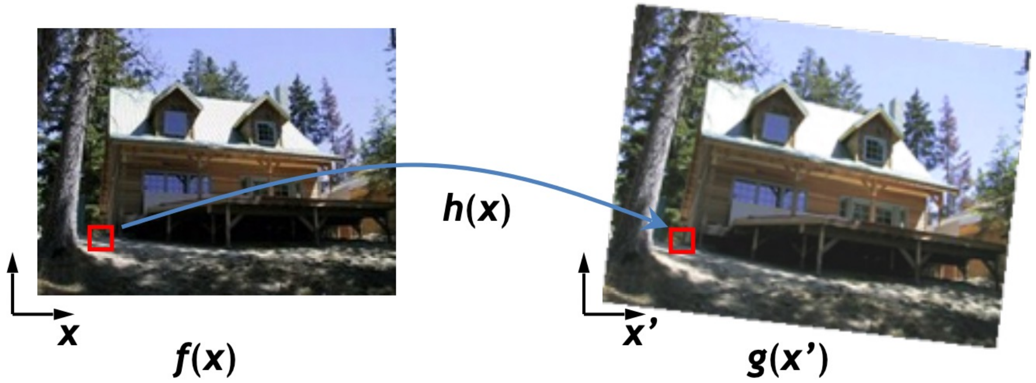
After RANSAC, use all inliers to improve transformation. This may change the transformation, so perform another reclassification of inliers-outliers.

RANSAC is fast but it can accommodate relatively small number of outliers.

Voting is efficient but computationally expensive.

Image Warping: Forward

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



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

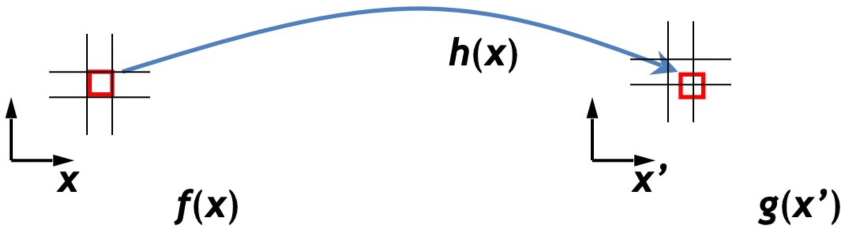
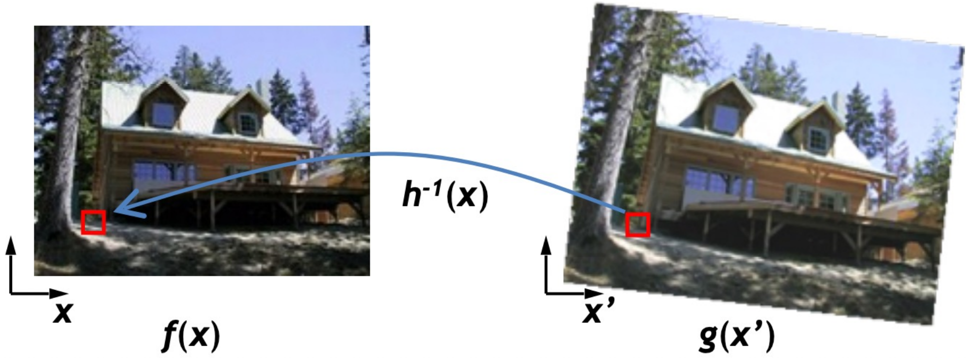
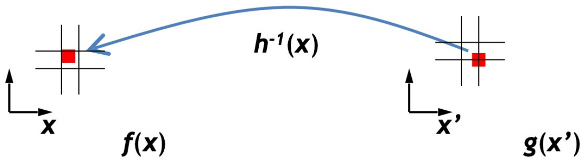


Image Warping: Backward

Get each pixel g(x’) from its corresponding location x’ = h(x) in f(x) ?



- What if pixel comes from “between” two pixels?
- Answer: *resample* color value from *interpolated* source image

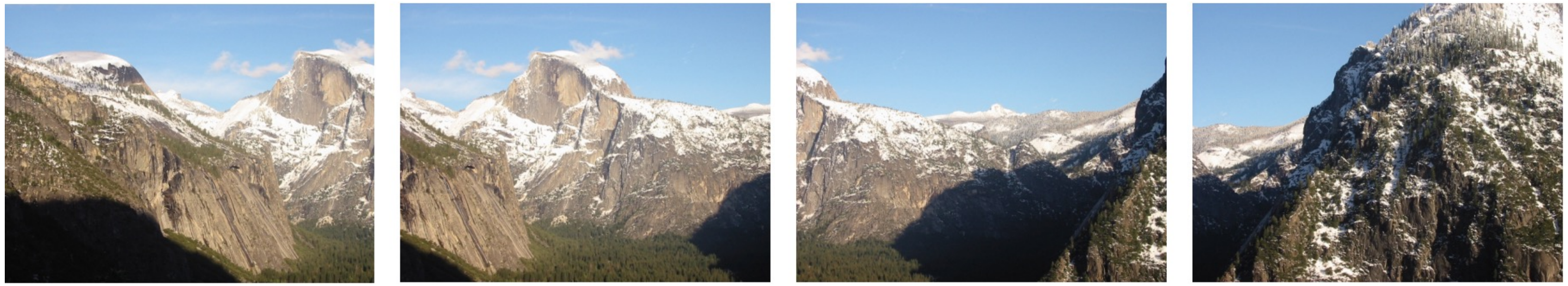


Homography estimation: Direct Linear Transformation (DLT)

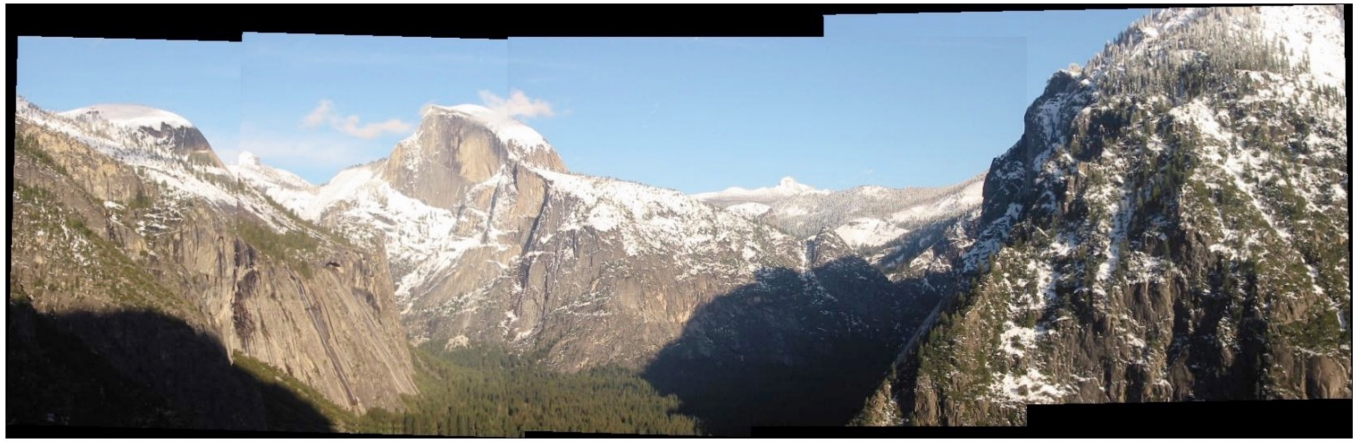
Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$?

Panoramic Image Stitching

Given a set of images



Obtain:



Panoramic Image Stitching

1. Detect and match keypoints
2. Estimate transformation
3. Perform RANSAC and refine transformation

Question: What if the images are not aligned?

P1: Image warping

1. Warp simpsons.jpeg to bus.jpeg such that simpsons appear on the bus advertisement

You can manually select the image features for matching

P2: Image stitching

1. Build a panaroma using keble images.
2. Use at least 3 methods to find features.
3. Estimate transformations with and without RANSAC
4. Manually match images and estimate transformation
5. Is any result from 3 is close to 4 ?
6. Repeat the experiment on various images of “La Place des Jacobins”.

Bonus: Can you automatically align images?