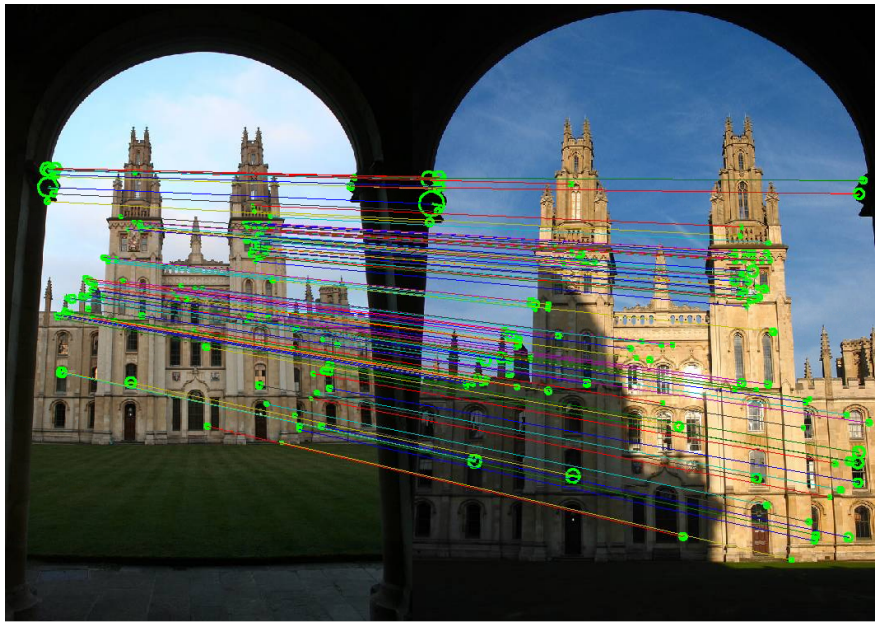
Introduction to Computer Vision

Lecture 3 - Classic Methods II and Deep Learning I

Prof. He Wang

Recap of Last Lecture

- Task 1: lane detection
- Techniques involved:
  - Edge detection
  - Line fitting

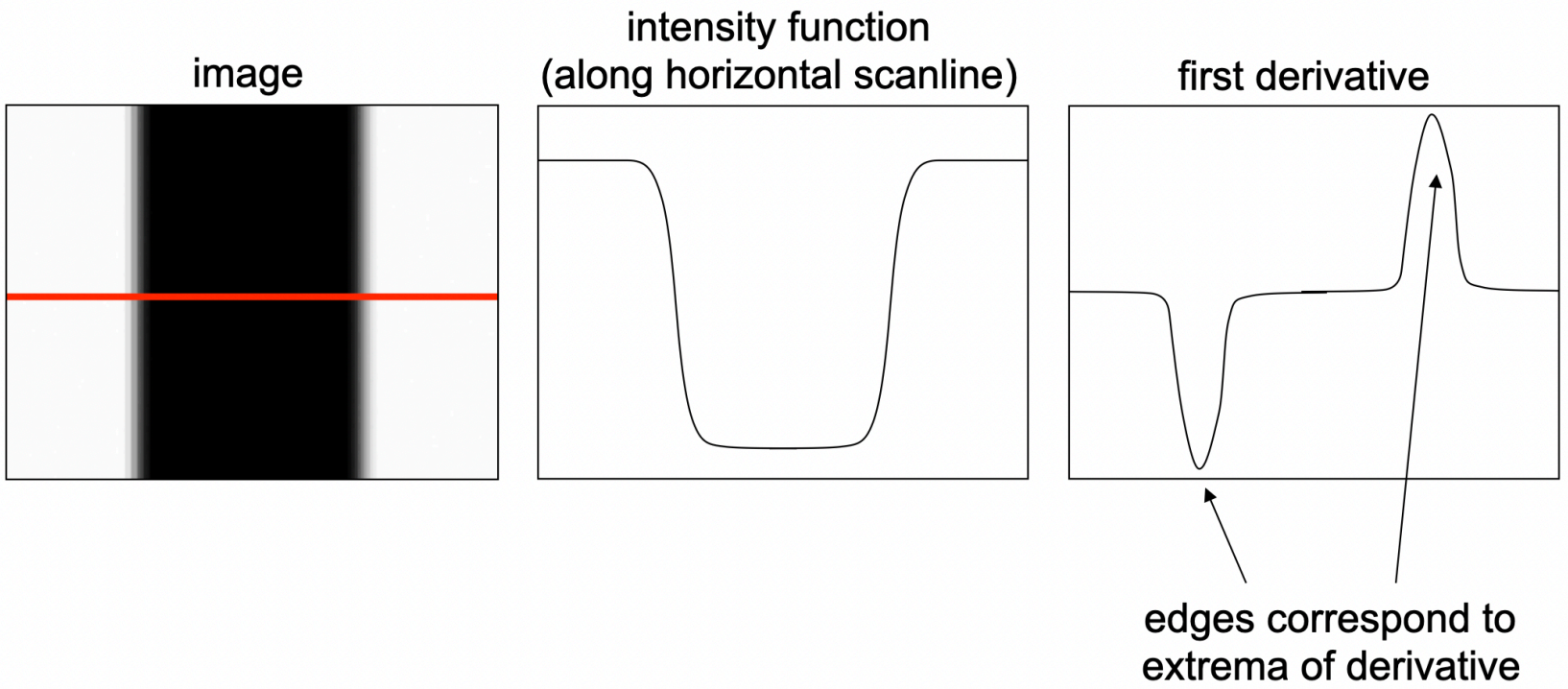


- Task 2: find matching
- Techniques involved:
  - Keypoint detection
  - Feature extraction

Edge Detection & Line Fitting

Edge

- What is an edge?



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.

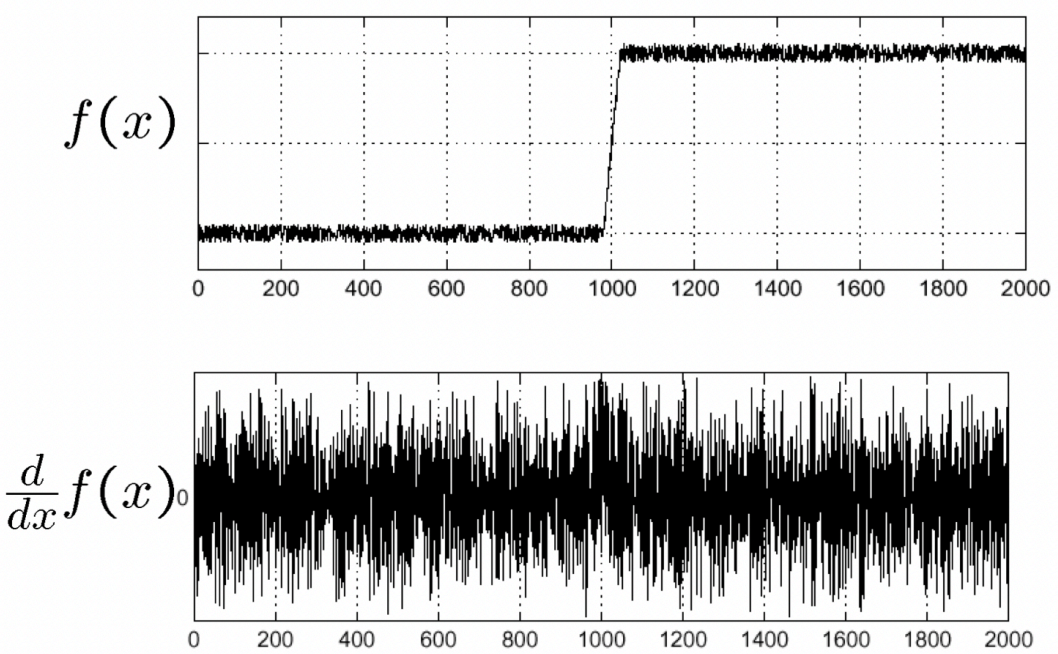
JJ. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Canny Edge Detector

- Three hyperparameters:
  - σ in Gaussian filter
  - maxVal and minVal in hysteresis thresholding

Effects of Noises

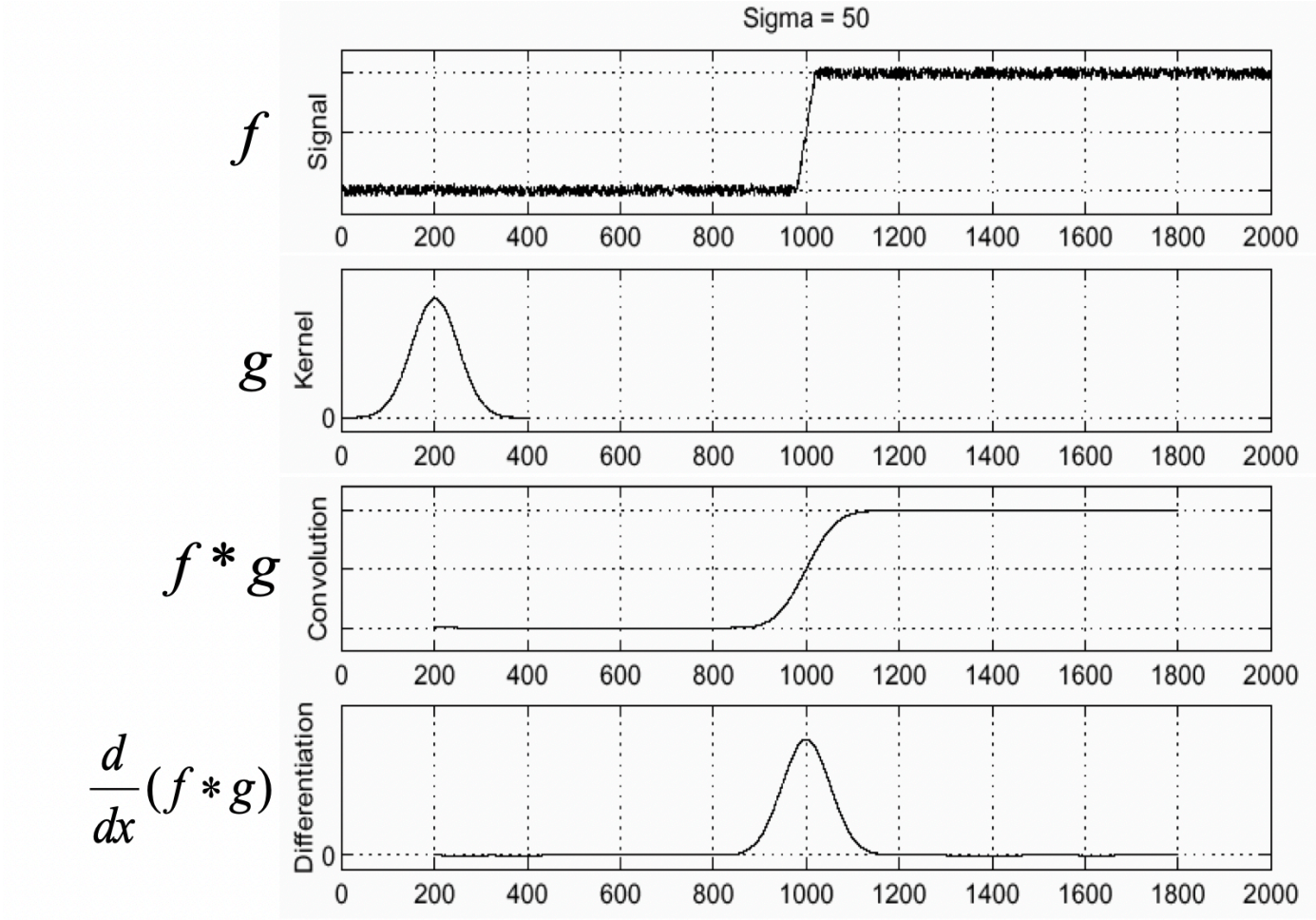
- Consider one row in the image



- 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



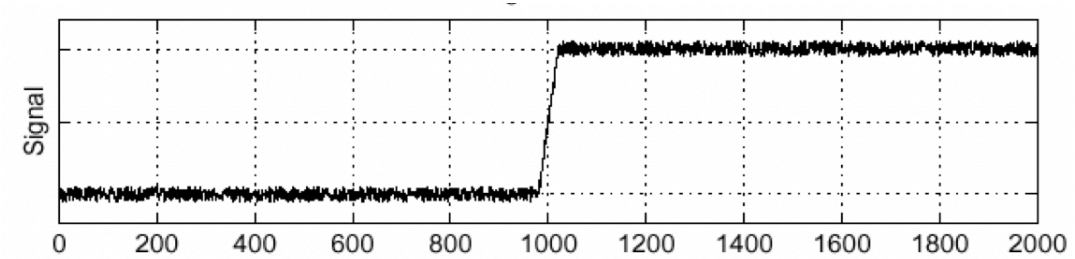
Source: Steven Seitz

Derivative Theorem of Convolution

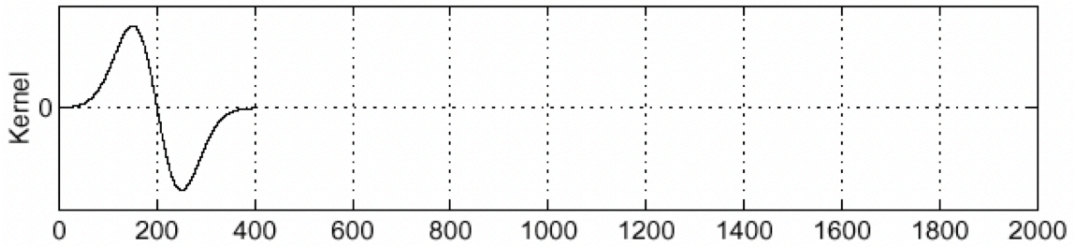
- Theorem:

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

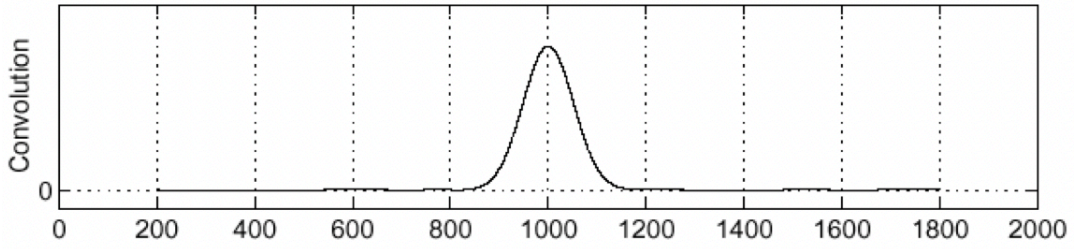
f



$\frac{d}{dx} g$



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



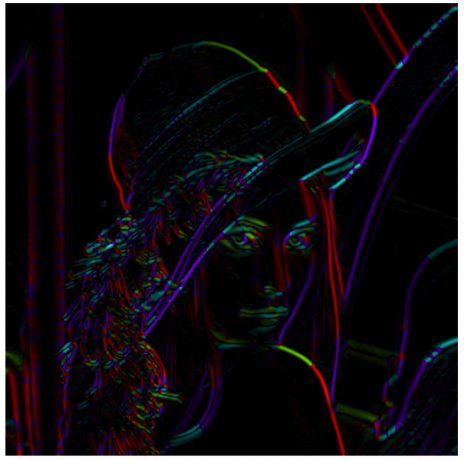
- Saves us one operation.

Compute Gradient

x-derivative of Gaussian

y-derivative of Gaussian

Gradient magnitude



Thresholding and Gradient orientation

Source: Feifei Li

Effects of σ

original

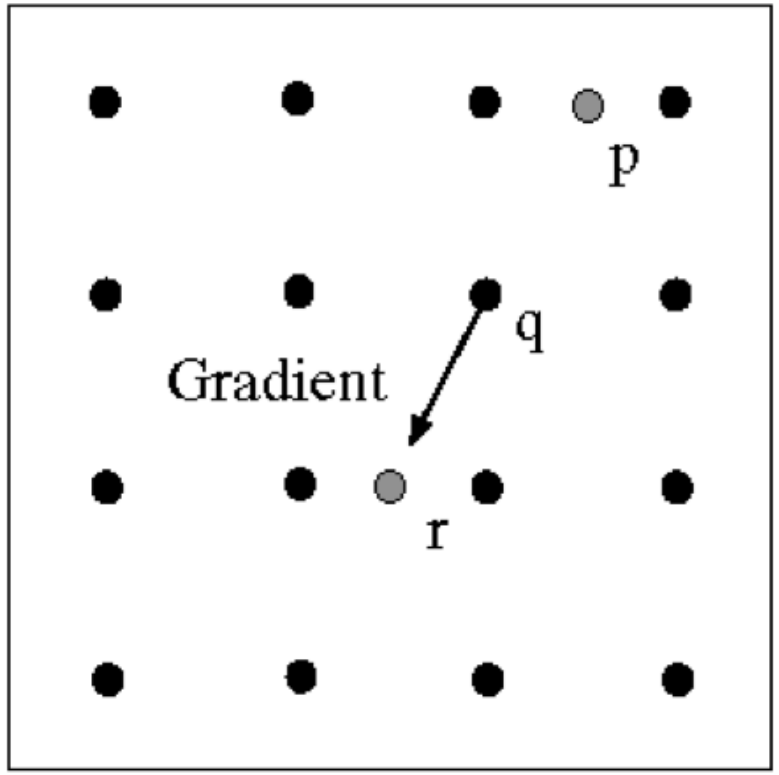
Canny with $\sigma = 1$

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

Non-Maximal Suppression (NMS)

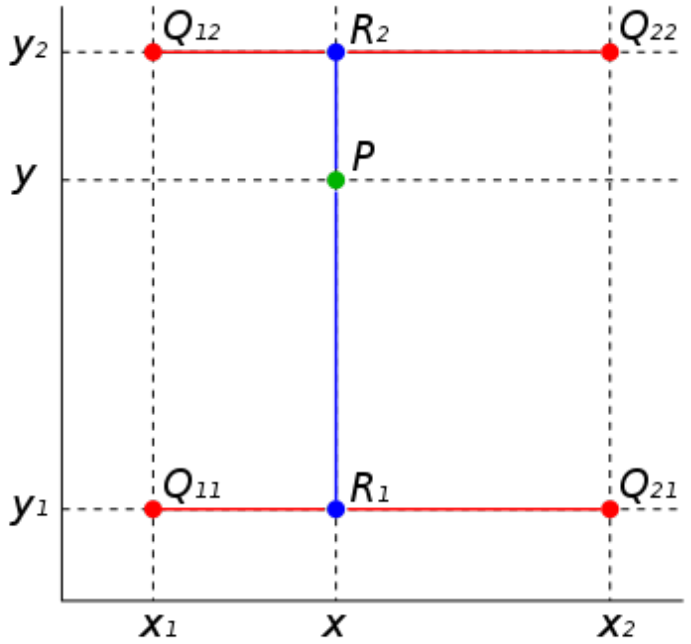


- For each point q on grids, compute the gradient $g(q)$.
- Move along the gradient to get two neighbors: $r = q + g(q), p = q - g(q)$
- Perform bilinear interpolation to get $g(p)$ and $g(r)$.
- If the magnitude of g(q) is larger than $g(p)$ and $g(r)$, q is a maximum that should be kept.

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*?



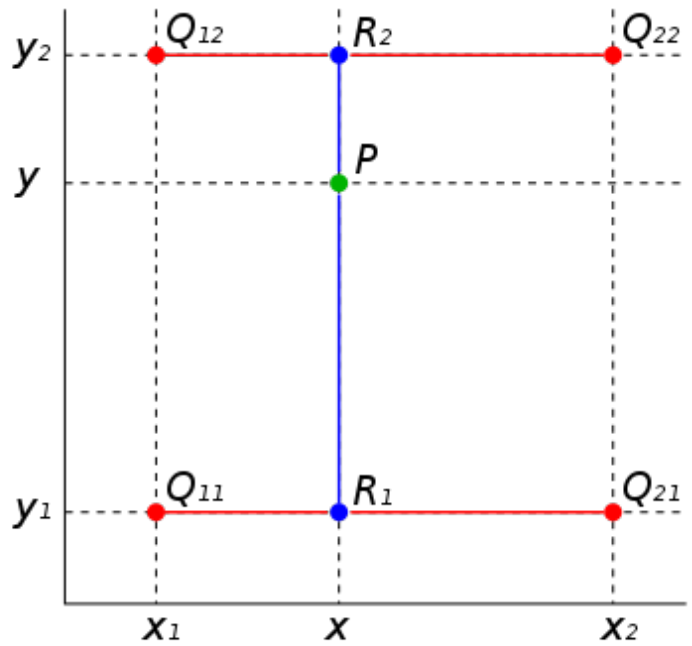
First, linear interpolate to obtain $f(R_1)$ and $f(R_2)$

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

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

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 $f(R_2)$

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

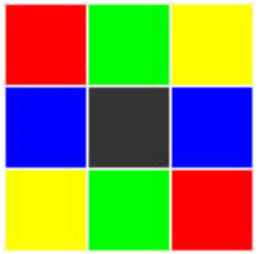
$$R_2: \quad f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

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

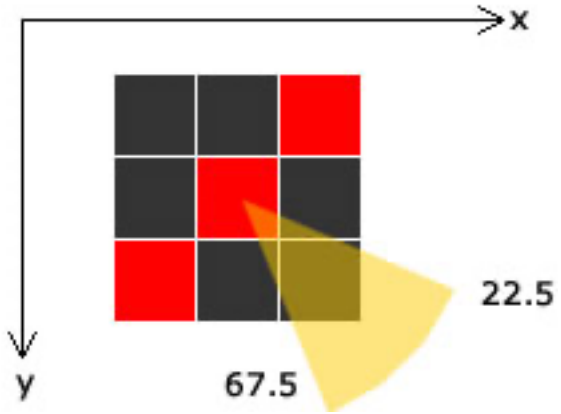
$$\begin{aligned} P: \quad f(x, y) &= \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2) \\ &= \frac{y_2 - y}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \right) + \frac{y - y_1}{y_2 - y_1} \left(\frac{x_2 - x}{x_2 - x_1} f(Q_{12})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

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

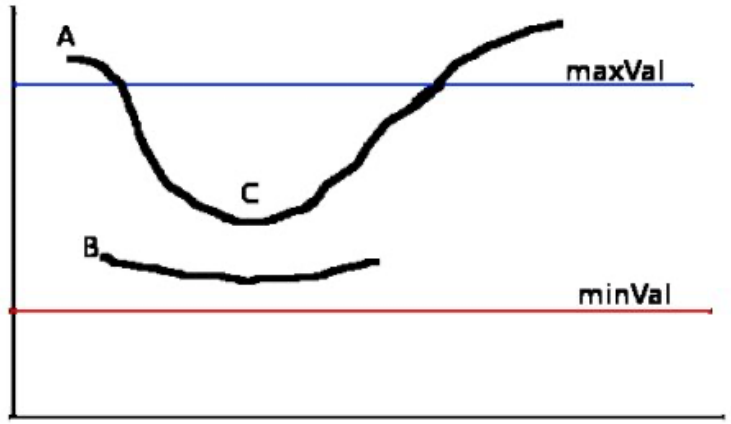
- Use a high threshold (maxVal) to start edge curves and a low threshold (minVal) to continue them.
  - Pixels with gradient magnitudes $>$ maxVal should be reserved
  - Pixels with gradient magnitudes $<$ minVal should be removed.
  - How to decide maxVal and minVal? Examples:
    - maxVal = 0.3 $\times$ 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!

Summary of Canny Edge Detection

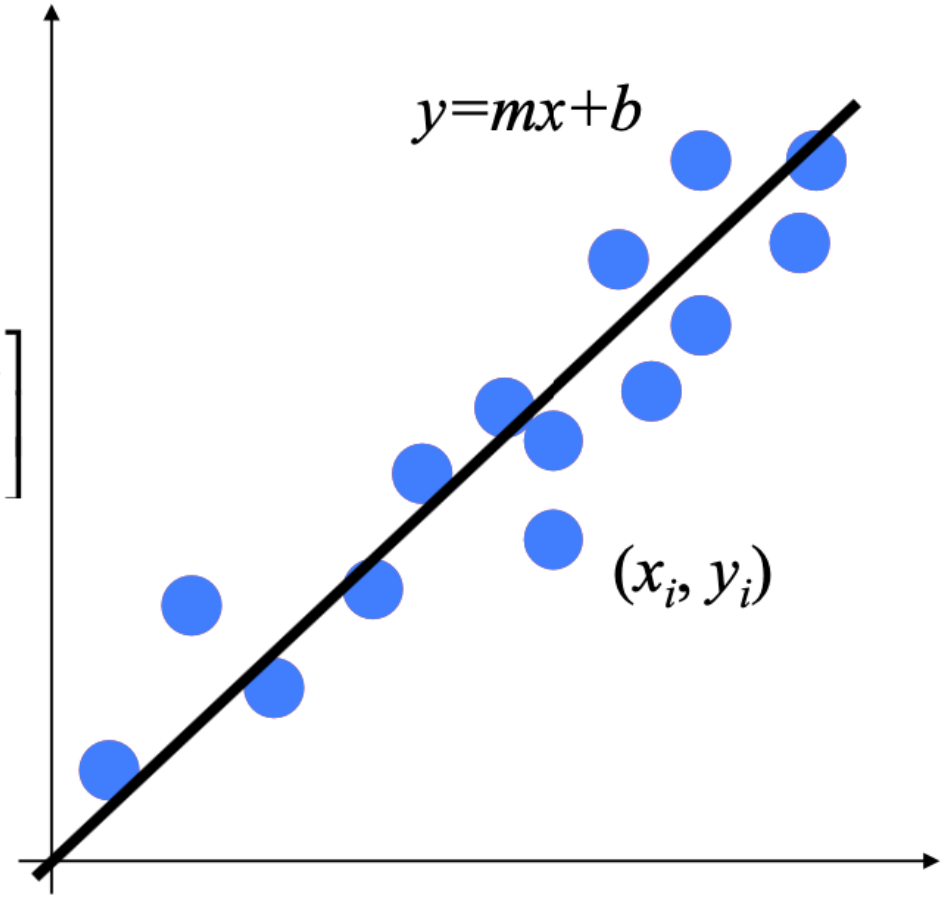
- Edge: where pixel intensity changes drastically
- Jointly detecting edge and smoothing by convolving with the derivative of a Gaussian filter
- Non-maximal suppression
- Thresholding and linking (hysteresis):

Line Fitting: 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 $ax+by=d$

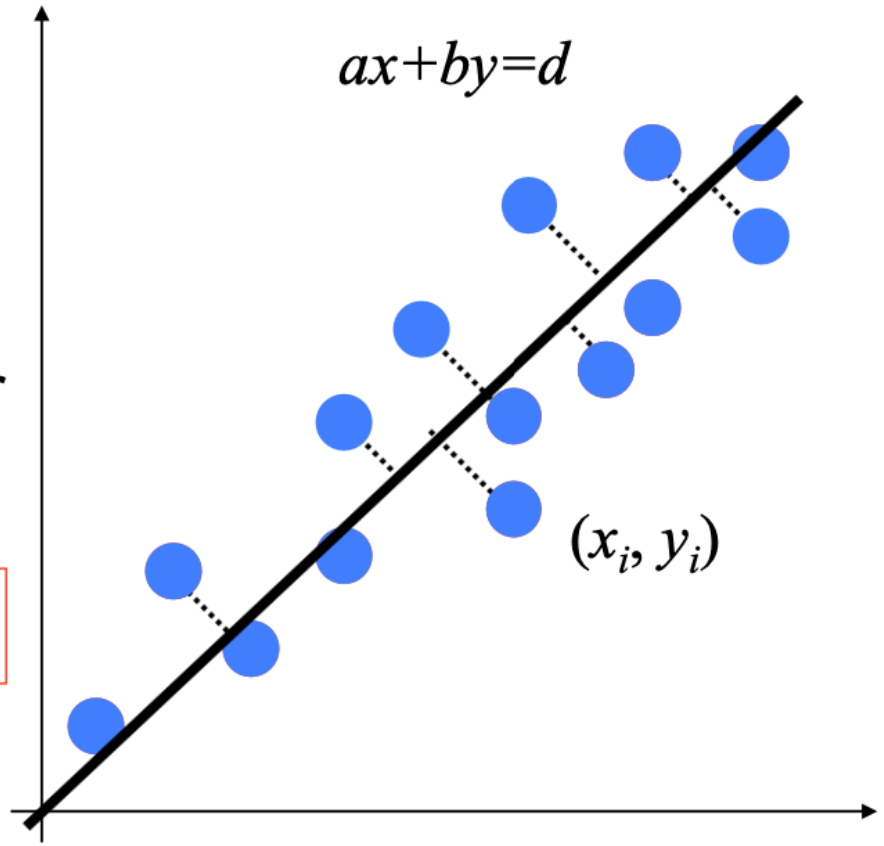
- Find (a, b, d) to minimize the sum of squared perpendicular distances

[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 using SVD:

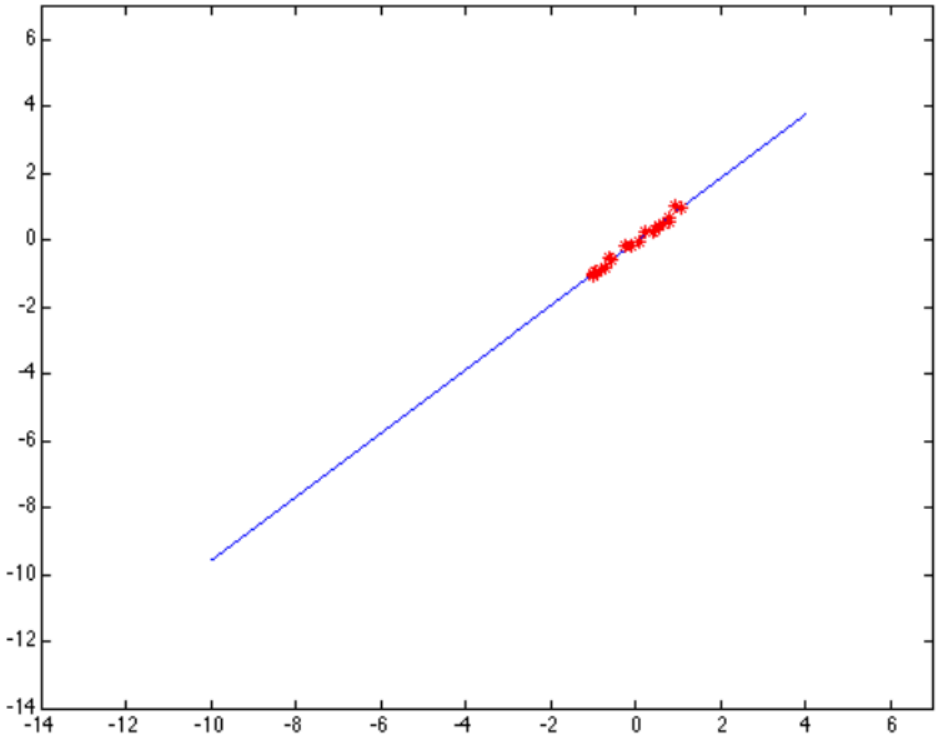
$$\mathrm{A\,h} = 0 \qquad \text{A is rank deficient}$$

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

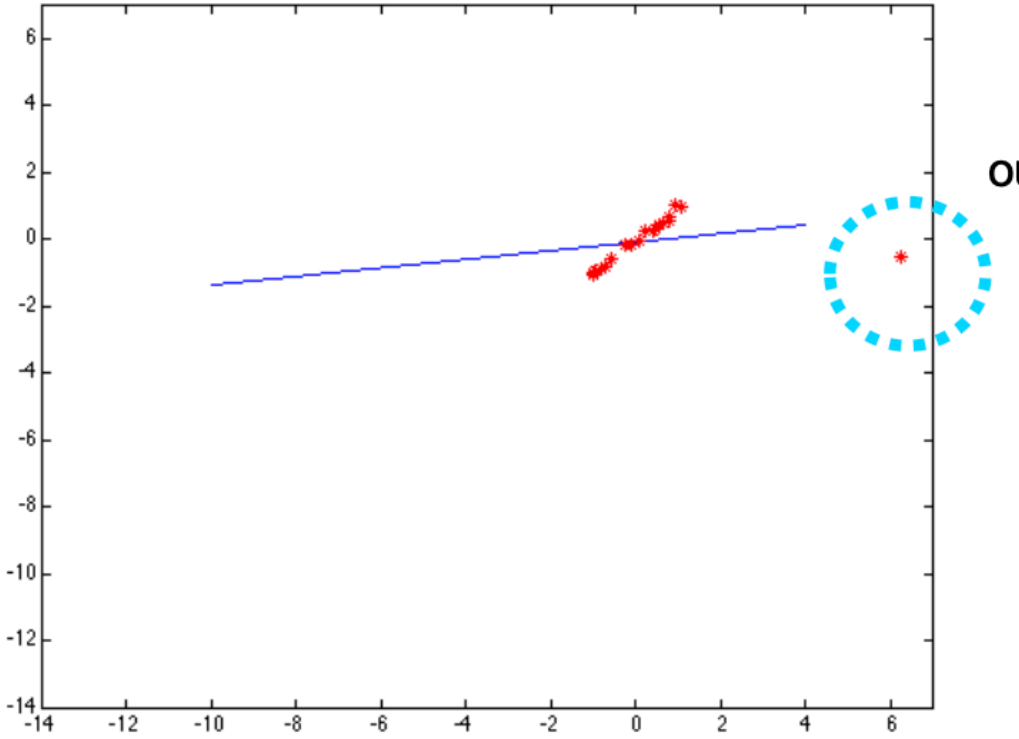
$$A = UDV^T$$

$$\mathrm{h} = \text{last column of}\ \ \mathrm{V}$$

Robustness



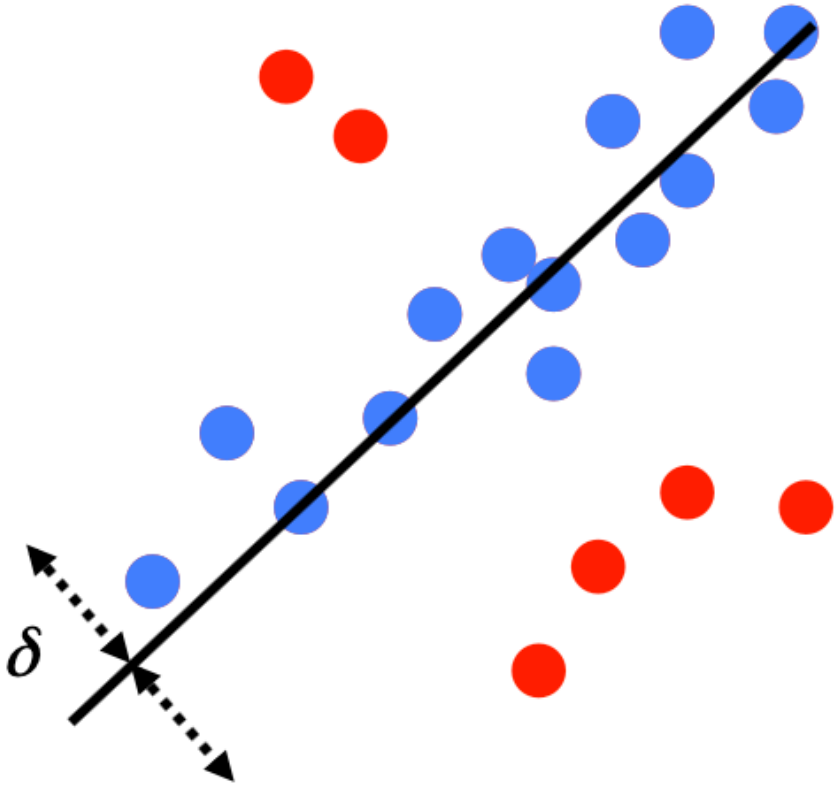
Robust to small noises.



Sensitive to outliers.

RANSAC: RANdom SAmple Consensus

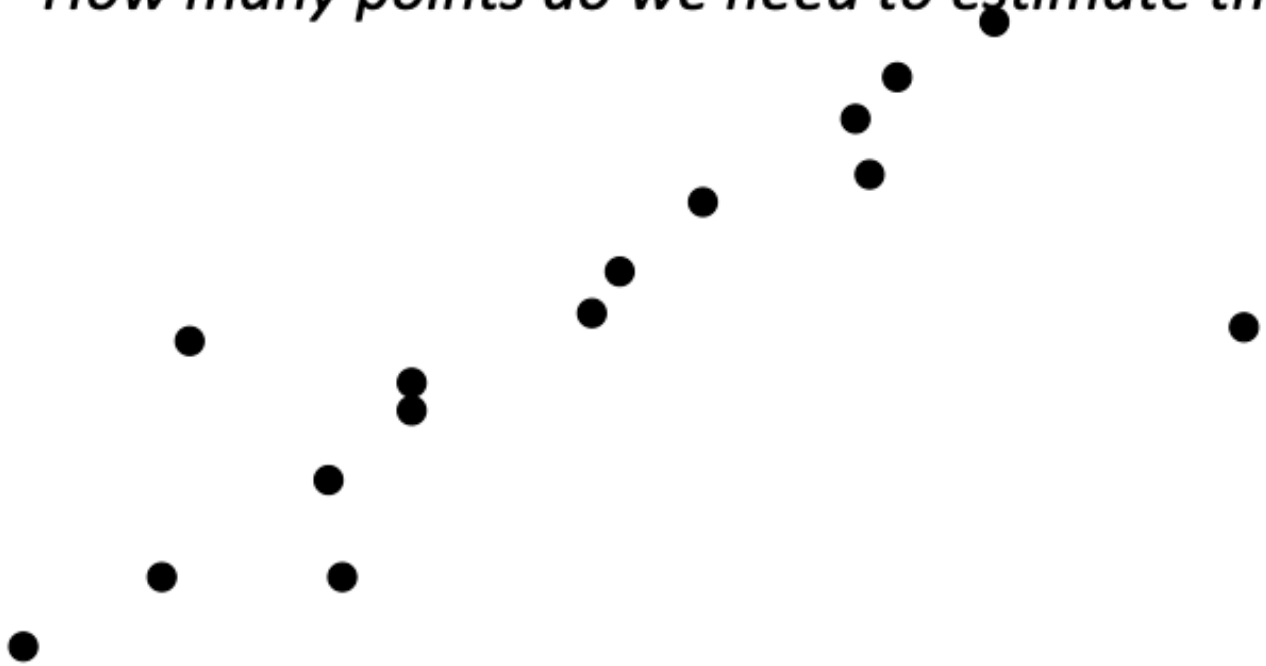
- Idea: we need to find a line that has the largest supporters (or inliers)



Fischler & Bolles in ‘81.

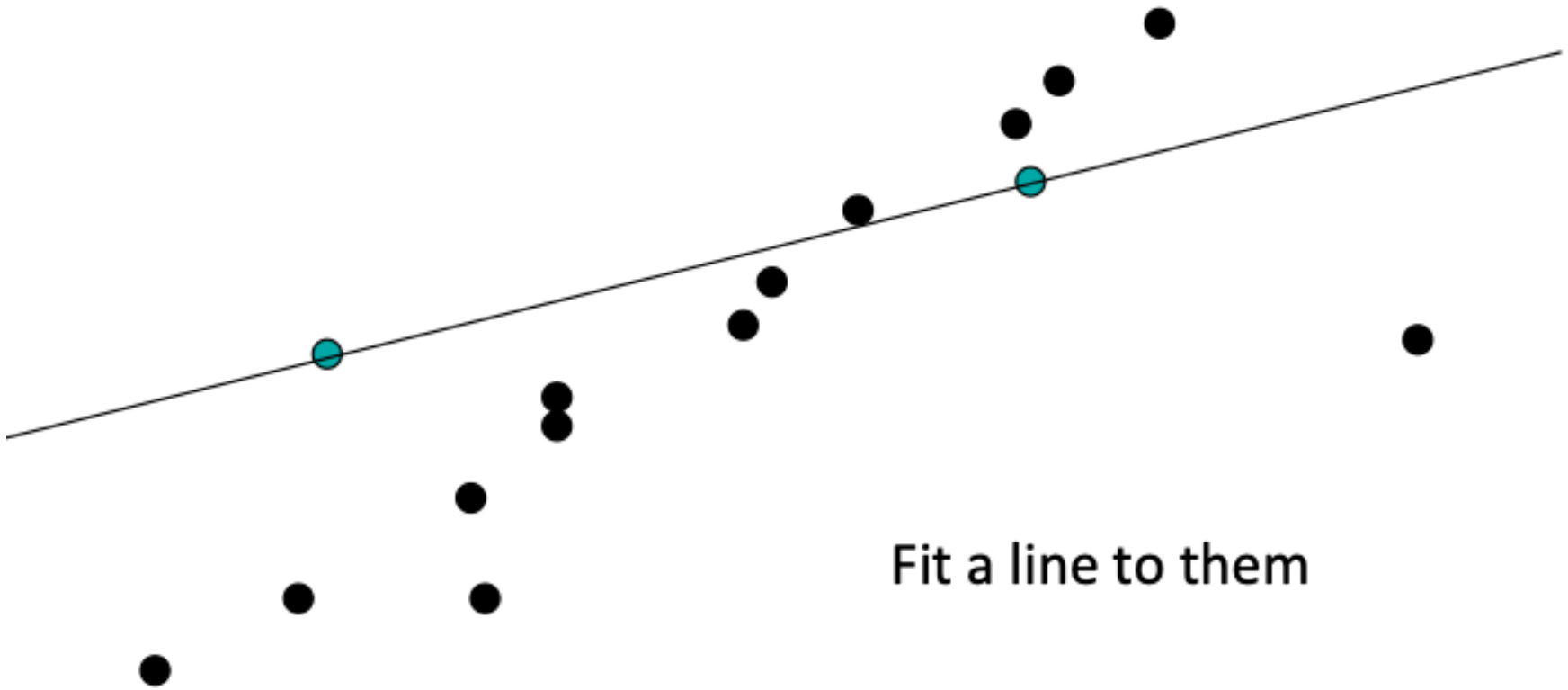
RANSAC Line Fitting

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



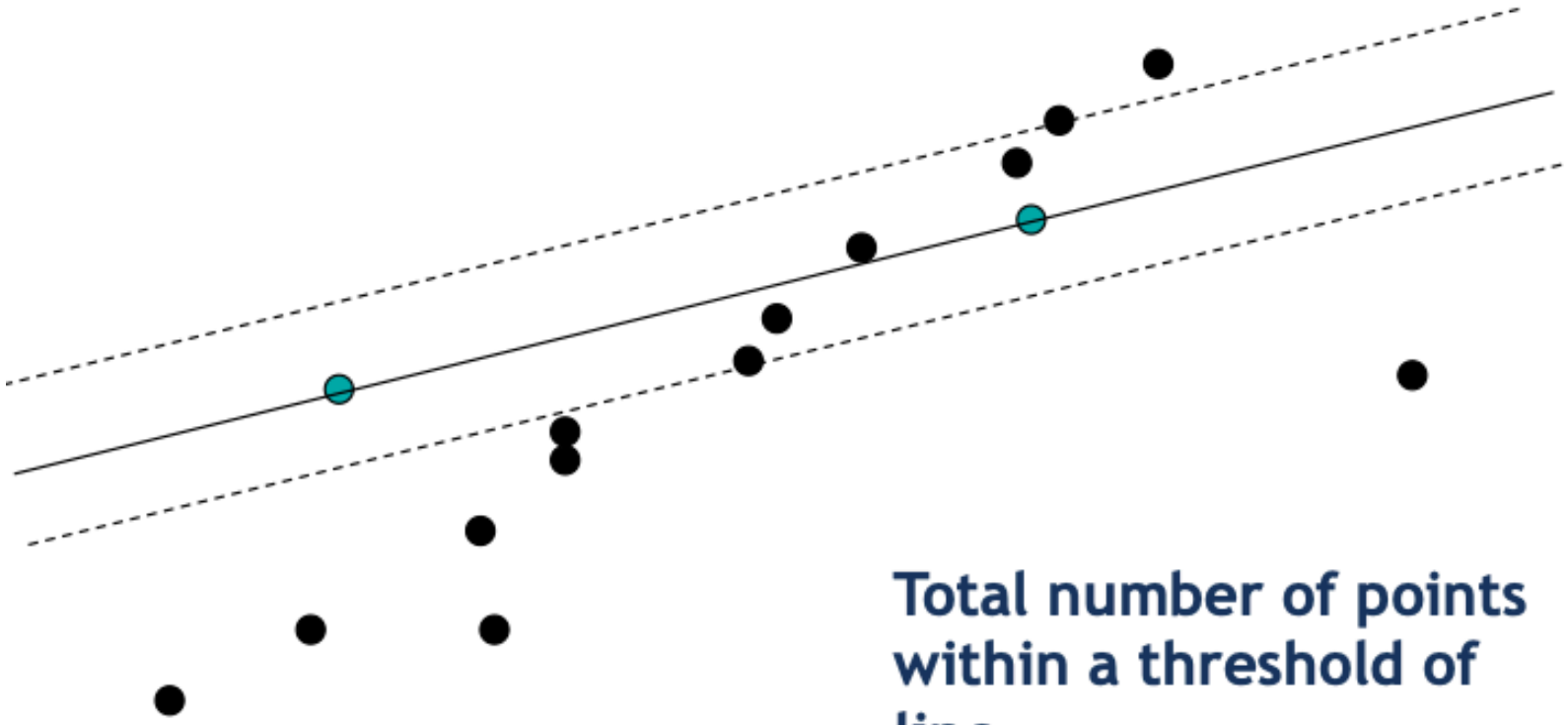
RANSAC Line Fitting

- Task: Estimate the best line



RANSAC Line Fitting

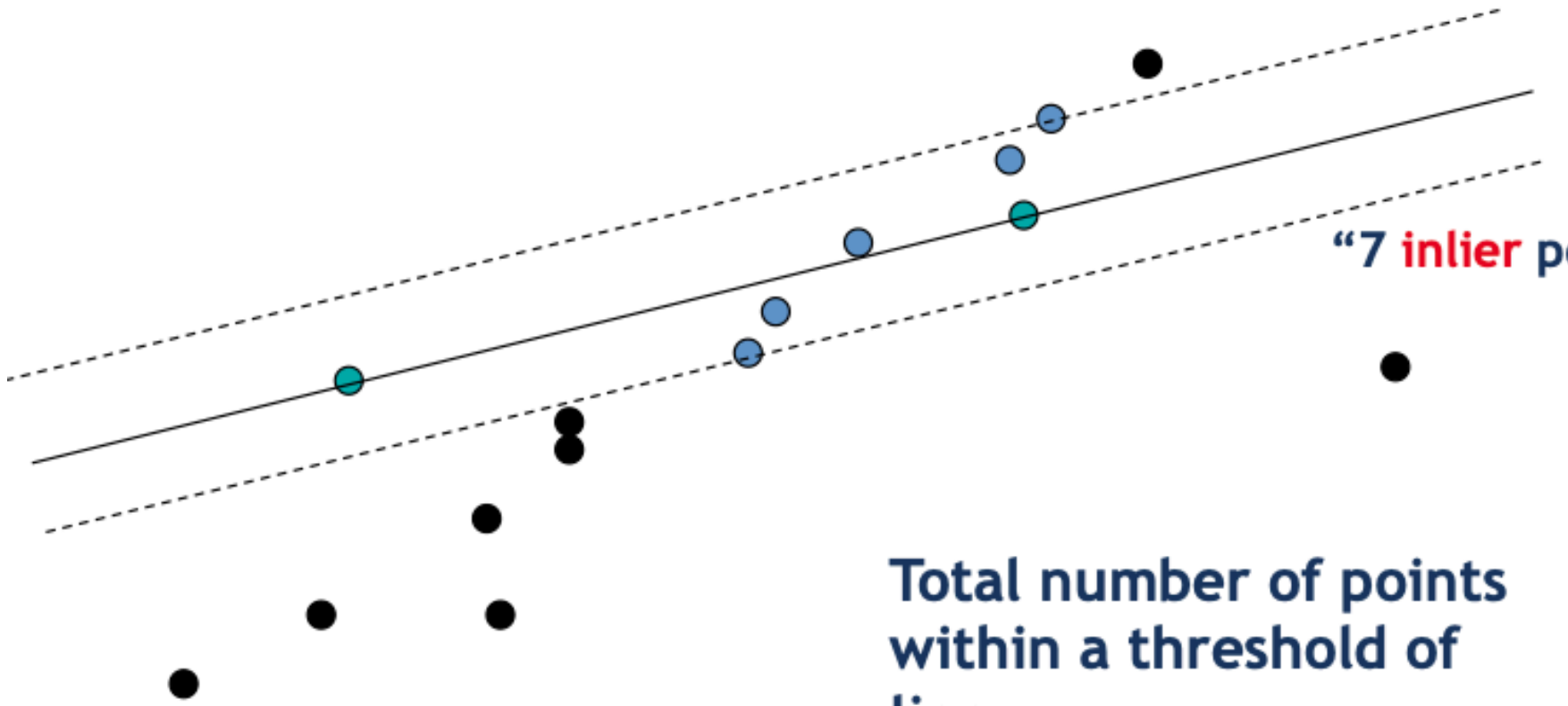
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC Line Fitting

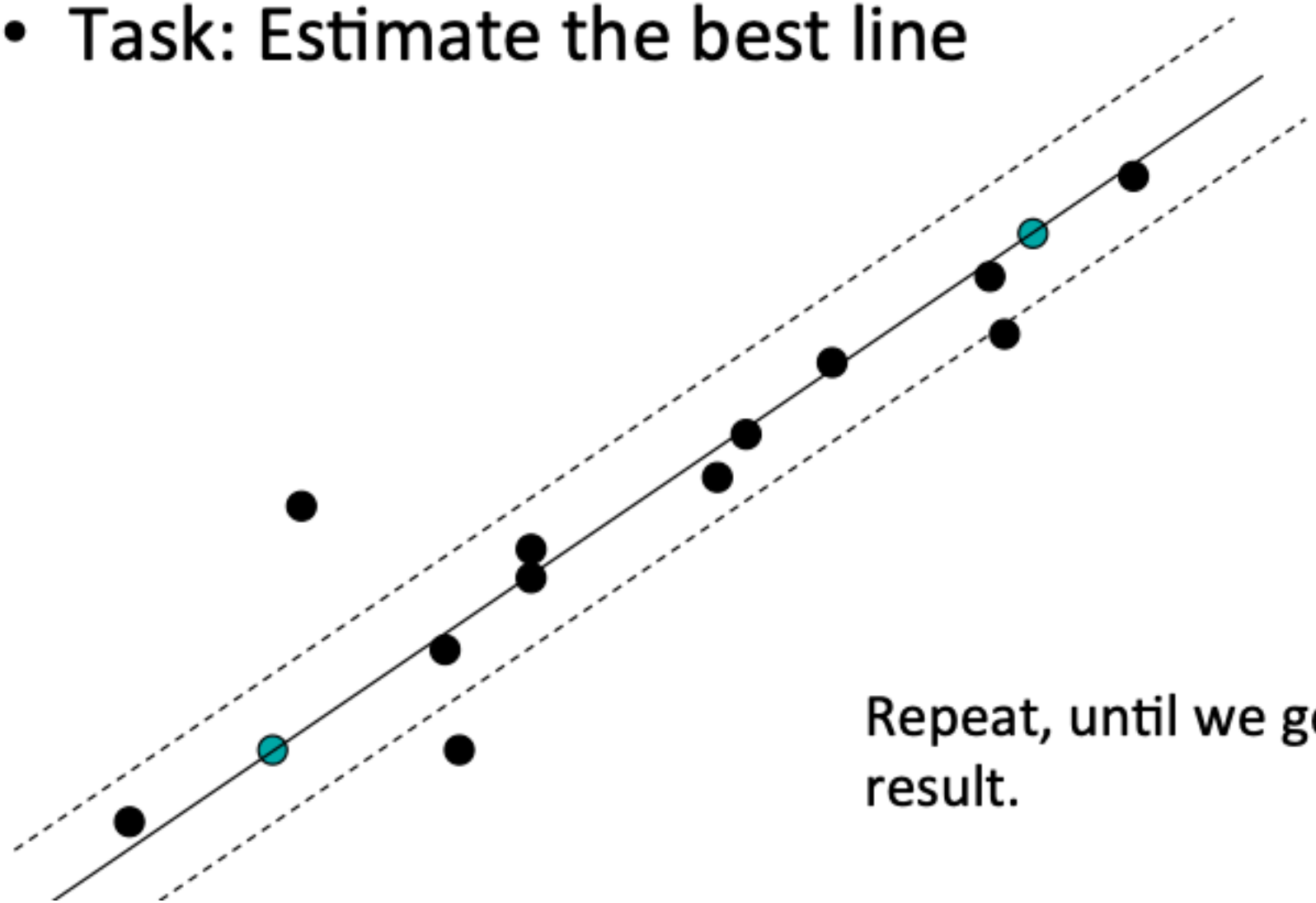
- Task: Estimate the best line



Slide credit: Jinxiang Chai

RANSAC Line Fitting

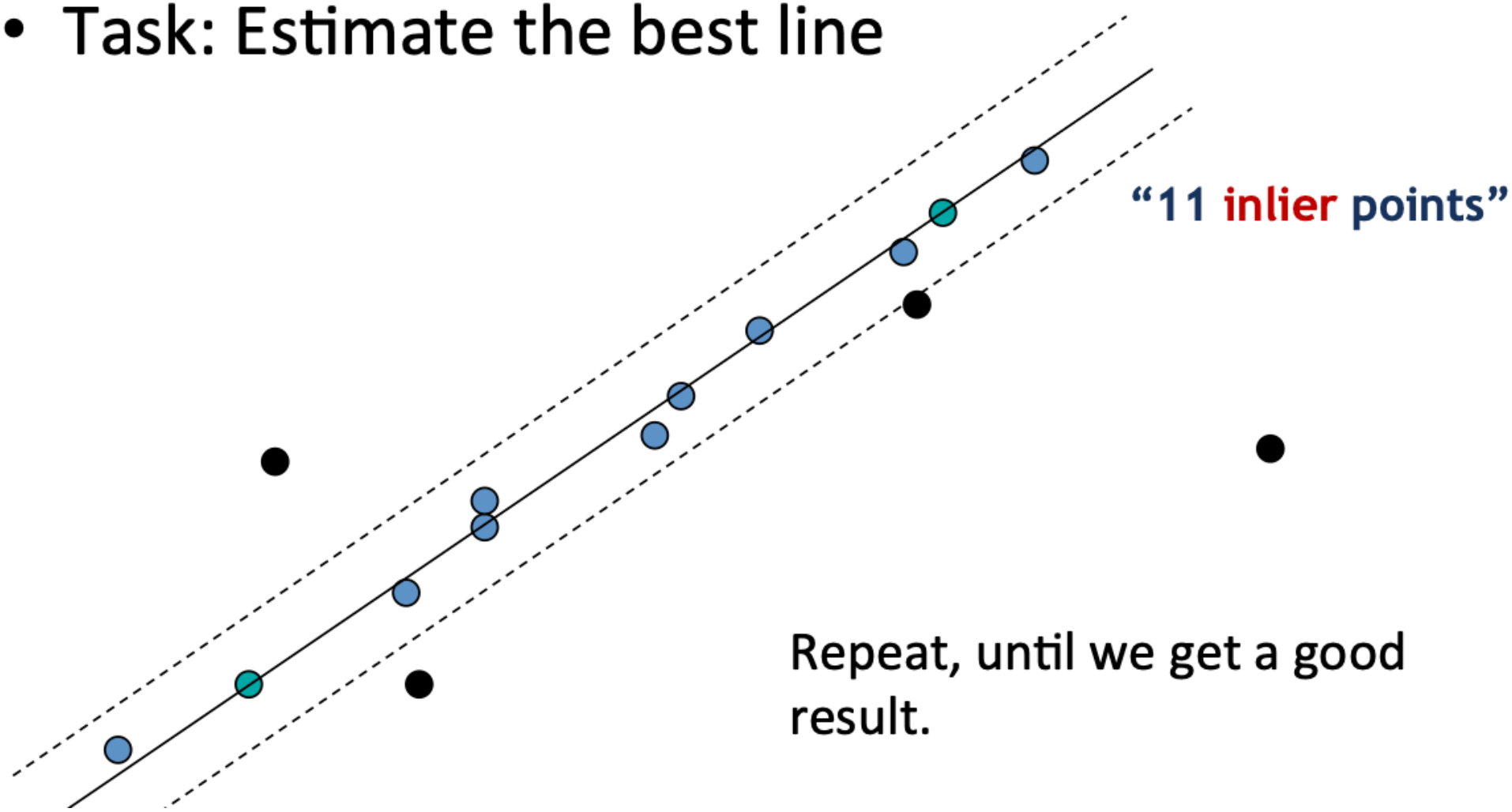
- Task: Estimate the best line



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?

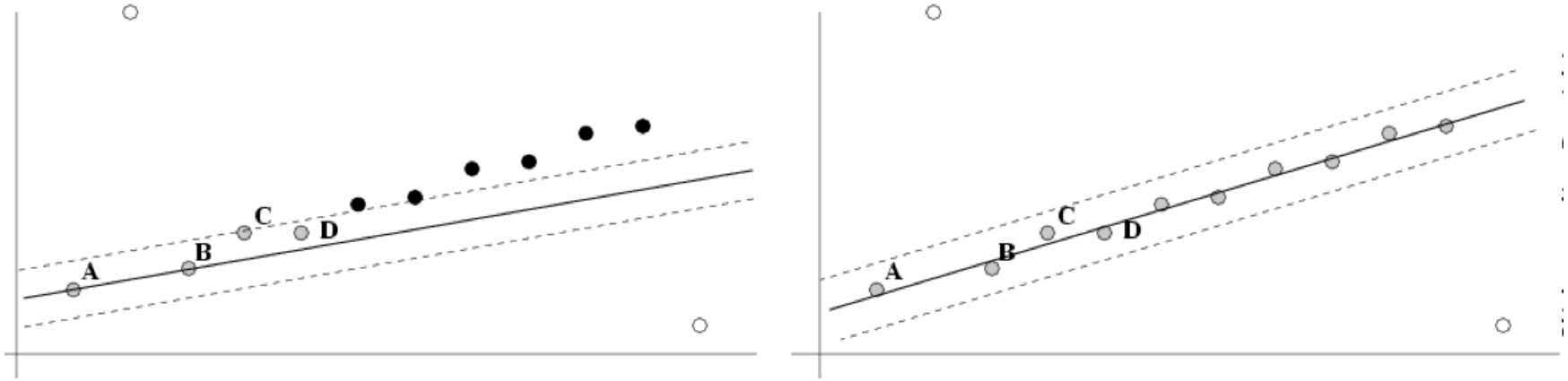
- How many samples are needed?
  - Suppose w is fraction of inliers (points from line).
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen.
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$

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

Slide credit: David Lowe

After RANSAC

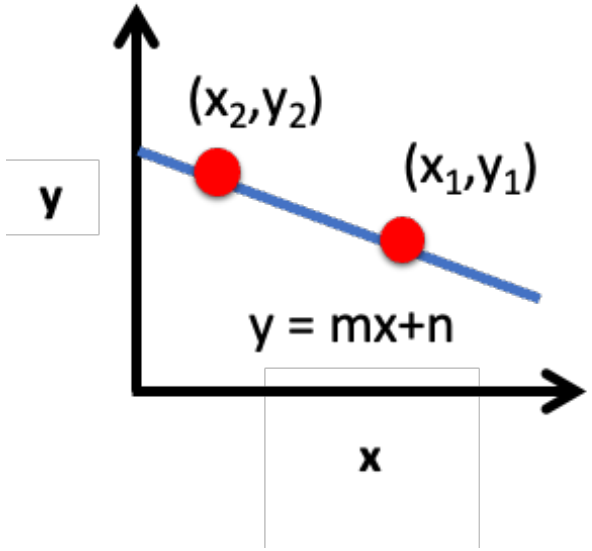
- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



RANSAC: Pro and Con

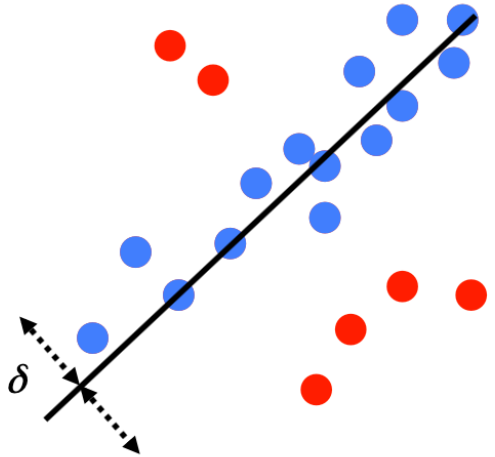
- **Pros:**
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons:**
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective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

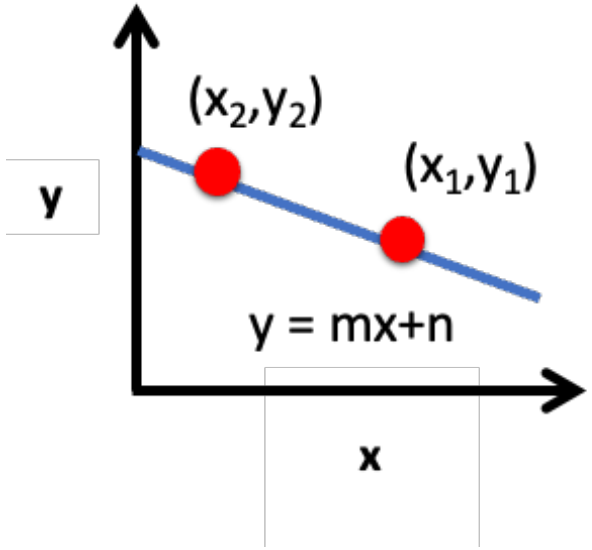
RANSAC



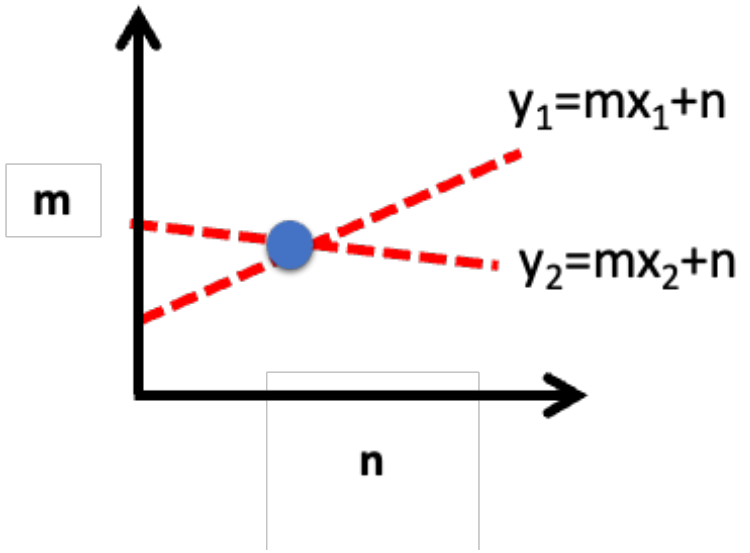
Define a inlier threshold distance in the vector space, each point votes for the best hypothesis.

Hough Transform

Original space



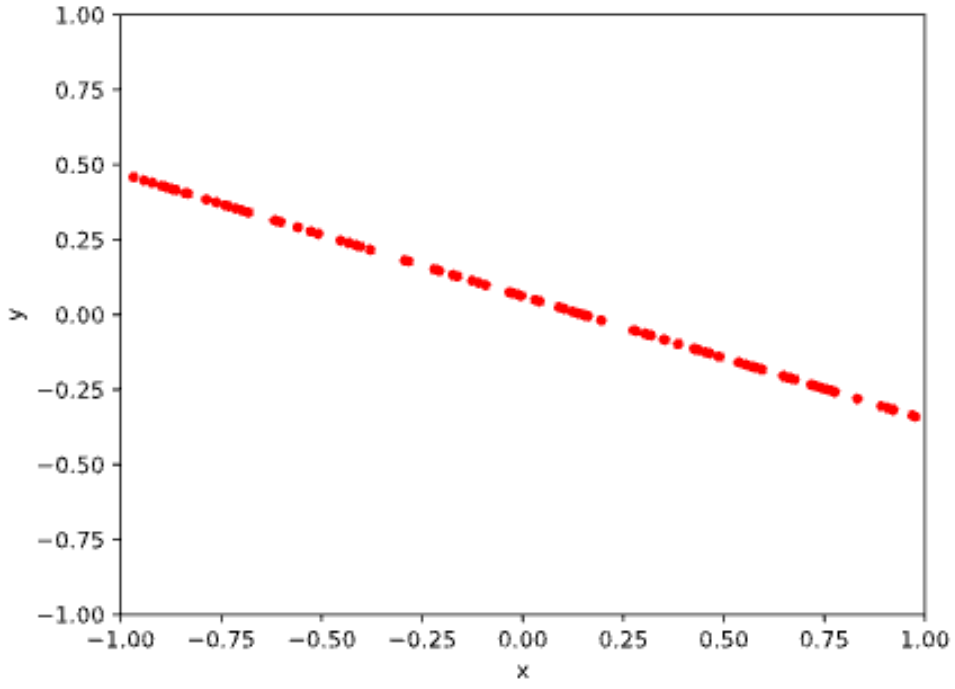
Hough space



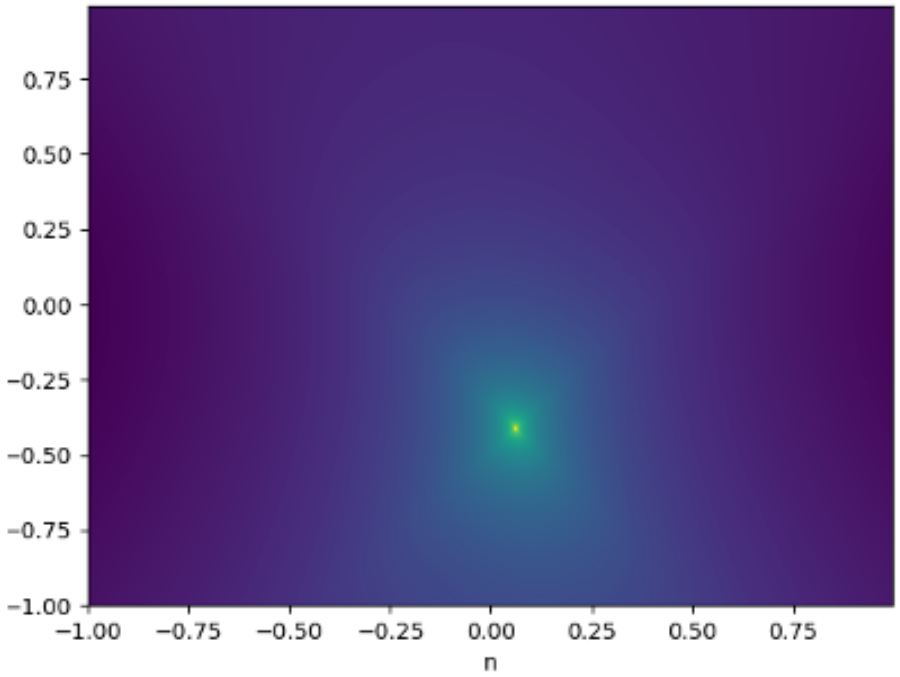
Given points in the vector space, find (m,n) in the parameter space

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

Hough Transform w/o Noise



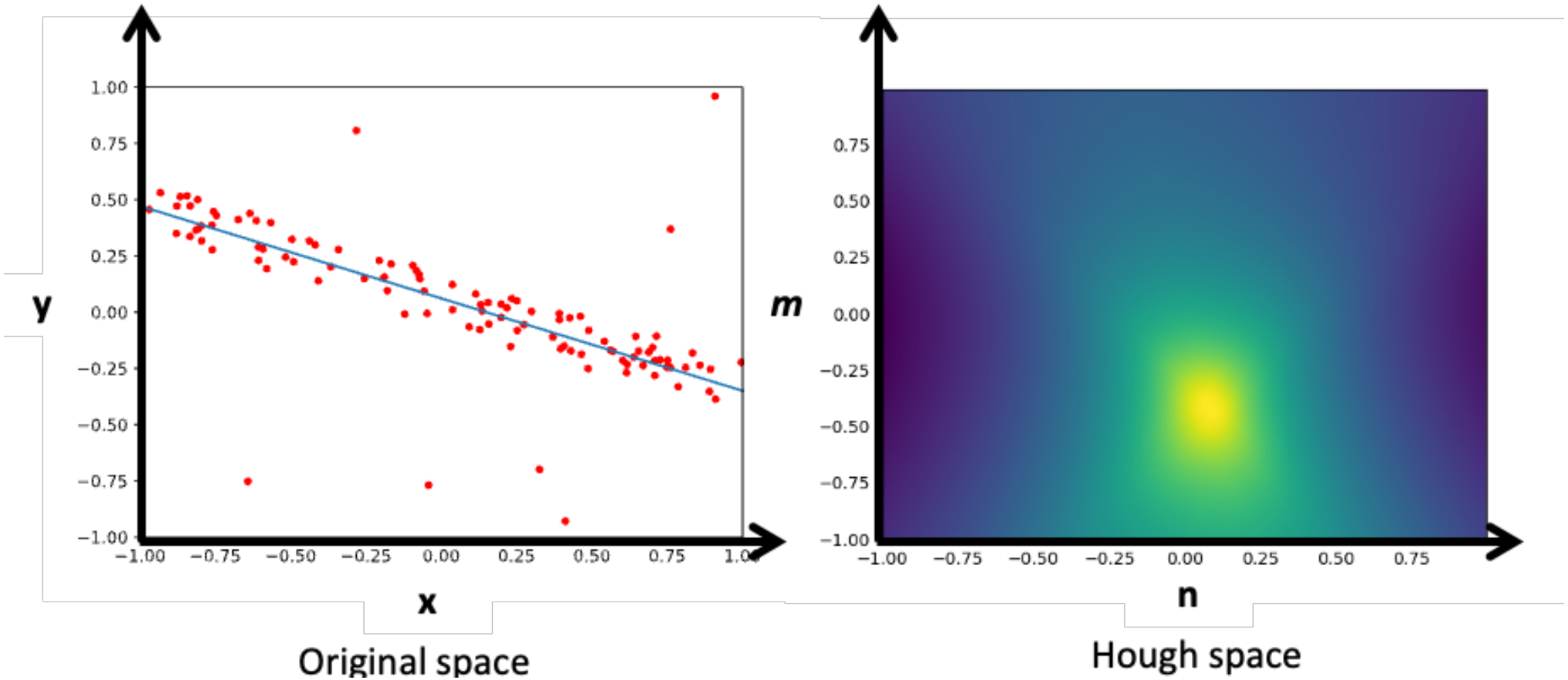
Original space



Hough space

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

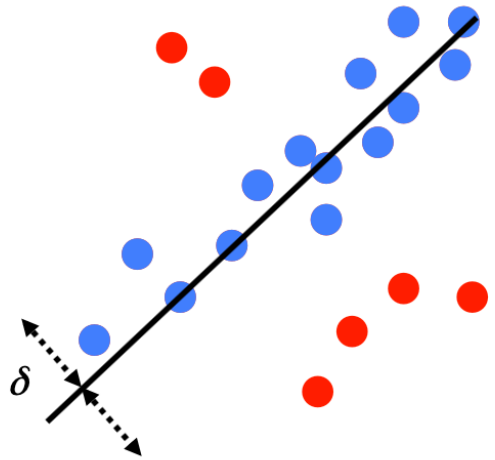
Hough Transform w/ Noise and Outliers



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

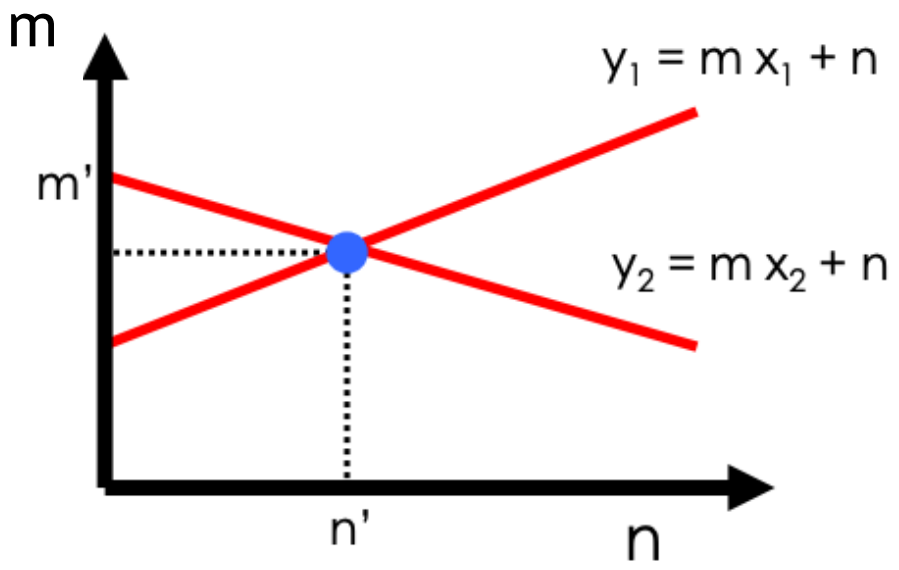
From the Perspective of Voting

RANSAC



Voting in the original space

Hough transform



Voting in the parameter space

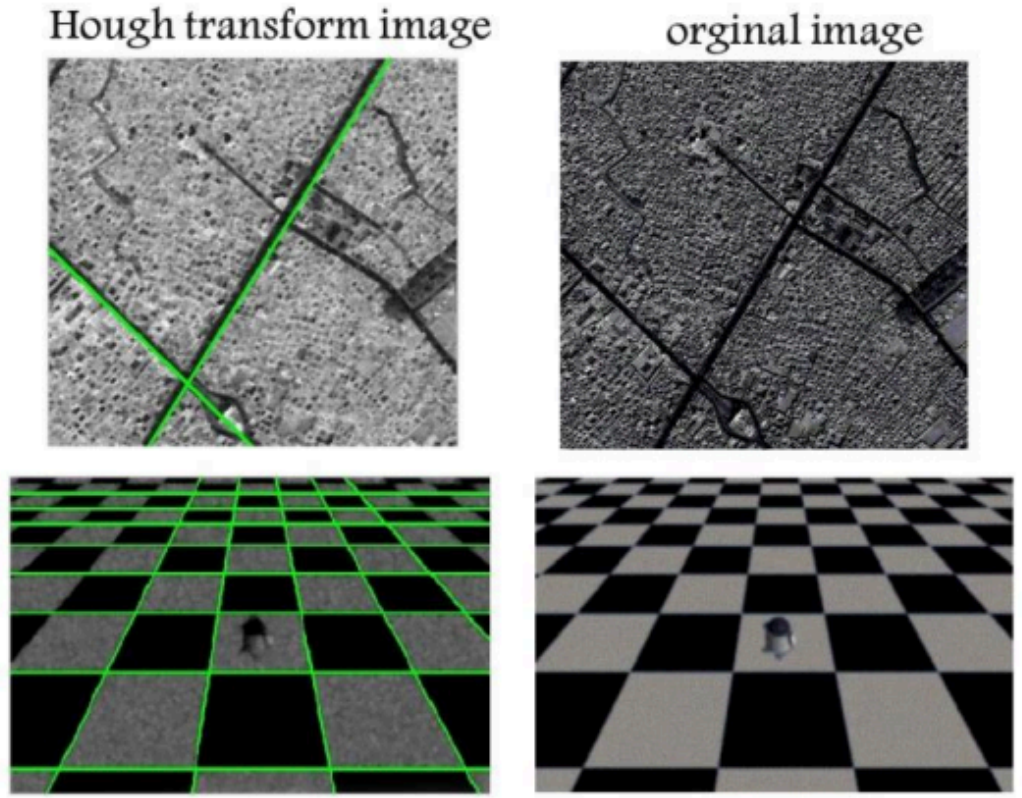
Robust Fitting: RANSAC vs. Hough Transform

RANSAC

- Single mode: robust for outliers

Hough Transform

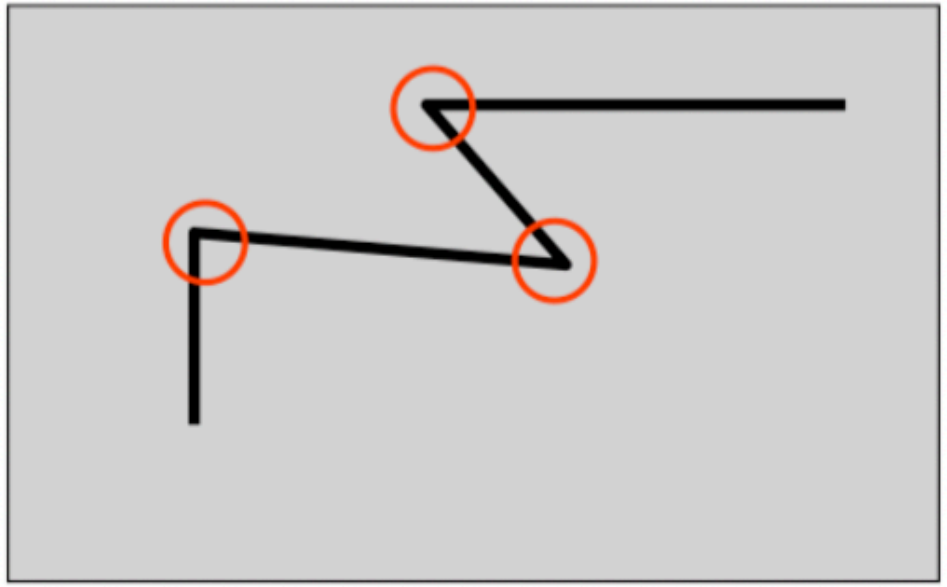
- Less robust compared to RANSAC (spurious peak)
- Can handle multiple modes well



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.

Corner Detection & Feature Descriptors

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

Step-by-Step Harris Detector

- Input: two images

Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Compute corner response θ

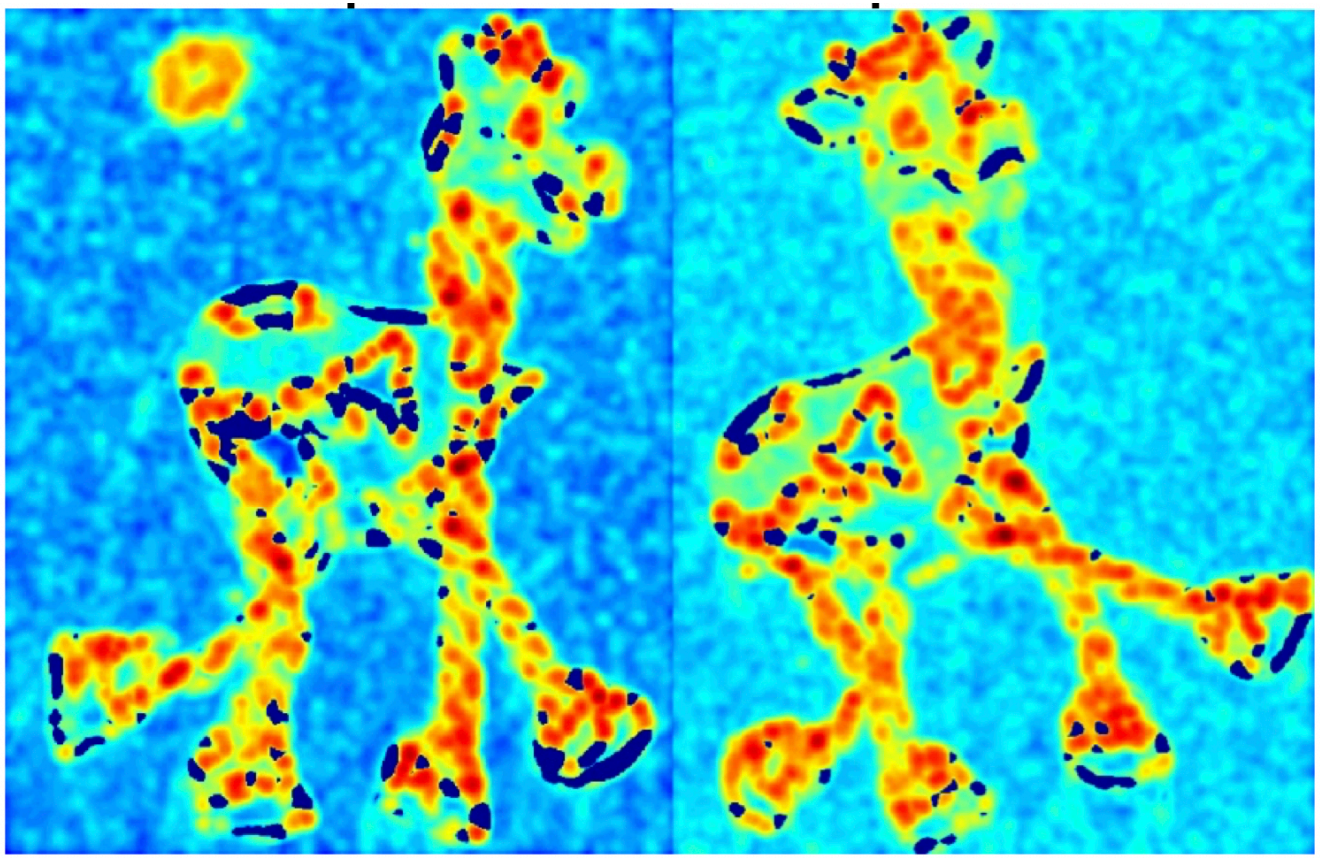


Image borrowed from Stanford CS131

Step-by-Step Harris Detector

- Thresholding and perform non-maximal suppression

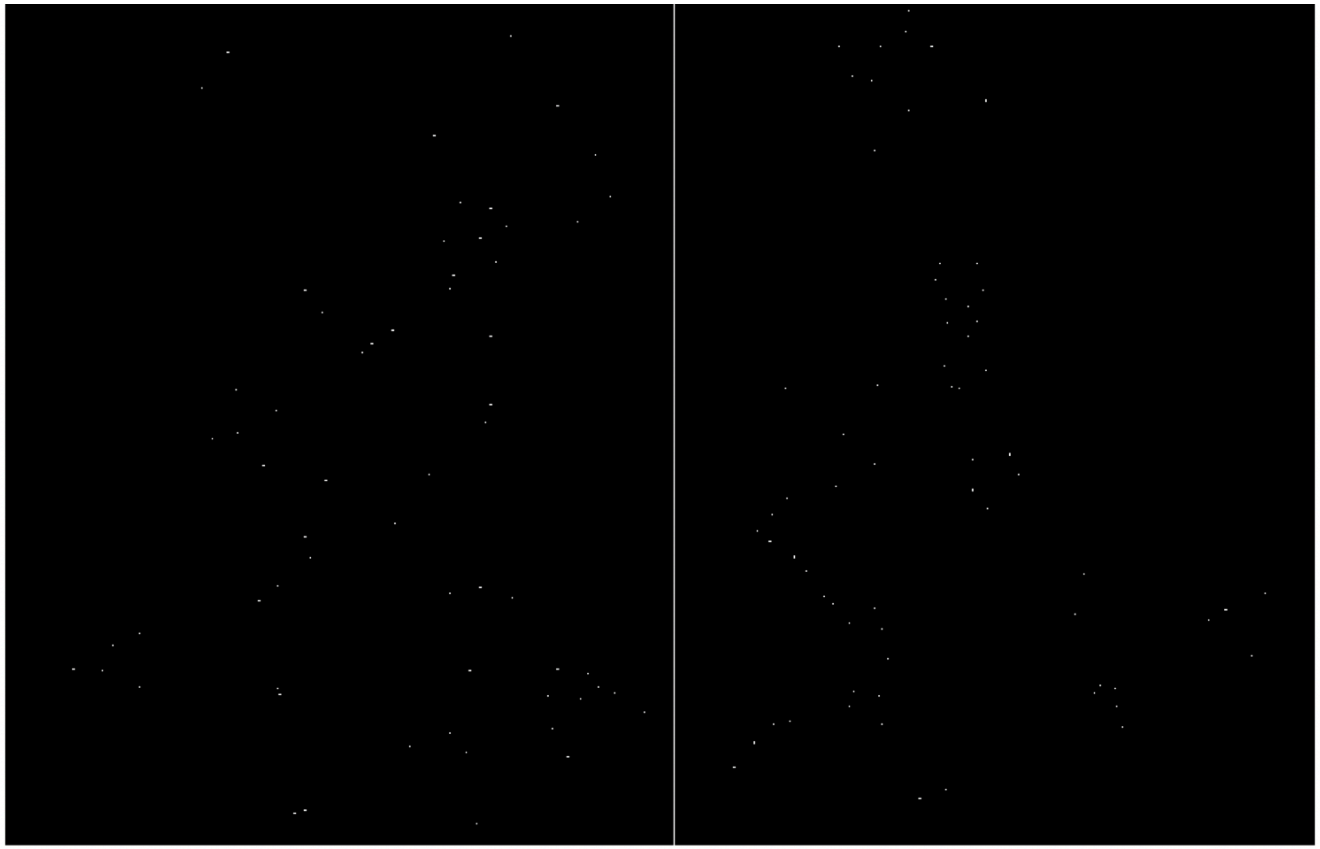


Image borrowed from Stanford CS131

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.

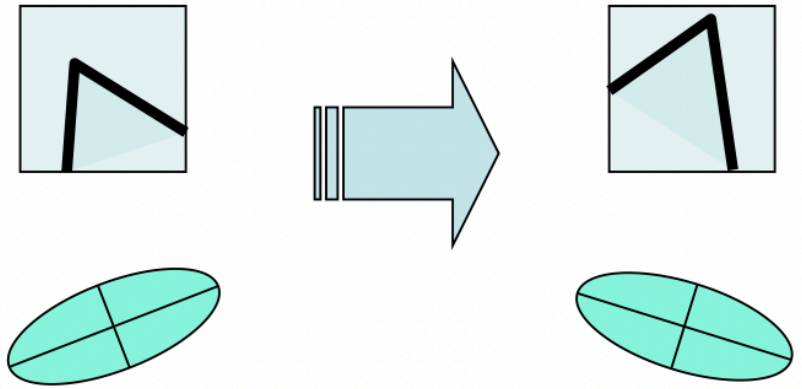


Image borrowed from Stanford CS131

Properties of Harris Detector

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

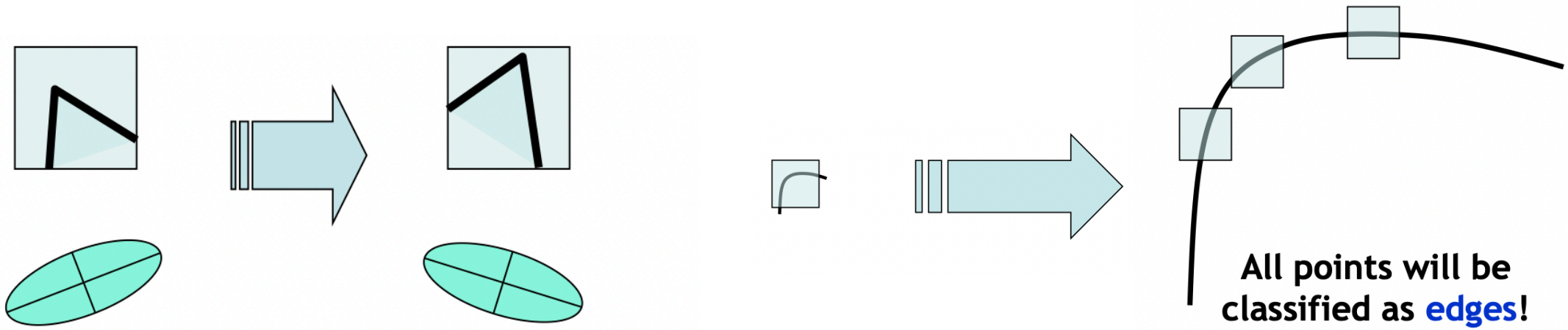
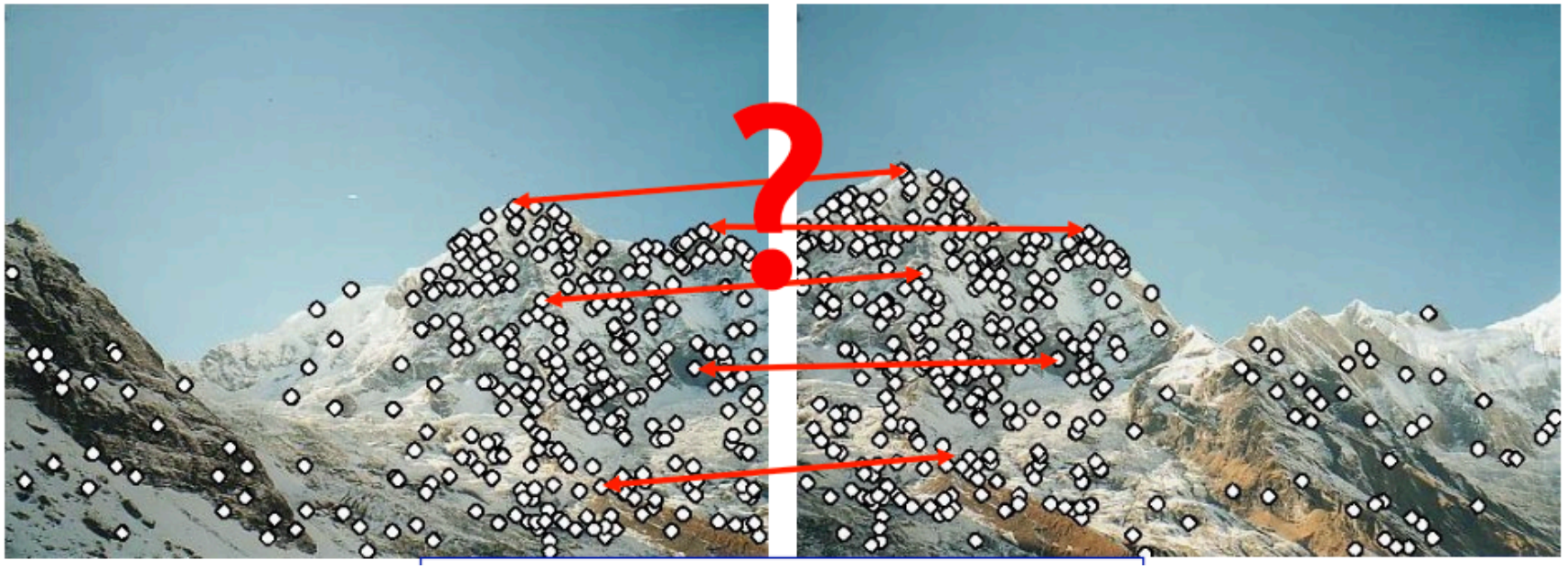


Image borrowed from Stanford CS131

Local Descriptors

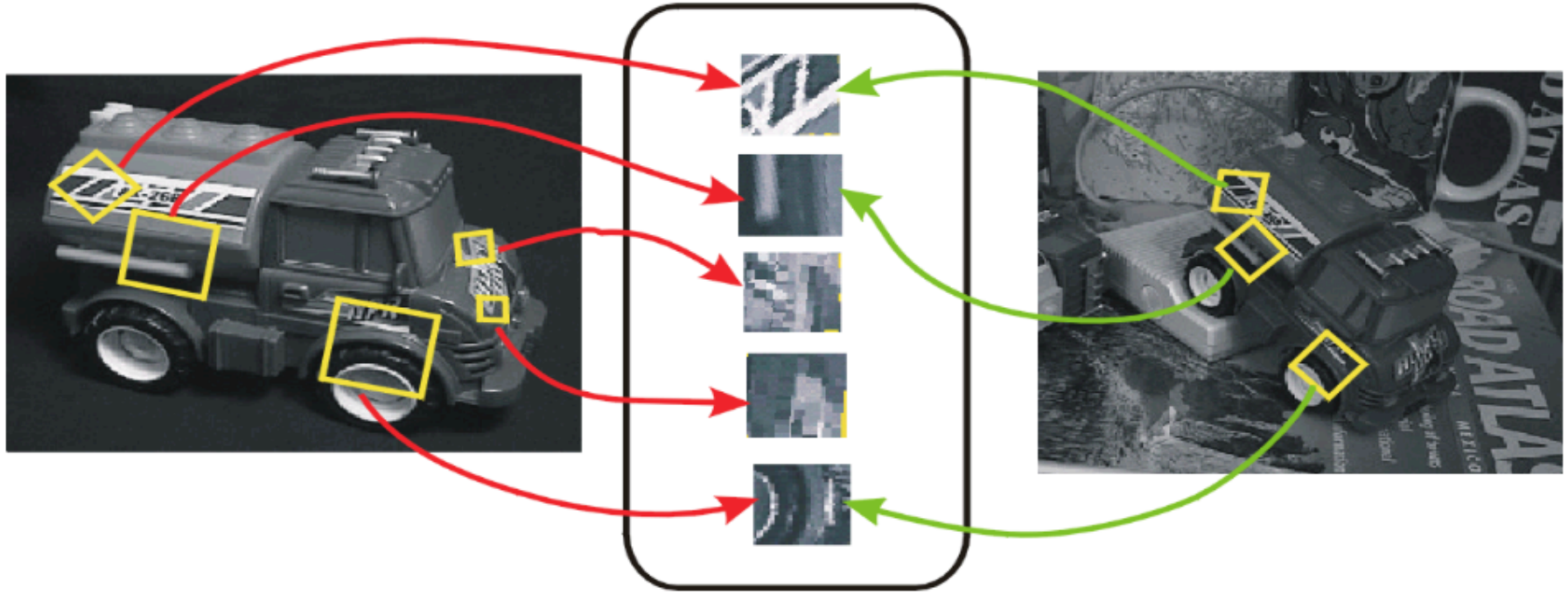
- We know how to detect points
- Next question:

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:

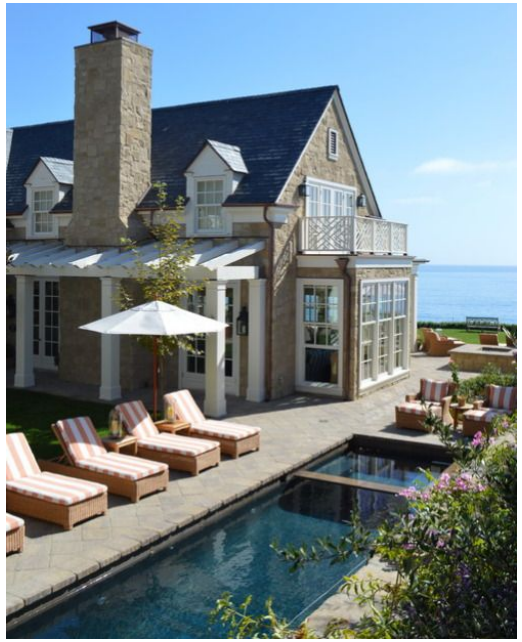
- Location
- Size
- Building time
- Current condition
- House style
- …

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

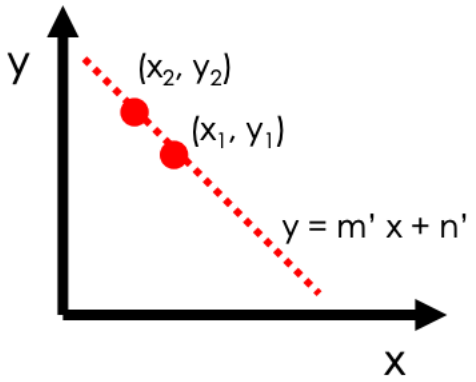
- Low-level vision
  - Image processing
  - Edge/corner detection
  - Feature extraction
- Mid-level Vision
  - Grouping
  - Inferring scene geometry (3D reconstruction)
  - Inferring camera and object motion
- **High-level vision (where deep learning wins!)**
  - Object recognition
  - Scene understanding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

Machine Learning 101

From Line Fitting to Neural Network Training

- When we have some observations $\{(x, y)\}$, we want to find the relationship behind y and x.

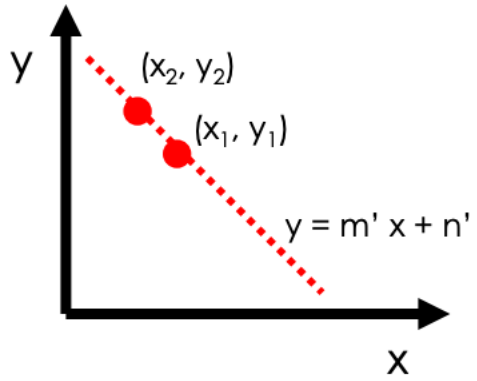
- Line fitting: we know the relationship is a line, so we use $y = mx + b$ to fit (m, n)



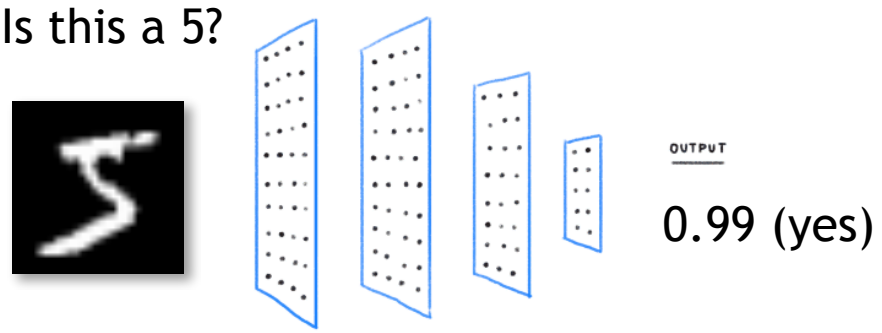
From Line Fitting to Neural Network Training

- When we have some observations $\{(x, y)\}$, we want to find the relationship behind y and x.

- Line fitting: we know the relationship is a line, so we use $y = mx + b$ to fit (m, n).



- Training neural network: similarly, we use a parametric model $y = h_\theta(x)$ to fit, however we usually have less understanding of h_θ.



Outline

- Set up the task
- Prepare the data —> Need a labeled dataset.
- Built a model —> construct your neural network
- Decide the fitting/training objective —> Loss function
- Perform fitting —> Training by running optimization
- Testing —> Evaluating on test data

Task: Binary Classification — Is This Digit a 5?

Data

Model

Output

$x =$ 

$h($  $,\theta)$

?

$y \in \{0,1\}$

Parameters or weights

Yes or No

MINIST Dataset of handwritten digits

Task: Binary Classification — Is This Digit a 5?

Data

Model

Output

$x =$ 

$h($  $,\theta)$

?

$y \in \{0,1\}$

Parameters or weights

Probability of being a 5

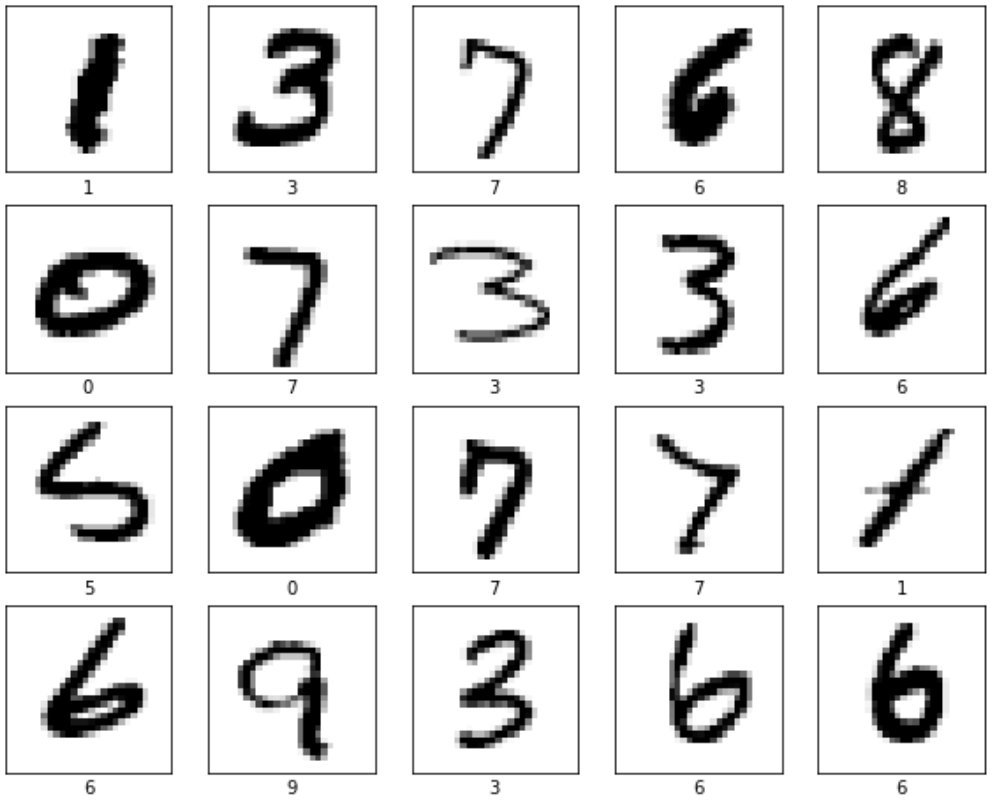
$p(y = 1 \mid x)$

MINIST Dataset of handwritten digits

Outline

- Set up the task
- Prepare the data —> Need a labeled dataset.
- Built a model —> construct your neural network
- Decide the fitting/training objective —> Loss function
- Perform fitting —> Training by running optimization
- Testing —> Evaluating on test data

Data



From MNIST dataset

- 70000 images in total
- Basic elements of One Data Point
  - One digit image $x^{(i)}$: 28×28 pixels
  - Paired with a label $y^{(i)} \in \{0,1\}$
- Training data X, labels Y

Outline

- Set up the task
- Prepare the data —> Need a labeled dataset.
- Built a model —> construct your neural network
- Decide the fitting/training objective —> Loss function
- Perform fitting —> Training by running optimization
- Testing —> Evaluating on test data

Model: Logistic Regression

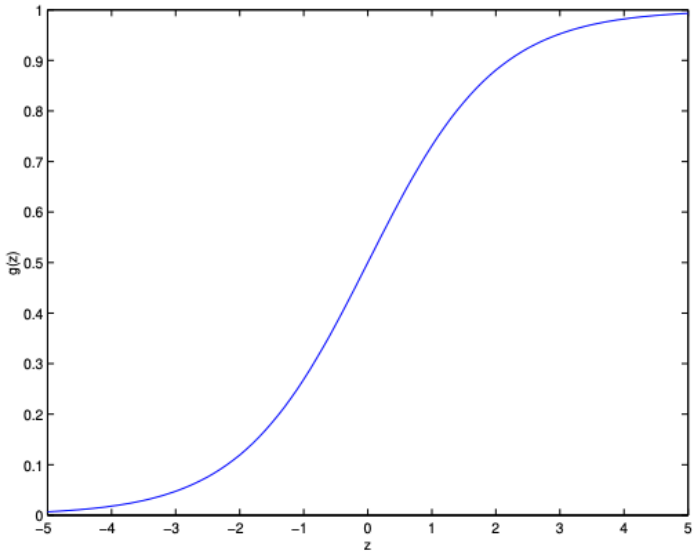
- Image 28×28: flatten to a one-dimensional vector $x \in \mathbb{R}^{784}$

- Classification function:
  - Let's assume a linear function $h(x) = g(\theta^T x)$

- Here we need a function $g(z)$ to convert $z = w^T x \in (-\infty, \infty)$ to $(0,1)$

Sigmoid Function

$$g(z) = \frac{1}{1 + e^{-z}}$$



Sigmoid function

Final model: $f(x) = g(\theta^T x)$

Outline

- Set up the task
- Prepare the data —> Need a labeled dataset.
- Built a model —> construct your neural network
- Decide the fitting/training objective —> Loss function
- Perform fitting —> Training by running optimization
- Testing —> Evaluating on test data

Maximum Likelihood Estimation

- In statistics, **maximum likelihood estimation (MLE)** is a method of estimating the parameters of an assumed probability distribution, given some observed data.

- This is achieved by maximizing **a likelihood function** so that, under the assumed statistical model, **the observed data is most probable.**

- The point in the parameter space (**network weight** θ) that maximizes the likelihood function is called the maximum likelihood estimate.

Task: Binary Classification — Is This Digit a 5?

Data

Model

Output

$x =$ 

$h($  $,\theta)$

?

$y \in \{0,1\}$

Parameters or weights

$p(y = 1 \mid x)$

MINIST Dataset of handwritten digits

Probability of One Data Point

- Classification function: $h_\theta(x)$

$$p(y = 1 \mid x; \theta) = h_\theta(x)$$

$$p(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

- Writing more compactly to handle both $y = 0$ and $y = 1$,

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

Probability of All Data Points

- Assume all the data points are independent, then

$$p(Y|X;\theta) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{n} (h_\theta(x^{(i)}))^{y^{(i)}}(1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

$$\log p(Y|X;\theta) = \sum_{i=1}^{n} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))$$

Loss: Negative Log-likelihood

- Loss: the thing you want to minimize

- Negative log-likelihood (NLL) loss

$$\mathscr{L}(\theta) = -\log p(Y|X;\theta)$$

$$= -\sum_{i=1}^{n} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

Outline

- Set up the task
- Prepare the data —> Need a labeled dataset.
- Built a model —> construct your neural network
- Decide the fitting/training objective —> Loss function
- Perform fitting —> Training by running optimization
- Testing —> Evaluating on test data

Optimization

$\mathcal{L}(\theta)$

θ

How would you go to the very bottom?

Next Week:
Lecture 4, Deep Learning II