Introduction to Computer Vision

Lecture 3 - Classic Vision II

Prof. He Wang

Start with A Task: Lane Detection

How to detect the lane?

https://medium.com/@realderektan/self-driving-car-project-part-1-lane-lines-detector-6d960e2b023

Start with Detecting Edges

- Edge detector

https://towardsdatascience.com/edge-detection-in-python-a3c263a13e03

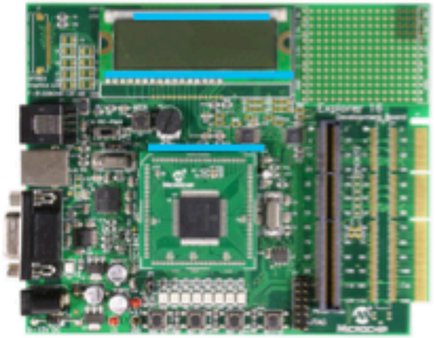
Summary of Canny Edge Detection

- Edge: where pixel intensity changes drastically
- Jointly detecting edge and smoothing by convolving with the derivative of a Gaussian filter
- Non-maximal suppression
- Thresholding and linking (hysteresis):

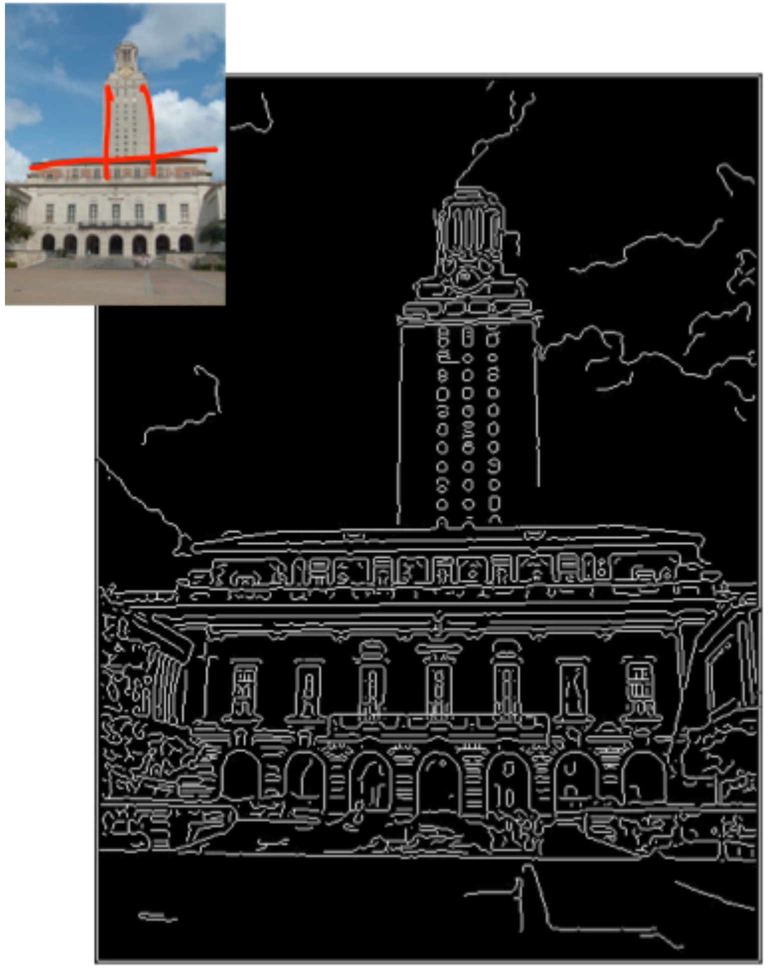
Line Fitting

Line Detection

- Many objects are characterized by presence of straight lines
- Detect lines?

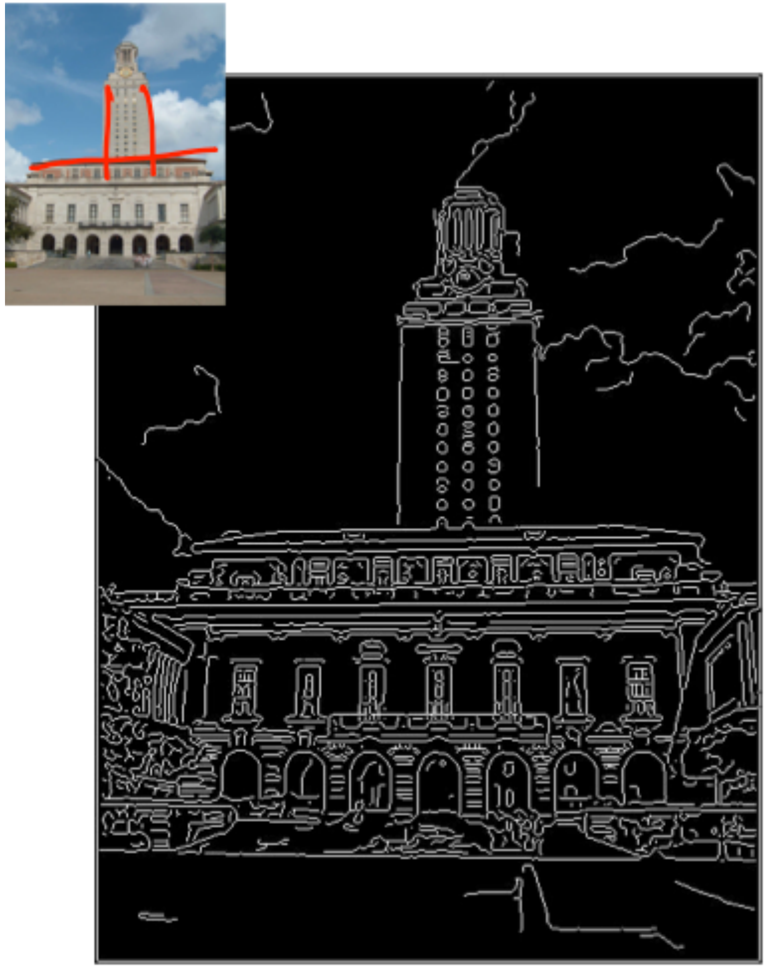


Challenge of Line Detection



- Aren’t we done just by doing edge detection?

Challenge of Line Detection



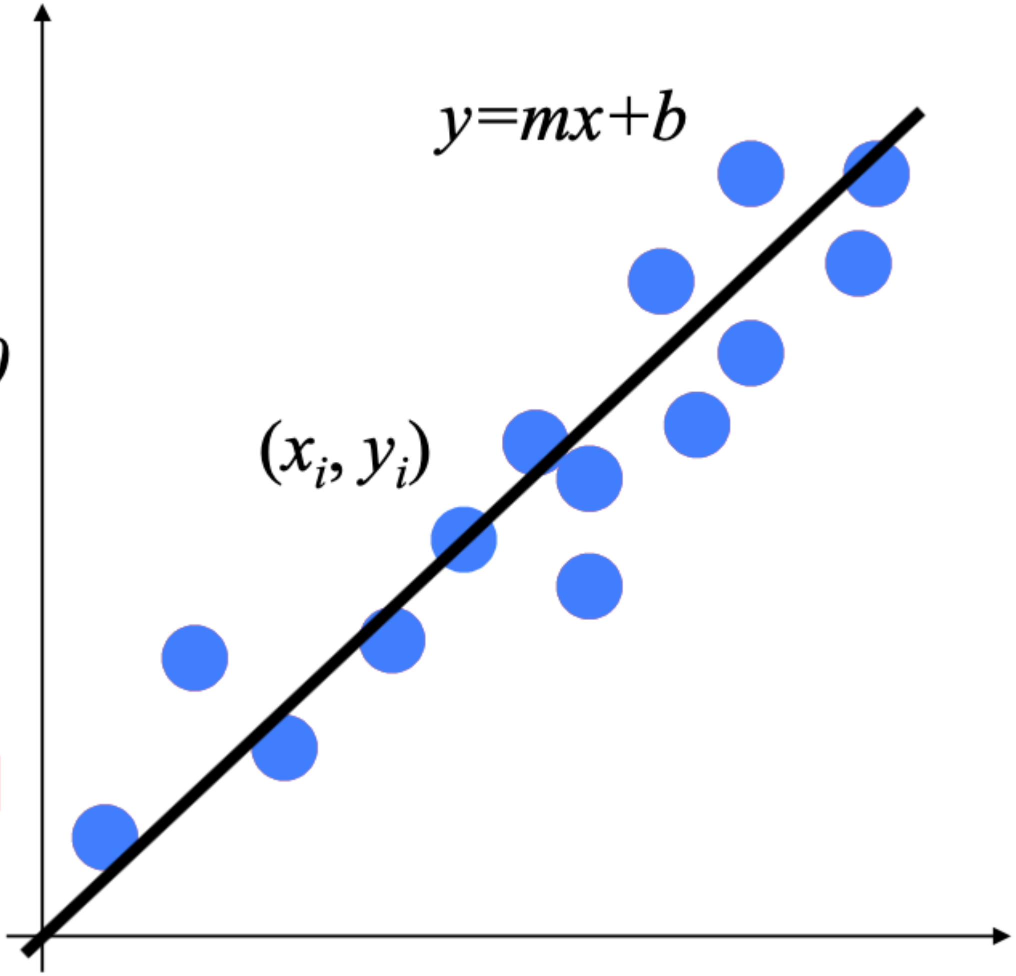
- Aren’t we done just by doing edge detection?
- No, there are many problems:
  - Occlusion
  - Not a straight line
  - Multiple lines, which one?

Line Fitting: Least Square Method

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i - m\,x_i - b = 0$

  [Eq. 1]
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$ [Eq. 2]



Line Fitting: Least Square Method

$$\boxed{E = \sum_{i=1}^{n} (y_i - mx_i - b)^2}$$ [Eq. 2]

$$E = \sum_{i=1}^{n} \left(y_i - \begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right)^2 = \left\| \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} \right\|^2 = \|Y - XB\|^2$$ [Eq. 3]

$$= (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$ [Eq. 4]

Find B=[m, b]^T that minimizes E

$$\frac{dE}{dB} = -2X^T Y + 2X^T XB = 0$$ [Eq. 5]

$$X^T XB = X^T Y$$ [Eq. 7]

Normal equation

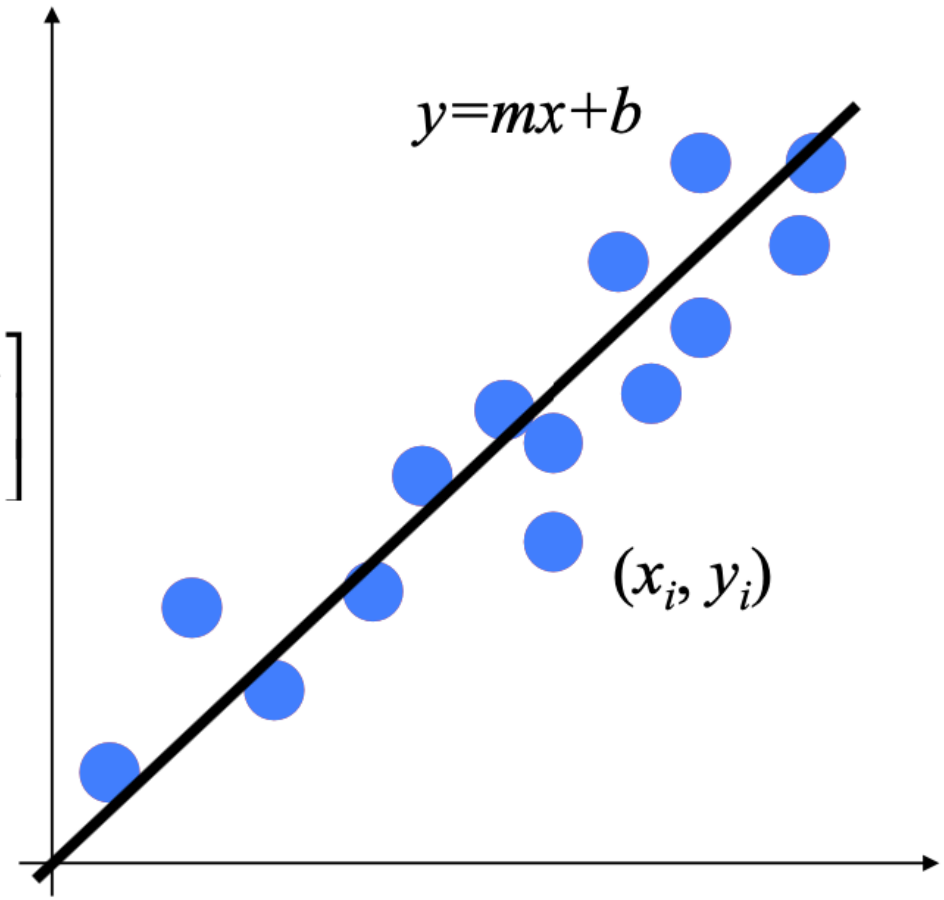
$$\boxed{B = \left(X^T X\right)^{-1} X^T Y}$$ [Eq. 6]

Line Fitting: Least Square Method

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

$$\boxed{B = \left(X^T X\right)^{-1} X^T Y} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

[Eq. 6]



Limitations

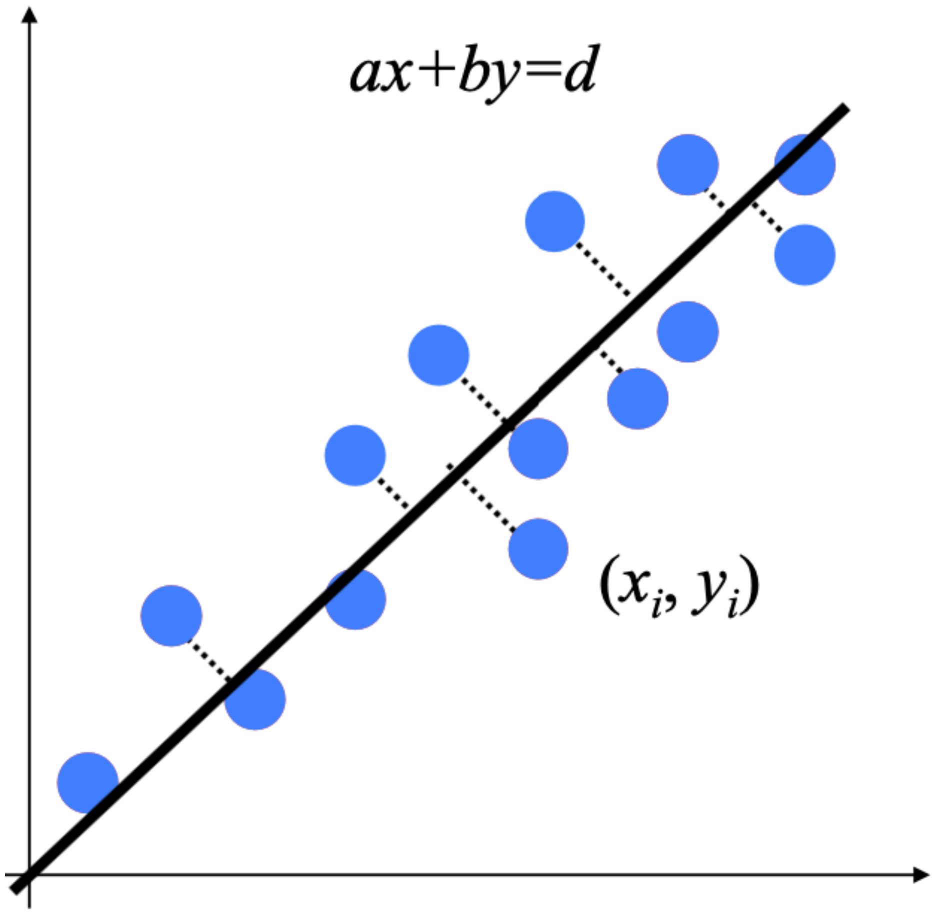
- Fails completely for vertical lines

Line Fitting for the General Equation of a Line

$$E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$$

$$A = \begin{bmatrix} x_1, y_1, 1 \\ \cdots \\ x_i, y_i, 1 \\ \cdots \\ x_n, y_n, 1 \end{bmatrix} \qquad h = \begin{bmatrix} a \\ b \\ d \end{bmatrix}$$

$$\underbrace{\boxed{A}}_{\text{data}}\ \underbrace{\boxed{h}}_{\text{model parameters}} = 0$$ [Eq. 9]



Line Fitting for the General Equation of a Line

$$A\,h = 0 \qquad \text{A is rank deficient}$$

$$\text{Minimize } \| A\,h \| \quad \text{subject to } \boxed{\| h \| = 1}$$

To avoid trivial solution $h = 0$, we need a constraint for h

Line Fitting for the General Equation of a Line

Optimization problem: $\text{Minimize} \; \| A\, h \| \quad \text{subject to} \quad \| h \| = 1$

Solve h using **Singular Value Decomposition (SVD):**

$$A_{n\times 3} = U_{n\times n} D_{n\times 3} V_{3\times 3}^T$$

where $U_{n\times n}$ and $V_{3\times 3}$ are orthogonal matrices ($V^T V = I_{3\times 3}$ and $V = [c_1, c_2, c_3]$),

$$D = \begin{bmatrix} diag\{\lambda_1, \lambda_2, \lambda_3\} \\ O \end{bmatrix}$$ and $|\lambda_1| > |\lambda_2| > |\lambda_3|$.

SVD is an extension of Eigenvalue decomposition (only works for square matrices $A_{n\times n}$) to general matrices $A_{n\times m}$.

Line Fitting for the General Equation of a Line

Optimization problem: $\text{Minimize} \| A\,h \| \quad \text{subject to} \quad \| h \| = 1$

Solve h using **Singular Value Decomposition (SVD):** $A_{n\times 3} = U_{n\times n} D_{n\times 3} V^T_{3\times 3}$

Given $V^T_{3\times 3} = \begin{bmatrix} c_1^T \\ c_2^T \\ c_3^T \end{bmatrix}$ (note $\{c_i\}$ forms an orthogonal basis),

then $h = \alpha_1 c_1 + \alpha_2 c_2 + \alpha_3 c_3$ (note $\alpha_1^2 + \alpha_2^2 + \alpha_3^2 = 1$ since $||h|| = 1$)

$$\therefore Ah = U_{n\times n} D_{n\times 3} V^T_{3\times 3} h_{3\times 1} = U_{n\times n} D_{n\times 3} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = U_{n\times n} \begin{bmatrix} diag\{\lambda_1\alpha_1, \lambda_2\alpha_2, \lambda_3\alpha_3\} \\ O \end{bmatrix}$$

Line Fitting for the General Equation of a Line

Optimization problem: $\text{Minimize} \, \| A \, h \| \quad \text{subject to} \quad \| h \| = 1$

Solve h using **Singular Value Decomposition (SVD):** $A_{n\times 3} = U_{n\times n} D_{n\times 3} V^T_{3\times 3}$

$$Ah = U_{n\times n} D_{n\times 3} V^T_{3\times 3} h_{3\times 1} = U_{n\times n} D_{n\times 3} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = U_{n\times n} \begin{bmatrix} diag\{\lambda_1\alpha_1, \lambda_2\alpha_2, \lambda_3\alpha_3\} \\ O \end{bmatrix}$$

$$\therefore \|Ah\|^2 = \left\| \begin{bmatrix} diag\{\lambda_1\alpha_1, \lambda_2\alpha_2, \lambda_3\alpha_3\} \\ O \end{bmatrix} \right\|^2 = (\lambda_1\alpha_1)^2 + (\lambda_2\alpha_2)^2 + (\lambda_3\alpha_3)^2$$

(Orthogonal matrix U doesn't affect the norm)

Note $\alpha_1^2 + \alpha_2^2 + \alpha_3^2 = 1$ and $|\lambda_1| > |\lambda_2| > |\lambda_3|$, $\quad \therefore \|Ah\|^2 \geq \lambda_3^2$

Summary: Line Fitting for the General Equation of a Line

$$\boxed{E = \sum_{i=1}^{n} (ax_i + by_i - d)^2}$$

$$A = \begin{bmatrix} x_1, y_1, 1 \\ \cdots \\ x_i, y_i, 1 \\ \cdots \\ x_n, y_n, 1 \end{bmatrix} \quad h = \begin{bmatrix} a \\ b \\ d \end{bmatrix}$$

Optimization problem: Minimize $\| \mathrm{A\,h} \|$ subject to $\| \mathrm{h} \| = 1$

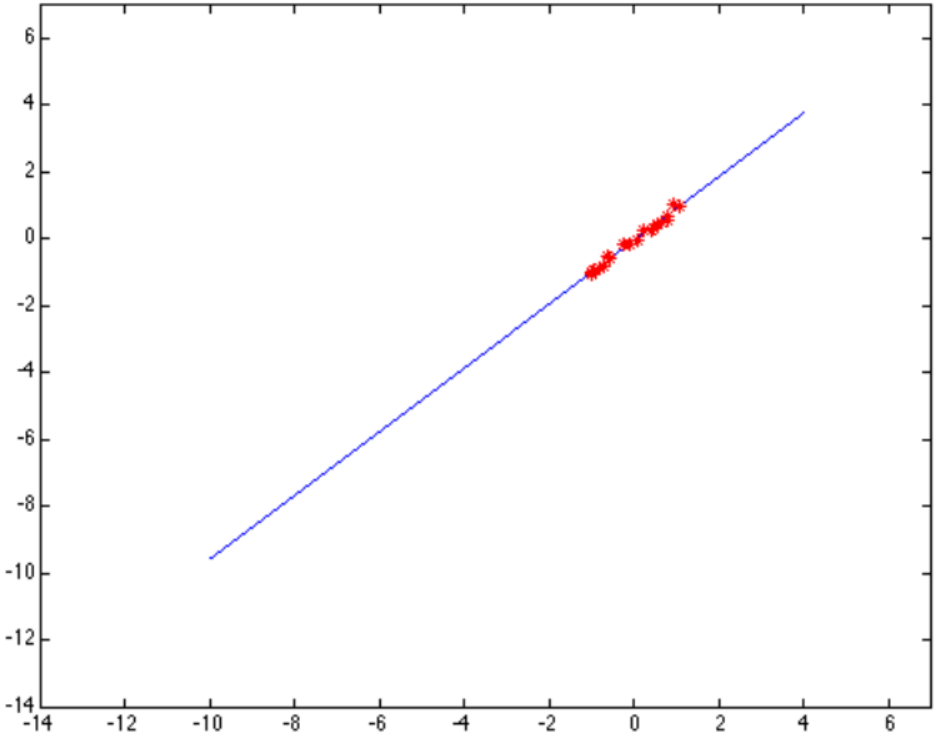
Analytical solution of h using **SVD**: $A_{n\times 3} = U_{n\times n} D_{n\times 3} V^T_{3\times 3}$

where $U_{n\times n}$ and $V_{3\times 3}$ are orthogonal matrices ($V^T V = I_{3\times 3}$ and $V = [c_1, c_2, c_3]$),

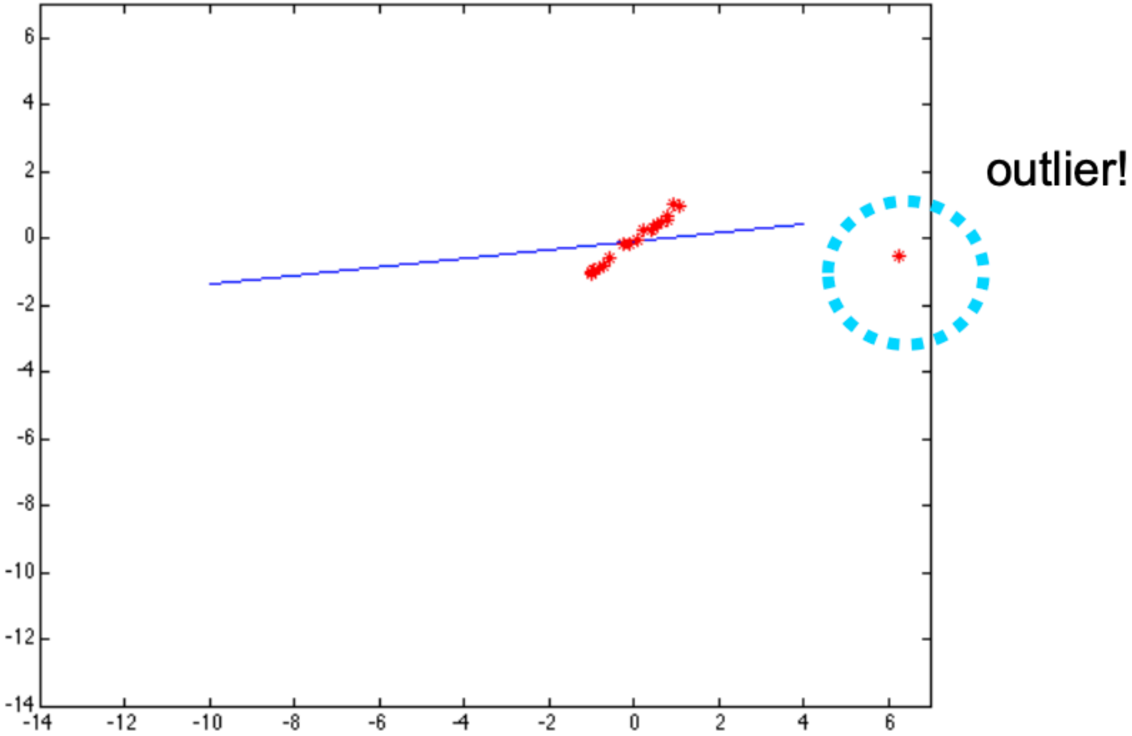
$$D = \begin{bmatrix} diag\{\lambda_1, \lambda_2, \lambda_3\} \\ O \end{bmatrix}$$ and $|\lambda_1| > |\lambda_2| > |\lambda_3|$.

Final solution: $\boxed{h = c_3 \quad \text{(last column of } V\text{)}}$

Robustness



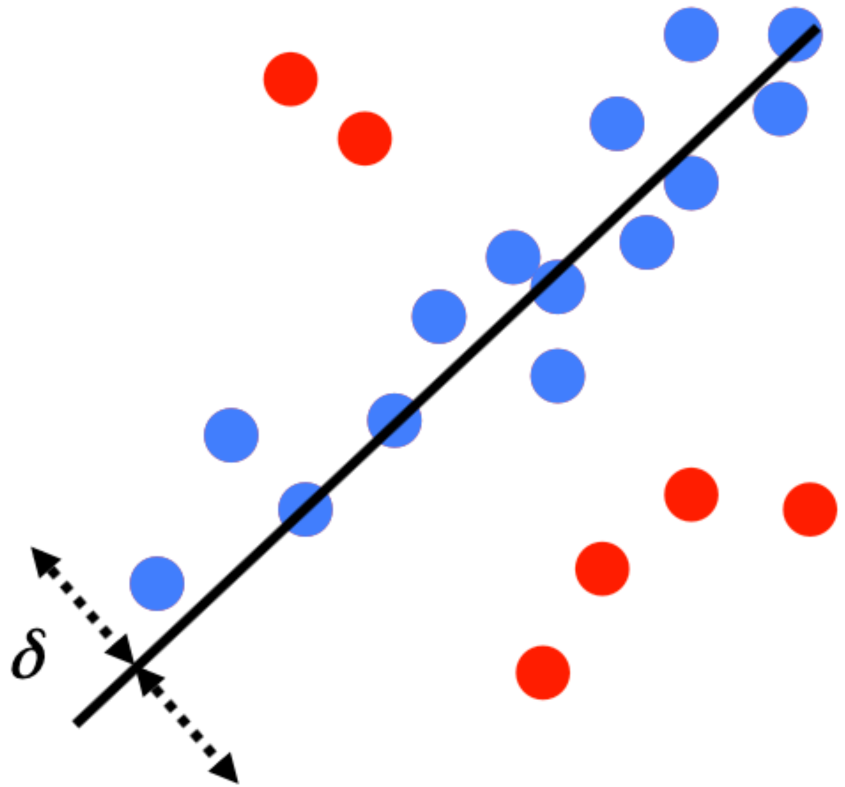
Robust to small noises.



Sensitive to outliers.

RANSAC: RANdom SAmple Consensus

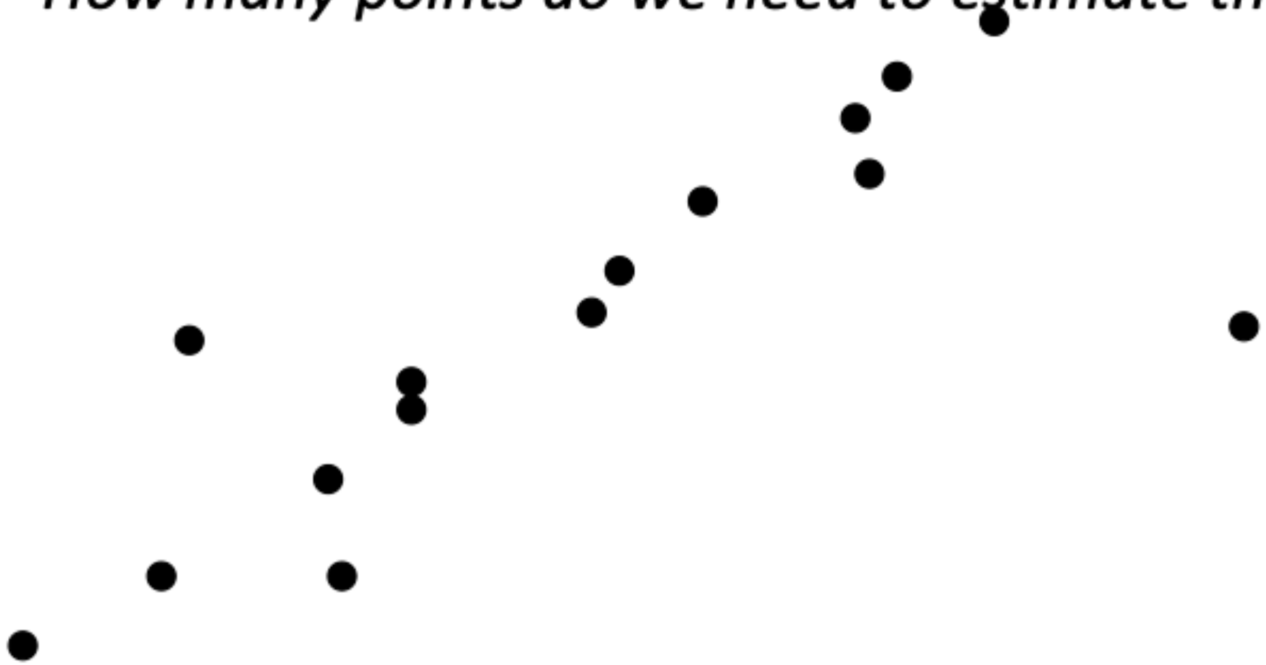
- Idea: we need to find a line that has the largest supporters (or inliers)



Fischler & Bolles in '81.

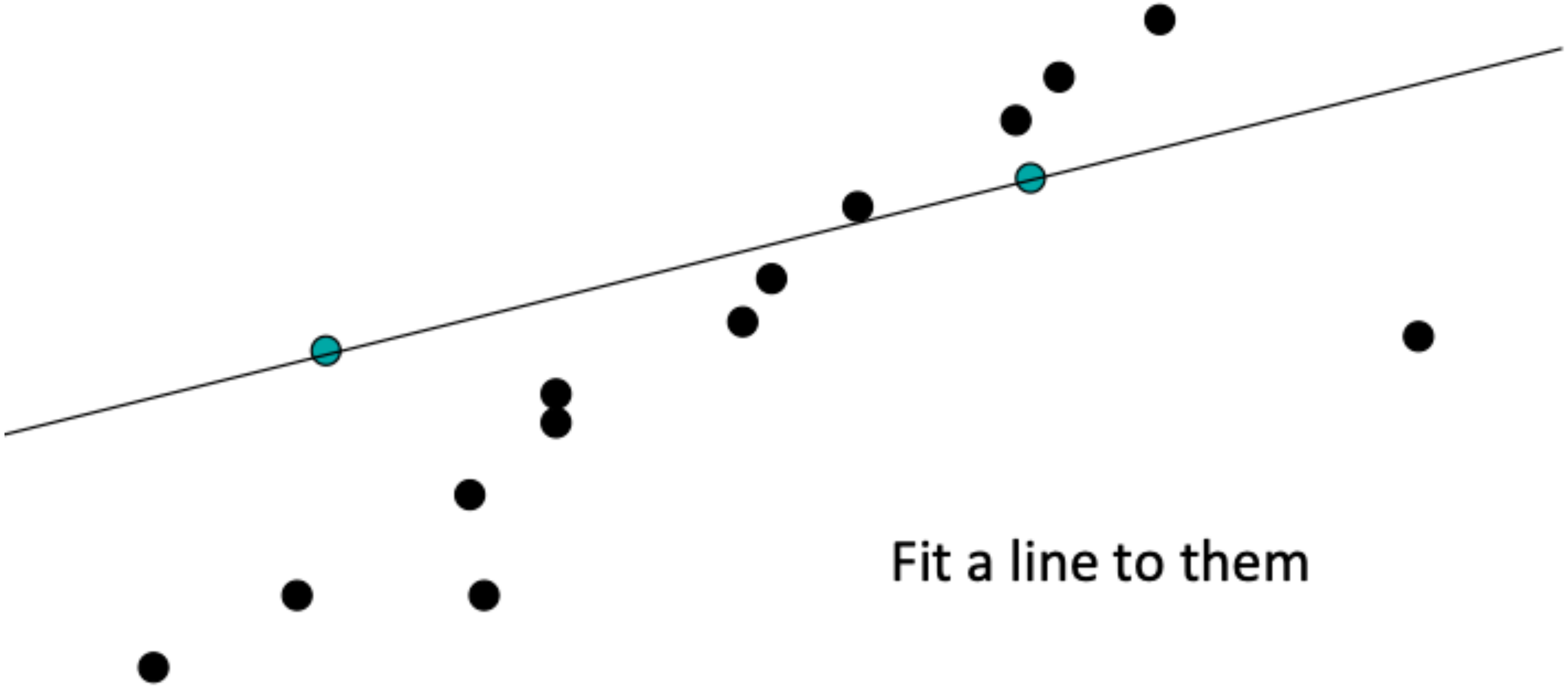
RANSAC Line Fitting

- Task: Estimate the best line
  - *How many points do we need to estimate the line?*



RANSAC Line Fitting

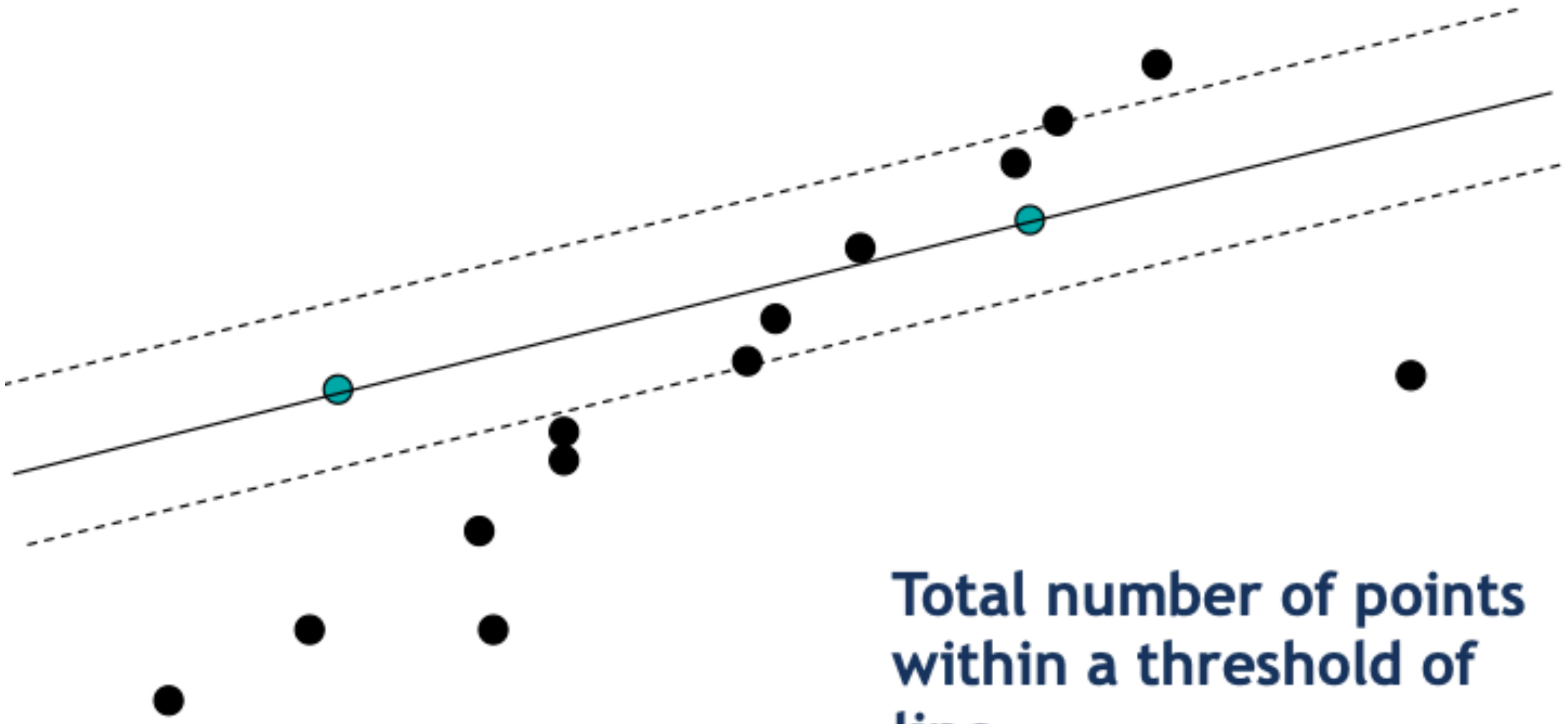
- Task: Estimate the best line



Fit a line to them

RANSAC Line Fitting

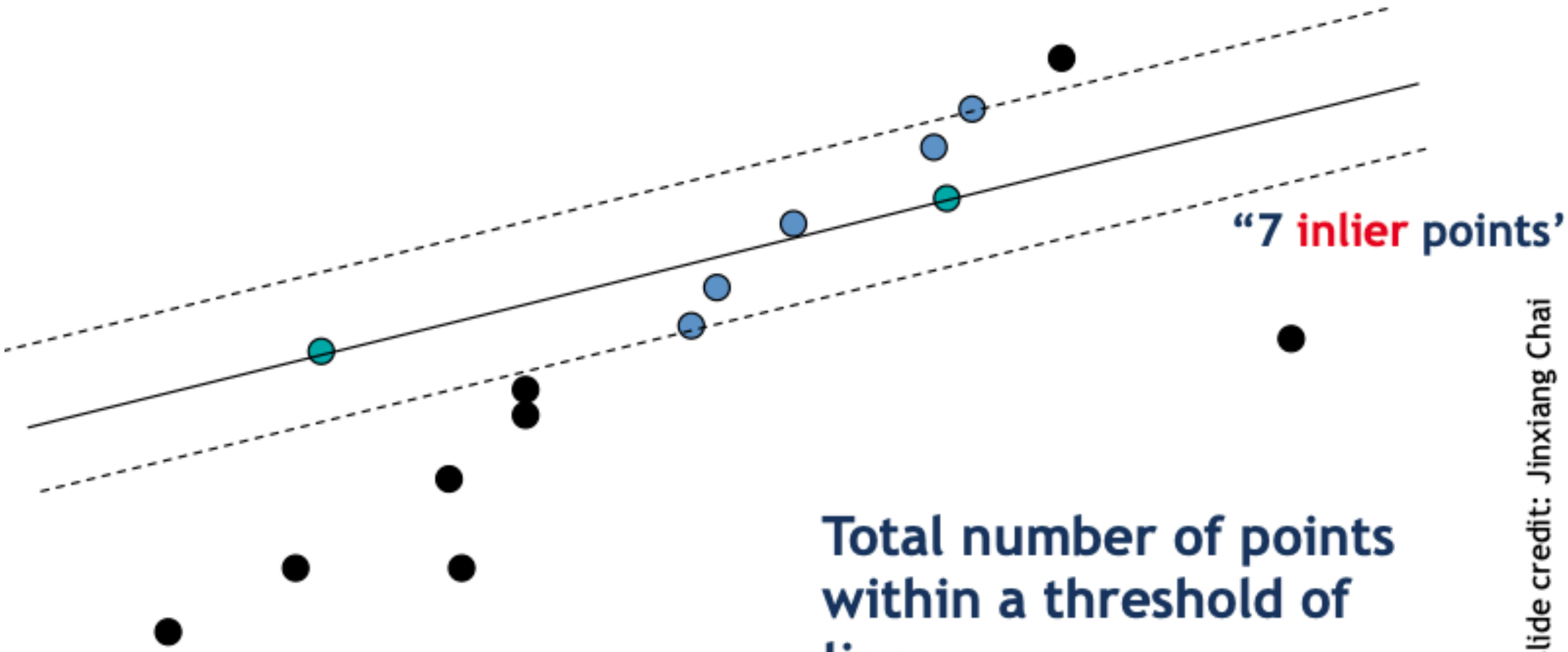
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC Line Fitting

- Task: Estimate the best line

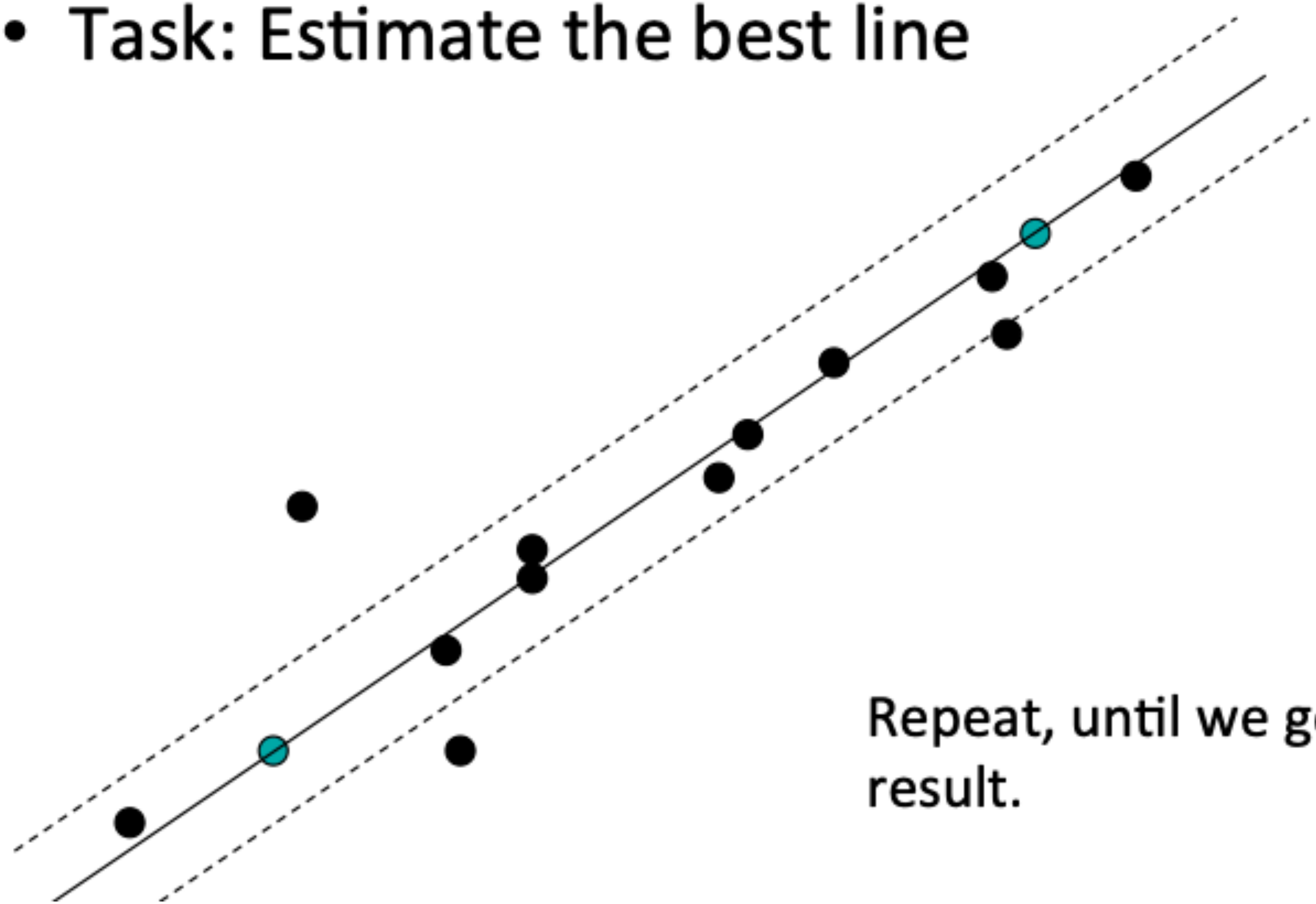


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC Line Fitting

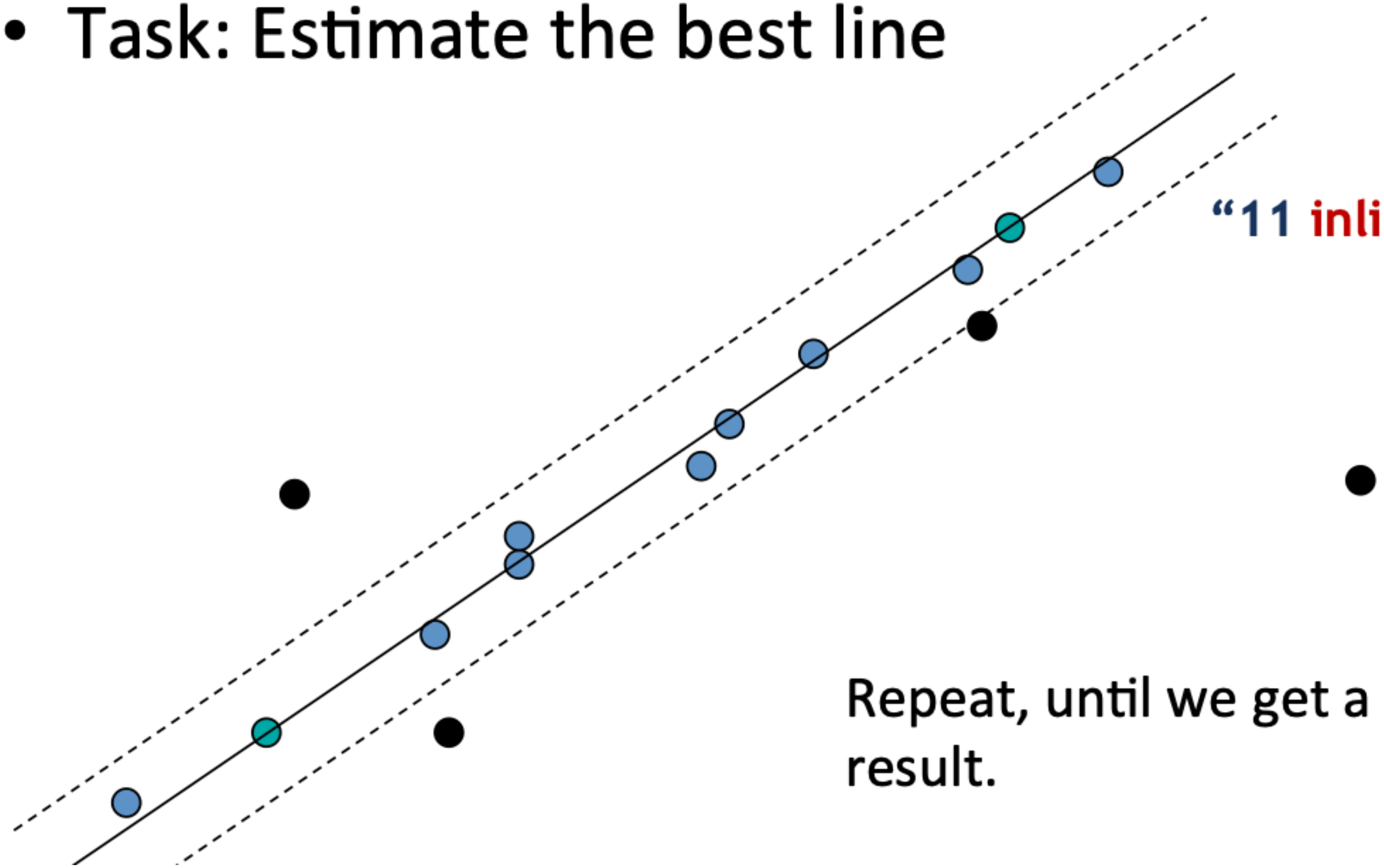
- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC Line Fitting

- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC Line Fitting

<u>RANSAC loop</u>:

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

This is a sequential version, you should implement a parallel version.

RANSAC: How Many Samples?

- How many samples are needed?
  - Suppose w is fraction of inliers (points from line).
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen.
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$

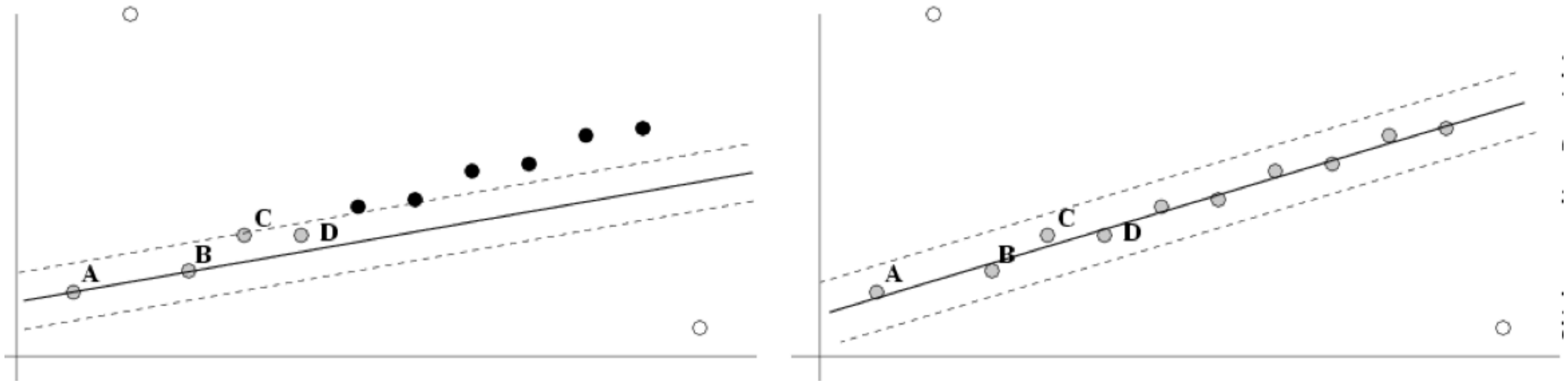
$\Rightarrow$ Choose the minimal n for solving a hypothesis

$\Rightarrow$ Choose k high enough to keep the prob. below a desired failure rate

Edited from David Lowe

After RANSAC

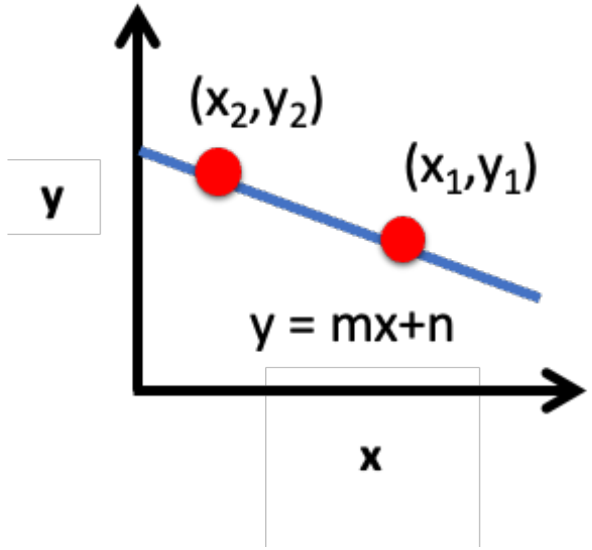
- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



RANSAC: Pro and Con

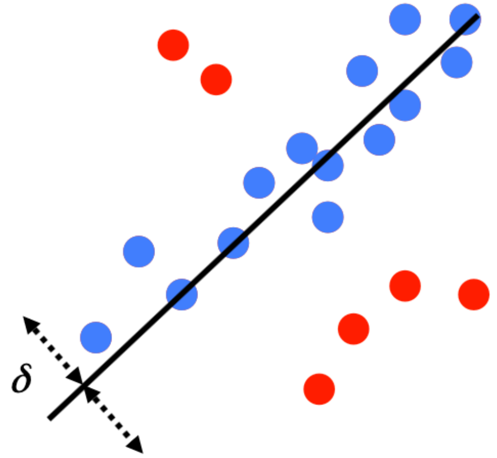
- **Pros:**
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons:**
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- A voting strategy, The Hough transform, can handle high percentage of outliers

From the Perspective of Voting



Given points in the vector space, find (m,n) in the parameter space

RANSAC

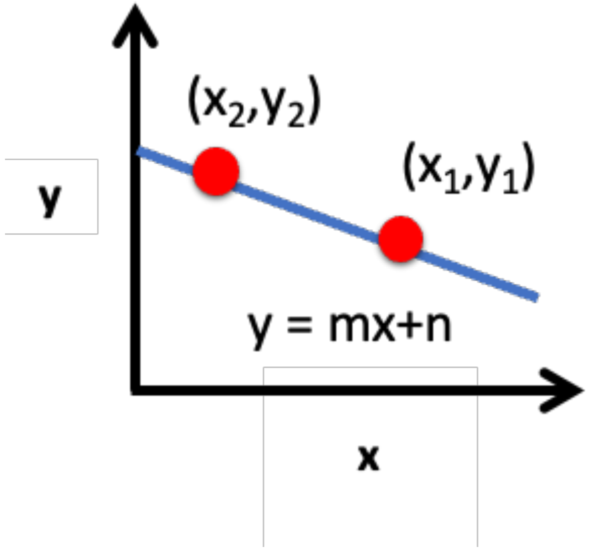


Define a inlier threshold distance in the vector space, each point votes for the best hypothesis.

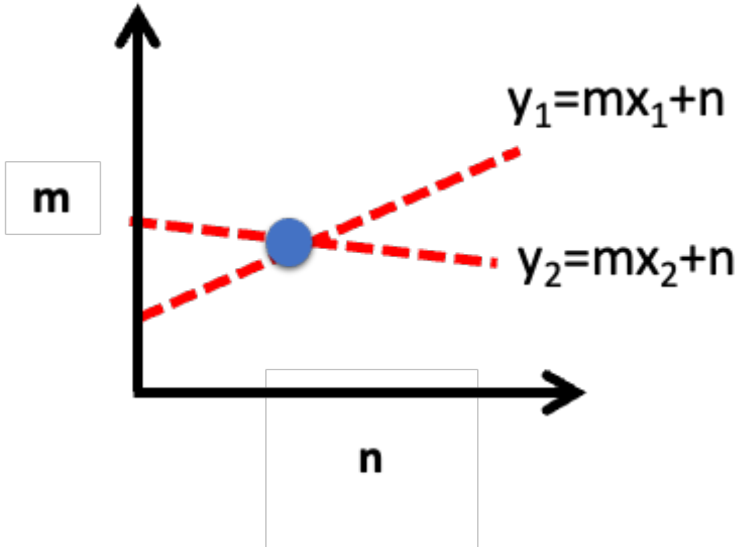
Read by Yourself

Hough Transform

Original space



Hough space

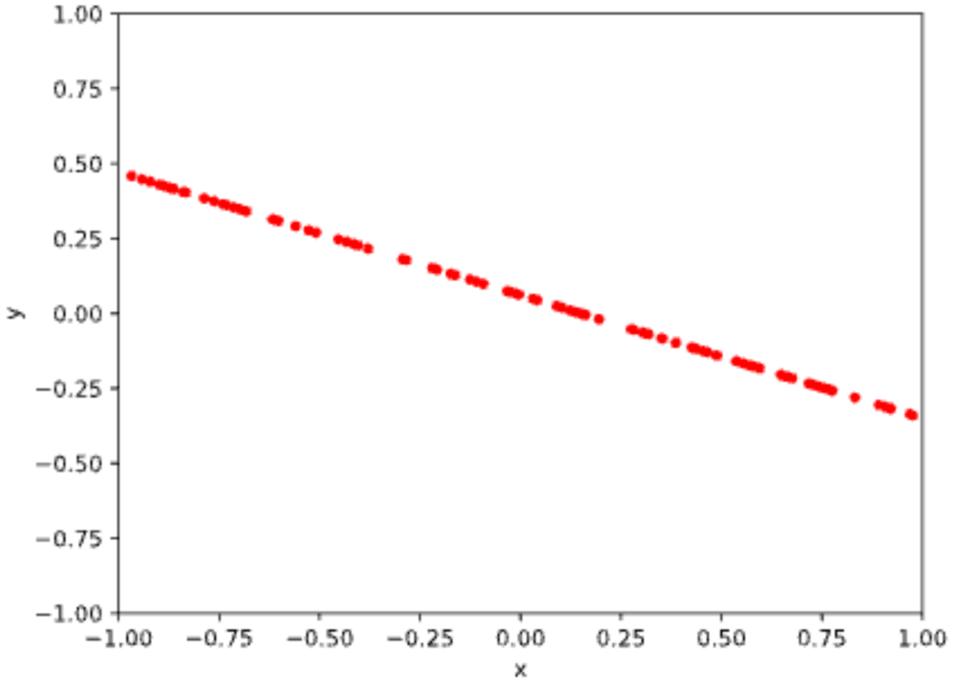


Given points in the vector space, find (m,n) in the parameter space

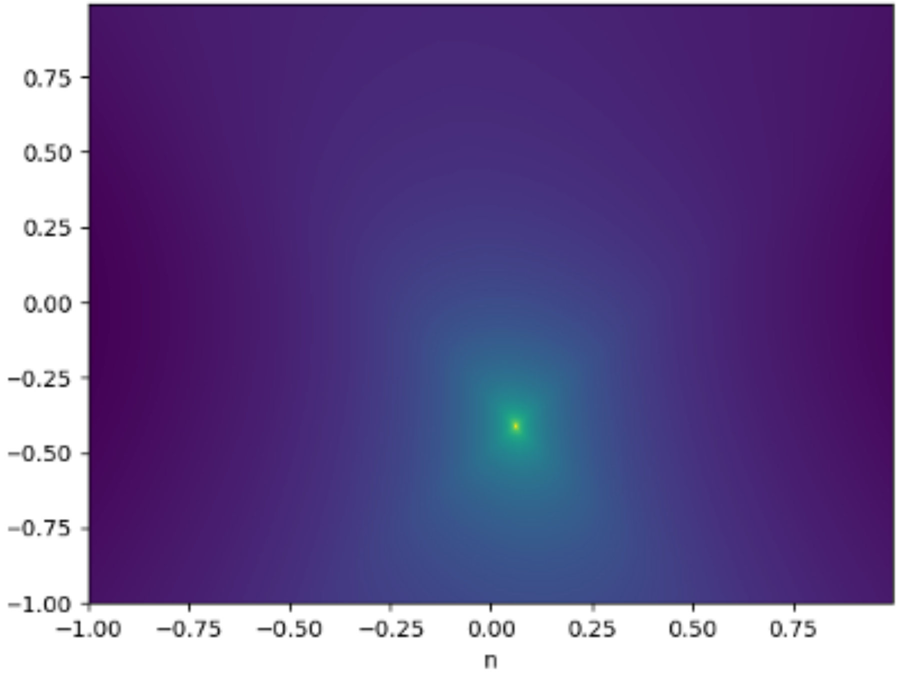
Read by Yourself

The intersection in the parameter space is (m, n)

Hough Transform w/o Noise



Original space

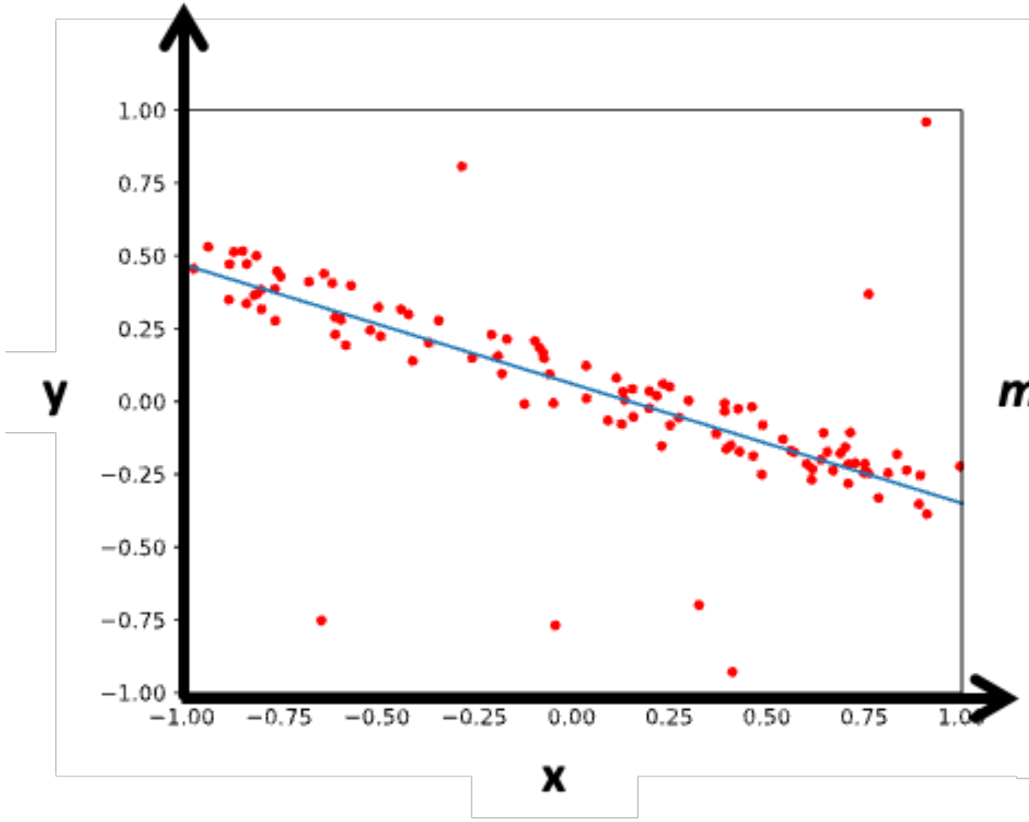


Hough space

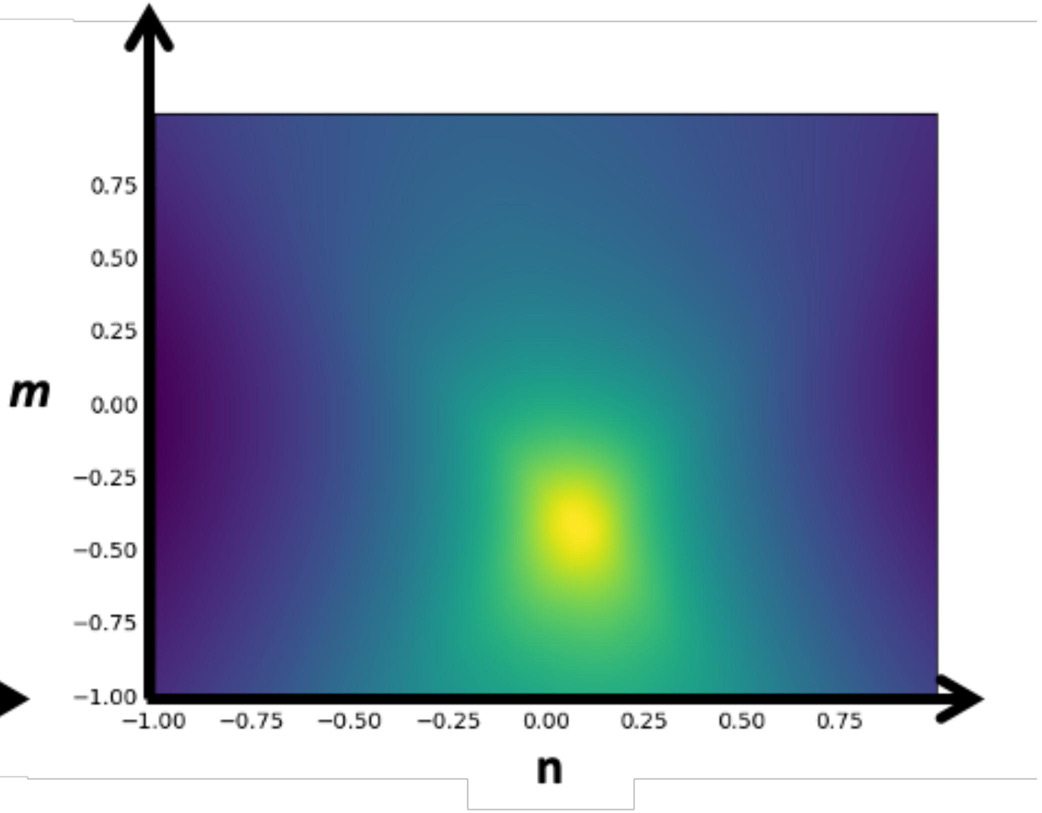
Ground truth: y= -0.4106 x + 0.0612
Fitted result: y = -0.412 x + 0.060

Read by Yourself

Hough Transform w/ Noise and Outliers



Original space



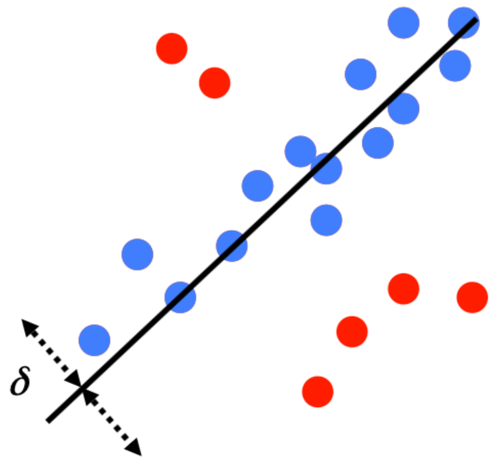
Hough space

Ground truth: y= -0.4106 x + 0.0612
Fitted result: y = -0.412 x + 0.076

Read by Yourself

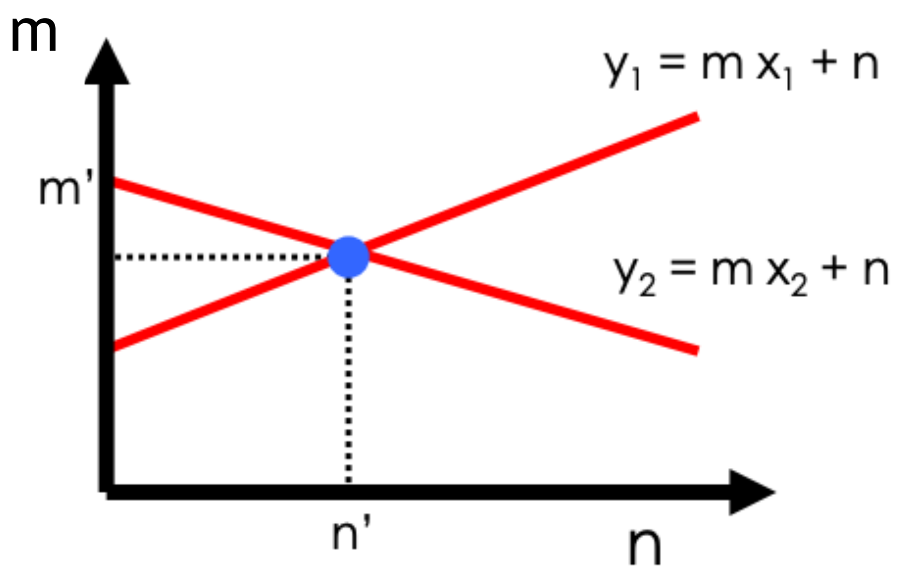
From the Perspective of Voting

RANSAC



Voting in the original space

Hough transform



Voting in the parameter space

Read by Yourself

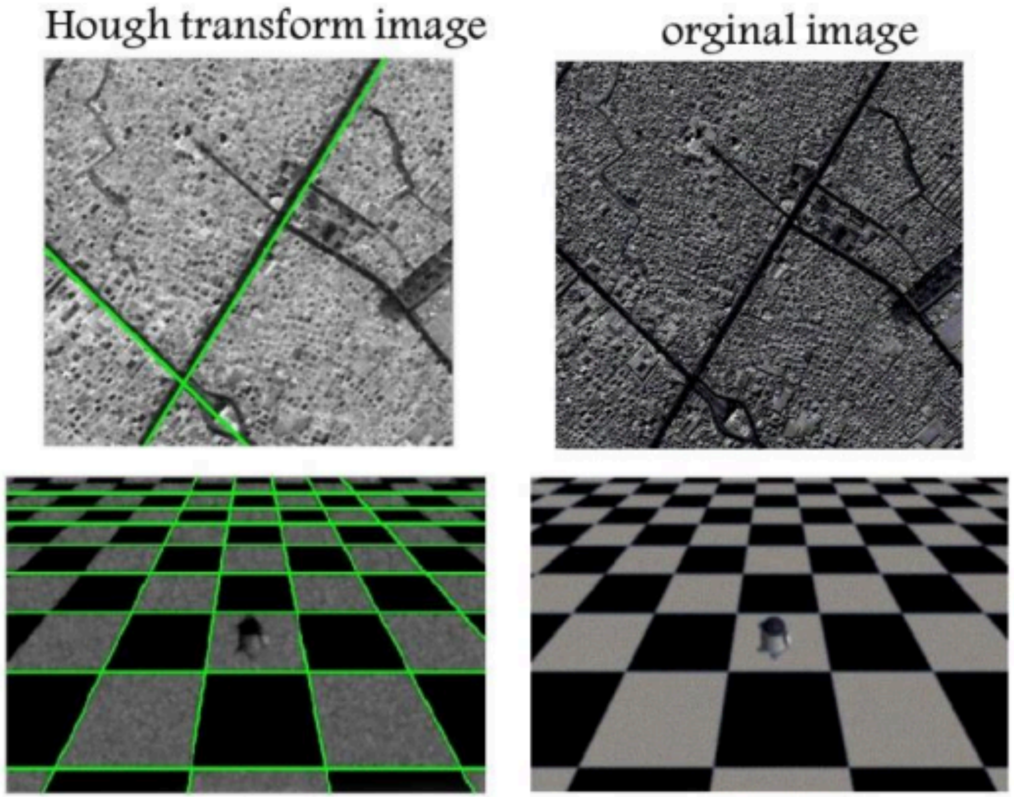
Robust Fitting：RANSAC vs. Hough Transform

RANSAC

- Single mode: robust for outliers

Hough Transform

- Less robust compared to RANSAC (spurious peak)
- Can handle multiple modes well



Read by Yourself

Parsa, Younes, Hasan Hosseinzadeh, and Mehdi Effatparvar. "Development Hough transform to detect straight lines using pre-processing filter." *International Journal of Information, Security and Systems Management* 4.2 (2015): 448-456.

Summary of Line Detection

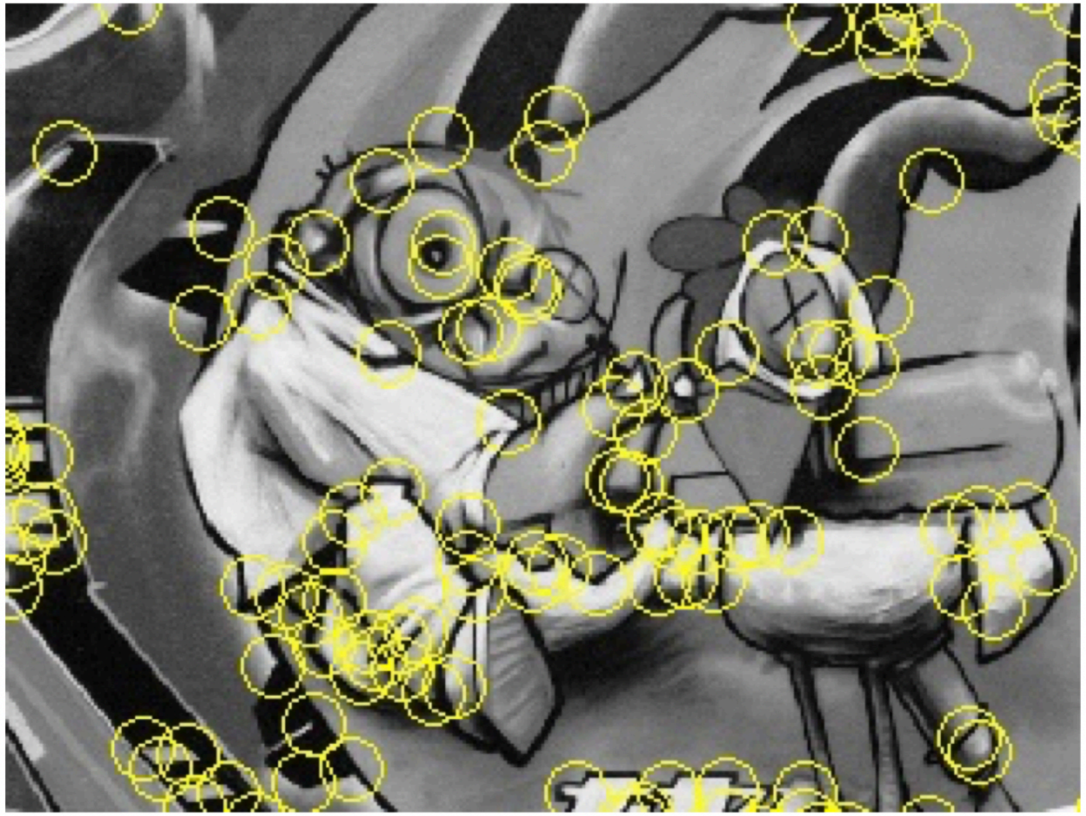
- A modular based approach: gradient -> edge -> line
- Need high robustness for every module, *e.g.*, denoising in gradient image, robust line fitting



Corner Detection

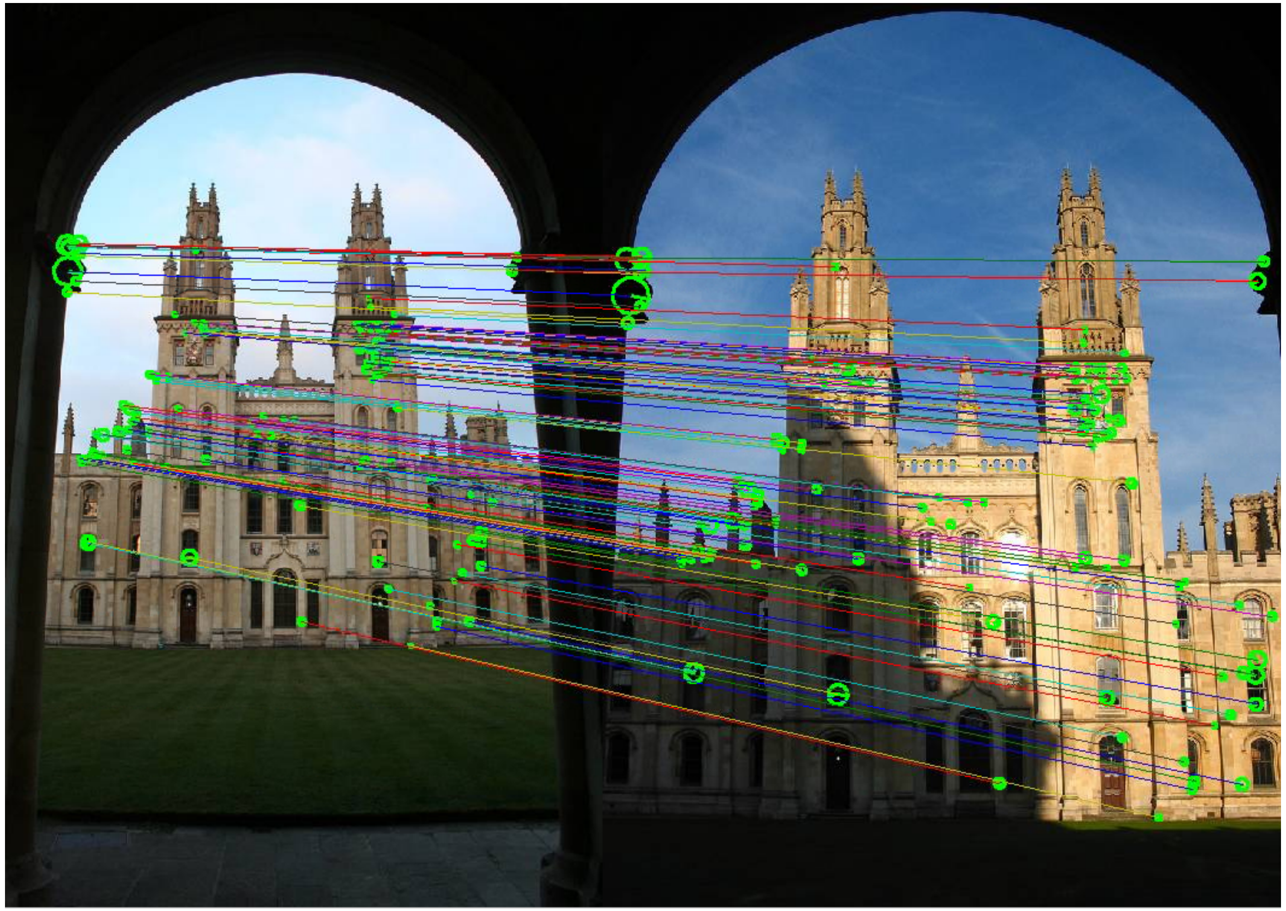
Some slides are borrowed from Stanford CS131.

Keypoint Localization



- In addition to edges, keypoints are also important to detect.

Applications: Image Matching



Separately detect keypoints and then find matching.

What Points are Keypoints?

- Saliency: interesting points

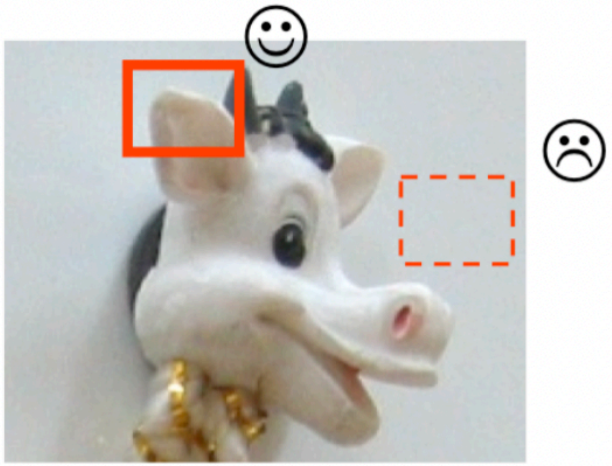
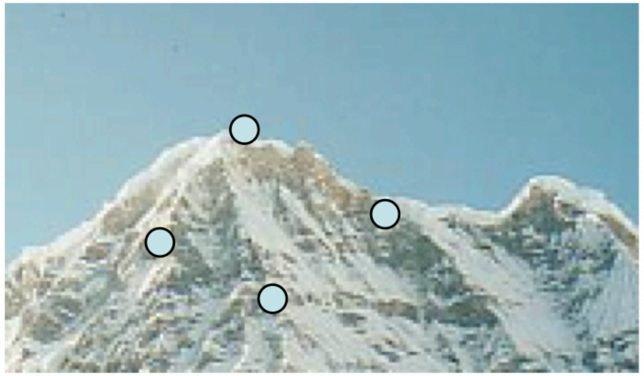


Image borrowed from Stanford CS131

More Requirements

- Saliency: interesting points
- Repeatability: detect the same point independently in both images



No chance to match!

Image borrowed from Stanford CS131

More Requirements

- Repeatability: detect the same point independently in both images
- Saliency: interesting points
- Accurate localization

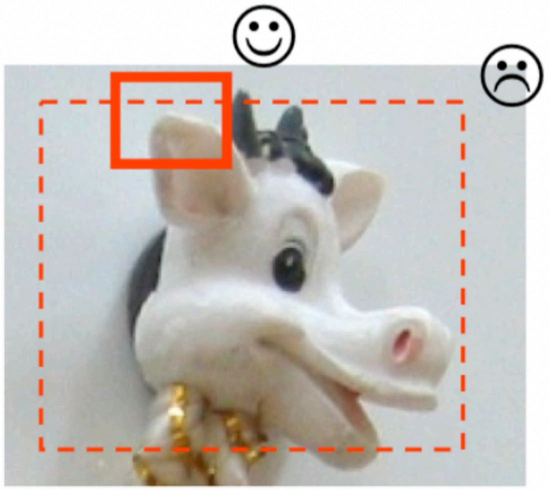
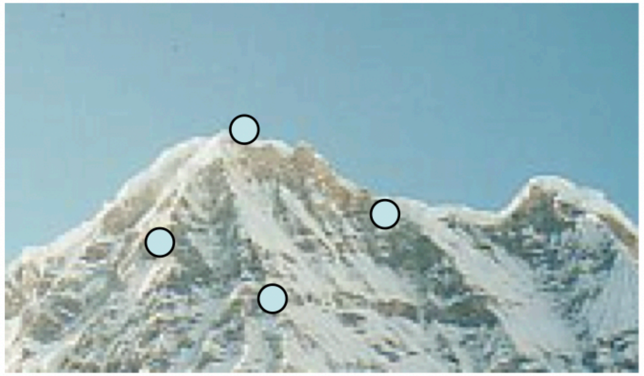


Image borrowed from Stanford CS131

More Requirements

- Repeatability: detect the same point independently in both images
- Saliency: interesting points
- Accurate localization
- Quantity: sufficient number



No chance to match!

Image borrowed from Stanford CS131

Repeatability and Invariance

- For a keypoint detector to be repeatable, it has to be invariant to:
  - Illumination
  - Image scale
  - Viewpoint

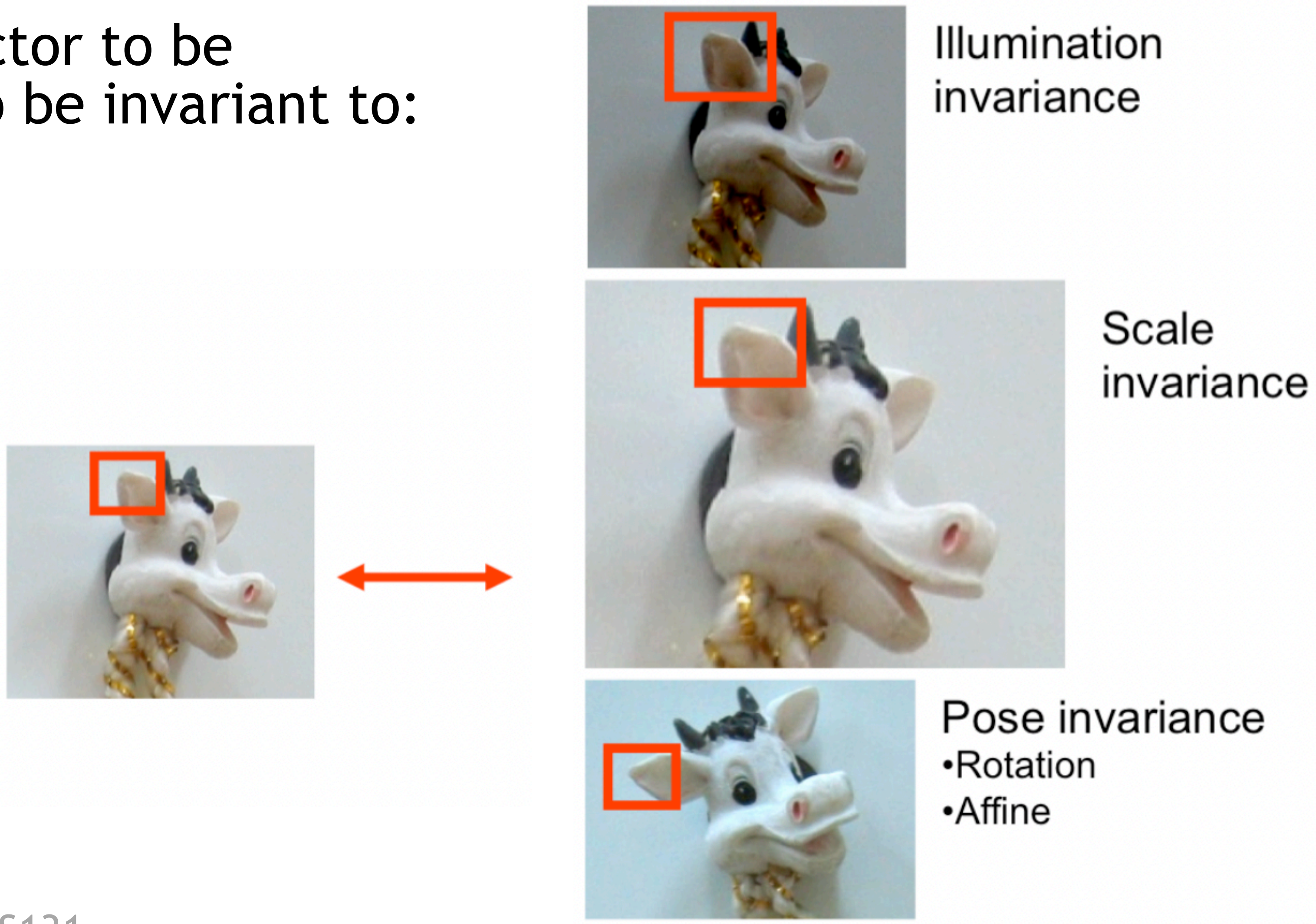
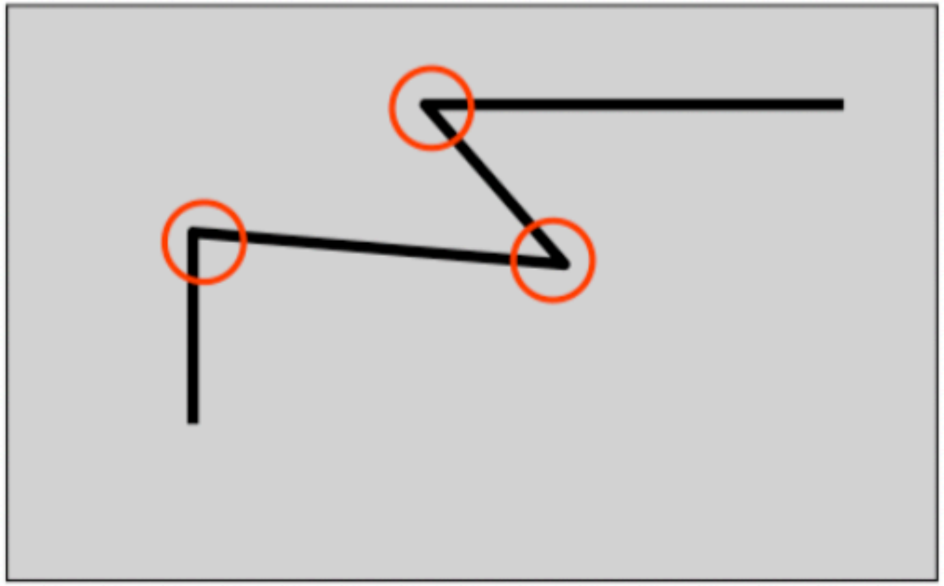


Image borrowed from Stanford CS131

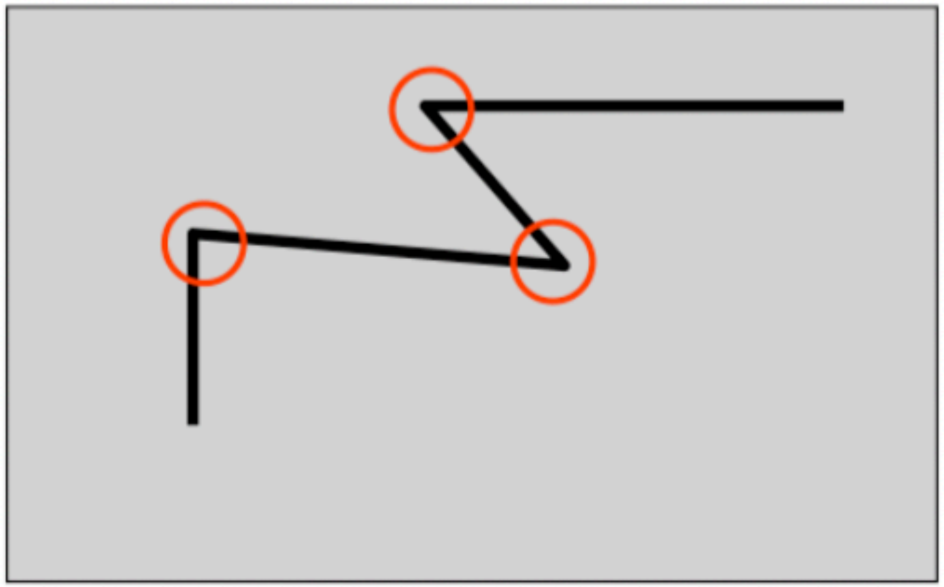
Corners as Keypoints



- Corners are such kind of keypoints, because they are
  - Salient;
  - Repeatable (one corner would still be a corner from another viewpoint);
  - Sufficient (usually an image comes with a lot of corners);
  - Easy to localize.

Image borrowed from Stanford CS131

The Properties of a Corner

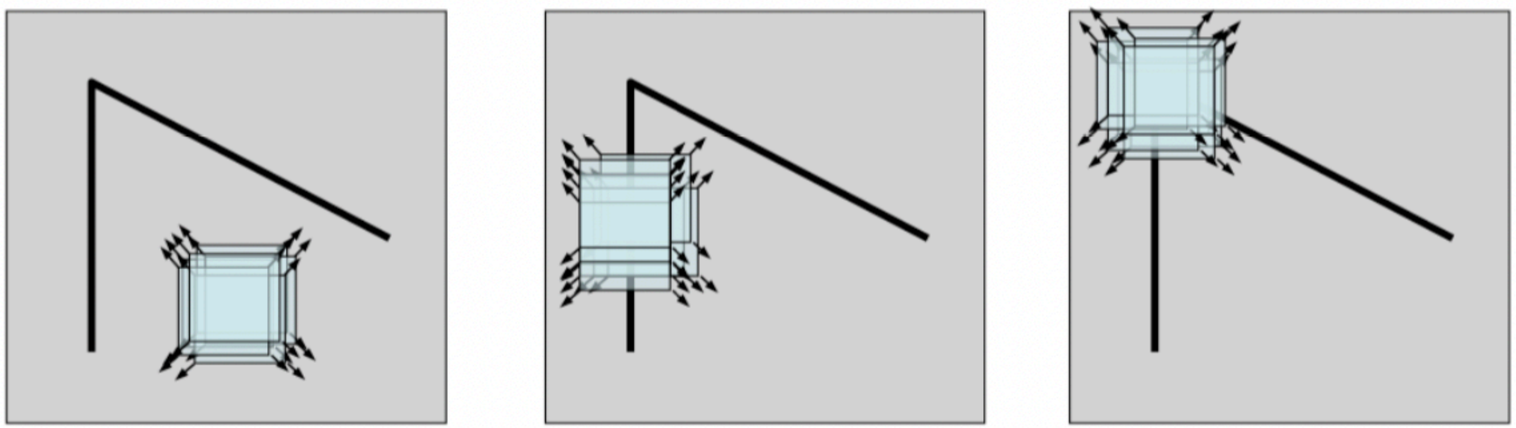


- The key property of a corner: In the region around a corner, image gradient has two or more dominant directions

Image borrowed from Stanford CS131

The Basic Idea of Harris Corner

- Move a window and explore intensity changes within the window



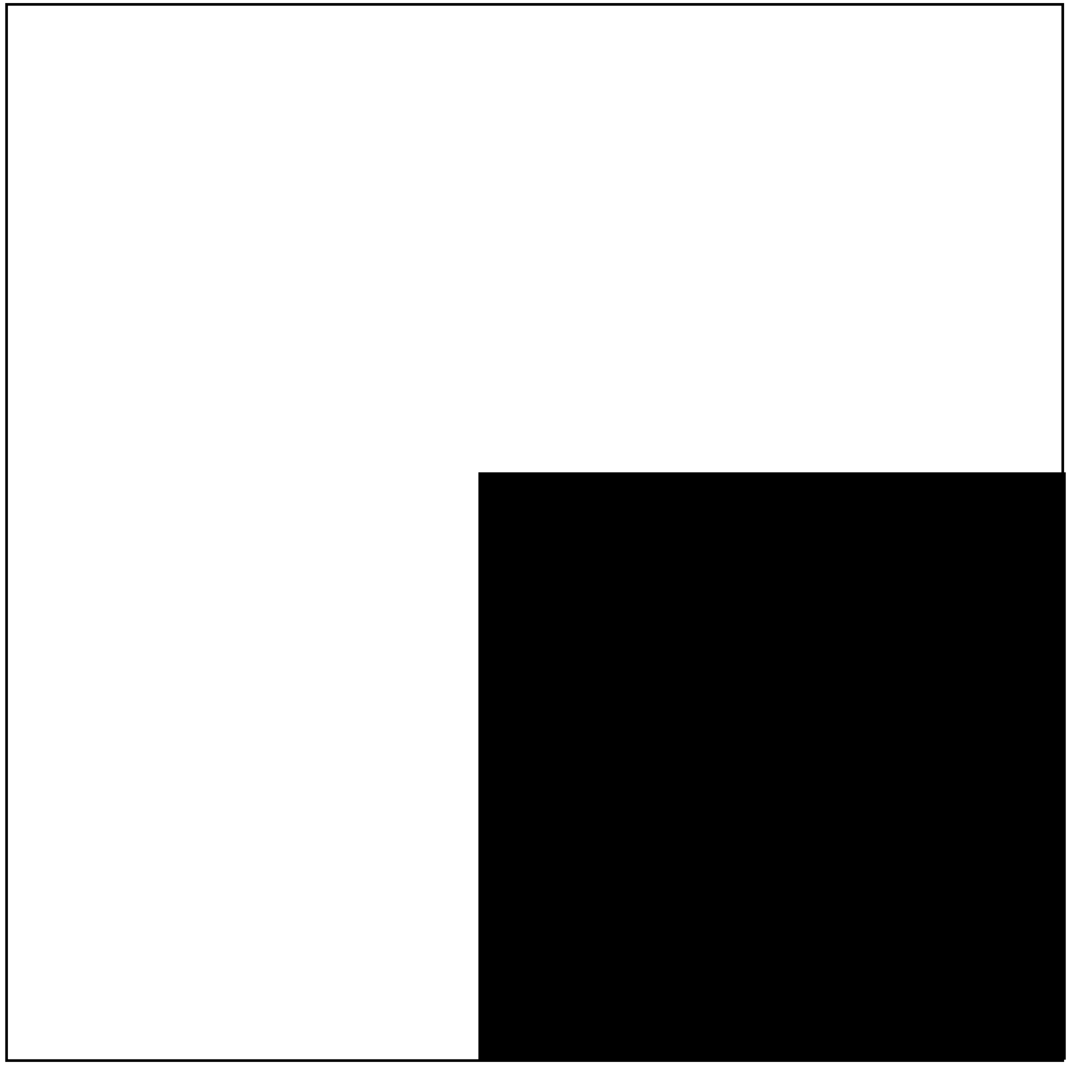
Flat region: no change in all directions

Edge: no change along the edge direction

Corner: significant change in all directions

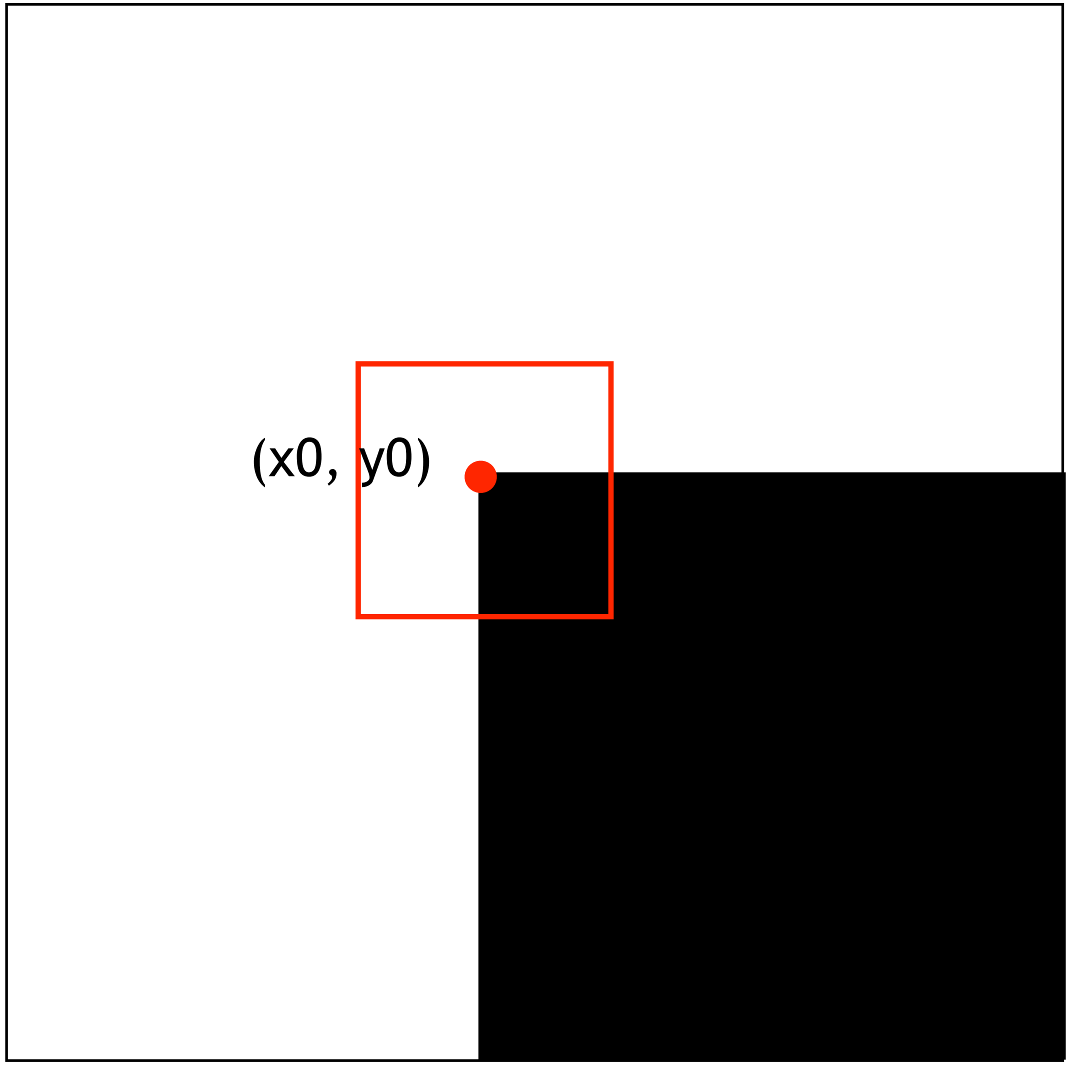
Image borrowed from Stanford CS131

The Basic Idea of Harris Corner



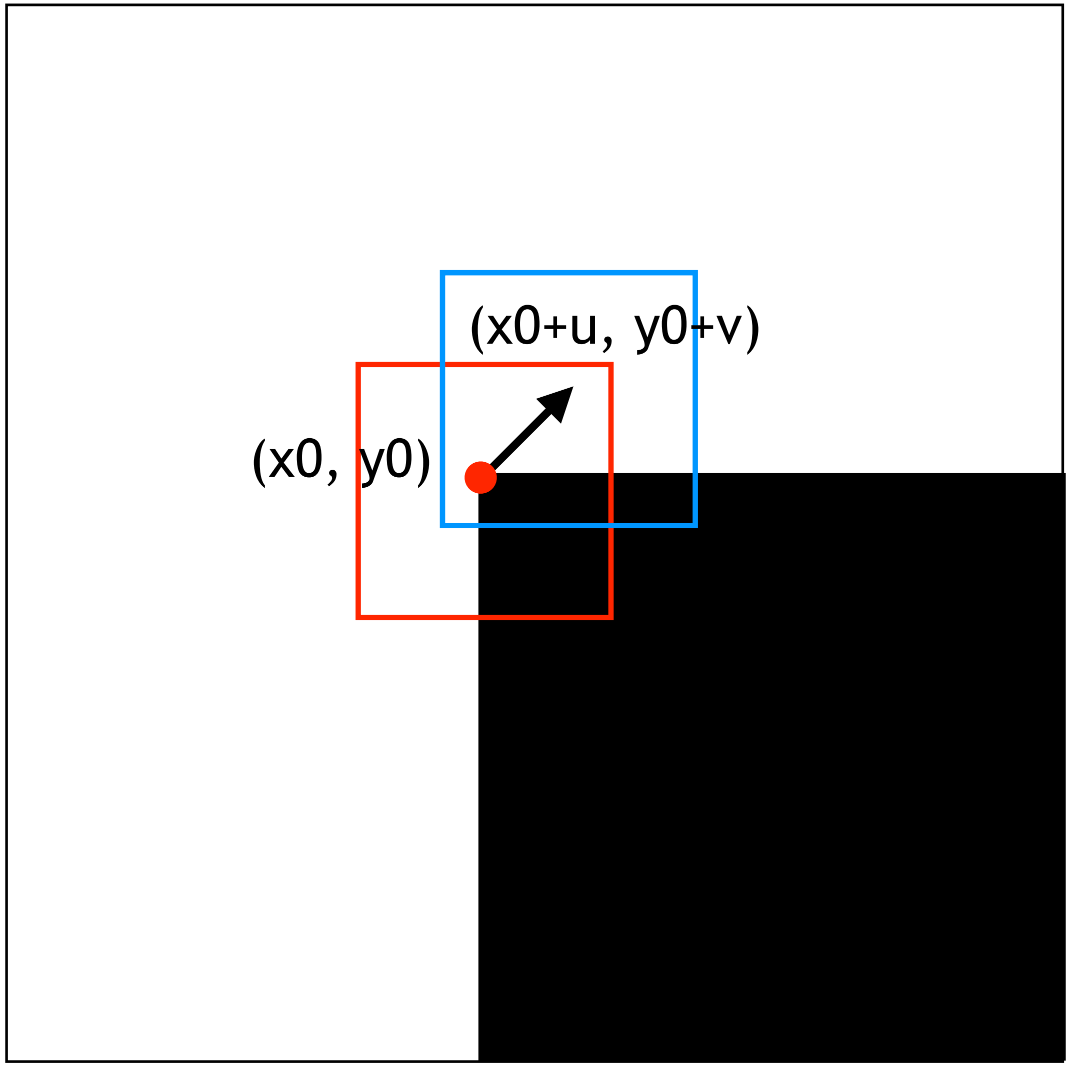
Original image

The Basic Idea of Harris Corner



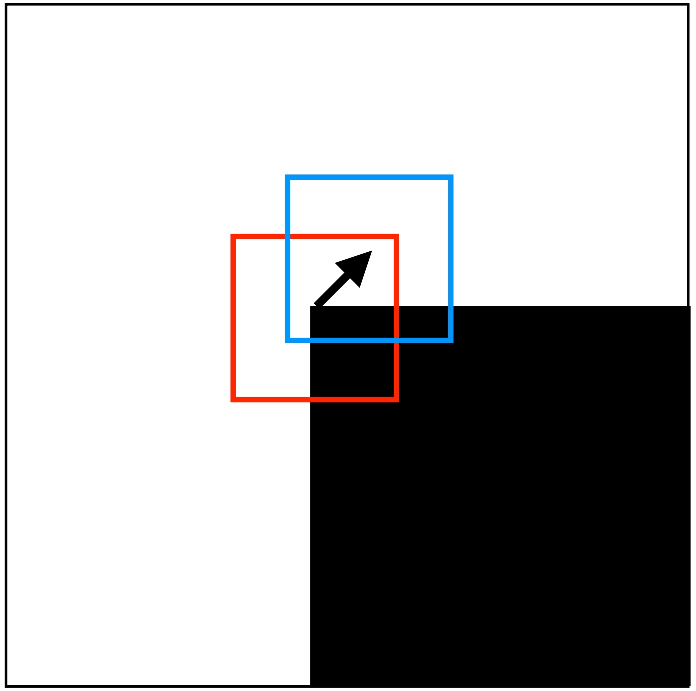
Local neighborhood of a corner point (x0, y0)

The Basic Idea of Harris Corner



Move the window by (u, v)

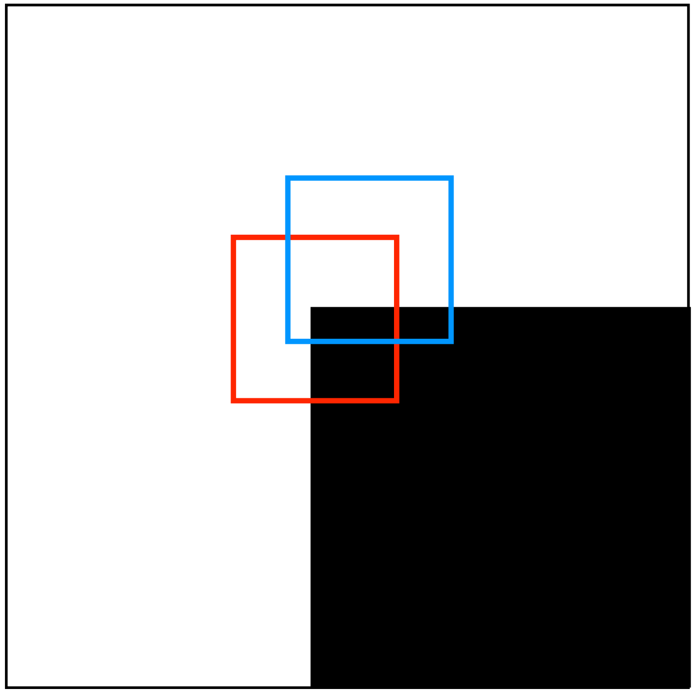
The Basic Idea of Harris Corner



Local neighborhood of a corner point (x0, y0)

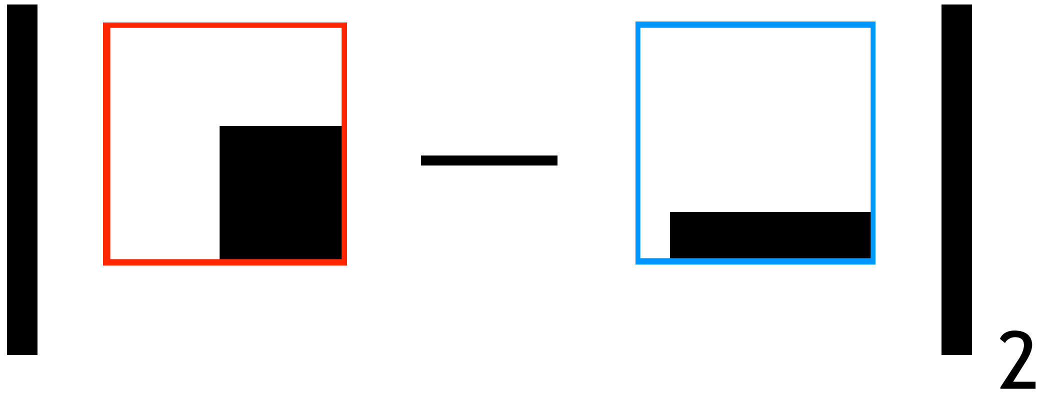
Local neighborhood of point (x0+u, y0+v)

The Basic Idea of Harris Corner



After moving (u, v), the squared difference within the window

$E_{x_0,y_0}(u, v)=$



$$= \sum_{(x,y)\in N} [I(x+u, y+v) - I(x,y)]^2$$

Where N is the neighborhood of (x0, y0)

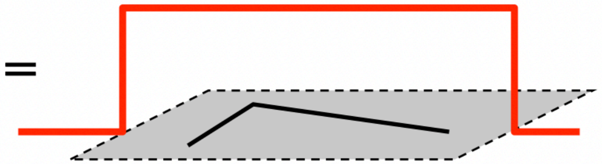
Notation

Square intensity difference

$$D_{u,v}(x, y) = [I(x + u, y + v) - I(x, y)]^2$$

Rectangle Window Function

$w(x,y) =$



1 in window, 0 outside

$$= \begin{cases} 1, & if \quad -b < x, y < b \\ 0, & else\,. \end{cases}$$

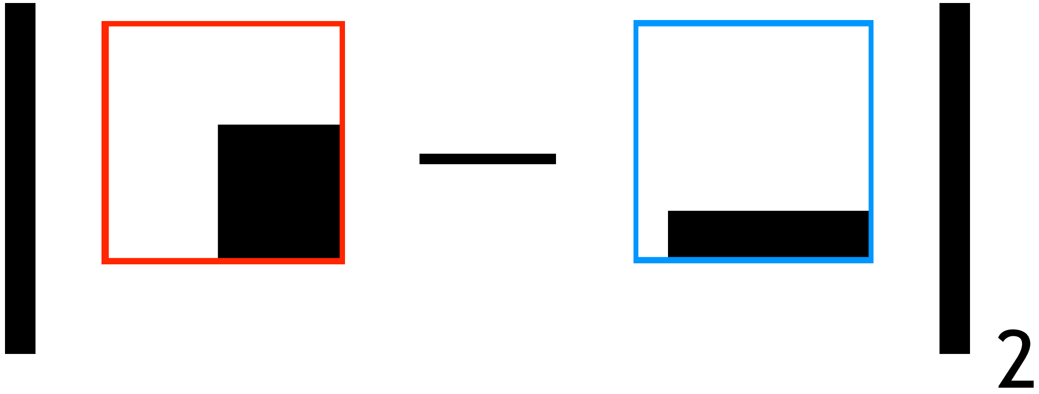
(b: half-width of the window)

Rectangle window function
when the center is at (x_0, y_0)

$$w'_{x_0,y_0}(x, y) = w(x - x_0, y - y_0)$$

The Basic Idea of Harris Corner Detector

$E_{x_0,y_0}(u, v) =$



$$= \sum_{(x,y)\in N} [I(x+u, y+v) - I(x, y)]^2$$

$$= \sum_{x,y} w'_{x_0,y_0}(x, y)[I(x+u, y+v) - I(x, y)]^2 = \sum_{x,y} w'_{x_0,y_0}(x, y) D_{u,v}(x, y)$$

$$= \sum_{x,y} w(x - x_0, y - y_0) D_{u,v}(x, y) = \sum_{x,y} w(x_0 - x, y_0 - y) D_{u,v}(x, y)$$

$$= w * D_{u,v}$$

Harris Detector

Since u and v are both small, we apply first-order Taylor expansion:

$$I[x+u, y+v] - I[x, y] \approx I_x u + I_y v$$

$$\therefore D_{u,v}(x, y) = (I[x+u, y+v] - I[x, y])^2 \approx (I_x u + I_y v)^2 = [u, v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\therefore E_{x_0, y_0}(u, v) = w * D_{u,v} = [u, v] \; \underbrace{w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}}_{} \begin{bmatrix} u \\ v \end{bmatrix}$$

A function of x_0, y_0

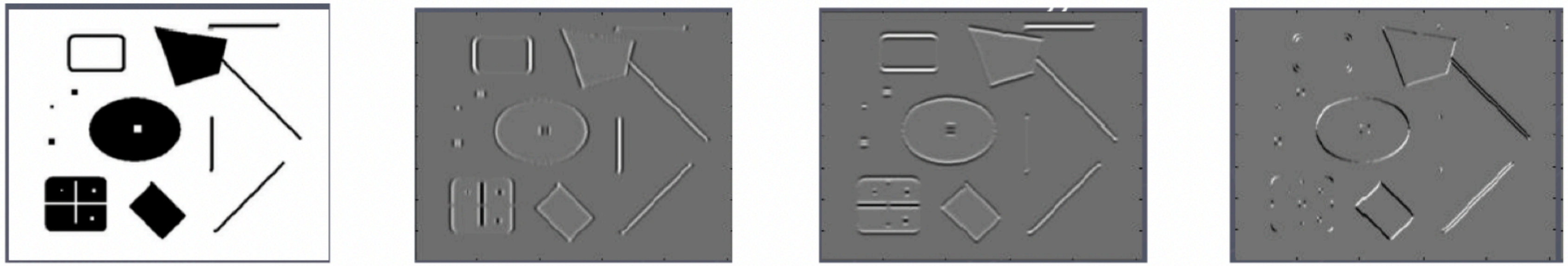


Image I $\quad$ I_x $\quad$ I_y $\quad$ $I_x I_y$

Image borrowed from Stanford CS131

Harris Detector

If we are checking the corner at (x_0, y_0), then the energy after moving the window by (u, v) is:

$$E_{(x_0,y_0)}(u, v) \approx [u, v]\ M(x_0, y_0)\ \begin{bmatrix} u \\ v \end{bmatrix}$$

where $$M(x, y) = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} w * I_x^2 & w * (I_x I_y) \\ w * (I_x I_y) & w * I_y^2 \end{bmatrix}$$

Image borrowed from Stanford CS131

Harris Detector

$$M(x, y) = w * \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} w * I_x^2 & w * (I_xI_y) \\ w * (I_xI_y) & w * I_y^2 \end{bmatrix}$$

- M is a symmetric matrix.
- M is a positive semi-definite matrix (since all its principle minors ≥ 0).
- Simple case: $M(x_0, y_0)$: is diagonal $\quad M(x_0, y_0) = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \quad (\lambda_1 \geq 0,\ \lambda_2 \geq 0)$

$$\therefore E_{(x_0, y_0)}(u, v) \approx [u, v]\ M(x_0, y_0) \begin{bmatrix} u \\ v \end{bmatrix} = \lambda_1 u^2 + \lambda_2 v^2$$

- This corresponds to an axis-aligned corner.
- If either $\lambda \approx 0$, this is not a corner.

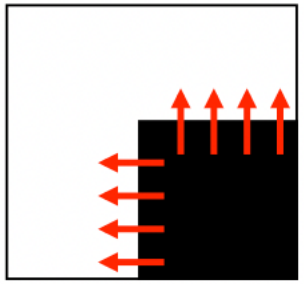


Image borrowed from Stanford CS131

Harris Detector

- General case:

since M is a symmetric square matrix, perform eigenvalue-decomposition:

$$M(x,y) = \begin{bmatrix} w * I_x^2 & w * (I_x I_y) \\ w * (I_x I_y) & w * I_y^2 \end{bmatrix} = Q \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q^T \quad (\lambda_1 \geq 0, \;\; \lambda_2 \geq 0)$$

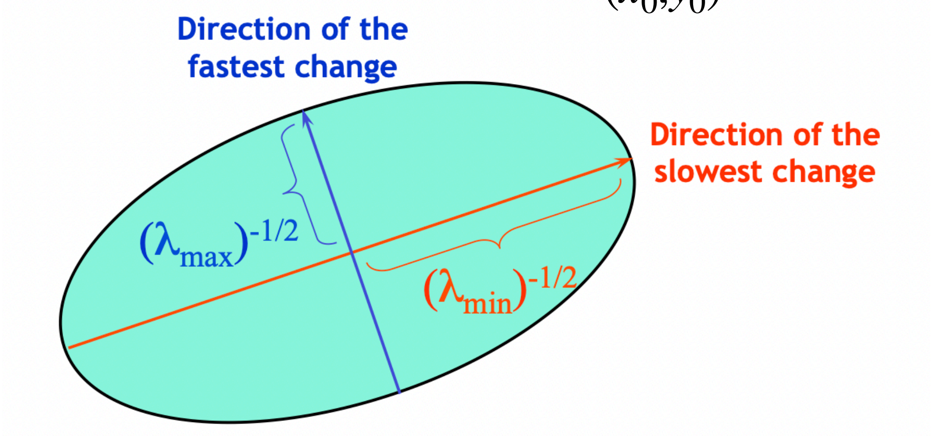
Q is an orthogonal matrix, $\{\lambda_i\}$ are the eigenvalues of M!

Harris Detector

- General case: since M is a symmetric matrix, perform eigen-decomposition:

$$M(x_0, y_0) = \begin{bmatrix} w * I_x^2 & w * (I_x I_y) \\ w * (I_x I_y) & w * I_y^2 \end{bmatrix} = Q \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} Q^T \quad (\lambda_1 \geq 0, \ \ \lambda_2 \geq 0)$$

$$\therefore E_{(x_0,y_0)}(u, v) \approx \lambda_1 u'^2 + \lambda_2 v'^2 \qquad \text{where} \quad \begin{bmatrix} u' \\ v' \end{bmatrix} = Q \begin{bmatrix} u \\ v \end{bmatrix}$$

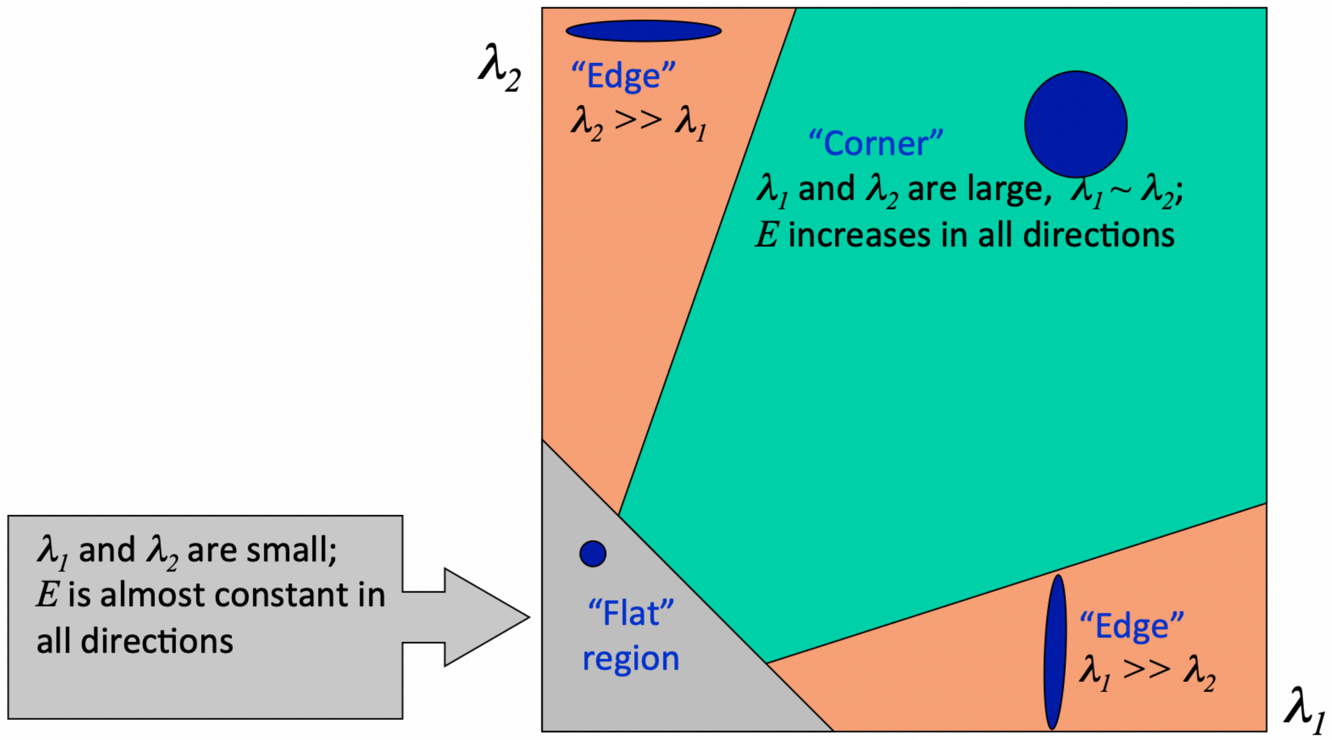


The energy landscape is a paraboloid!

Image borrowed from Stanford CS131

Eigenvalues

- Classification of the type of the image point according to the eigenvalues of M.



Two conditions must be satisfied:

$$\lambda_1, \lambda_2 > b$$

$$\frac{1}{k} < \frac{\lambda_1}{\lambda_2} < k$$

Image borrowed from Stanford CS131

Corner Response Function θ

- Fast approximation:

$\lambda_1, \lambda_2 > b \Longleftrightarrow \lambda_1\lambda_2 - 2t > 0$ and $t = b^2/2$

$$\theta = \frac{1}{2}\underline{(\lambda_1\lambda_2 - 2\alpha(\lambda_1 + \lambda_2)^2)} + \frac{1}{2}\underline{(\lambda_1\lambda_2 - 2t)}$$

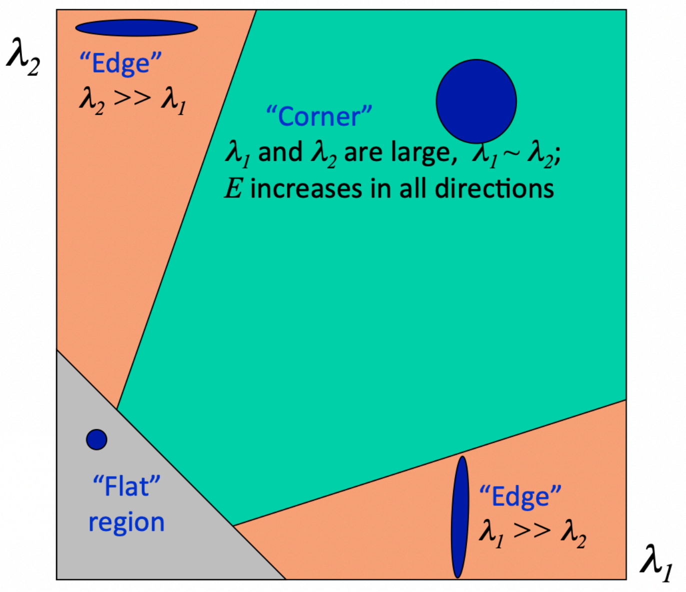
$\frac{1}{k} < \frac{\lambda_1}{\lambda_2} < k \Longleftrightarrow \lambda_1\lambda_2 - 2\alpha(\lambda_1 + \lambda_2)^2 > 0$ and $\alpha = 1/2(k + 1/k)^2$

If $k \approx 3$, then $\alpha \approx 0.045$

$$= \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 - t$$

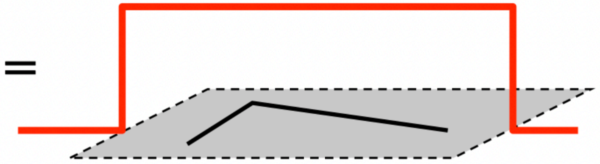
$$= \underline{det(M)} - \alpha\underline{Tr(M)}^2 - t$$

Orthogonal transformation won't change the determinant and trace of a matrix



Choices of Window Functions

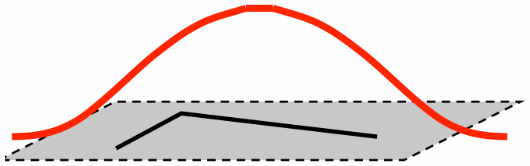
Rectangle "hard" window

$w(x,y) =$ 

1 in window, 0 outside

or

Isotropic "soft" window

$g_\sigma(x, y) =$ 

Gaussian

w is not rotation-invariant.

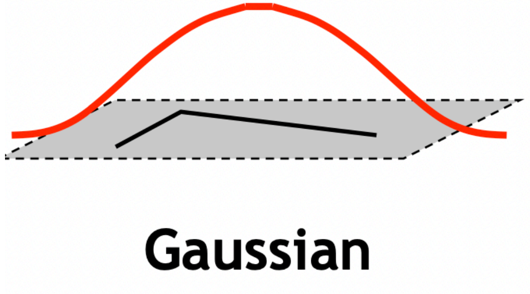
g_σ is rotation-invariant.

$$M(x_0, y_0) = \begin{bmatrix} w * I_x^2 & w * I_x I_y \\ w * I_x I_y & w * I_y^2 \end{bmatrix}$$

$$M(x_0, y_0) = \begin{bmatrix} g_\sigma * I_x^2 & g_\sigma * I_x I_y \\ g_\sigma * I_x I_y & g_\sigma * I_y^2 \end{bmatrix}$$

Using Gaussian Filter

$g_\sigma(x, y) =$



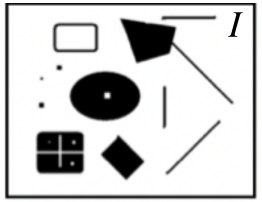
$$M(x_0, y_0) = \begin{bmatrix} g_\sigma * I_x^2 & g_\sigma * I_xI_y \\ g_\sigma * I_xI_y & g_\sigma * I_y^2 \end{bmatrix} = \begin{bmatrix} g(I_x^2) & g(I_xI_y) \\ g(I_xI_y) & g(I_y^2) \end{bmatrix}$$

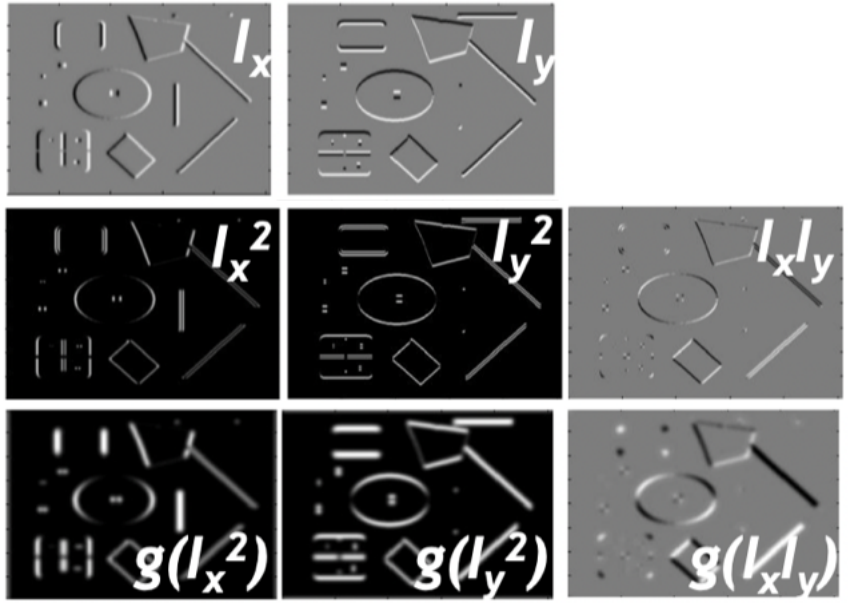
$$\therefore \theta(x_0, y_0) = \det(M(x_0, y_0)) - \alpha Tr(M(x_0, y_0))^2 - t$$

$$= (g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2) - \alpha[g(I_x^2) + g(I_y^2)]^2 - t$$

The Whole Process of Harris Detector

1. Image derivatives





2. Square of derivatives

3. Rectangle window or Gaussian filter

4. Corner response function

$$\theta = g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2 - t$$

5. Thresholding to obtain a binary mask $\theta(x_0, y_0) > 0$

6. Non-maximum suppression

Next week: Lecture 4,
Deep Learning I