Introduction to Computer Vision

Lecture 3 - Classic Vision II

Prof. He Wang

Edge Detection (Cont’d)

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

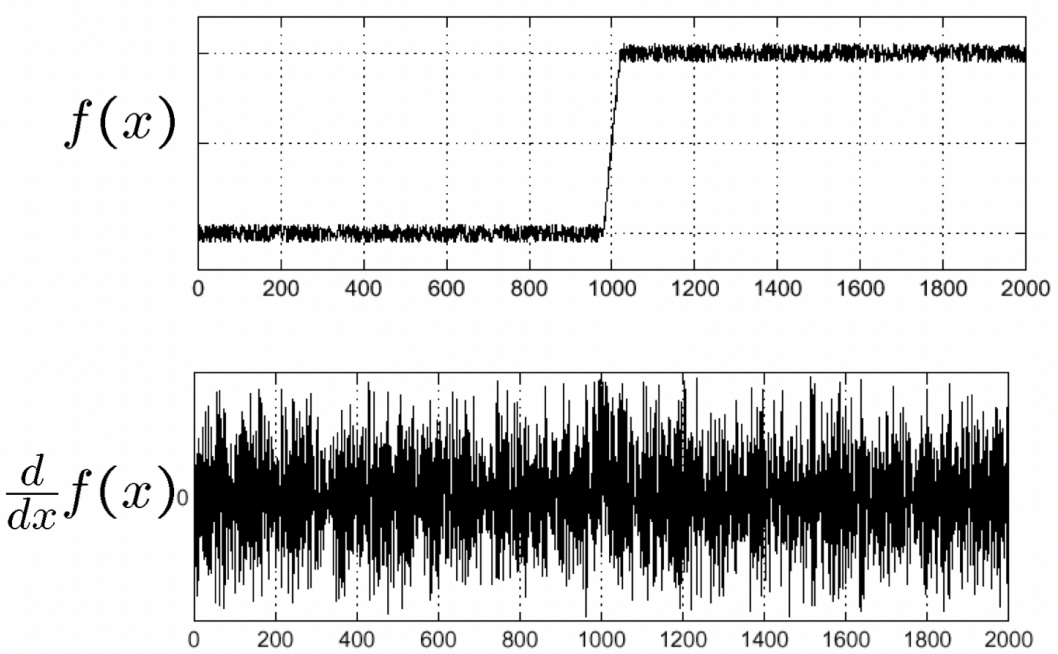
Start with Detecting Edges

- Edge detector

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

Effects of Noises

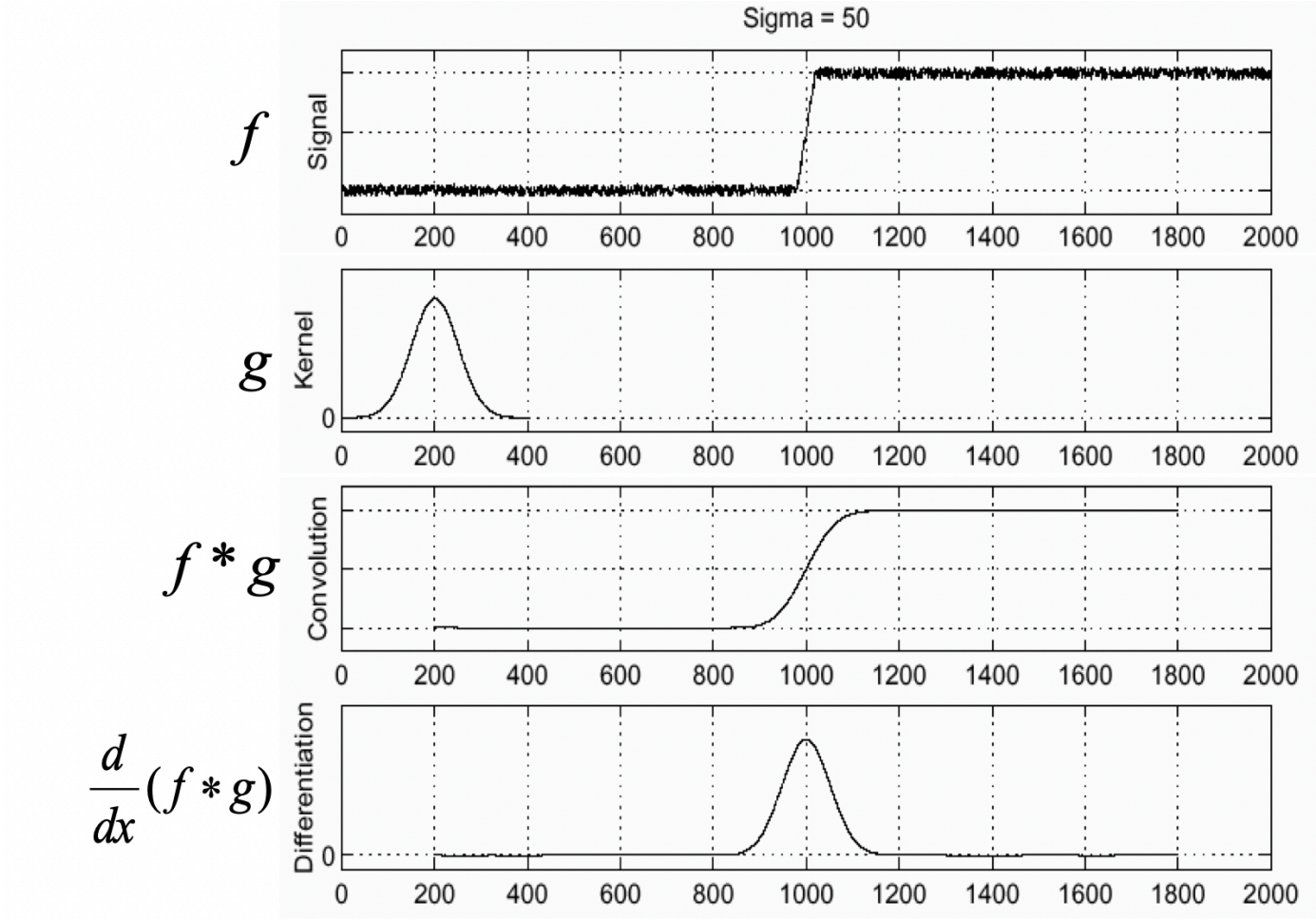
- Consider one row in the image



- Image gradients are too sensitive to noise.
- Gradients of the true edge is overwhelmed by noises.
- We need smoothing!

Source: Steven Seitz

Smoothing by a Low-Pass Filter



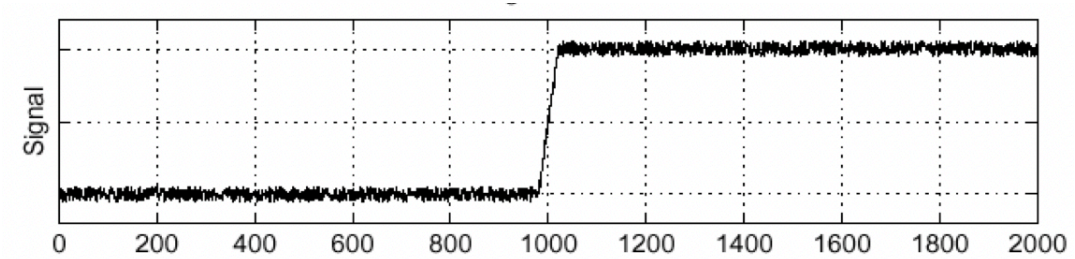
Source: Steven Seitz

Derivative Theorem of Convolution

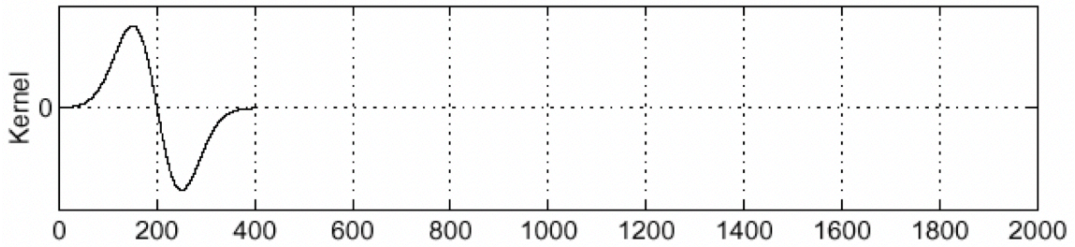
- Theorem:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

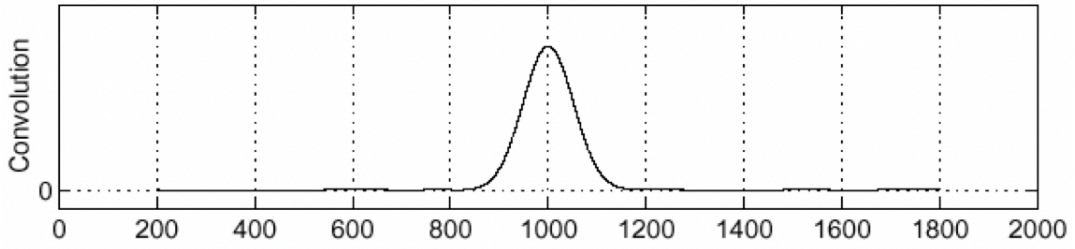
f



$\frac{d}{dx} g$

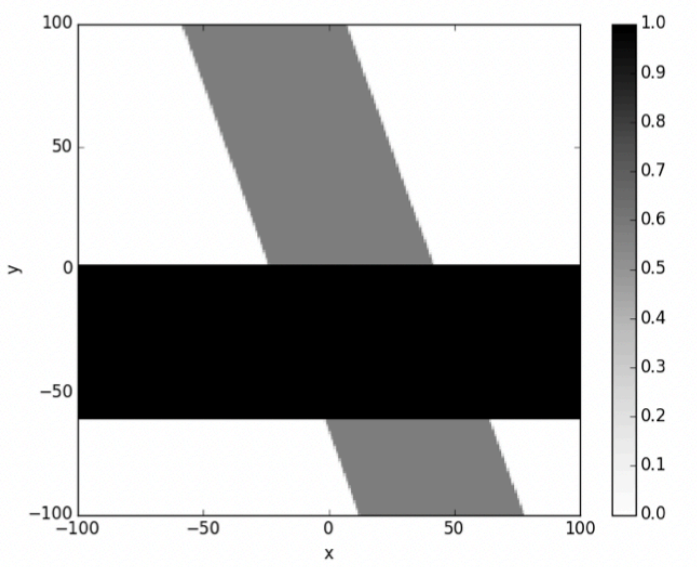
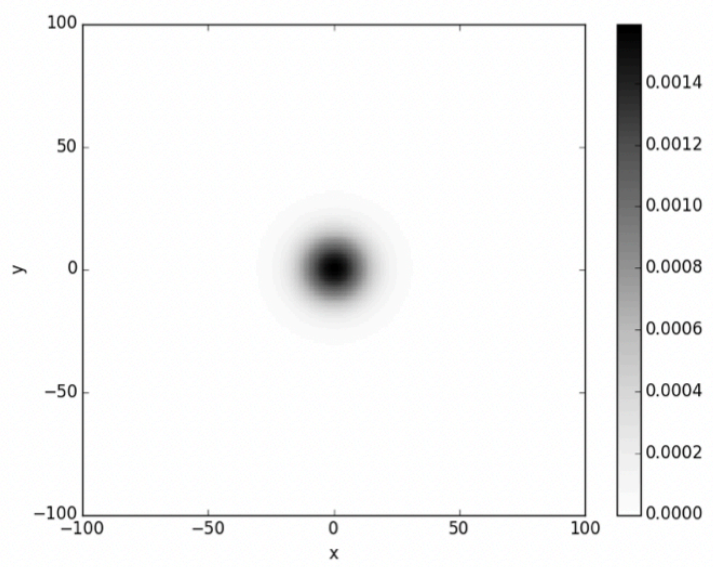


$f * \frac{d}{dx} g$

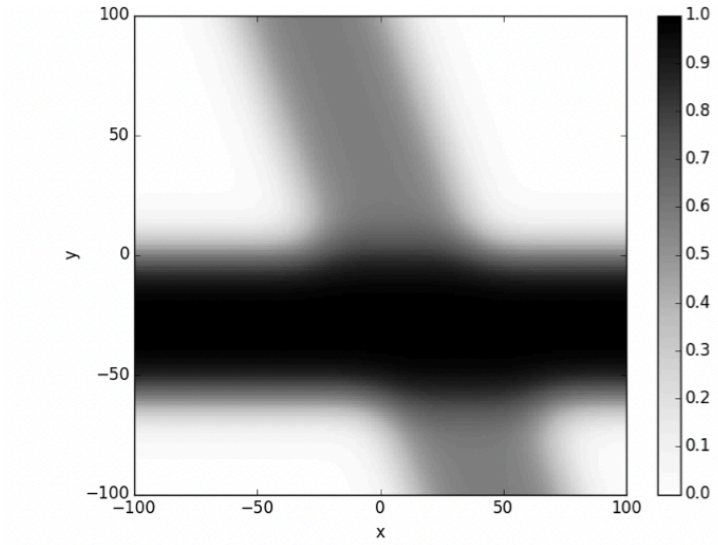


- Saves us one operation.

Two-Dimensional Convolution

f  $*$ 

$$g = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

$=$ 

$$(f * g)[m, n] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] g[m - k, n - l]$$

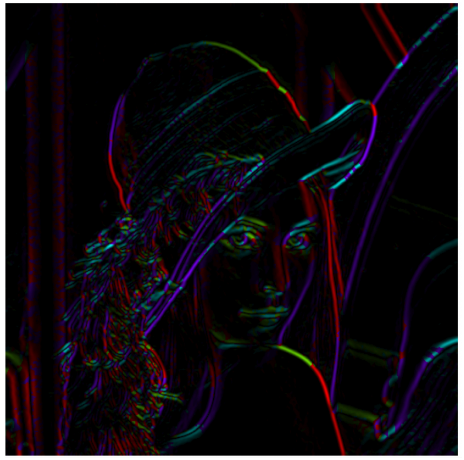
Source: Feifei Li

Compute Gradient

x-derivative of Gaussian

y-derivative of Gaussian

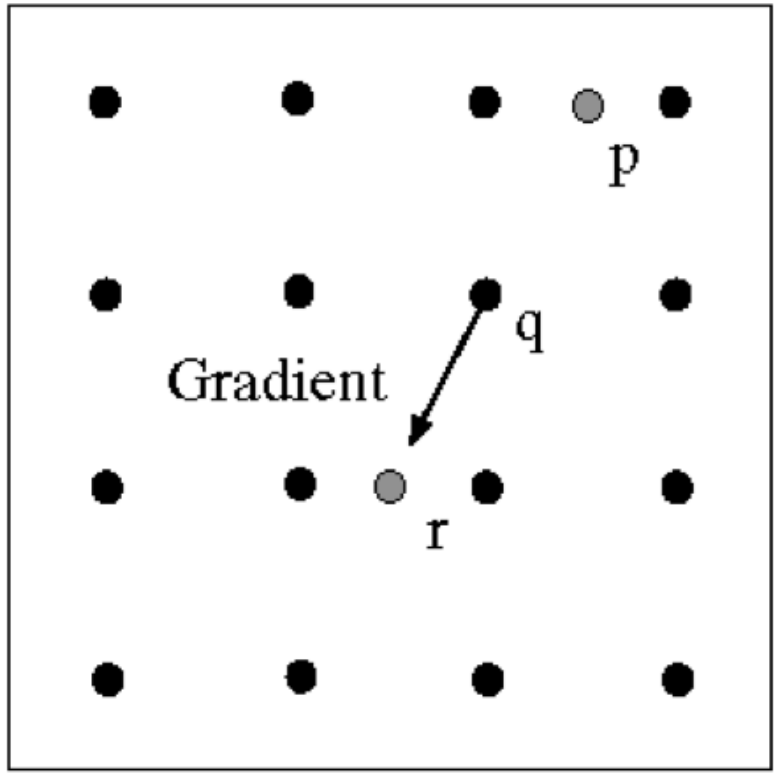
Gradient magnitude



Thresholding and Gradient orientation

Source: Feifei Li

Non-Maximal Suppression (NMS)

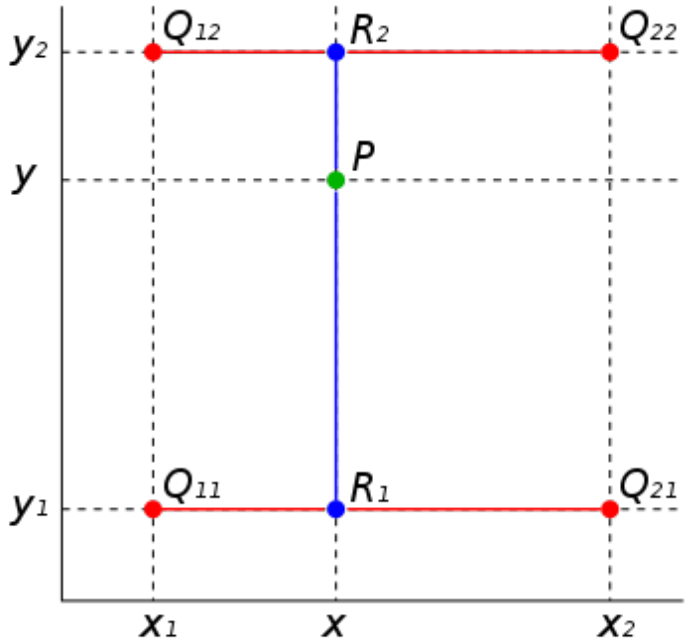


- For each point q on grids, compute the gradient $g(q)$.
- Move along the gradient to get two neighbors: $r = q + g(q), p = q - g(q)$
- Perform bilinear interpolation to get $g(p)$ and $g(r)$.
- If the magnitude of g(q) is larger than $g(p)$ and $g(r)$, q is a maximum that should be kept.

Source: J. Hayes

Bilinear Interpolation

For $P(x, y)$, given its four surrounding grid points $f(Q_{11}), f(Q_{12}), f(Q_{21})$ and $f(Q_{22})$,
how to obtain $f(P)$ via *bilinear interpolation*?



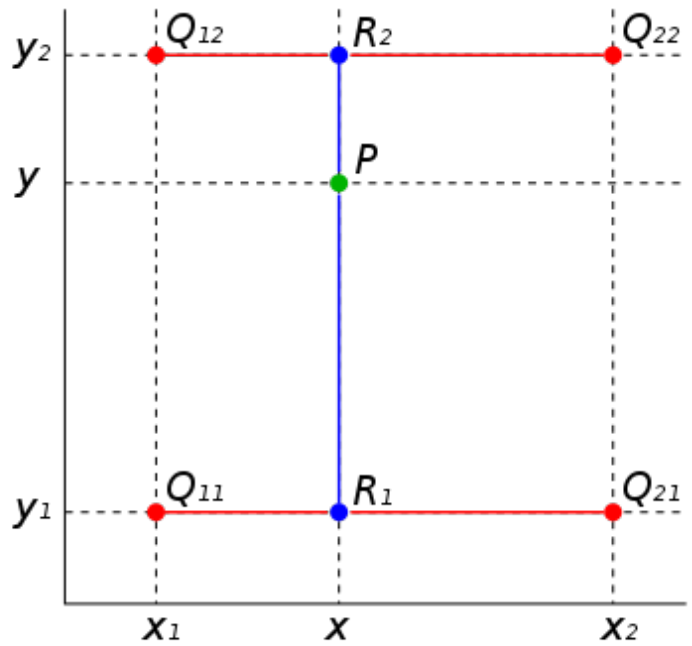
First, linear interpolate to obtain $f(R_1)$ and $f(R_2)$

$$R_1: \quad f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$

$$R_2: \quad f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Bilinear Interpolation

For $P(x, y)$, given its four surrounding grid points $f(Q_{11}), f(Q_{12}), f(Q_{21})$ and $f(Q_{22})$, how to obtain $f(P)$ via *bilinear interpolation*?



First, linear interpolate to obtain $f(R_1)$ and $f(R_2)$

$$R_1: \quad f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$

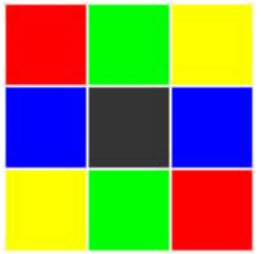
$$R_2: \quad f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

Then, linear interpolate between $f(R_1)$ and $f(R_2)$ to obtain $f(P)$:

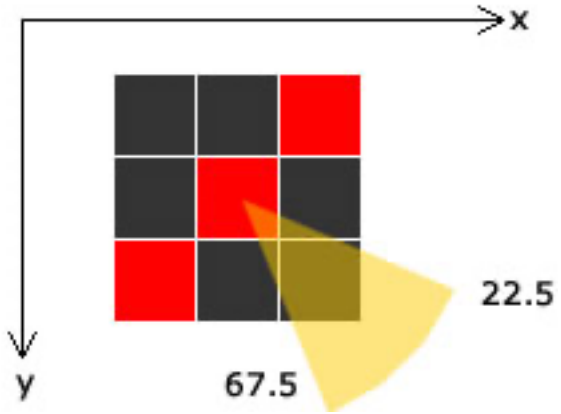
$$\begin{aligned} P: \quad f(x, y) &= \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \right) \end{aligned}$$

A Simplified Version of NMS

The orientation of each pixel is put into one of the four bins.



Example: gradient orientation from 22.5 to 67.5 degrees



To check if the central red pixel belongs to an edge, you need to check if the gradient is maximum at this point. You do this by comparing its magnitude with the top left pixel and the bottom right pixel.

Before and After NMS

Thin multi-pixel wide “ridges” down to single pixel width

Source: J. Hayes

Hysteresis Thresholding

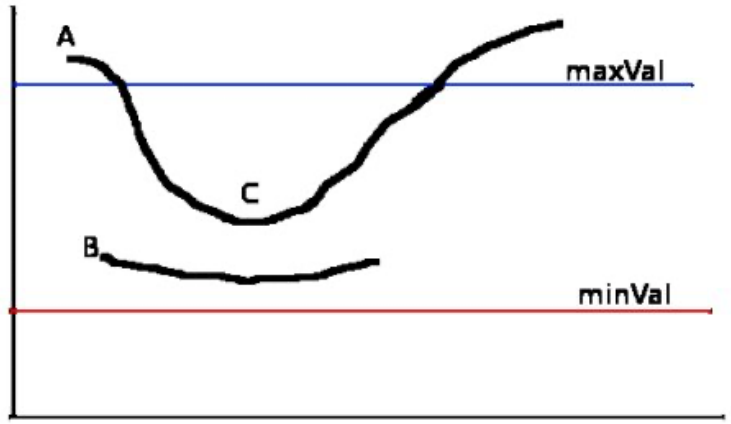
- Use a high threshold (maxVal) to start edge curves and a low threshold (minVal) to continue them.
  - Pixels with gradient magnitudes $>$ maxVal should be reserved
  - Pixels with gradient magnitudes $<$ minVal should be removed.
  - How to decide maxVal and minVal? Examples:
    - maxVal = 0.3 $\times$ average magnitude of the pixels that pass NMS
    - minVal = 0.1 $\times$ average magnitude of the pixels that pass NMS

Source: J. Hayes

Edge Linking

- Drop-outs?

- Now using the direction information and the lower threshold, we'll "grow" these edges.
  - If the current pixel is not an edge, check the next one.
  - If it is an edge, check the two pixels in the direction of the edge (ie, perpendicular to the gradient direction). If either of them (or both)



Edge Linking

- Drop-outs?
- Now using the direction information and the lower threshold, we'll "grow" these edges.
  - If the current pixel is not an edge, check the next one.
  - If it is an edge, check the two pixels in the direction of the edge (ie, perpendicular to the gradient direction). If either of them (or both)
    - have the direction in the same bin as the central pixel
    - gradient magnitude is greater than minVal
    - they are the maximum compared to their neighbors (NMS for these pixels), then you can mark these pixels as an edge pixel

Edge Linking

- Drop-outs?
- Now using the direction information and the lower threshold, we'll "grow" these edges.
  - If the current pixel is not an edge, check the next one.
  - If it is an edge, check the two pixels in the direction of the edge (ie, perpendicular to the gradient direction). If either of them (or both)
    - have the direction in the same bin as the central pixel
    - gradient magnitude is greater than minVal
    - they are the maximum compared to their neighbors (NMS for these pixels), then you can mark these pixels as an edge pixel
  - Loop until there are no changes in the image Once the image stops changing, you've got your canny edges! That's it! You're done!

Canny Edge Detector

- The most widely used edge detector in computer vision
- Canny shows that the first derivative of the Gaussian closely approximates the operator that optimizes the product of signal-to-noise ratio and localization.

JJ. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Tradeoff between Smoothing and Localization

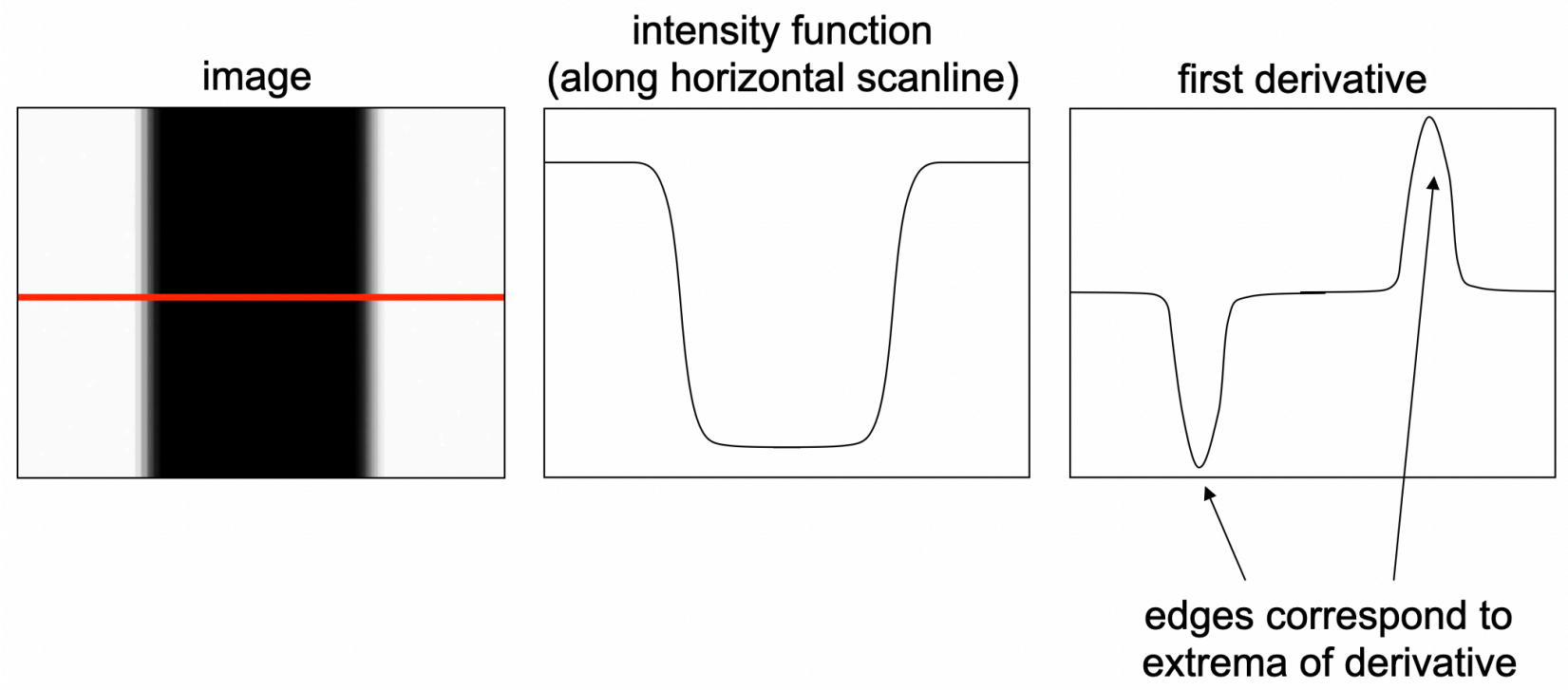
original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

- Note a larger σ corresponds to stronger smoothing.
- Smoothed derivative reduces noises but blurs edges.
- Find edges at different scales.

Image credit: J. Hayes

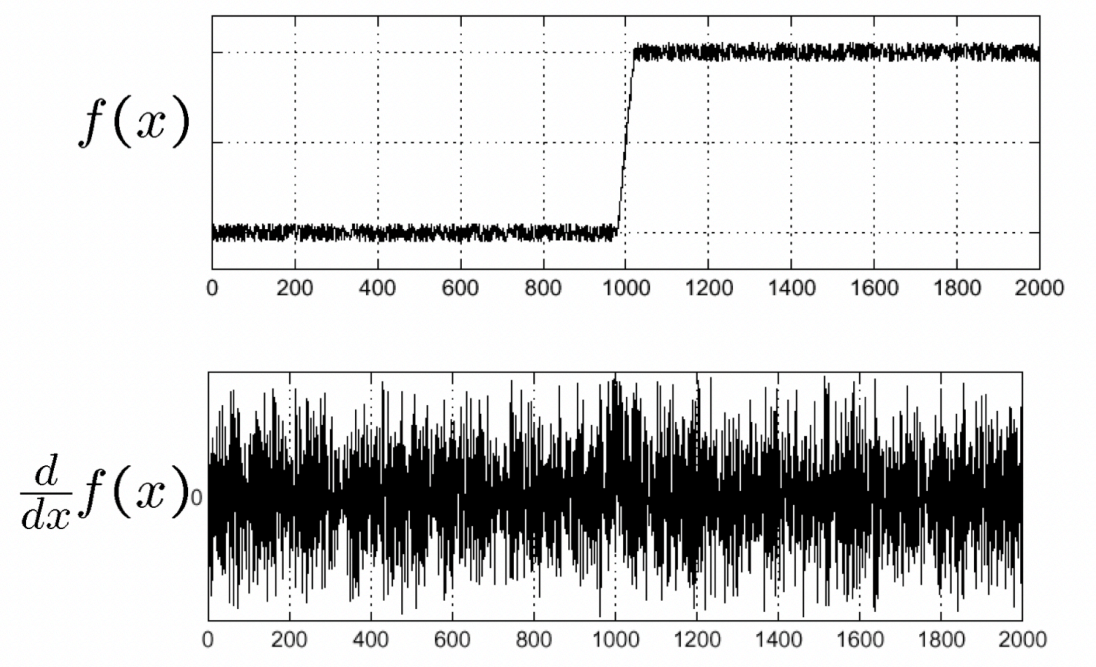
Summary of Edge Detection

- What is an edge?



Summary of Edge Detection

- Edge: where pixel intensity changes drastically
- Compute image gradient to find edge, however noises can be overwhelming and fail the detection



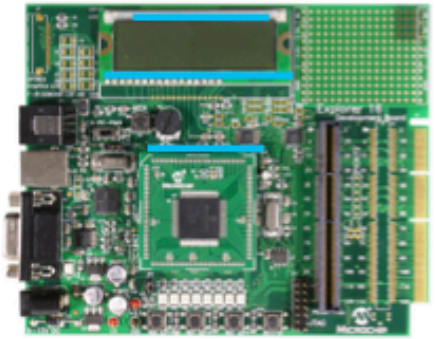
Summary of Canny Edge Detection

- 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):

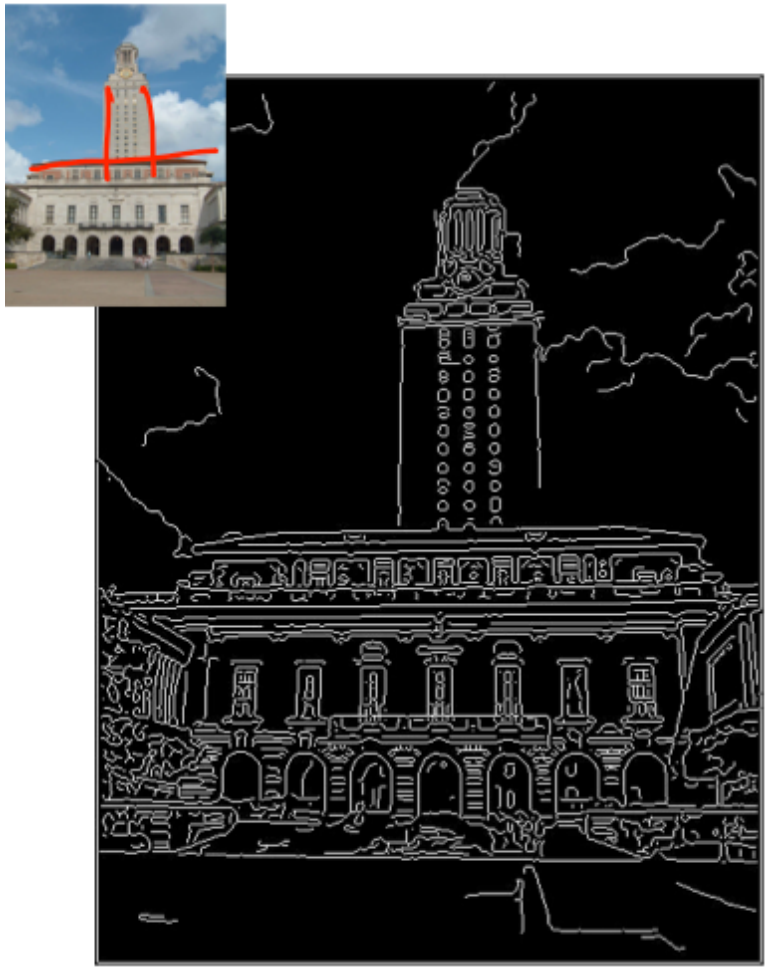
Line Fitting

Line Detection

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

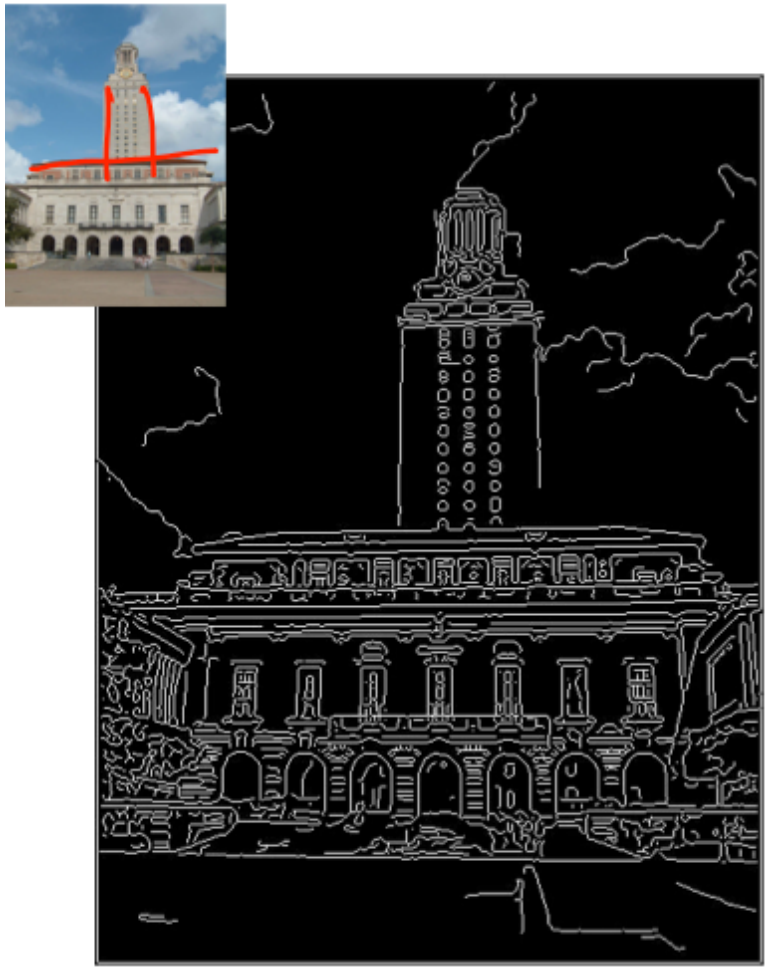


Challenge of Line Detection



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

Challenge of Line Detection

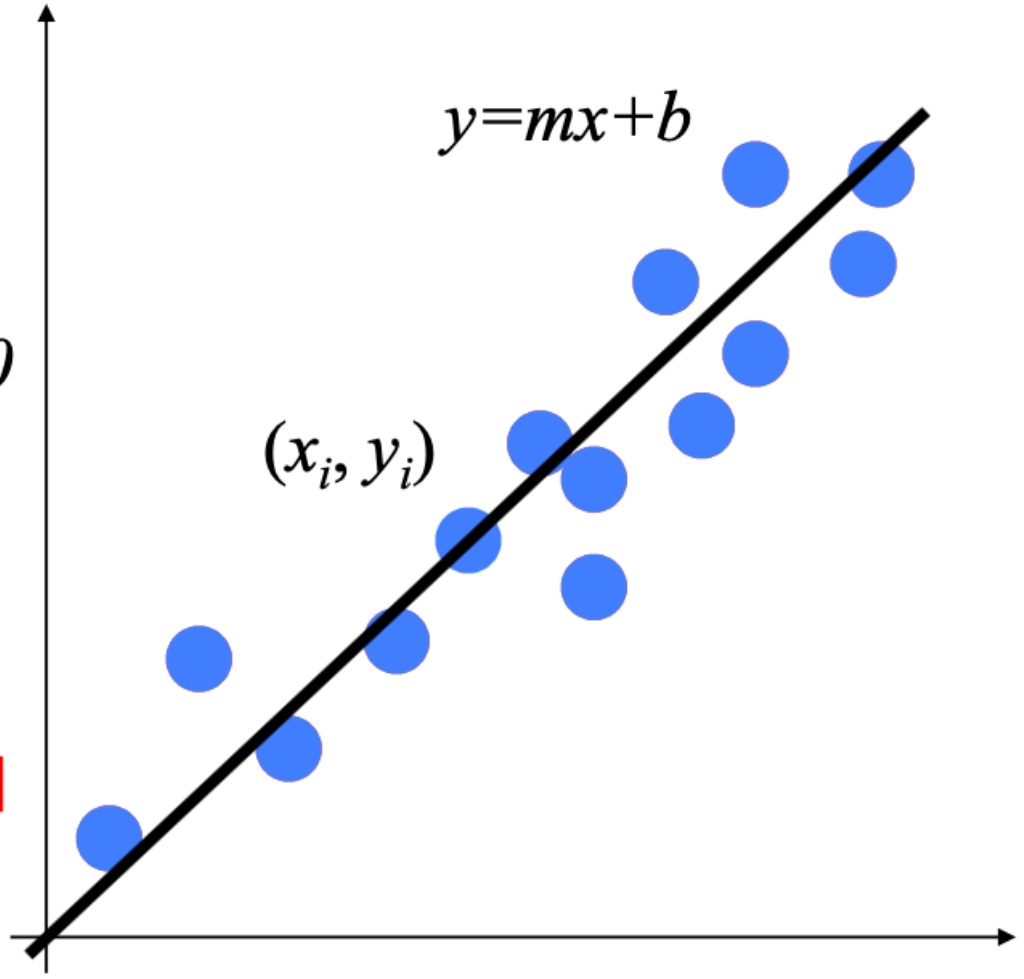


- Aren’t we done just by doing edge detection?
- No, there are many problems:
  - Occlusion
  - Not a straight line
  - Multiple lines, which one?

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

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



Line Fitting: Least Square Method

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

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

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

Find B=$[m, b]^{\mathrm{T}}$ that minimizes E $\quad \frac{dE}{dB} = -2X^T Y + 2X^T XB = 0$ [Eq. 5]

$$\mathrm{X}^{\mathrm{T}}\mathrm{XB} = \mathrm{X}^{\mathrm{T}}\mathrm{Y}$$ [Eq. 7]

Normal equation

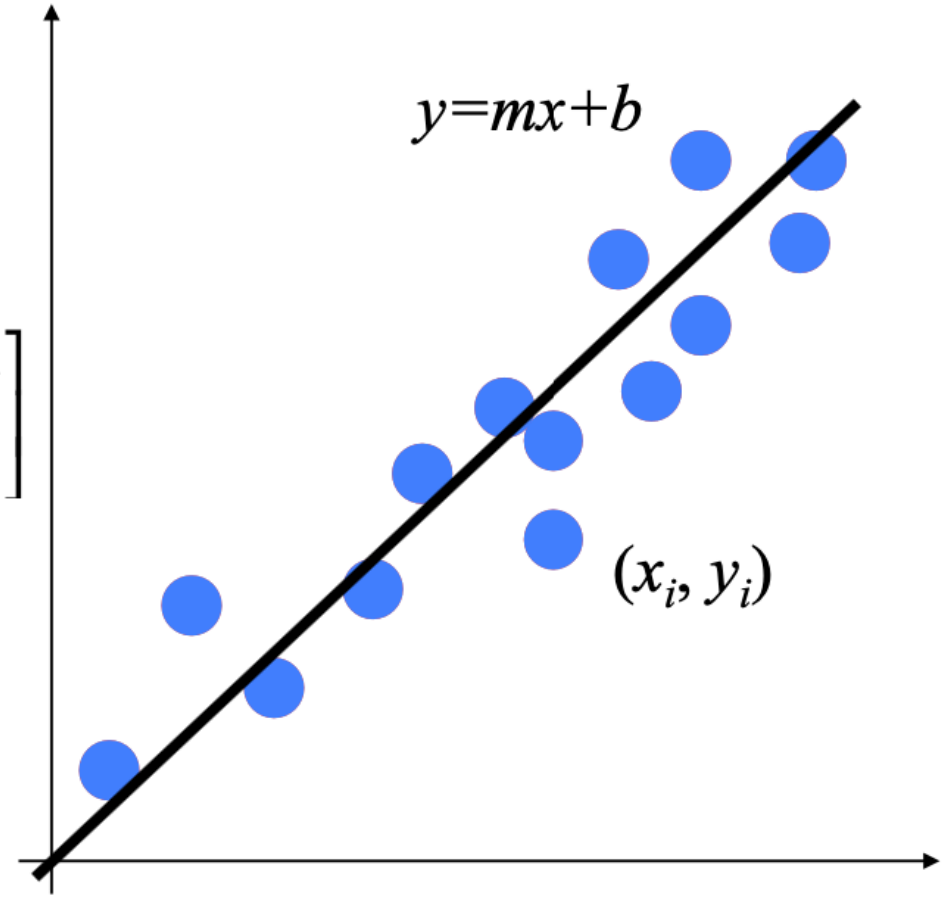
$$\mathrm{B} = \left(\mathrm{X}^{\mathrm{T}}\mathrm{X}\right)^{-1}\mathrm{X}^{\mathrm{T}}\mathrm{Y}$$ [Eq. 6]

Line Fitting: Least Square Method

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

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

[Eq. 6]



Limitations

- Fails completely for vertical lines

Line Fitting: Least Square Method

- Distance between point (x_n, y_n) and line $ax+by=d$

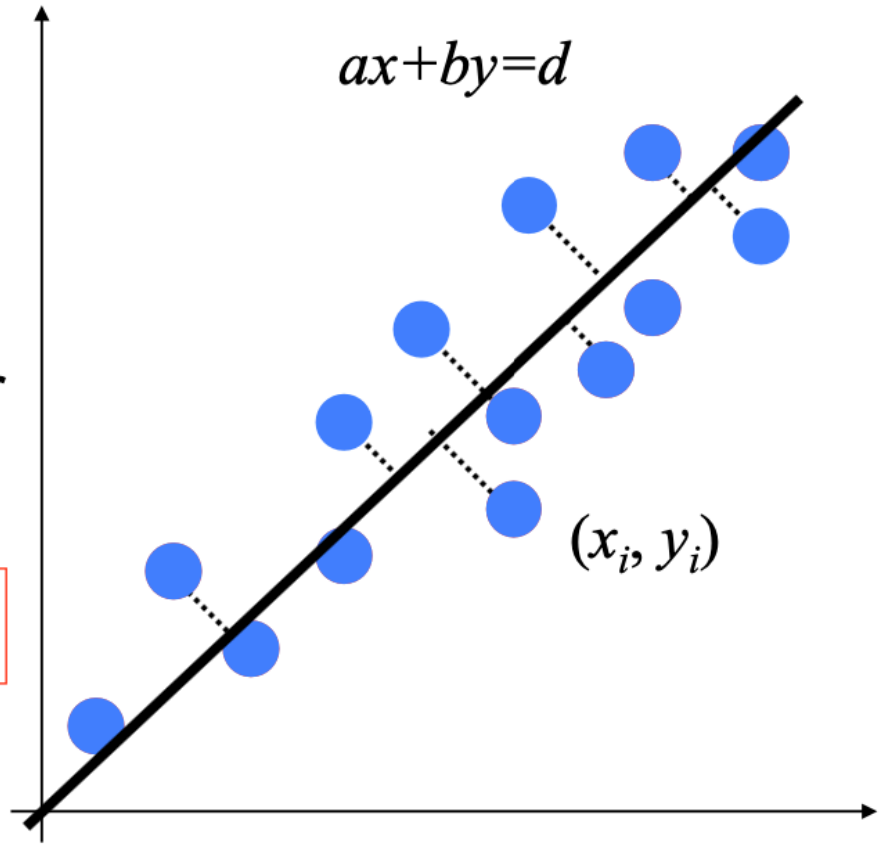
- Find (a, b, d) to minimize the sum of squared perpendicular distances

[Eq. 8]

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

$$Ah = 0 \quad \text{[Eq. 9]}$$

data model parameters



Line Fitting: Least Square Method

Solve $h = (a, b, d)$ subject to $a^2 + b^2 = 1$:

1) Apply translation $x_i' = x_i - \frac{1}{n}\sum_j x_j, y_i' = y_i - \frac{1}{n}\sum_j y_j$

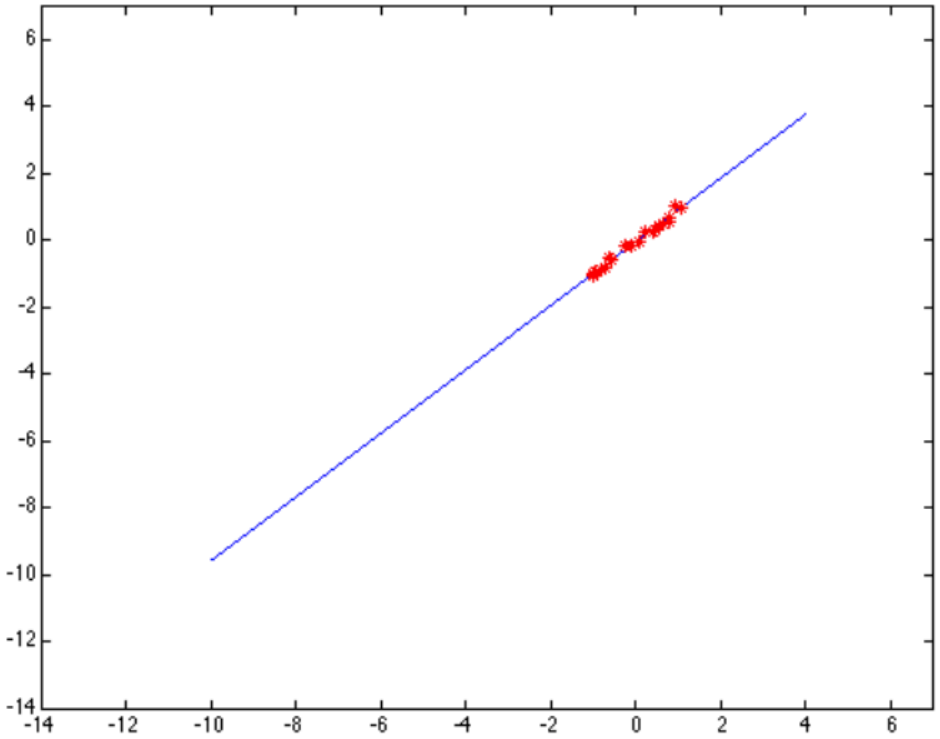
2) $E = \sum_j (ax_j' + by_j' - d')^2$ should also be minimized

3) $0 = \frac{\partial E}{\partial d'} = \frac{\partial}{\partial d'}\sum_j (ax_j' + by_j' - d')^2 = 2d' \rightarrow E = \sum_j (ax_j' + by_j')^2$

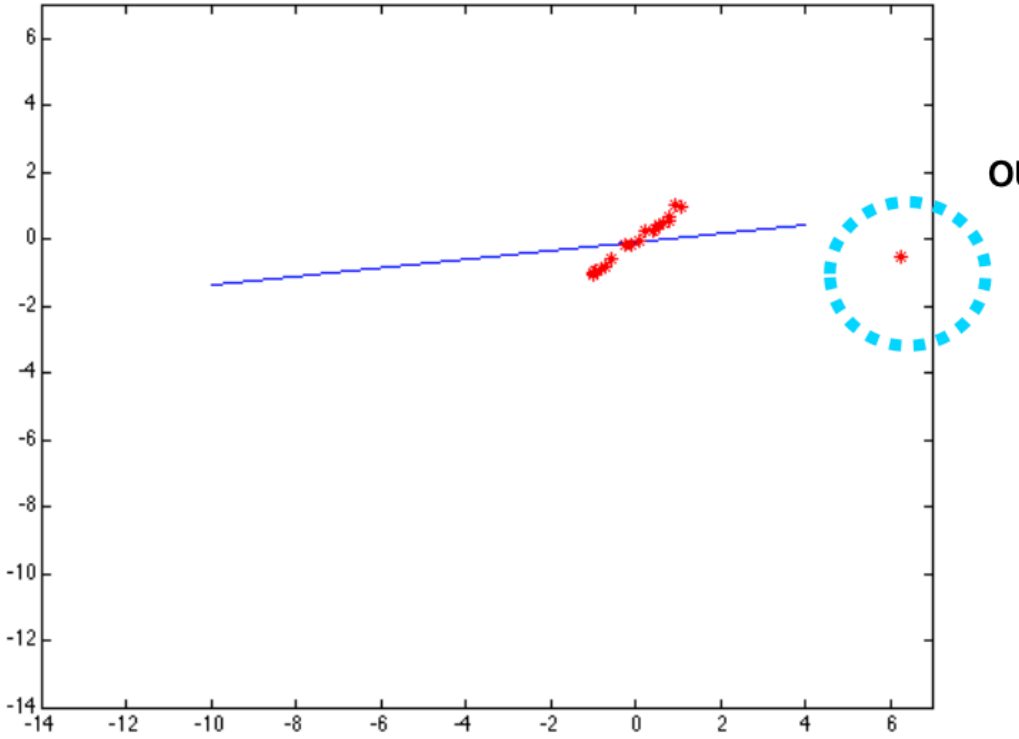
4) $A'h' = 0$, A' is rank deficient, Minimize $\|A'h'\|$ subject to $\|h'\| = 1$

5) $A' = U'D'V'^T$, h'=last column of V'

Robustness



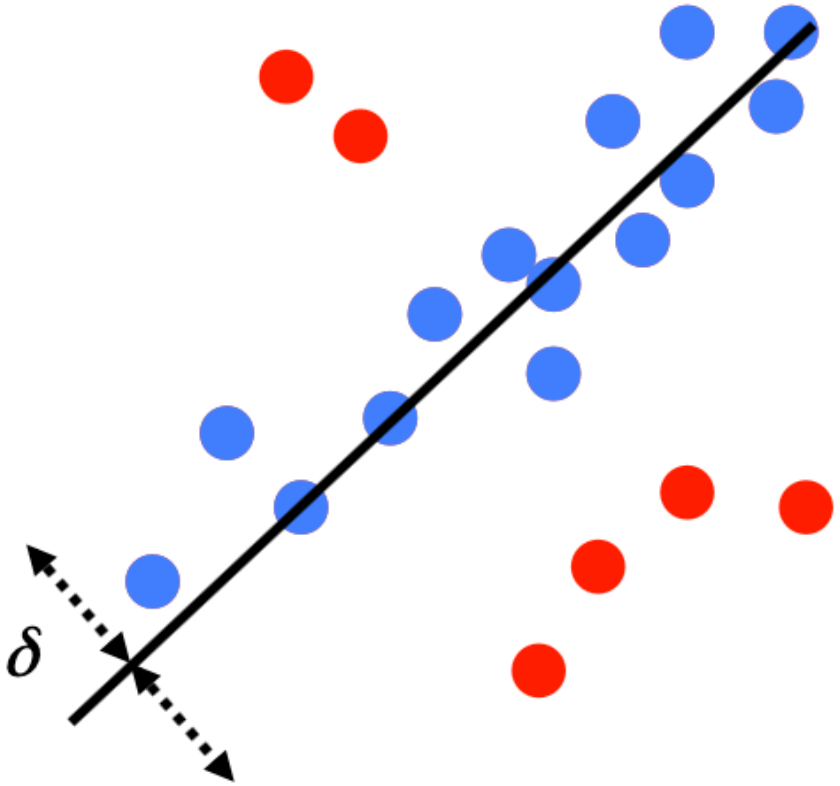
Robust to small noises.



Sensitive to outliers.

RANSAC: RANdom SAmple Consensus

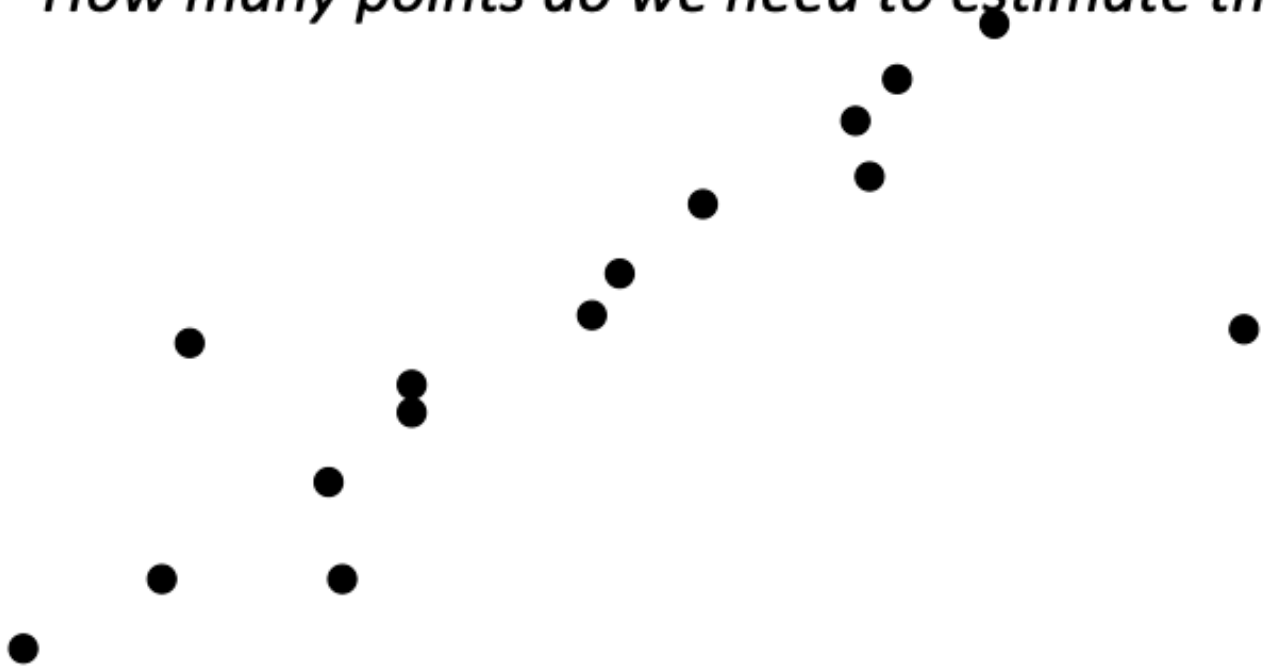
- Idea: we need to find a line that has the largest supporters (or inliers)



Fischler & Bolles in '81.

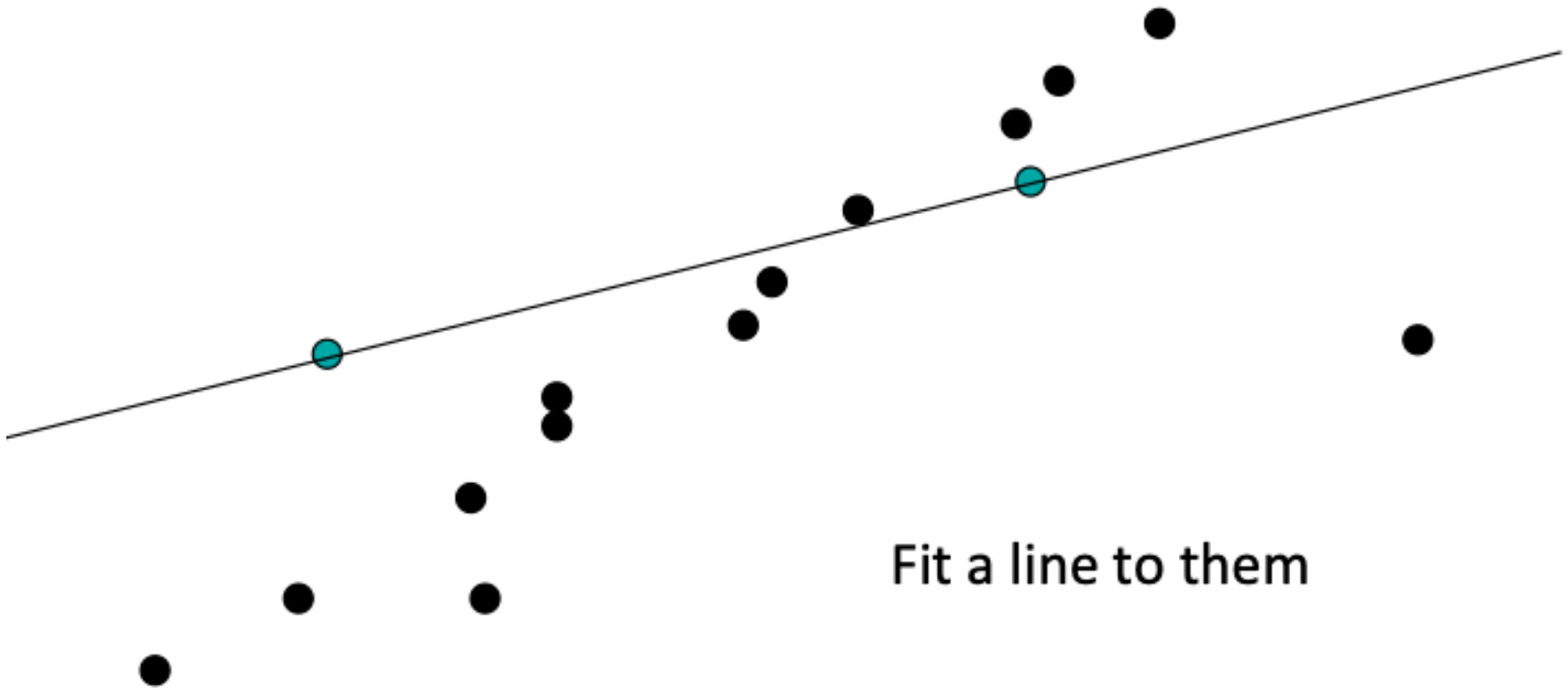
RANSAC Line Fitting

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



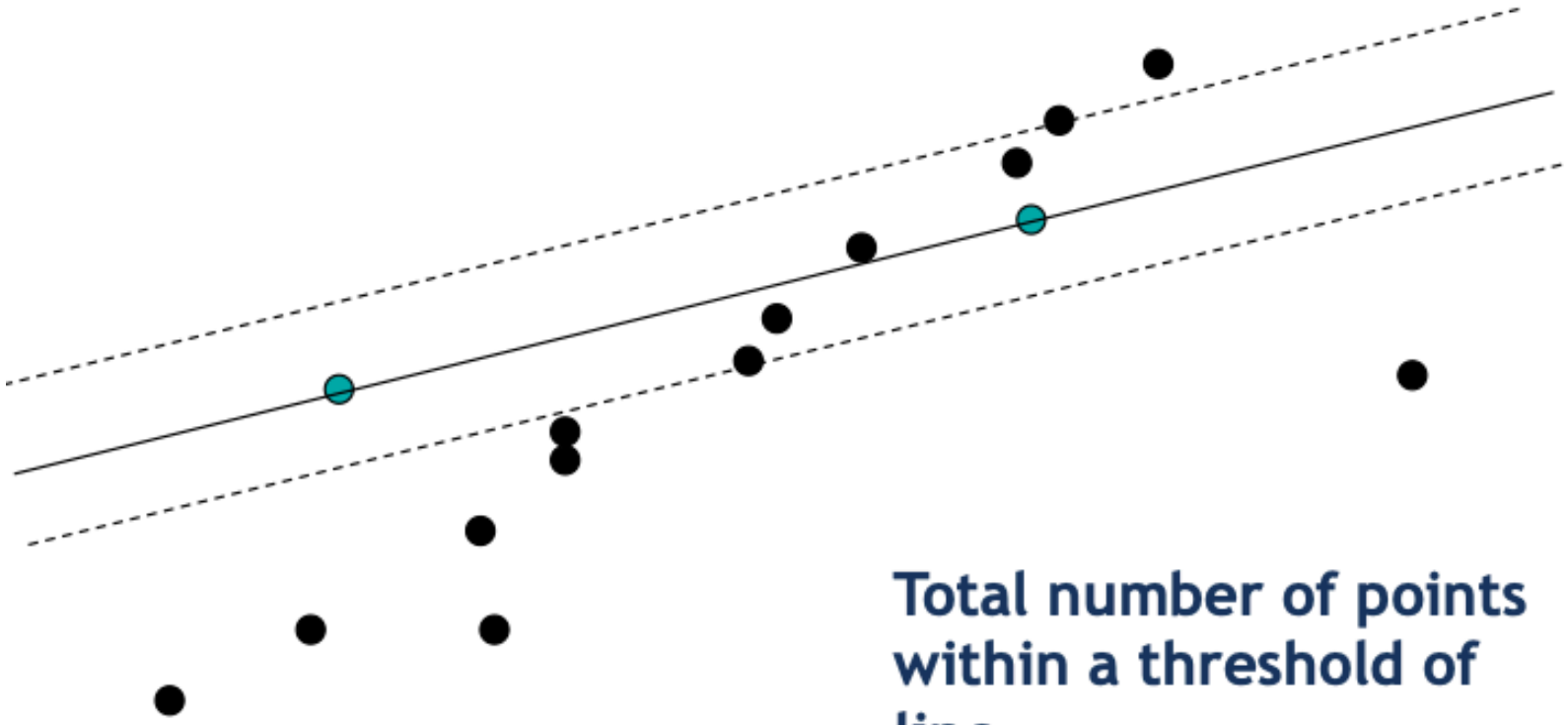
RANSAC Line Fitting

- Task: Estimate the best line



RANSAC Line Fitting

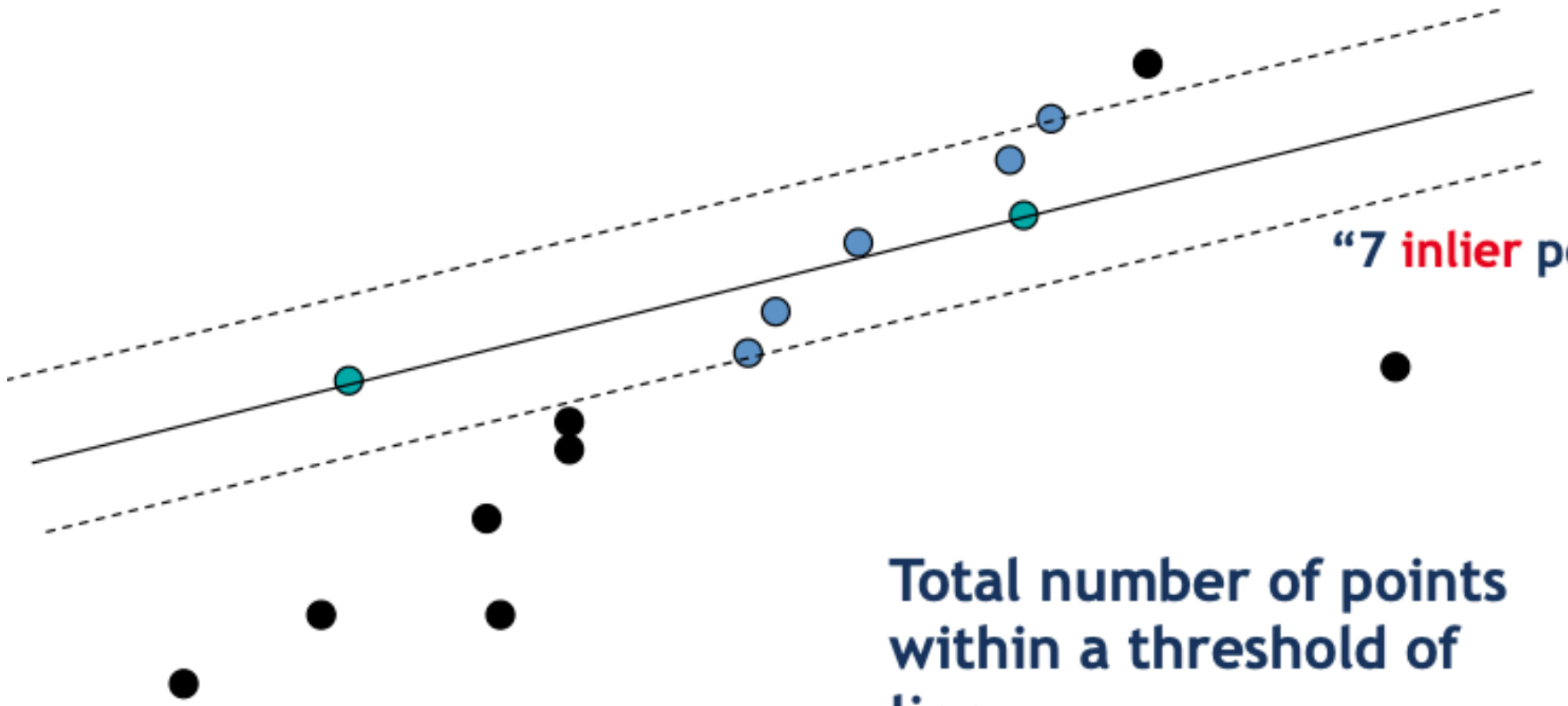
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC Line Fitting

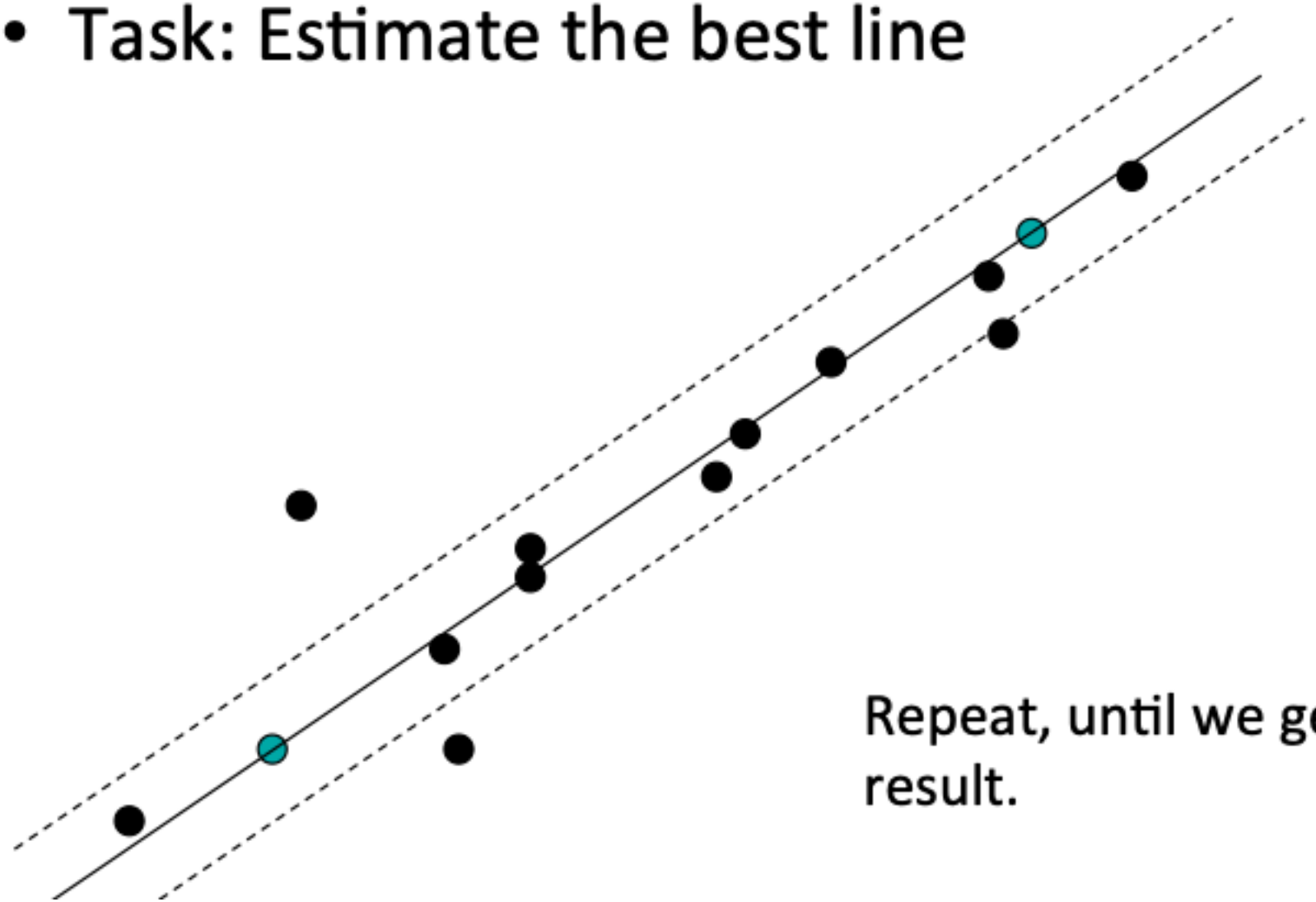
- Task: Estimate the best line



Slide credit: Jinxiang Chai

RANSAC Line Fitting

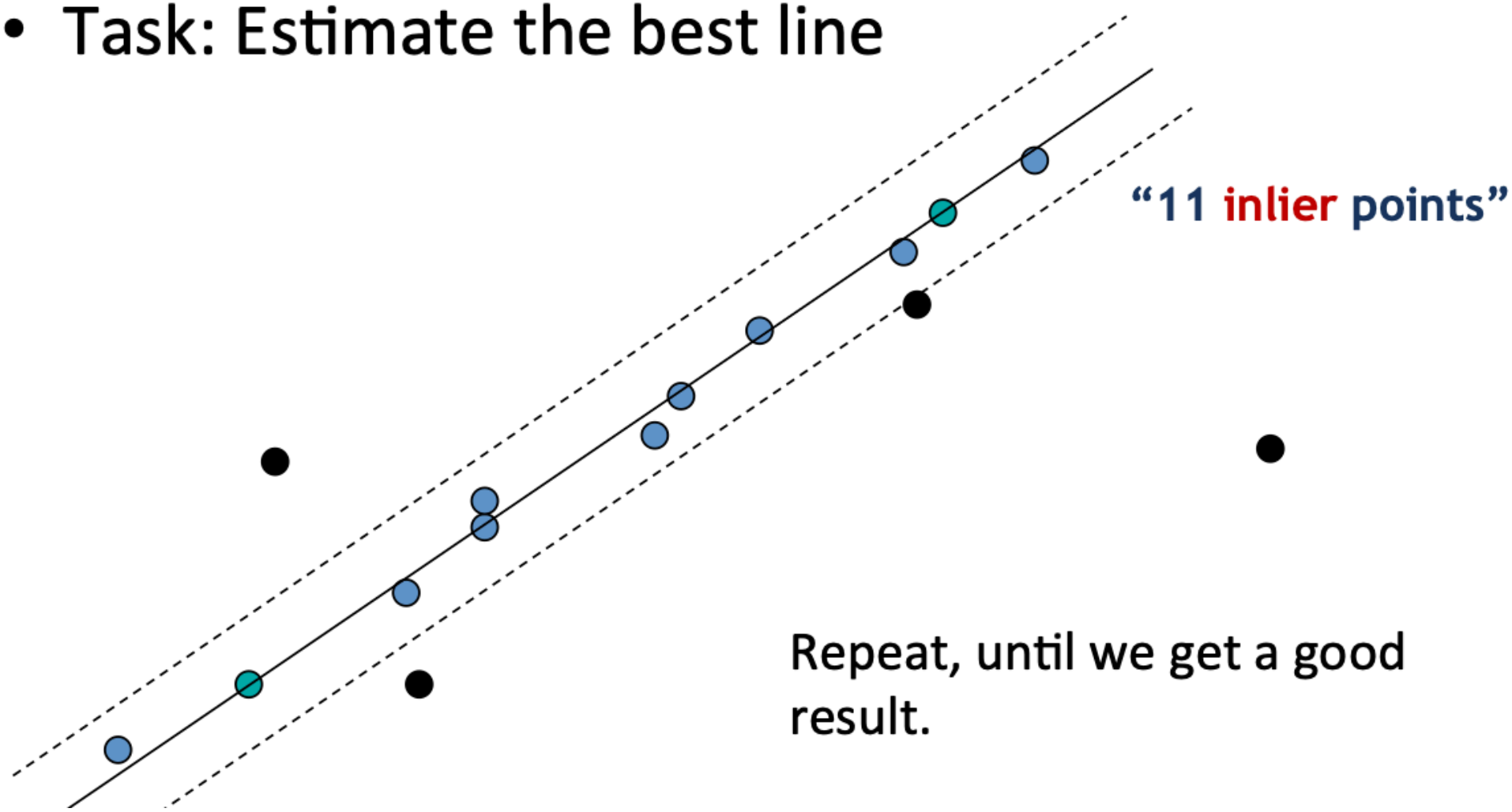
- Task: Estimate the best line



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

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: How Many Samples?

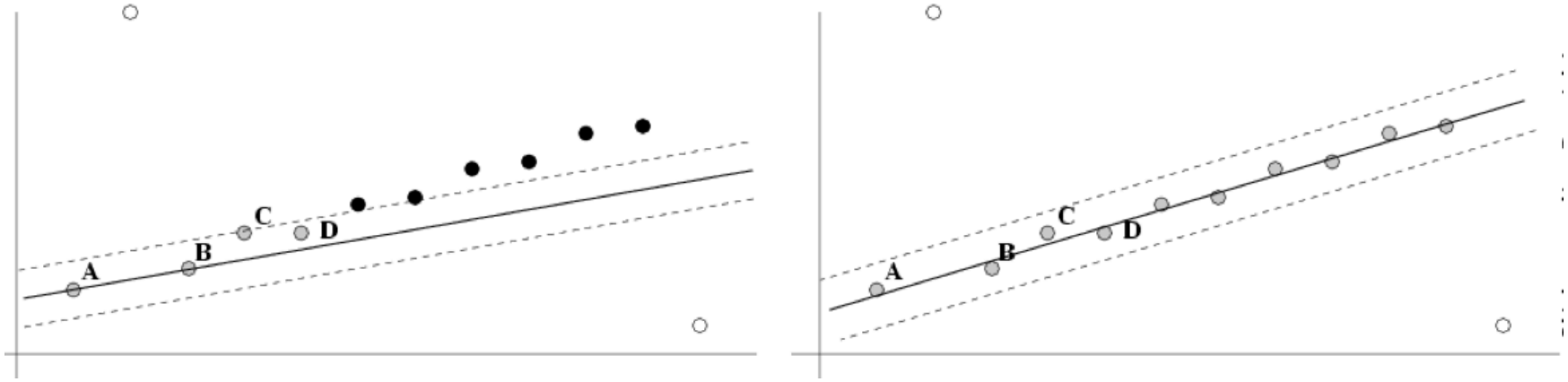
- How many samples are needed?
  - Suppose w is fraction of inliers (points from line).
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen.
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$

$\Rightarrow$ Choose k high enough to keep this below desired failure rate.

Slide credit: David Lowe

After RANSAC

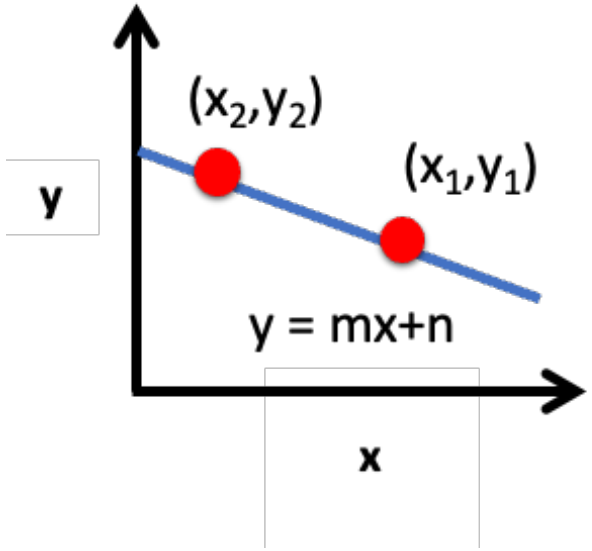
- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



RANSAC: Pro and Con

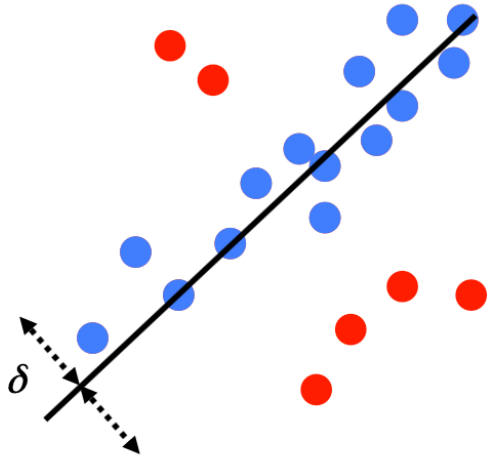
- **Pros:**
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons:**
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective 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

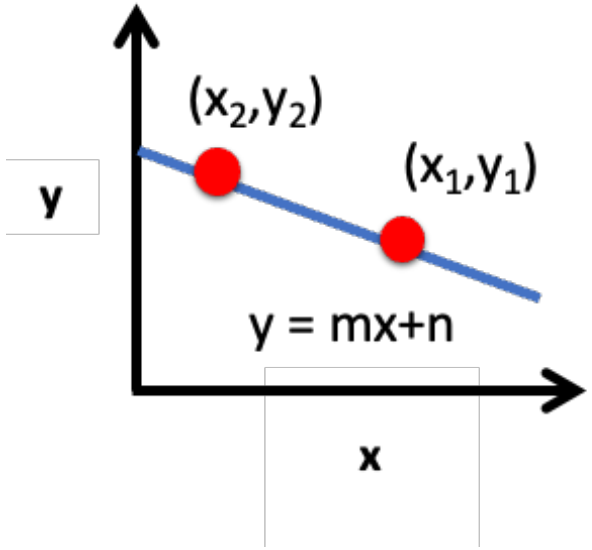
RANSAC



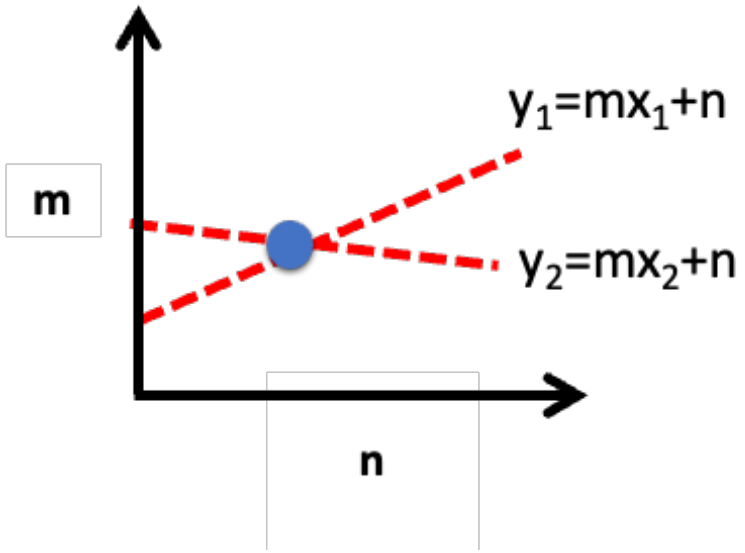
Define a inlier threshold distance in the vector space, each point votes for the best hypothesis.

Hough Transform

Original space



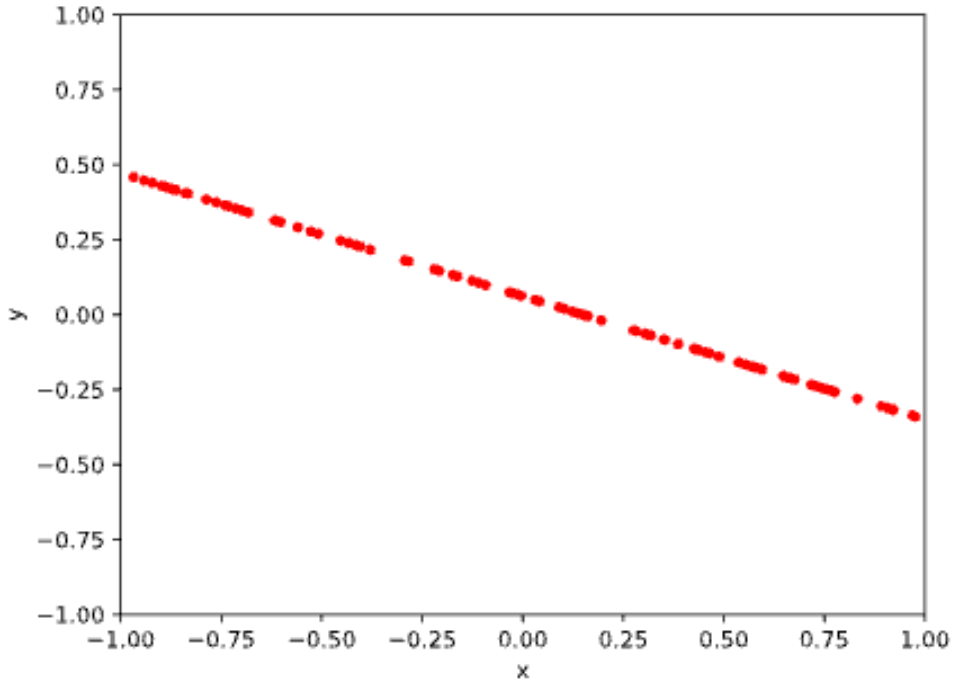
Hough space



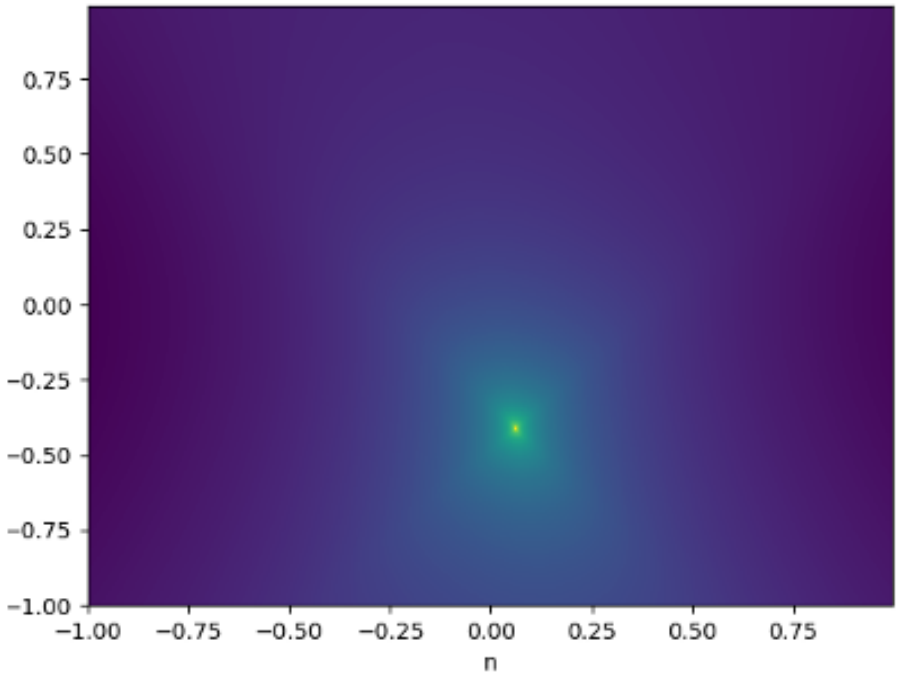
Given points in the vector space, find (m,n) in the parameter space

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

Hough Transform w/o Noise



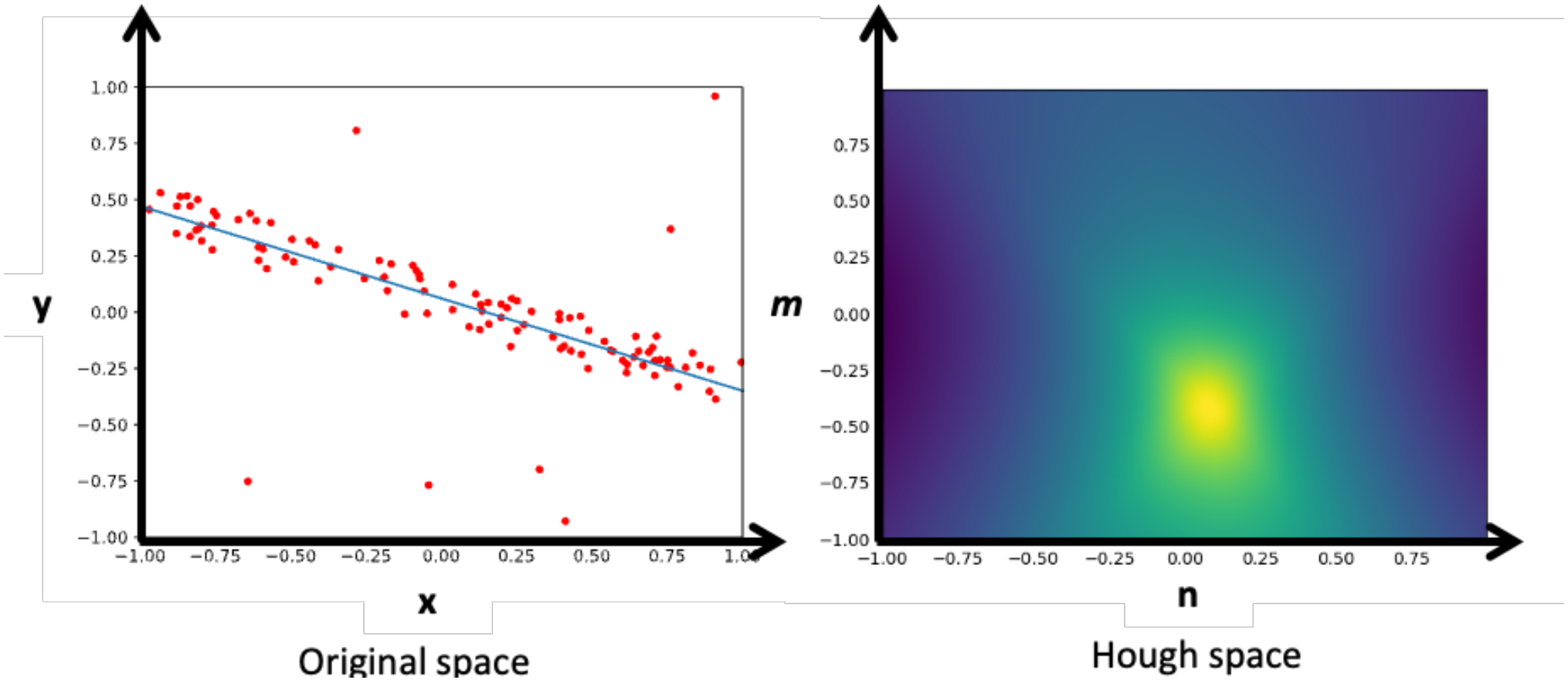
Original space



Hough space

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

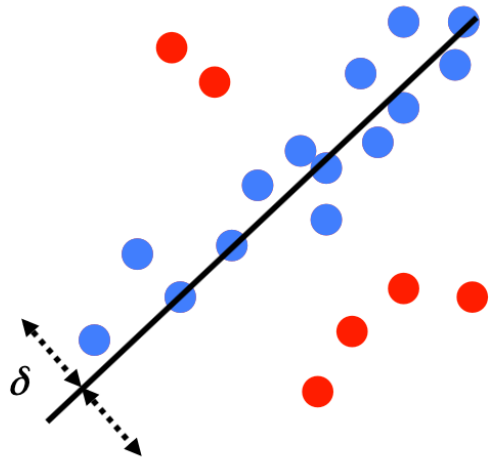
Hough Transform w/ Noise and Outliers



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

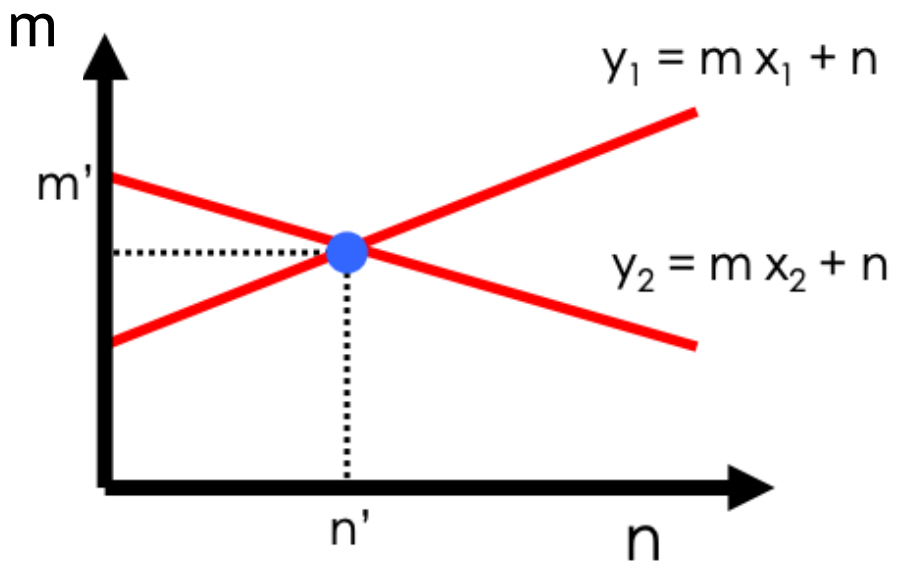
From the Perspective of Voting

RANSAC



Voting in the original space

Hough transform



Voting in the parameter space

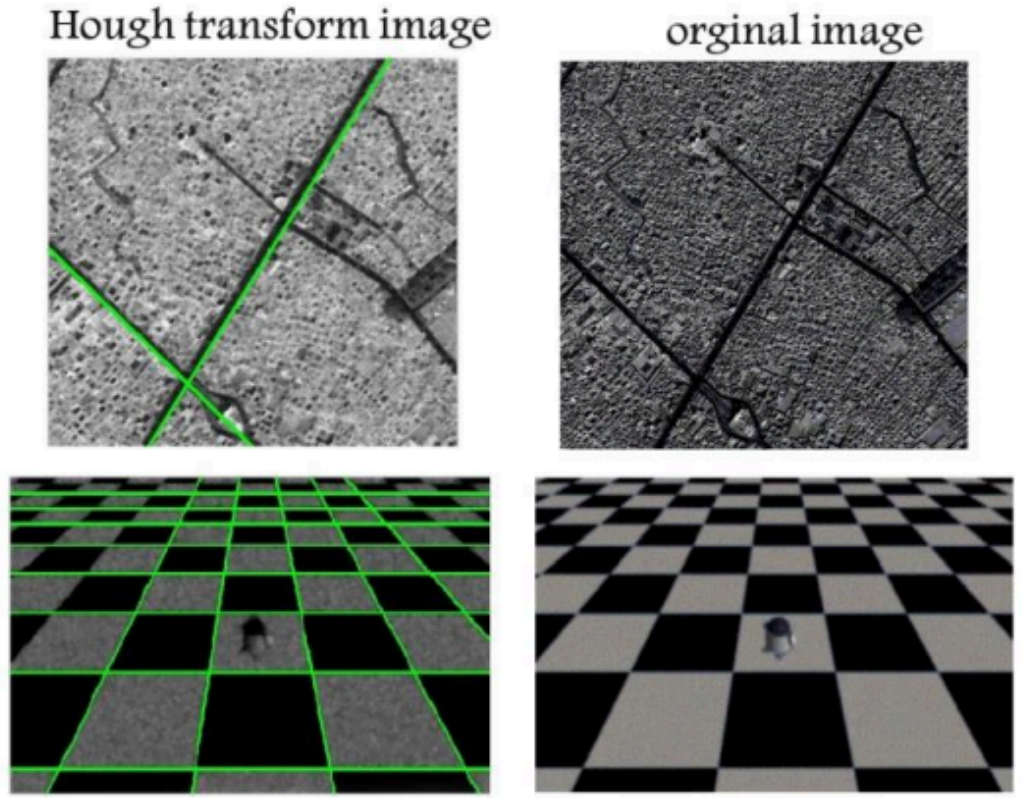
Robust Fitting: RANSAC vs. Hough Transform

RANSAC

- Single mode: robust for outliers

Hough Transform

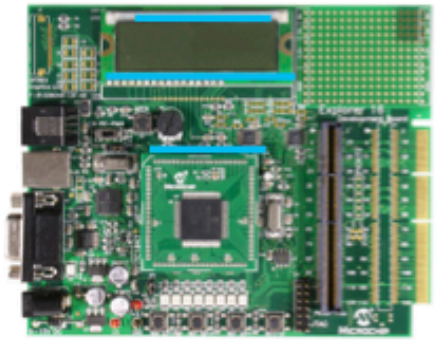
- Less robust compared to RANSAC (spurious peak)
- Can handle multiple modes well



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

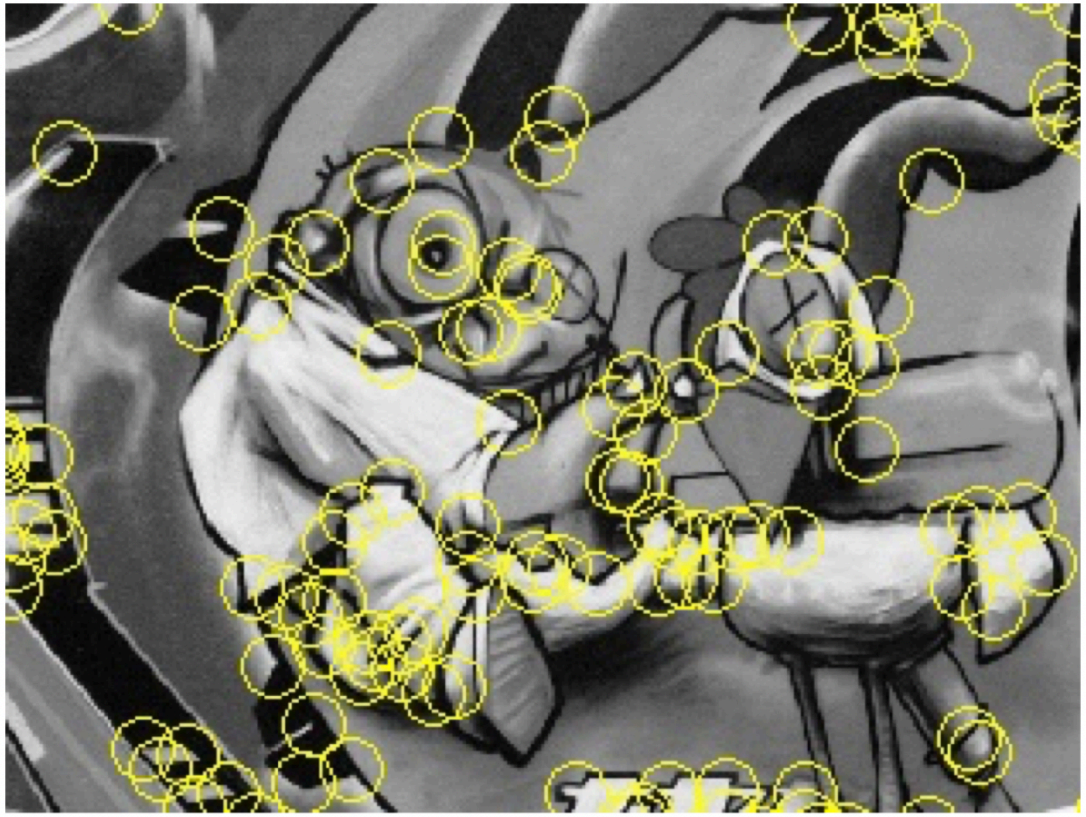
- Gradient -> Edge -> Line



Corner Detection

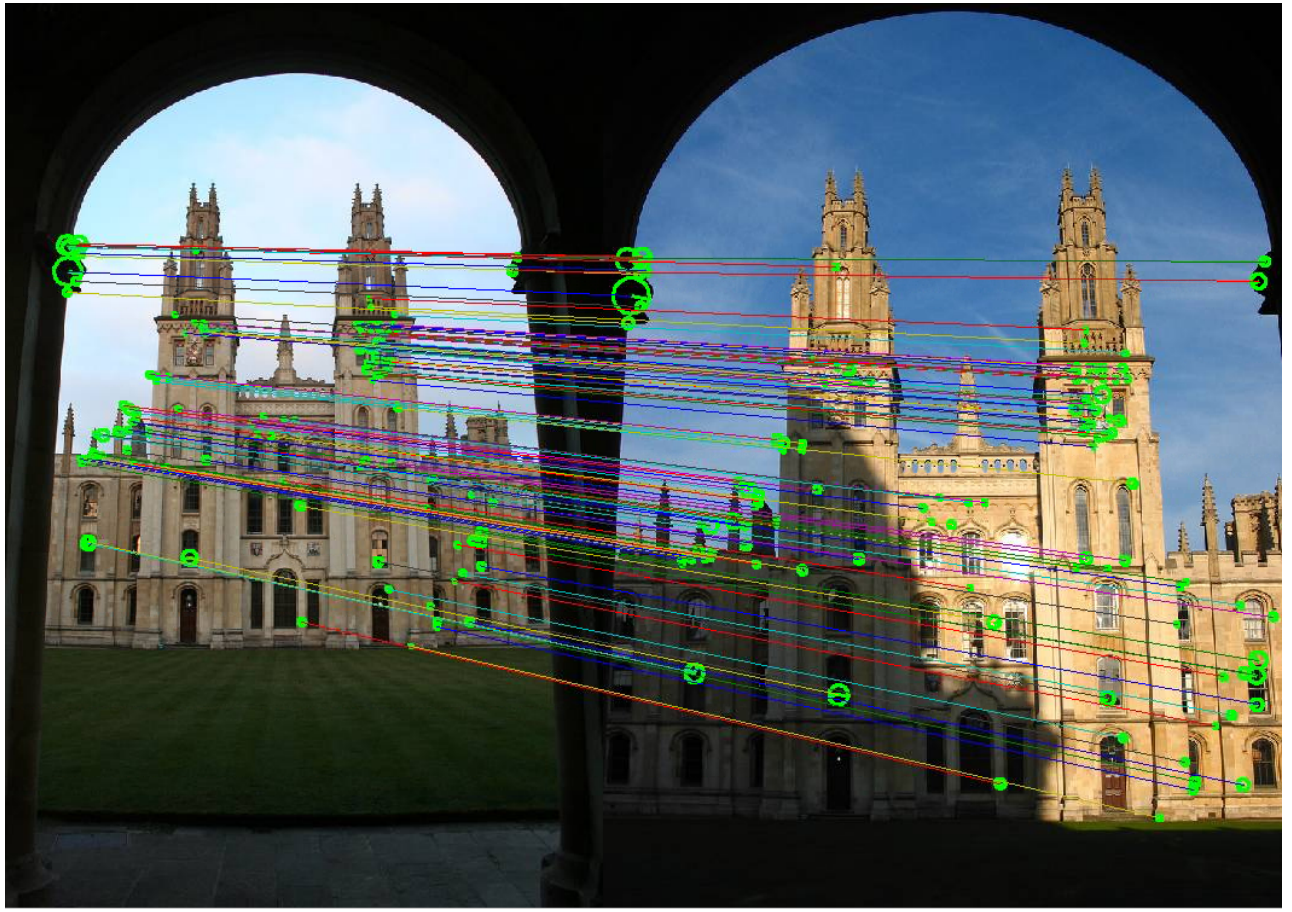
Some slides are borrowed from Stanford CS131.

Keypoint Localization



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

Applications: Image Matching



Separately detect keypoints and then find matching.

Slide borrowed from Stanford CS131

What Points are Keypoints?

- Saliency: interesting points

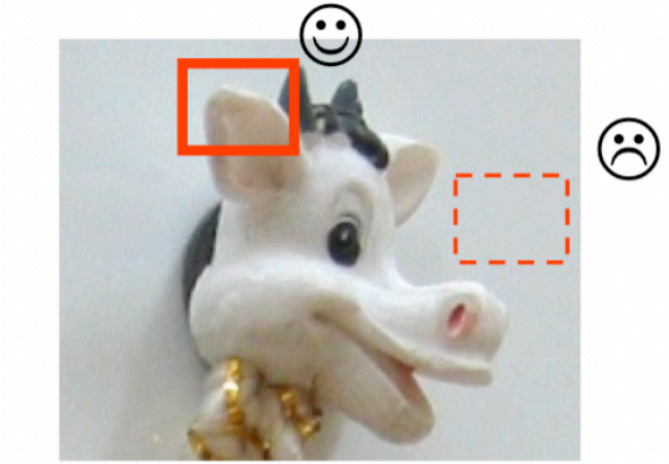
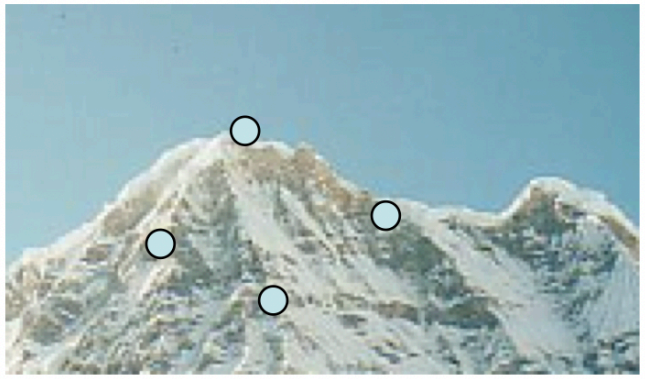


Image borrowed from Stanford CS131

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

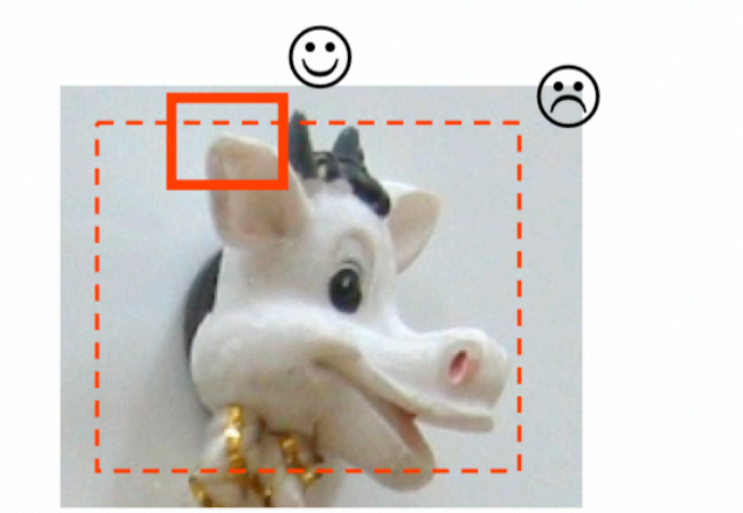
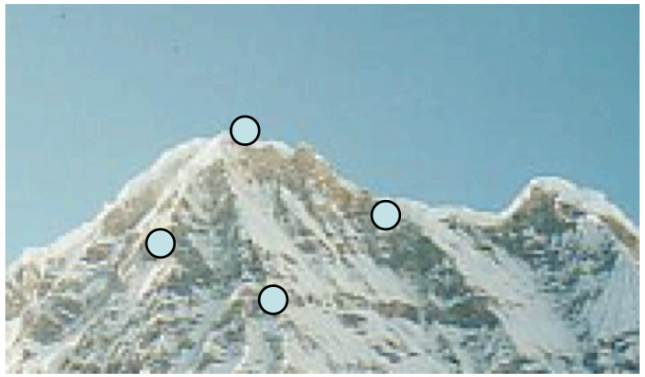


Image borrowed from Stanford CS131

More Requirements

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

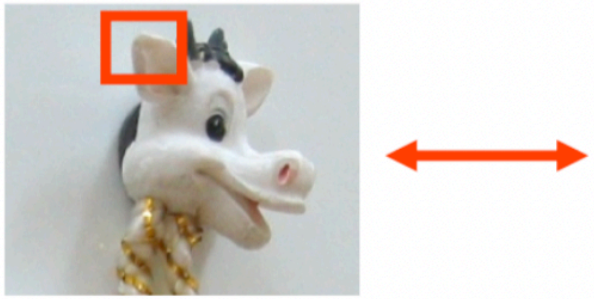


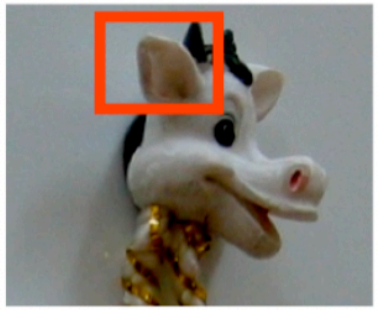
No chance to match!

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





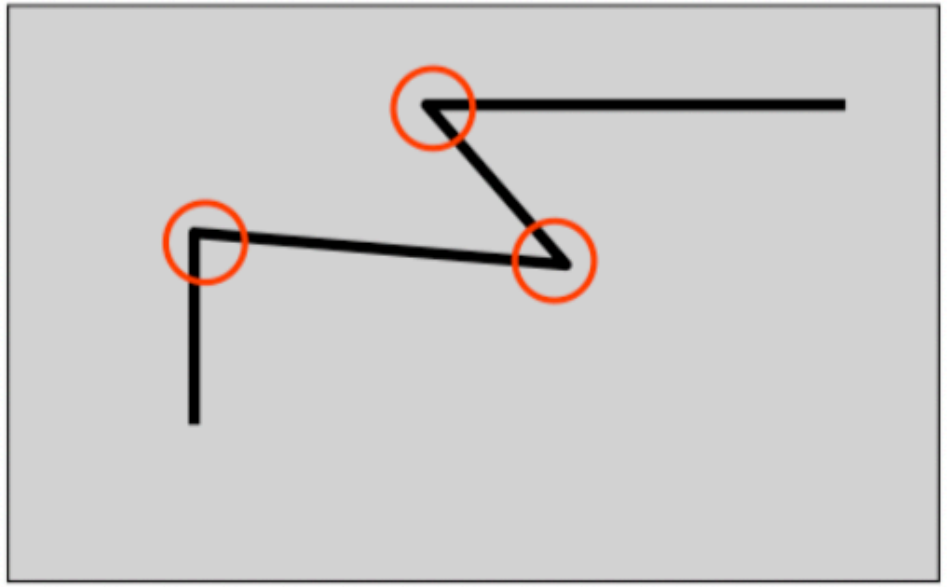
Illumination invariance

Scale invariance

Pose invariance
•Rotation
•Affine

Image borrowed from Stanford CS131

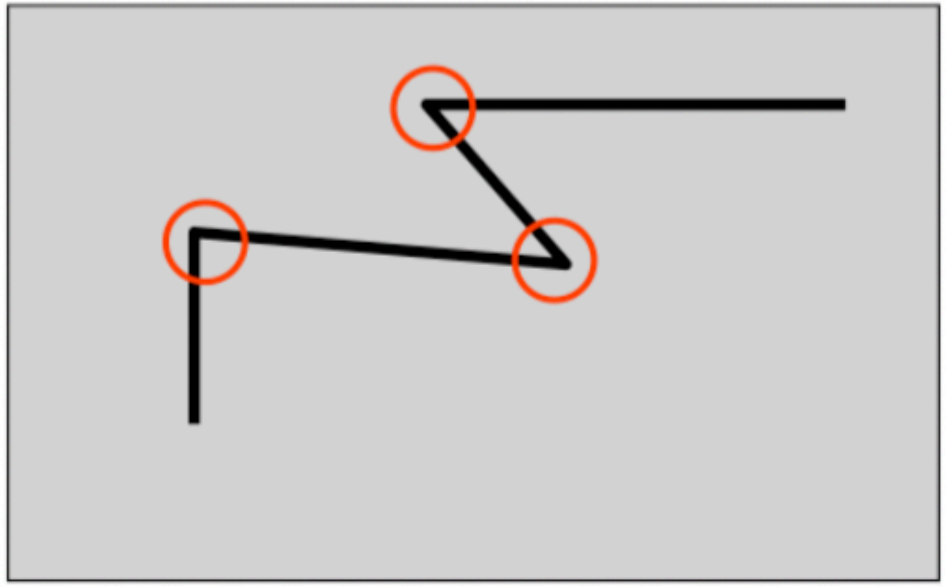
Corners as Keypoints



- 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

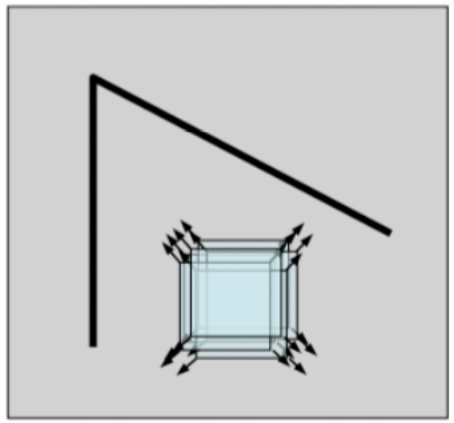


- The key property of a corner: In the region around a corner, image gradient has two or more dominant directions

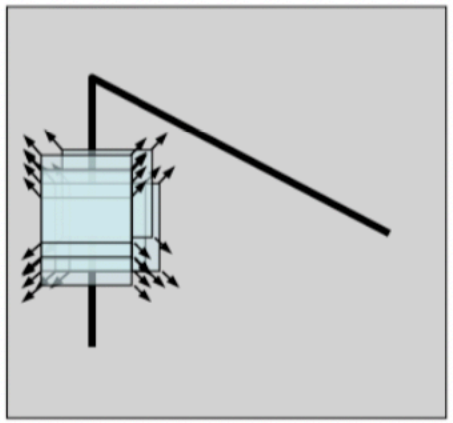
Image borrowed from Stanford CS131

The Basic Idea of Harris Corner

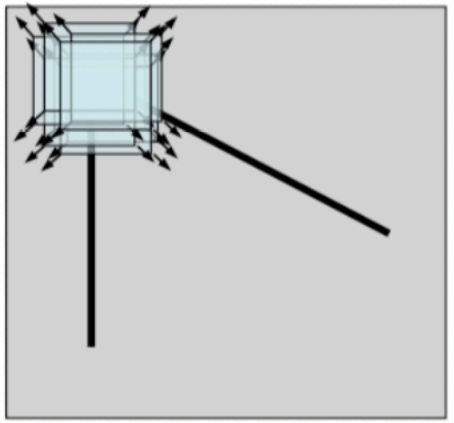
- Move a window and explore intensity changes within the window



Flat region: no change in all directions



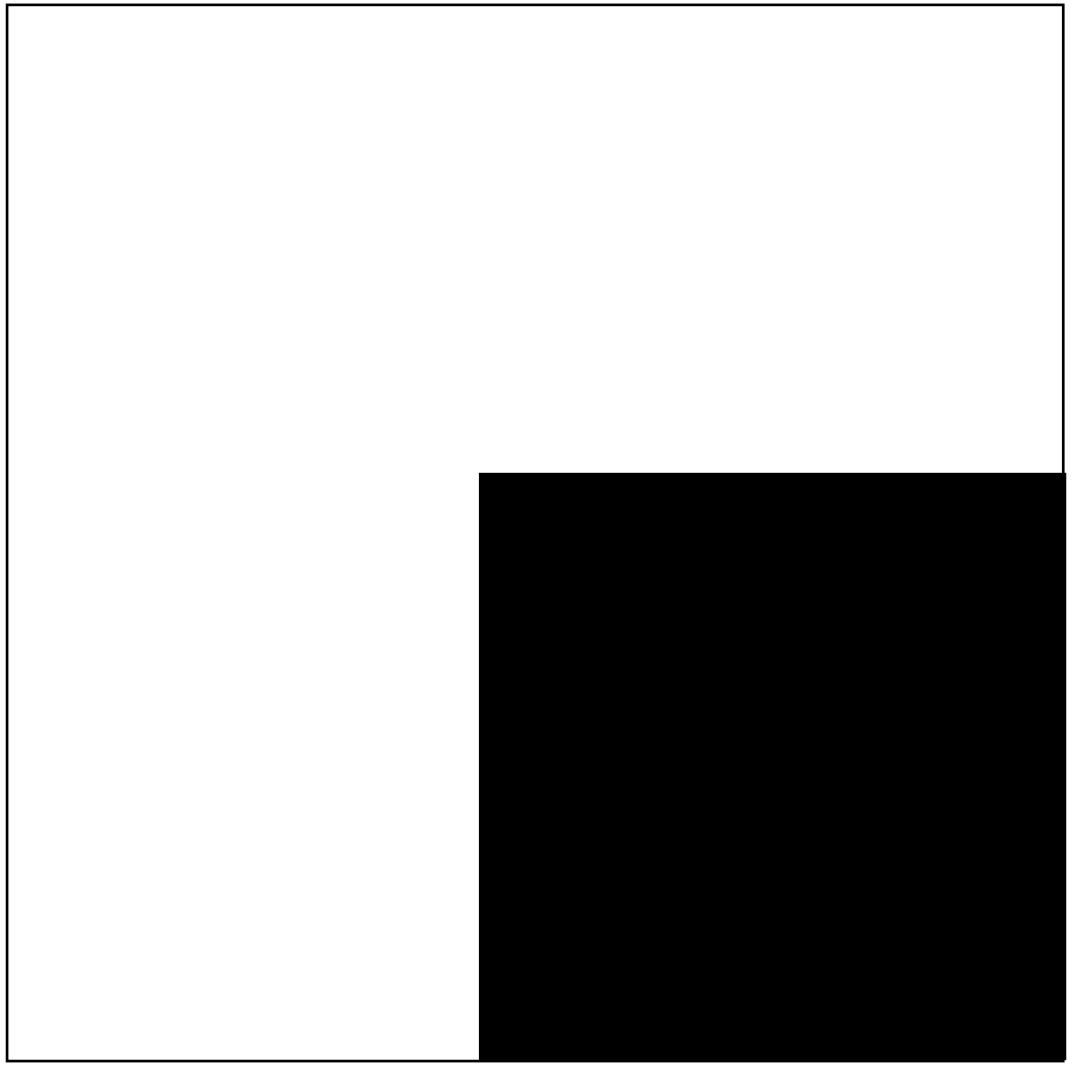
Edge: no change along the edge direction



Corner: significant change in all directions

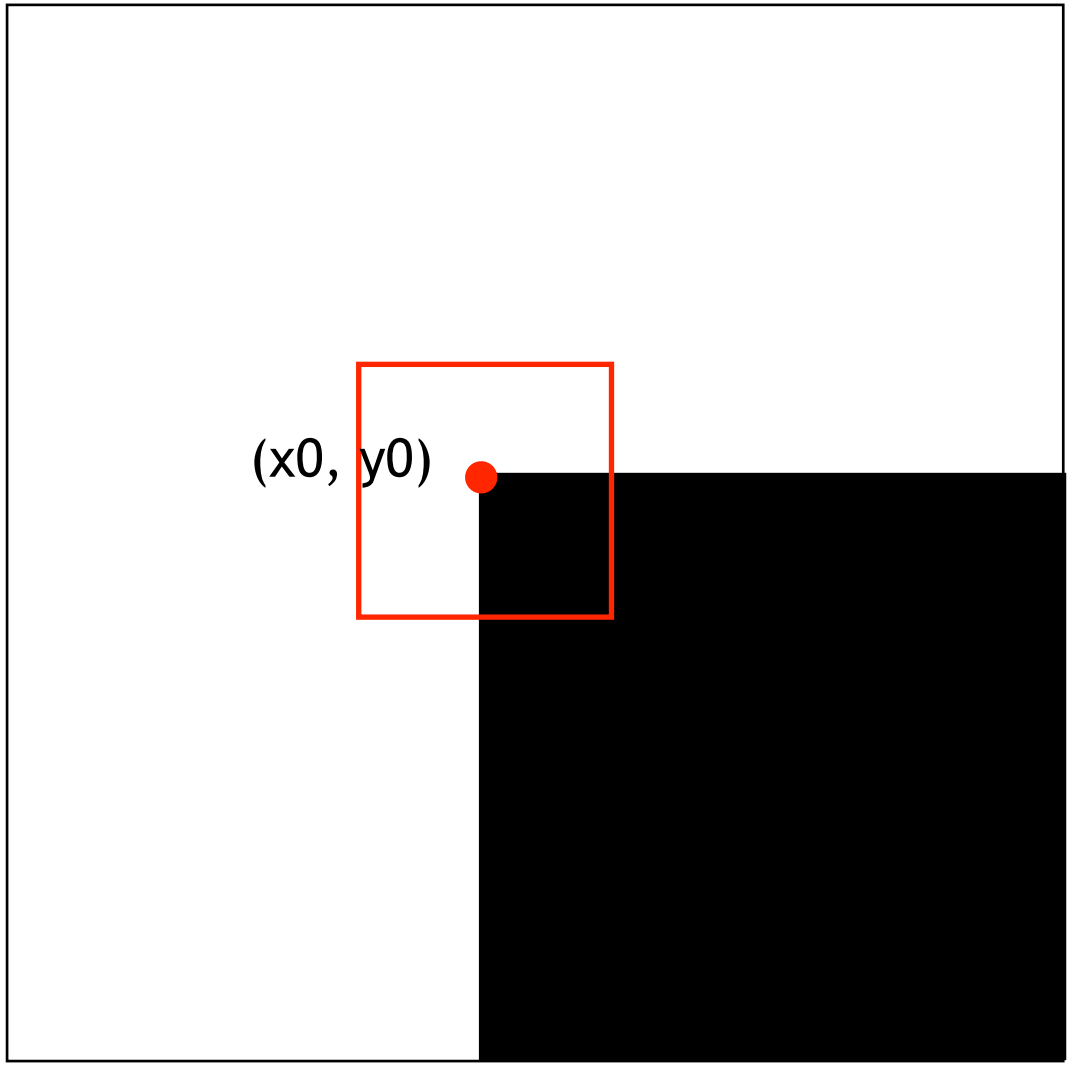
Image borrowed from Stanford CS131

The Basic Idea of Harris Corner



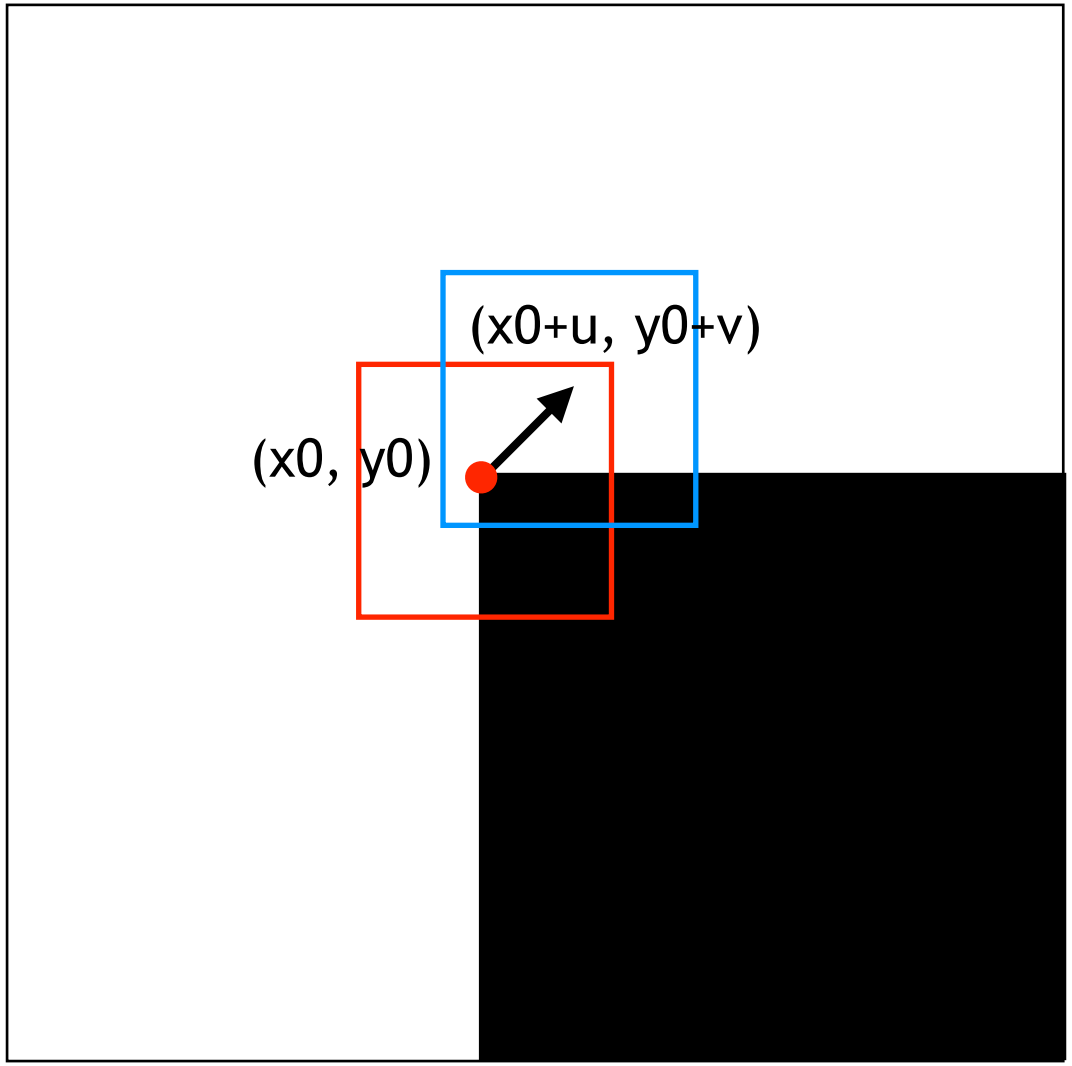
Original image

The Basic Idea of Harris Corner



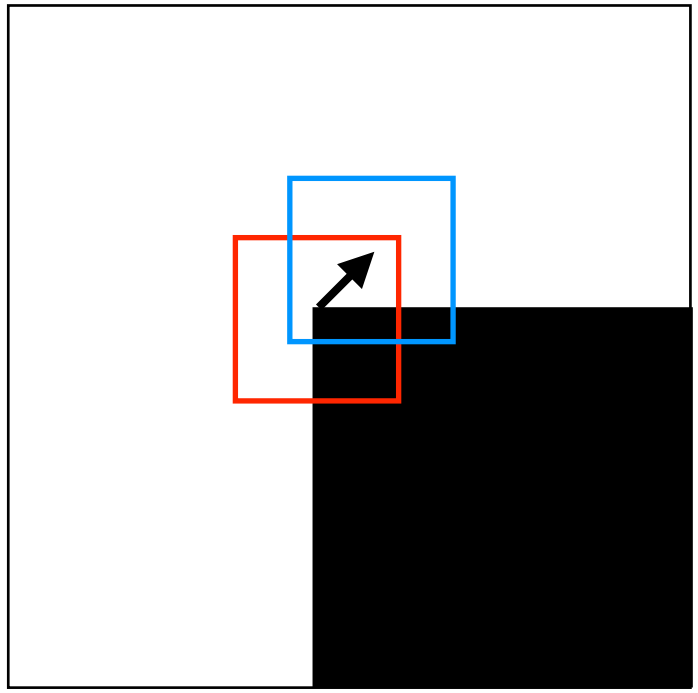
Local neighborhood of a corner point (x0, y0)

The Basic Idea of Harris Corner



Move along direction (u, v)

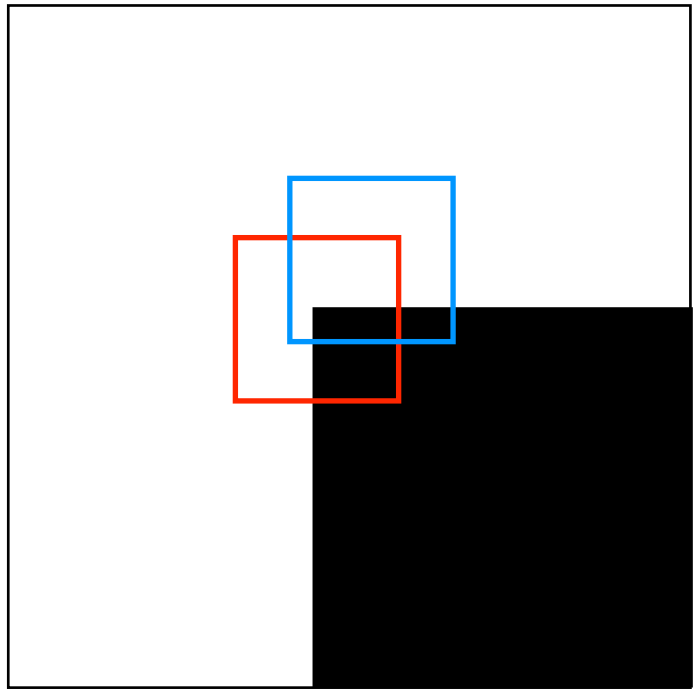
The Basic Idea of Harris Corner



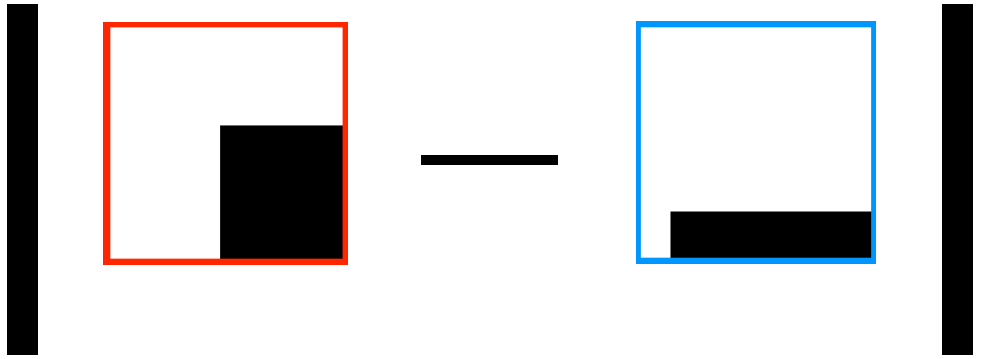
Local neighborhood of a corner point (x0, y0)

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

The Basic Idea of Harris Corner



Change along direction (u, v) =

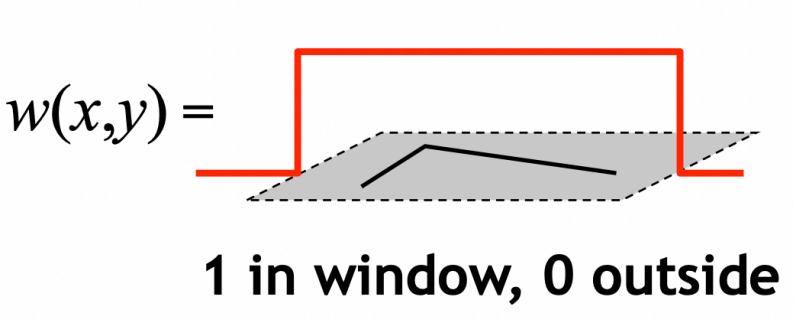


$$= \sum_{(x,y)\in N} [I(x+u, y+v) - I(x,y)]^2$$

Where N is the neighborhood of (x0, y0)

Notation

Rectangle Window Function

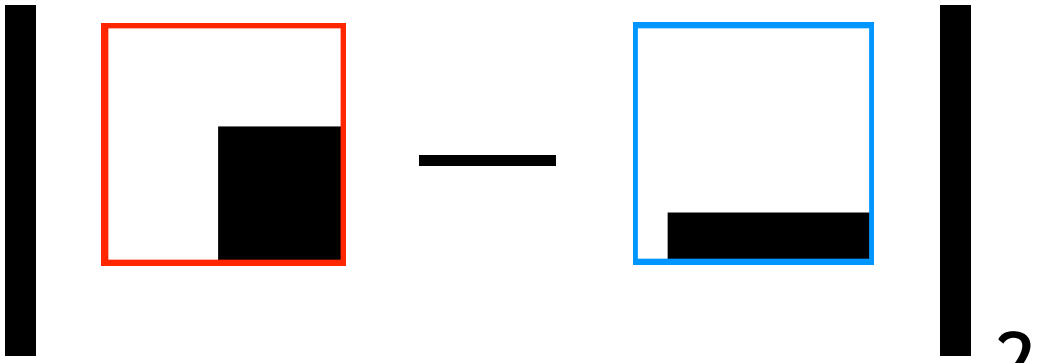


$$w'(x, y) = w(x - x_0, y - y_0)$$

Square intensity difference

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

The Basic Idea of Harris Corner



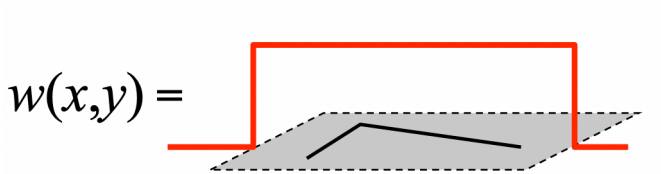
$$= \sum_{(x,y)\in N} [I(x+u, y+v) - I(x,y)]^2$$

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

$$= \sum_{x,y} w'(x,y) D(x,y)$$

$$= w' * D$$

Rectangle Window Function



1 in window, 0 outside

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

Square intensity difference

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

Harris Detector

First-order Taylor expansion: $I[x+u, y+v] - I[x,y] \approx I_x u + I_y v$

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

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

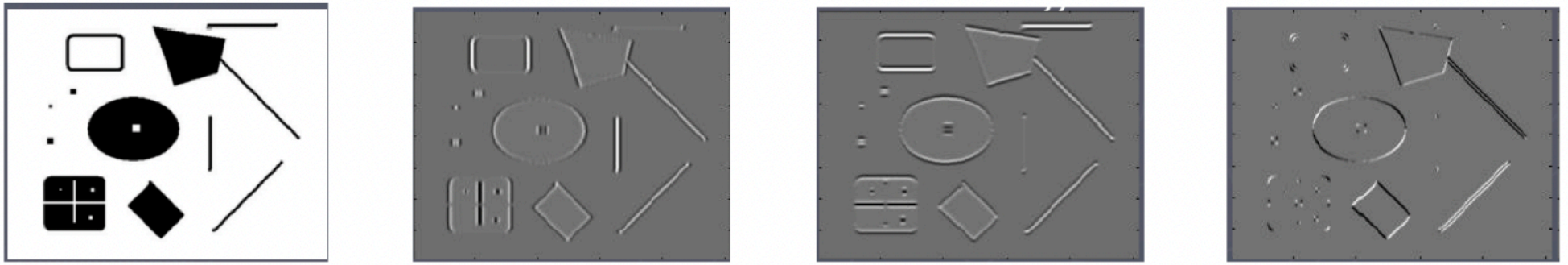


Image I I_x I_y $I_x I_y$

Image borrowed from Stanford CS131

Harris Detector

If we are checking the corner at (x_0, y_0), then the change along direction (u0, v0) 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

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

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

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

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

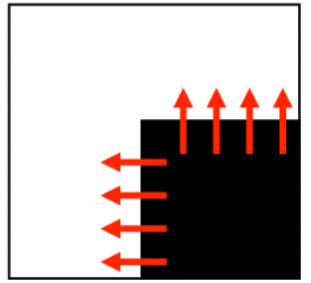


Image borrowed from Stanford CS131

Harris Detector

- General case:

  since M is a symmetric matrix, perform eigendecomposition:

$$M(x,y) = w' * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \qquad (\lambda_1 \geq 0, \ \ \lambda_2 \geq 0)$$

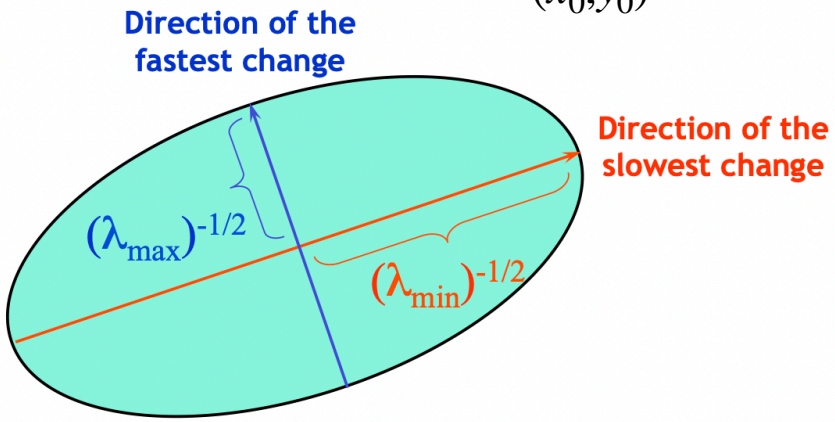
R is an orthogonal matrix, λs are the eigenvalues of M!

Harris Detector

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

$$M(x,y) = w' * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R \qquad (\lambda_1 \geq 0, \quad \lambda_2 \geq 0)$$

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

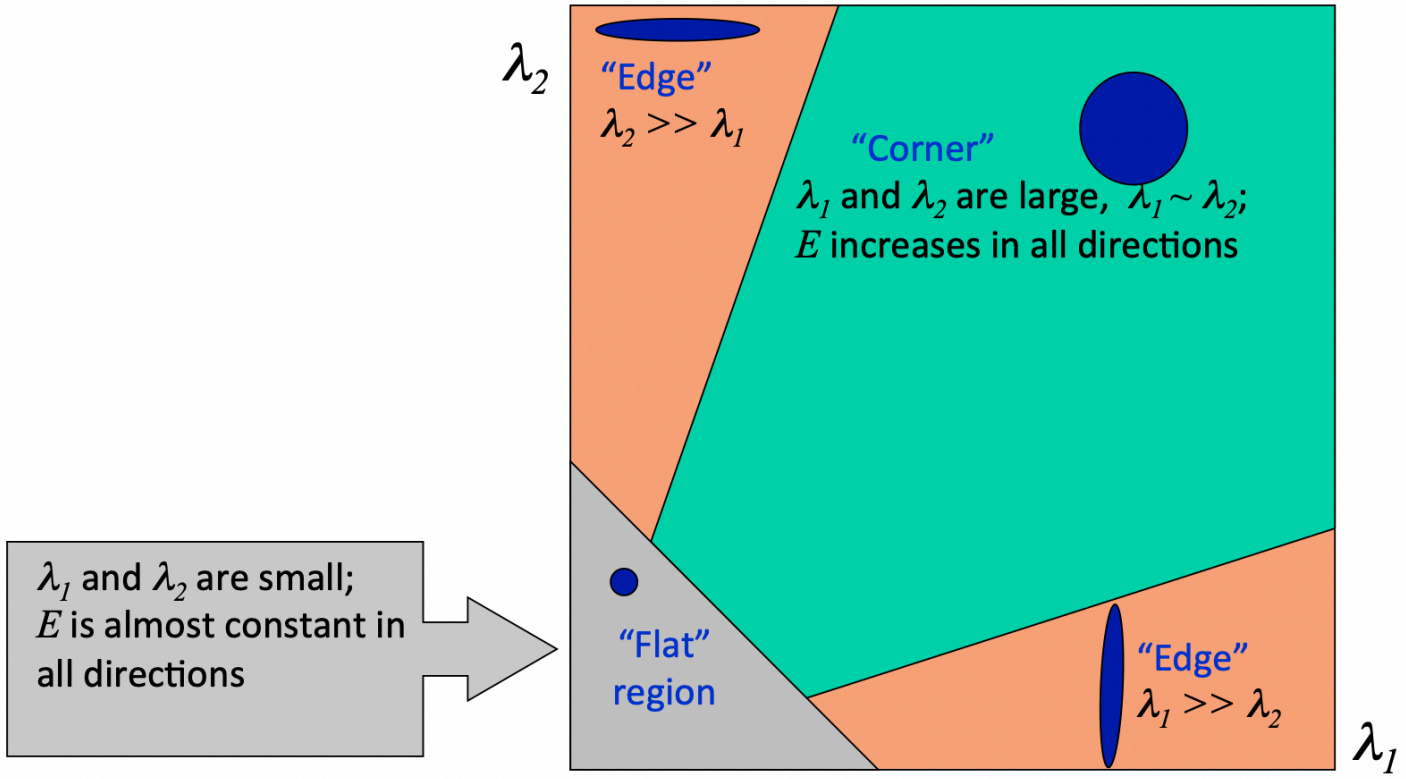


The energy landscape is a paraboloid!

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:

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

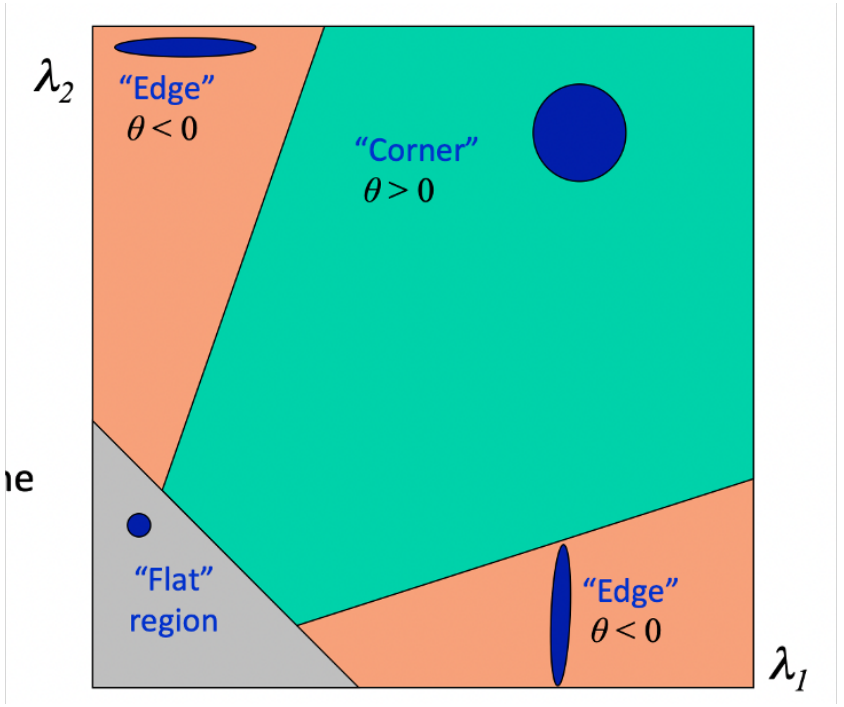
α in [0.04,0.06]

$$\frac{1}{k} < \frac{\lambda_1}{\lambda_2} < k \qquad \lambda_1, \lambda_2 > b$$

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

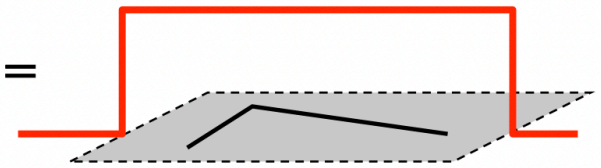
$$= det(M) - \alpha Trace(M)^2 - t$$

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



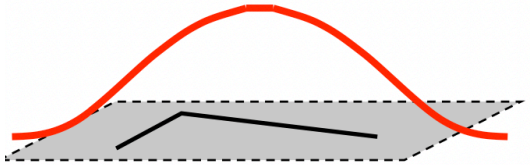
Choices of Window Functions

Rectangle window

$w(x,y) =$ 

1 in window, 0 outside

or



Gaussian

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

Not rotation-invariant.

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

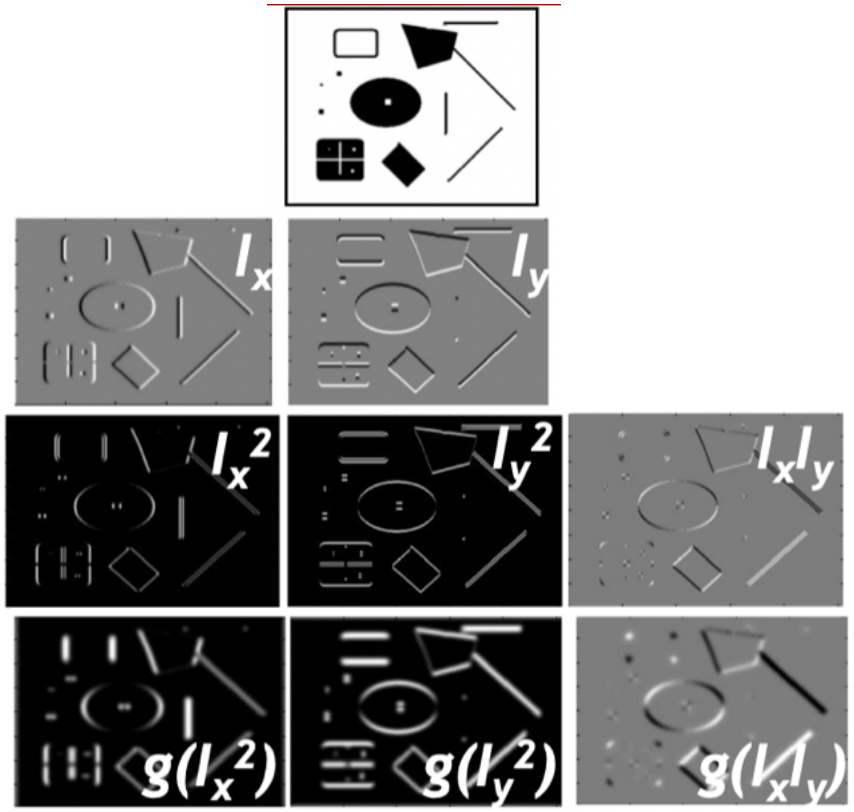
Rotation-invariant.

Summary of Harris Detector

1. Image derivatives

2. Square of derivatives

3. Rectangle window or Gaussian filter



4. Corner response function

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

5. Non-maximum suppression

Step-by-Step Harris Detector

- Input: two images

Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Compute corner response θ

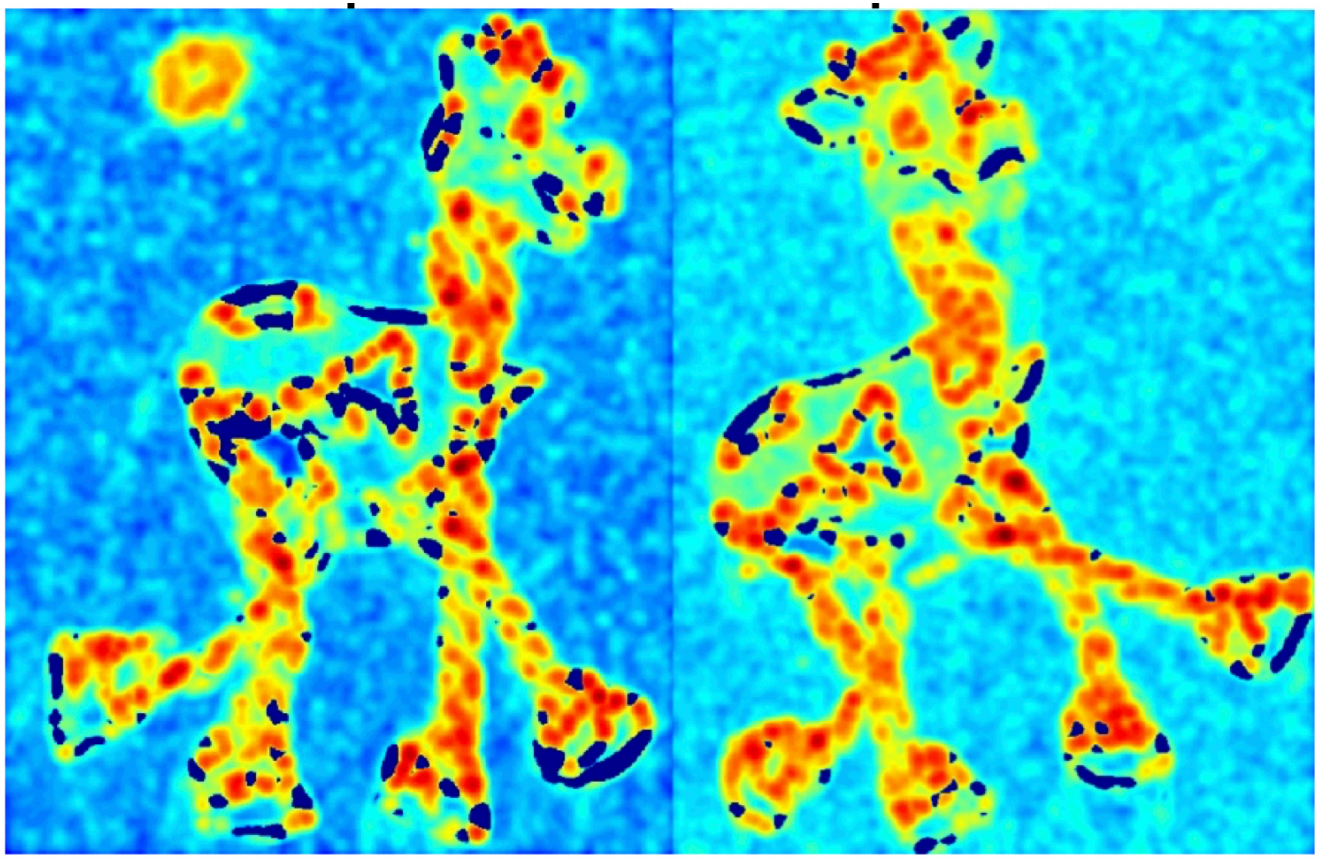


Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Thresholding and perform non-maximal suppression

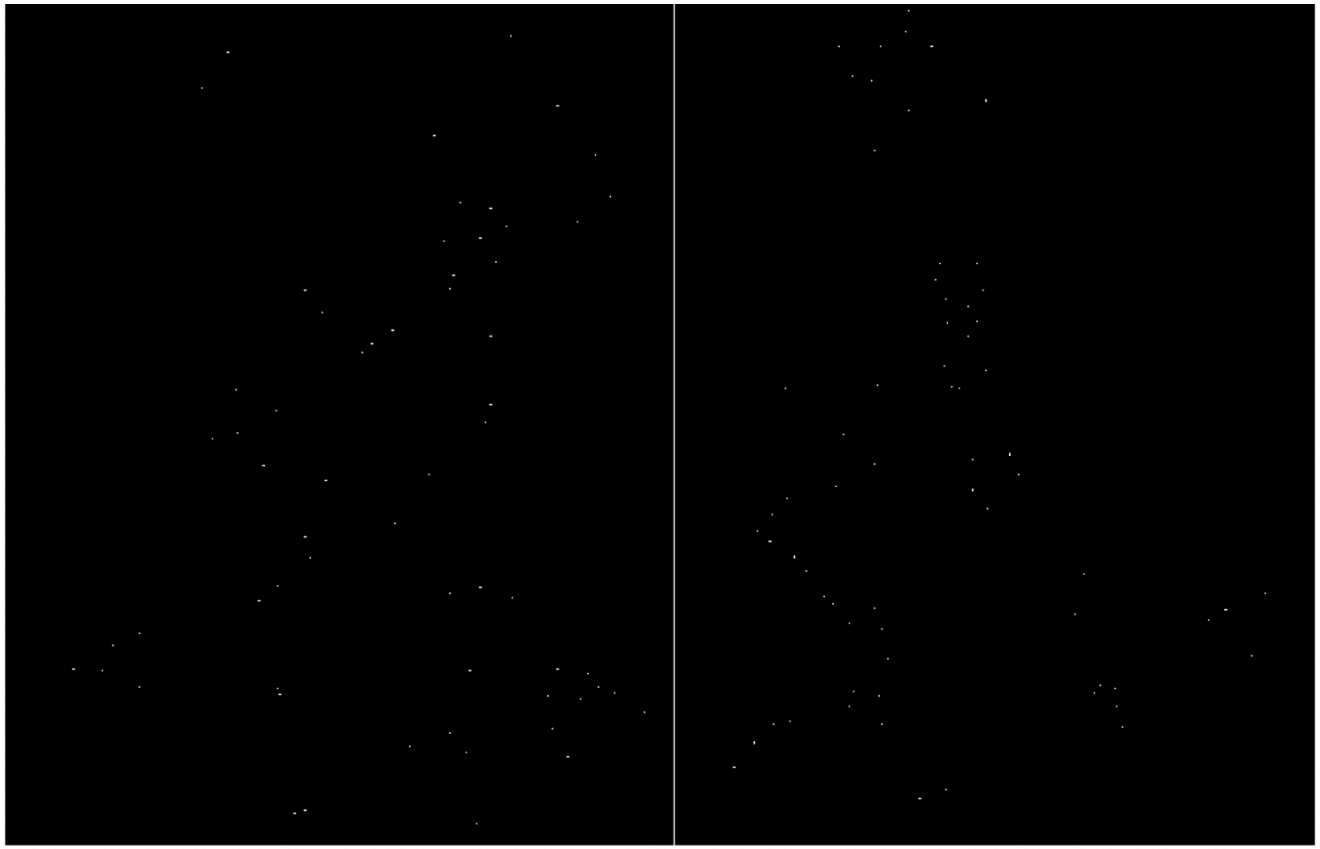


Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Results

Image borrowed from Stanford CS131

Properties of Harris Detector

- Corner response is equivariant with both translation and image rotation.

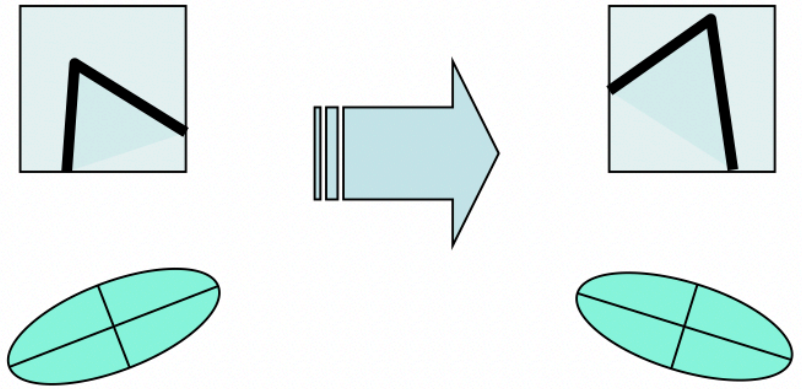


Image borrowed from Stanford CS131

Properties of Harris Detector

- Corner response is equivariant with both translation and image rotation.
- Not invariant to scale.

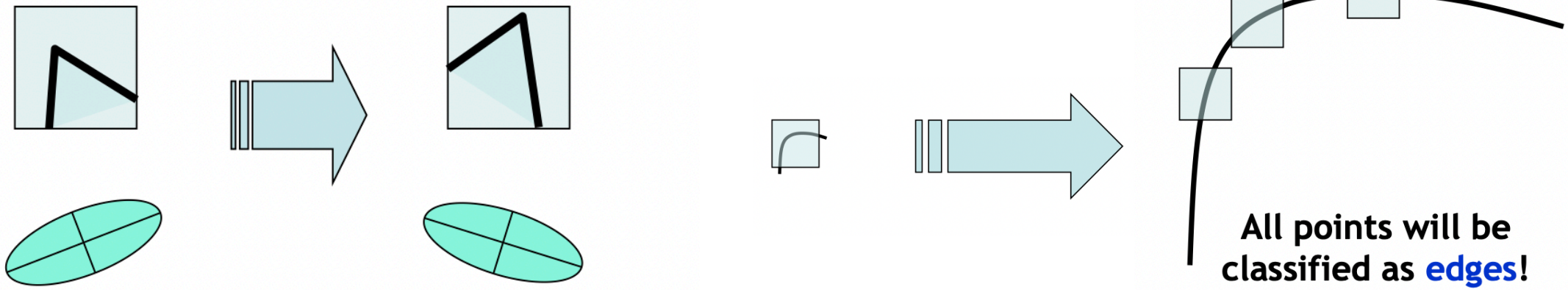
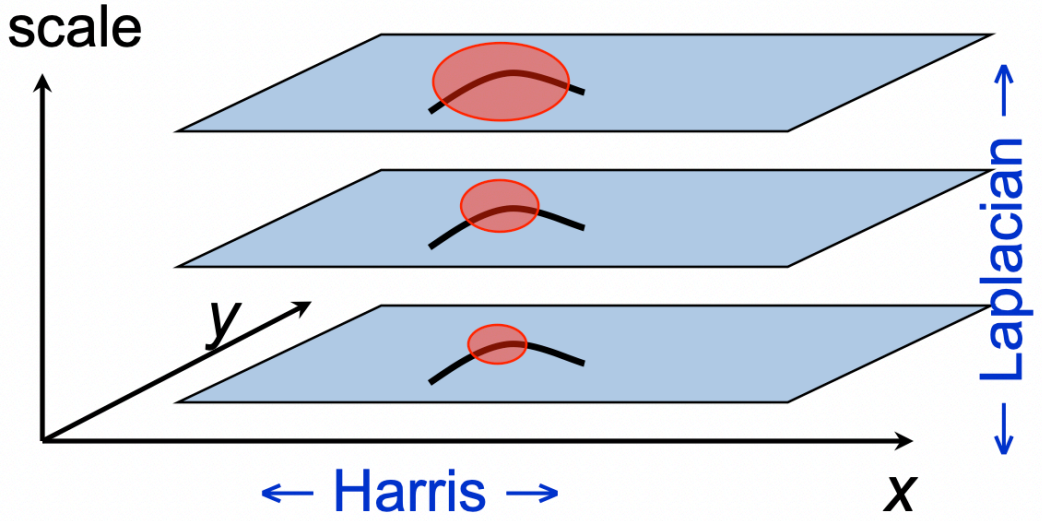


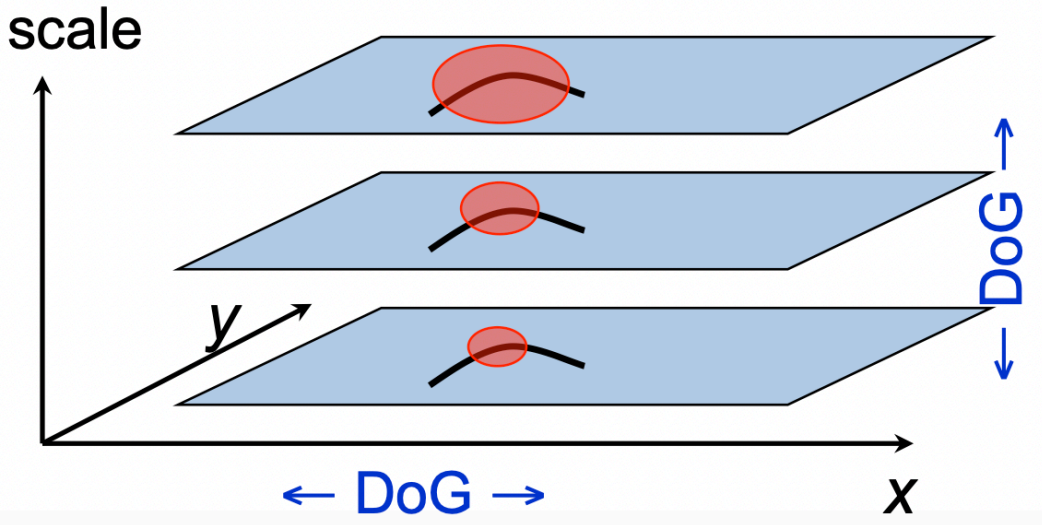
Image borrowed from Stanford CS131

Scale Invariant Detectors

- Harris-Laplacian[1]
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale



- SIFT (Lowe)[2]
  Find local maximum of:
  - Difference of Gaussians in space and scale



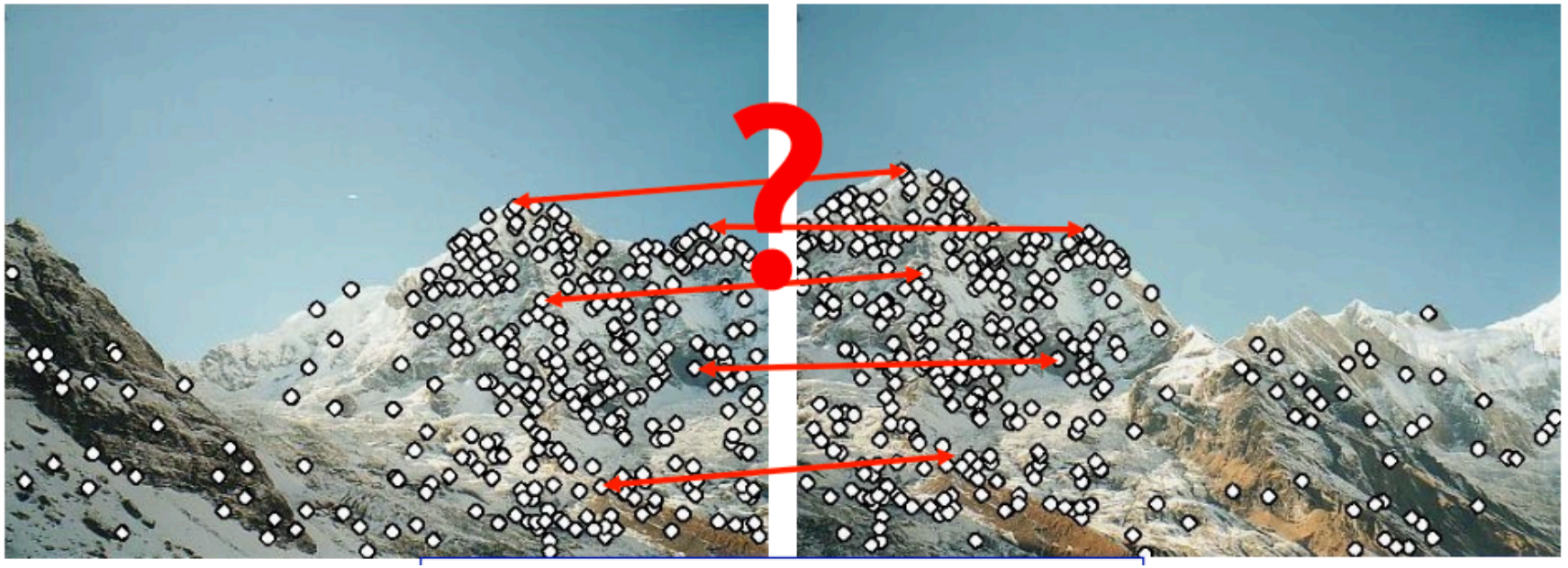
Slides borrowed from Stanford CS131

Feature Description

Local Descriptors

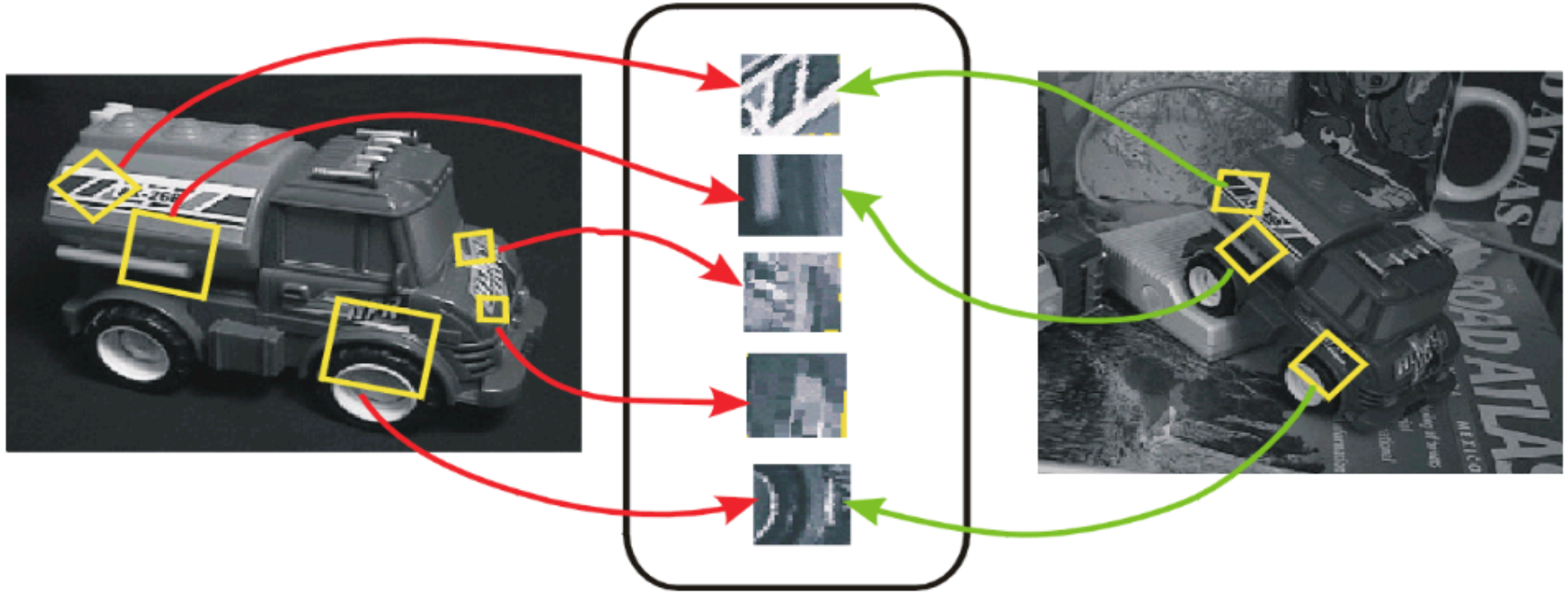
- We know how to detect points
- Next question:

How to *describe* them for matching?



Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



Feature

- A feature is any piece of information which is relevant for solving the computational task related to a certain application.

Useful features F:

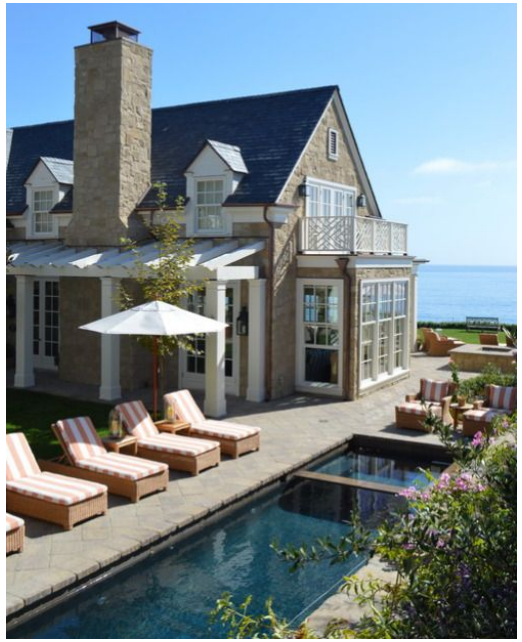
- Location
- Size
- Building time
- Current condition
- House style
- …

How much is this house?

Model

- Based on the features, we can build a model.

- Heuristic model:
  $y = (10 - \text{location}) \times \text{area}$



Useful features F:

- Location
- Size
- Building time
- Current condition
- House style
- …

How much is this house?

Model

- Based on the features, we can build a model.

- Heuristic model:
  $y = (10 - \text{location}) \times \text{area}$

- Parametric model:
  $y = \phi_\theta(F)$
  - when we have some observations, we can fit θ

Useful features F:

- Location
- Size
- Building time
- Current condition
- House style
- …

How much is this house?

Model

- Based on the features, we can build a model.

- Heuristic model:
  $y = (10 - \text{location}) \times \text{area}$

- Parametric model:
  $y = \phi_\theta(F)$

- Deep vision model $y = \phi_\theta(I)$

Useful features F:

- Location
- Size
- Building time
- Current condition
- House style
- …

How much is this house?

Topic Switch

- Low-level vision
  - Image processing
  - Edge/corner detection
  - Feature extraction
- Mid-level Vision
  - Grouping
  - Inferring scene geometry (3D reconstruction)
  - Inferring camera and object motion
- **High-level vision (where deep learning wins!)**
  - Object recognition
  - Scene understanding
  - Activity understanding

Next week: Lecture 4,
Deep Learning I