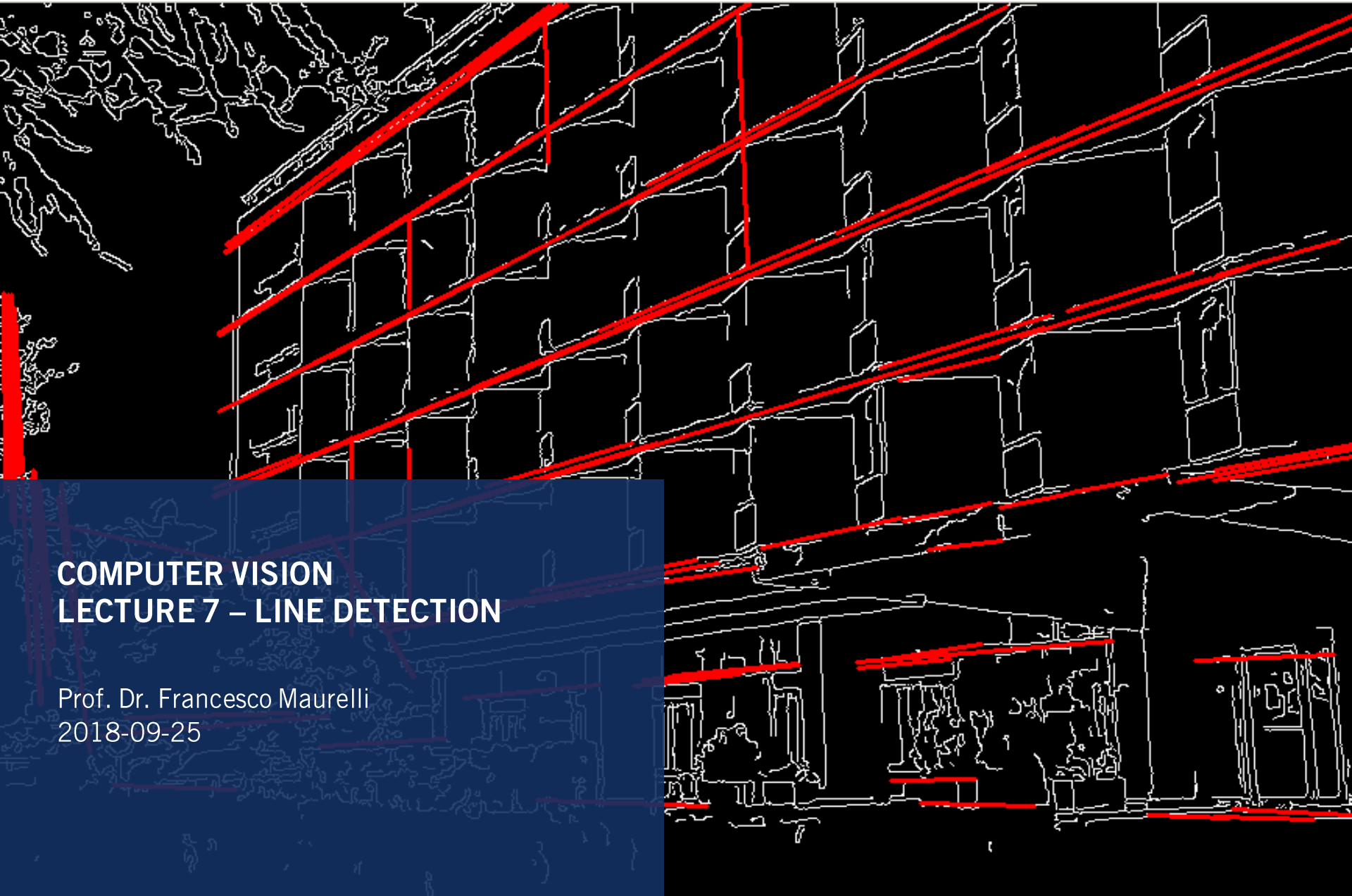
JACOBS UNIVERSITY



COMPUTER VISION
LECTURE 7 – LINE DETECTION

Prof. Dr. Francesco Maurelli

2018-09-25

WHAT WE WILL LEARN TODAY

1. Recap Edge detection
2. Hough Transform
3. RANSAC
4. Intro to Features

EDGE DETECTION

JACOBS UNIVERSITY

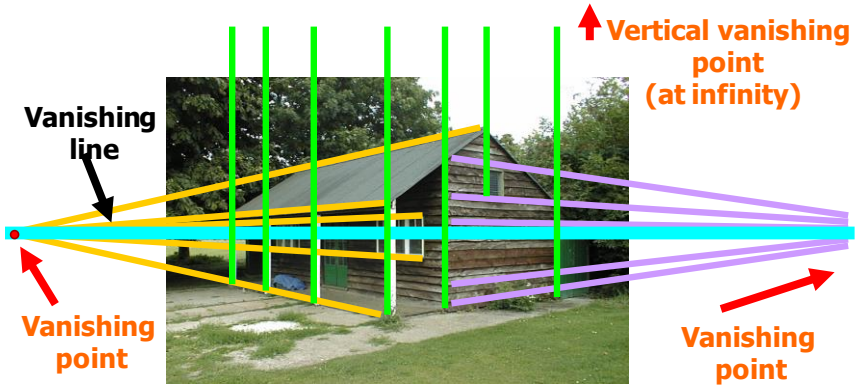
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

WHY DO WE CARE ABOUT EDGES?

1. Extract information, recognize objects

1. Recover geometry and viewpoint



ORIGINS OF EDGES

surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

JACOBS UNIVERSITY

Backward

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Forward

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Central

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

Backward filter: [0 1 -1]

$$f(x) - f(x-1) = f'(x)$$

Forward: [-1 1 0]

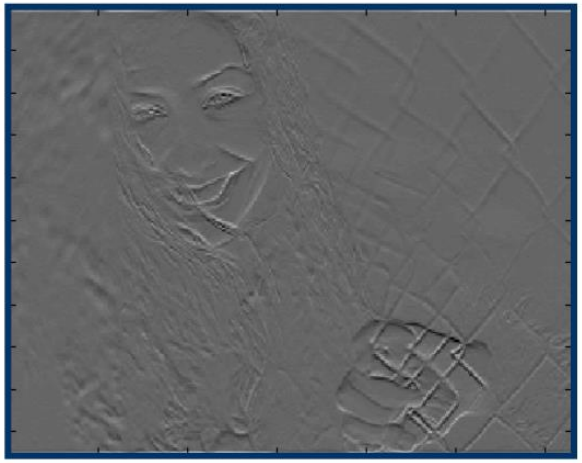
$$f(x) - f(x+1) = f'(x)$$

Central: [1 0 -1]

$$f(x+1) - f(x-1) = f'(x)$$

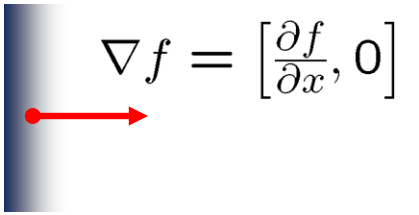
3X3 IMAGE GRADIENT FILTERS

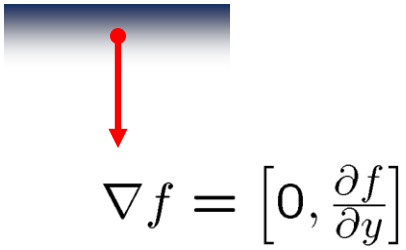
$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

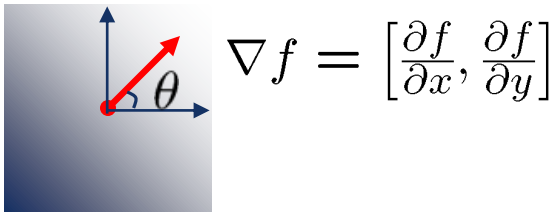


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

1.The gradient of an image:







The gradient vector points in the direction of most rapid increase in intensity

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

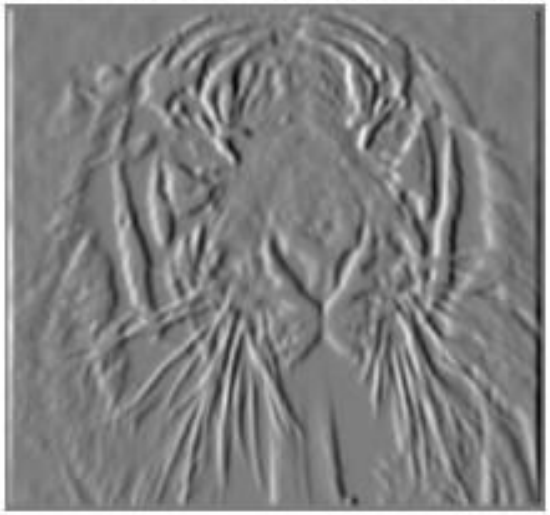
FINITE DIFFERENCES: EXAMPLE

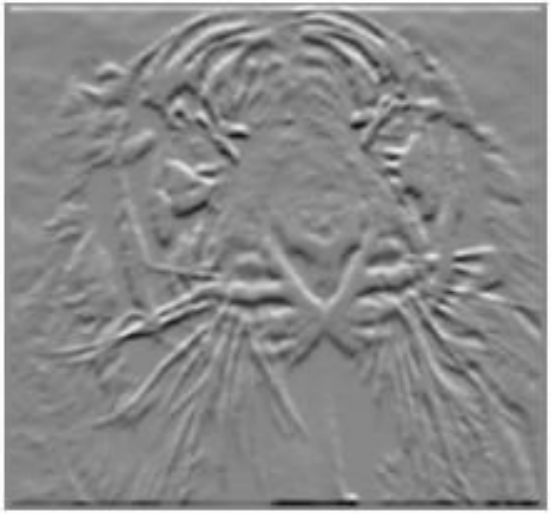
JACOBS UNIVERSITY

Original Image

Gradient magnitude

x-direction





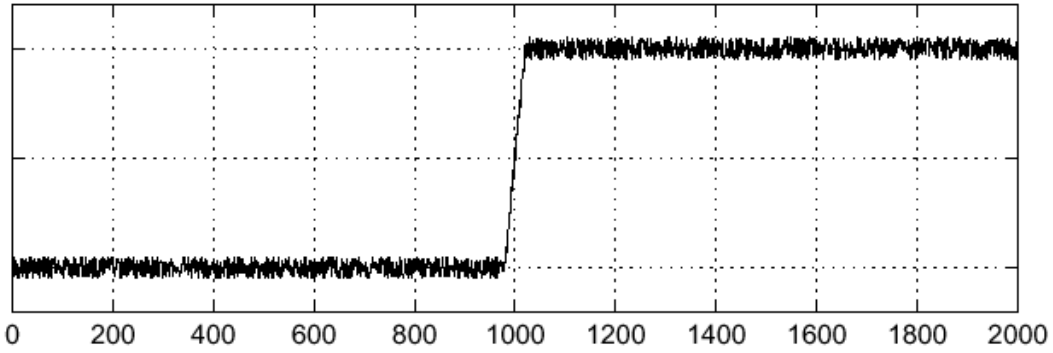
y-direction

JACOBS UNIVERSITY

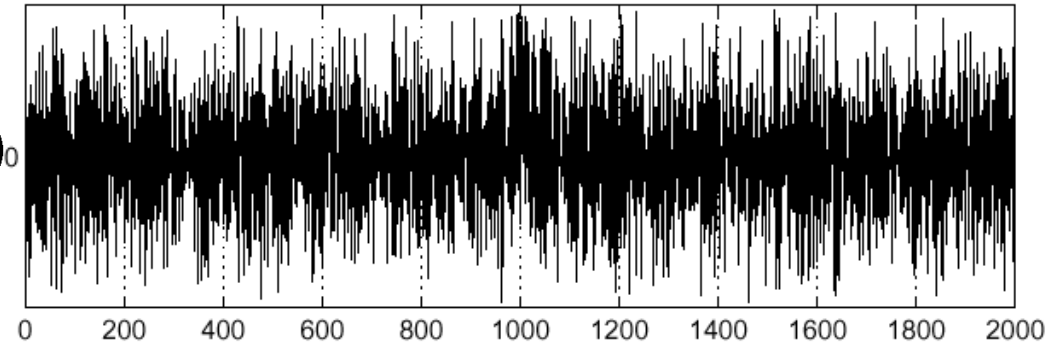
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$

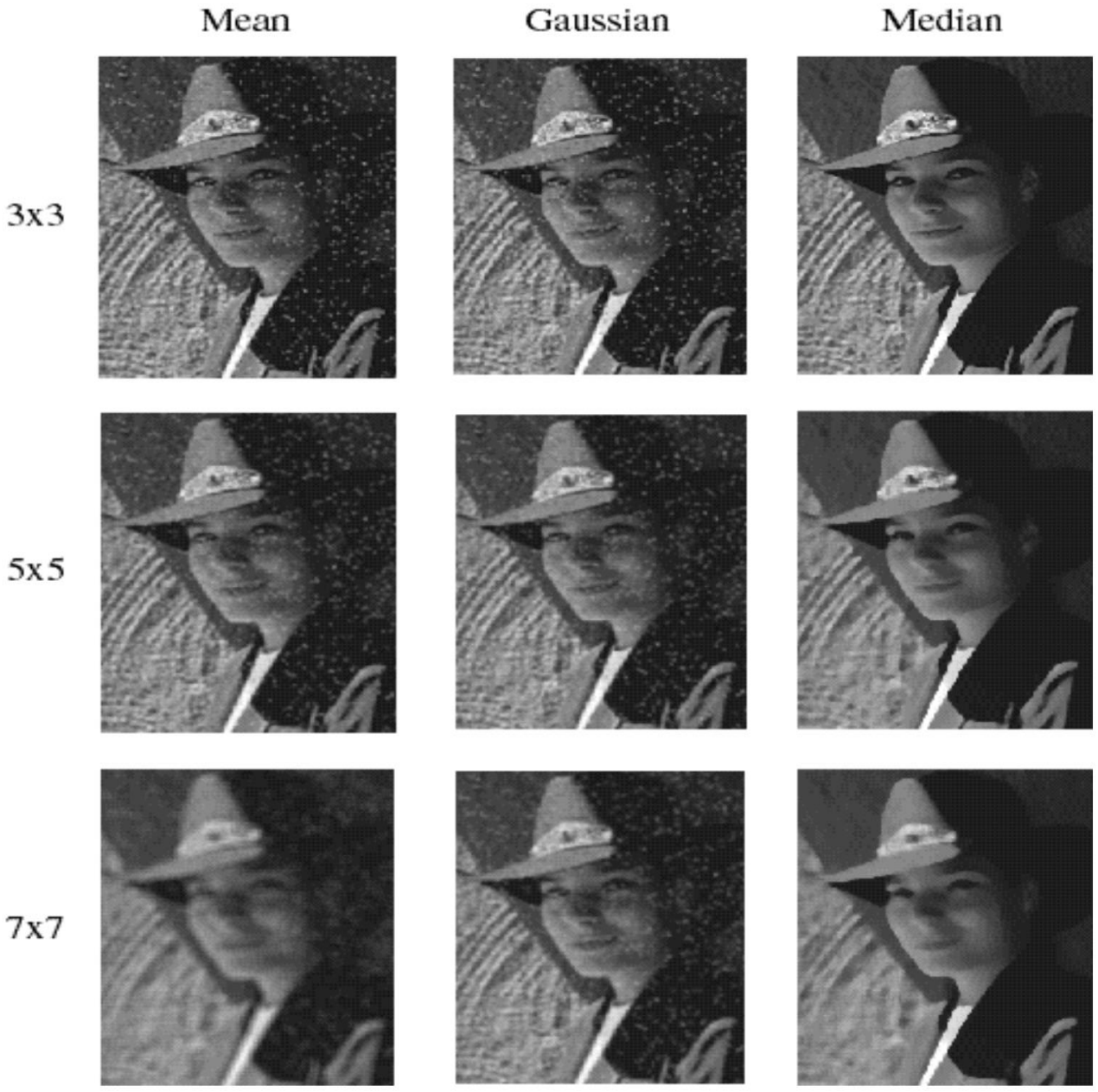


$\frac{d}{dx}f(x)$

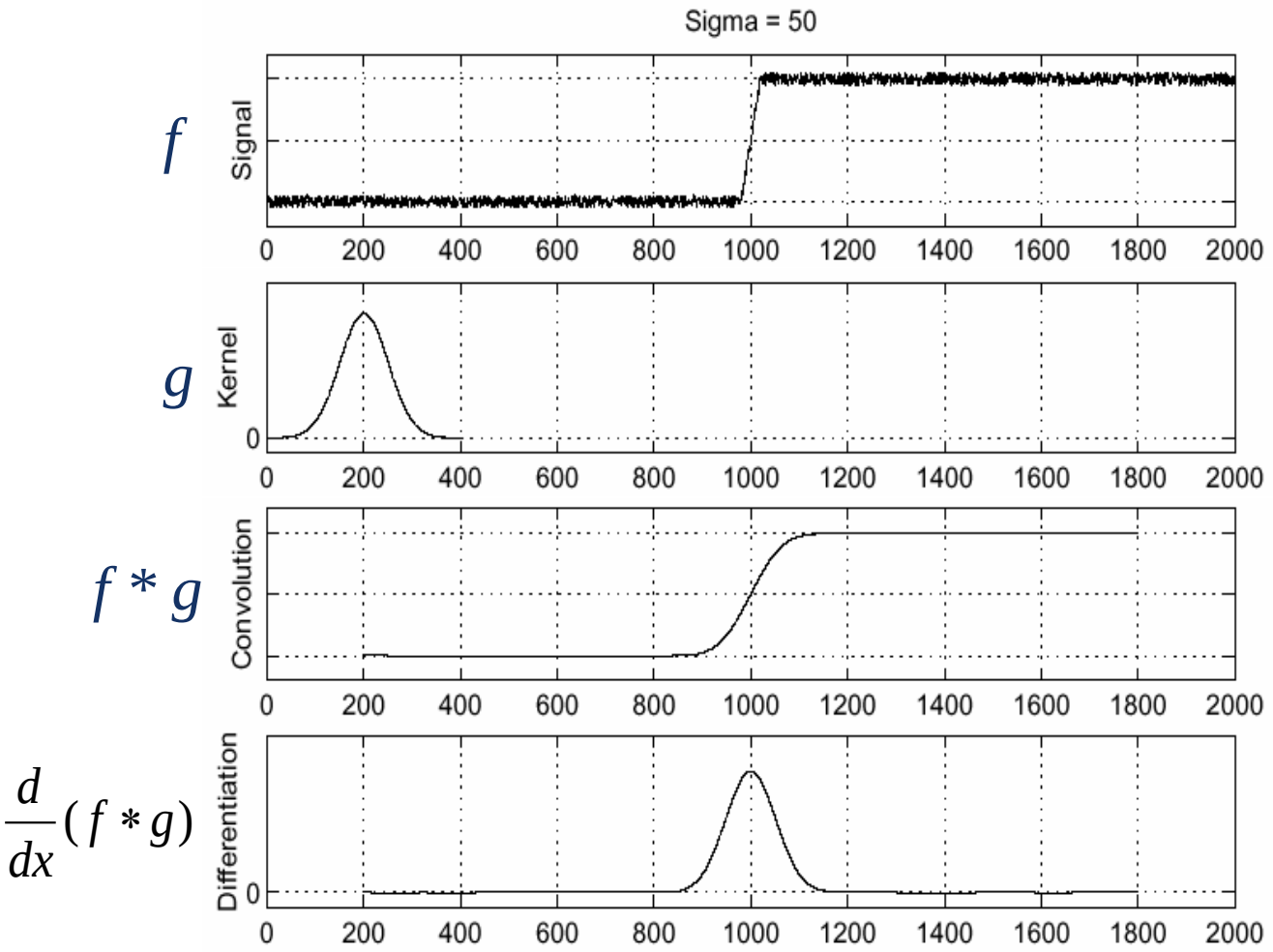


Where is the edge?

SMOOTHING WITH DIFFERENT FILTERS



SOLUTION: SMOOTH FIRST



to find edges, look for peaks in $\frac{d}{dx}(f * g)$

DERIVATIVE THEOREM OF CONVOLUTION

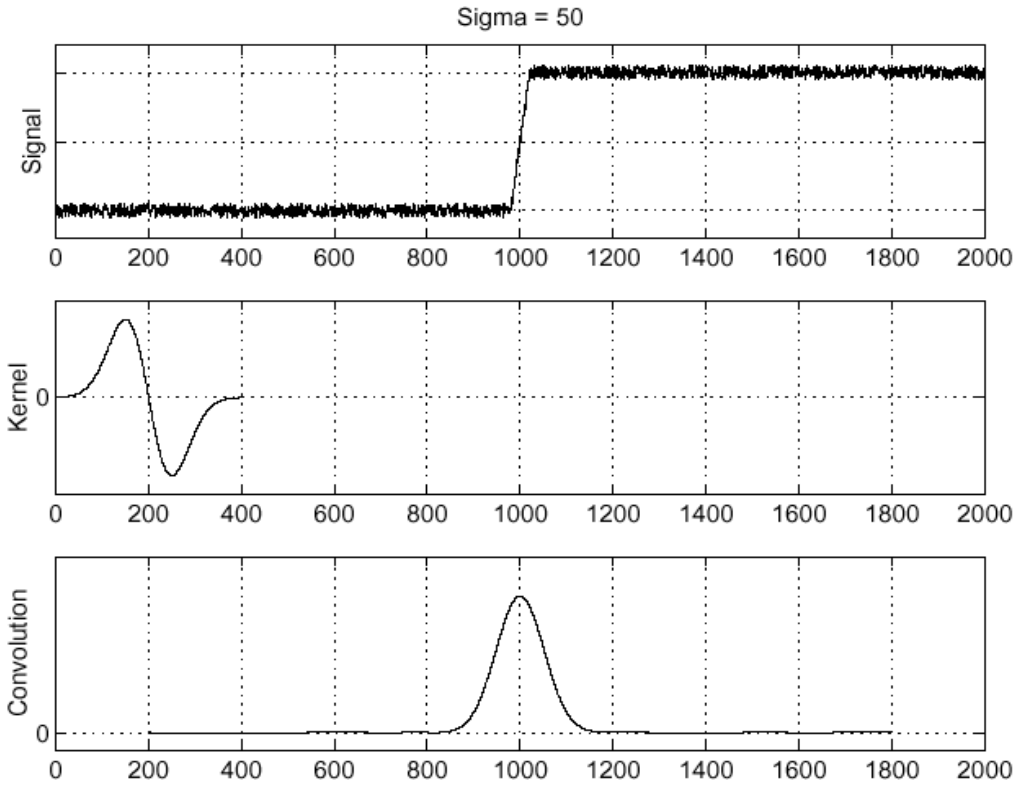
- This theorem gives us a very useful property: $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$

- This saves us one operation:

f

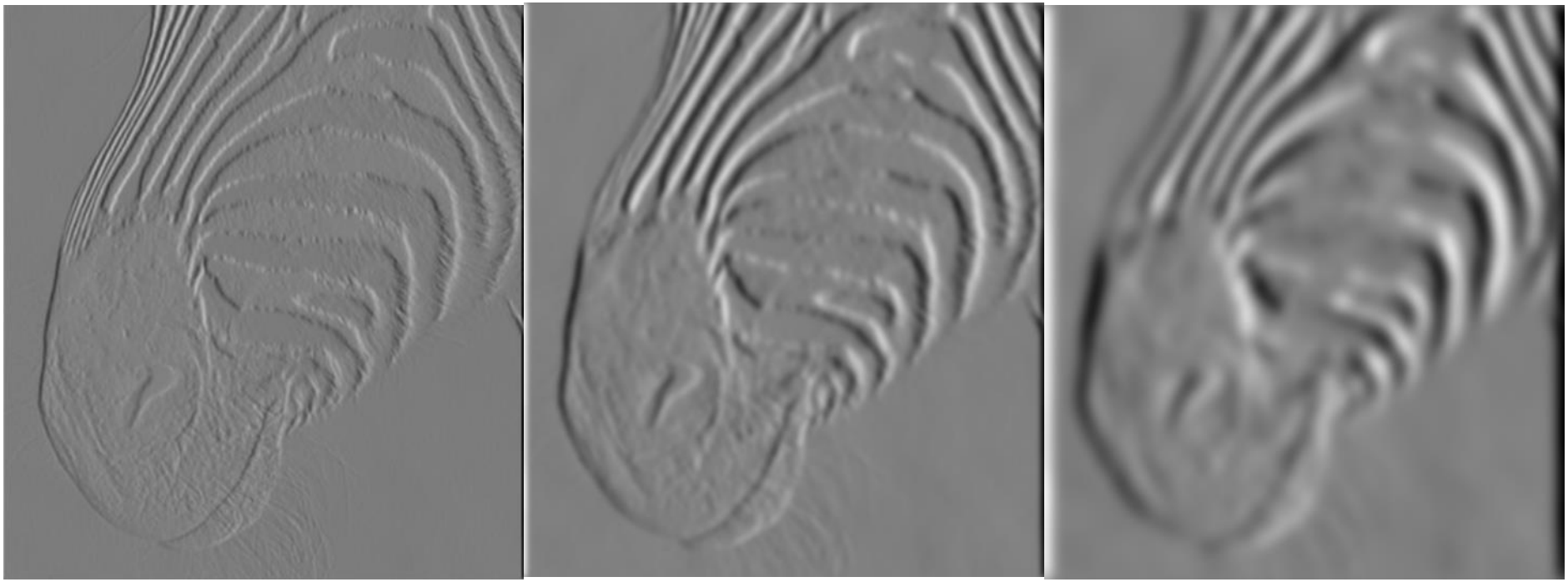
$\frac{d}{dx} g$

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



Source: S. Seitz

TRADEOFF BETWEEN SMOOTHING AT DIFFERENT SCALES



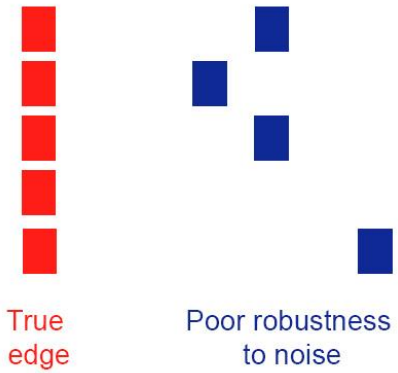
1 pixel 3 pixels 7 pixels

Smoothed derivative removes noise, but blurs edge.
Also finds edges at different “scales”.

JACOBS UNIVERSITY

Criteria for an "optimal" edge detector:

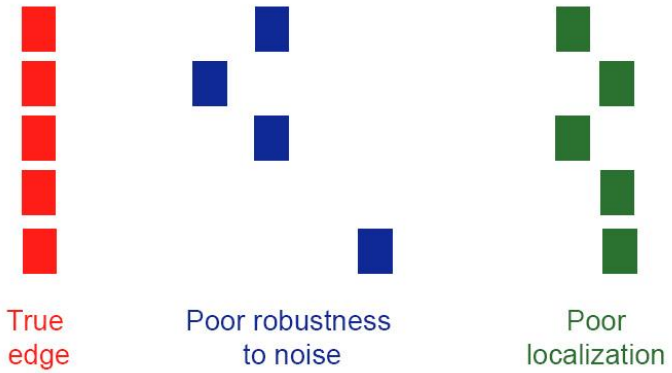
- **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)



JACOBS UNIVERSITY

Criteria for an “optimal” edge detector:

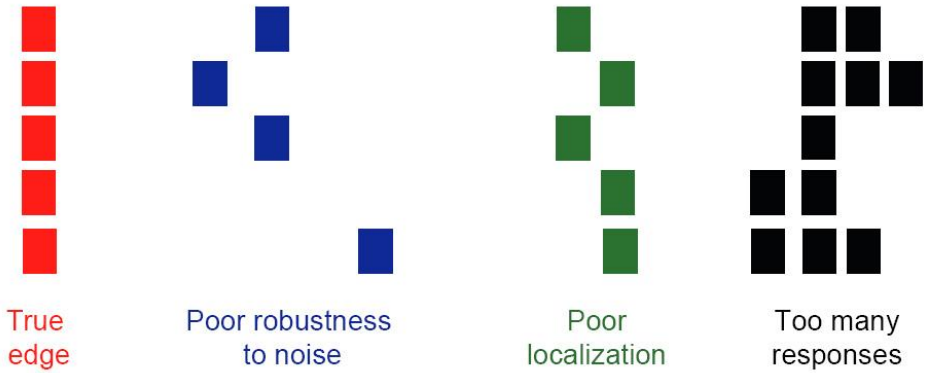
- **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
- **Good localization:** the edges detected must be as close as possible to the true edges



JACOBS UNIVERSITY

Criteria for an "optimal" edge detector:

- **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
- **Good localization:** the edges detected must be as close as possible to the true edges
- **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge

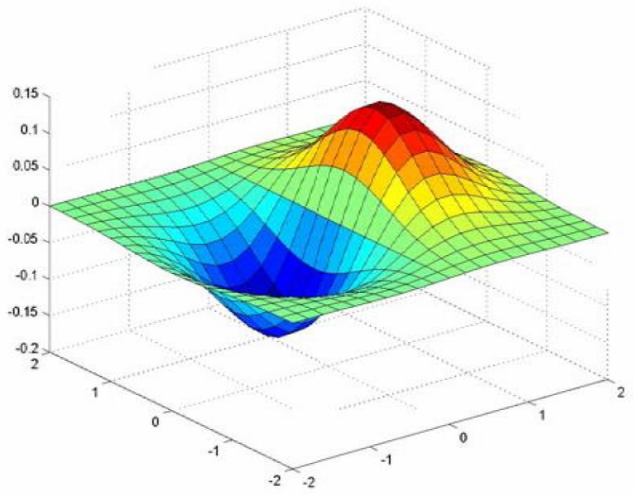


CANNY EDGE DETECTOR

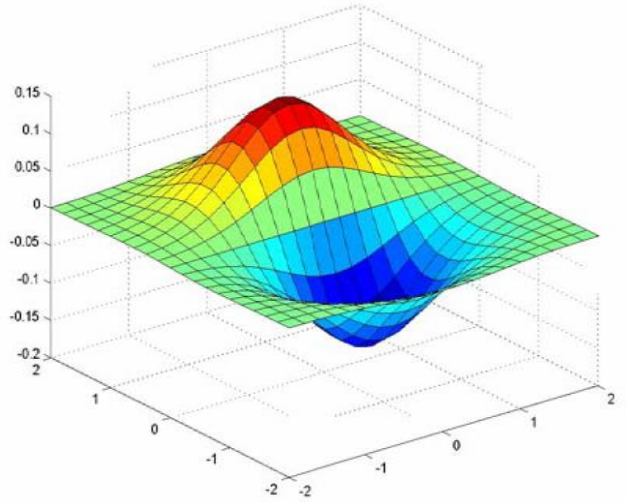
JACOBS UNIVERSITY

1. Suppress Noise
2. Compute gradient magnitude and direction
3. Apply Non-Maximum Suppression
   - Assures minimal response
4. Use hysteresis and connectivity analysis to detect edges

DERIVATIVE OF GAUSSIAN FILTER

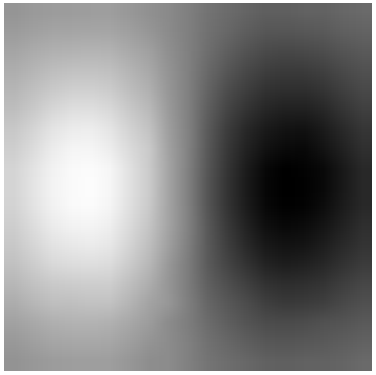


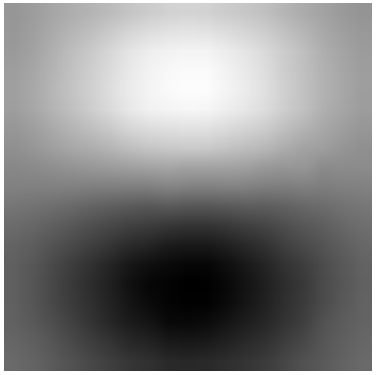
x-direction



y-direction

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

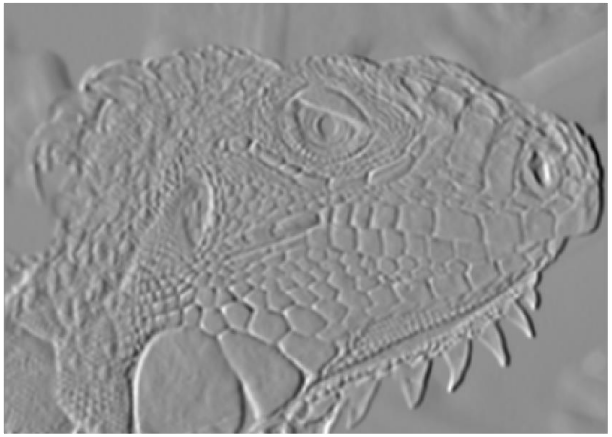




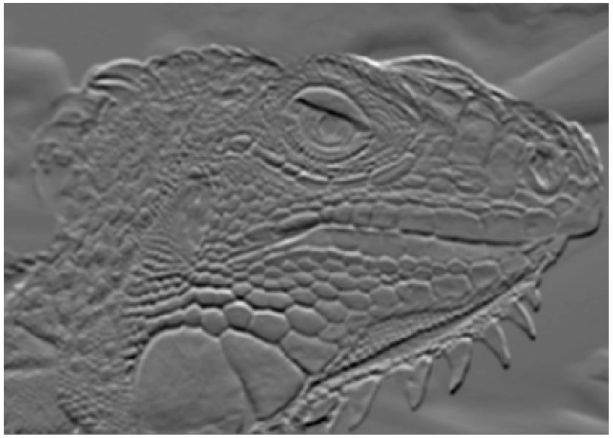
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

COMPUTE GRADIENTS

JACOBS UNIVERSITY



X-Derivative of Gaussian



Y-Derivative of Gaussian

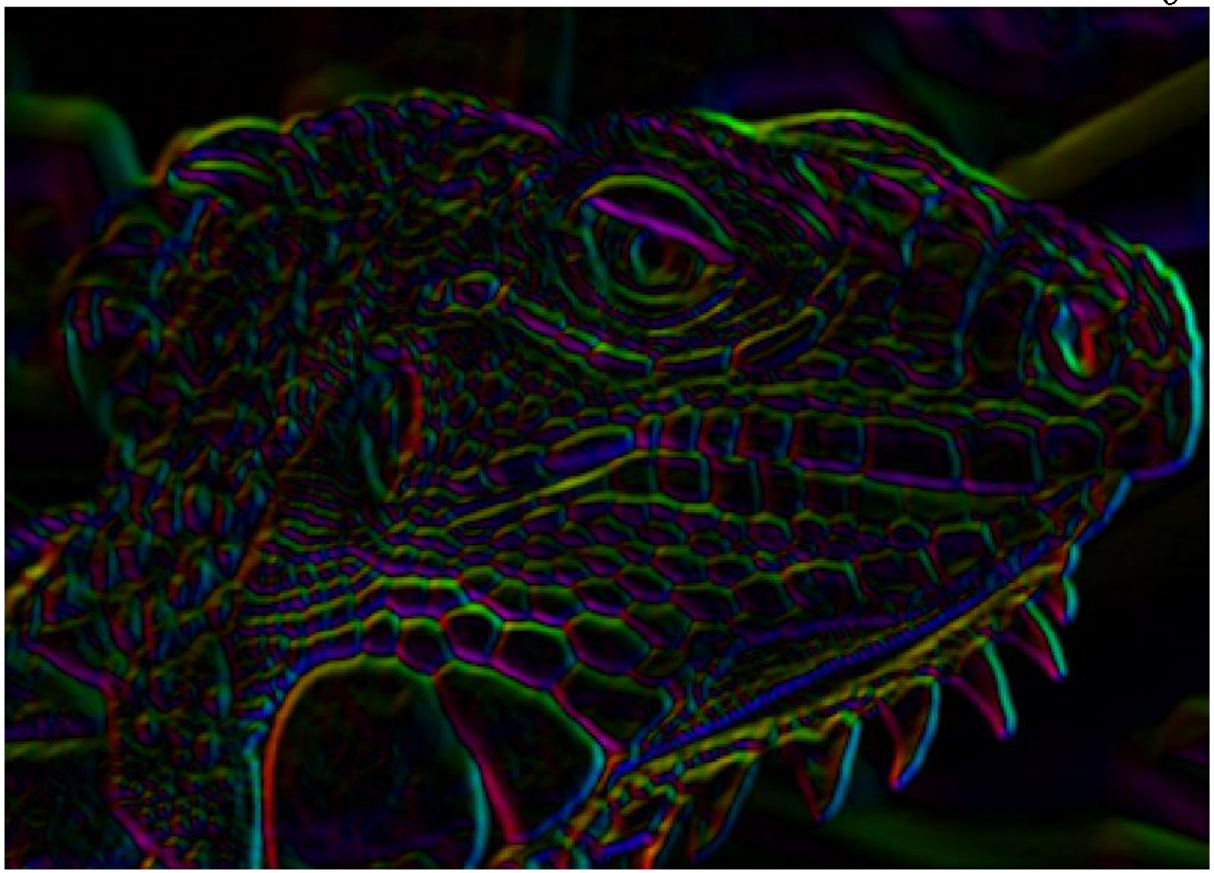
Gradient Magnitude

Source: J. Hayes

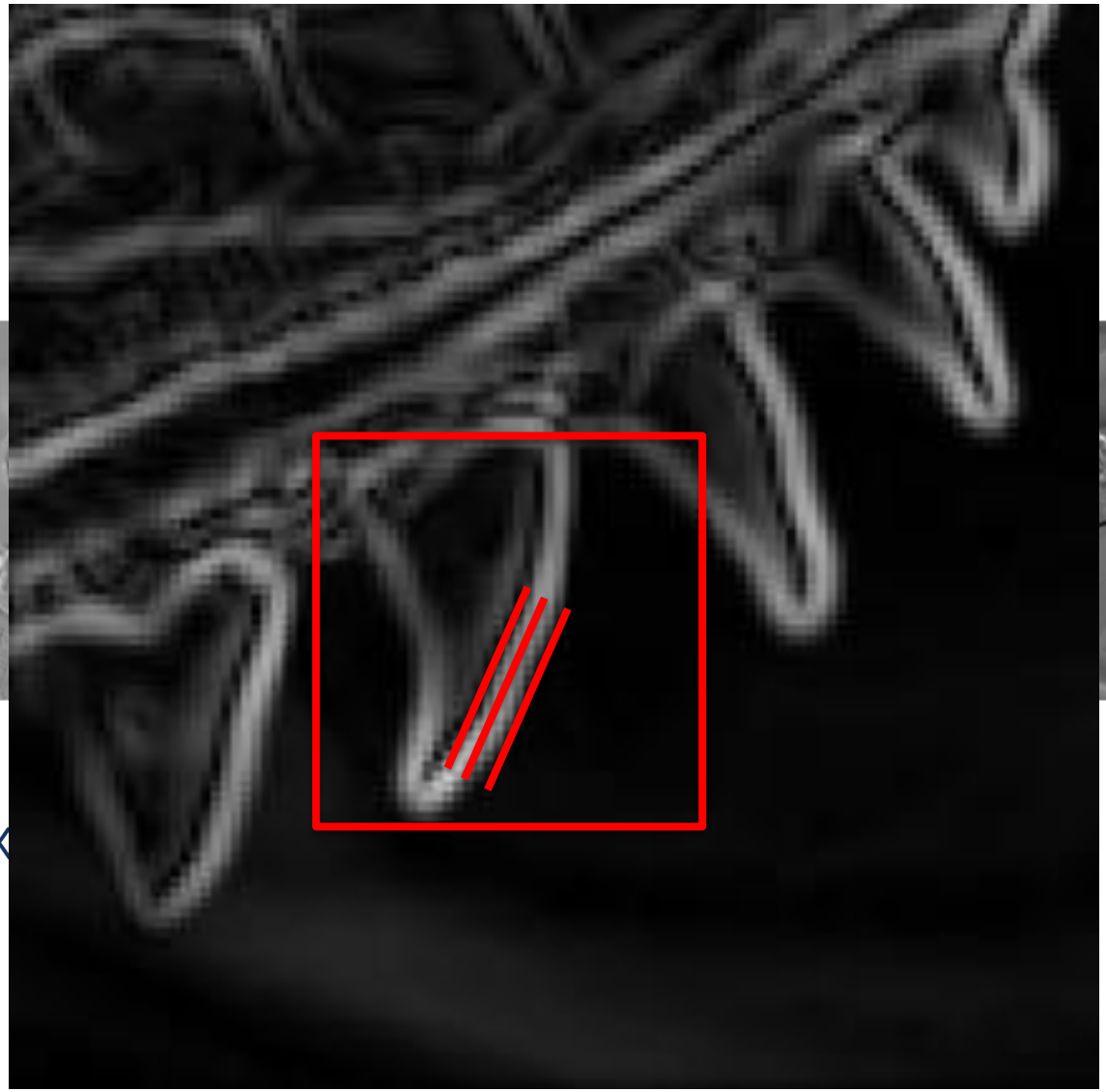
GET ORIENTATION AT EACH PIXEL

JACOBS UNIVERSITY

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$



COMPUTE GRADIENTS



X

Gradient Magnitude

NON-MAX SUPPRESSION

JACOBS UNIVERSITY

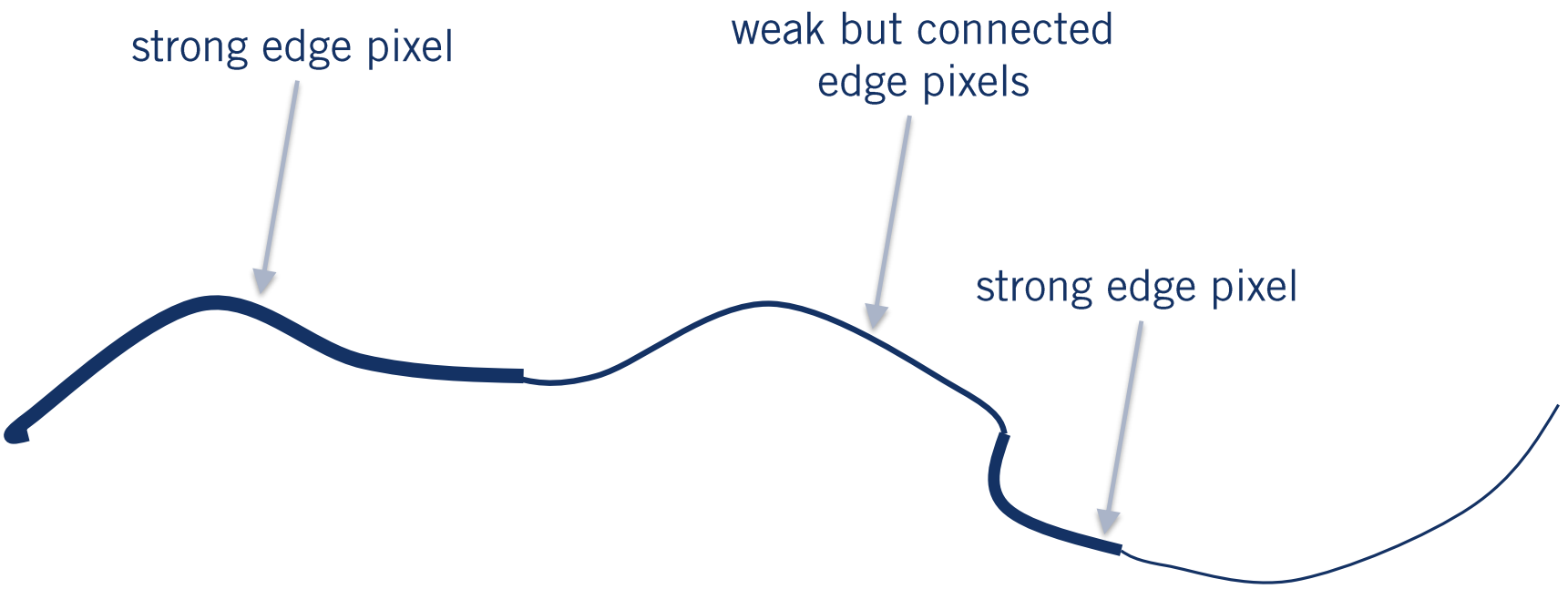
Before

After

If the gradient at a pixel is

1. above High, declare it as an ‘strong edge pixel’
2. below Low, declare it as a “non-edge-pixel”
3. between Low and High
   - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to a ‘strong edge pixel’ directly or via pixels between Low and High

HYSTERESIS THRESHOLDING

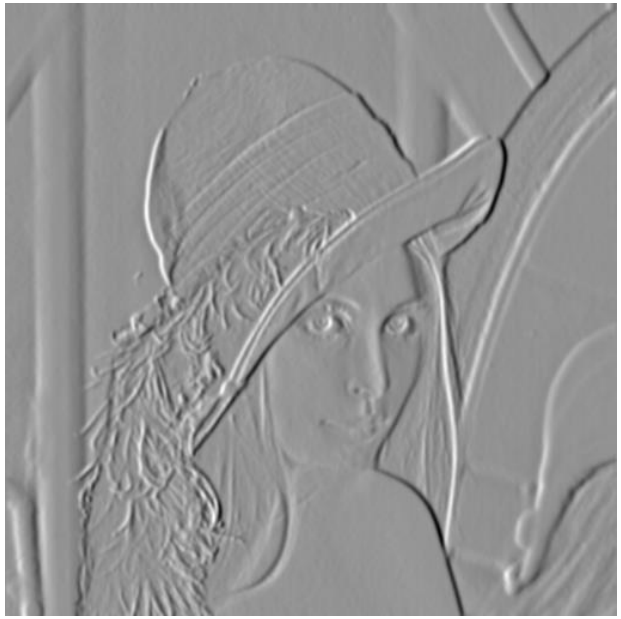


Source: S. Seitz

FINAL CANNY EDGES

Original Image

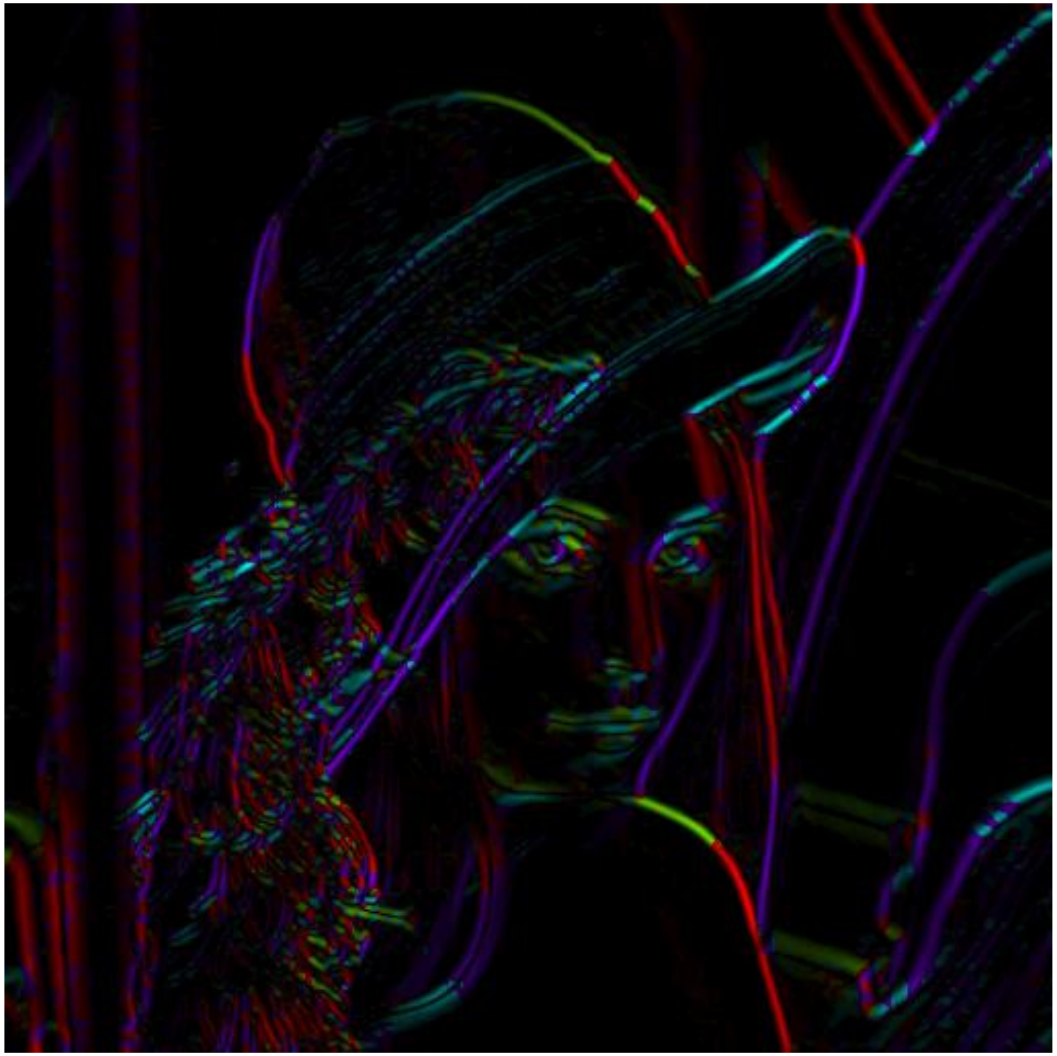
JACOBS UNIVERSITY



X-Derivative

Y-Derivative

Gradient Magnitude



Orientation at each pixel

Before non-max suppression

After non-max suppression

Hysteresis thresholding

Final Canny Edges

EFFECT OF σ (GAUSSIAN KERNEL SPREAD/SIZE)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

FITTING AS SEARCH IN PARAMETRIC SPACE

JACOBS UNIVERSITY

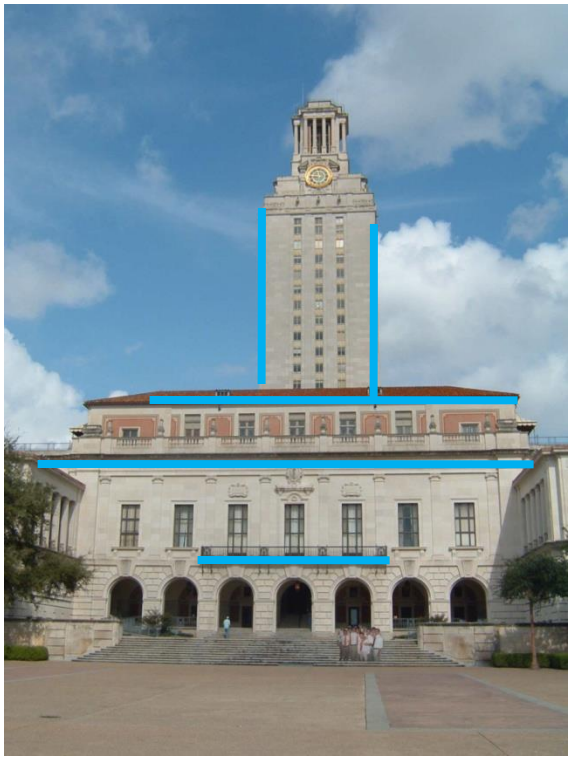
- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can't tell whether a point belongs to a given model just by looking at that point.
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features

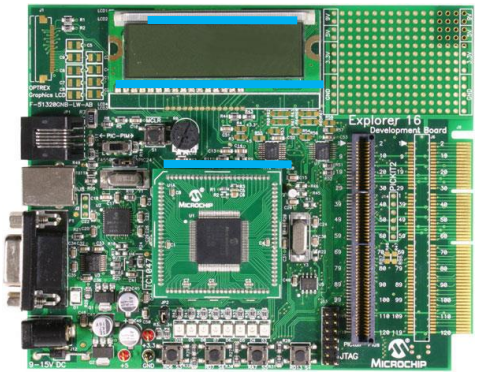
LINE FITTING

JACOBS UNIVERSITY

Why?

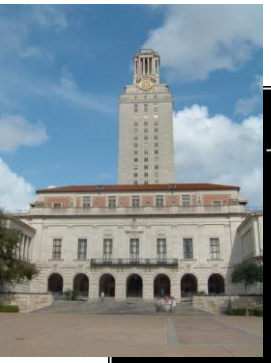
Many objects characterized by presence of straight lines

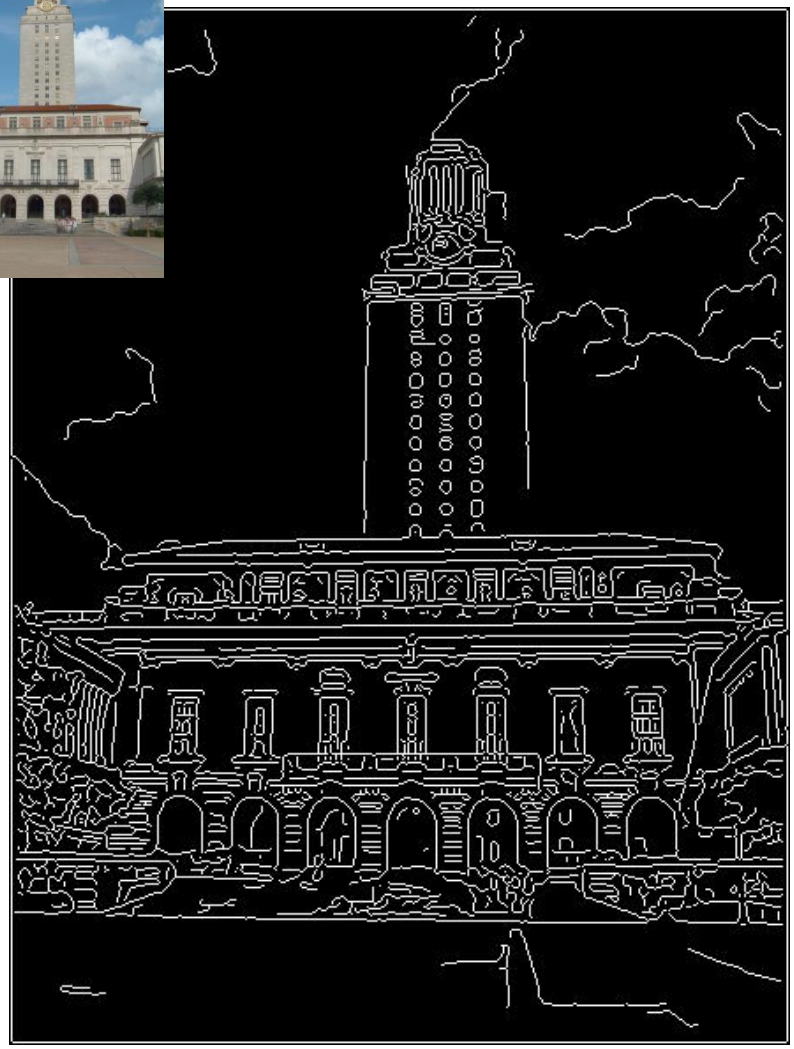


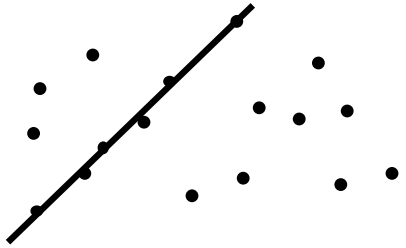


Wait, why aren't we done just by running edge detection?

DIFFICULTY OF LINE FITTING







- Extra edge points (clutter), multiple models:
  - Which points go with which line, if any?
- Only some parts of each line detected, and some parts are missing:
  - How to find a line that bridges missing evidence?
- Noise in measured edge points, orientations:
  - How to detect true underlying parameters?

INTRO TO HOUGH TRANSFORM

JACOBS UNIVERSITY

- The Hough transform (HT) can be used to detect lines.
- It was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972).
- The goal is to find the location of lines in images.
- Caveat: Hough transform can detect lines, circles and other structures ONLY if their parametric equation is known.
- It can give robust detection under noise and partial occlusion

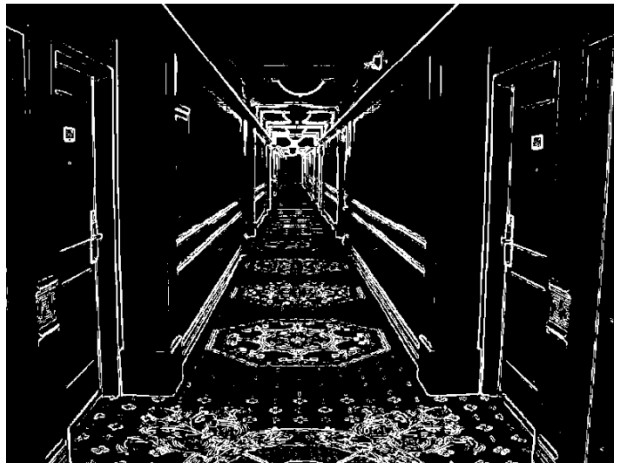
PRIOR TO HOUGH TRANSFORM

JACOBS UNIVERSITY

Assume that we have performed some edge detection, and a thresholding of the edge magnitude image.

Thus, we have some pixels that may partially describe the boundary of some objects.





JACOBS UNIVERSITY

We wish to find sets of pixels that make up straight lines.

Consider a point of known coordinates $(x_i;y_i)$

- There are many lines passing through the point (x_i, y_i).

Straight lines that pass that point have the form $y_i = a*x_i + b$

- Common to them is that they satisfy the equation for some set of parameters (a, b)

JACOBS UNIVERSITY

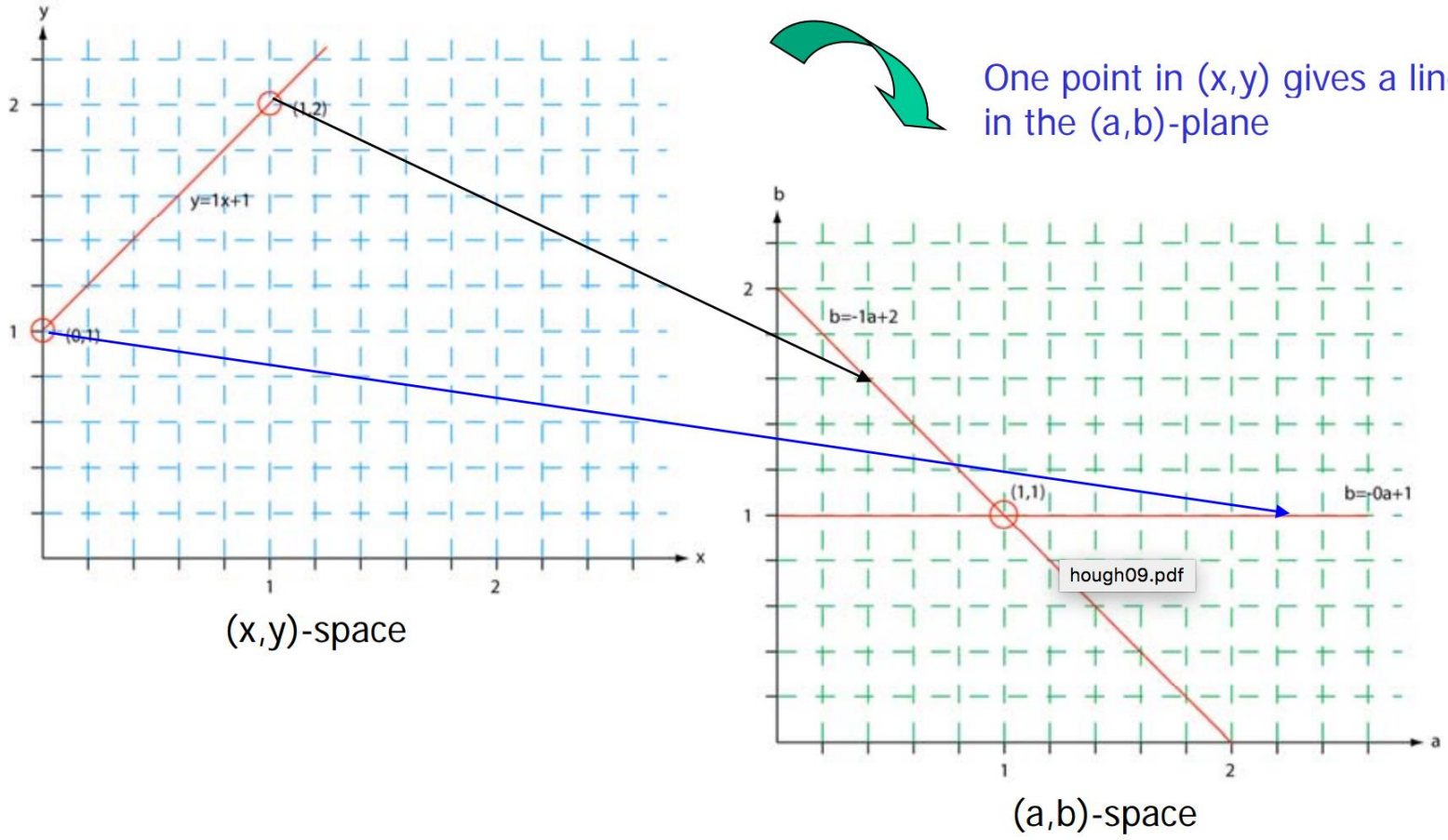
This equation can obviously be rewritten as follows:

- $b = -a^*x_i + y_i$
- We can now consider x and y as parameters
- a and b as variables.

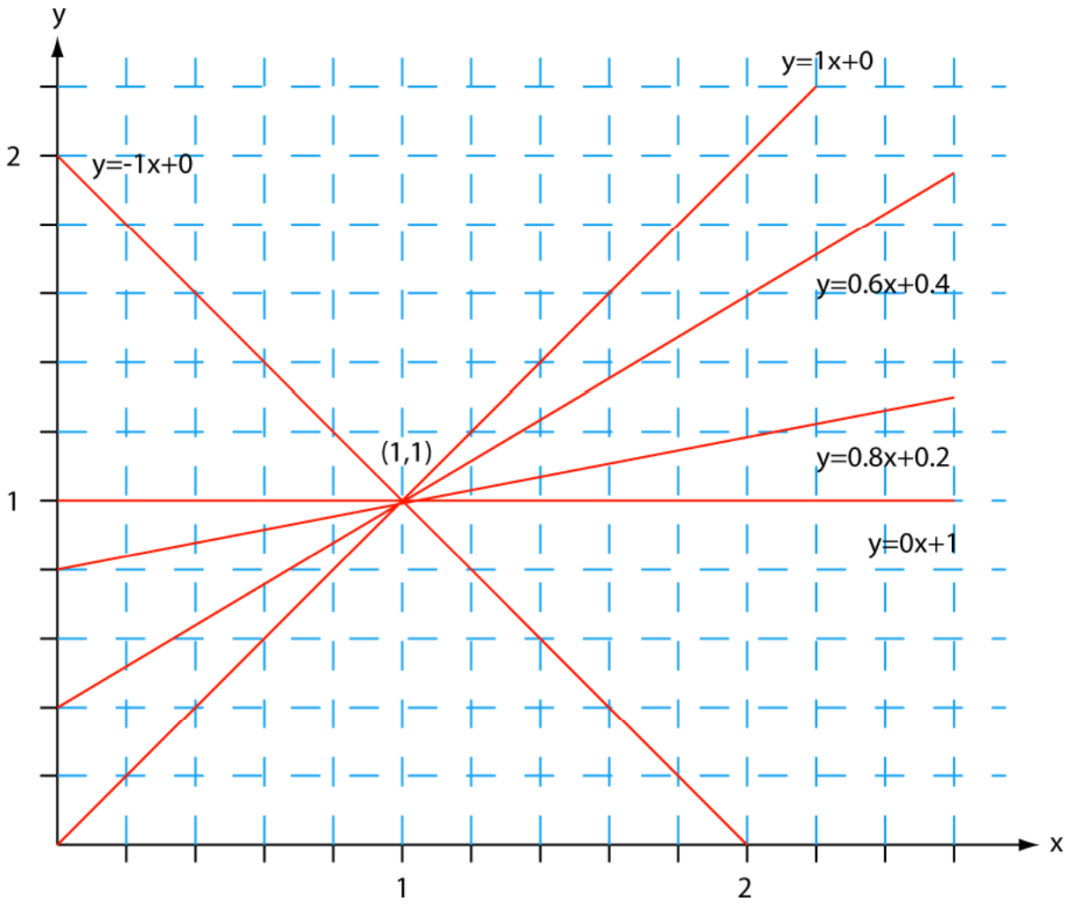
This is a line in (a, b) space parameterized by x and y.

- So: a single point in x_1,y_1-space gives a line in (a,b) space.
- Another point (x_2, y_2) will give rise to another line (a,b) space.

DETECTING LINES USING HOUGH TRANSFORM



DETECTING LINES USING HOUGH TRANSFORM



- Two points (x_1, y_1) and$(x_2\ y_2)$ define a line in the (x, y) plane.
- These two points give rise to two different lines in (a,b) space.
- In (a,b) space these lines will intersect in a point (a' b')
- All points on the line defined by (x_1, y_1) and $(x_2\ , y_2)$ in (x, y) space will parameterize lines that intersect in (a', b') in (a,b) space.

ALGORITHM FOR HOUGH TRANSFORM

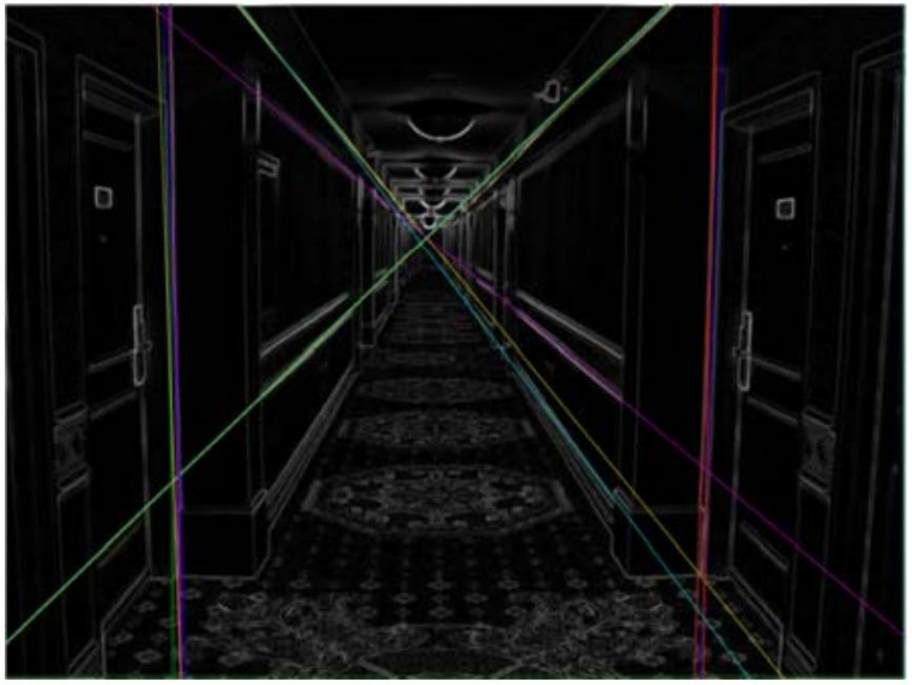
JACOBS UNIVERSITY

1. Quantize the parameter space (a b) by dividing it into cells
2. This quantized space is often referred to as the accumulator cells.
3. Count the number of times a line intersects a given cell.
   - For each pair of points (x_1, y_1) and (x_2, y_2) detected as an edge, find the intersection (a',b') in (a, b)space.
   - Increase the value of a cell in the range $[[a_{min}, a_{max}],[b_{min},b_{max}]]$ that (a', b') belongs to.
   - Cells receiving more than a certain number of counts (also called 'votes') are assumed to correspond to lines in (x,y) space.

OUTPUT OF HOUGH TRANSFORM

JACOBS UNIVERSITY

1. Here are the top 20 most voted lines in the image:



JACOBS UNIVERSITY

Advantages:

- Conceptually simple.
- Easy implementation
- Handles missing and occluded data very gracefully.
- Can be adapted to many types of forms, not just lines

Disadvantages:

- Computationally complex for objects with many parameters.
- Looks for only one single type of object
- Can be “fooled” by “apparent lines”.
- The length and the position of a line segment cannot be determined.
- Co-linear line segments cannot be separated.

RANSAC [FISCHLER & BOLLES 1981]

JACOBS UNIVERSITY

1. RANdom SAmple Consensus

2. Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use only those.

3. Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

RANSAC [FISCHLER & BOLLES 1981]

JACOBS UNIVERSITY

RANSAC loop:

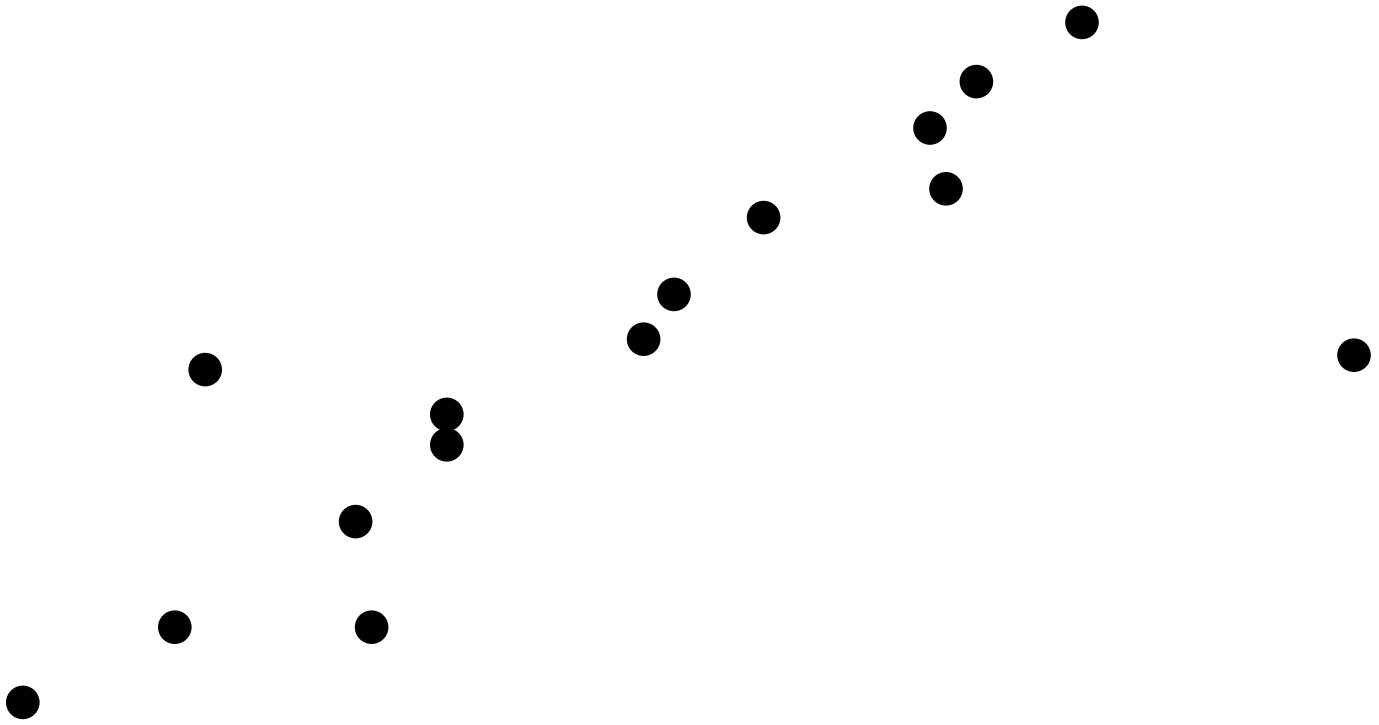
1. Randomly select a *seed group* of points
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers
5. Keep the transformation with the largest number of inliers

RANSAC LINE FITTING EXAMPLE

JACOBS UNIVERSITY

1. Task: Estimate the best line

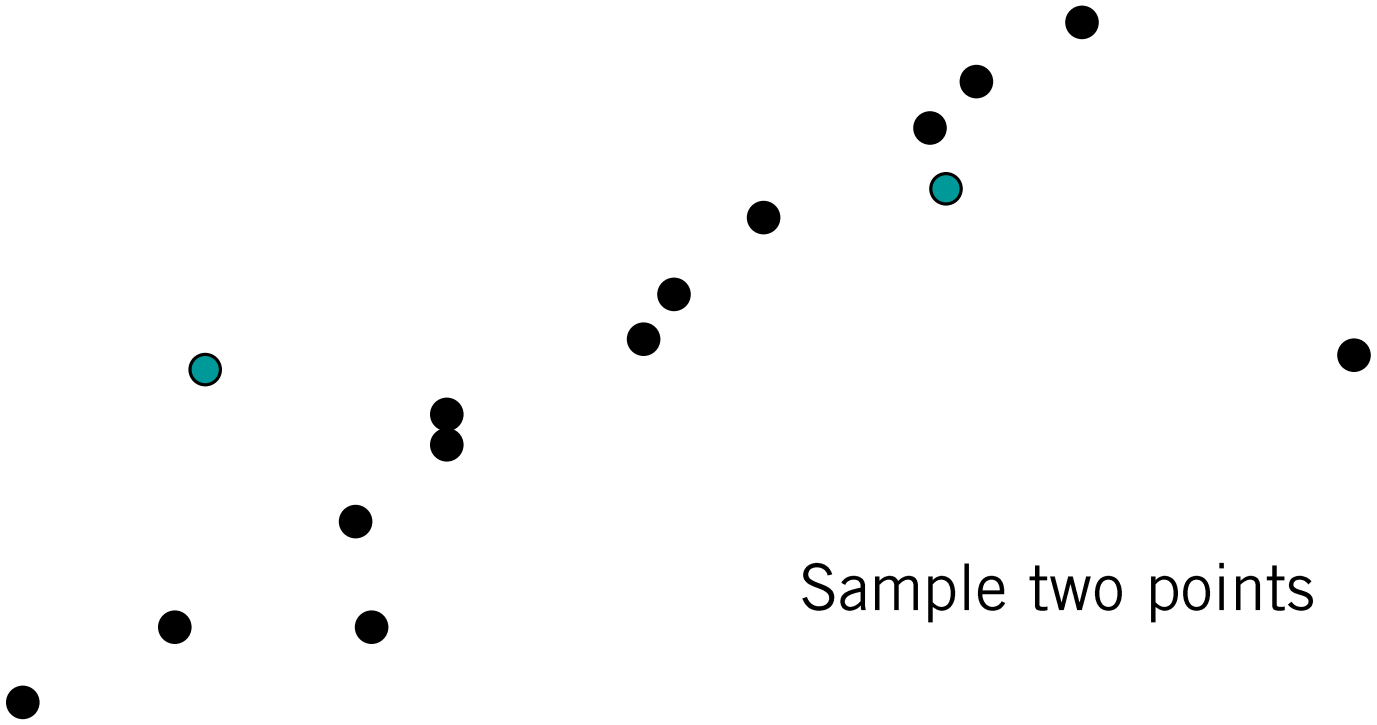
- *How many points do we need to estimate the line?*



RANSAC LINE FITTING EXAMPLE

JACOBS UNIVERSITY

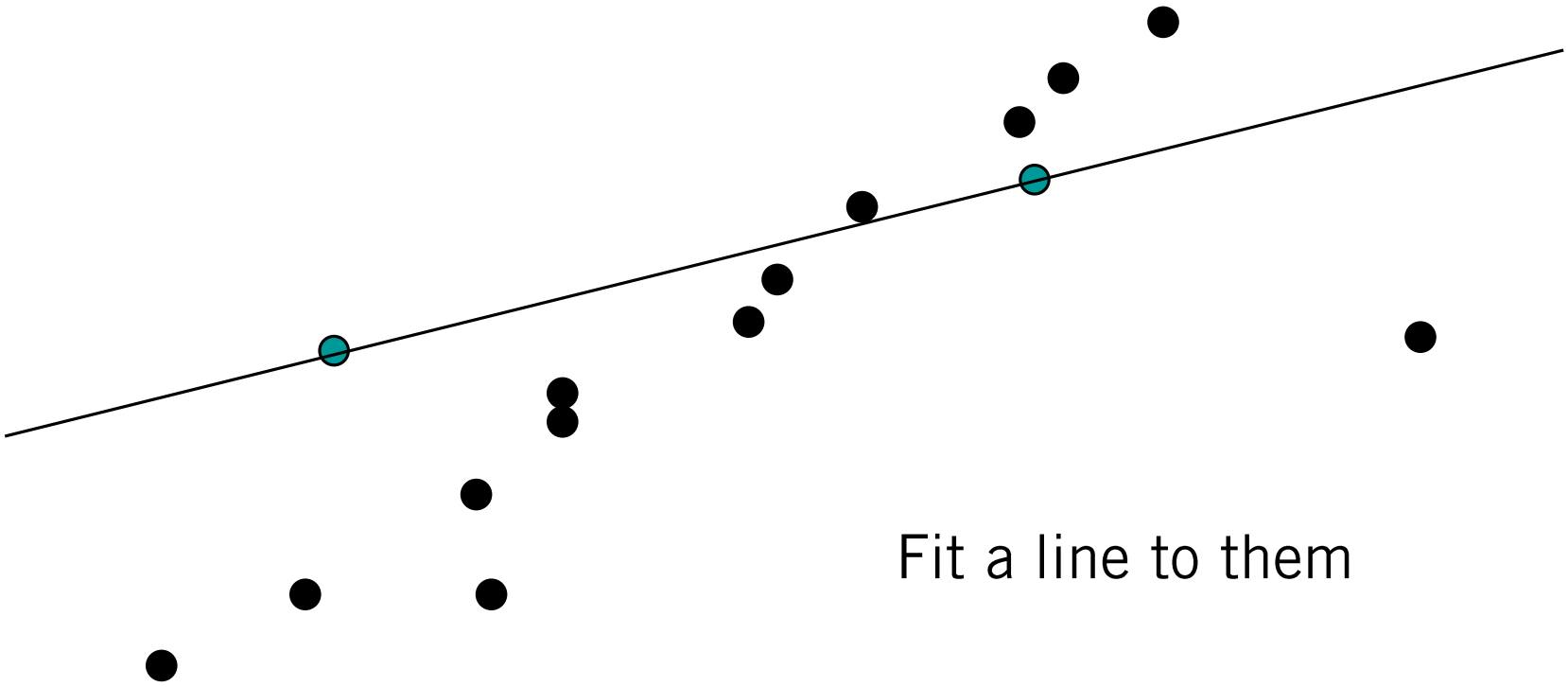
1. Task: Estimate the best line



RANSAC LINE FITTING EXAMPLE

JACOBS UNIVERSITY

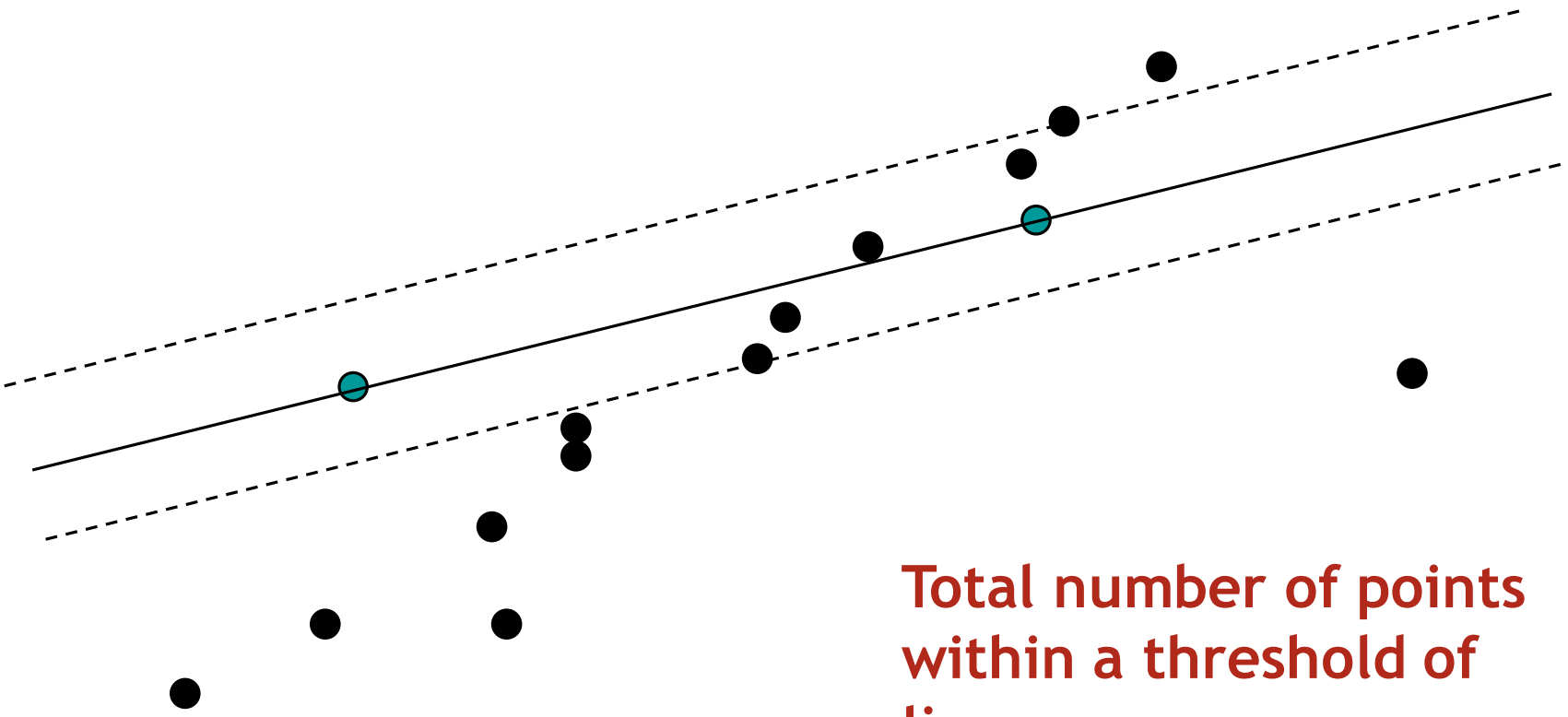
1. Task: Estimate the best line



RANSAC LINE FITTING EXAMPLE

JACOBS UNIVERSITY

1. Task: Estimate the best line

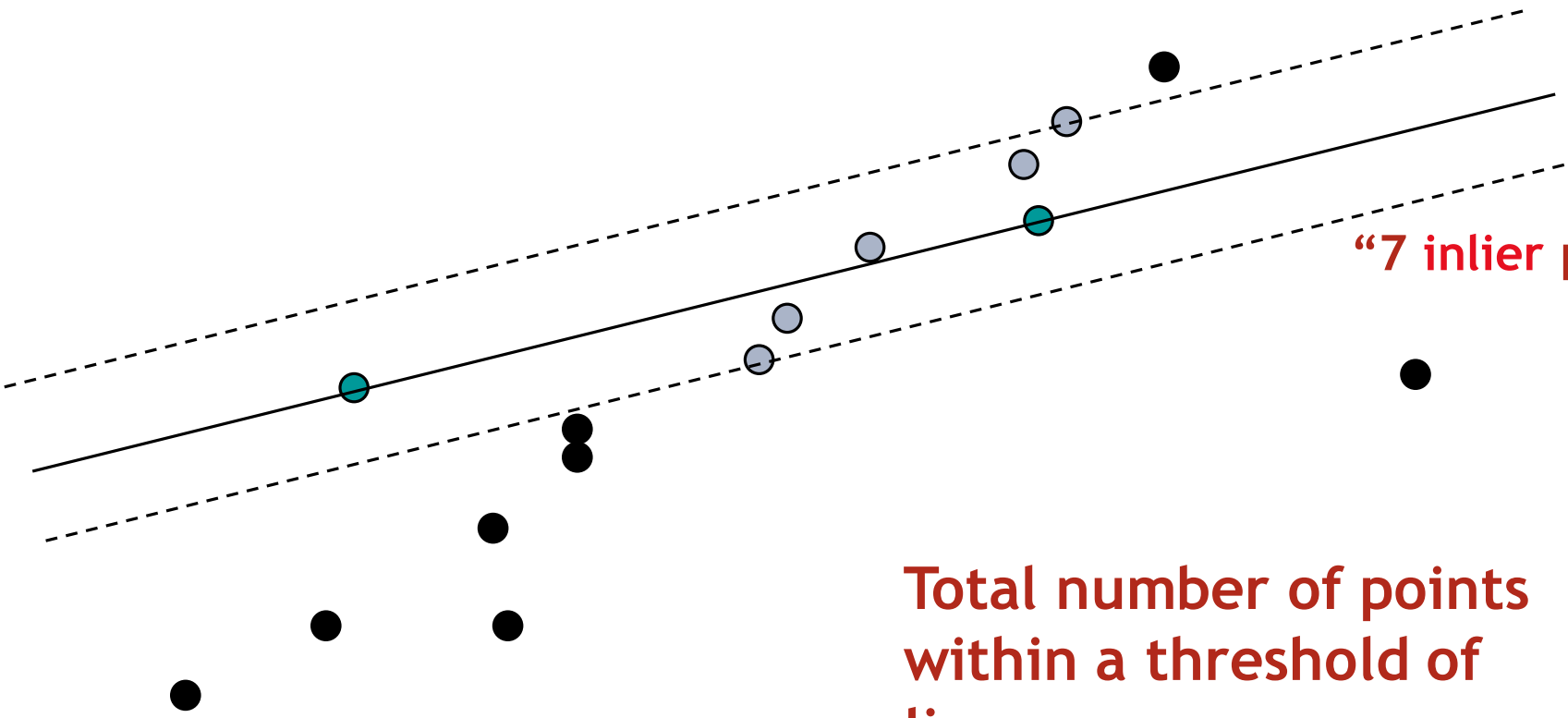


Total number of points within a threshold of line.

RANSAC LINE FITTING EXAMPLE

JACOBS UNIVERSITY

1. Task: Estimate the best line

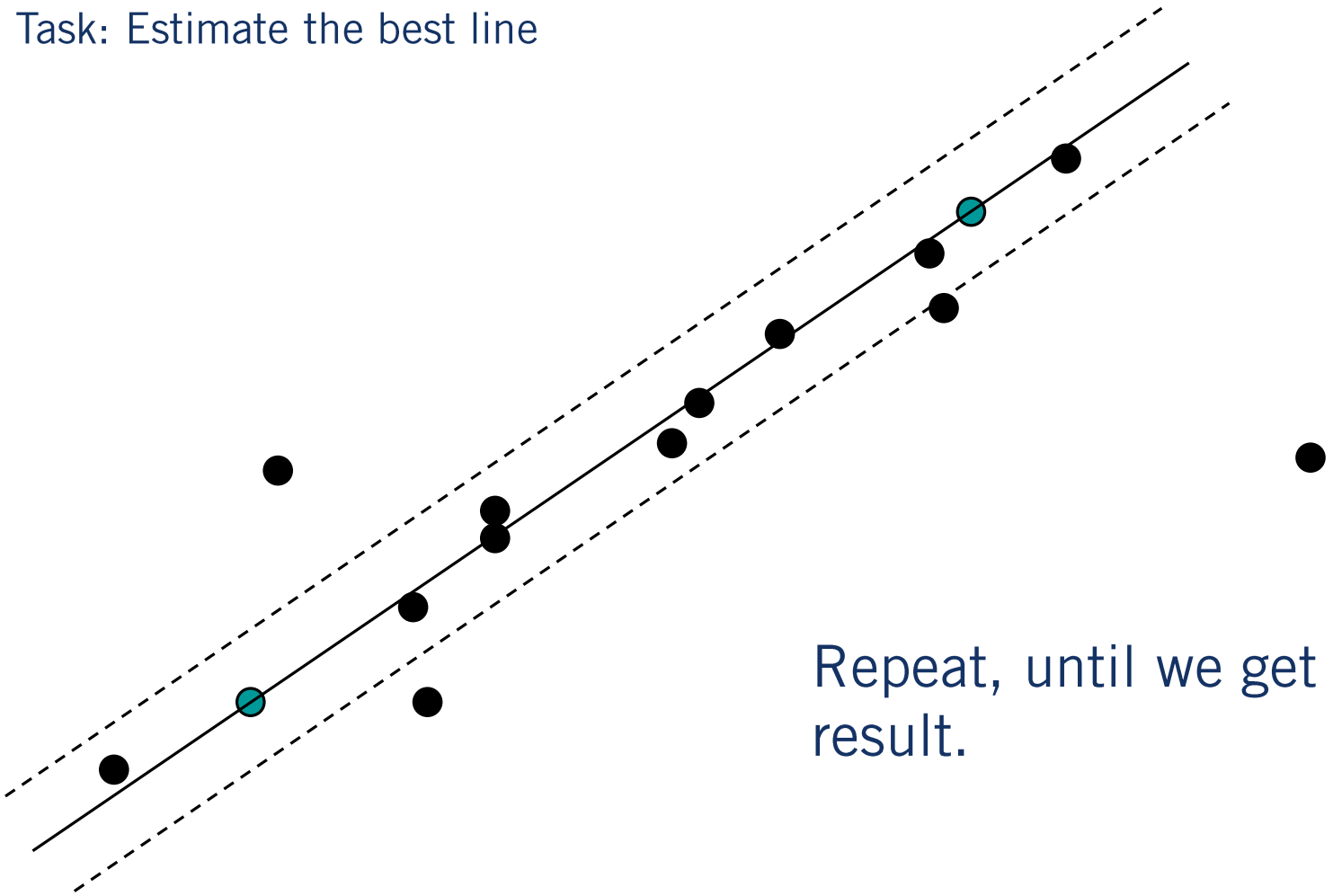


"7 inlier points"

Total number of points within a threshold of line.

RANSAC LINE FITTING EXAMPLE

1. Task: Estimate the best line

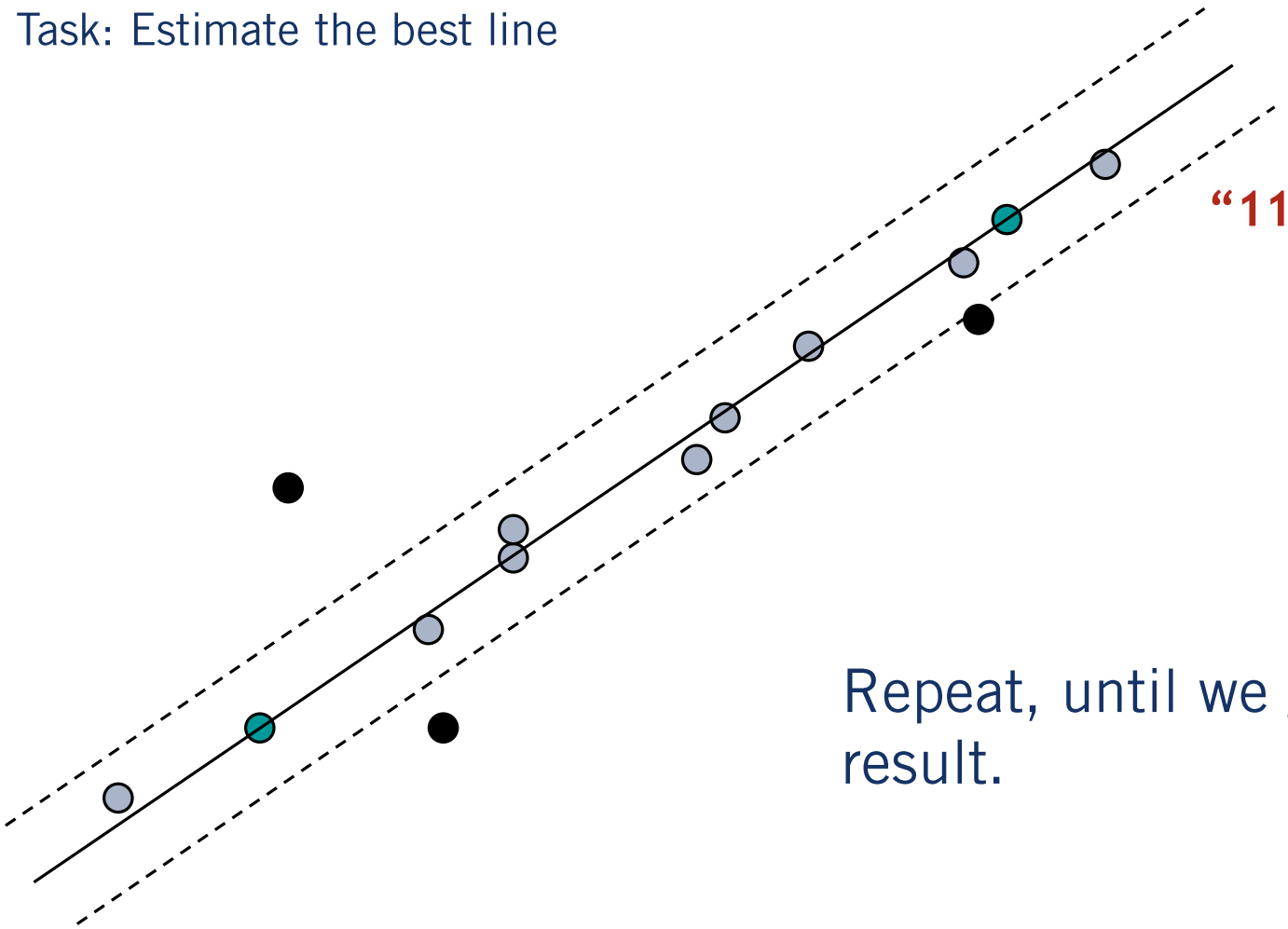


Repeat, until we get a good result.

RANSAC LINE FITTING EXAMPLE

JACOBS UNIVERSITY

1. Task: Estimate the best line



“11 inlier points”

Repeat, until we get a good result.

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
    n — the smallest number of points required
    k — the number of iterations required
    t — the threshold used to identify a point that fits well
    d — the number of nearby points required
      to assert a model fits well
Until k iterations have occurred
    Draw a sample of n points from the data
      uniformly and at random
    Fit to that set of n points
    For each data point outside the sample
      Test the distance from the point to the line
        against t; if the distance from the point to the line
        is less than t, the point is close
    end
    If there are d or more points close to the line
      then there is a good fit. Refit the line using all
      these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

JACOBS UNIVERSITY

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

1. Prob. that all k samples fail is: $(1-w^n)^k$

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

RANSAC: COMPUTED K (P=0.99)

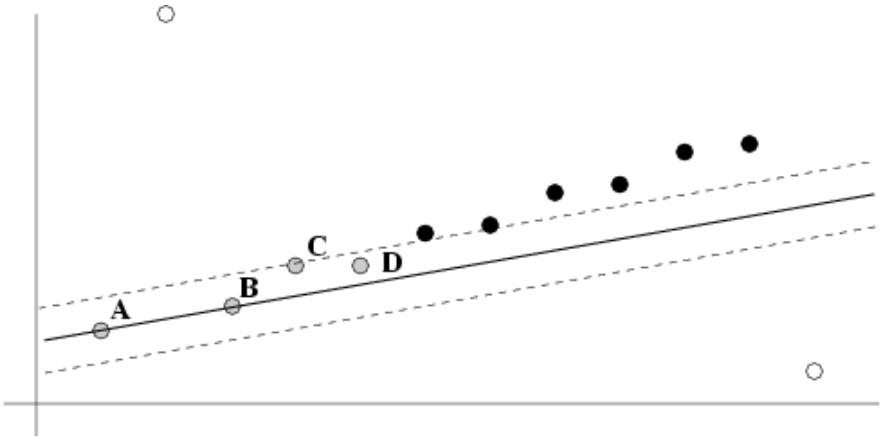
JACOBS UNIVERSITY

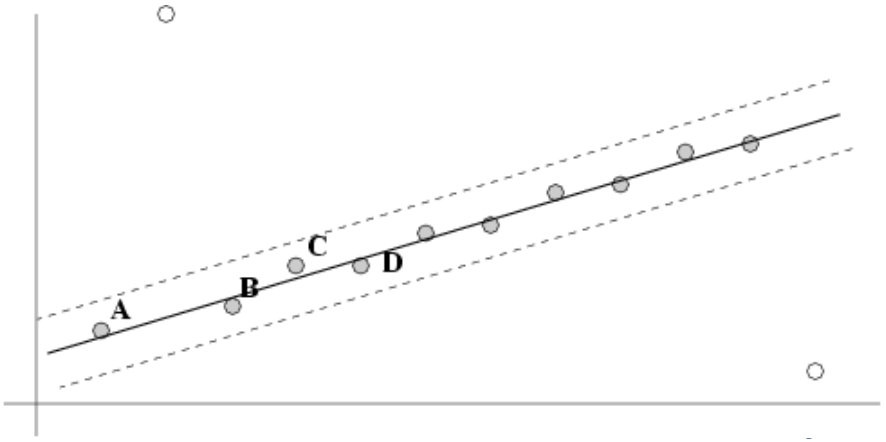
Sample size n	Proportion of outliers						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

AFTER RANSAC

JACOBS UNIVERSITY

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





- **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

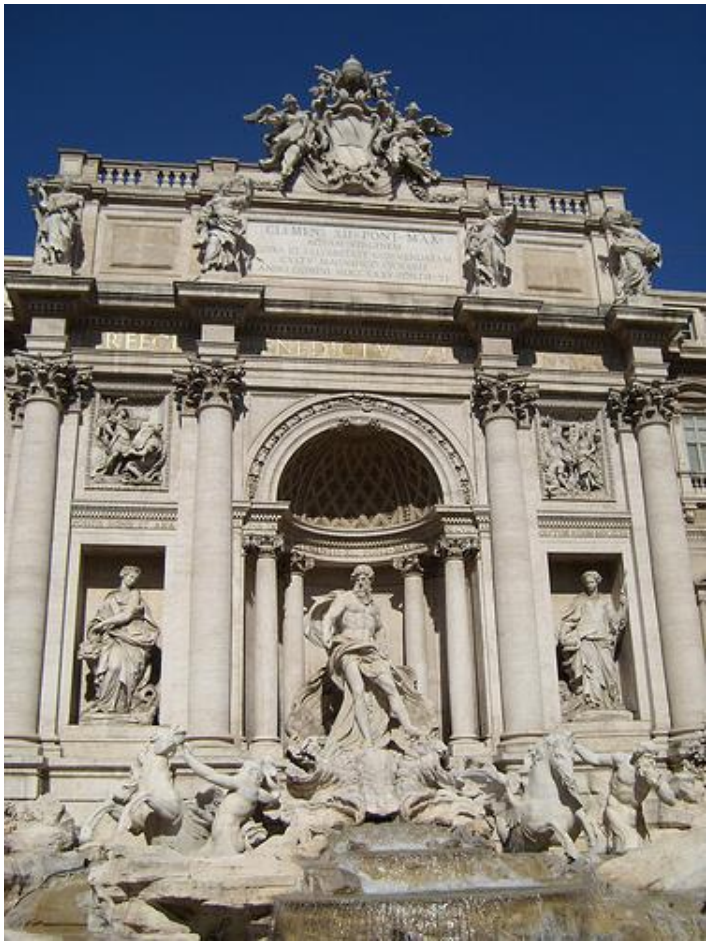
IMAGE MATCHING: A CHALLENGING PROBLEM

JACOBS UNIVERSITY

IMAGE MATCHING: A CHALLENGING PROBLEM

JACOBS UNIVERSITY

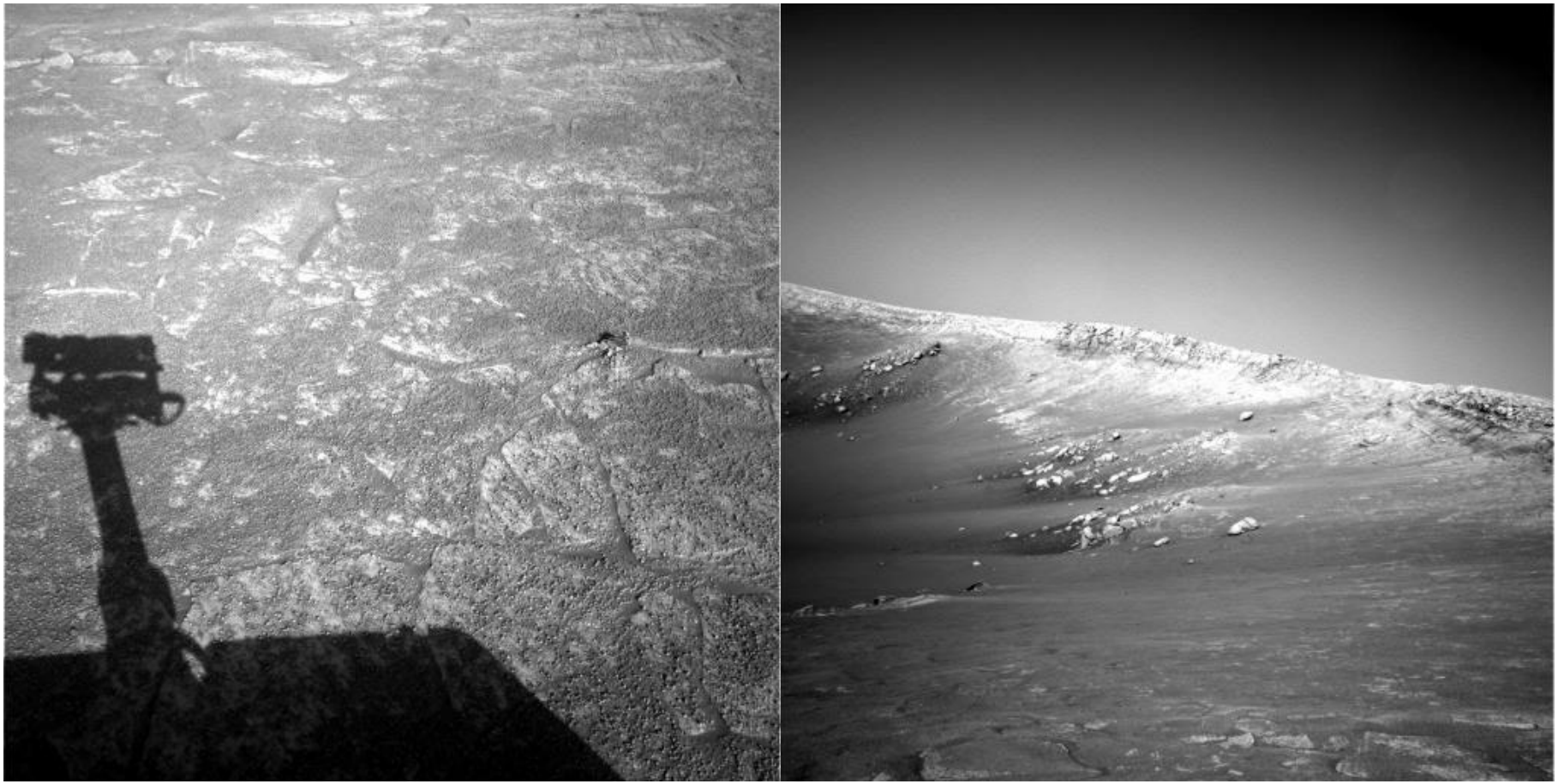
by Diva Sian



by swashford

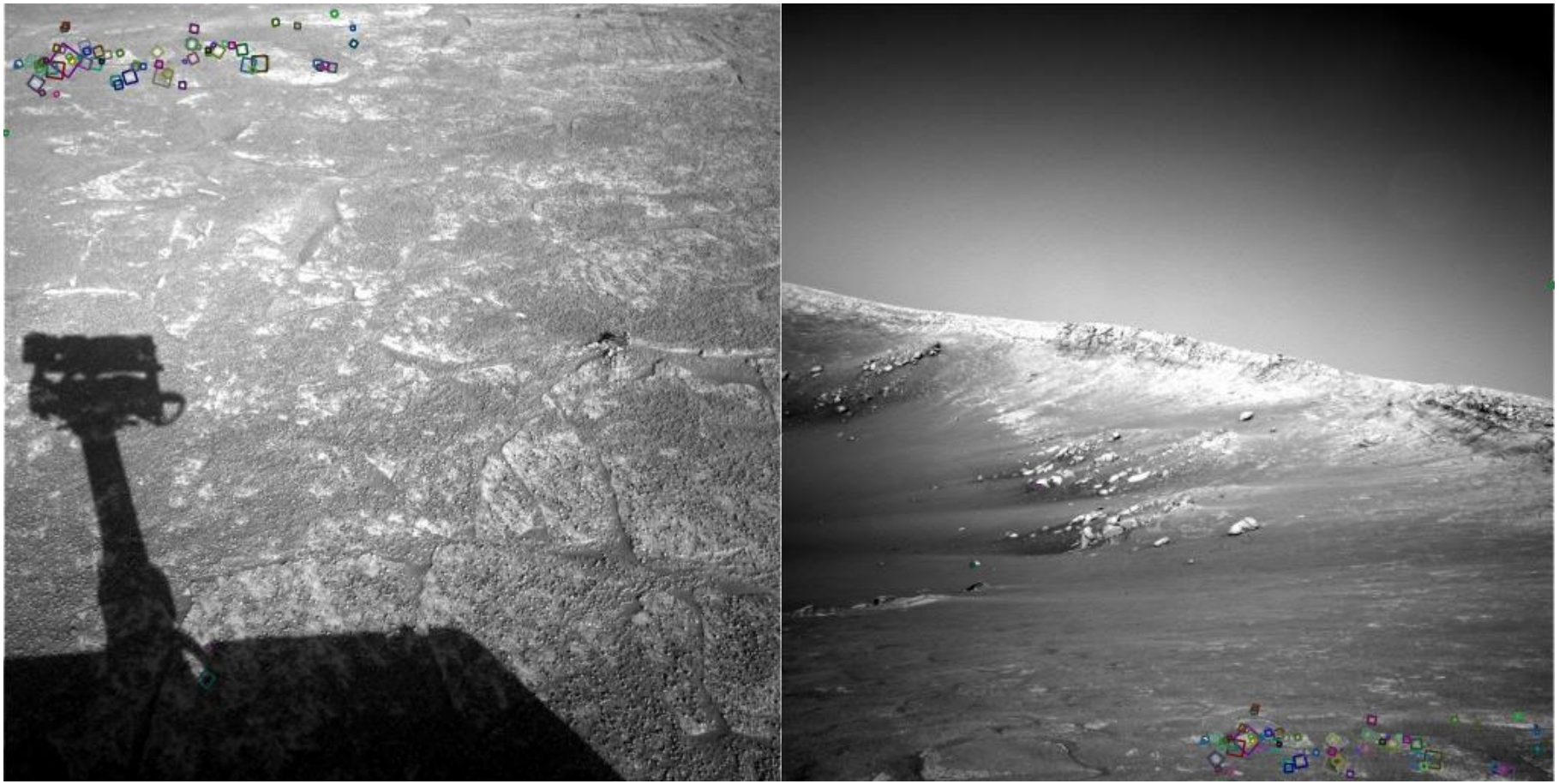
HARDER CASE

EVEN HARDER ...



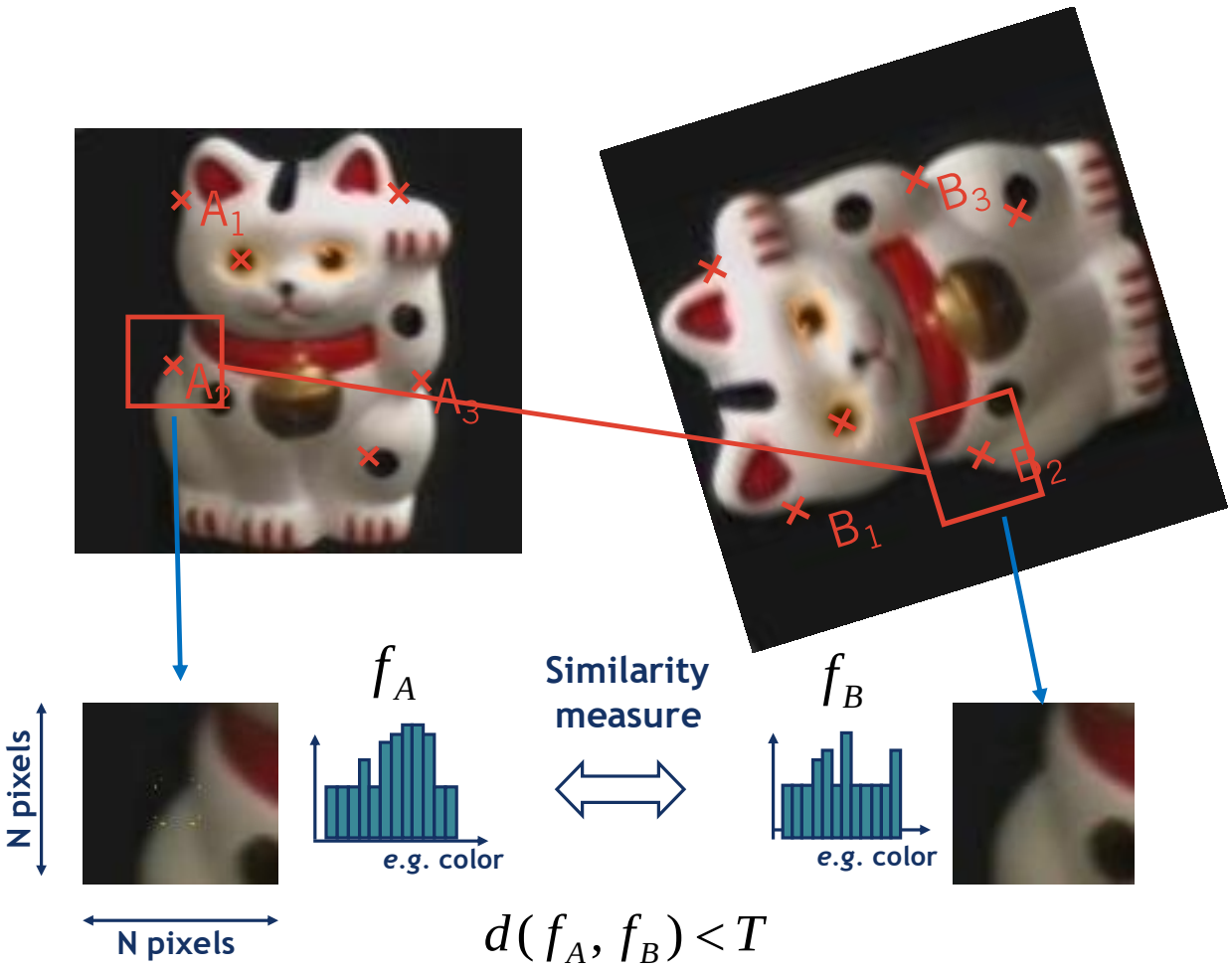
NASA Mars Rover images

EVEN HARDER ...



NASA Mars Rover images with SIFT feature matches
(Figure by Noah Snavely)

GENERAL APPROACH



1. Find a set of distinctive key-points
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

JACOBS UNIVERSITY

Problem 1:

- Detect the same point *independently* in both images

No chance to match!

We need a repeatable detector!

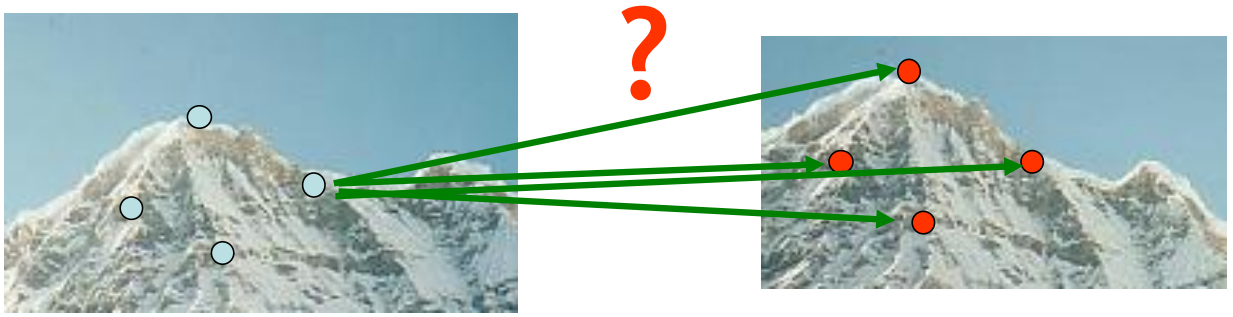
JACOBS UNIVERSITY

Problem 1:

- Detect the same point *independently* in both images

Problem 2:

- For each point correctly recognize the corresponding one



We need a reliable and distinctive descriptor!

INVARIANCE: GEOMETRIC TRANSFORMATIONS