Computer Vision
Model Fitting

WS 2019/2020

Prof. Dr. Simone Frintrop

Computer Vision Group, Department of Informatics
University of Hamburg, Germany

Content

- From gradient to binary edges: The Canny detector

Fitting models to data points for detecting structures such as lines, circles, etc:

- Hough Transform
- RANSAC

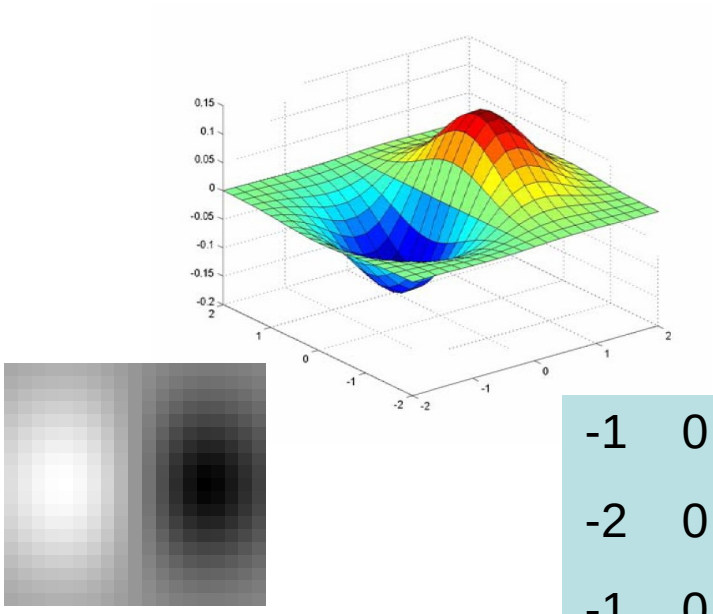
Gradients ⊂ Edges

[Gonzales/Woods]

Gradient magnitude from Filters

Spatial Domain

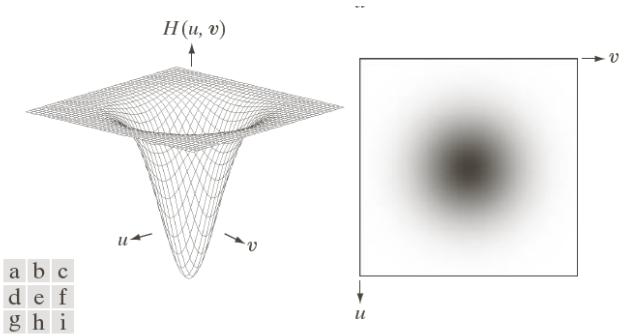
- Use filters: Sobel, Prewitt, etc
- Apply with convolution



-1	0	1
-2	0	2
-1	0	1

Frequency Domain

- Transform image to frequency domain
- Apply high-pass filters by manipulating the spectrum
- Transform spectrum back to spatial domain



Mostly, the spatial domain is used

- What we get from edge filters (here: Sobel):

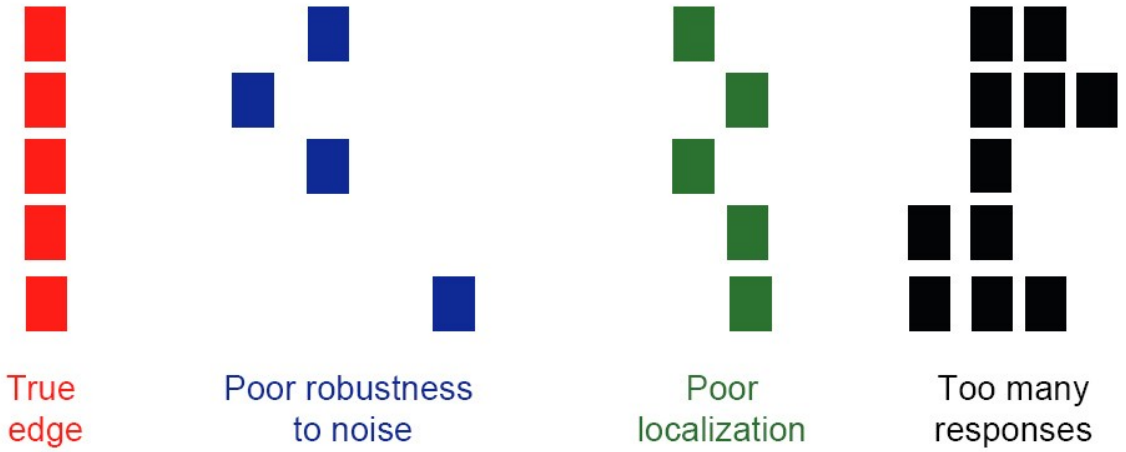
- We want a binary image: edge pixel or no edge pixel?
- Try thresholding:

[Gonzales/Woods]

Designing an Edge Detector

Criteria for an “optimal” edge detector:

- Low error rate: the optimal detector should 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 should be as close as possible to the true edges.
- Single response: the detector should return one point only for each true edge point; that is, minimize the number of local maxima around the true edge.



Canny Edge Detector

- **Canny edge detector:**
  by John F. Canny (1986)
- Probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise

John F. Canny

John Canny in his office at University of California, Berkeley (2013)

Nationality	Australian
Fields	Computer scientist
Institutions	Berkeley
Alma mater	Adelaide University MIT
Doctoral students	Ming C. Lin Dinesh Manocha
Known for	Canny edge detector

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

Canny Edge Detector

Algorithm of the Canny Edge Detector:

1. Smooth image and compute partial derivatives (e.g. apply Sobel)
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Hysteresis thresholding

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

[Source: D. Lowe, L. Fei-Fei]

The Canny Edge Detector

Original image

Apply Sobel filters:

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

to obtain the partial derivatives

$\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$

The Canny Edge Detector

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

Gradient magnitude

Canny Edge Detector

Algorithm of the Canny Edge Detector:

1. Smooth image and compute partial derivatives (e.g. apply Sobel)
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Hysteresis thresholding

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

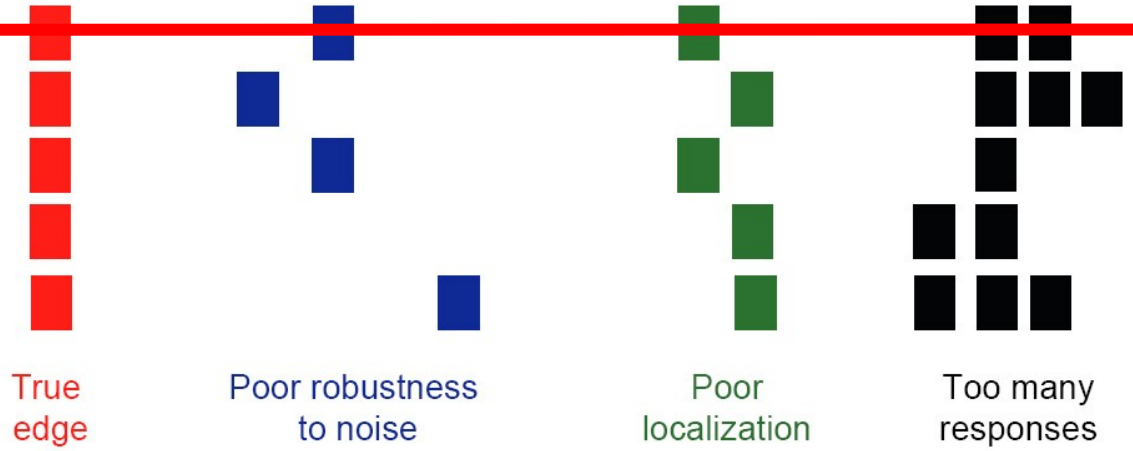
[Source: D. Lowe, L. Fei-Fei]

Remember

Designing an Edge Detector

Criteria for an “optimal” edge detector:

- Low error rate: the optimal detector should 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 should be as close as possible to the true edges.
- Single response: the detector should return one point only for each true edge point; that is, minimize the number of local maxima around the true edge.

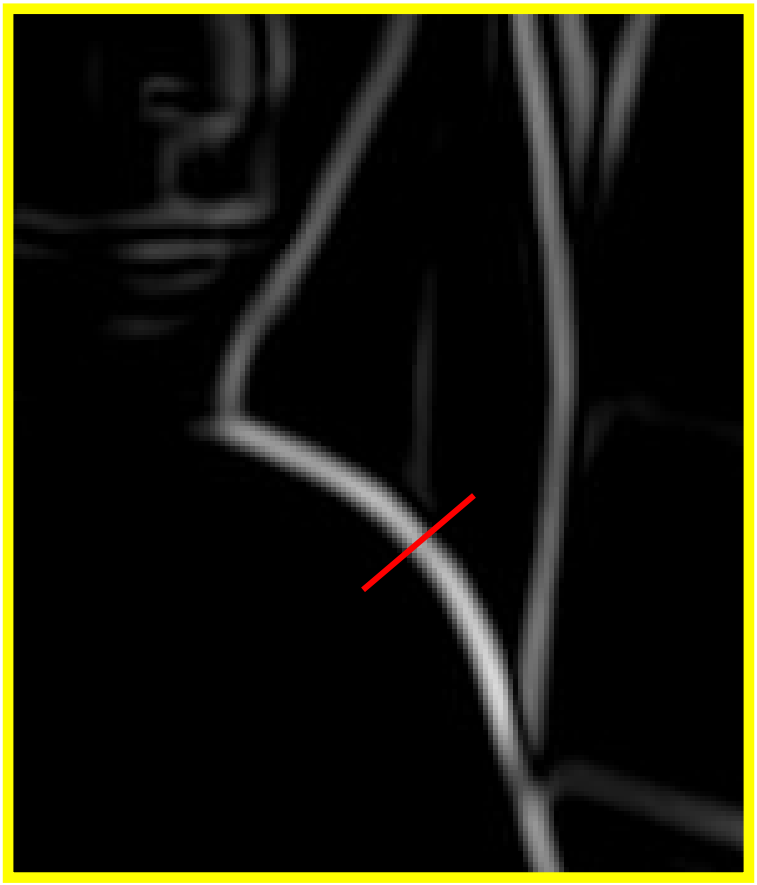


The Canny Edge Detector

We would like a one-pixel-wide curve instead this thick curve

Thresholding

The Canny Edge Detector

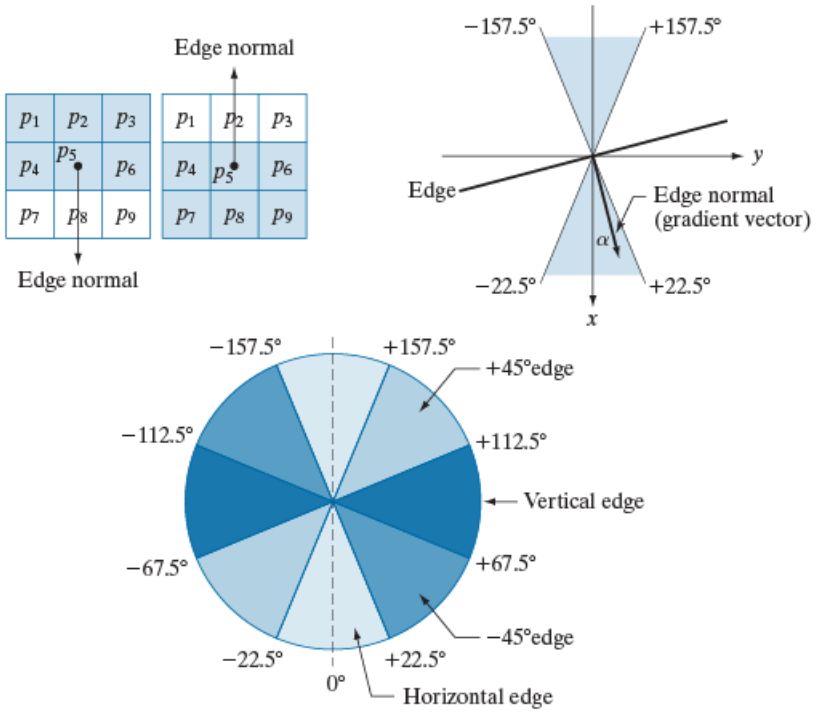


Check if pixel is local maximum along gradient direction, select single max across width of the edge

Thresholding

Non-maxima Suppression

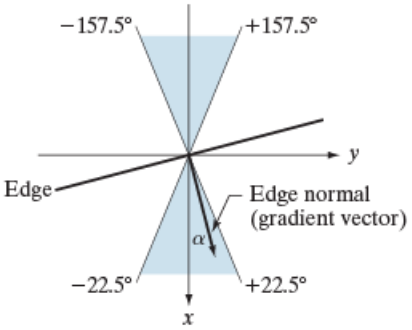
- Quantize gradient orientation into 4 bins (0°,45°,90°,135°)

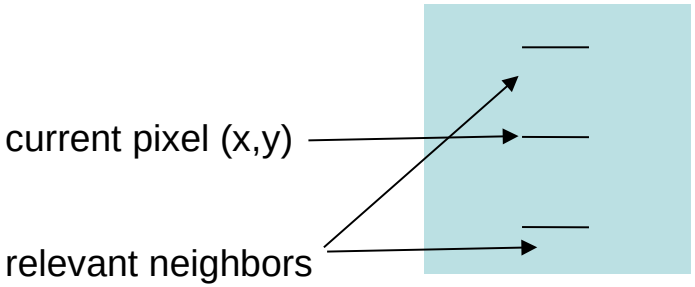


[Gonzales/Woods]

Non-maxima Suppression

- Quantize gradient orientation into 4 bins (0°,45°,90°,135°)
- Select current gradient orientation $\theta(x, y)$
- Compare magnitude of current pixel M(x,y) with magnitudes of neighbors in gradient direction:
  - M(x,y) < any neighbor ▭ set to 0
  - Otherwise ▭ keep value





Horizontal edge. Compare with fields above and below

Canny

- We have now: a grayscale image with thin edges
- Edges are still not binary values but contain the gradient magnitude value
- We want: a binary image (edge/non-edge)

- Simple approach: thresholding
- Problem:
  - If threshold too low: many false edges (false positives)
  - If threshold too high: valid edge points eliminated (false negatives)

- Solution: Hysteresis thresholding (with two thresholds)

Canny Edge Detector

Algorithm of the Canny Edge Detector:

1. Smooth image and compute partial derivatives (e.g. apply Sobel)
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Hysteresis thresholding

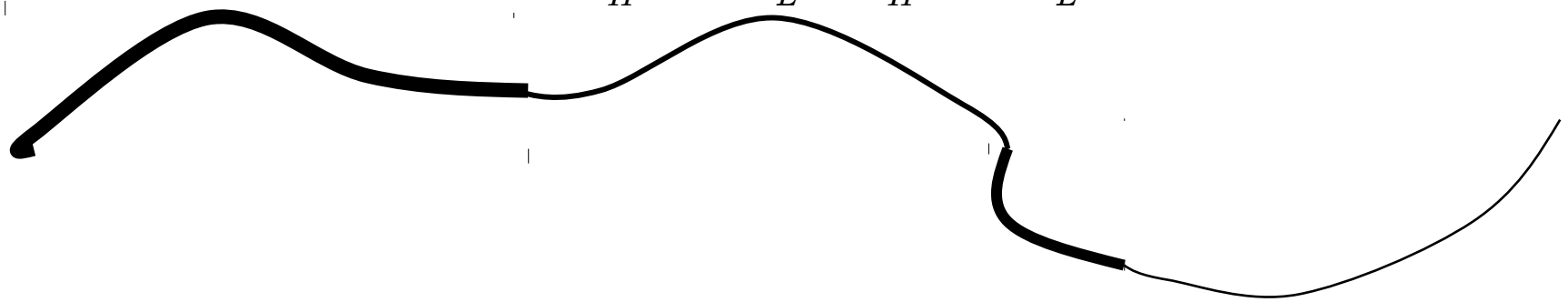
J. Canny, A Computational Approach To Edge Detection, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679-714, 1986.

[Source: D. Lowe, L. Fei-Fei]

Solution: Hysteresis Thresholding

- Hysteresis: the dependence of the state of a system on its history (here: dependence of pixel on its neighbors)
- Idea: Maintain two thresholds T_H and T_L
  - Use T_H to find strong edges to start edge chain
  - Use T_L to find weak edges which continue edge chain
- Typical ratio of thresholds:

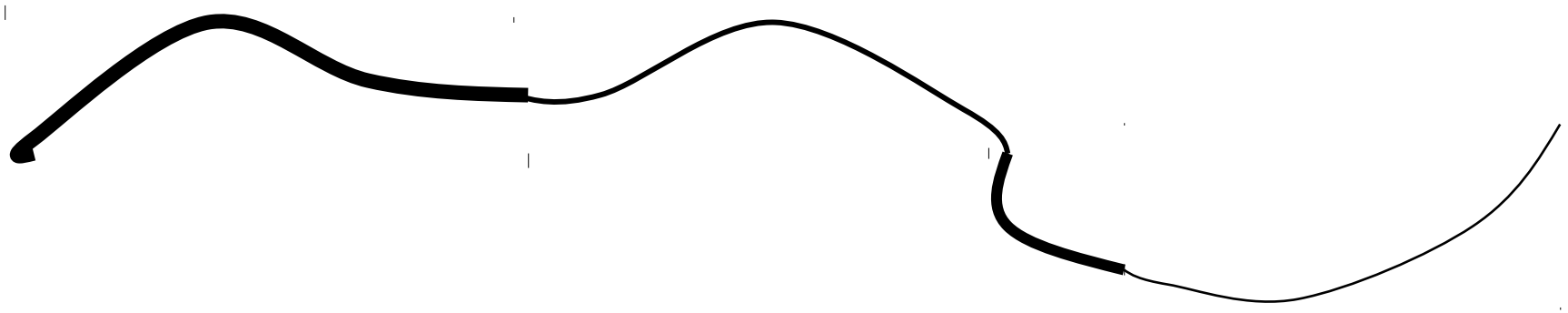
$$T_H = 2T_L or T_H = 3T_L$$



Hysteresis Thresholding

- Strong edges: $g_H(x,y)$ (edges with value > T_H)
- Weak edges: $g_L(x,y)$ (edges with value > T_L but < T_H)

- Edges in the final map:
  - All strong edge points from $g_H(x,y)$
  - and weak edge points from $g_L(x,y)$ that are adjacent to a strong edge or a previously accepted weak edge point (repeat)

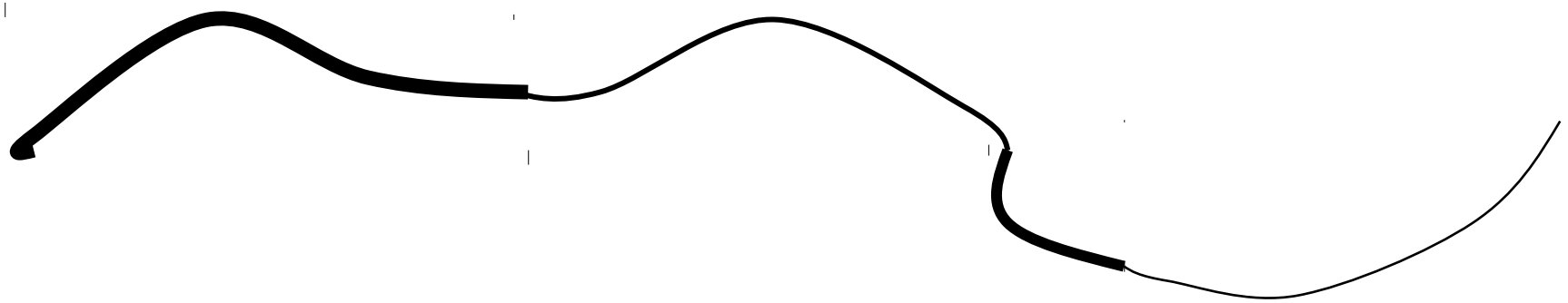


[Gonzalez/Woods]

Hysteresis Thresholding

Hysteresis thresholding algorithm:

- Define low and high threshold
- Mark all strong edge pixels as valid edge pixels
- For each unvisited valid edge pixel p
  - For all neighbors n of p (given e.g. an 8-neighborhood)
    - If n is weak edge pixel: mark n as valid edge pixel



Hysteresis Thresholding

Original image

High threshold
(strong edges)

Low threshold
(weak edges)

courtesy of G. Loy

Hysteresis threshold

Source: L. Fei-Fei

Canny Edge Detector

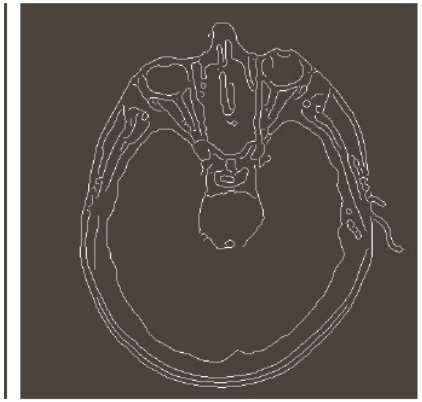
Algorithm of the Canny Edge Detector:

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei

Results Canny



Object Boundaries vs. Edges

Be careful:
edges do not always correspond to object boundaries

Background

Texture

Shadows

Slide adapted from Kristen Grauman

From edges to lines

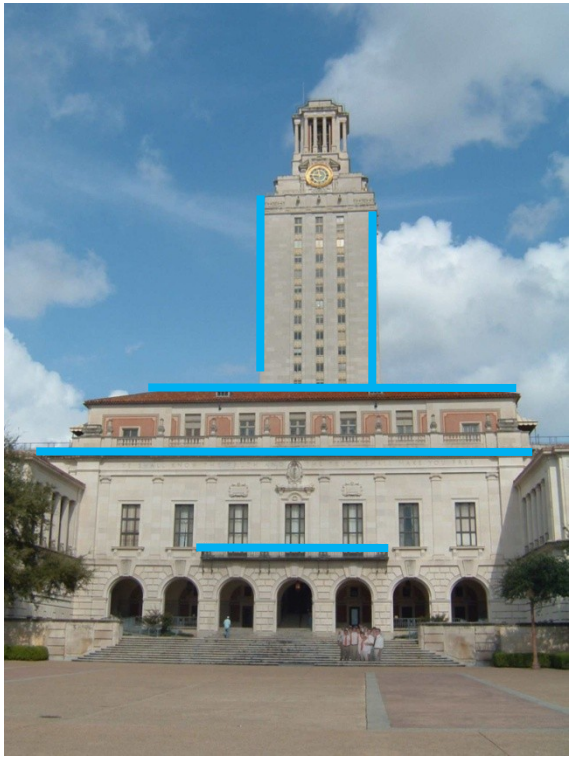
Canny gives us edge pieces (*edgels*) that are

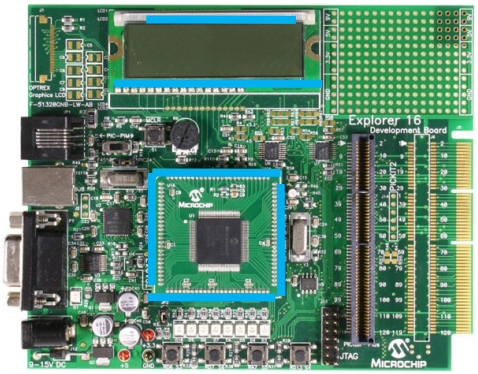
- thin
- longer than the pieces from gradient magnitude computation
- less noisy

Let’s see how we can use these edgels to detect straight lines

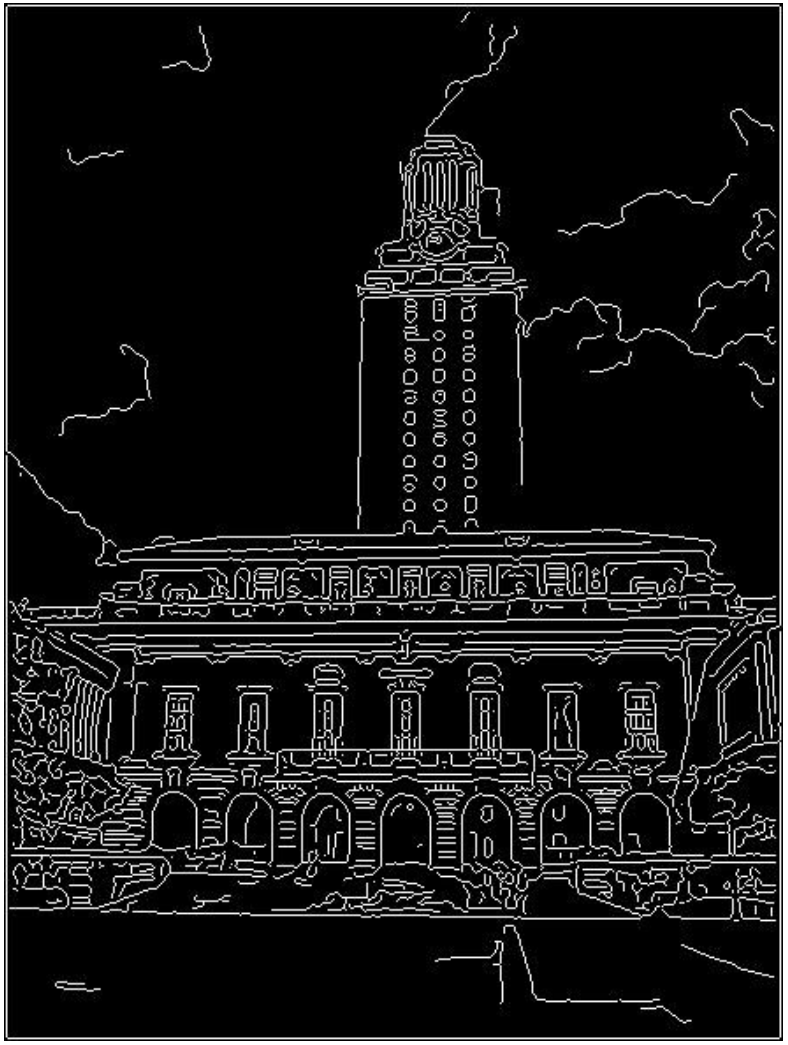
Lines

- Why fit lines?
  - Many objects are characterized by presence of straight lines





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?

Hough Transform

- Invented by Hough in 1959 (patented in 1962)

United States Patent Office

3,069,654
Patented Dec. 18, 1962

1

3,069,654
METHOD AND MEANS FOR RECOGNIZING COMPLEX PATTERNS
Paul V. C. Hough, Ann Arbor, Mich., assignor to the United States of America as represented by the United States Atomic Energy Commission
Filed Mar. 25, 1960, Ser. No. 17,715
6 Claims. (Cl. 340—146.3)

2

of the point on the line segment from the horizontal midline 109 of the frameiet 108.

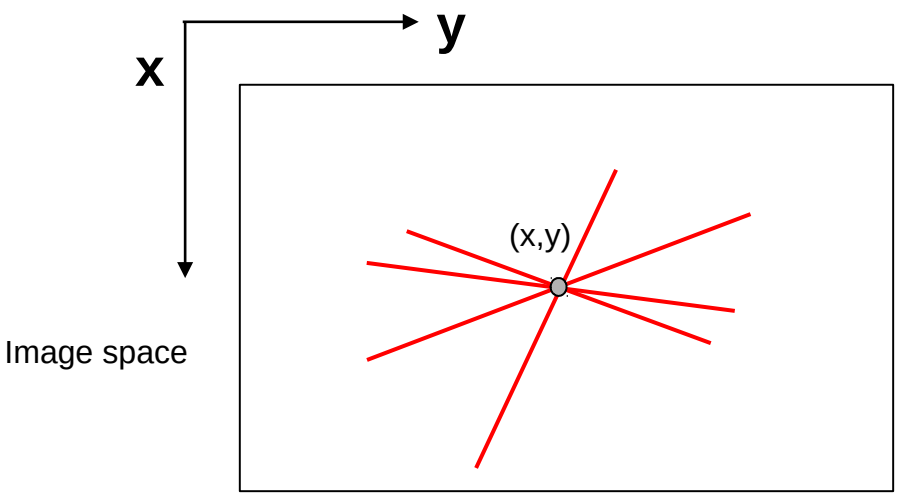
(3) Each line in the transformed plane is made to have an intercept with the horizontal midline 101 of the picture 100 equal to the horizontal coordinate of its respective point on the line segment in frameiet 108.

Thus, for a given reference point 110 on line segment 102 a line 110A is drawn in the plane transform 102A. The reference point 110 is approximately midway between

- Generalized by Duda and Hart in 1972
- The Hough Transform is a method to test whether a set of pixels lies on a specified shape (lines, circles, etc.)
- Idea: pixels vote for plausible models
- Finds also imperfect shapes

Hough Transform

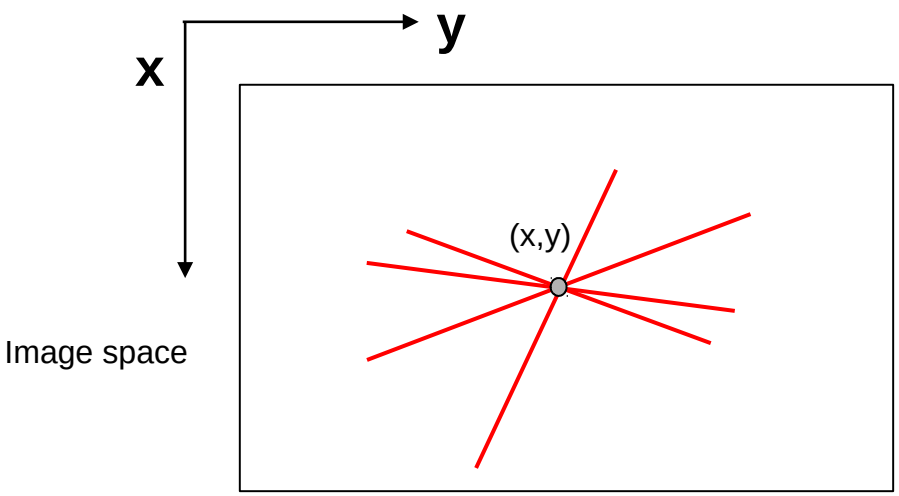
- Idea: pixels vote for plausible models
- Given: a pixel (x,y) (candidate for our model, can be e.g. an edge pixel from Canny)
- Vote for every possible line going through (x,y)

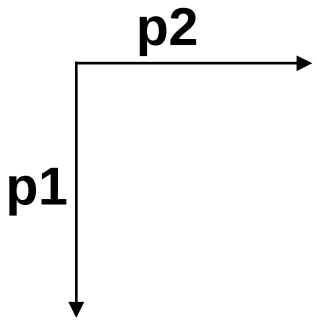


- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of “good” features.

Hough Transform

- Idea: pixels vote for plausible models
- Given: a pixel (x,y) (candidate for our model, can be e.g. an edge pixel from Canny)
- Vote for every possible line going through (x,y)



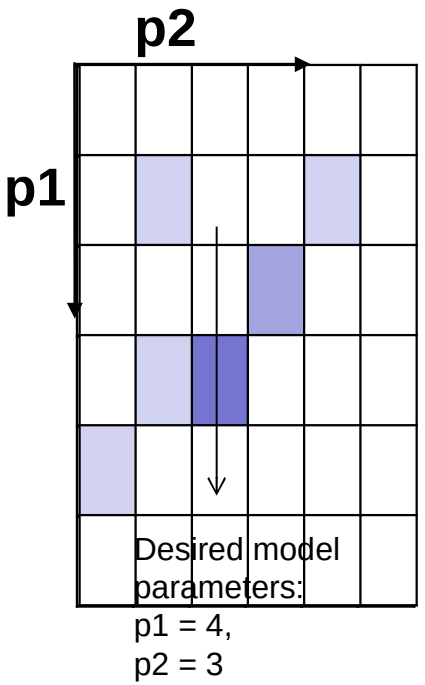


Hough space, spanned by parameters p1 and p2

- Voting takes place in a parameter space called *Hough space*

Hough Transform

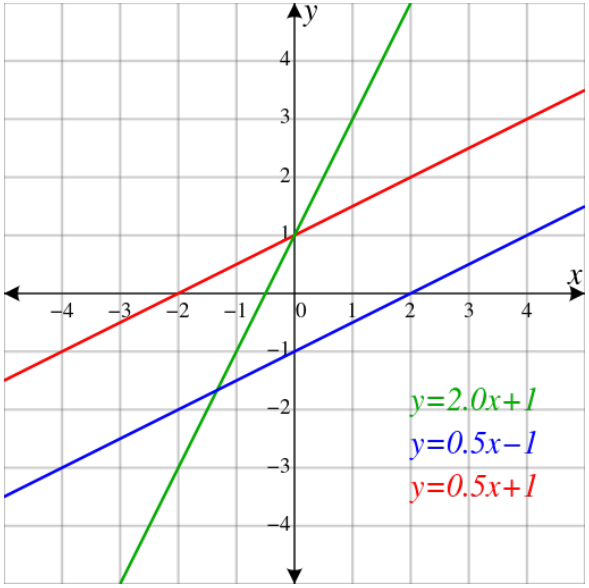
- Hough Transform Algorithm, first sketch:
  - Create binary edge image (e.g. with Canny)
  - Determine parameters that represent desired model (e.g. a line) and span parameter space (Hough space) by these parameters
  - Determine bins that subdivide Hough space into *accumulator cells*
  - For each edge point:
    - Determine all possible model parameters and increment corresponding cells by 1
  - Look for peaks in the Hough space: the parameters of the peak cells are the parameters of the desired model (e.g. line)



Line Representation

- We can represent a line by $y = m_0 x + b_0$ (slope-intercept form)
- Which lines cannot be represented with this form?

Image space

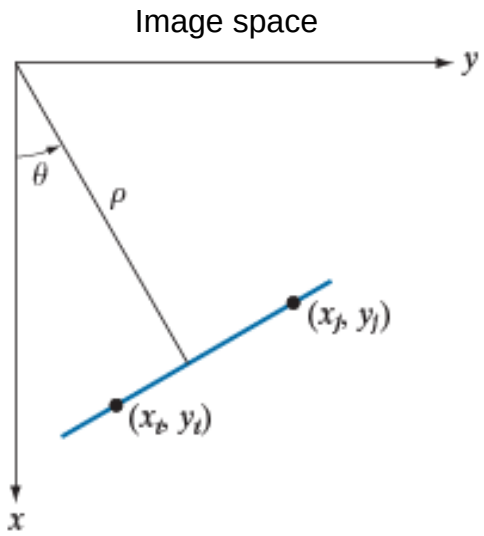


- Undefined for vertical lines.

[Wikipedia: Line (geometry)]

Line Representation

Instead represent lines by the Hesse normal form:



$$x\cos\theta + y\sin\theta = \rho$$

ρ : perpendicular distance from line to origin

θ : angle of perpendicular to x-axis

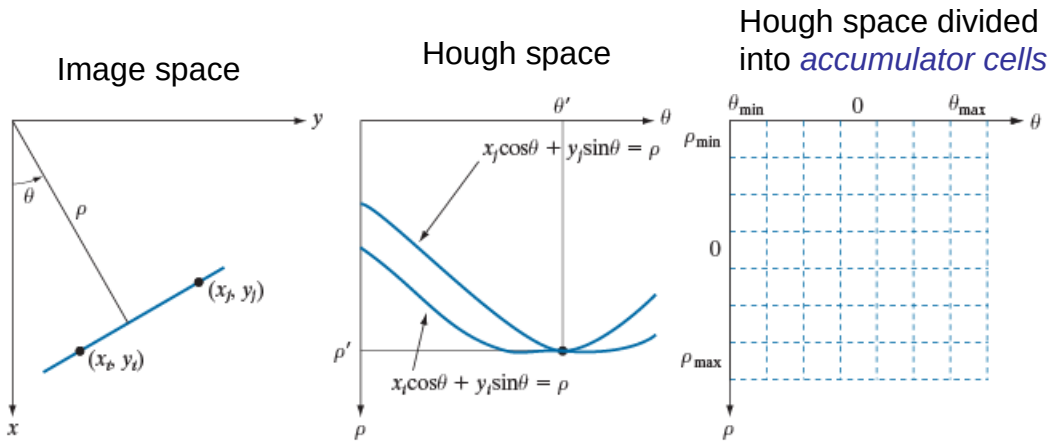
The parameters ρ and θ span our Hough space

[Gonzales/Woods]

Hough Space

Hough space: a parameter space that represents structures (e.g. lines) based on their parameters

For lines: the parameters ρ and θ span our Hough space



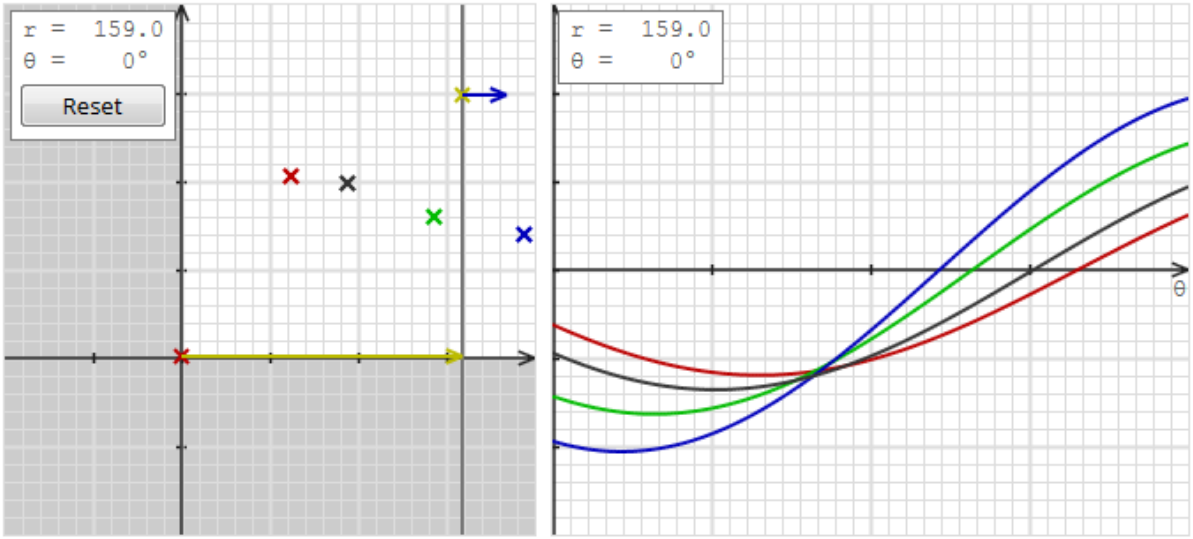
All lines going through a point (x,y) lie on a sinusoidal curve in Hough space. The intersection denotes the desired line that passes through both (several) points

[Gonzalez/Woods]

Hough Transform

Interactive Demo at:

http://matlabtricks.com/post-39/understanding-the-hough-transform



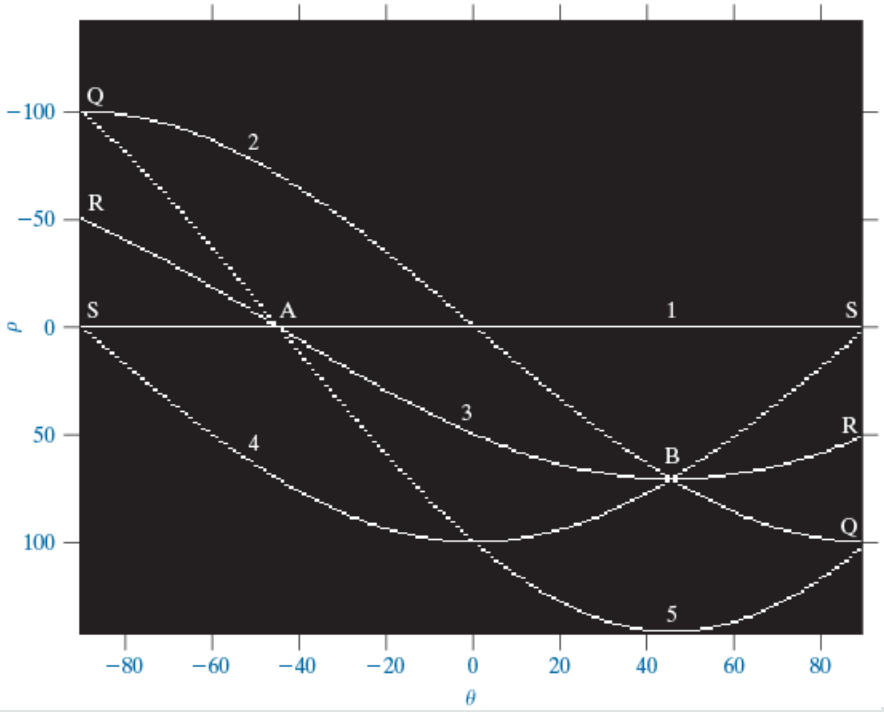
- Click on several points that are on a line and see what happens
- Click on a point that is not on this line

Hough Transform

Another example:

Image with 5 white points

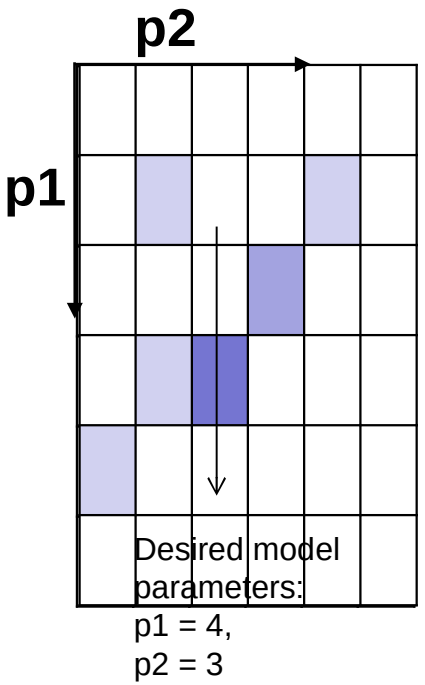
Corresponding Hough space for lines
Note intersection points A and B.
What do they tell us?



[Gonzales/Woods]

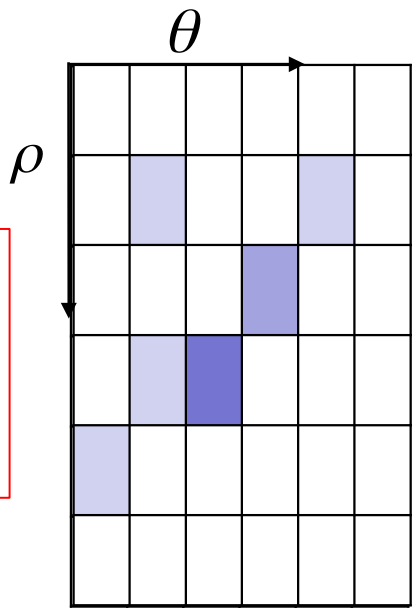
Hough Transform

- Hough Transform Algorithm, first sketch:
  - Create binary edge image (e.g. with Canny)
  - Determine parameters that represent desired model (e.g. a line) and span parameter space (Hough space) by these parameters
  - Determine bins that subdivide Hough space into *accumulator cells*
  - For each edge point:
    - Determine all possible model parameters and increment corresponding cells by 1
  - Look for peaks in the Hough space: the parameters of the peak cells are the parameters of the desired model (e.g. line)



Hough Transform

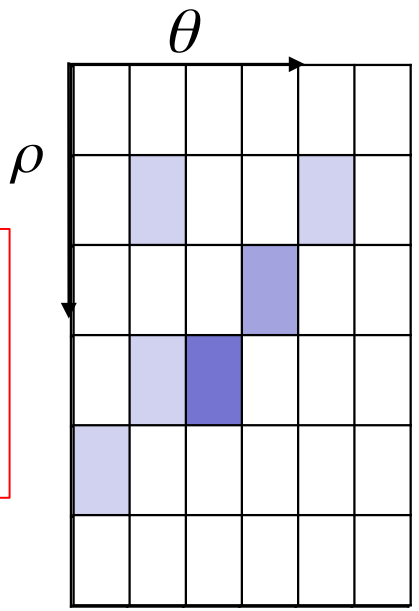
- Hough Transform Algorithm for lines:
  - Create binary edge image (e.g. with Canny)
  - Span parameter space (Hough space) by parameters ρ and θ
  - Determine bins that subdivide Hough space into *accumulator cells*
  - For each edge point (x,y):
    - For each possible orientation θ:
      - Compute the corresponding $\rho = x\cos\theta + y\sin\theta$
      - Increment the corresponding grid cell: H[θ,ρ]
  - Look for peaks in the Hough space: the parameters of the peak cells are the parameters of the desired line
  - Peak at H[θ,ρ]: output corresponding line $\rho = x\cos\theta + y\sin\theta$

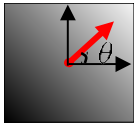


Hough Transform

- Hough Transform Algorithm for lines:
  - Create binary edge image (e.g. with Canny)
  - Span [illegible] parameters
  - Determine bins that subdivide Hough space into *accumulator cells*
  - For each edge point (x,y):
    - For each possible orientation θ:
      - Compute the corresponding $\rho = x\cos\theta + y\sin\theta$
      - Increment the corresponding grid cell: H[θ,ρ]
  - Look for peaks in the Hough space: the parameters of the peak cells are the parameters of the desired line
  - Peak at H[θ,ρ]: output corresponding line $\rho = x\cos\theta + y\sin\theta$

Do we really have to test all ρ possible orientations?



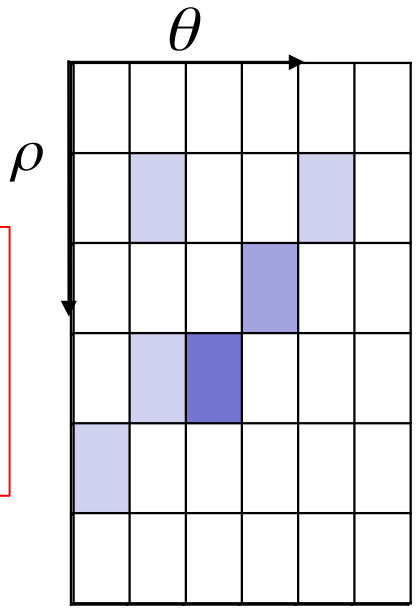


$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$

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

Hough Transform Extension

- Hough Transform Algorithm for lines:
  - Create binary edge image (e.g. with Canny)
  - Span parameter space (Hough space) by parameters ρ and θ
  - Determine bins that subdivide Hough space into *accumulator cells*
  - For each edge point (x,y):
    - ~~For each possible orientation θ:~~ θ = gradient at (x,y)
      - Compute the corresponding $\rho = x\cos\theta + y\sin\theta$
      - Increment the corresponding grid cell: H[θ,ρ]
  - Look for peaks in the Hough space: the parameters of the peak cells are the parameters of the desired line
  - Peak at H[θ,ρ]: output corresponding line $\rho = x\cos\theta + y\sin\theta$



Hough Transform Extension

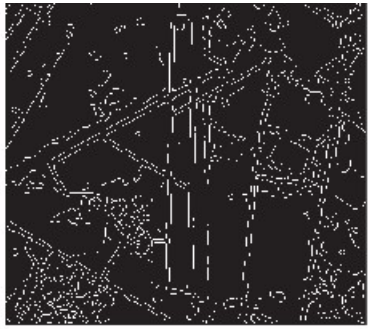
Extension 2: Give more votes for stronger edges
(use magnitude of gradient)

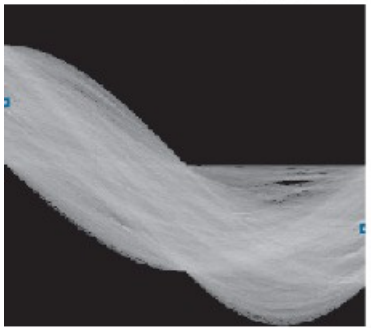
Example

Aerial image of an airport:

Image

Canny edge map





Hough space (blue boxes: vertical lines)

The two vertical lines

The two vertical lines

[Gonzales/Woods]

From lines to line segments

- The Hough transform gives us infinite lines
- To get line segments:
  - Look for edge support in Canny edge map or in gradient image
  - Determine distance between disconnected pixels
  - If gap between points is smaller than a threshold, bridge gap
  - Otherwise: disconnect line

[Gonzales/Woods]

Example: HT for Straight Lines

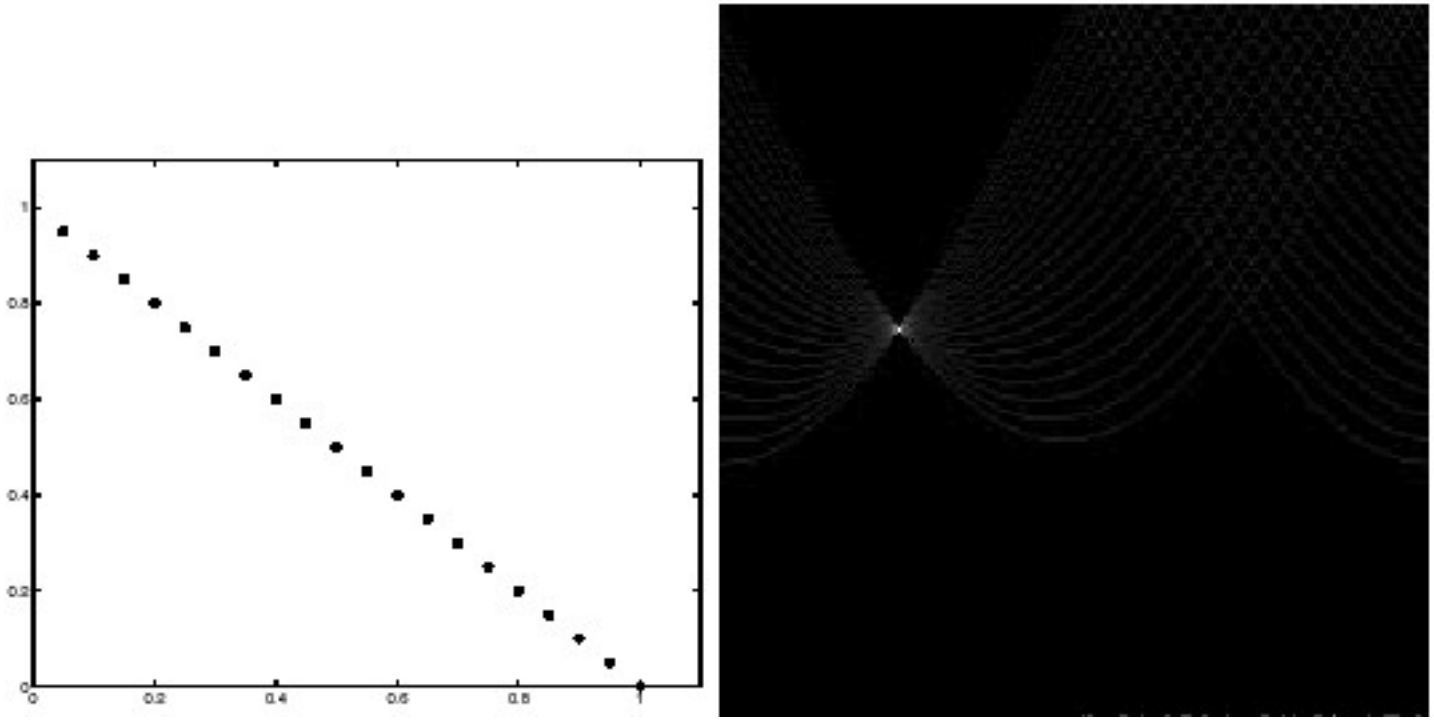
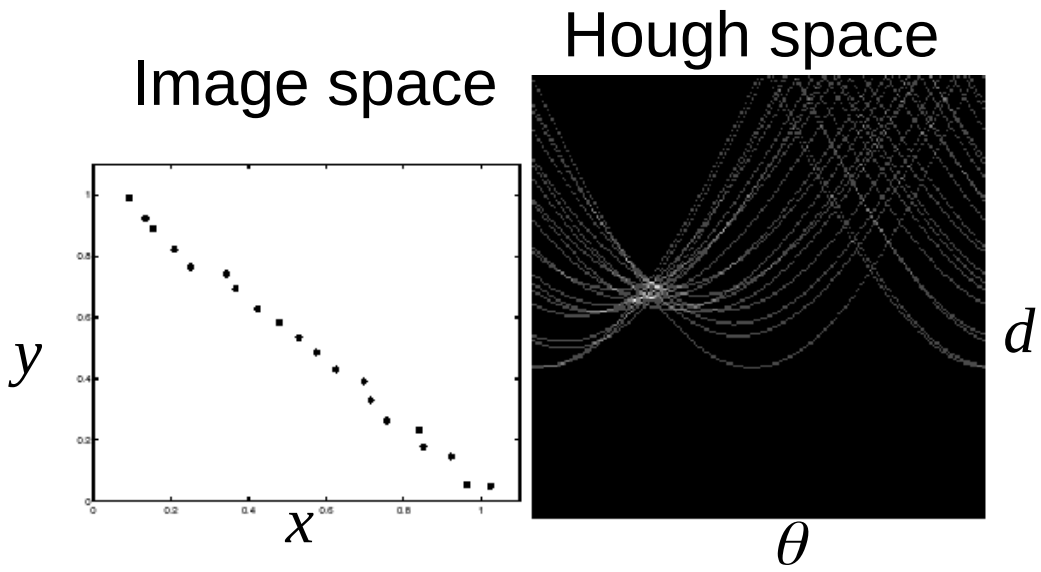


Image space
edge coordinates

Votes

Bright value = high vote count
Black = no votes

Impact of Noise on Hough Transform



Noise has effect that lines do not cross at single point

Q: how could we deal with that?

A: make bins larger

Q: What problems does this cause?

A1: We might get false positives (points that do not belong to the same line are grouped)

A2: we lose precision (how exactly is the line oriented??)

Impact of Noise on Hough Transform

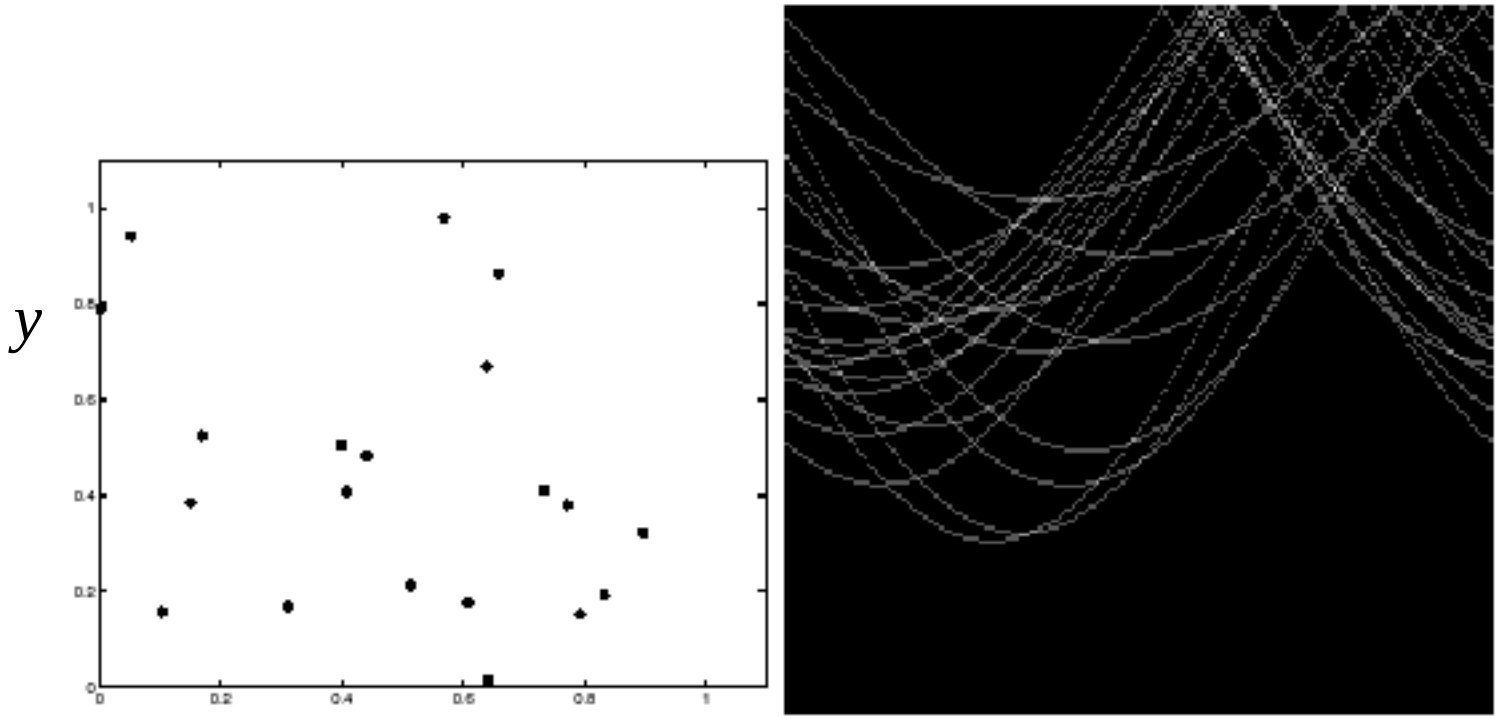


Image space
edge coordinates

Votes

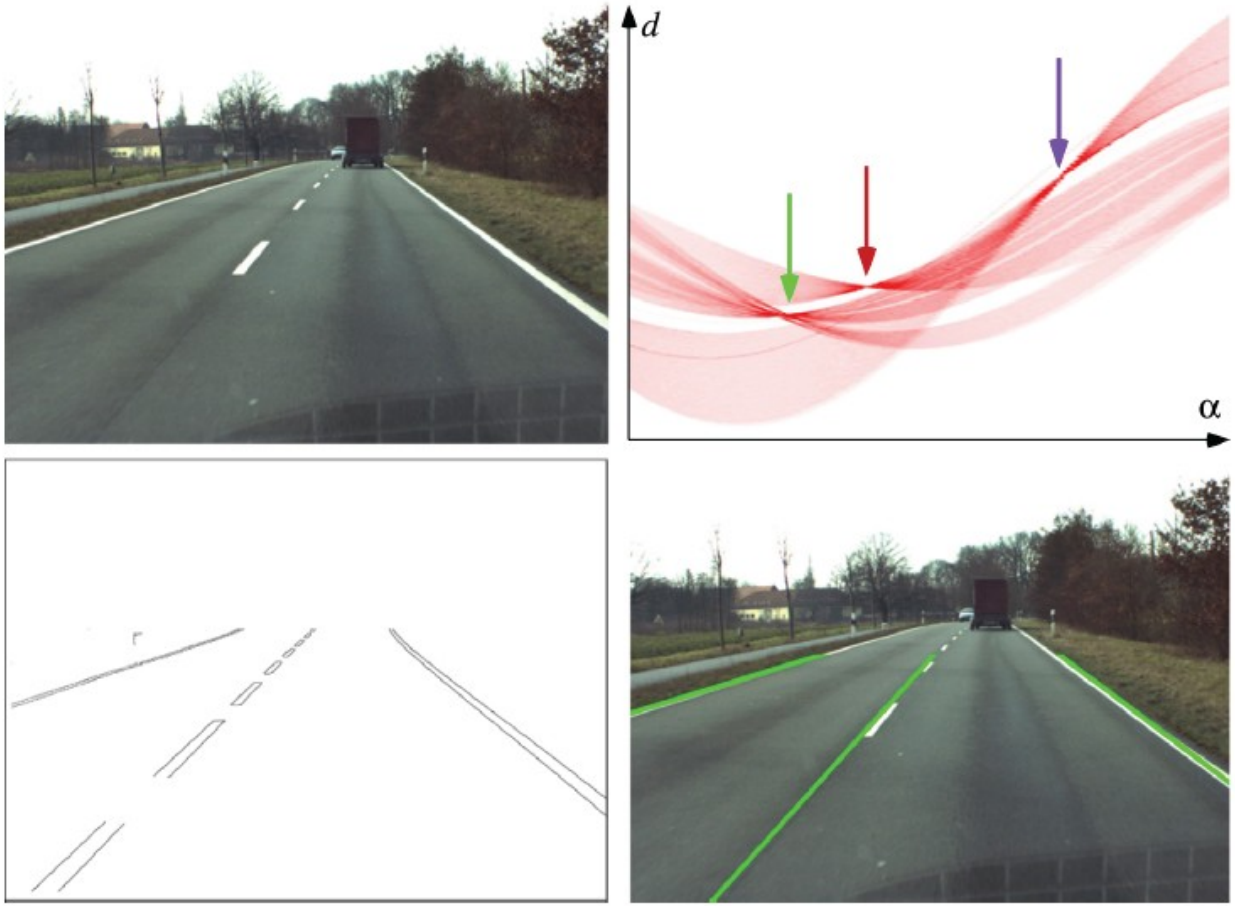
Here, everything appears to be “noise”, or random edge points, but we still see peaks in the Hough space.

Example

Visualizing Hough Transform: Saburo Okita

https://www.youtube.com/watch?v=ykl8jTsuJKI

Example: Lane detection



Analysis of Hough space *top*, *right* for detected edge points (*bottom*, *left*) of input image (*top*, *left*) leads to three line segments

[Klette 2014]

Example: Lane detection

Lane departure using Hough Line Transform: Gustaf Sundberg

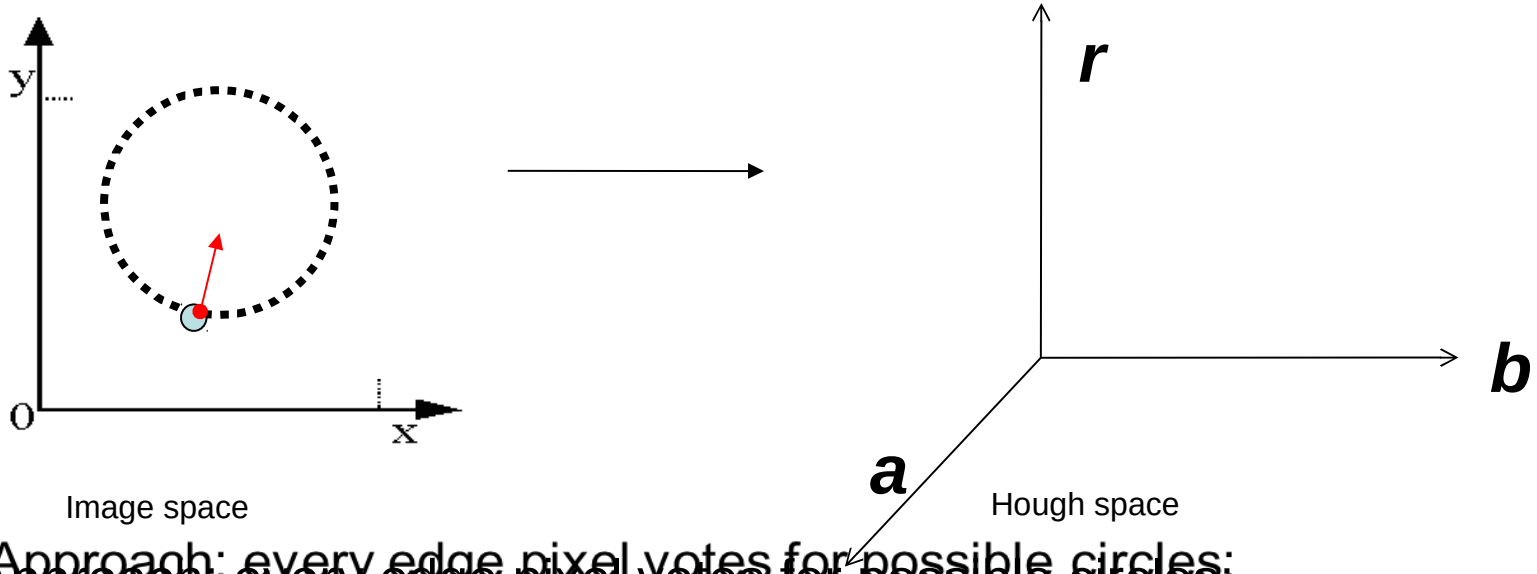
https://www.youtube.com/watch?v=IE2SIzDdFM0

Hough Transform for Circles

Circle: Which parameters do we have?

center (a, b) and radius r: $(x_i - a)^2 + (y_i - b)^2 = r^2$

We have a 3D parameter space



- Approach: every edge pixel votes for possible circles:
  Loop over (a, b) pairs, compute r
- Can also be improved by regarding gradient at (x,y)

Hough Transform for Circles

Circle: Which parameters do we have?

center (a, b) and radius r: $(x_i - a)^2 + (y_i - b)^2 = r^2$

We have a 3D parameter space

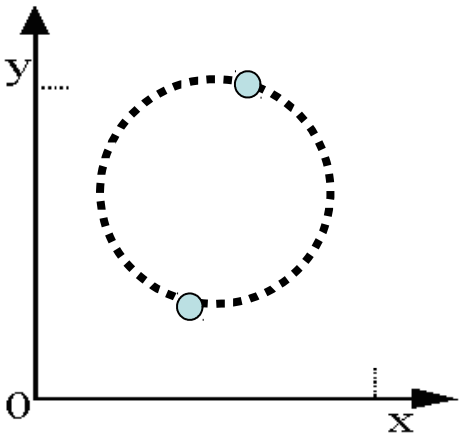
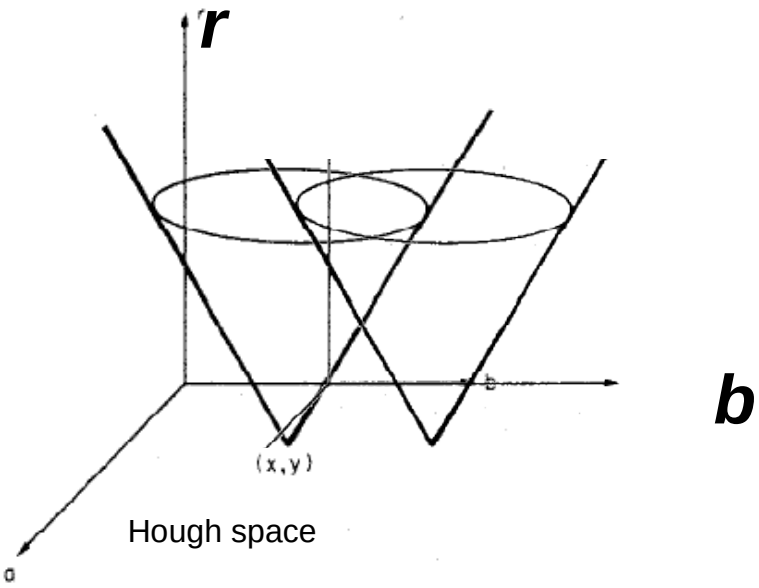


Image space



Hough space

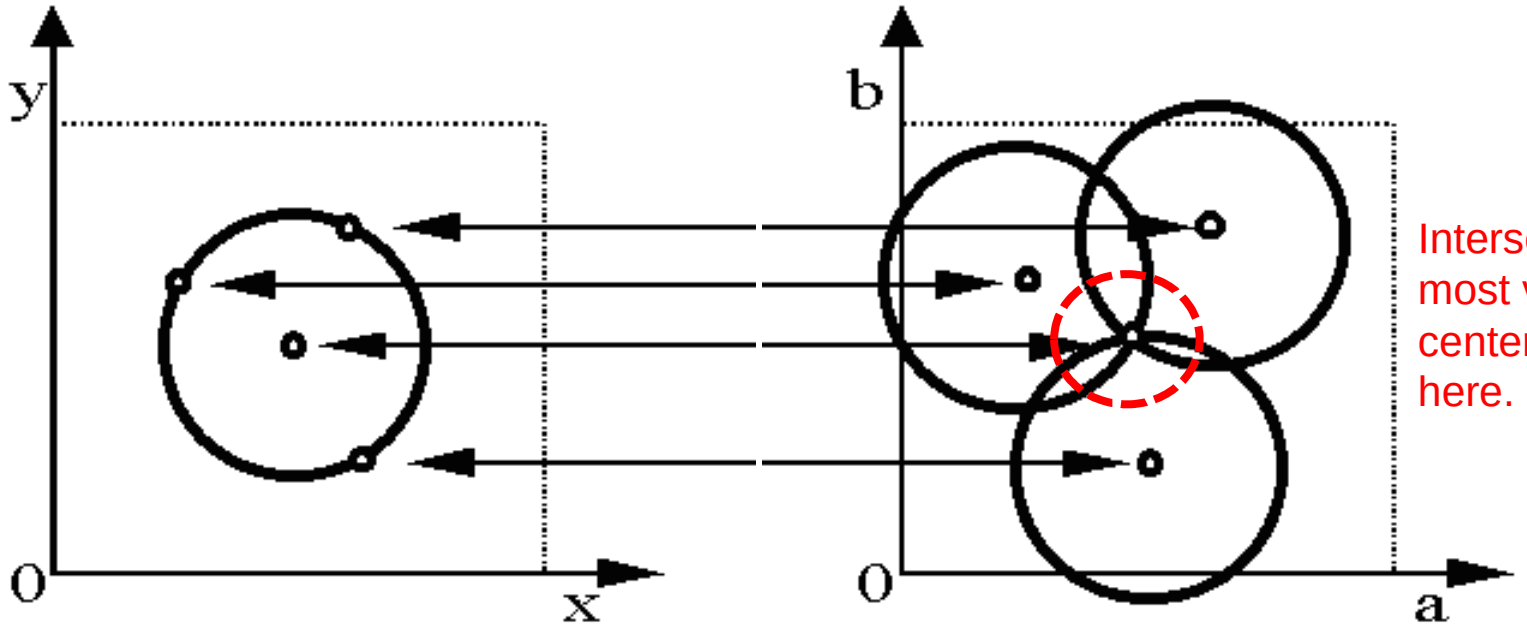
For a given edge point, the possible circles form a cone in Hough space

Hough Transform for Circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

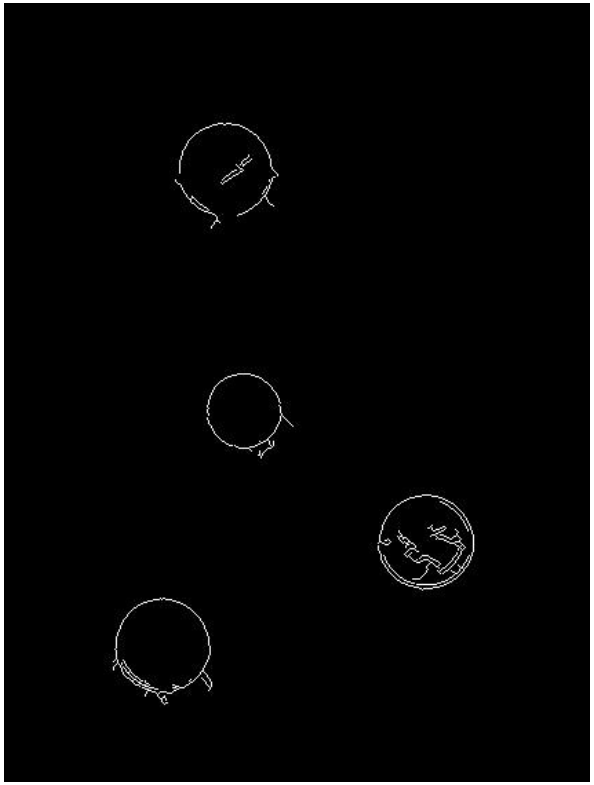
- For a fixed radius r, unknown gradient direction



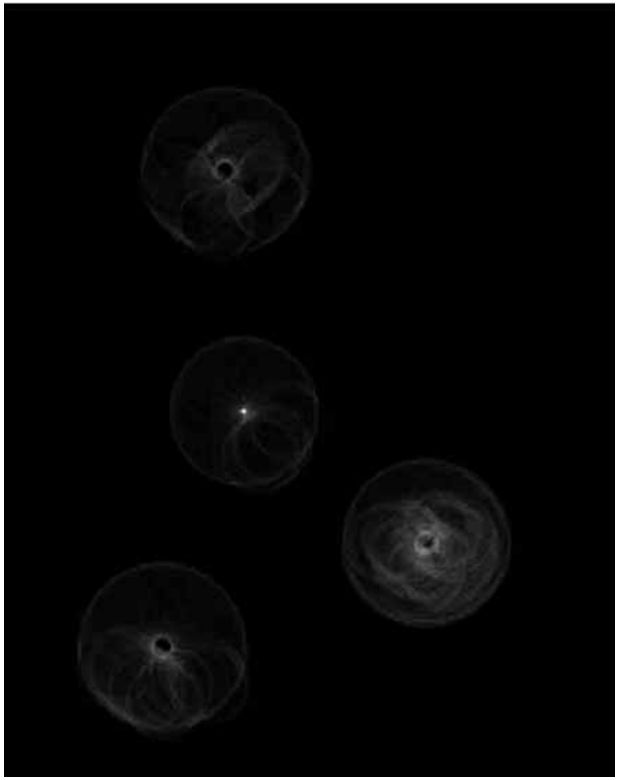
Example: Detecting Circles with Hough Transform

Original

Edges



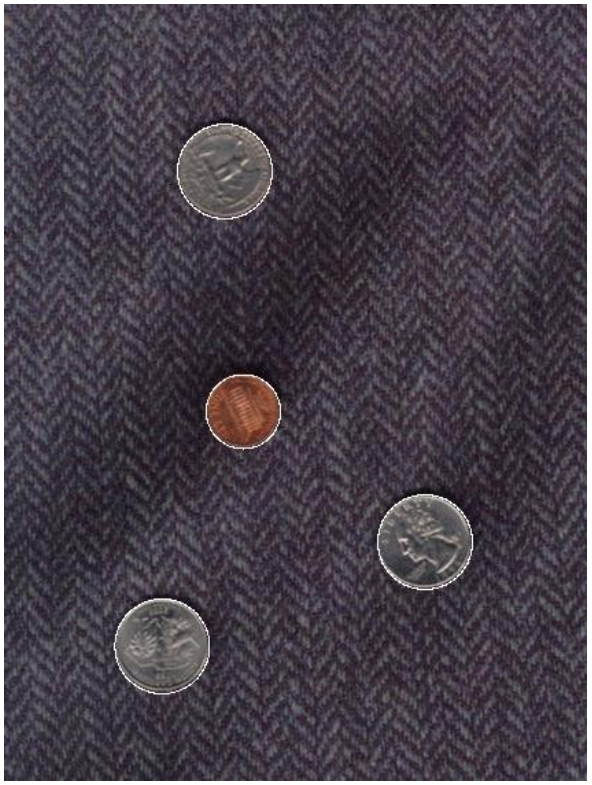
Votes: Penny



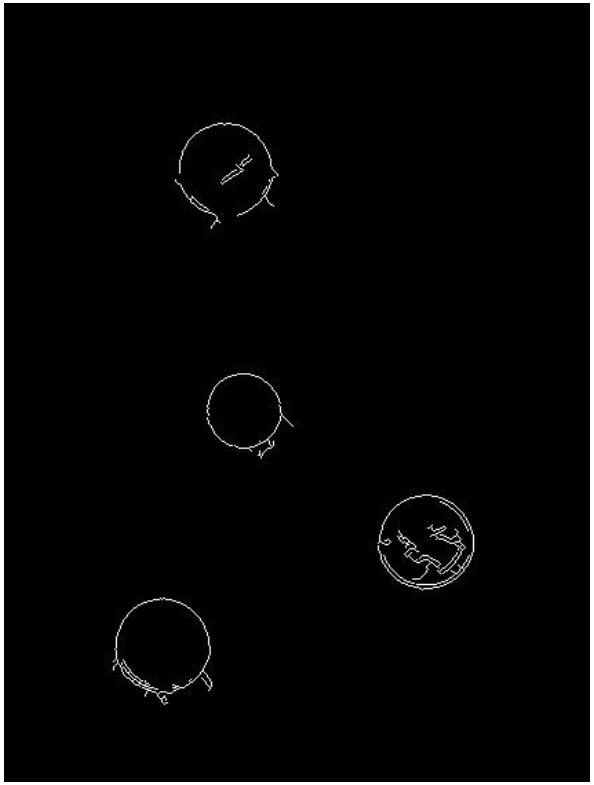
[Coin finding sample images from: Vivek Kwatra]

Example: Detecting Circles with Hough Transform

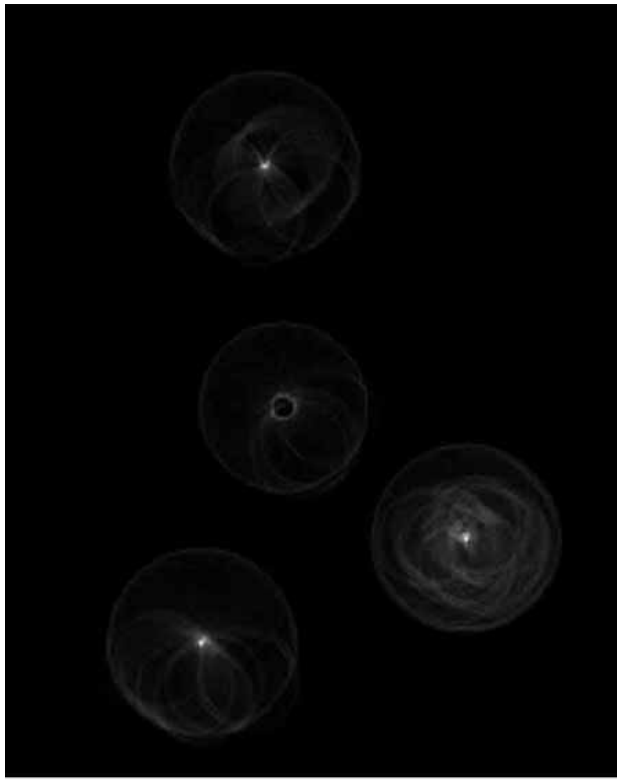
Combined detections



Edges



Votes: Quarter

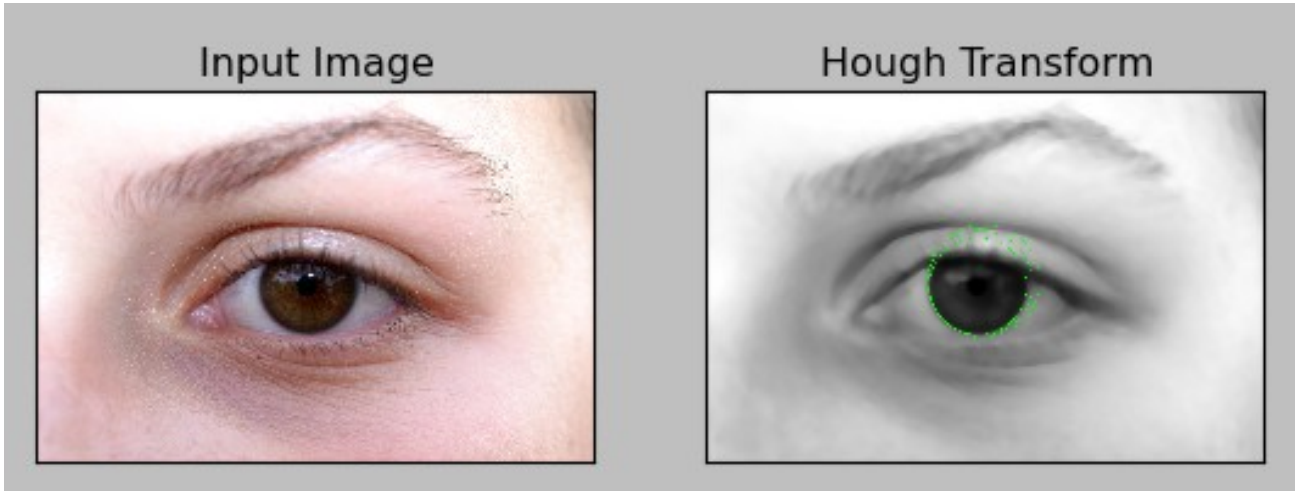


[Coin finding sample images from: Vivek Kwatra]

Example: Detecting Circles with Hough Transform

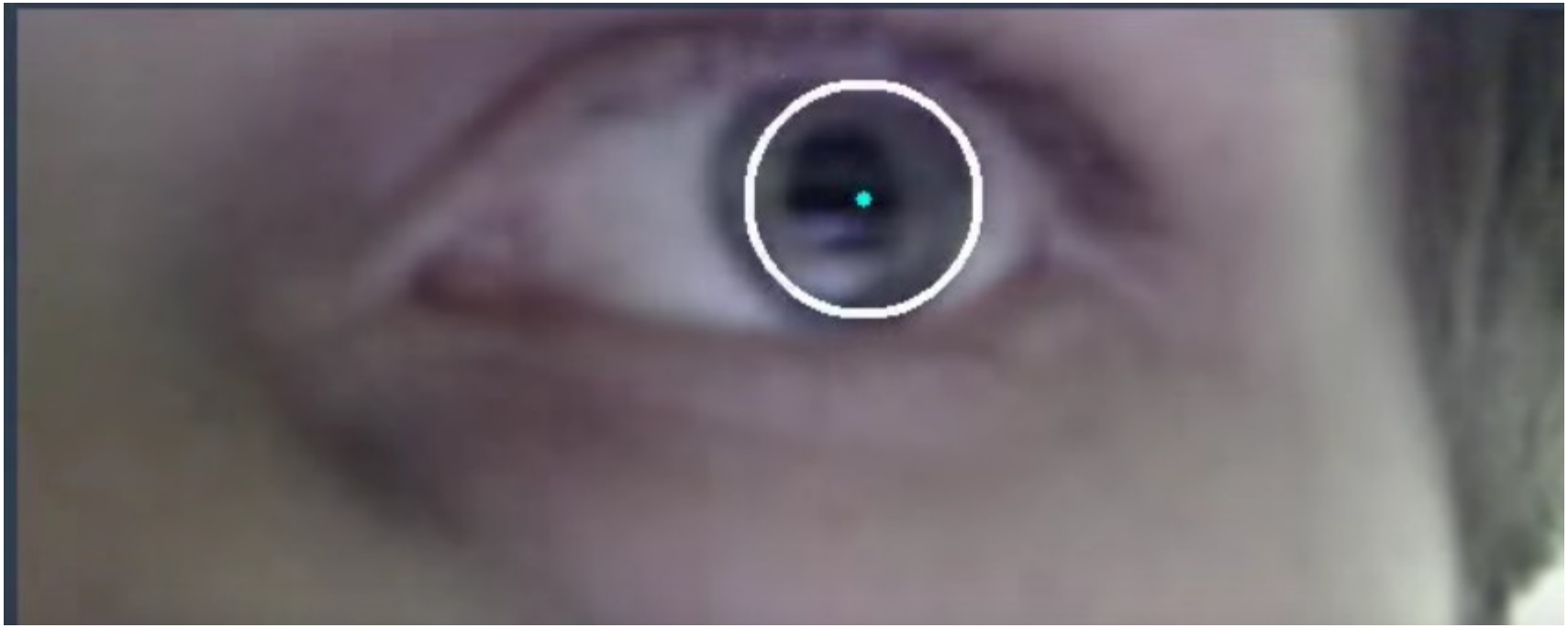
https://www.youtube.com/watch?v=vn9y7t9iqC8

Example: Iris detection



[http://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Hough%20Circle_Transform.php]

Example: Iris detection



Eye tracking using circle Hough transform: Paweł Gawliczek

https://www.youtube.com/watch?v=tNI9nW-aGOg

Example: Traffic Sign Detection

Fig: Detected Circle after applying CHT

Fig: Detected Ellipse after applying Ellipse Detection

[Ravaso Patil]

Example: Traffic Sign Detection

realtime road sign detection using hough transform (Hochschule-Rhein-Main Wiesbaden): MrMarkovicho

https://www.youtube.com/watch?v=szFEpCLUMYE

Voting: Practical Tips

- Minimize irrelevant elements first (take edge points with significant gradient magnitude)
- Choose a good grid / discretization
  - Too coarse: too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- Vote for neighbors also (smoothing in accumulator array)
- Utilize direction of edge (gradient) to reduce free parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes.

Hough Transform: Pros and Cons

Pros

- All points are processed independently, so can cope with occlusion
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Can detect multiple instances of a model in a single pass

Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: hard to pick a good grid size

Generalized Hough Transform

- The Generalized Hough Transform (GHT) detects arbitrary shapes by template matching
- Model is represented by a *reference point* ***a*** (e.g. center of mass) and displacement vectors of each edge point to ***a***

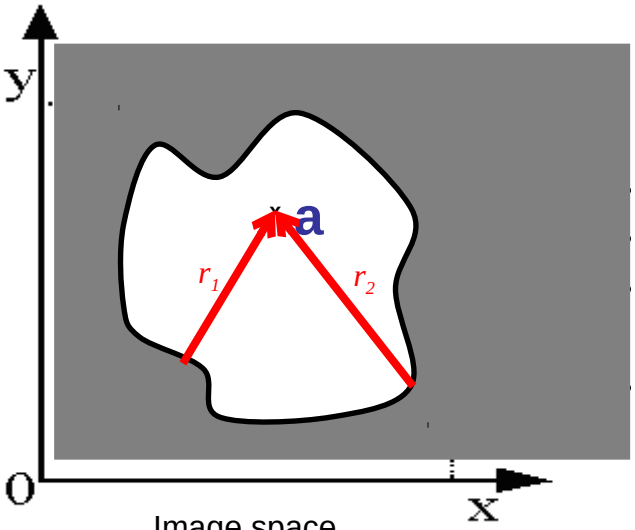


Image space

[Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980]


Dana H. Ballard

[Image left: Kristen Grauman]

Generalized Hough Transform

Store the model shape:

- At each boundary point, compute displacement vector: $r_i = a - p_i$.
- For a given model shape: store these vectors in a table indexed by gradient orientation θ.

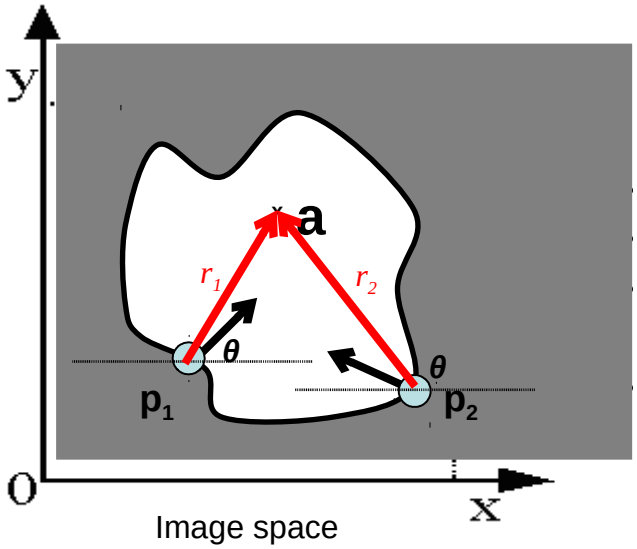


Image space

Generalized Hough Transform

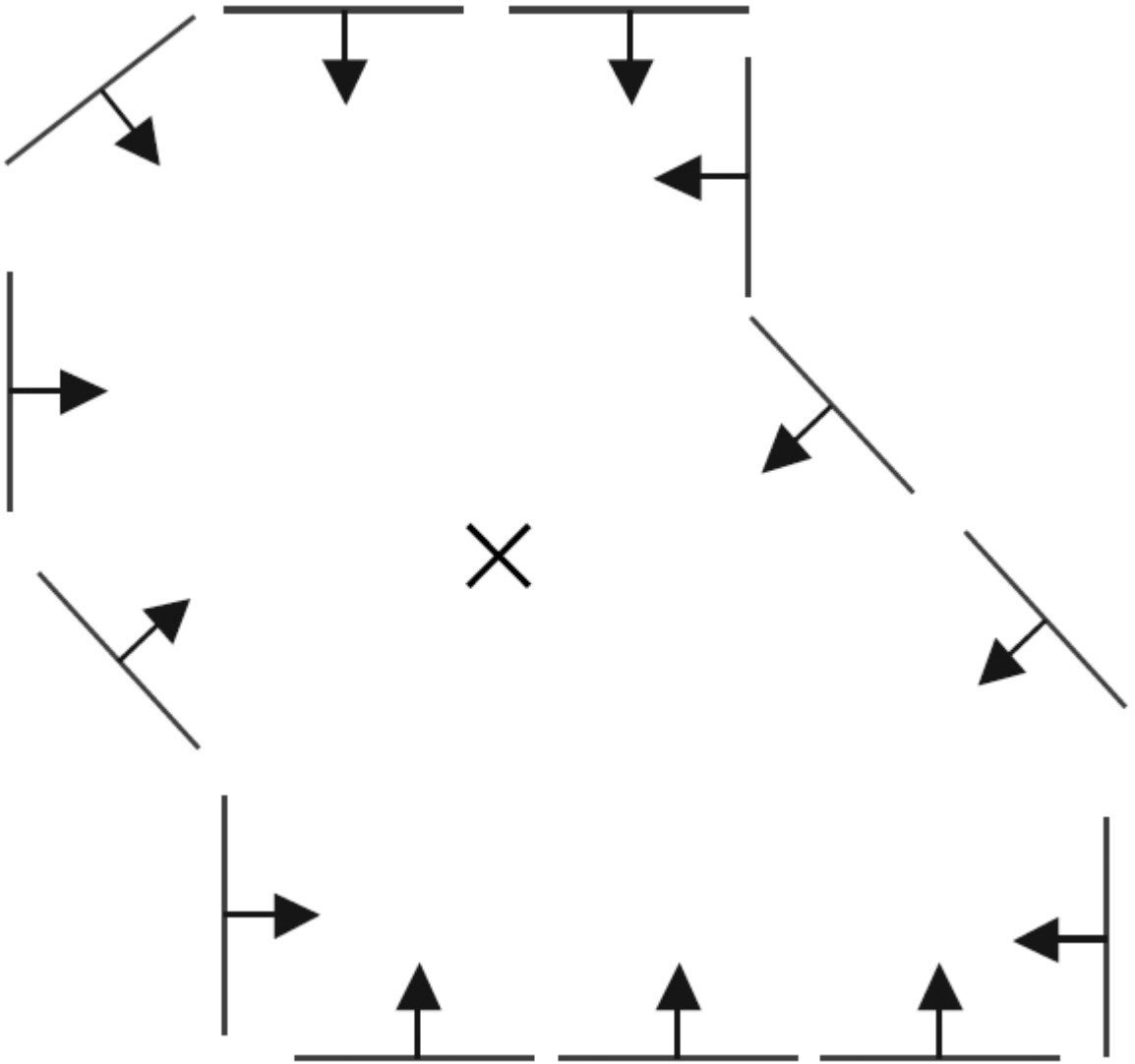
To *detect* the model shape in a new image:

- For each edge point
  - compute gradient orientation θ
  - retrieve displacement vector(s) r from table
  - compute position of reference point(s) and vote for reference point
- Peak in Hough space is reference point with most supporting edges

- *This assumes translation is the only transformation, i.e., orientation and scale are fixed.*
- *Extension:* store additionally the *orientation* and two orthogonal *scale factors* $s = (s_x, s_y)$ of the model

Example: Generalized Hough Transform

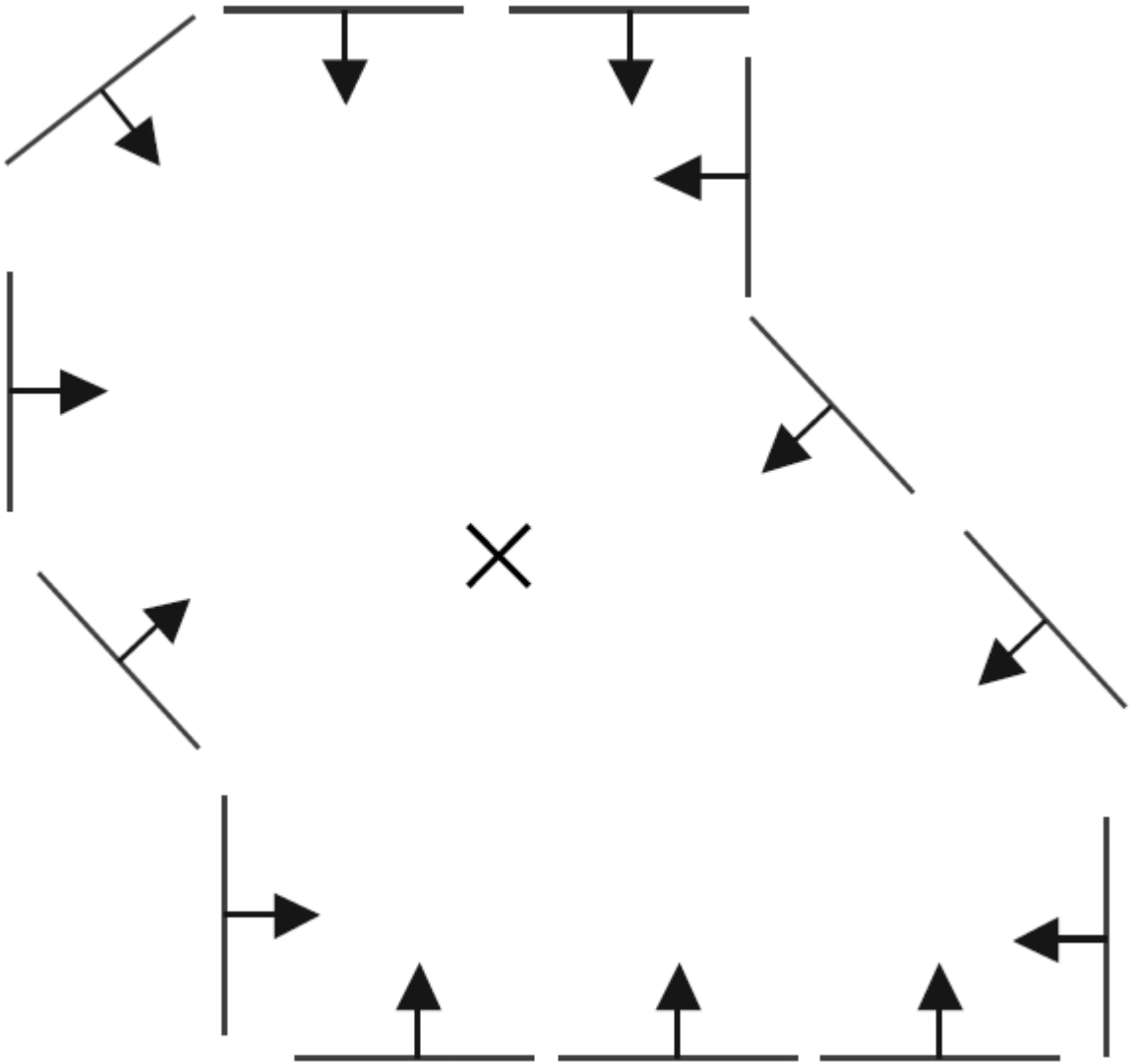
ored a table of
as a function of
his model shape.



Model shape

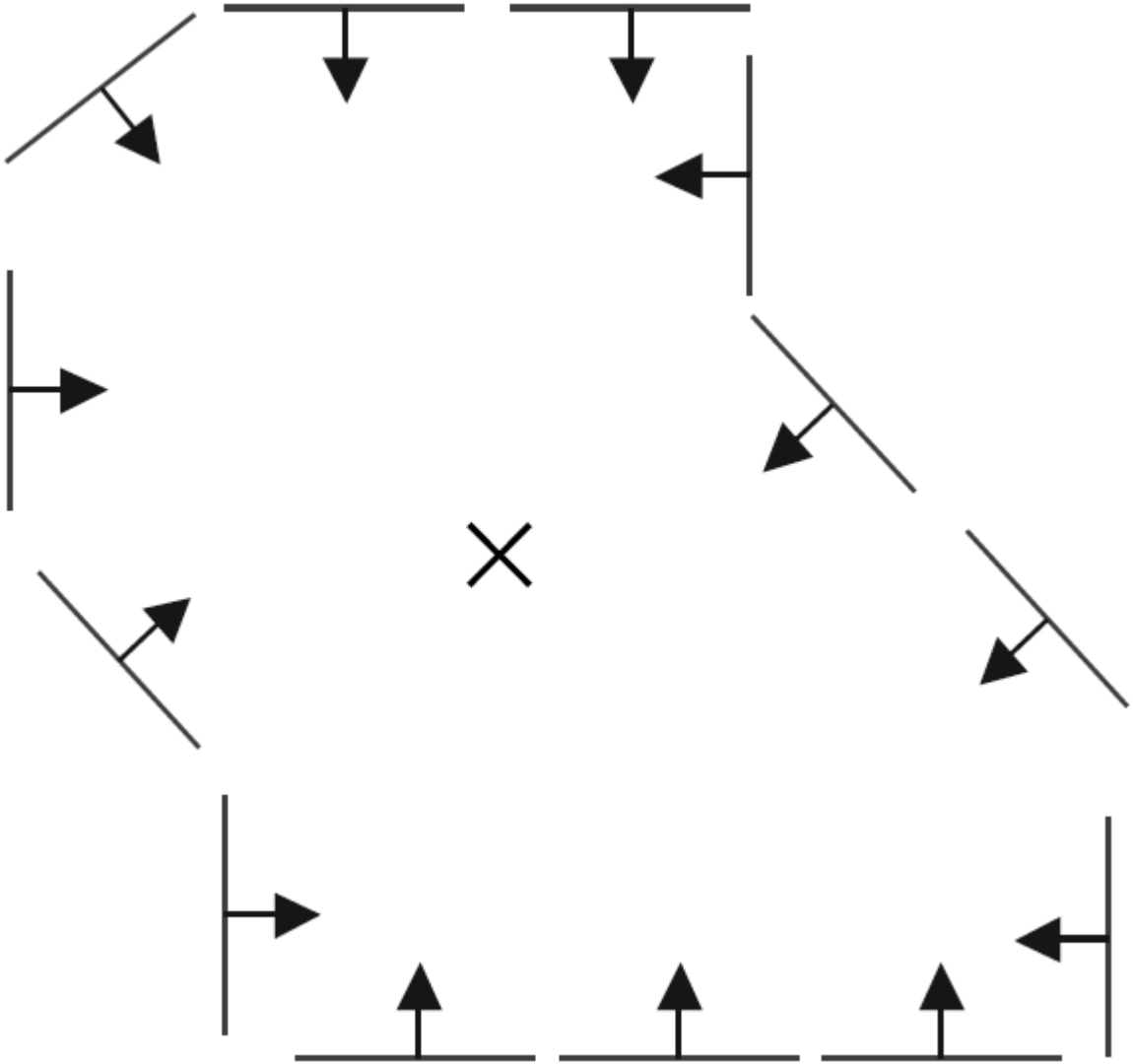
Example: Generalized Hough Transform

at some edge
ew image, and
f that shape.



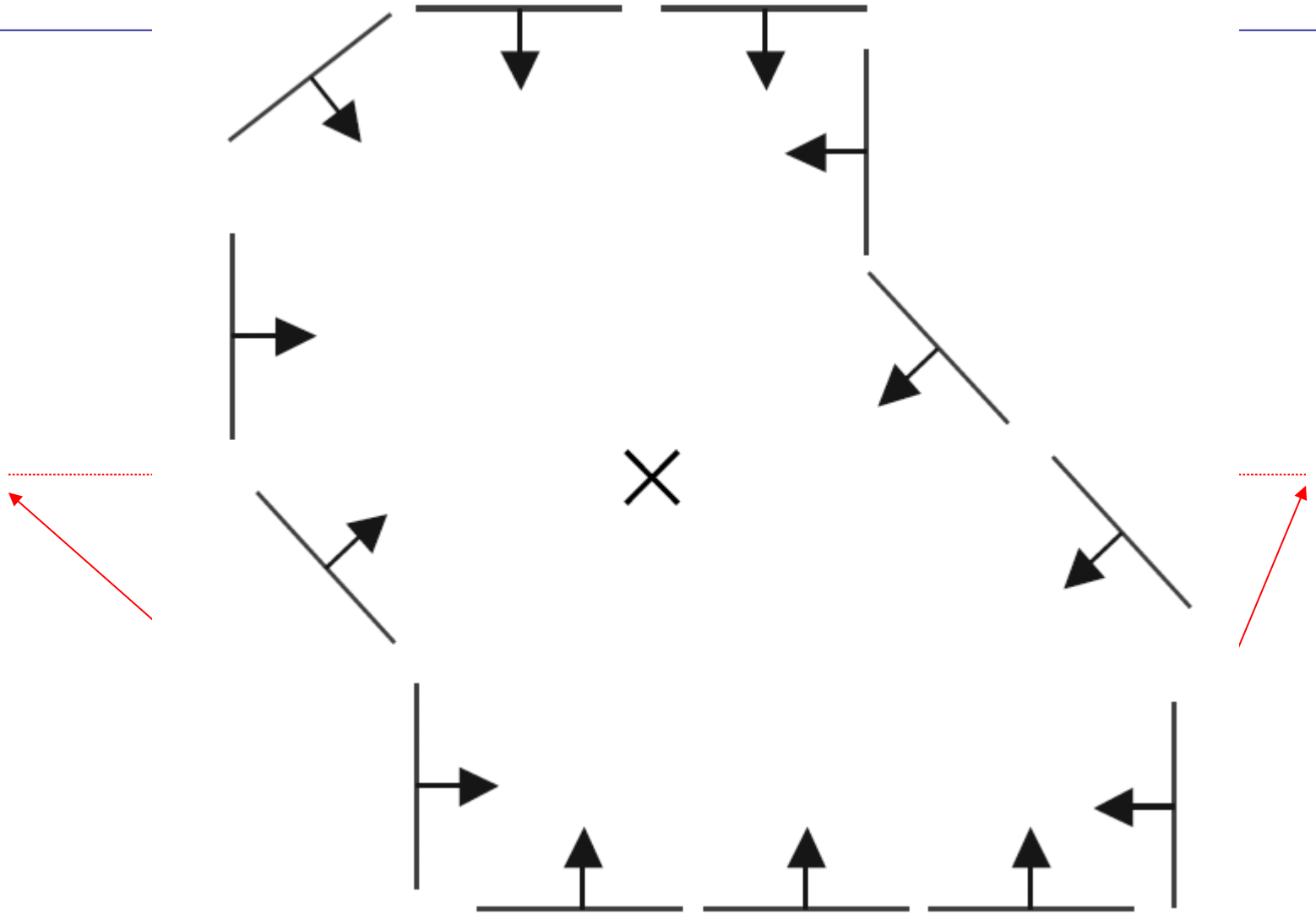
Displacement vectors for model points

Example: Generalized Hough Transform



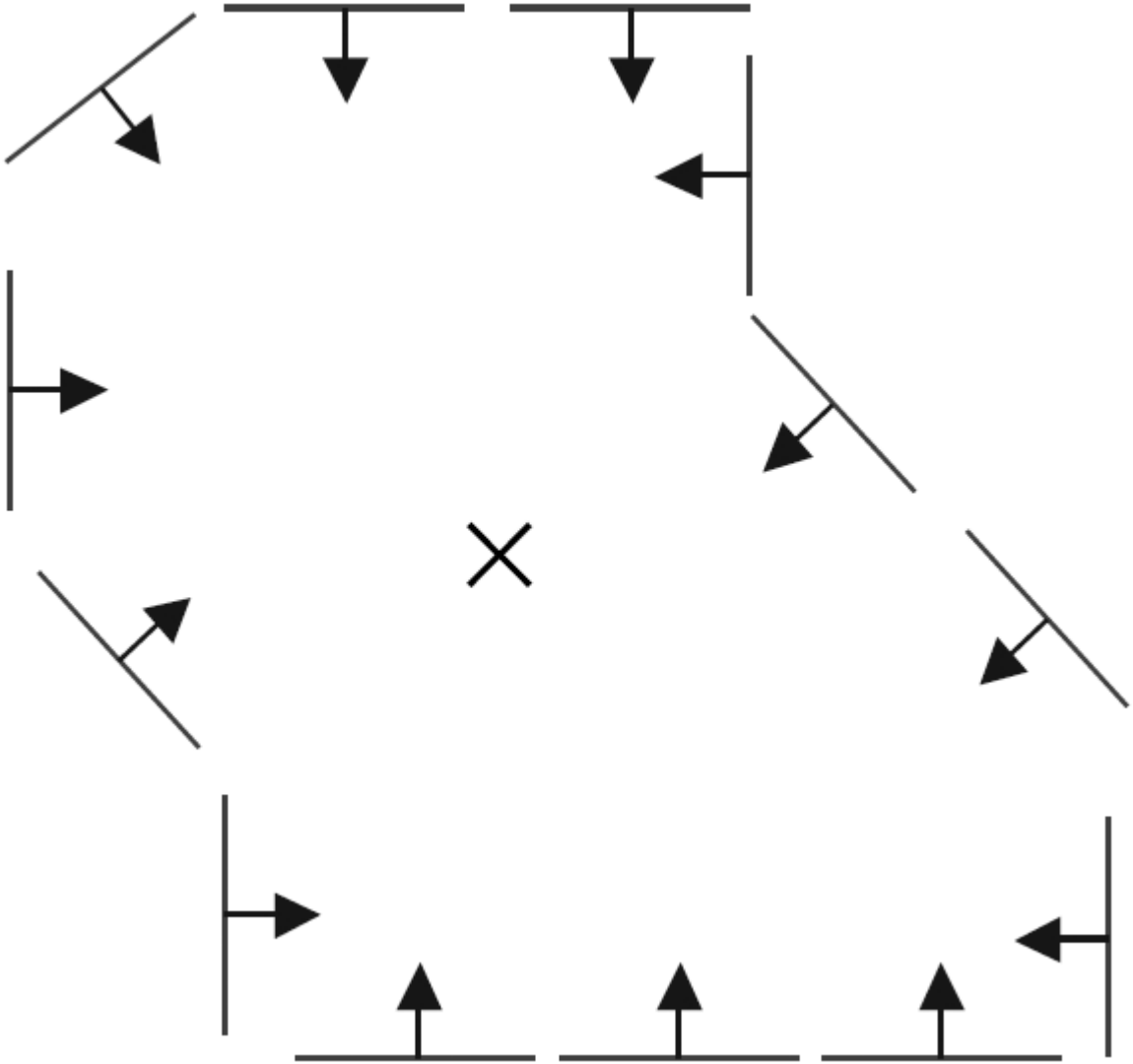
Range of voting locations for test point

Example: Generalized Hough Transform

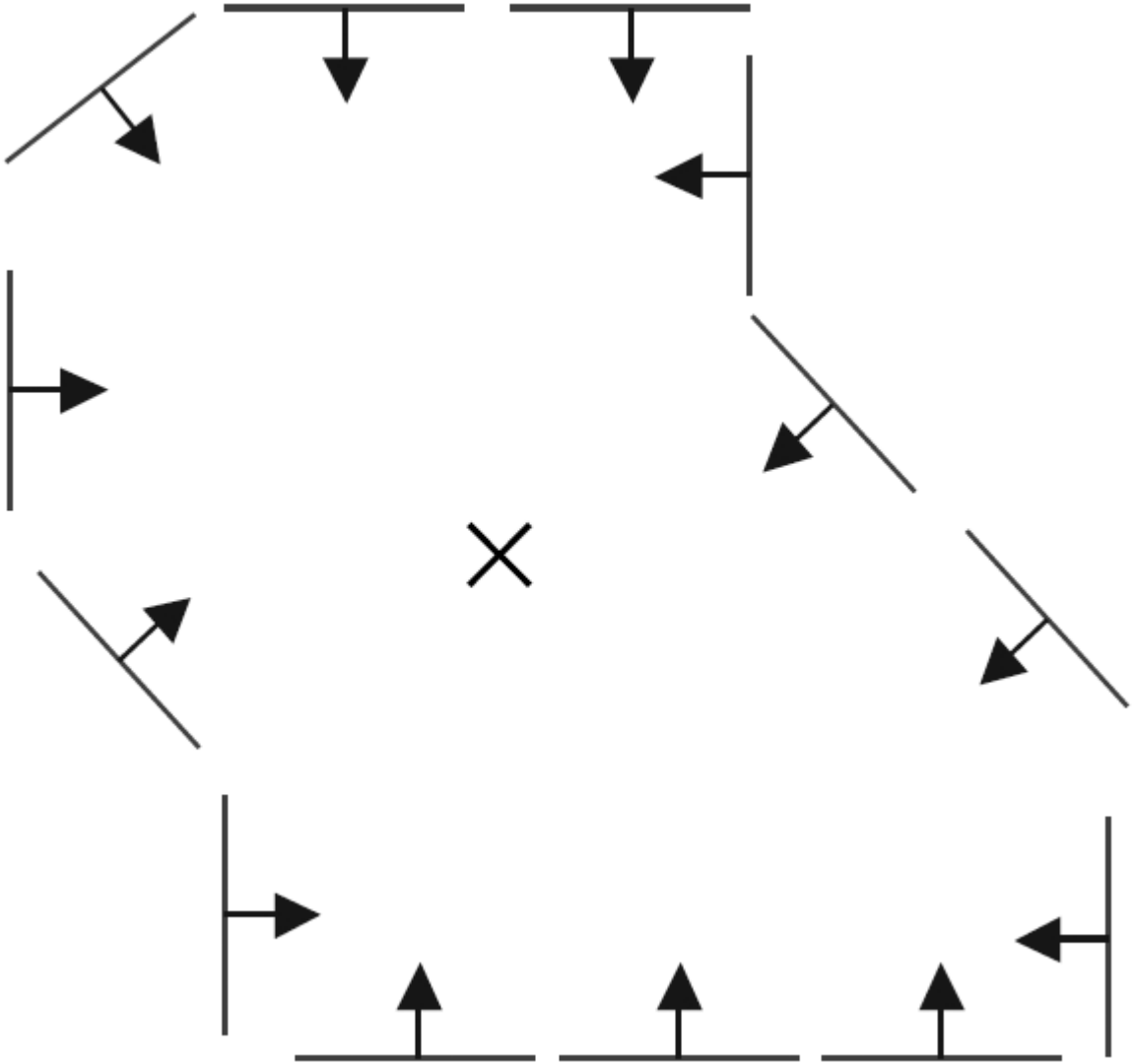


Range of voting locations for test point

Example: Generalized Hough Transform

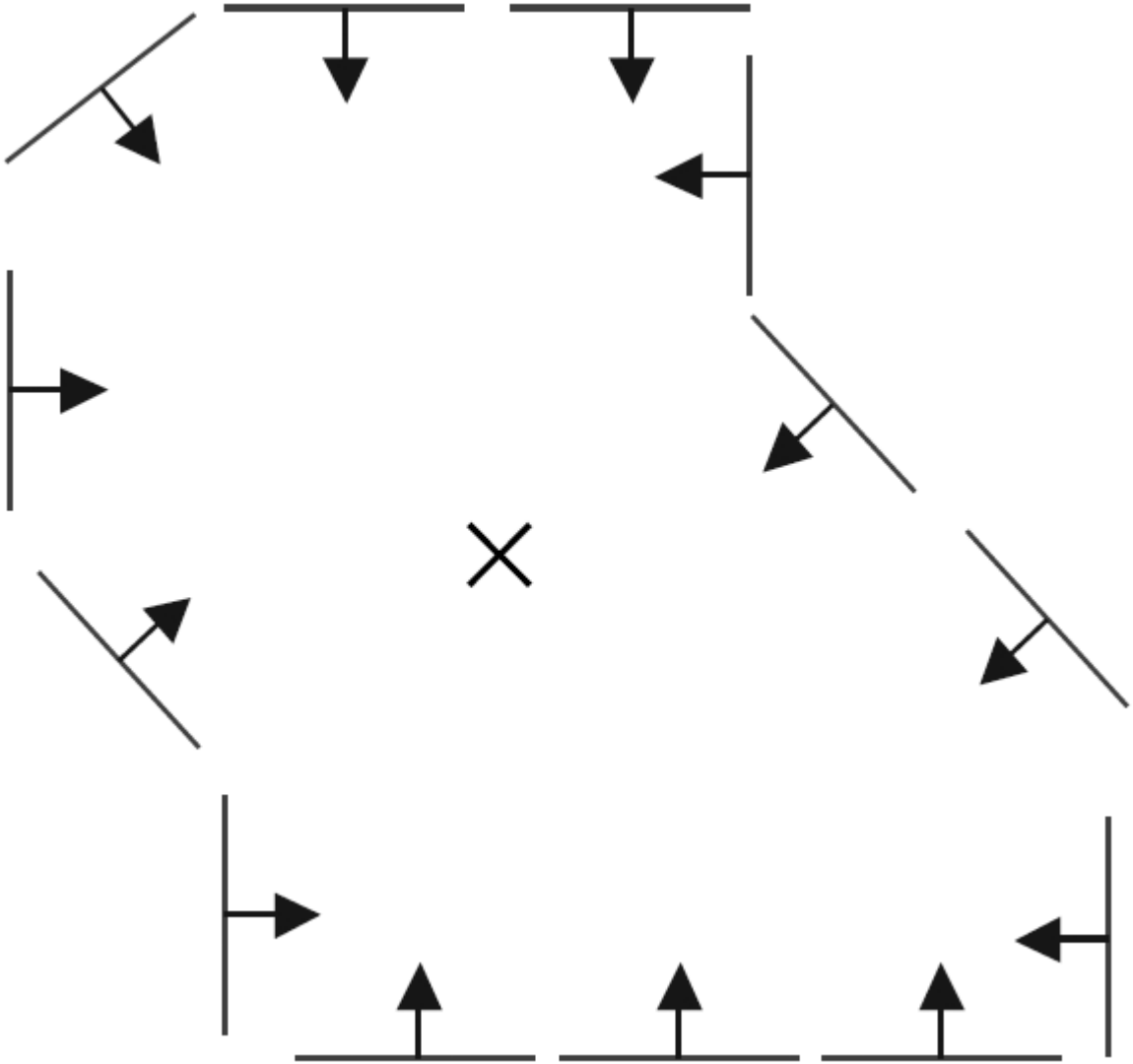


Example: Generalized Hough Transform



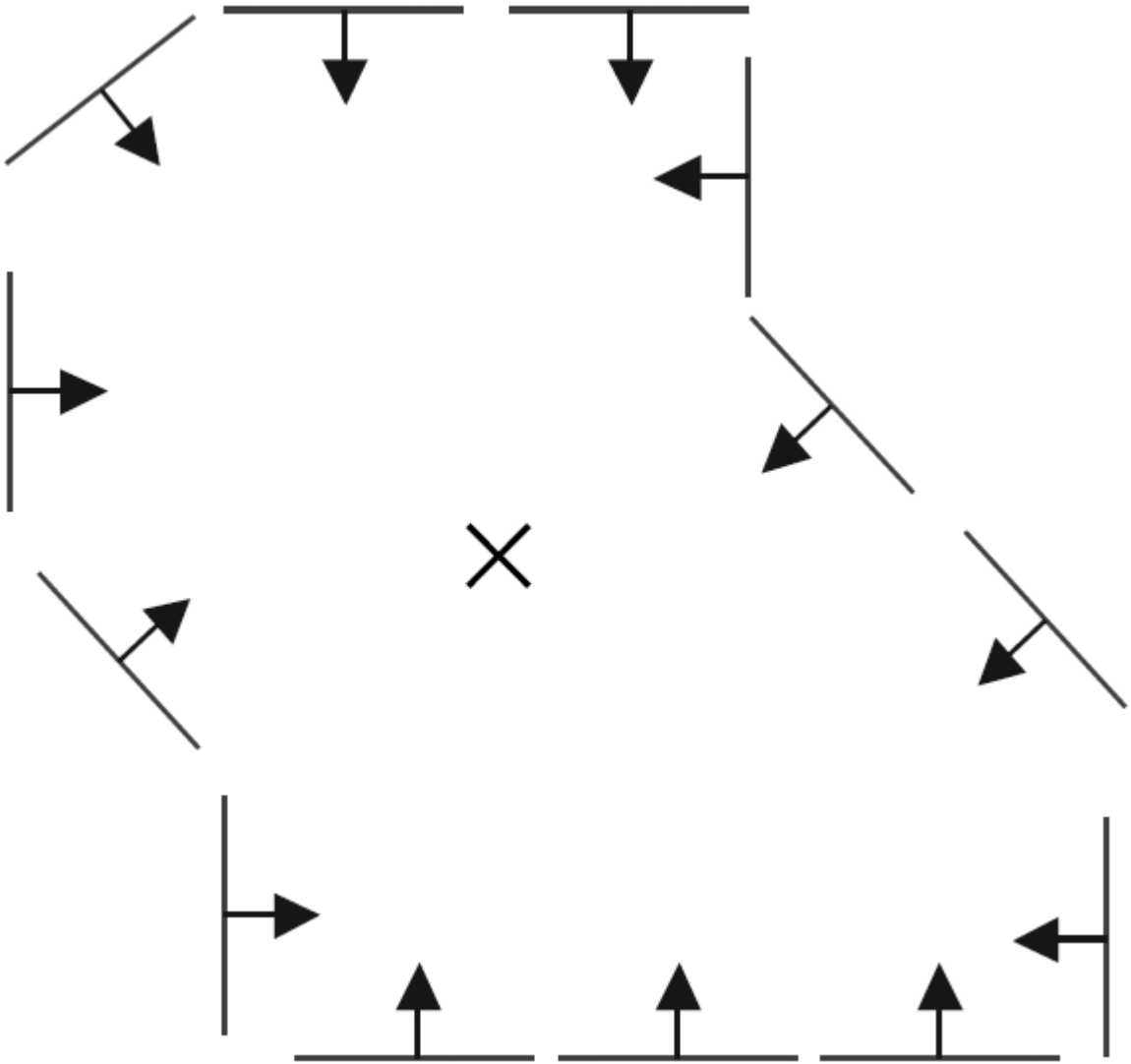
Displacement vectors for model points

Example: Generalized Hough Transform



Range of voting locations for test point

Example: Generalized Hough Transform



Votes for points with $\theta =$ ↙

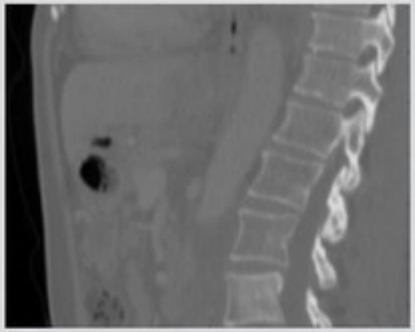
A publication by
Computer VisionNews
RSIP VISION
The magazine of the algorithm community
December 2016

10 We Tried for You NEW! ComputerVision News

The Generalized Hough Transform

Example

Let us consider the following test image: a sagittal slice of a CT-scan



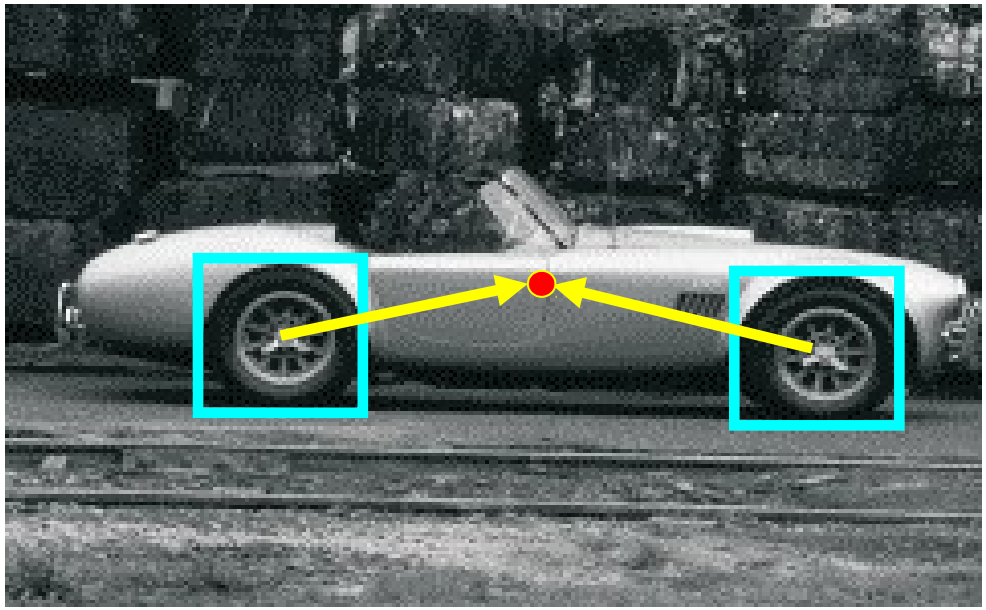
and the following template (Spinal vertebrae) to be detected inside the image

ed for You

http://www.rsipvision.com/ComputerVisionNews-2016December

Application in Recognition

- Instead of indexing displacements by gradient orientation, index by “visual codeword”.



Training image

Visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
Robust Object Detection with Interleaved Categorization and Segmentation, International Journal of Computer Vision, Vol. 77(1-3), 2008.

Application in Recognition

- Instead of indexing displacements by gradient orientation, index by “visual codeword”.

Test image

RANSAC

An alternative to Hough voting:

RANSAC (**RANdom SAmple Consensus)**

- An iterative method to estimate parameters of a model from a set of observed data
- **Idea:** Compute model not from all points but from random subset and check how it fits to the rest of the data (if an outlier is chosen, the resulting model won't have much support)

Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography
MA Fischler, RC Bolles - Communications of the ACM, 1981 - dl.acm.org
Abstract A new paradigm, **Random Sample Consensus** (RANSAC), for fitting a model to experimental data is introduced. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is thus ideally suited for applications
Cited by 15875 Related articles All 9 versions Web of Science: 5804 Cite Save More

RANSAC

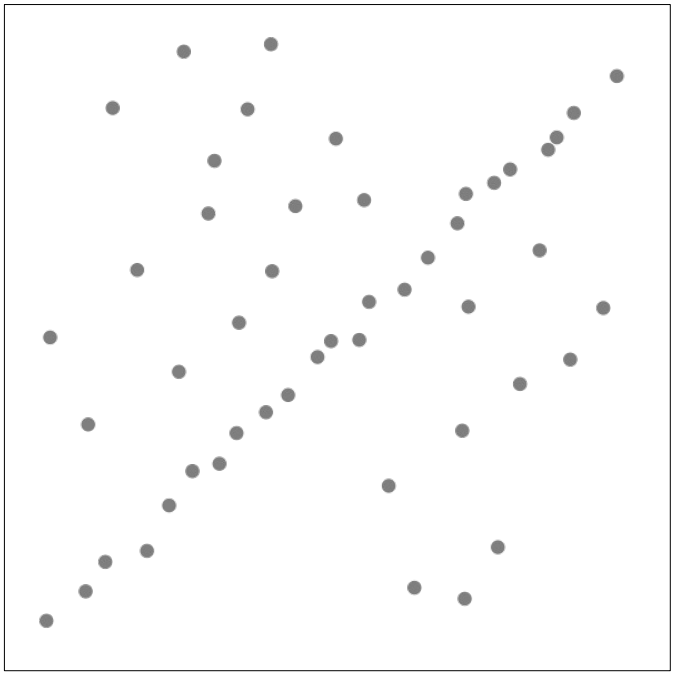
- RANSAC (RANdom SAmple Consensus)

Algorithm Sketch:

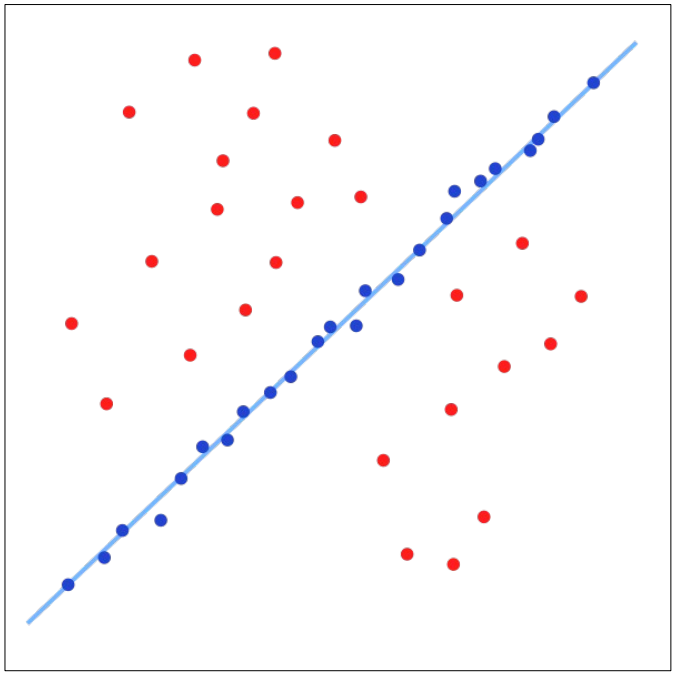
RANSAC Loop:

- Randomly choose data points to form a hypothesis H
- Measure how many other data points fit H
- If H better than H_{best}, $H_{best} := H$

RANSAC



A data set with many outliers for which a line has to be fitted.

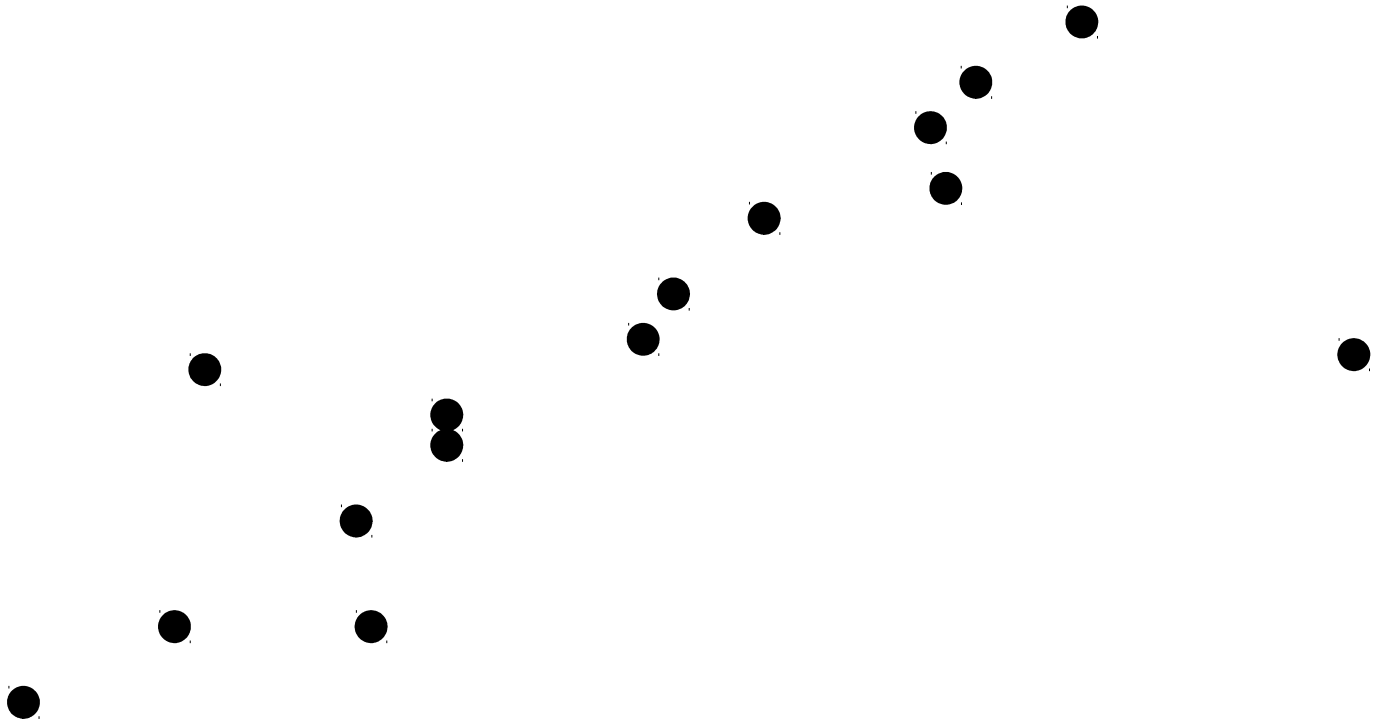


Fitted line with RANSAC; outliers have no influence on the result.

[Wikipedia: RANSAC]

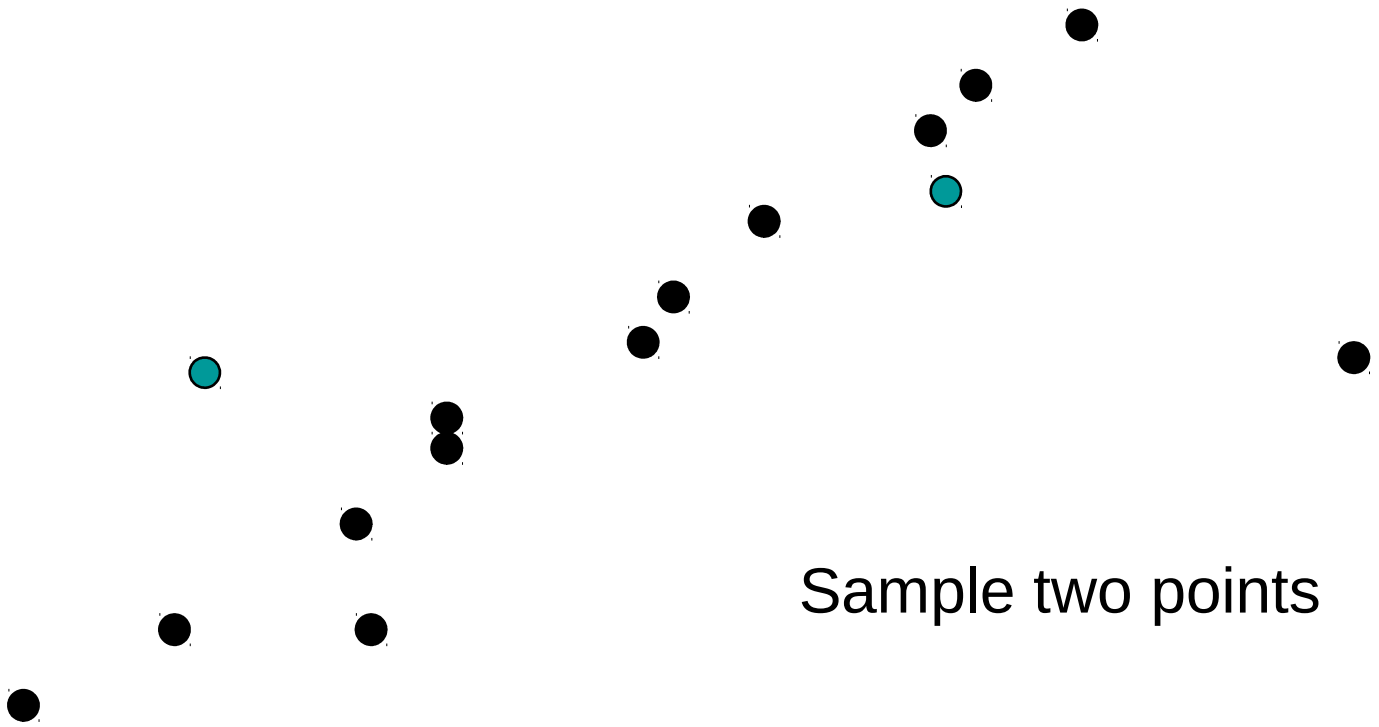
RANSAC Line Fitting Example

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



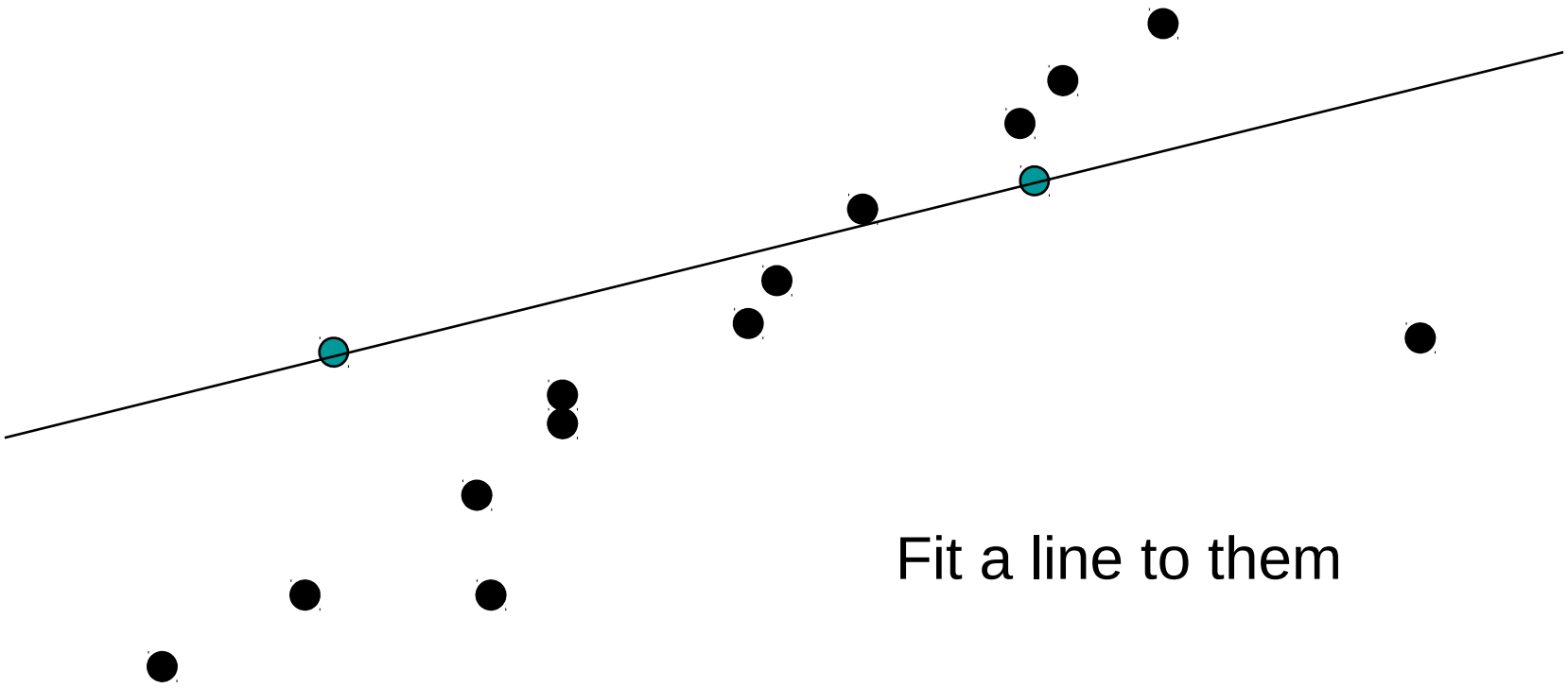
RANSAC Line Fitting Example

- Task: Estimate the best line



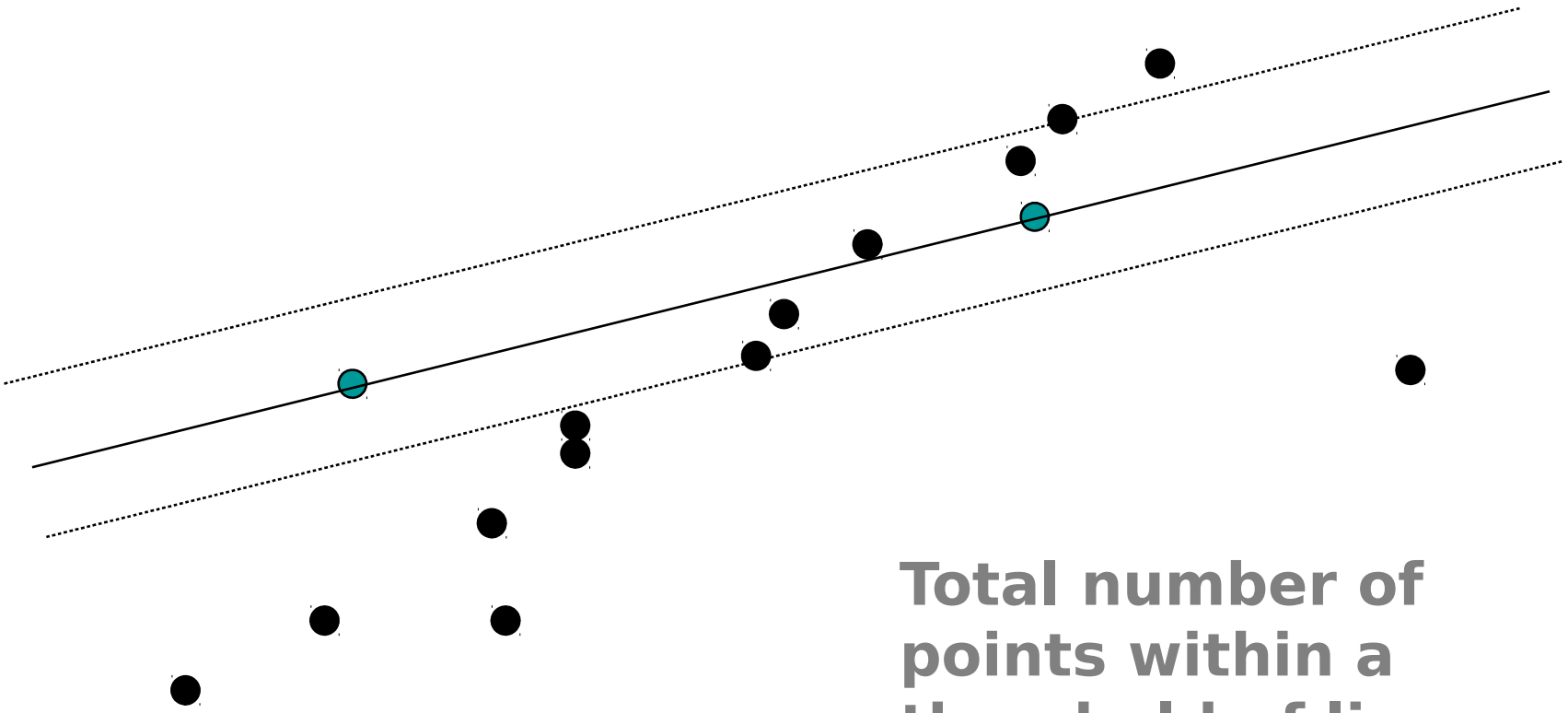
RANSAC Line Fitting Example

- Task: Estimate the best line



RANSAC Line Fitting Example

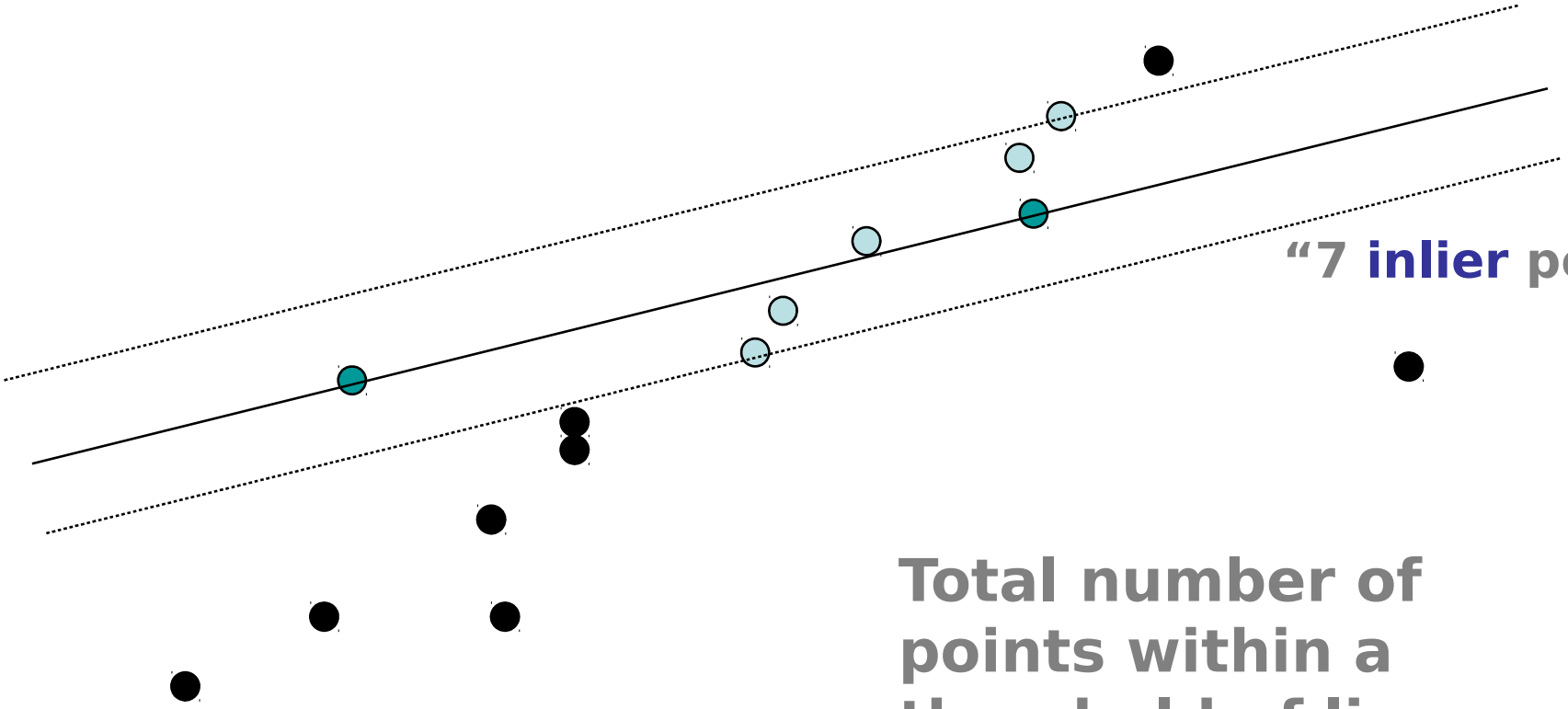
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC Line Fitting Example

- Task: Estimate the best line

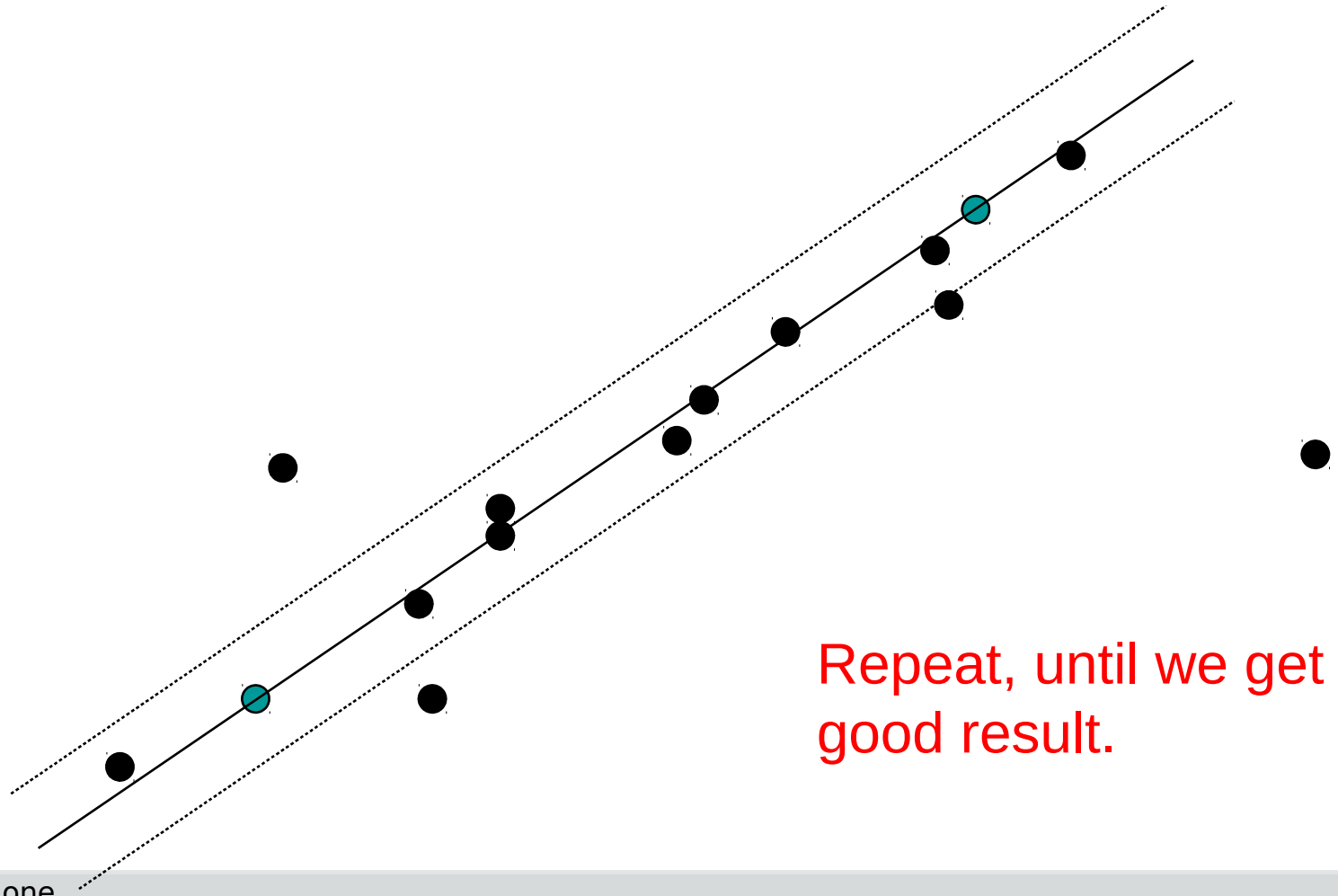


“7 inlier points”

Total number of points within a threshold of line.

RANSAC Line Fitting Example

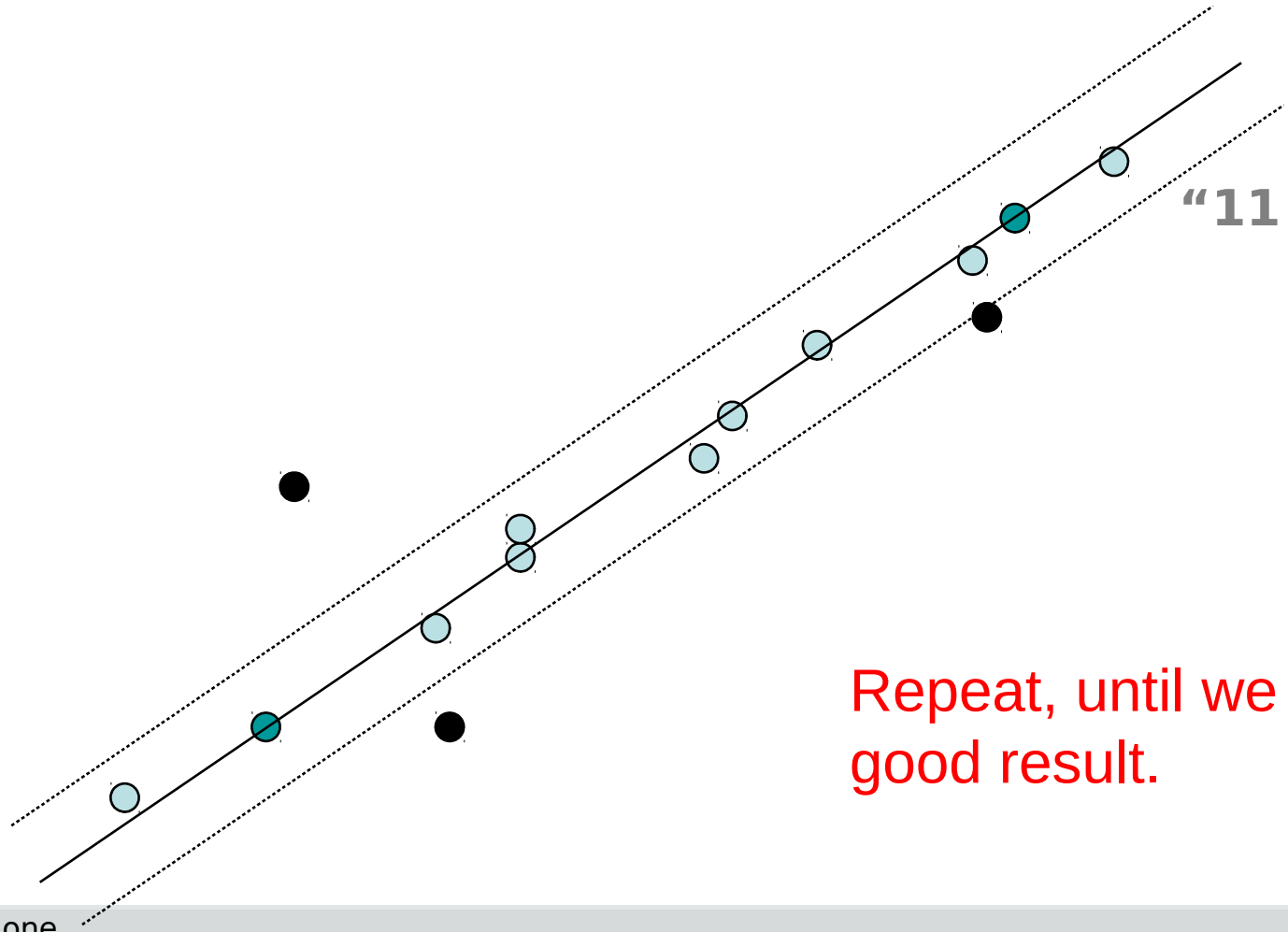
- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC Line Fitting Example

- Task: Estimate the best line



"11 **inlier** points"

Repeat, until we get a good result.

RANSAC

- The RANSAC Algorithm:
- Repeat the following steps k times:
  - Select a random minimal subset of the original data (for lines: two points)
  - Fit model to this set: hypothesis H
  - Test other data against H and determine number of *inliers* (points that fit to model)
  - If #inliers from H is larger than #inliers from best model H_{best}:

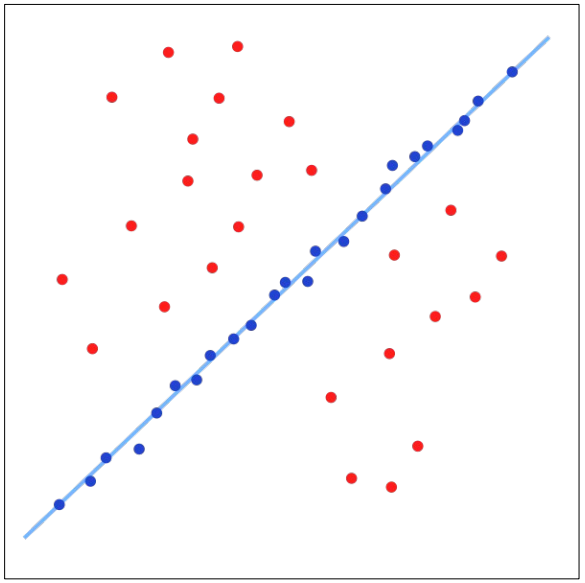
$$H_{best} := H$$

- Improve model by re-estimating it based on all inliers from H_{best}

[Wikipedia: RANSAC]

RANSAC: How many samples?

- How many samples are needed?
- Just **one** correct sample is required to give optimal results!
- Determine probability that **all** samples fail
- This failure probability should be lower than a threshold



RANSAC: How many samples?

- How many samples are needed?
  - Suppose e is fraction of inliers
  - m points needed to define hypothesis (2 for lines)
  - k iterations

- Prob. that a single point is correct (inlier): e
- Prob. that a single sample of m points is correct: e^m
- Prob. that a single sample of m points fails: $1 - e^m$
- Prob. that all k samples fail is: $(1 - e^m)^k$

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

RANSAC: How many samples?

- How many samples are needed? Example:
  - Suppose ϵ is fraction of inliers 0.9 (0.1 outliers)
  - m points needed to define hypothesis (2 for lines)
  - k iterations

- Prob. that a single point is correct (inlier): ϵ 0.9
- Prob. that a single sample of m points is correct: ϵ^m 0.81
- Prob. that a single sample of m points fails: $1 - \epsilon^m$ 0.19
- Prob. that all k samples fail is: $(1 - \epsilon^m)^k$ k=3: 0.0069

⇒ Choose k high enough to keep this below desired failure rate. (here: choose k=3 to achieve failure rate < 0.01)

RANSAC: Number of iterations

Number of required iterations k to achieve failure rate p<=0.01

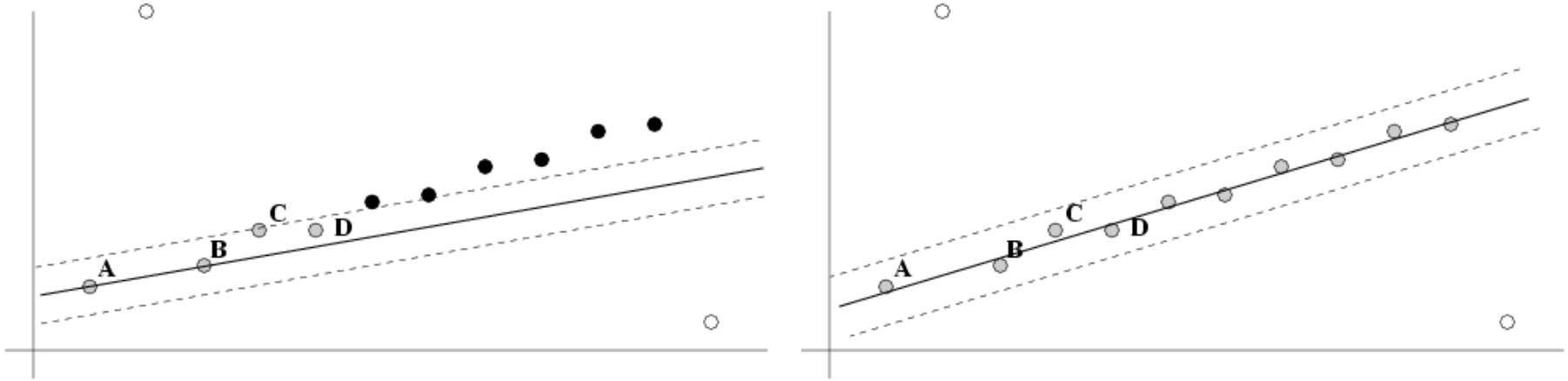
Sample size m	Proportion of outliers						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Example for lines (m=2): 10% outliers require 3 iterations

RANSAC

Last step of algorithm (after loop):

- Improve 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.

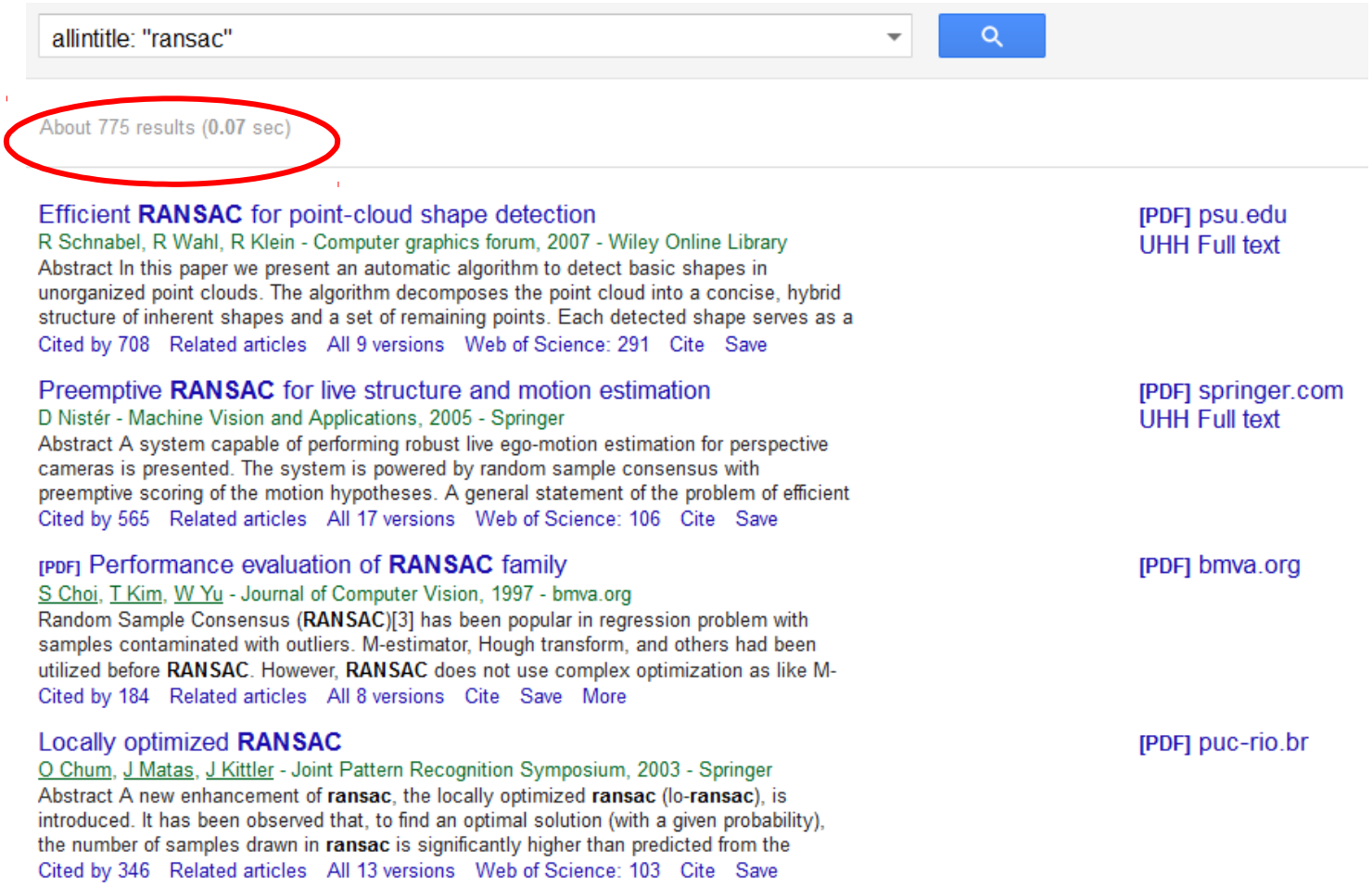


RANSAC

- RANSAC achieves a „good“ result with a certain probability
- The probability increases with the number of iterations
- The parameters (estimated number of inliers/outliers, allowed distance from model to count as inlier) have to be chosen according to the application

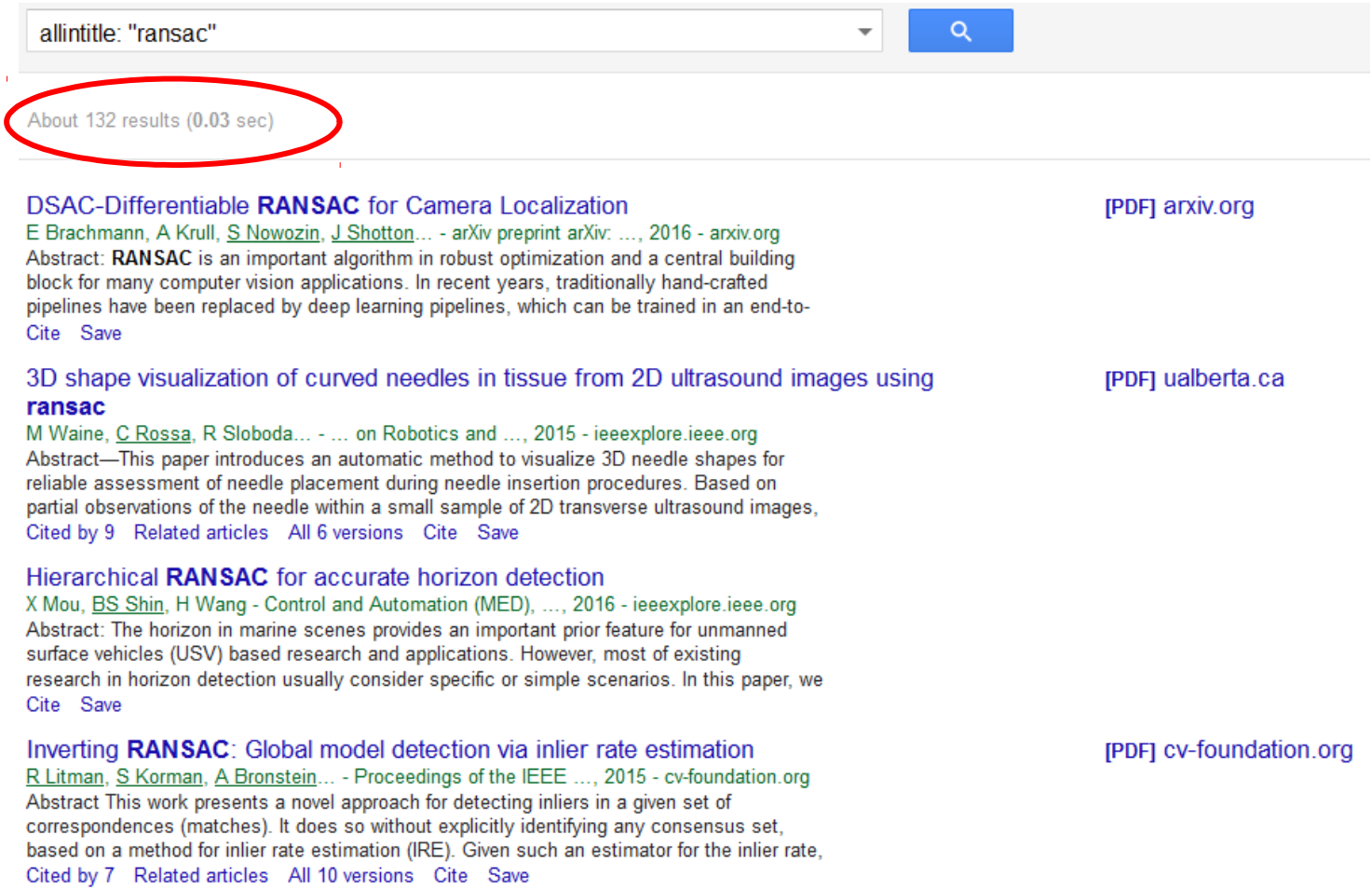
RANSAC Extensions

Since 1981: numerous improvements



RANSAC Extensions

Still often used. Recent publications (since 2015):



allintitle: "ransac"

About 132 results (0.03 sec)

DSAC-Differentiable **RANSAC** for Camera Localization [PDF] arxiv.org
E Brachmann, A Krull, S Nowozin, J Shotton... - arXiv preprint arXiv: ..., 2016 - arxiv.org
Abstract: **RANSAC** is an important algorithm in robust optimization and a central building block for many computer vision applications. In recent years, traditionally hand-crafted pipelines have been replaced by deep learning pipelines, which can be trained in an end-to-
Cite Save

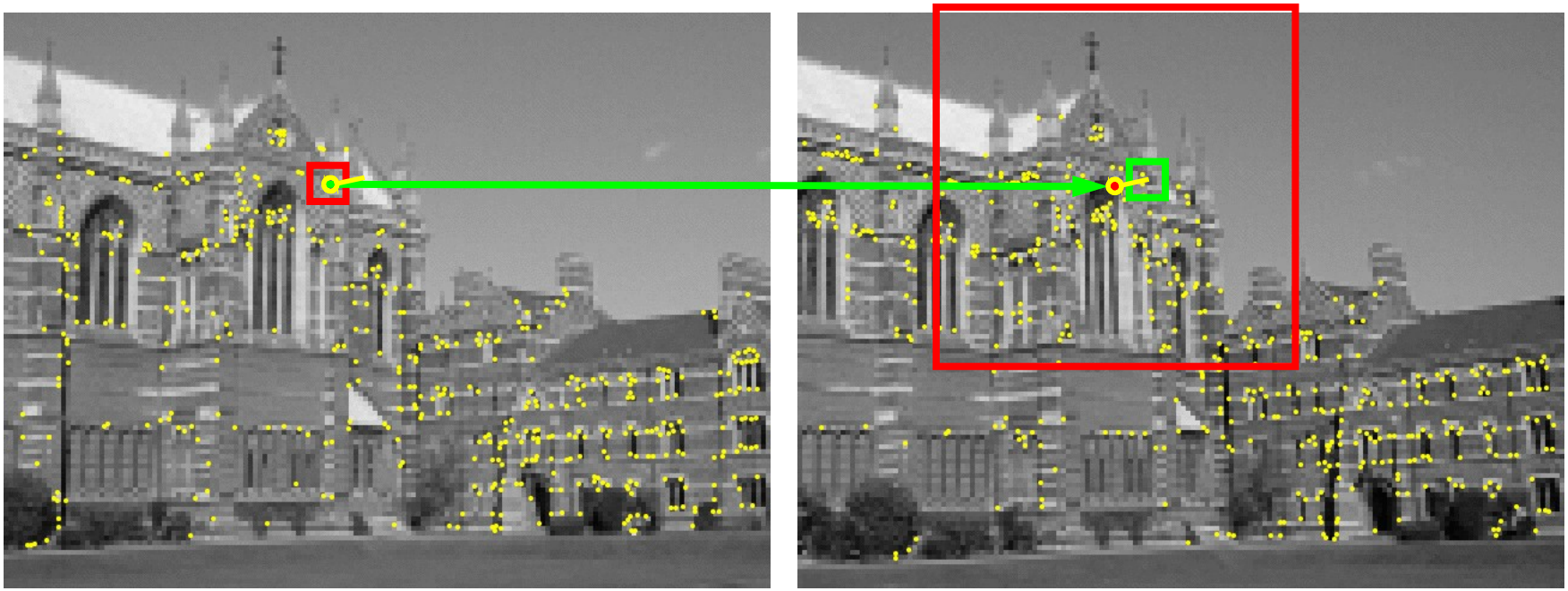
3D shape visualization of curved needles in tissue from 2D ultrasound images using **ransac** [PDF] ualberta.ca
M Waine, C Rossa, R Sloboda... - ... on Robotics and ..., 2015 - ieeexplore.ieee.org
Abstract—This paper introduces an automatic method to visualize 3D needle shapes for reliable assessment of needle placement during needle insertion procedures. Based on partial observations of the needle within a small sample of 2D transverse ultrasound images,
Cited by 9 Related articles All 6 versions Cite Save

Hierarchical **RANSAC** for accurate horizon detection
X Mou, BS Shin, H Wang - Control and Automation (MED), ..., 2016 - ieeexplore.ieee.org
Abstract: The horizon in marine scenes provides an important prior feature for unmanned surface vehicles (USV) based research and applications. However, most of existing research in horizon detection usually consider specific or simple scenarios. In this paper, we
Cite Save

Inverting **RANSAC**: Global model detection via inlier rate estimation [PDF] cv-foundation.org
R Litman, S Korman, A Bronstein... - Proceedings of the IEEE ..., 2015 - cv-foundation.org
Abstract This work presents a novel approach for detecting inliers in a given set of correspondences (matches). It does so without explicitly identifying any consensus set, based on a method for inlier rate estimation (IRE). Given such an estimator for the inlier rate,
Cited by 7 Related articles All 10 versions Cite Save

Example: Finding Feature Matches

- Find best stereo match within a square search window (here 300x300 pixels)
- Remove outliers

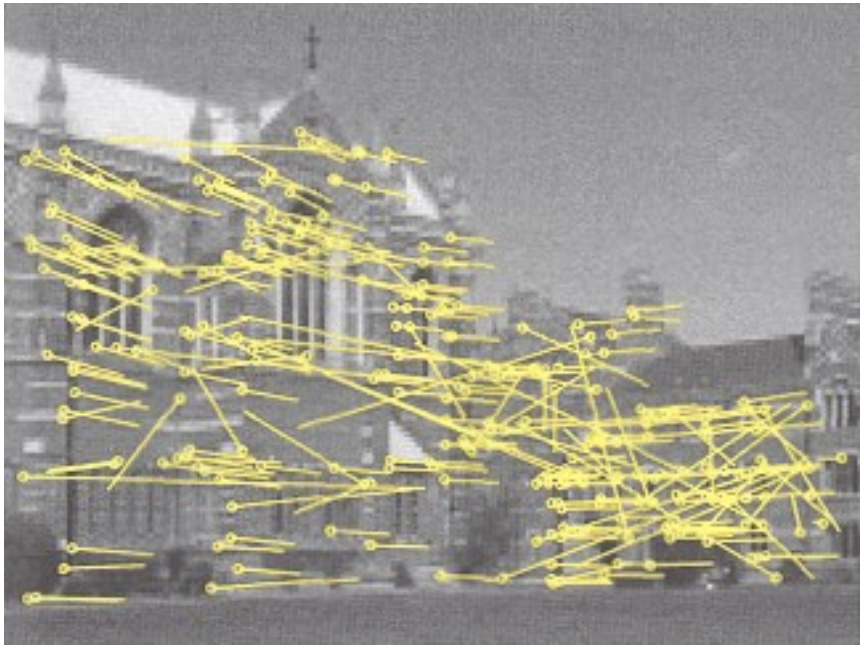


[Images from Hartley & Zisserman]

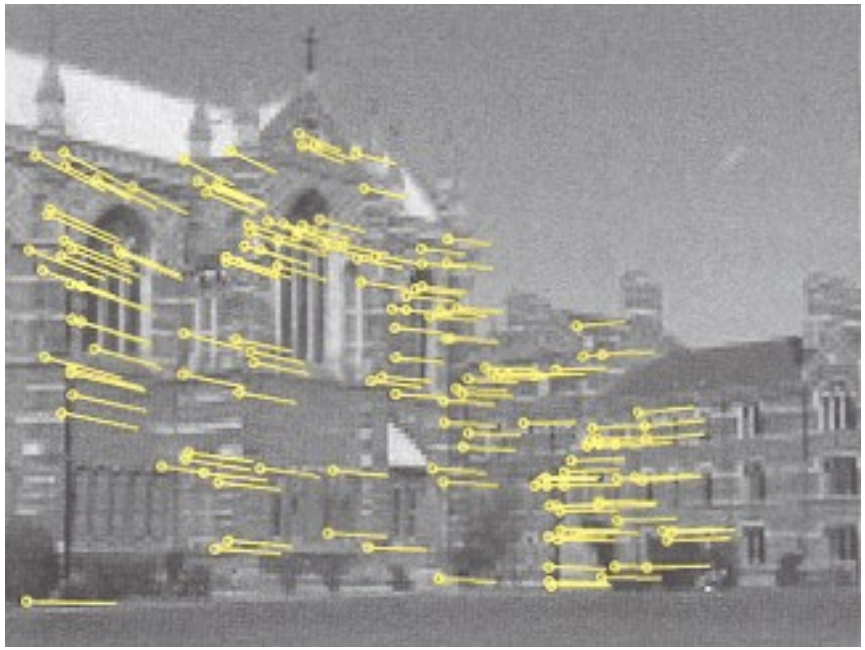
Example: Finding Feature Matches

- Find best stereo match within a square search window (here 300x300 pixels)
- Remove outliers

before RANSAC



after RANSAC



[Images from Hartley & Zisserman]

RANSAC

- Just for fun: The RANSAC Song (Daniel Wedge):
- https://www.youtube.com/watch?v=1YNjMxxXO-E

- When you have outliers you may face much frustration
  if you include them in a model fitting operation.
  But if your model's fit to a sample set of minimal size,
  the probability of the set being outlier-free will rise.
  Brute force tests of all sets will cause computational constipation.
- *N* random samples
  will provide an example
  of a fitted model uninfluenced by outliers. No need to test all combinations!
- Each random trial should have its own unique sample set
  and make sure that the sets you choose are not degenerate.
  N, the number of sets, to choose is based on the probability
  of a point being an outlier, and of finding a set that's outlier free.
  Updating *N* as you go will minimise the time spent.
- So if you gamble
  that *N* samples are ample
  to fit a model to your set of points, it's likely that you will win the bet.
- Select the set that boasts
  that its number of inliers is the most (you're almost there).
  Fit a new model just to those inliers and discard the rest,
  an estimated model for your data is now possessed!
  This marks the end point of your model fitting quest.

GHT versus RANSAC

GHT and RANSAC both estimate model parameters under a significant fraction of outliers

- GHT:
  - Efficient for small voting spaces (upper limit: 4D voting spaces)
  - In higher dimensions, it can handle larger percentage of outliers, since inconsistent votes are spread over a higher-dimensional volume
  - Find all model instances at once
  - Runtime independent on inlier ratio
- RANSAC:
  - Runtime depends on data points (searched in each iteration through all data points) ⇨ expensive for large datasets
  - Runtime depends on outlier ratio ⇨ expensive for high ratios
  - One-model approach: made to estimate only one model per data set
  - Scales better to high-dimensional models
  - Very general method: applicable to many problems

Summary

- The Canny edge detector creates a binary edge image from the gradient magnitude, using non-maxima suppression (for edge thinning) and hysteresis thresholding (for bridging gaps).
- The Hough Transform (HT) is a method to estimate the parameters of a model (e.g. line or circle). Each data point votes for all plausible models.
- The Generalized Hough Transform (GHT) is a generalization to arbitrary shapes
- RANSAC is a non-deterministic algorithm that randomly chooses data points to estimate a model.
- All methods can deal well with outliers

Primary Literature

Canny: Gonzalez/Woods, 4th ed, 2017: parts of chapter 10

Hough Transform:

- Klette: chapter 3.4
- Gonzalez/Woods, 4th ed, 2017: parts of chapter 10

RANSAC and Generalized Hough Transform:

- Visual Object Recognition, Synthesis Lectures on Artificial Intelligence and Machine Learning, April 2011, Kristen Grauman & Bastian Leibe, Chapter 5.2
  https://pdfs.semanticscholar.org/5255/490925aa1e01ac0b9a55e93ec8c82efc07b7.pdf

Secondary Literature

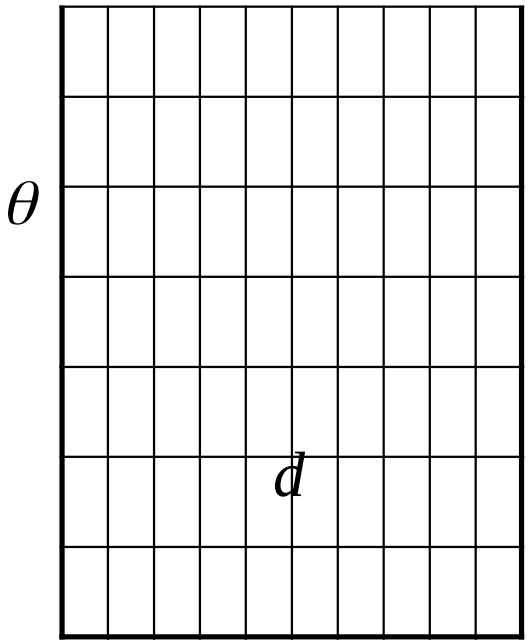
- J. Canny: “A Computational Approach To Edge Detection”, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.
- Dana H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, 1980
- Fischler, Martin A., and Robert C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM* 24.6 (1981): 381-395.
- B. Leibe, A. Leonardis, and B. Schiele, “Robust Object Detection with Interleaved Categorization and Segmentation”, *International Journal of Computer Vision*, Vol. 77(1-3), 2008.

Hough Transform Algorithm

Using the polar parameterization:

$$x\cos\theta + y\sin\theta = d$$

H: accumulator array (votes)



Basic Hough transform algorithm

1. Initialize $H[d,\theta] = 0$.
2. For each edge point (x,y) in the image

   for $\theta = 0$ to 180 // some quantization

   $$d = x\cos\theta + y\sin\theta$$

   H[d, θ] += 1
3. Find the value(s) of (d,θ) where $H[d,\theta]$ is maximal.
4. The detected line in the image is given by $d = x\cos\theta + y\sin\theta$

Hough Space

- Hough space: a parameter space that represents structures (e.g. lines) based on their parameters

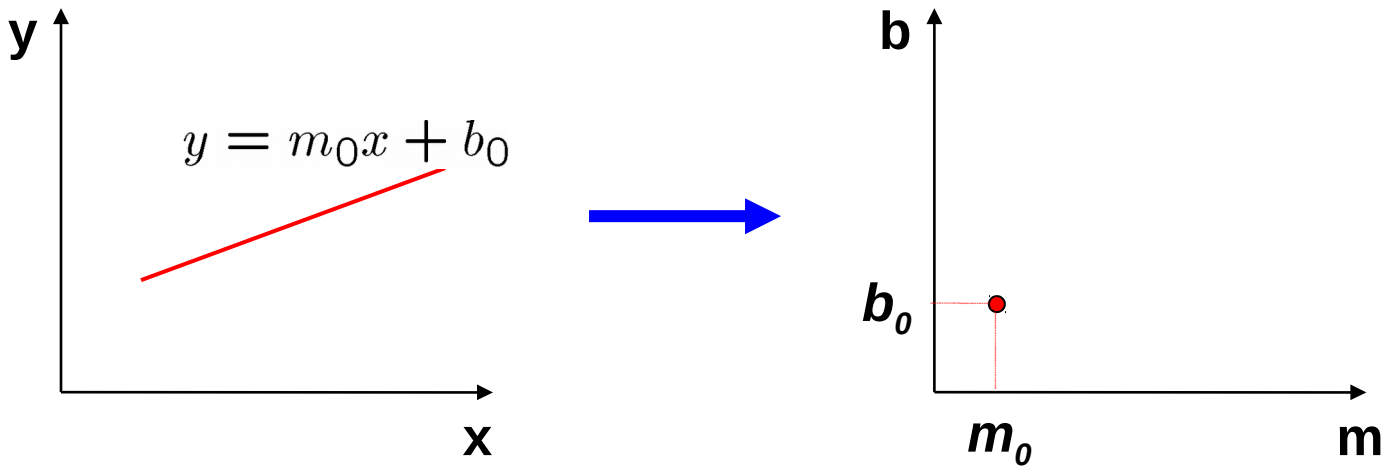
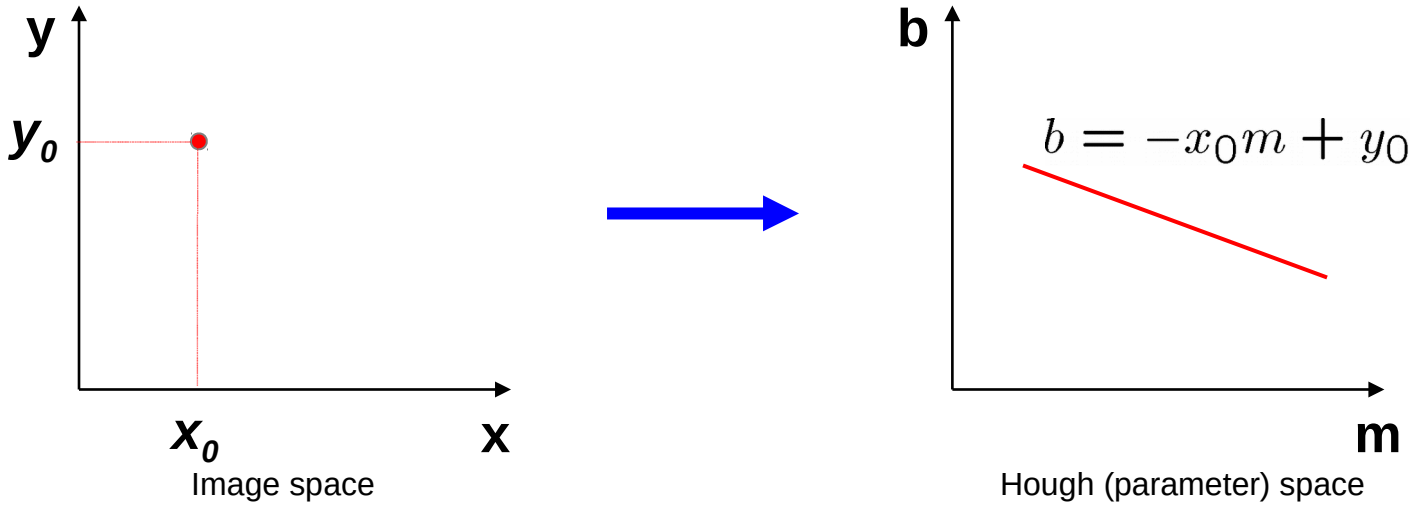


Image space

Hough (parameter) space

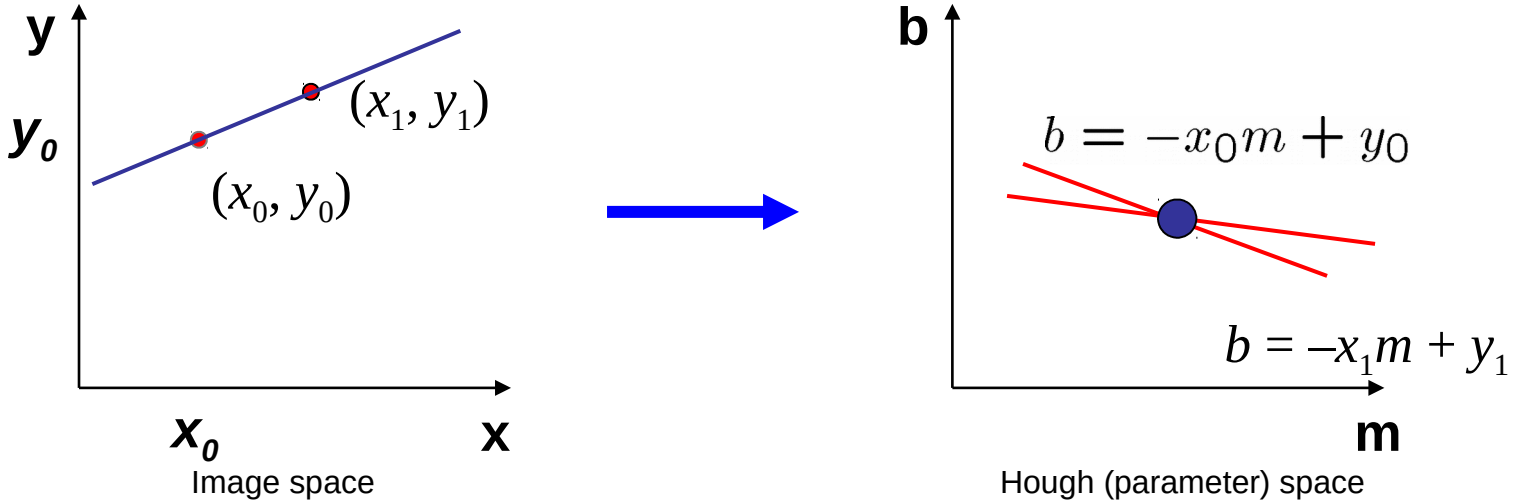
- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space.
  - To go from image space to Hough space:
    - Given a set of points (x,y), find all (m,b) such that $y = mx + b$

Finding Lines in an Image: Hough Space



- Connection between image (x,y) and Hough (m,b) spaces
  - A line in the image corresponds to a point in Hough space.
  - To go from image space to Hough space:
    - Given a set of points (x,y), find all (m,b) such that $y = mx + b$
  - What does a point (x_0, y_0) in the image space map to?
    - Answer: the solutions of $b = -x_0m + y_0$
    - This is a line in Hough space.

Finding Lines in an Image: Hough Space



- What are the line parameters for the line that contains both (x_0, y_0) and (x_1, y_1)?
  - It is the intersection of the lines

    $b = -x_0 m + y_0$ and

    $b = -x_1 m + y_1$

Finding Lines in an Image: Hough Space

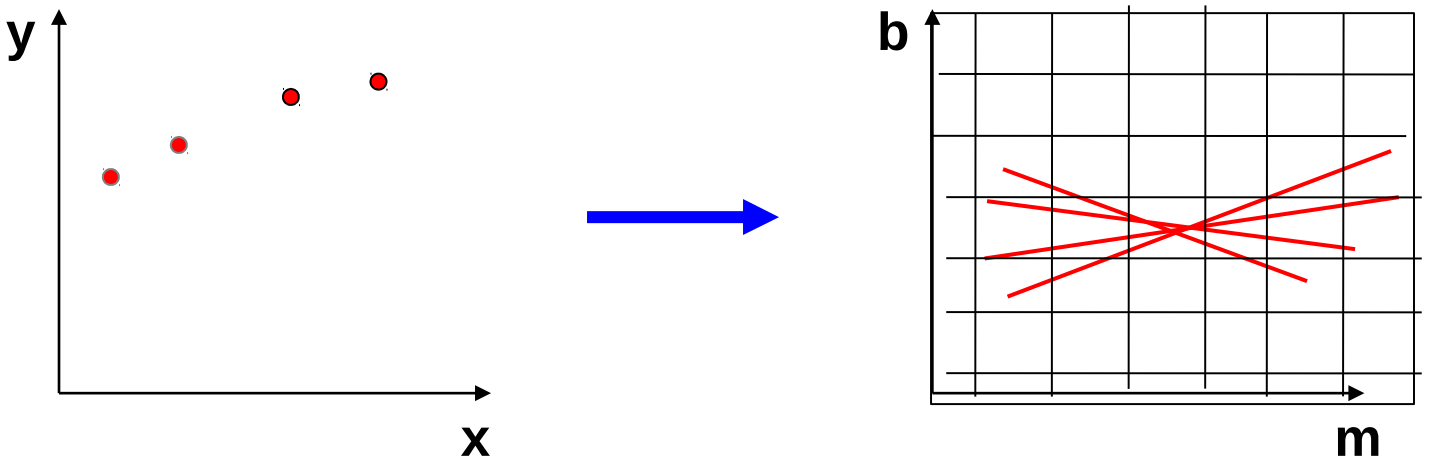


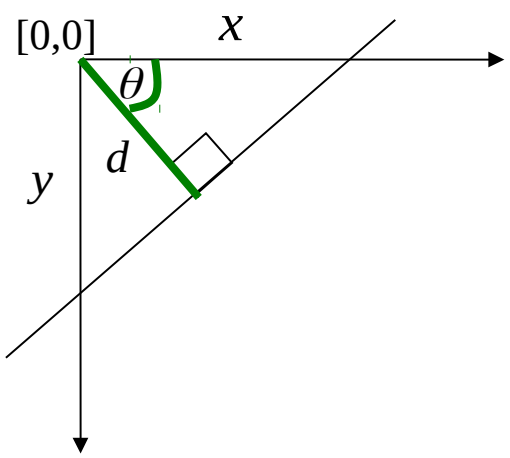
Image space

Hough (parameter) space

- How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?
  - Let each edge point in image space *vote* for a set of possible parameters in Hough space.
  - Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Polar Representation for Lines

- Issues with usual (m,b) parameter space: m can take on infinite values, undefined for vertical lines.
- Use instead the Hesse normal form: $x\cos\theta - y\sin\theta = d$



d : perpendicular distance from line to origin

θ : angle the perpendicular makes with the x-axis

- Point in image space
  ⇒ Sinusoid segment in Hough space

Hough Transform for Circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, gradient direction

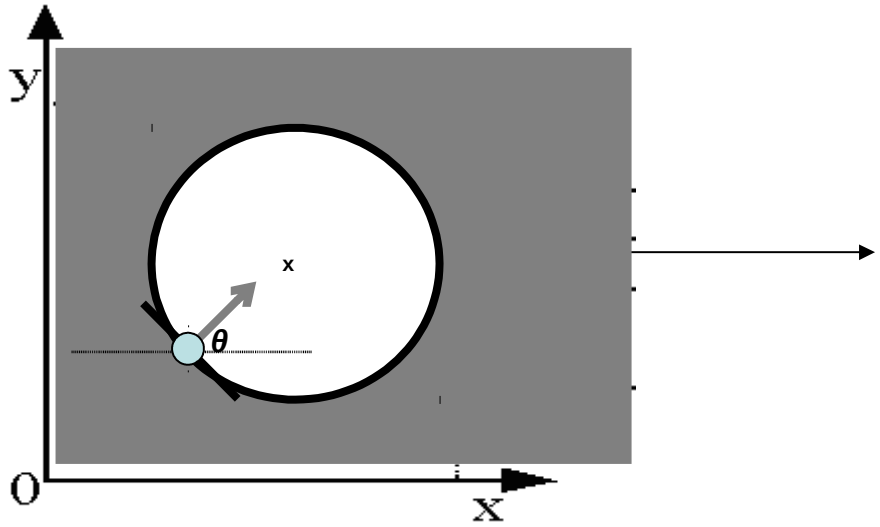
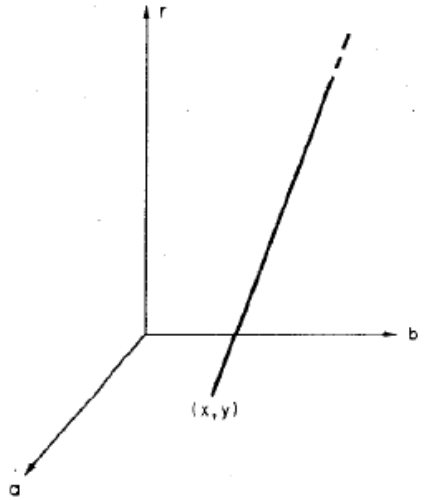


Image space



Hough space

Hough Transform for Circles

For every edge pixel (x,y) :

For each possible radius value r:

For each possible gradient direction θ:

// or use estimated gradient

$$a = x - r\cos(\theta)$$

$$b = y + r\sin(\theta)$$

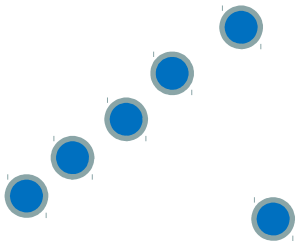
$$H[a,b,r] \mathrel{+}= 1$$

end

end

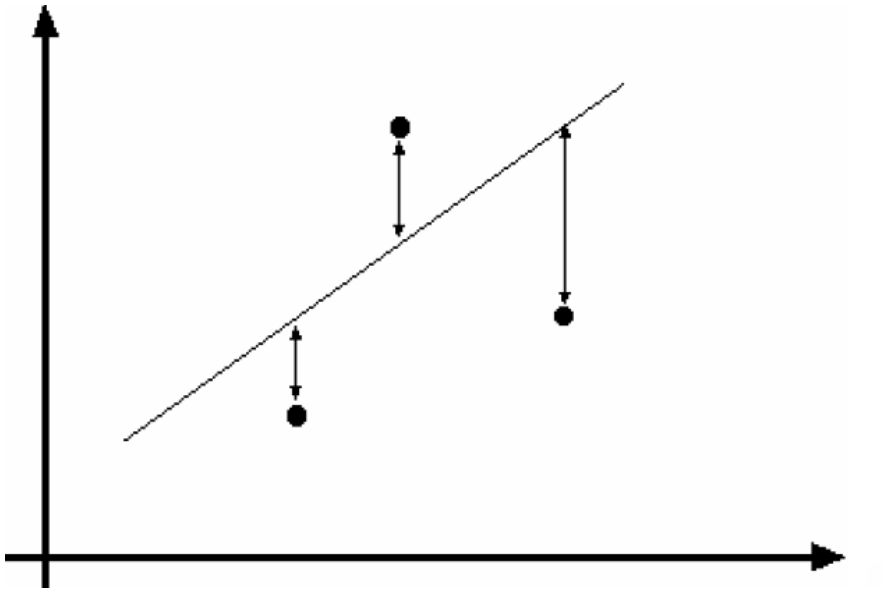
Problem: Outliers

- Outliers can hurt the quality of our parameter estimates, e.g.,
  - An erroneous pair of matching points from two images
  - A feature point that is noise or doesn't belong to the transformation we are fitting.



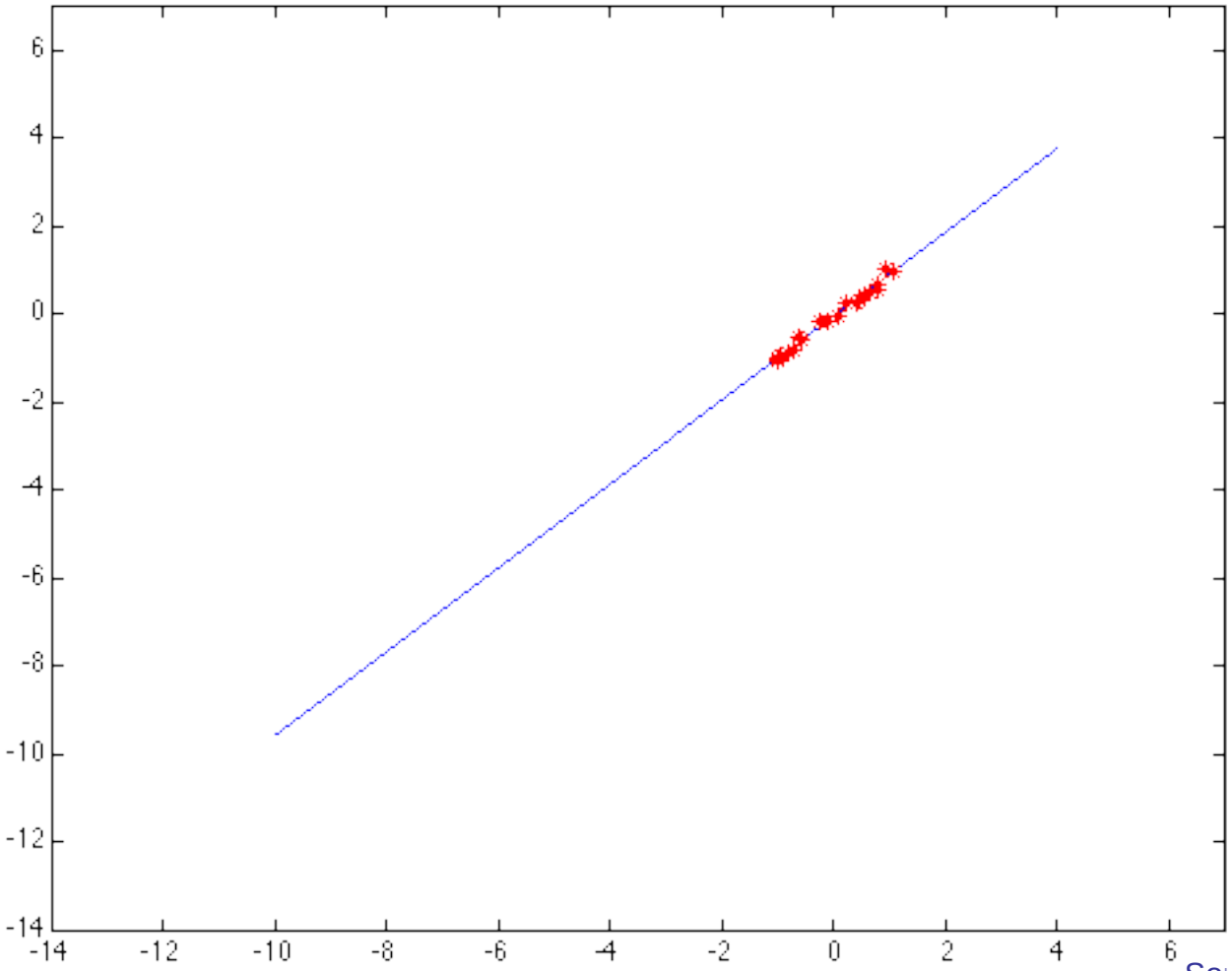
Example: Least-Squares Line Fitting

- Assuming all the points that belong to a particular line are known



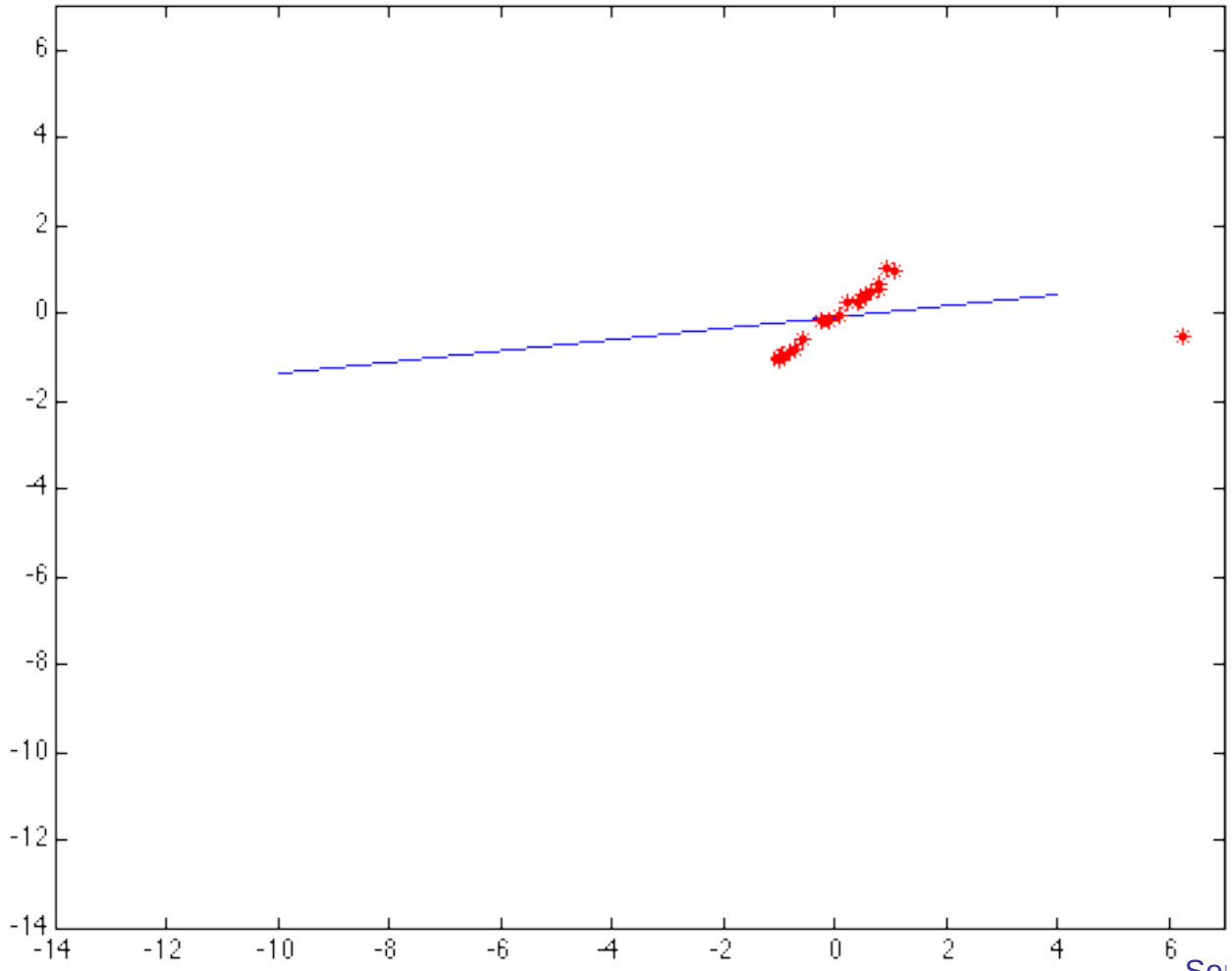
Source: Forsyth & Ponce

Outliers Affect Least-Squares Fit



Source: Forsyth & Ponce

Outliers Affect Least-Squares Fit



Source: Forsyth & Ponce