ELEC 474 – Machine Vision

GEOMETRIC PRIMITIVE EXTRACTION

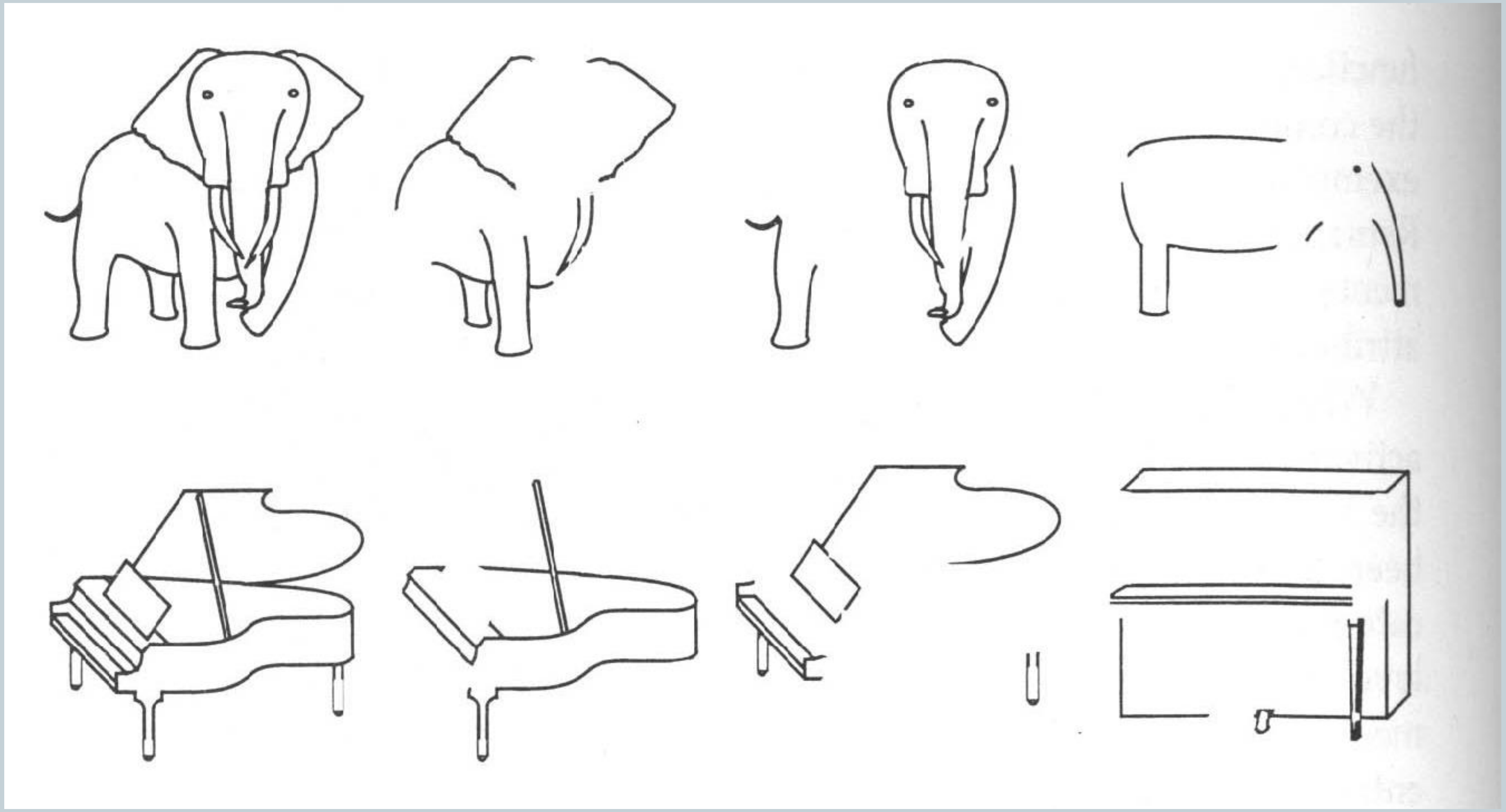
M.T. Ahmed, M. Greenspan

Contents

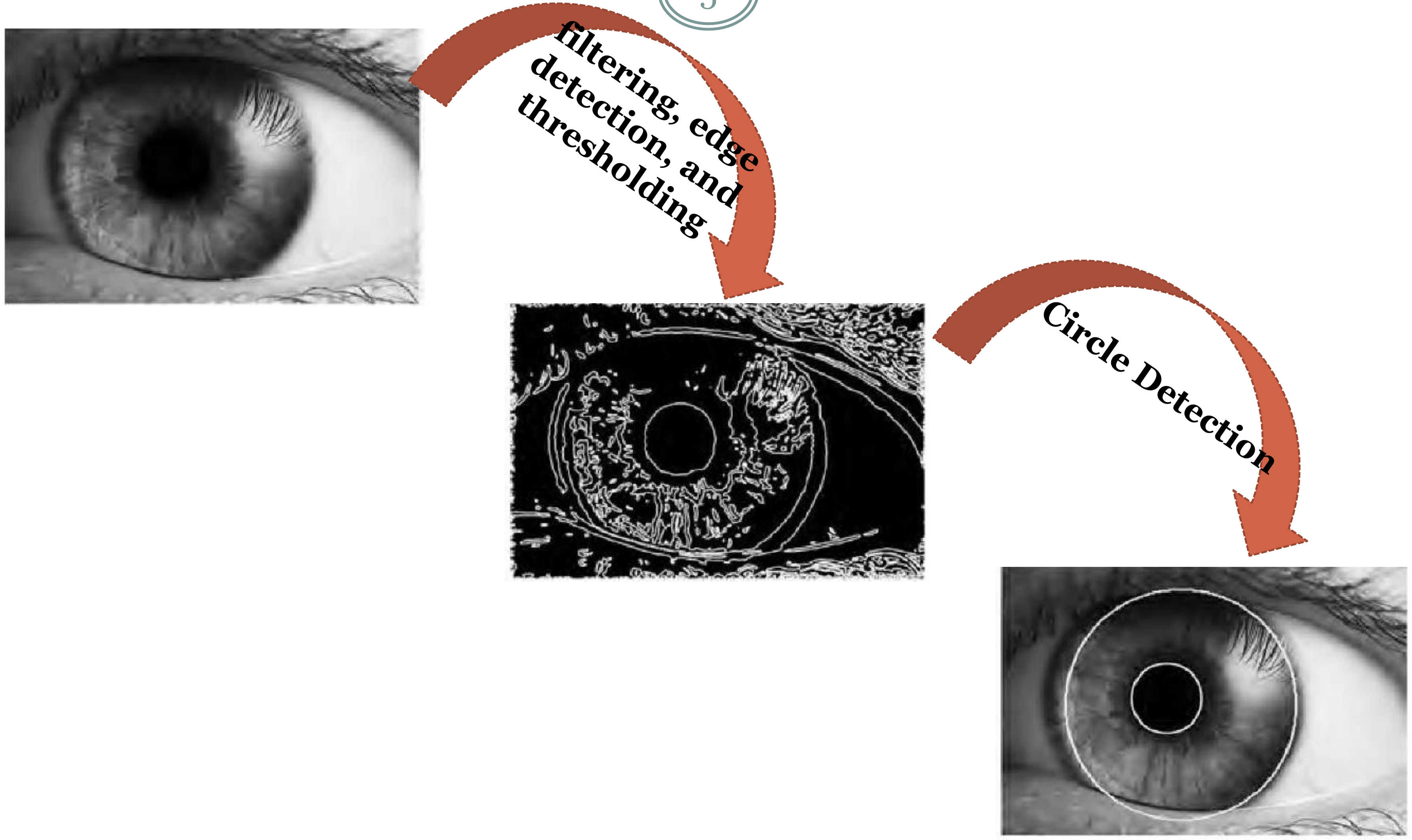
Introduction

- So far
  - Image Enhancement
    - Histogram Processing
    - Image Filters
  - Edges
    - Edge detection
    - Contours extraction
- This lecture
  - Associate more information (properties) to the edges

Introduction



Introduction



Introduction

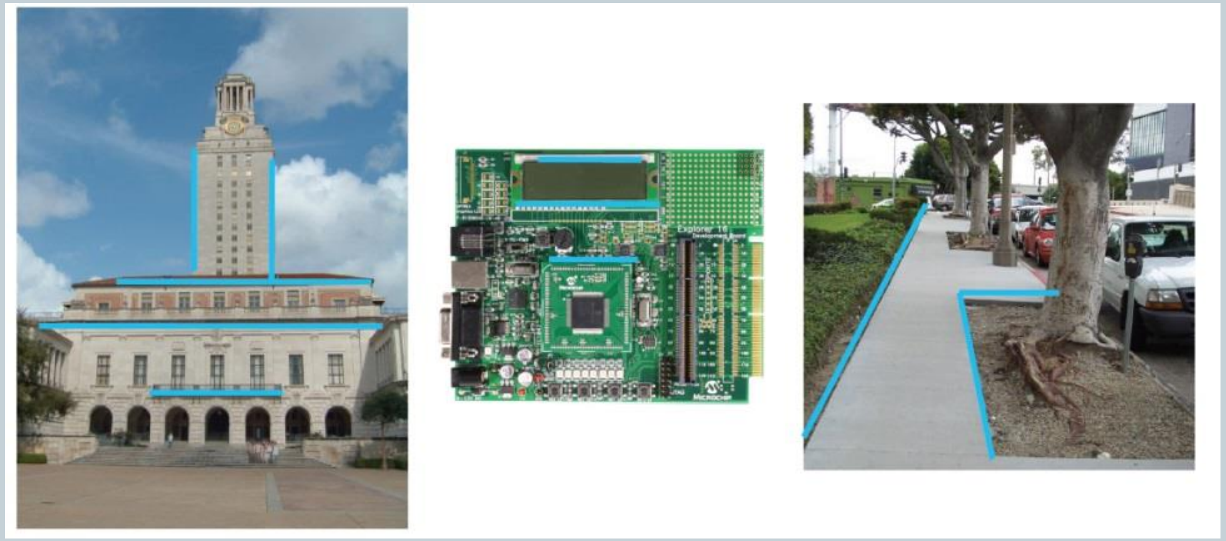
- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - No prior knowledge whether an edge pixel belongs to a given model
- Three main questions
  - What model represents this set of edge pixels best?
  - Which of several model instances gets which edge pixel?
  - How many model instances are there?

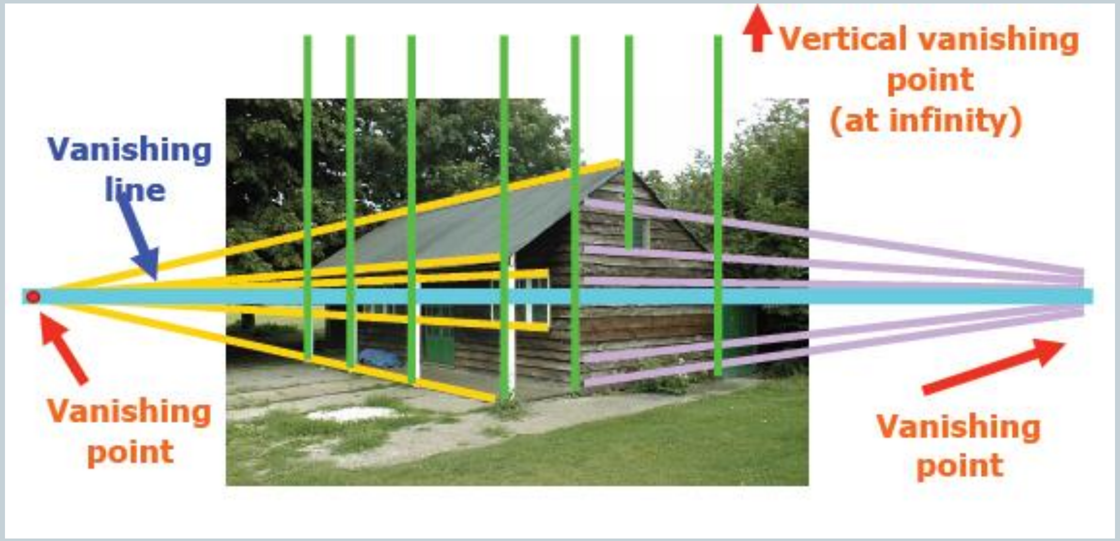
Introduction

- Objective: Aggregate interest points, and/or edge points, and/or contour points into higher level geometric primitives, such as:
    - Lines
    - Curves
    - Circles
    - Ellipses
    - Arbitrary shapes
- Two main algorithms
    - RANSAC
    - Hough Transform

Line Extraction

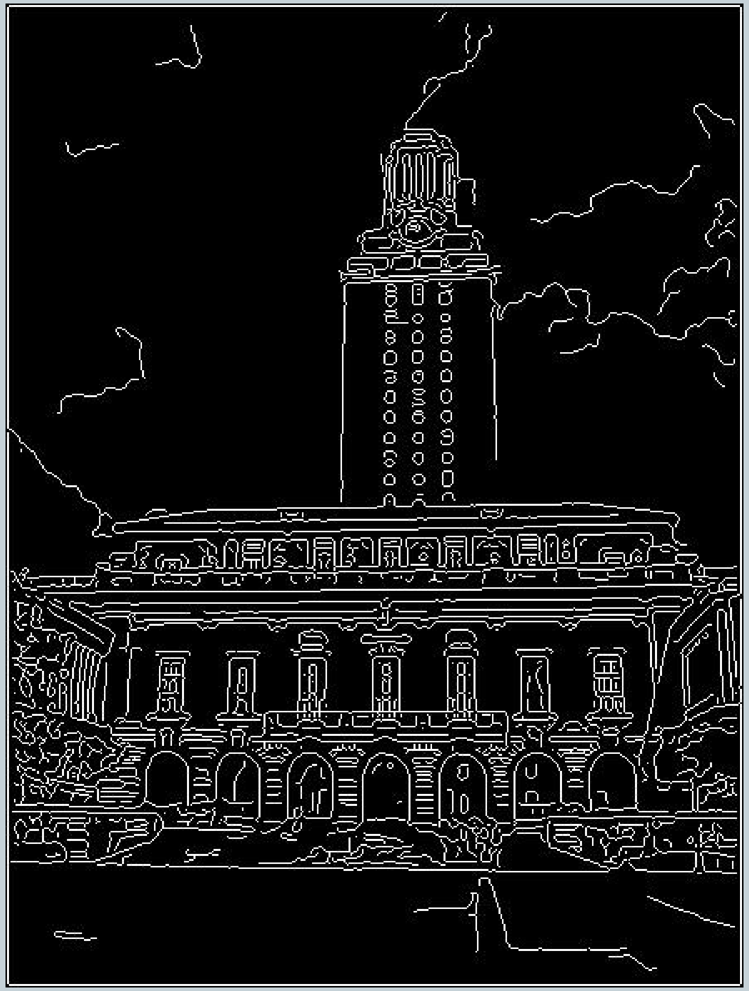
- A line is a higher order representative than an edge
  - Implies some interpretation of image content
- Can be used to recover geometry and viewpoint



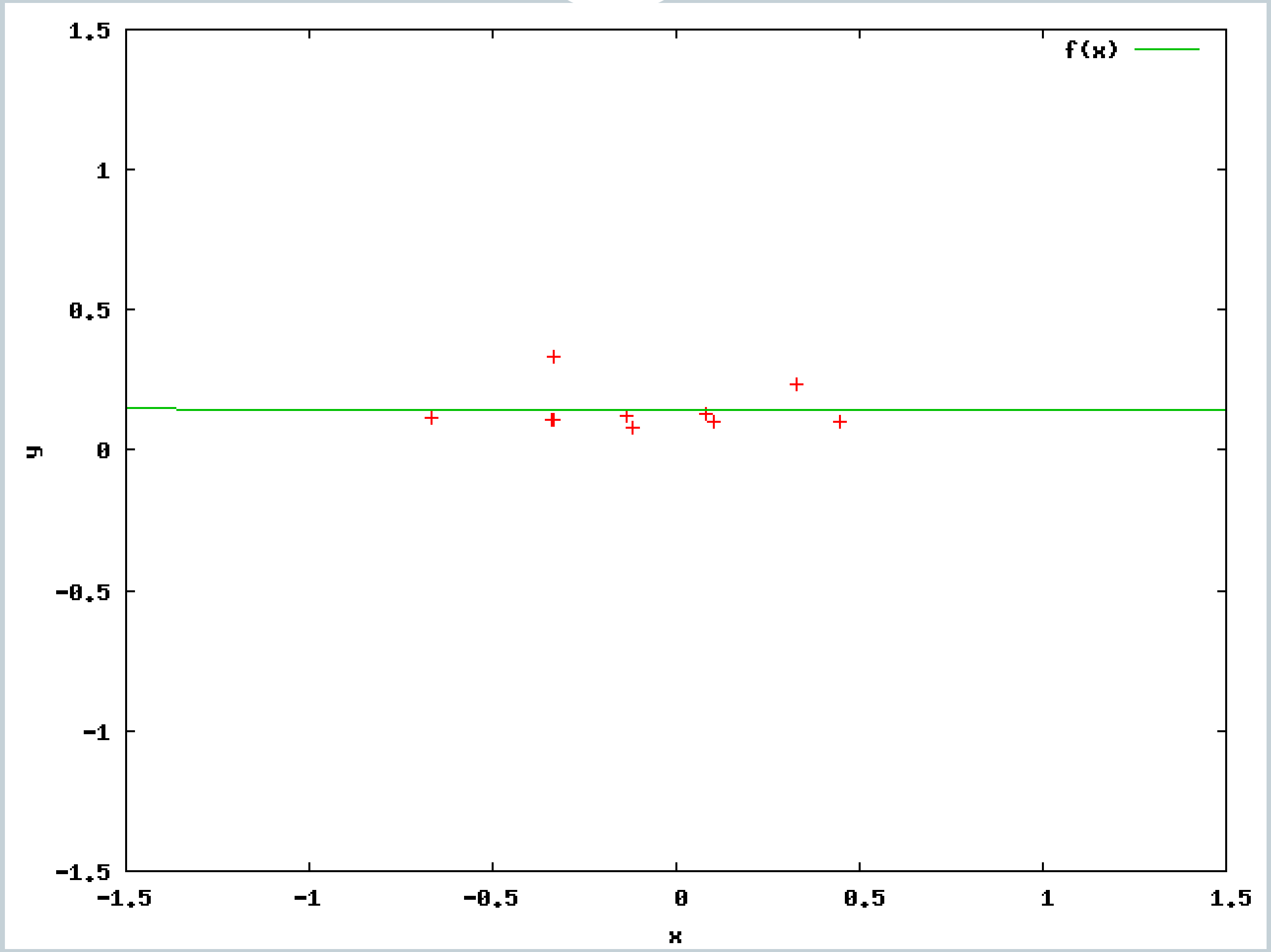


Line Extraction

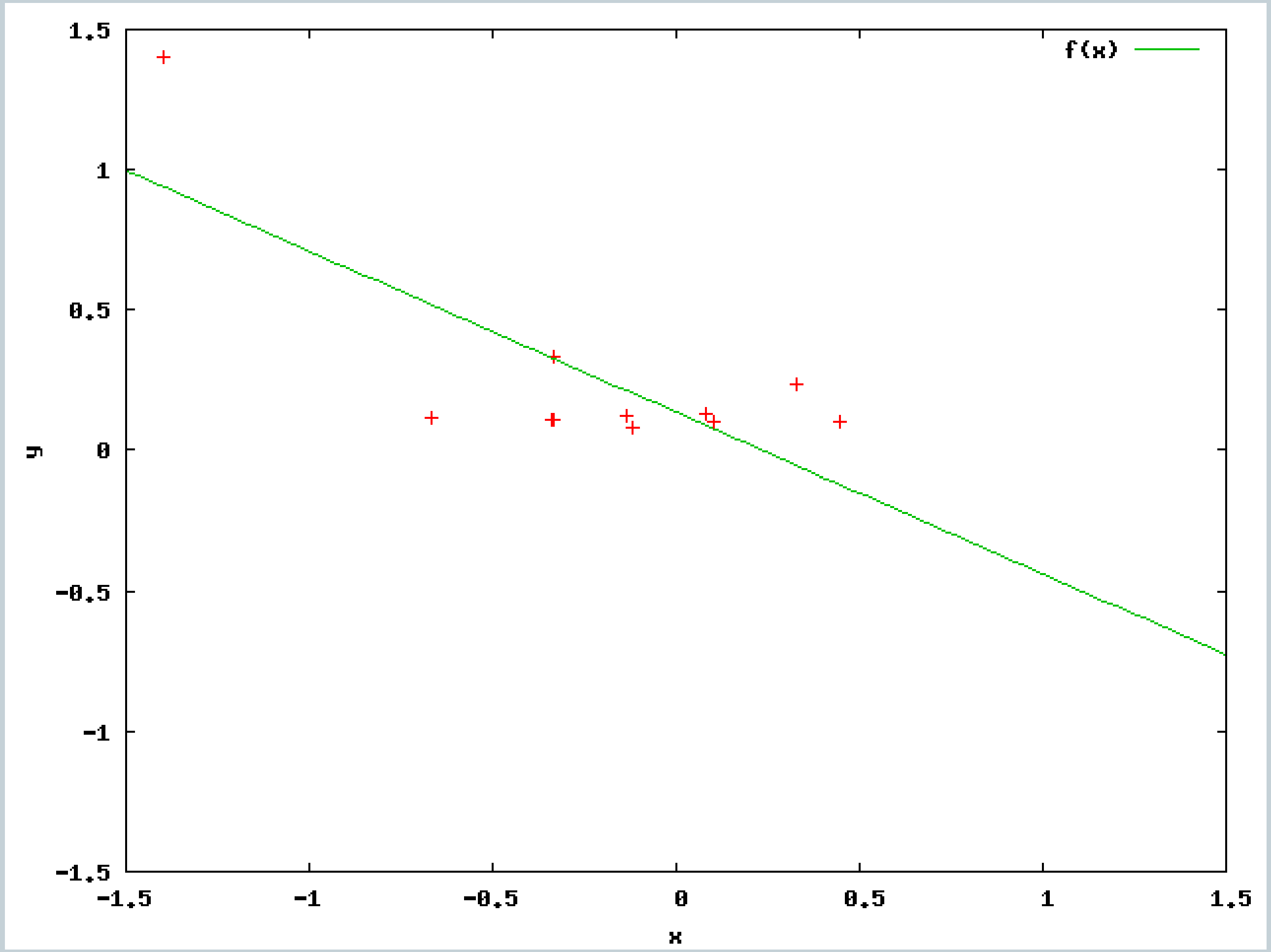
- Challenges
  - Clutter
    - Extra edge points, thus, multiple models
    - Which points go with which line, if any?
  - Missing part of lines
    - How to find a line that bridges missing evidence?
  - Noise in measured edges, orientations
    - How to detect true underlying parameters?



Line Extraction



Line Extraction



Line Extraction

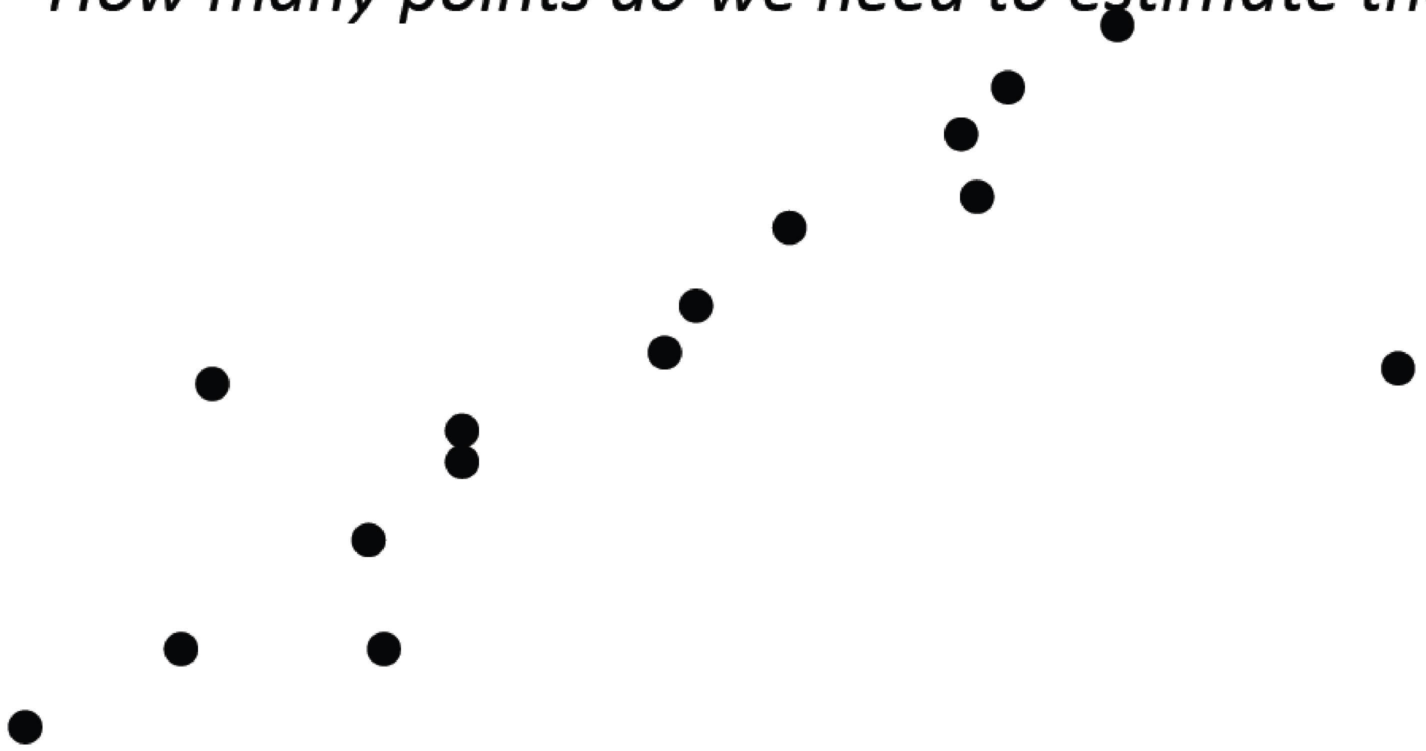
- RANSAC
  - **RAN**dom **SA**mple **C**onsensus
    - a.k.o. Generate-and-Test method
    - Based on Robust Statistics
    - Simple, and surprisingly flexible for many model fitting
  - Idea
    - Discard **outliers**
      - i.e. those points that do not nicely fit the model
    - Only use **inliers**
      - i.e. those points that do nicely fit the model
  - If an outlier is used to hypothesize a model, it will not be supported (voted) by the rest of the features

RANSAC

- Randomly select a seed set of points
  - Only select minimum number of seeds that are required to hypothesize a line
- Compute the transformation from seed set
- Find inliers to this transformation
  - Loop through all other points, and check their distance to the hypothsized line
- Keep the transformation with largest number of inliers
- Optional refinement
  - re-compute least-square estimate of final transformation using inliers only

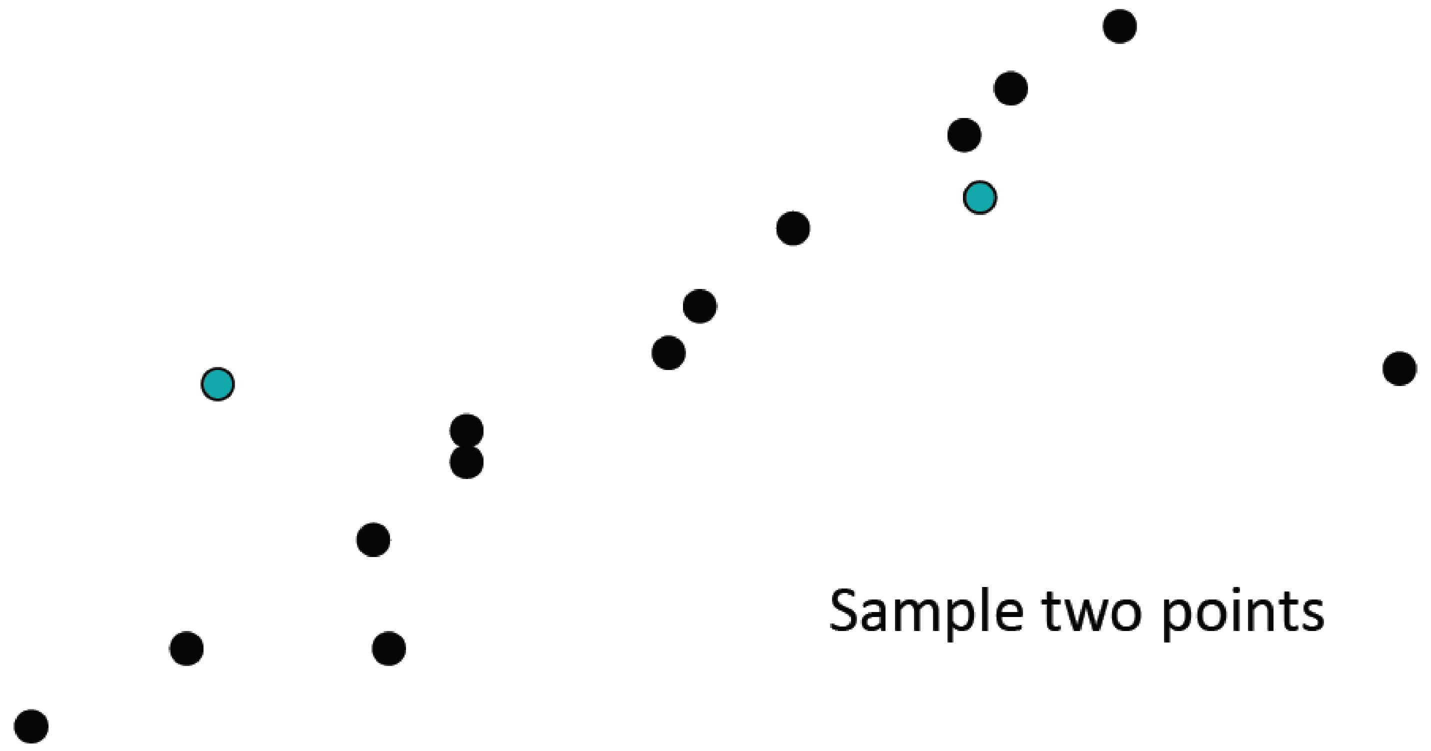
RANSAC for line extraction

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



RANSAC for line extraction

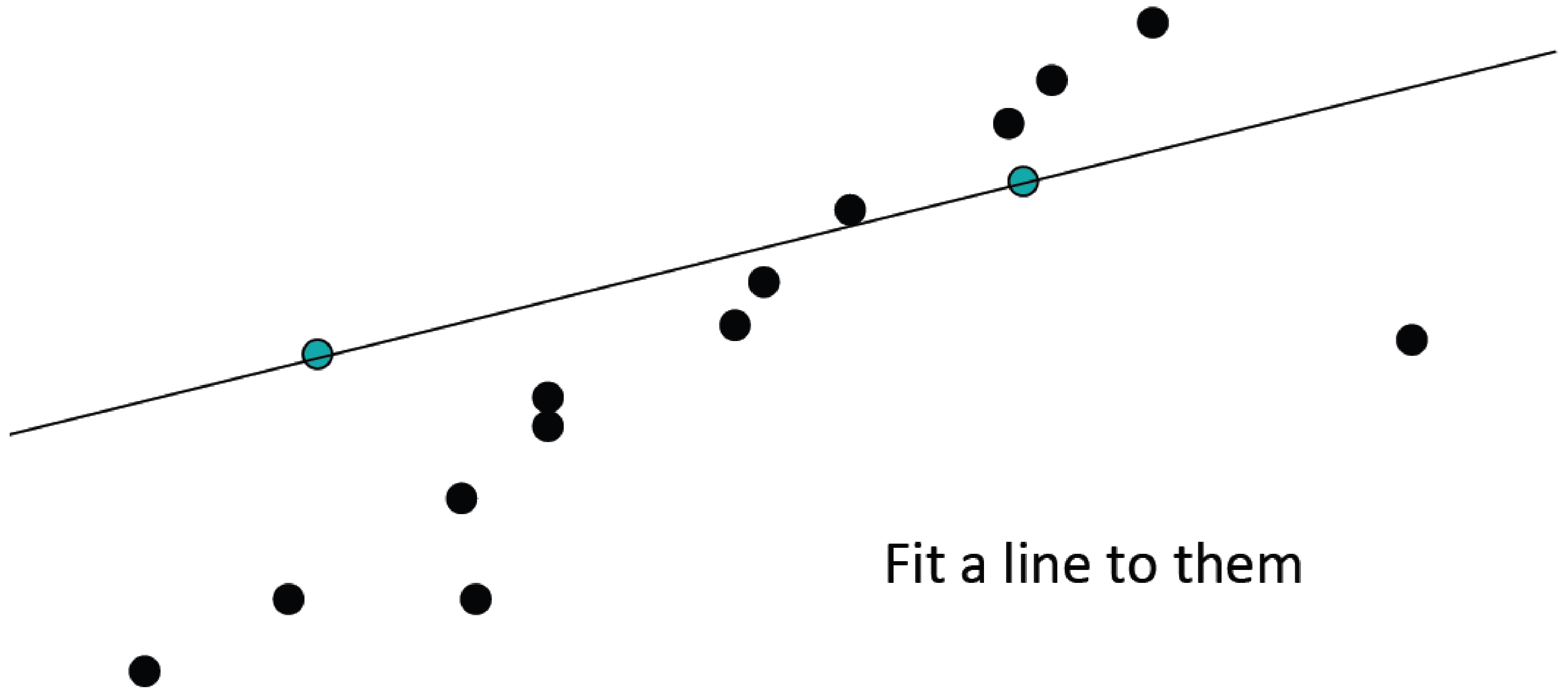
- Task: Estimate the best line



Sample two points

RANSAC for line extraction

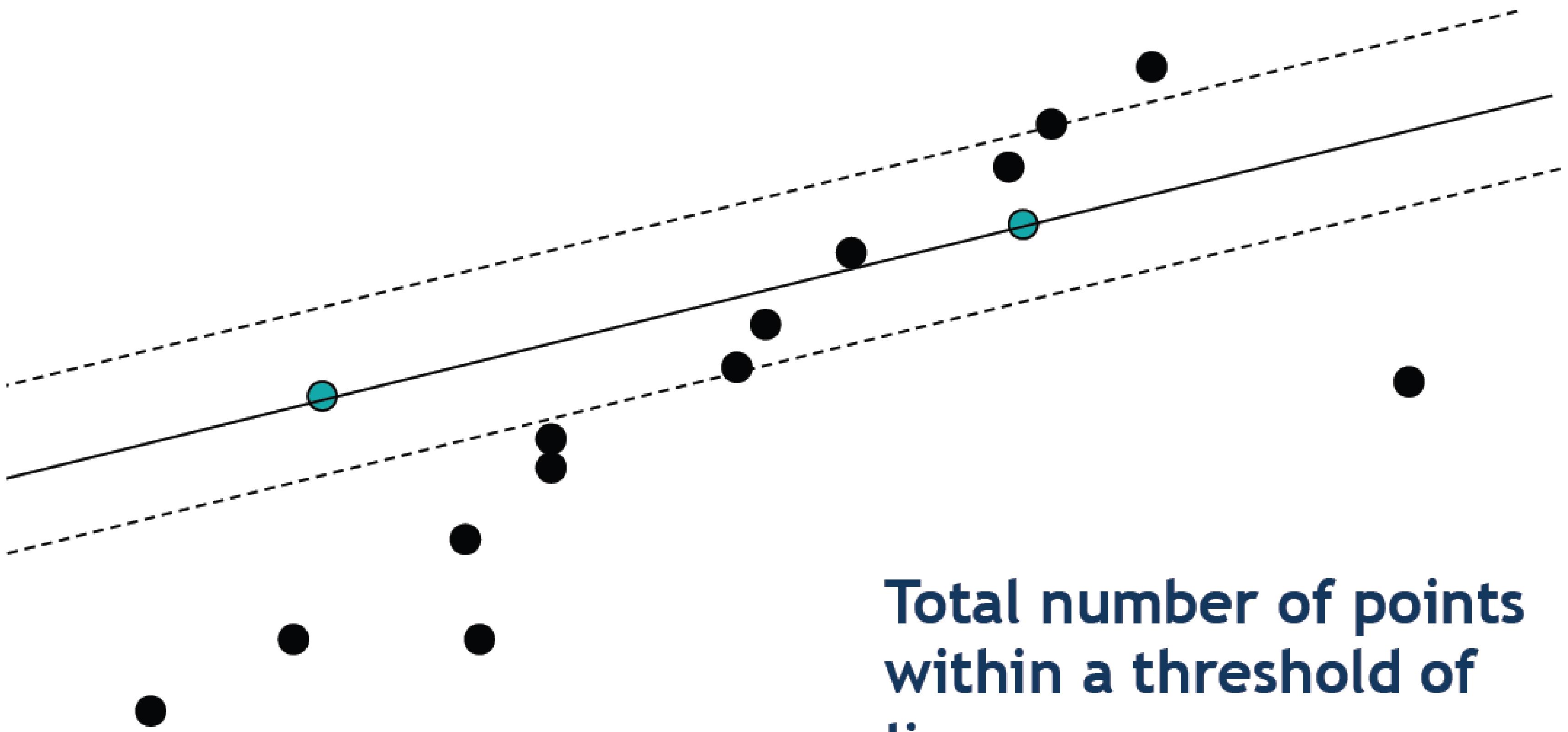
- Task: Estimate the best line



Fit a line to them

RANSAC for line extraction

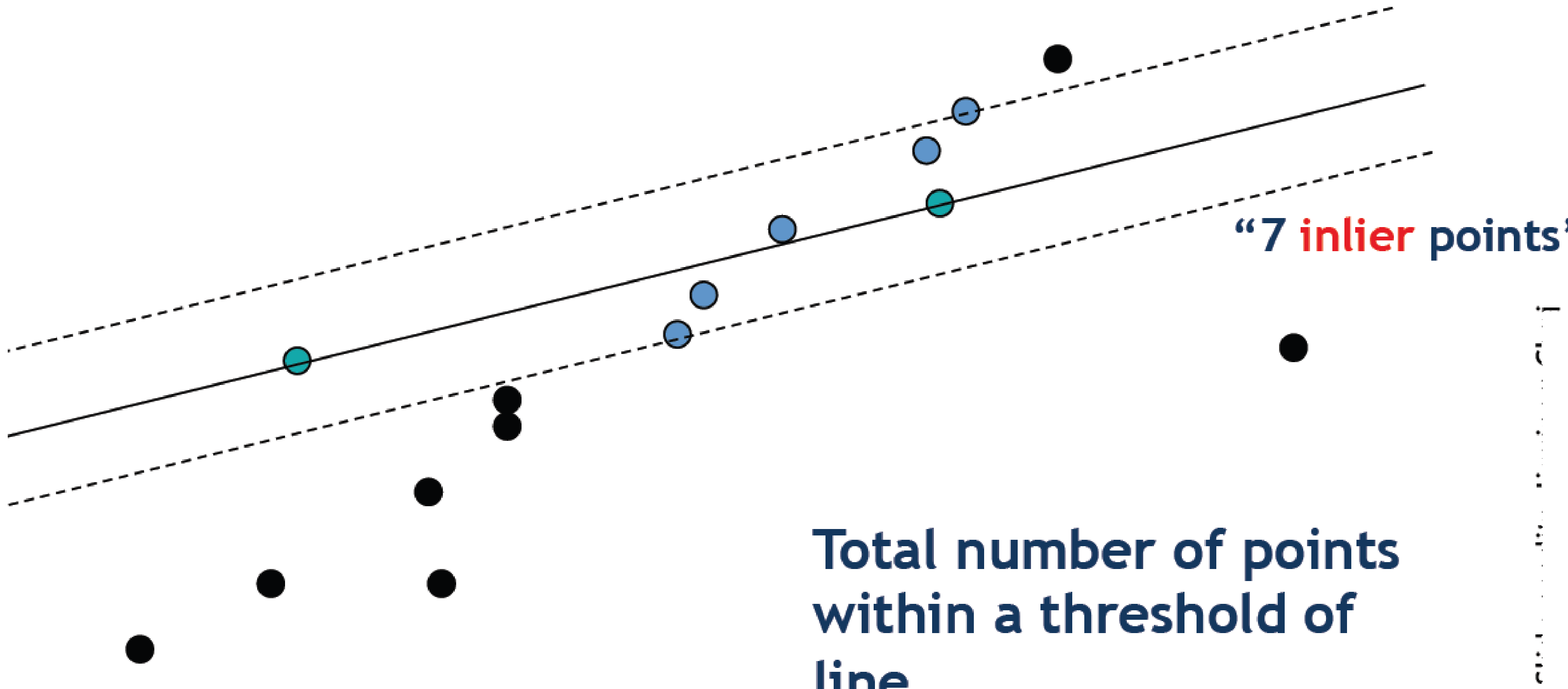
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC for line extraction

- Task: Estimate the best line

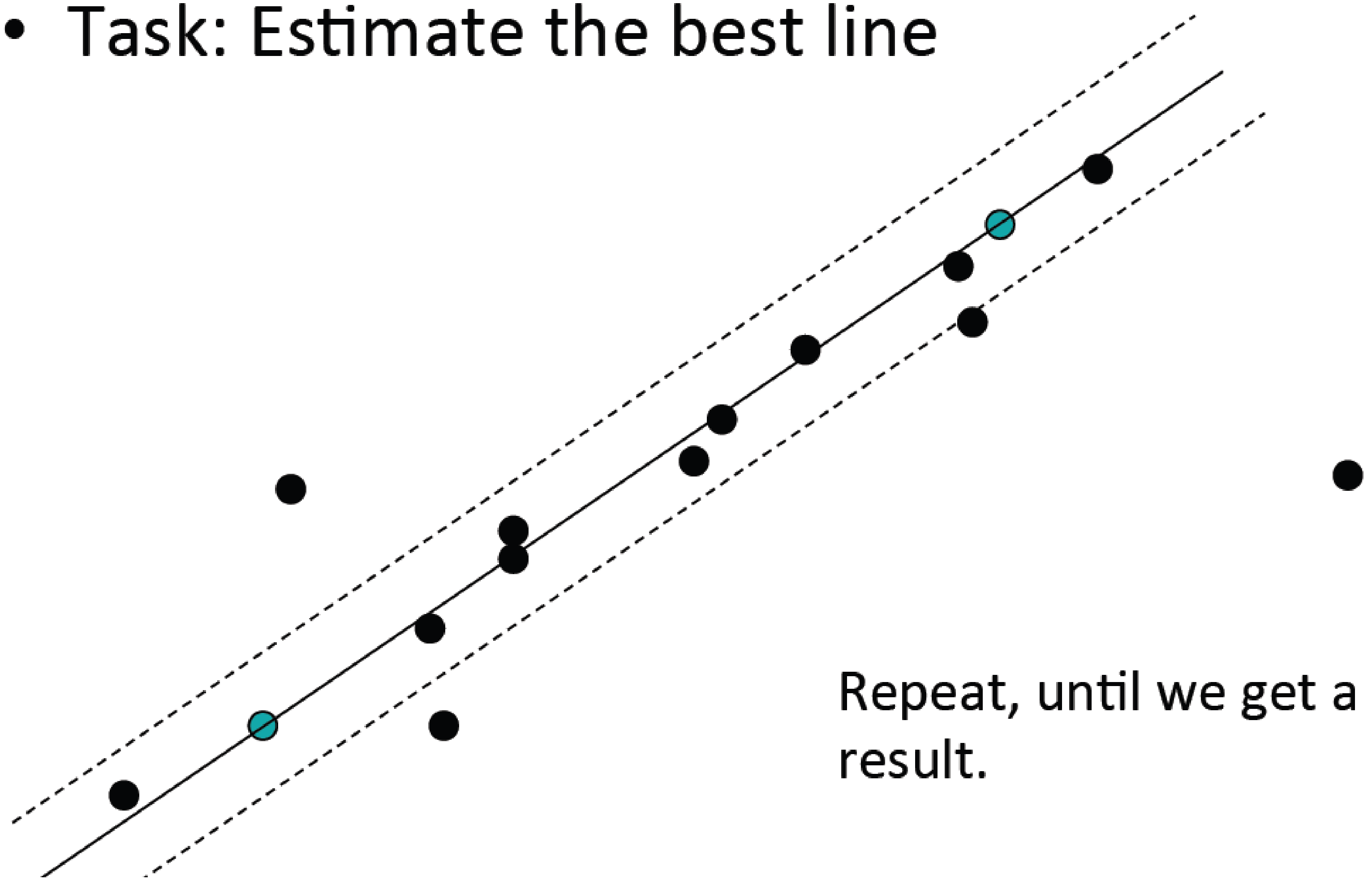


“7 inlier points”

Total number of points within a threshold of line.

RANSAC for line extraction

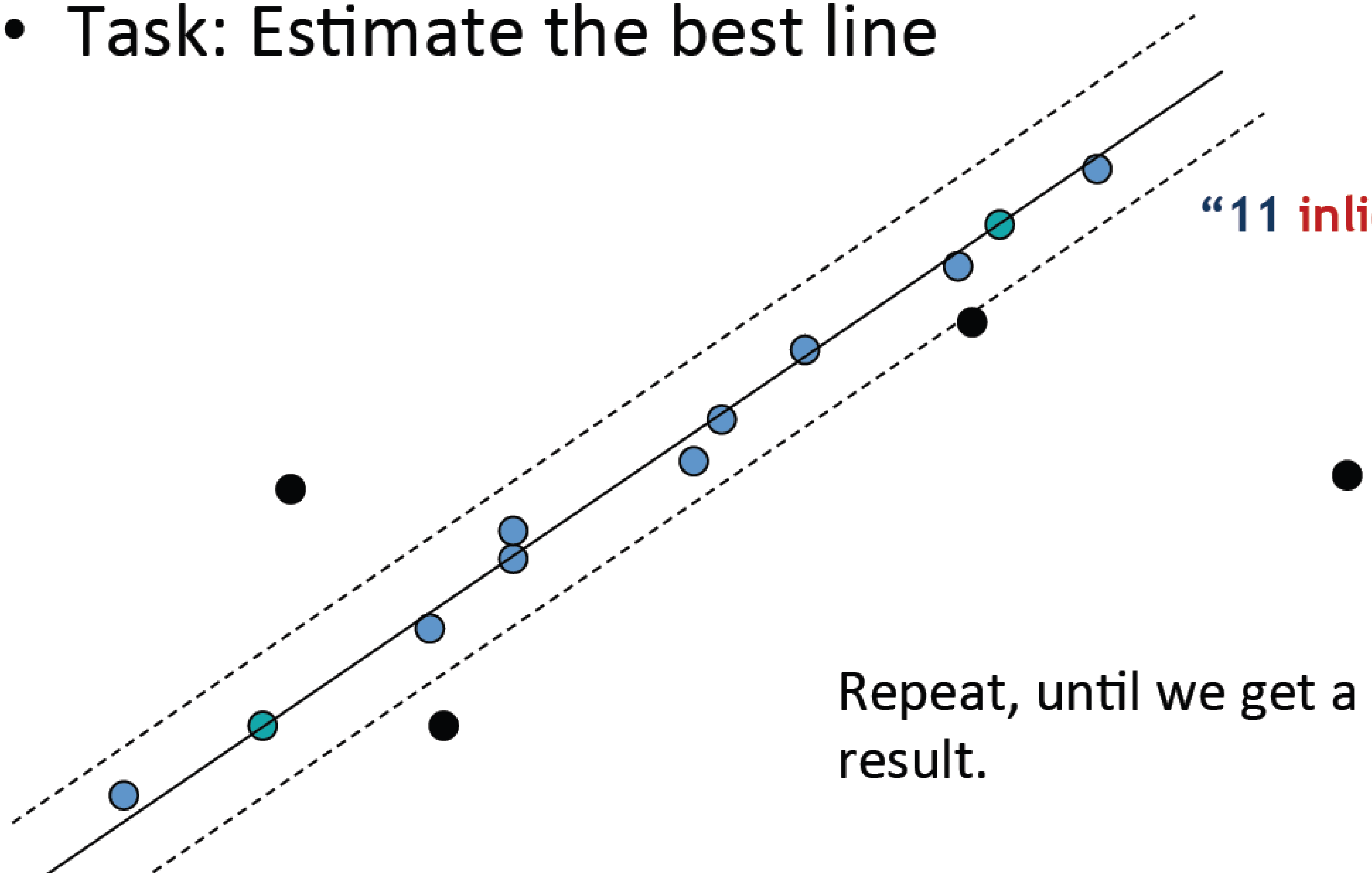
- Task: Estimate the best line



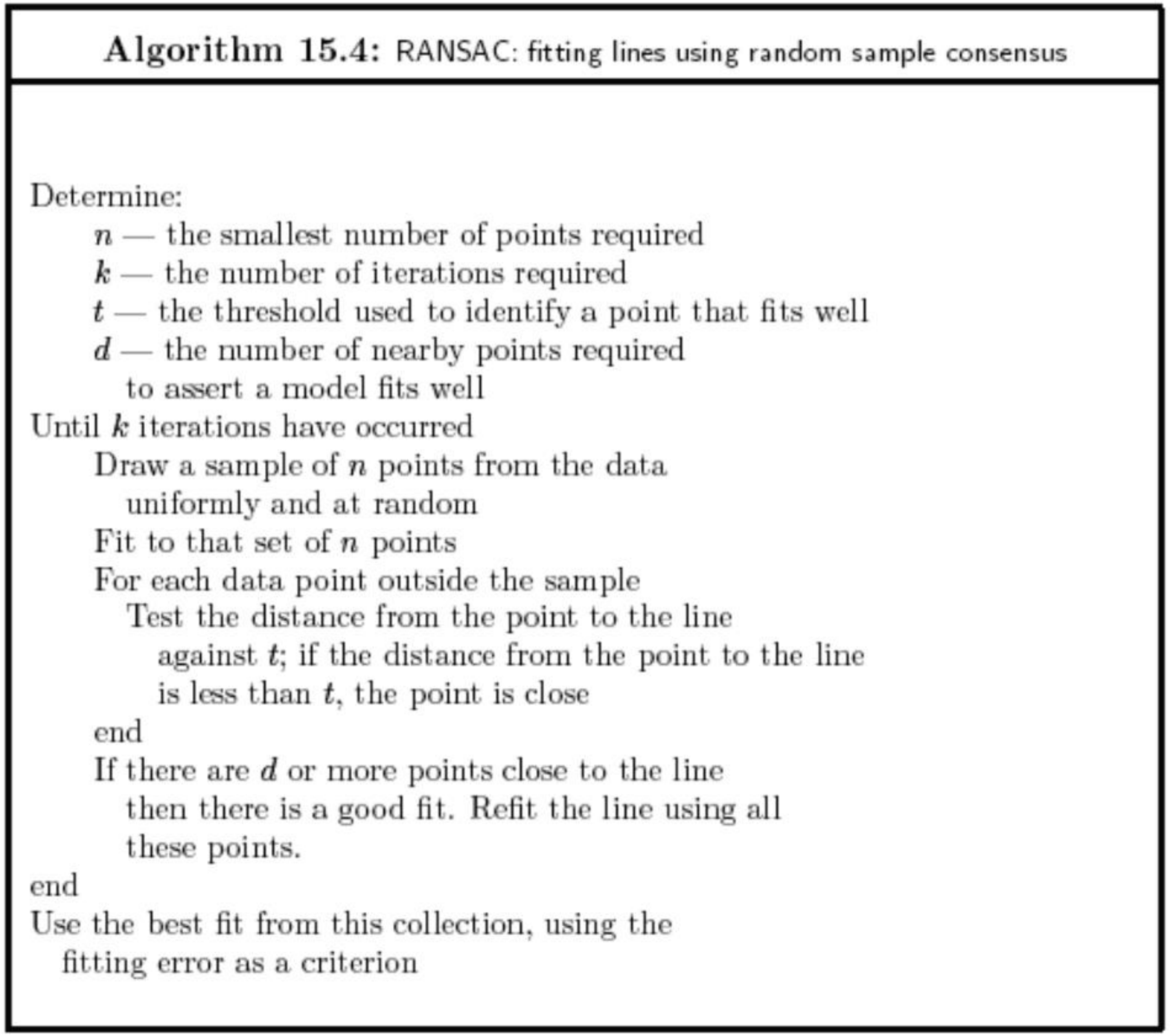
Repeat, until we get a good result.

RANSAC for line extraction

- Task: Estimate the best line



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
```

RANSAC

- How many random samples are needed?
  - Let ω is the probability of choosing an inlier each time a single point is selected

$$\omega = \frac{total\ inliers}{total\ points}$$

  - n = points required to define a model
    - for line $n = 2$
  - k = number of samples chosen randomly

RANSAC

- Probability that a single sample of n is an inlier:

$$P_i = \omega^n$$

- Probability that all k samples fail:

$$P_f = (1 - \omega^n)^k$$

- k must be chosen as high enough to keep this below desired failure rate

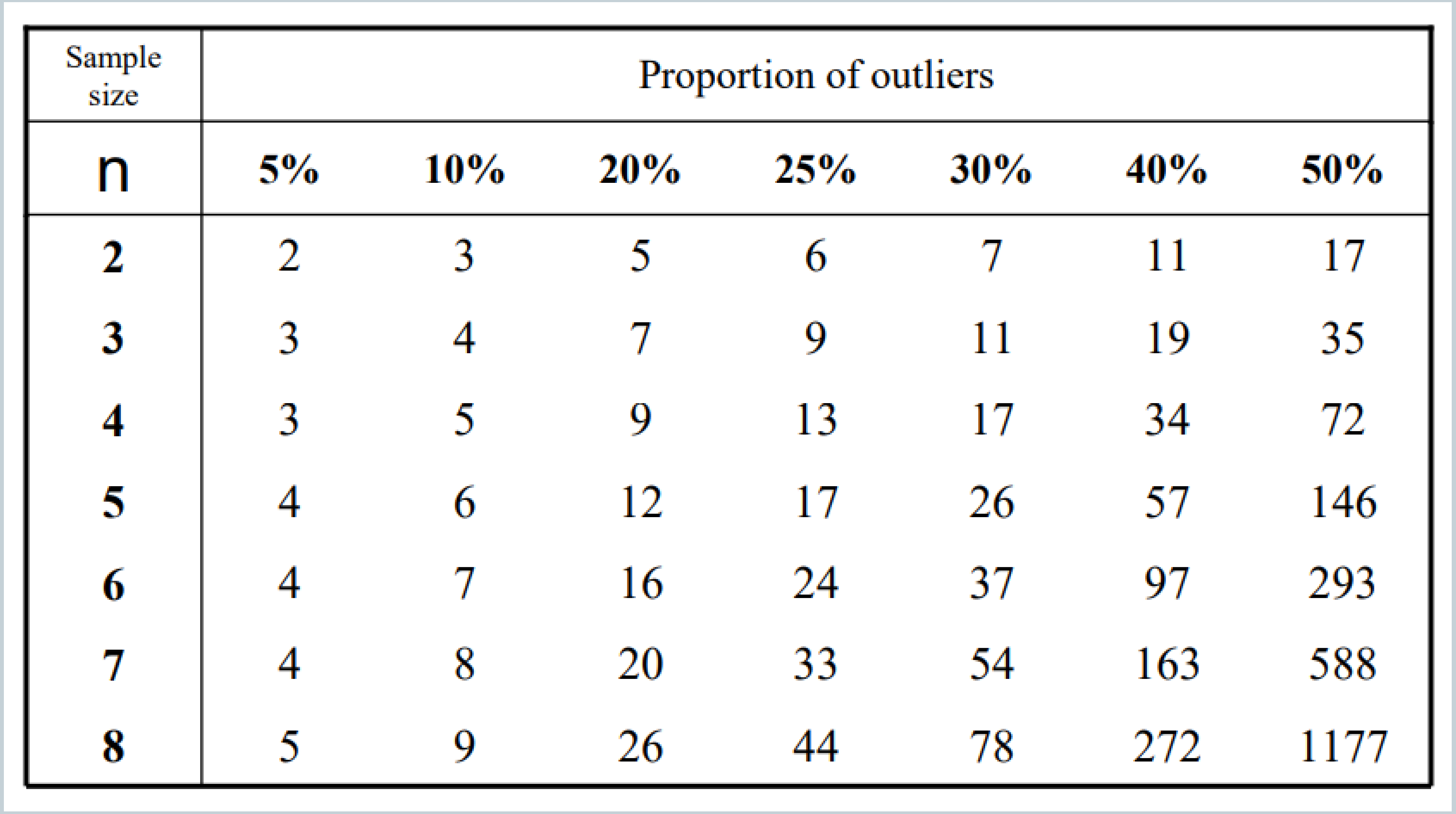
RANSAC

k	w	Pf	Ps=1-Pf
1	0.75	0.4375	0.5625
2	0.75	0.191406	0.808594
3	0.75	0.08374	0.91626
4	0.75	0.036636	0.963364
5	0.75	0.016028	0.983972
6	0.75	0.007012	0.992988
7	0.75	0.003068	0.996932
8	0.75	0.001342	0.998658
9	0.75	0.000587	0.999413
10	0.75	0.000257	0.999743
100	0.75	1.25E-36	1
1000	0.75	0	1

k	w	Pf	Ps=1-Pf
1	0.5	0.75	0.25
2	0.5	0.5625	0.4375
3	0.5	0.421875	0.578125
4	0.5	0.316406	0.683594
5	0.5	0.237305	0.762695
6	0.5	0.177979	0.822021
7	0.5	0.133484	0.866516
8	0.5	0.100113	0.899887
9	0.5	0.075085	0.924915
10	0.5	0.056314	0.943686
100	0.5	3.21E-13	1
1000	0.5	1.2E-125	1

k	w	Pf	Ps=1-Pf
1	0.1	0.99	0.01
2	0.1	0.9801	0.0199
3	0.1	0.970299	0.029701
4	0.1	0.960596	0.039404
5	0.1	0.95099	0.04901
6	0.1	0.94148	0.05852
7	0.1	0.932065	0.067935
8	0.1	0.922745	0.077255
9	0.1	0.913517	0.086483
10	0.1	0.904382	0.095618
100	0.1	0.366032	0.633968
1000	0.1	4.32E-05	0.999957

RANSAC: Computed k (Ps = 0.99)



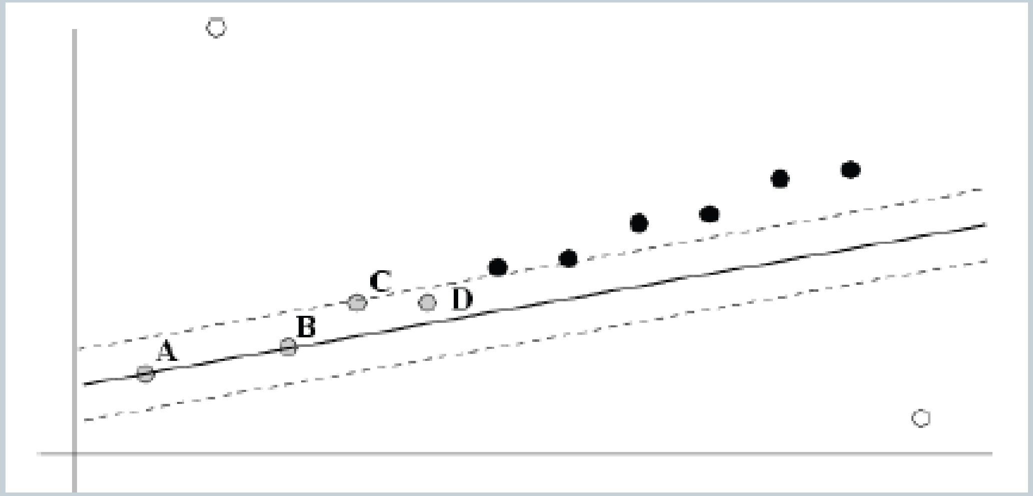
Sample size	Proportion of outliers						
n	**5%**	**10%**	**20%**	**25%**	**30%**	**40%**	**50%**
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

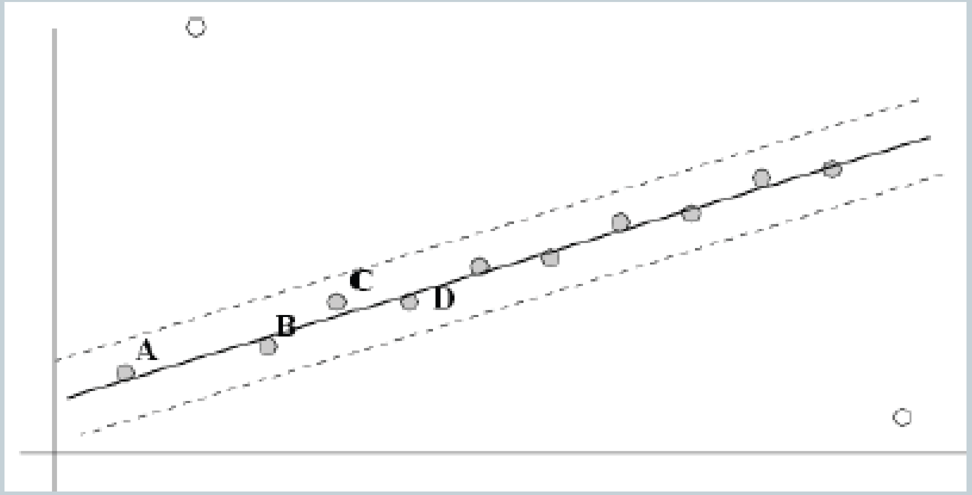
RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.

- For final result, improve the estimate using all inliers
  - e.g. with standard least-squares minimization

- This may change inliers, so alternate fitting with re-classification as inlier/outlier.





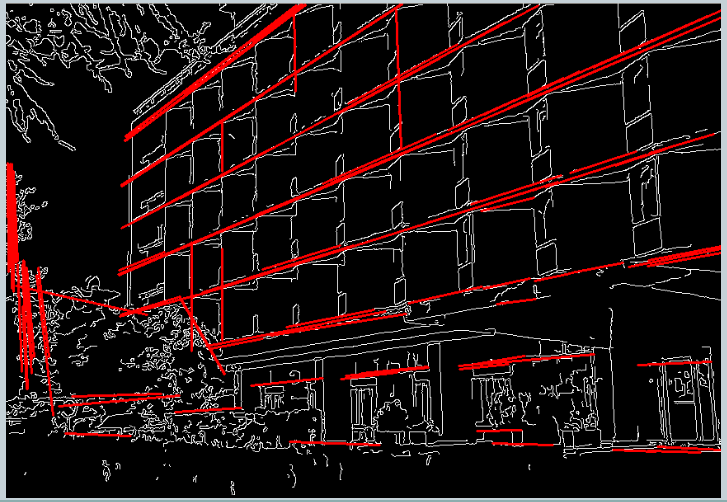
RANSAC

- **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
    - especially true for higher values of n
  - Many real problems have high rate of outliers
    - Sometimes, a selective choice of random subsets can also help

Hough Transform

- There is a very elegant and powerful technique to extract lines (and other primitives) called the ***Hough Transform***
- An alternative to RANSAC, that can be more time efficient
  - Although less space efficient
- Very well studied
  - A large collection of techniques



Voting

- It’s not feasible to check all combinations of features by fitting a model to each possible subset.

- Voting is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.

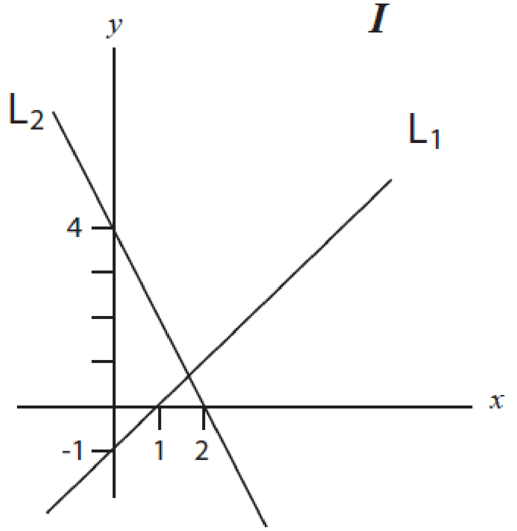
- Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.

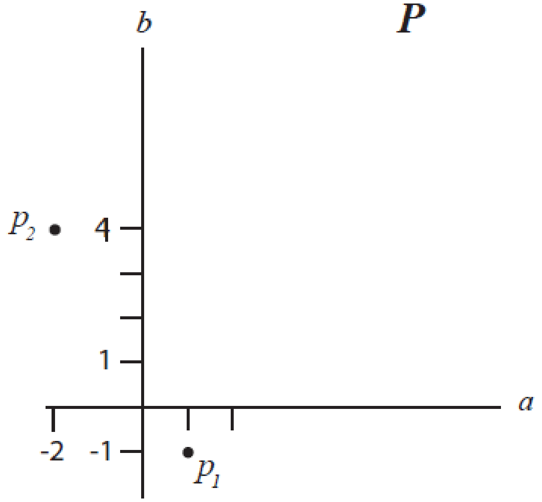
Hough Transform - Lines

- For an image $I = x \times y$, any line $L_i \in I$ can be describe as

$$y_i = a_i x_i + b_i$$

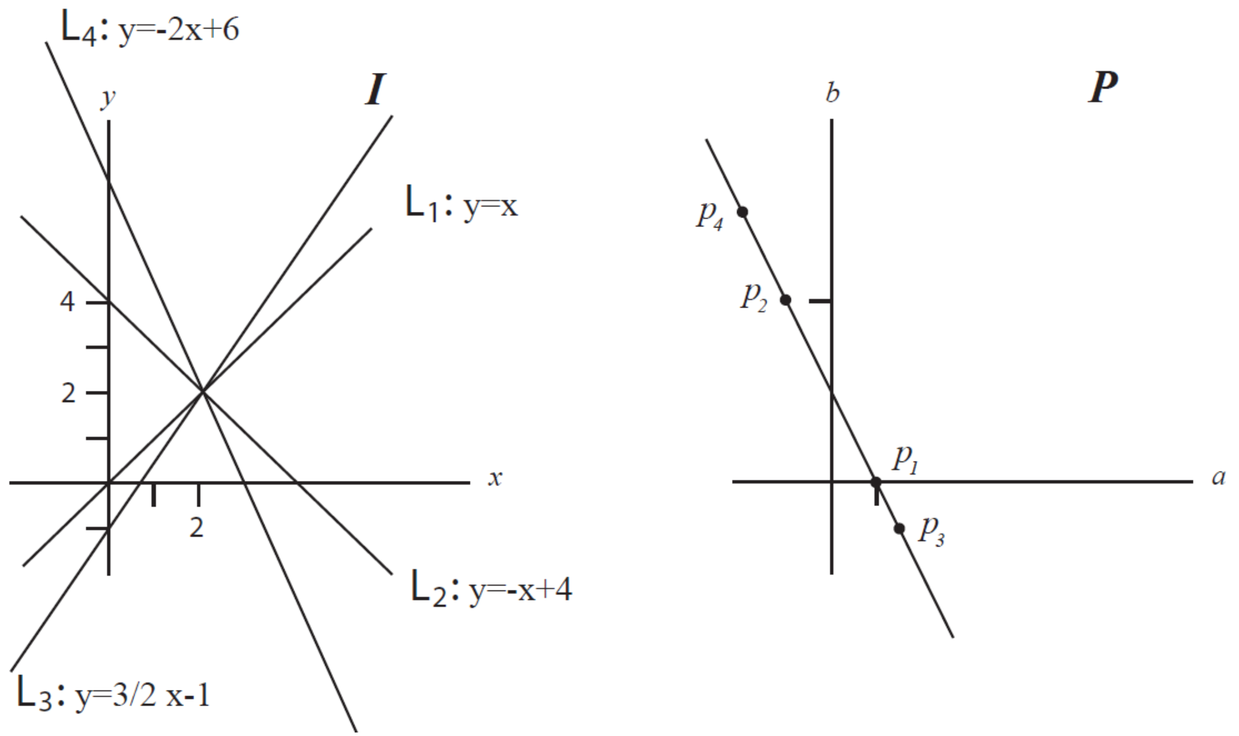
- A dual parameter space $P = a \times b$ where every point $p_i \in P$ corresponds to some line $L_i \in I$





Hough Transform - Lines

- Consider a point $m_i = (x_i, y_i)$
  - There exist a family of lines that intersect at m_i
  - Each line will map to a point $p_i \in P$

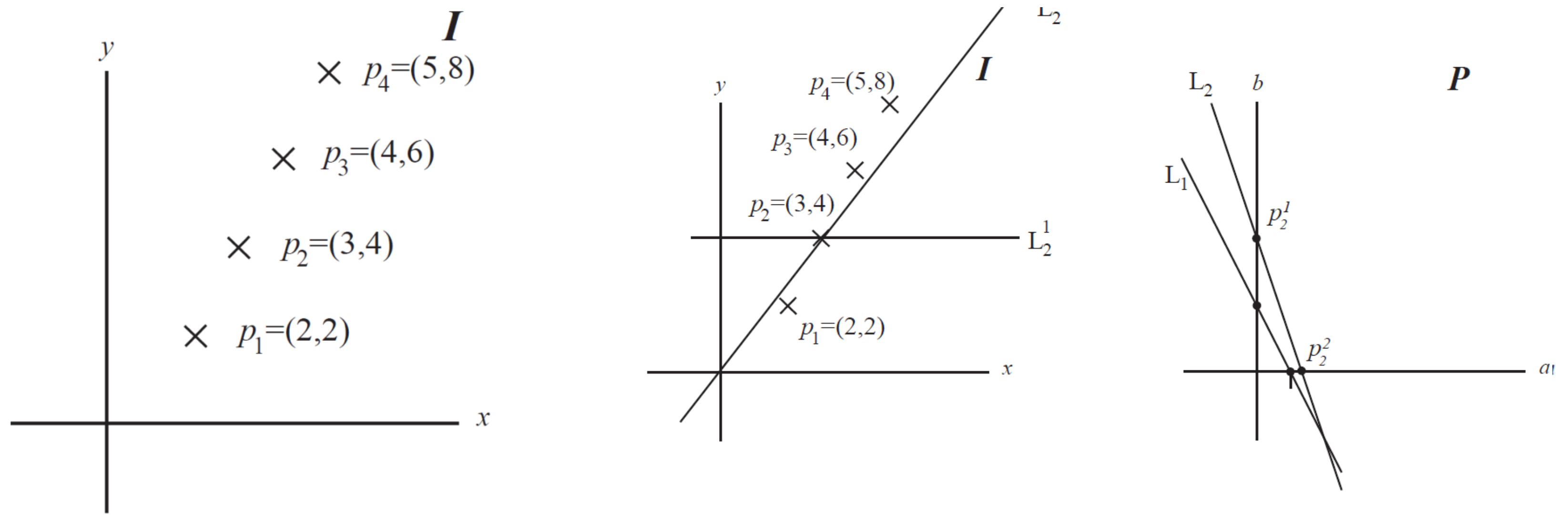


Hough Transform – Lines

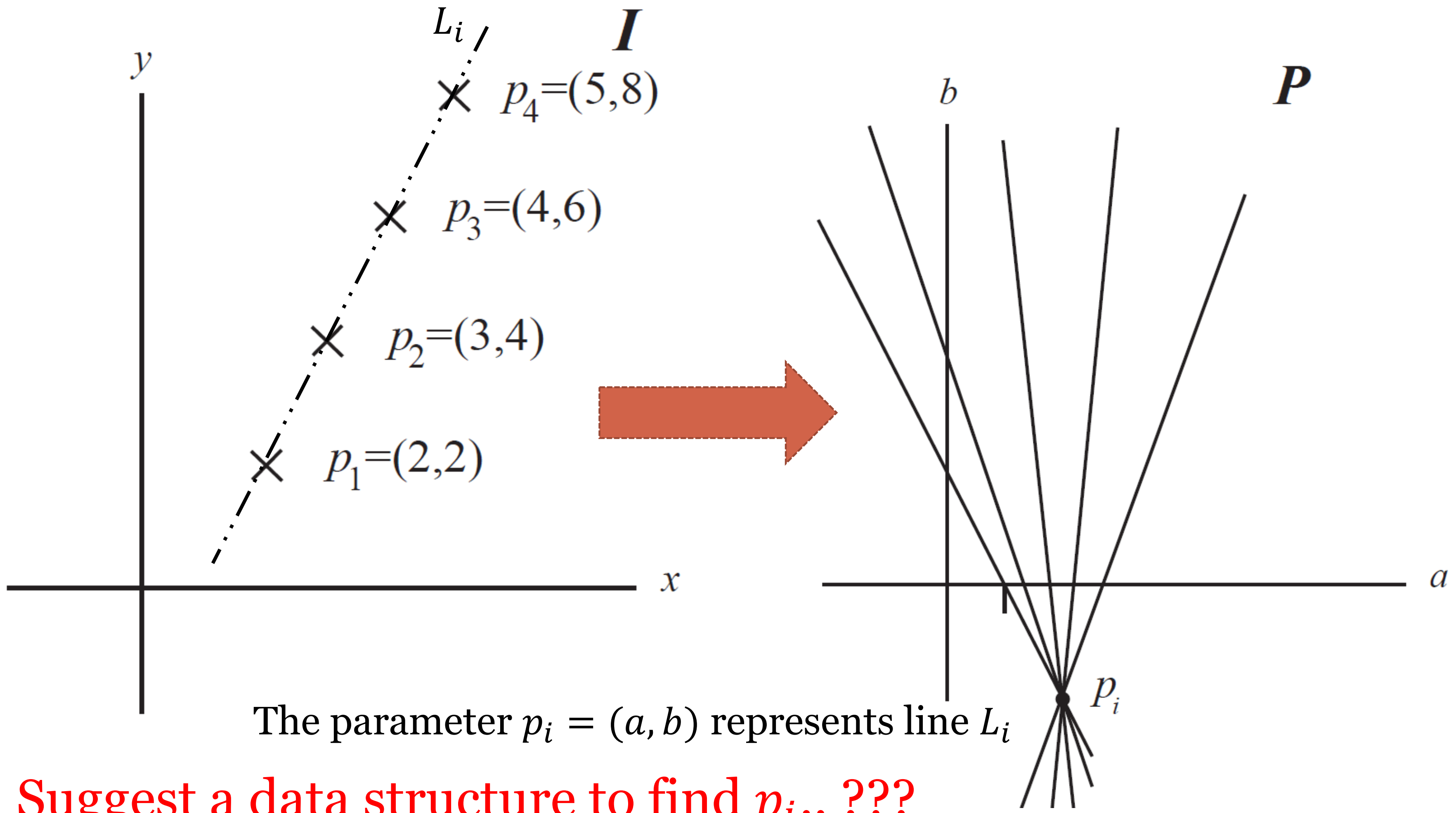
- The relationship between lines and points in two spaces is given as

$$L_i \in I \iff p_i \in P$$

$$p_i \in I \iff L_i \in P$$



Hough Transform – Lines



The parameter $p_i = (a, b)$ represents line L_i

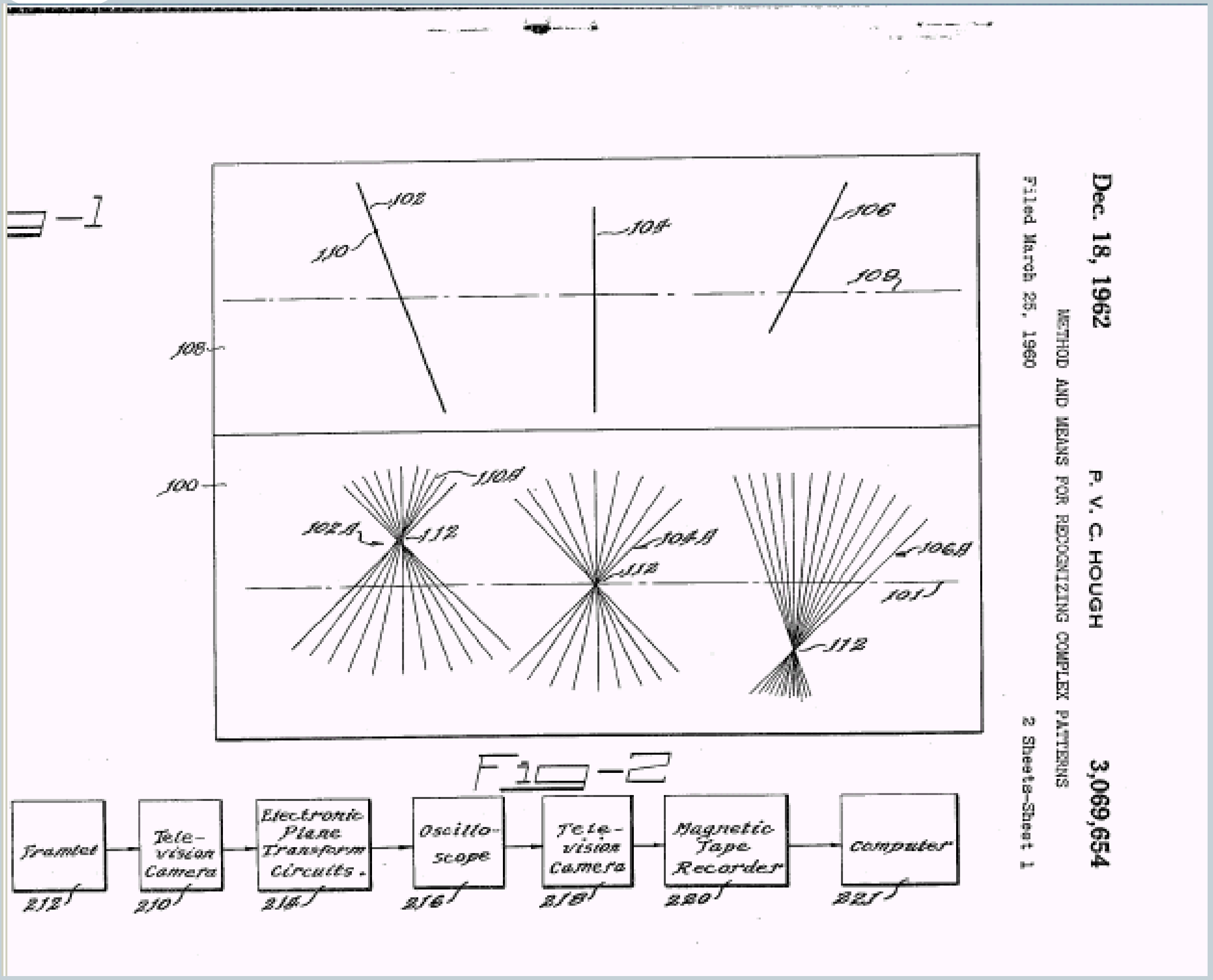
Suggest a data structure to find p_i.. ???

Hough Transform – Lines

- Algorithm
  1. For each $p_i \in I$, the associated $L_i \in P$ is calculated
  2. Every point $p_j \in P$ that intersects with L_i has its associated bin values incremented
     - Following this, bins with high values represent possible lines in I
  3. The parameter space P is thresholded, and the line equations are extracted
     - Highest peak in parameter space is best line in image space
     - Second highest peak is next best line, etc.

Hough Transform

- The Hough Transform was patented by Hough in 1962.
- It was subsequently introduced to the Computer Vision community by Duda and Hart in 1969.
- Throughout the 1970s and 1980s, the Hough Transform was one of the most well-explored methods in Computer Vision, with a multitude of variation and applications.



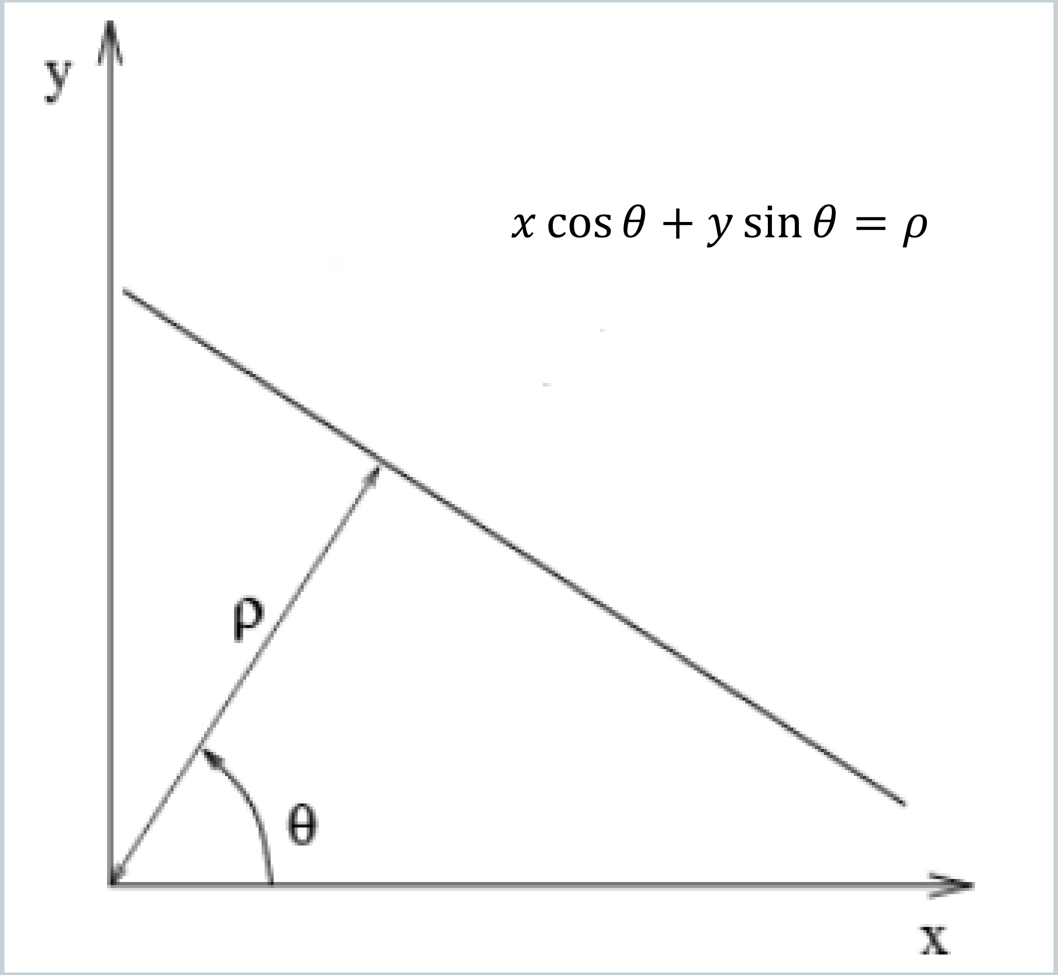
Hough Transform – Lines

- Pros
  - Robust: tolerant to dropouts and occlusions
    - The line doesn’t have to be connected or continuous
    - robust to outliers
  - Efficient: $O(mn)$
    - With n is number of projected points (features) in image
    - m is the maximum length of a line in the parameters space
    - Peak detection is achieved with a linear search through parameter space
- Cons
    - $a, b \in [-\infty, \infty]$, not possible to bin over this entire space
    - Vertical lines require infinite a

What is the solution?

Hough Transform – Lines

- Use Polar representation
  - A line $L \in I$ is represented uniquely by the perpendicular distance ρ from the line L to the origin and angle θ between ρ and the x-axis.



Hough Transform – Lines

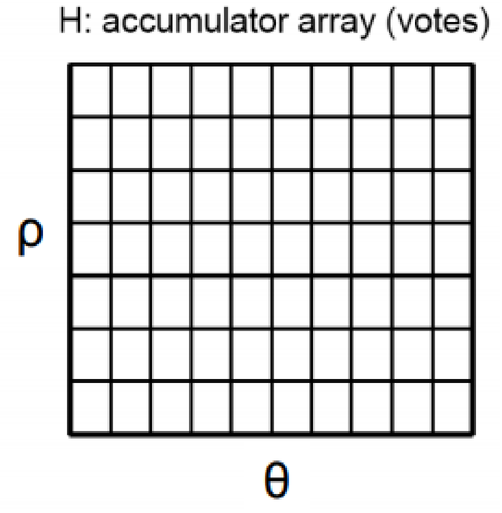
- Under the polar parametrization, a point $p \in I$ maps to a sinusoidal curve in P. The intersection of a family of such sinusoids in P still represents a line in I
- The main advantage of the polar parametrization is that both ρ and θ are bounded and can be uniformly quantized over their finite domains

Algorithm Outline

- Initialize accumulator H to all zeros
- For each edge point (x,y) in the image

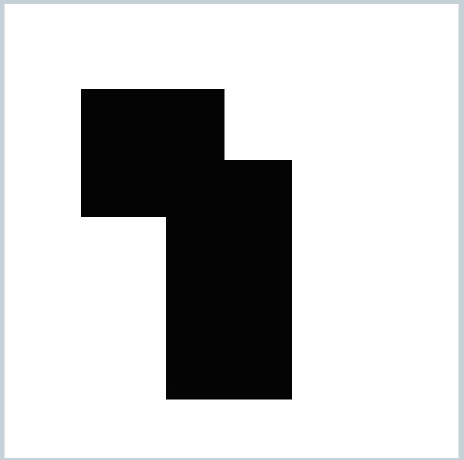
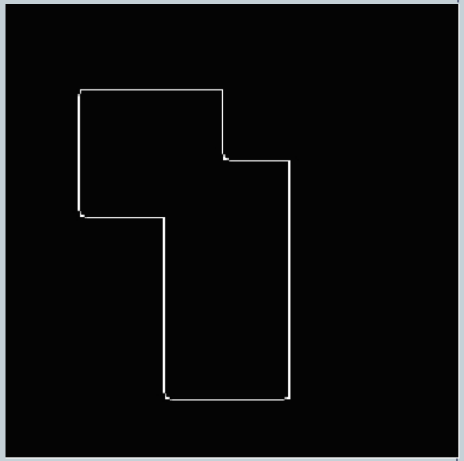
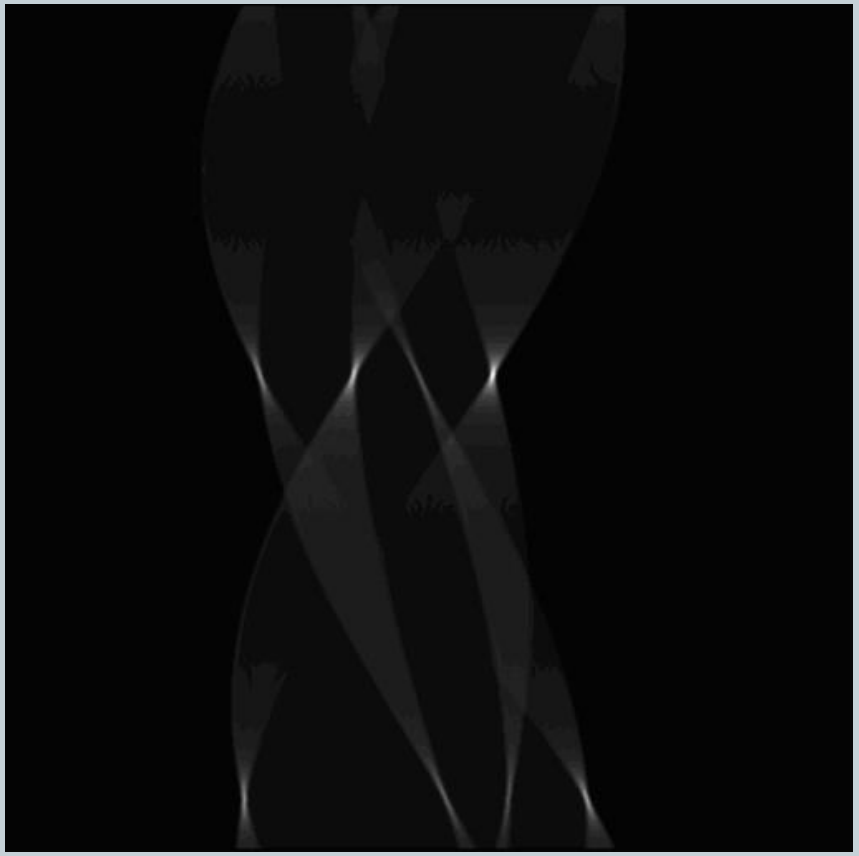
```
For θ = 0 to 180
    $\rho = x \cos\theta + y \sin\theta$
    $H(\theta, \rho) = H(\theta, \rho) + 1$
end
end
```

- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
  - The detected line in the image is given by $\rho = x \cos\theta + y \sin\theta$

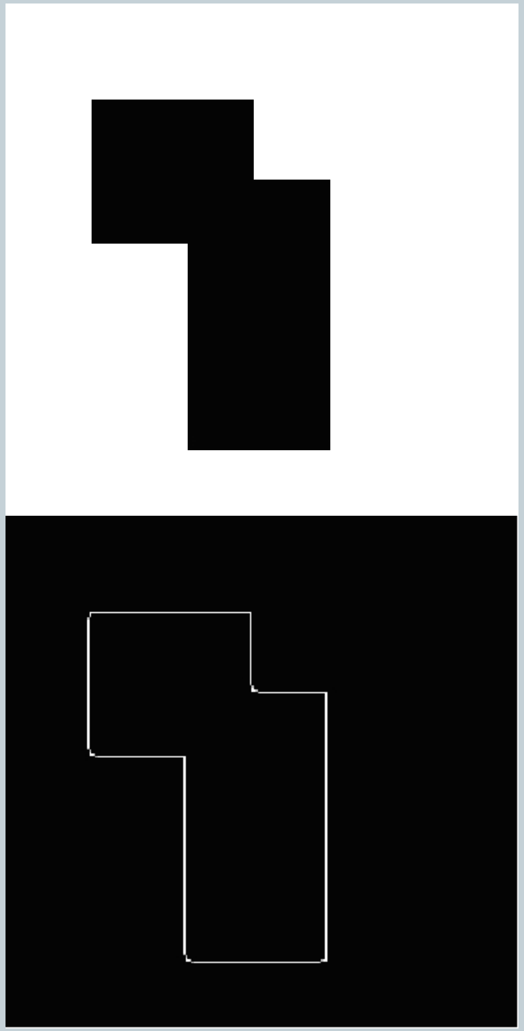


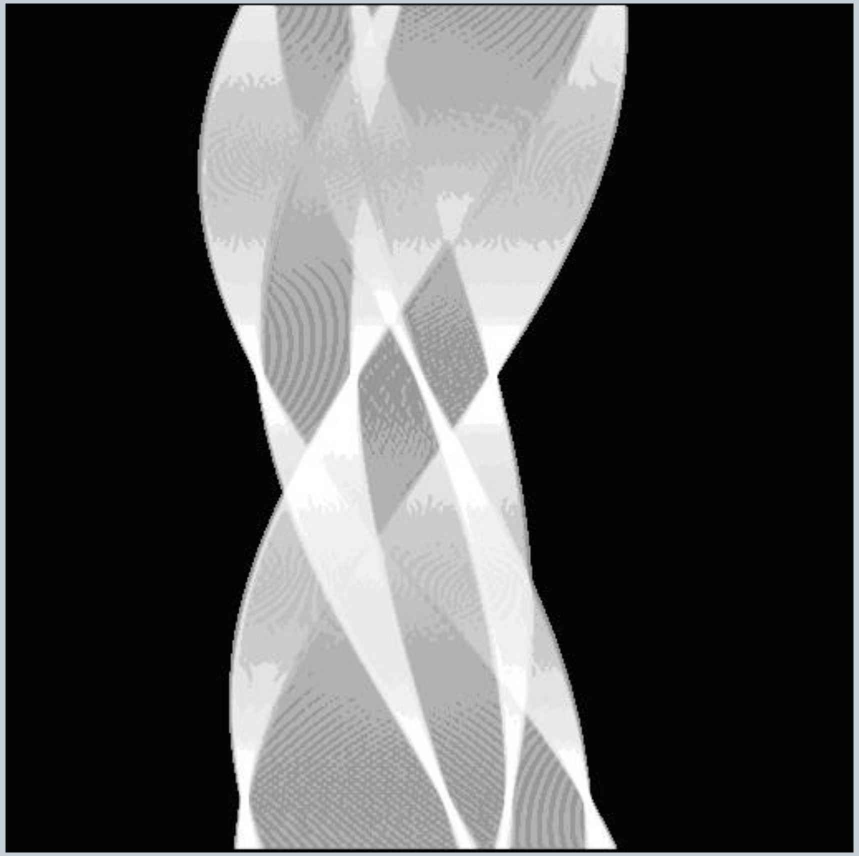
Hough Transform – Lines

- Synthetic Image
  - Overlapping Rectangles
  - Extracted Edges (Canny)
  - Polar Parameter Space Mapping

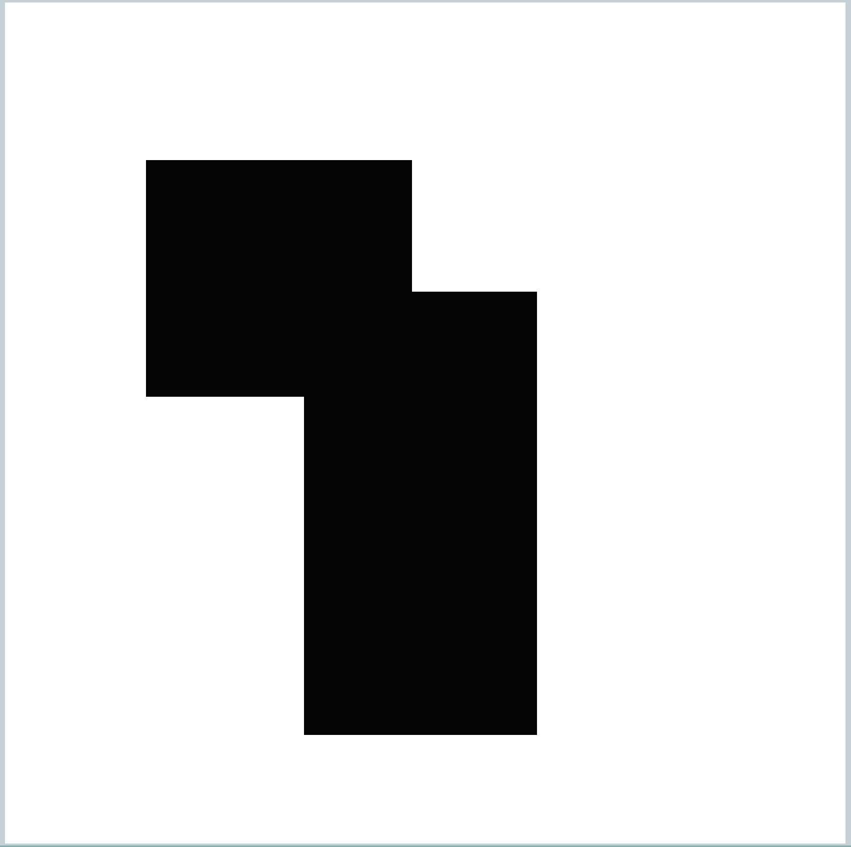
Hough Transform – Lines



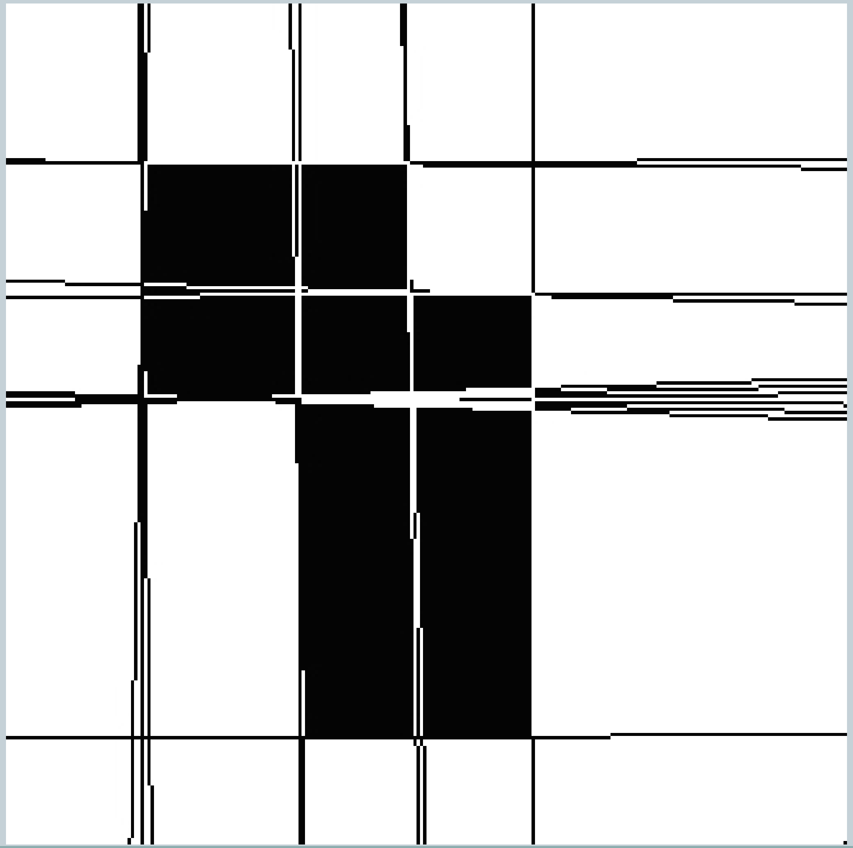


Hough Transform – Lines

Original Image

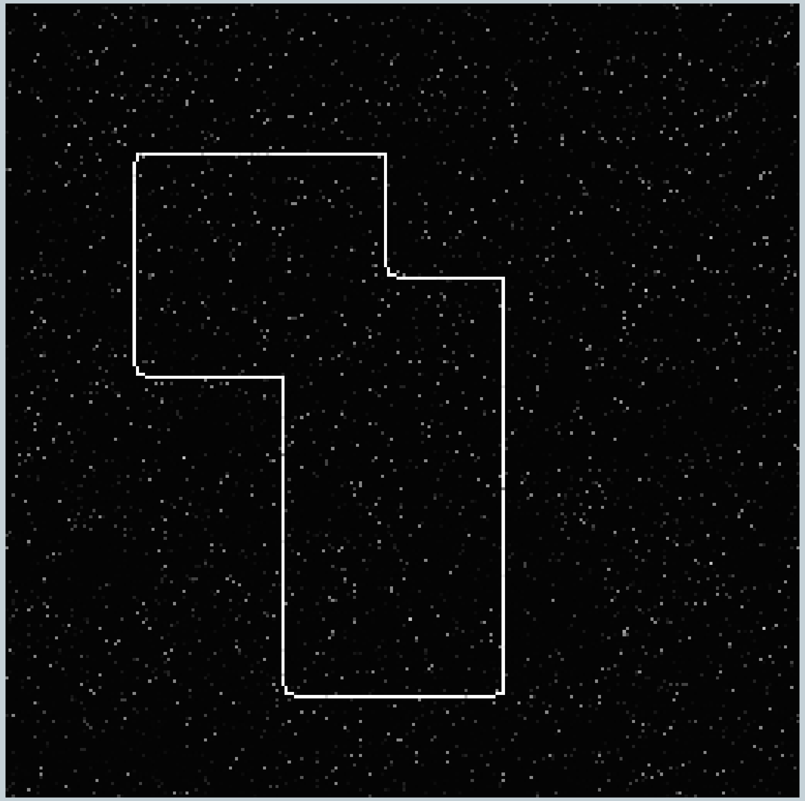


Output with lines

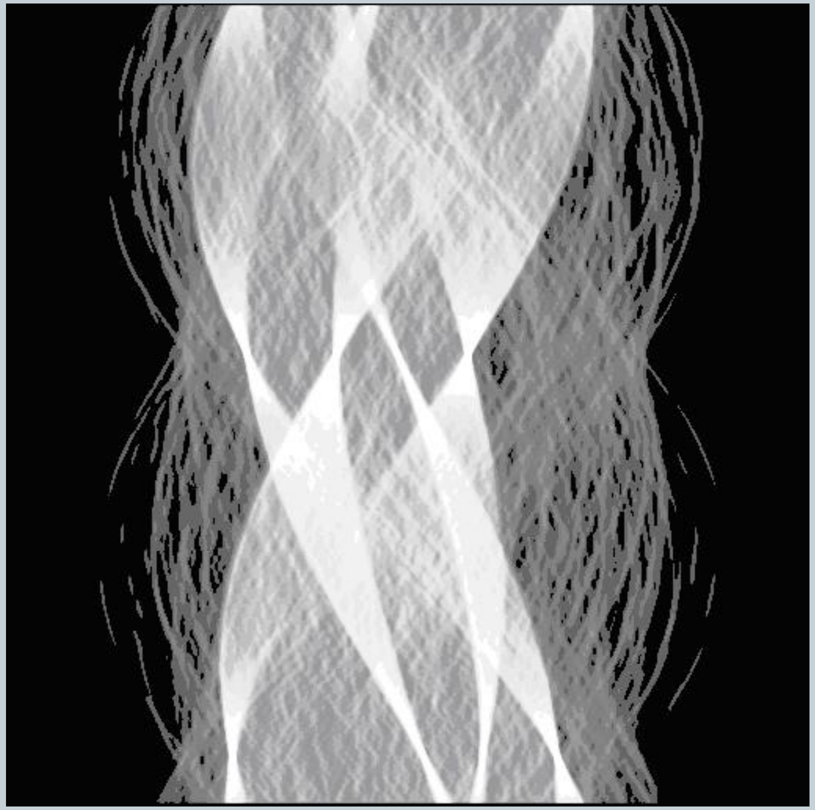


Hough Transform – Lines

Added Salt & Pepper Noise to edge image

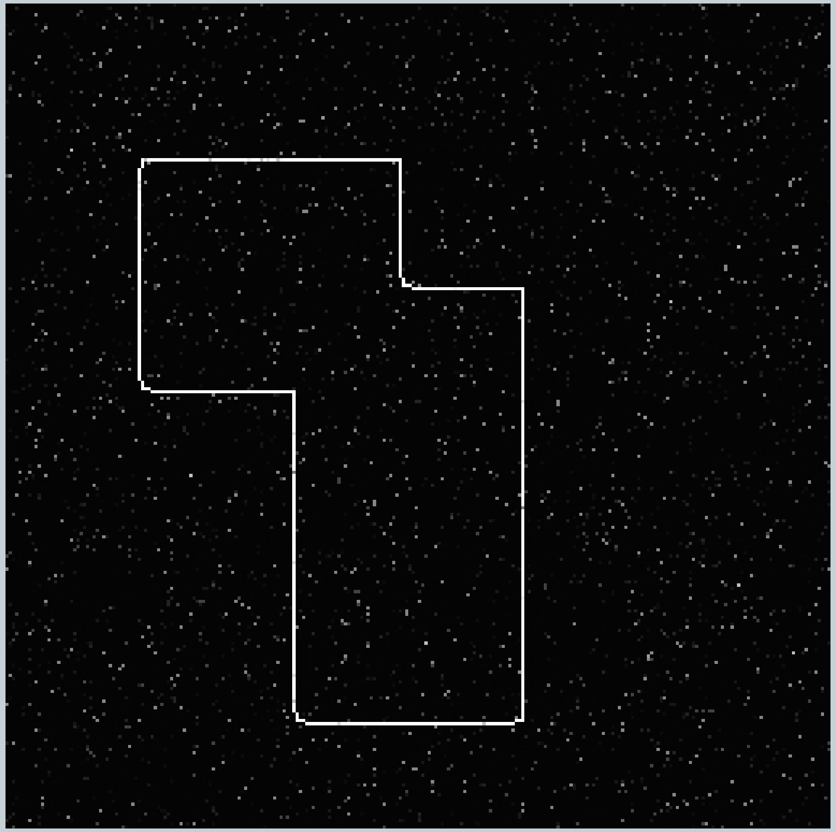


Polar Parameters

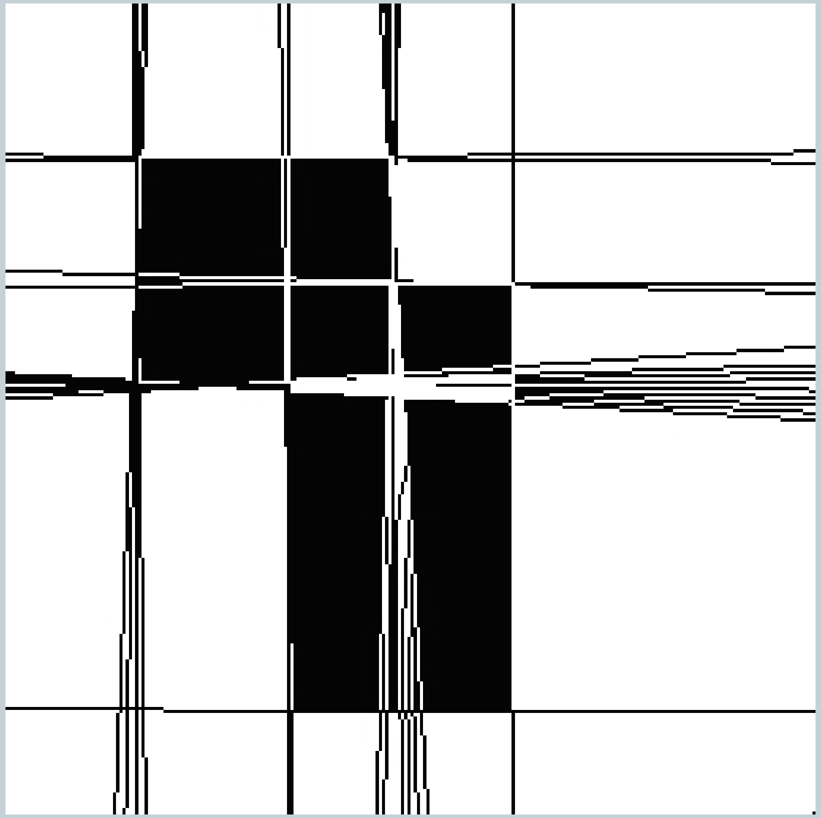


Hough Transform – Lines

Noisy edges

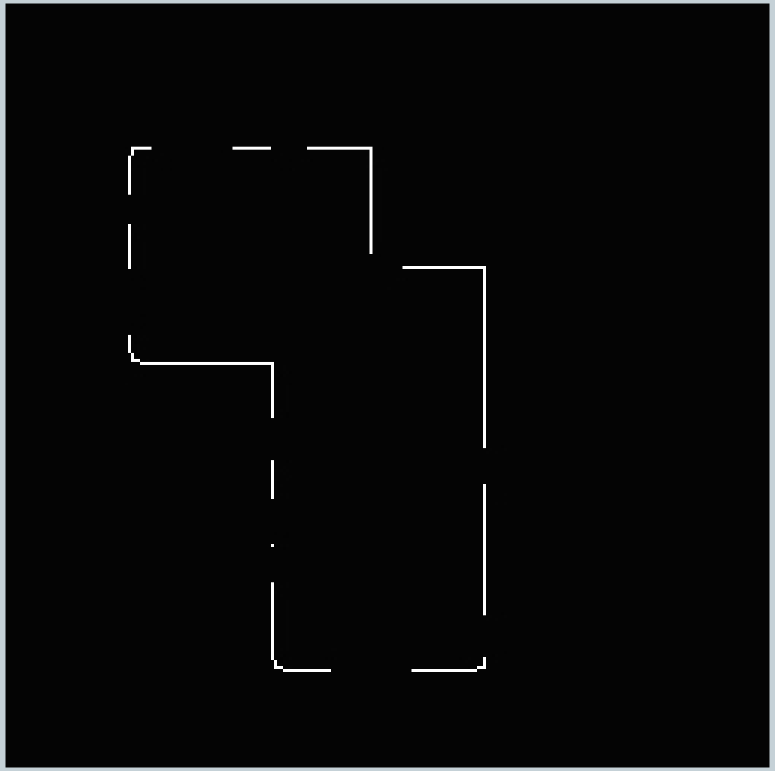


Output

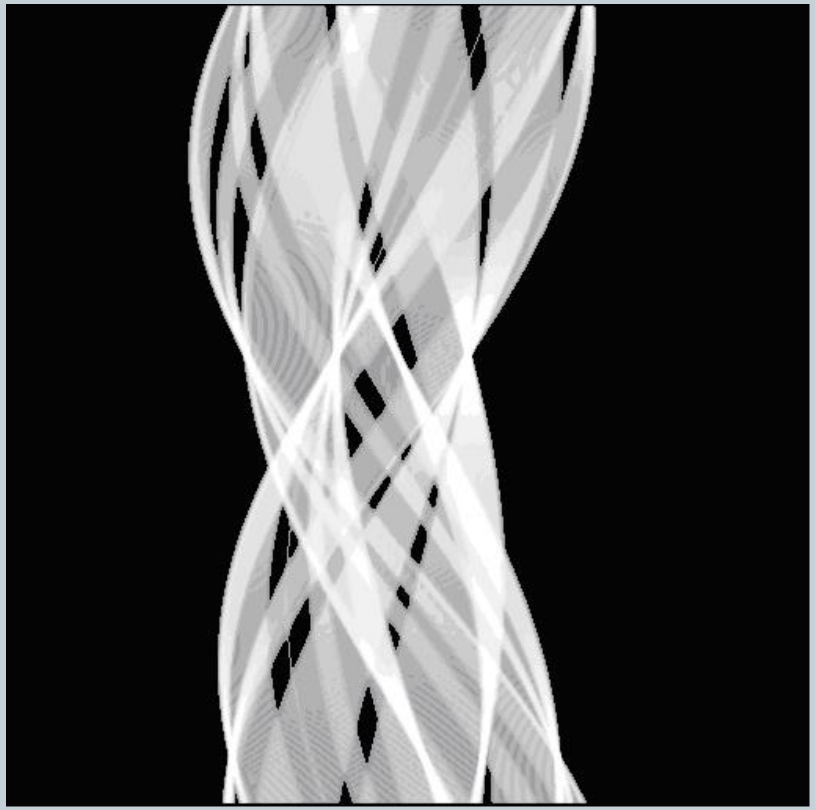


Hough Transform – Lines

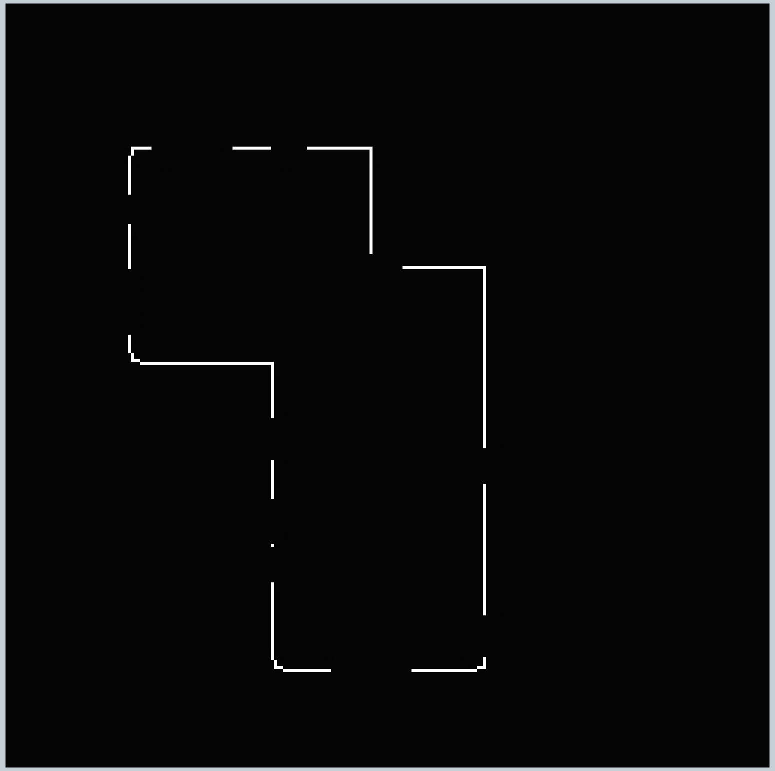
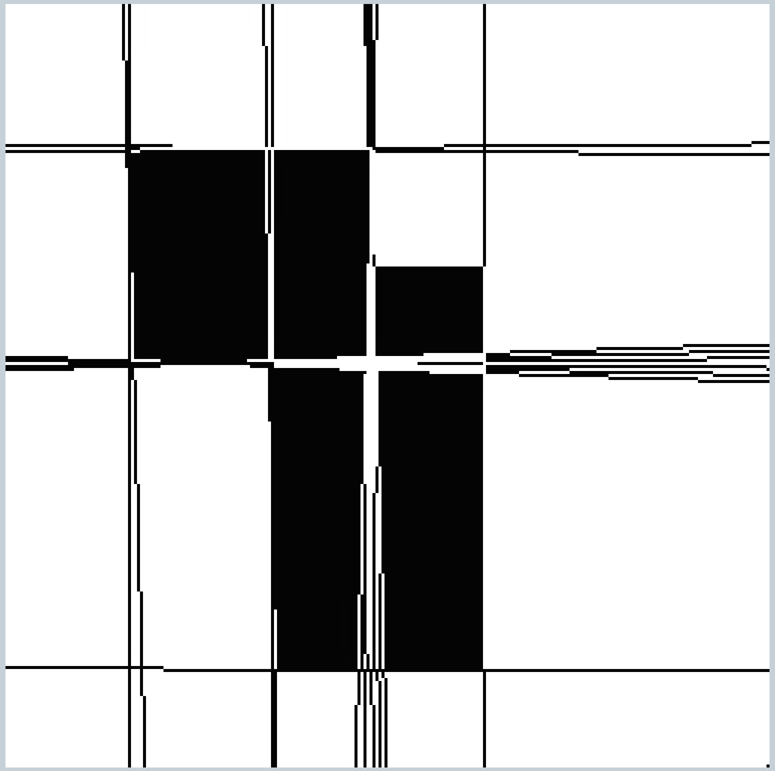
Extracted Edges (Canny) with Dropouts



Polar Coordinate

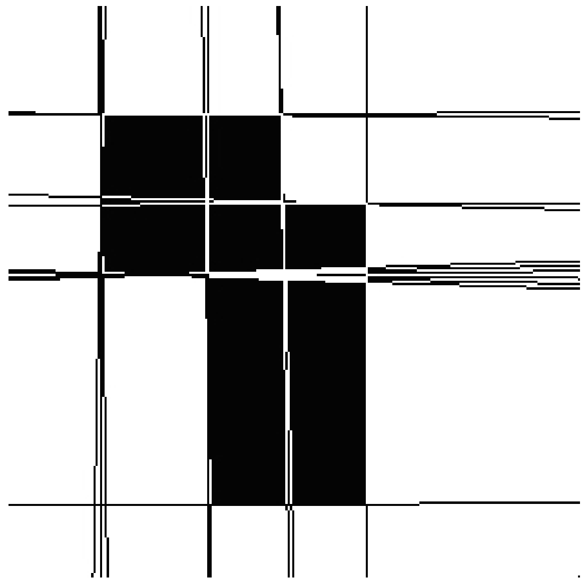


Hough Transform – Lines

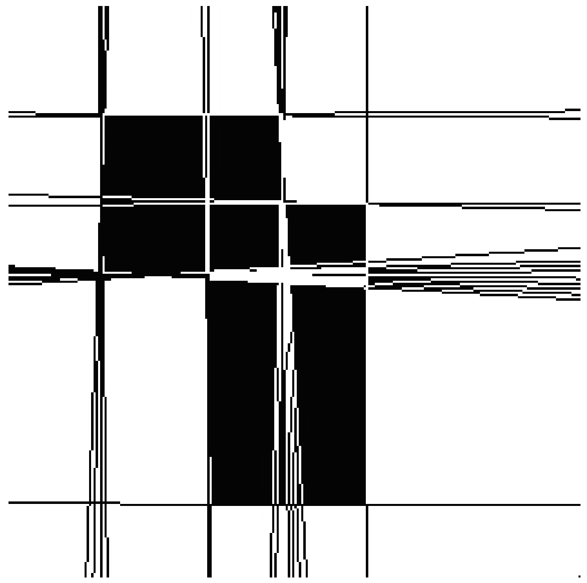
Extracted Edges (Canny) with Dropouts	**Output**
	

Hough Transform – Lines

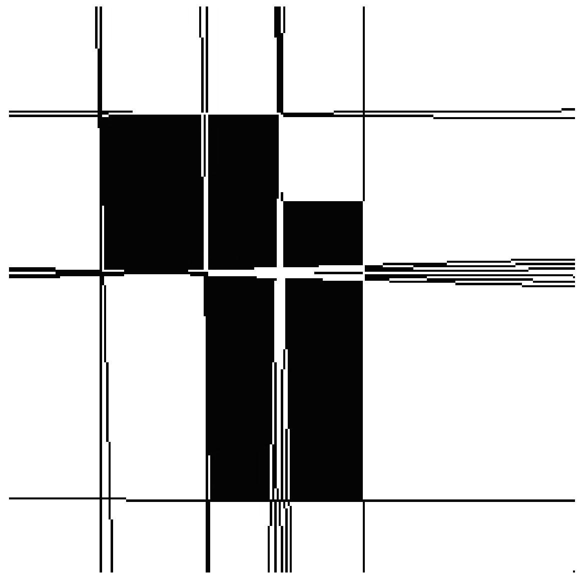
Output without Noise



Output with Noise



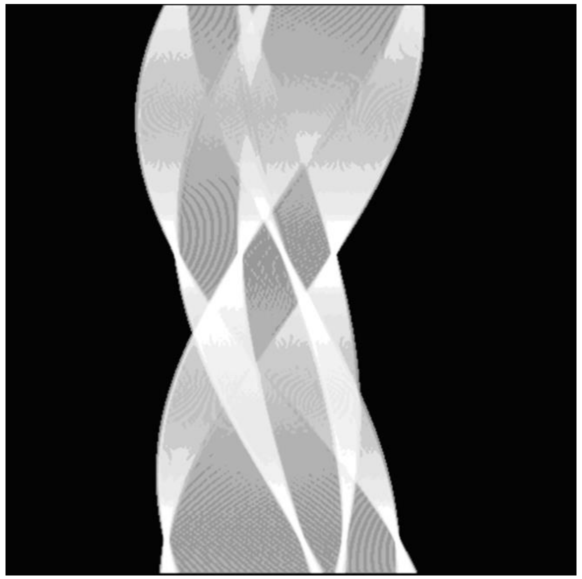
Output with Dropouts



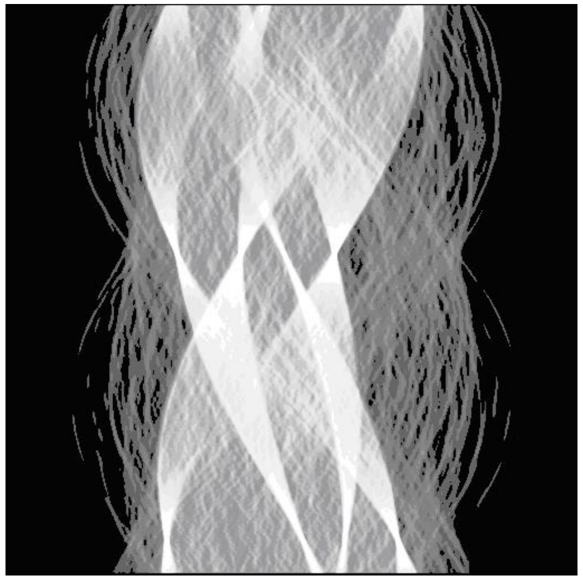
Hough Transform – Lines

Polar Coordinates

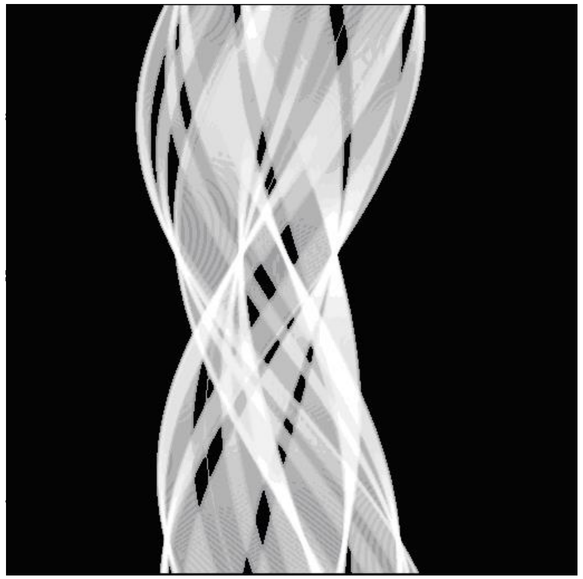
Without Noise



With Noise

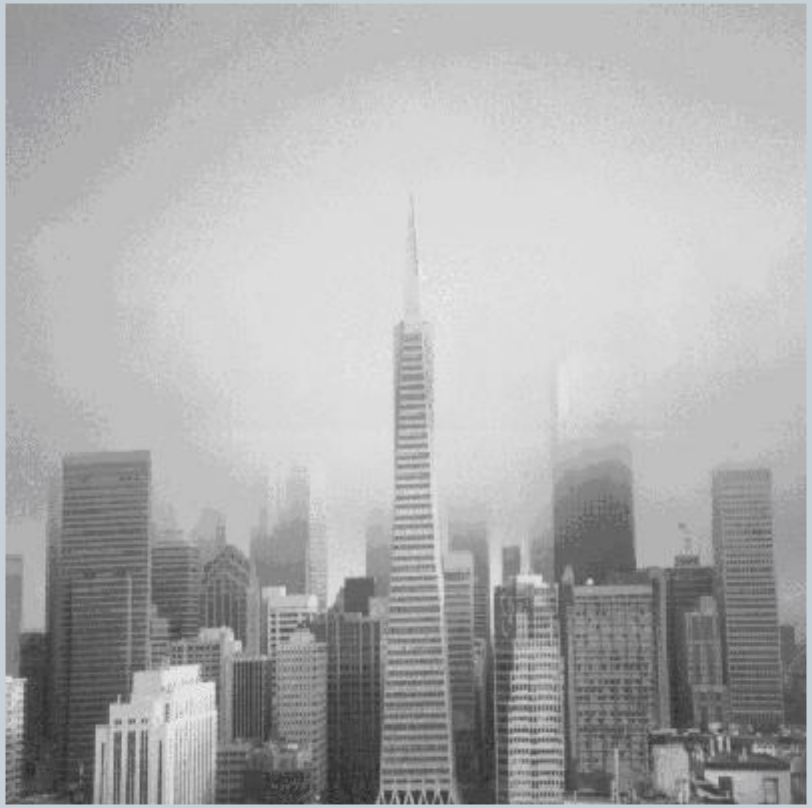


With Dropouts

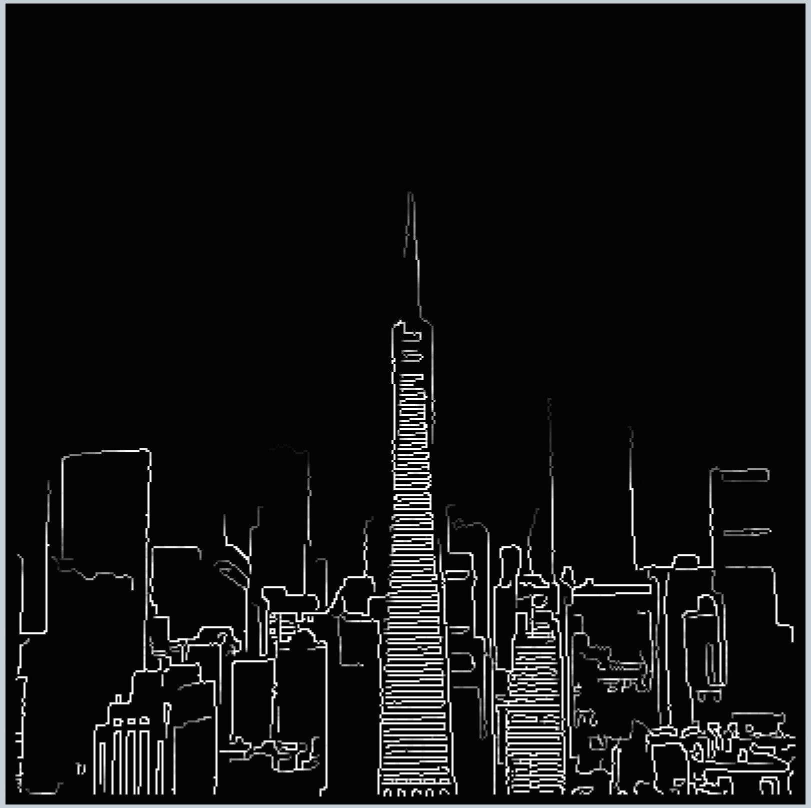


Hough Transform – Lines

Real Image

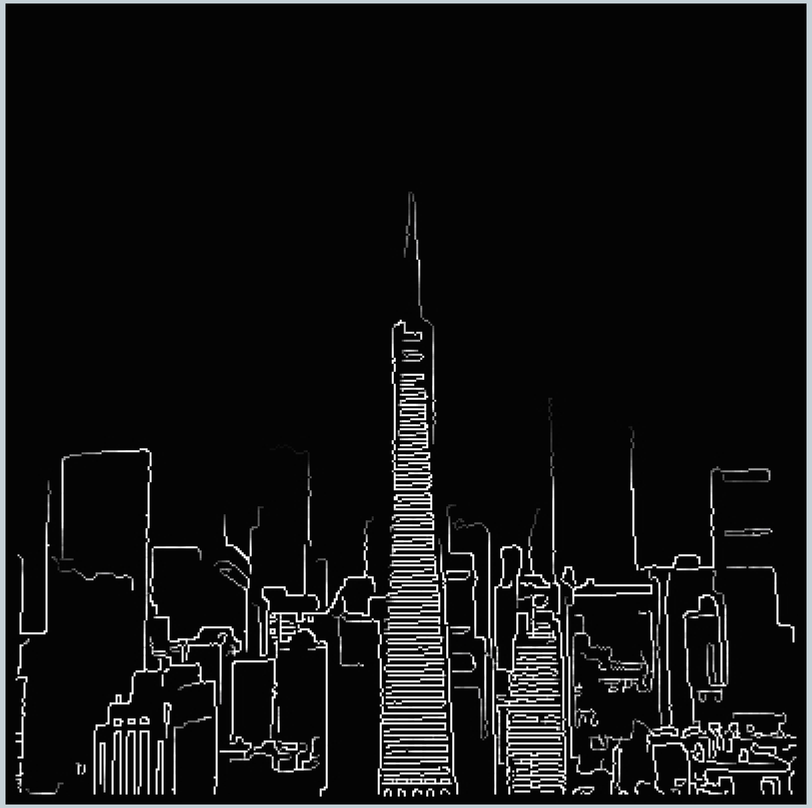


Extracted Edges (Canny)

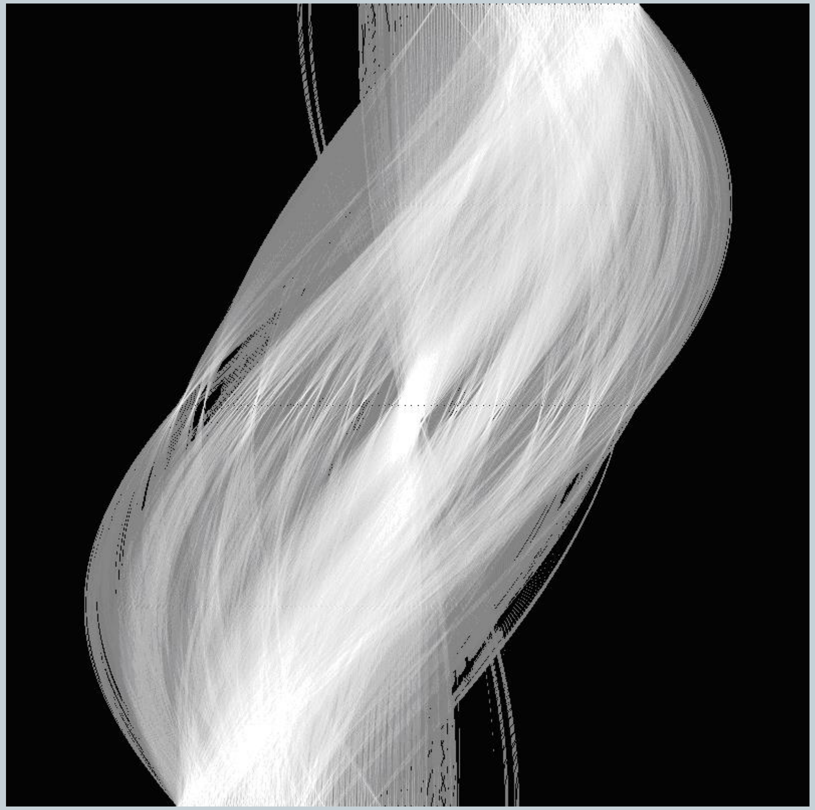


Hough Transform – Lines

Extracted Edges (Canny)

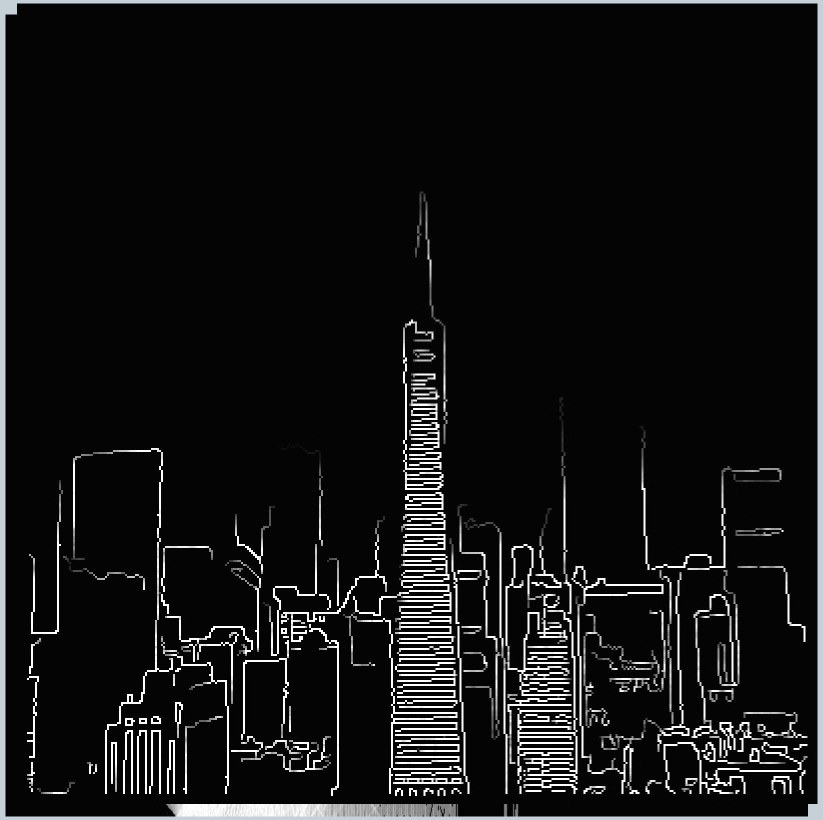


Polar Parameter Space Mapping

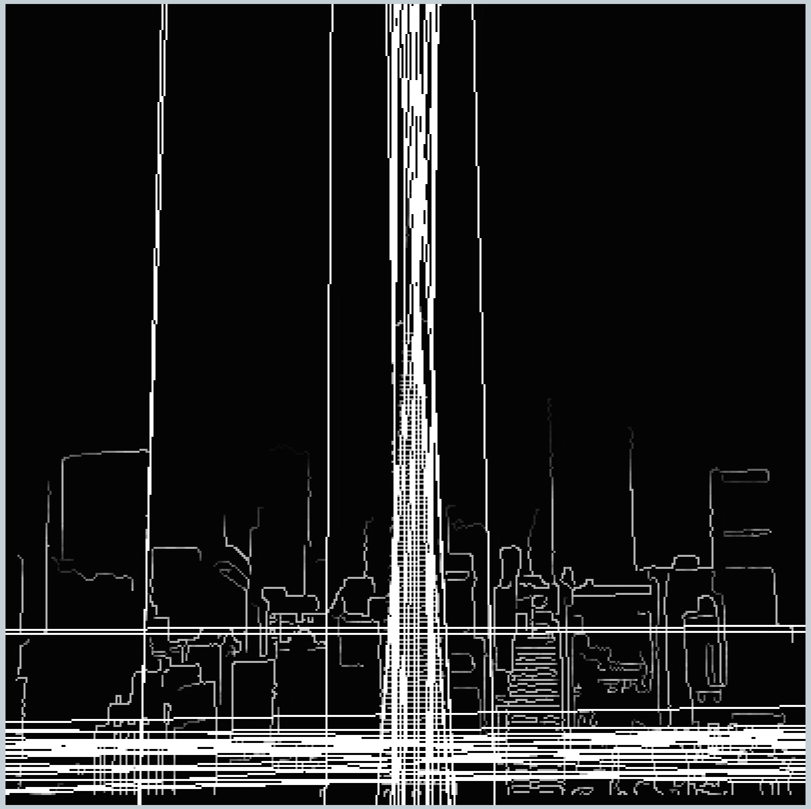


Hough Transform – Lines

Polar Parameter Space Mapping

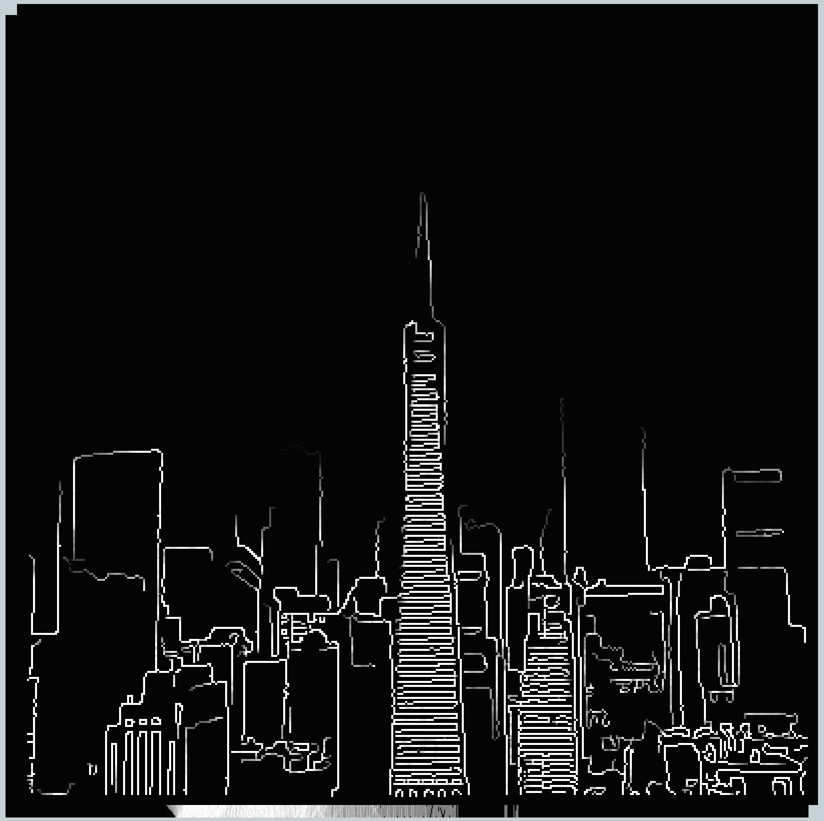


Extracted Lines (Threshold = 70%)

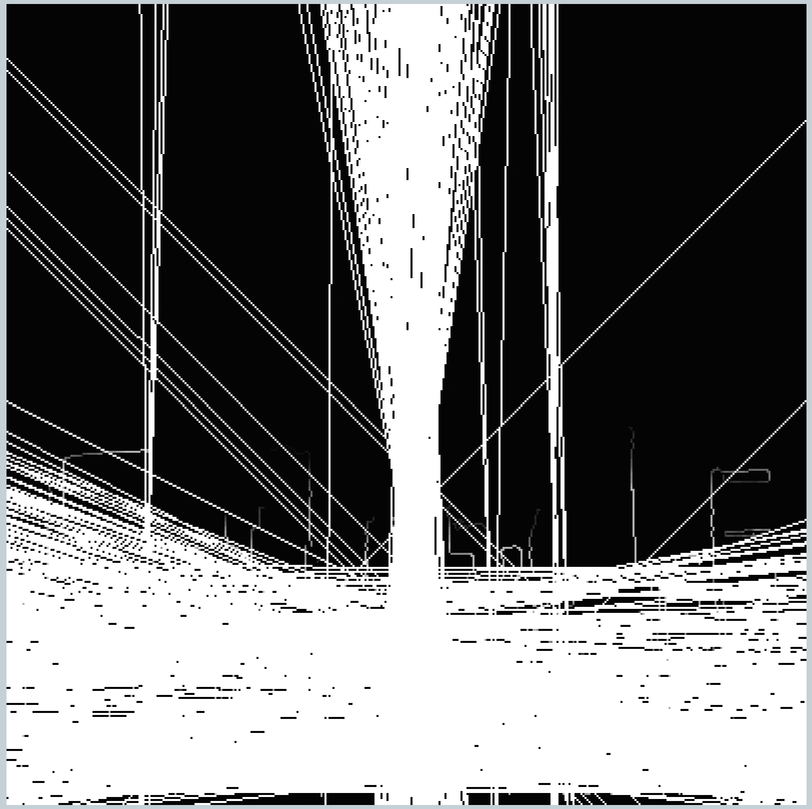


Hough Transform – Lines

Polar Parameter Space Mapping



Extracted Lines (Threshold = 50%)



Hough Transform – Circle

- The Hough transform can be extended to handle other types of geometric primitives.

- After lines, the next most popular primitives to extract are circles.

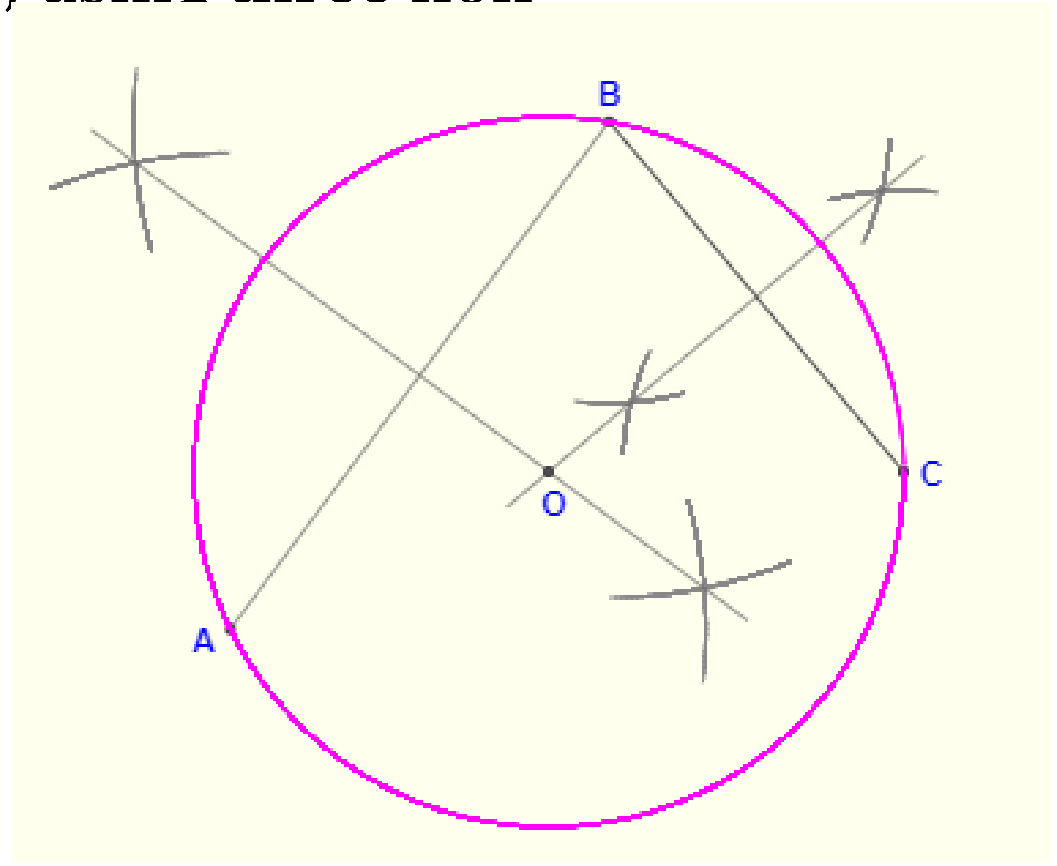
- There are two main changes to apply Hough transform to circles.
  - The first change is that the parameter space is 3-D rather than 2-D. The 3 axes of parameter space represent the x and y coordinates of the center of a circle, and the radius r.
  - The second required modification is that, whereas in the case of lines a vote was cast into parameter space for each image point, for circles, three points are required to generate each vote.

Hough Transform – Circle

- The method proceeds as follows:
  1. Randomly choose 3 non-collinear points $\{p1, p2, p3\}$ from I.
  2. Generate an expression for the unique circle that intersects with $\{p1, p2, p3\}$, and project its center and radius values $\{(x_c, y_c), r\}$ into P, incrementing the bin that contains the parameter set.
  3. Repeat Steps 1 and 2 until all possible sets of 3 non-collinear points have been selected.
  4. Threshold P to determine the peaks.

Hough Transform – Circle

- How to compute center and radius, using three non-collinear points?
  - Method 1:
    - Use perpendicular bisectors



Hough Transform – Circle

- How to compute center and radius, using three non-collinear points?
  - Method 2:
    - Use matrices and circle equation in general form

$$a(x^2 + y^2) + bx + cy + d = 0$$

$$\begin{aligned}
x &= \frac{(x_1^2+y_1^2)(y_2-y_3)+(x_2^2+y_2^2)(y_3-y_1)+(x_3^2+y_3^2)(y_1-y_2)}{2(x_1(y_2-y_3)-y_1(x_2-x_3)+x_2y_3-x_3y_2)} &&= -\frac{b}{2a} \\
y &= \frac{(x_1^2+y_1^2)(x_3-x_2)+(x_2^2+y_2^2)(x_1-x_3)+(x_3^2+y_3^2)(x_2-x_1)}{2(x_1(y_2-y_3)-y_1(x_2-x_3)+x_2y_3-x_3y_2)} &&= -\frac{c}{2a} \\
r &= \sqrt{(x-x_1)^2+(y-y_1)^2} &&= \sqrt{\frac{b^2+c^2-4ad}{4a^2}}
\end{aligned}$$

Hough Transform – Circle

$$a(x^2+y^2)+bx+cy+d=0$$

$$\begin{vmatrix} x^2+y^2 & x & y & 1 \\ x_1^2+y_1^2 & x_1 & y_1 & 1 \\ x_2^2+y_2^2 & x_2 & y_2 & 1 \\ x_3^2+y_3^2 & x_3 & y_3 & 1 \end{vmatrix} = 0$$

$$a = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, b = -\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix}, c = \begin{vmatrix} x_1^2+y_1^2 & x_1 & 1 \\ x_2^2+y_2^2 & x_2 & 1 \\ x_3^2+y_3^2 & x_3 & 1 \end{vmatrix}, d = -\begin{vmatrix} x_1^2+y_1^2 & x_1 & y_1 \\ x_2^2+y_2^2 & x_2 & y_2 \\ x_3^2+y_3^2 & x_3 & y_3 \end{vmatrix}$$

$$\begin{aligned} a =& \quad x_1(y_2-y_3)-y_1(x_2-x_3)+x_2y_3-x_3y_2 \\ b =& \quad (x_1^2+y_1^2)(y_3-y_2)+(x_2^2+y_2^2)(y_1-y_3)+(x_3^2+y_3^2)(y_2-y_1) \\ c =& \quad (x_1^2+y_1^2)(x_2-x_3)+(x_2^2+y_2^2)(x_3-x_1)+(x_3^2+y_3^2)(x_1-x_2) \\ d =& \quad (x_1^2+y_1^2)(x_3y_2-x_2y_3)+(x_2^2+y_2^2)(x_1y_3-x_3y_1)+(x_3^2+y_3^2)(x_2y_1-x_1y_2) \end{aligned}$$

Hough Transform – Circle

- There are two problems with the given formation
  1. Two or more of the three selected points are too close, the estimate of the circle parameters can be numerically inaccurate.
     - To avoid this, votes are generated only if the randomly selected three points are sufficiently separated.

  2. There will in general be far too many combinations of three points in an edge map to enumerate them all exhaustively. For n points, there are

$$\frac{n(n-1)(n-2)}{6} = O(n^3) \text{ triplets.}$$

     - For example, for $n = 10^4$ (a reasonably small edge image), then there will be $> 10^{12}$ triples, which is a huge and unmanageable number.

What to do to resolve this?

Hough Transform – Circle

- Randomized Hough Transform
  - Rather than exhaustively generating all possible sets of triplets, the RHT generates only a relatively small number.

  - If a circle comprises $p \times 100\%$ of the points in an edge map of n edge pixels, then the likelihood of selecting a point that falls on the circle is pn. The total number of triplets is then approximately:

$$\frac{[pn][p(n-1)][p(n-2)]}{6}$$

  - The chance of randomly selecting a triplet that falls on the circle is therefore ~ p^3.

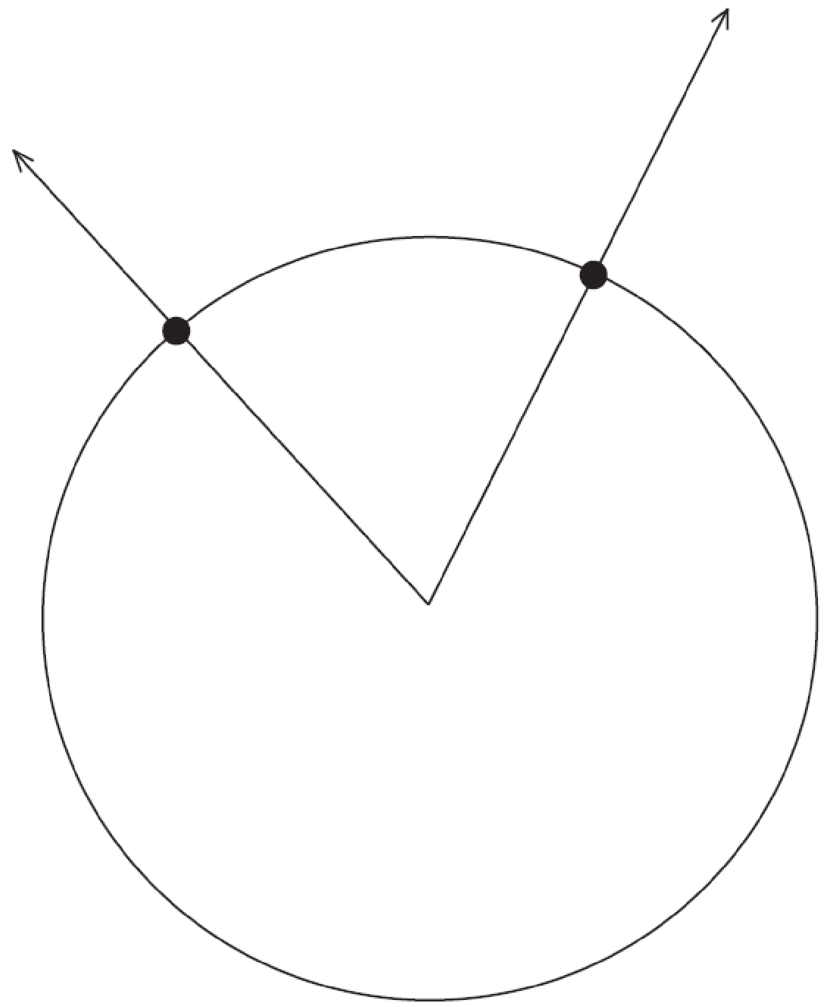
  - If $p = 0.5$, then half of the image edge pixels fall on the circle. The likelihood is that $p^3 = 12.5\%$ of a sufficiently large sample of triples will fall on the circle.

Hough Transform – Circle

- Probabilistic Hough Transform
  - The progression of the peaks in accumulator (parameter) space is tracked.
  - A true peak will have certain statistical properties with respected to its neighbouring peaks.
  - By monitoring the progression of the peaks, in this way it is possible to distinguish between true and false peaks sooner.
  - When a true peak is determined, its associated circle is identified, and the corresponding points are removed from the image.
  - This further improves the efficiency of the remaining processing, which is functioning on a reduced edge image size.
  - Processing will typically stop when the desired number of circles have been extracted.

Hough Transform – Circle

- Another Variation
  - The edge orientation information (e.g. as may be derived from the image Gradient) can be used to advantage.
  - If the tangent of each edge point is known, then it is possible to hypothesize a candidate circle using only 2, rather than 3 points.
  - This is achieved by extending rays from each of the two points in the opposite direction of their respective tangents.
  - These two rays will intersect at the circle center, which will provide both the center coordinate of the circle.
  - The distance between the center and either of the original two points will then provide the circle radius.



Generalized Hough Transform

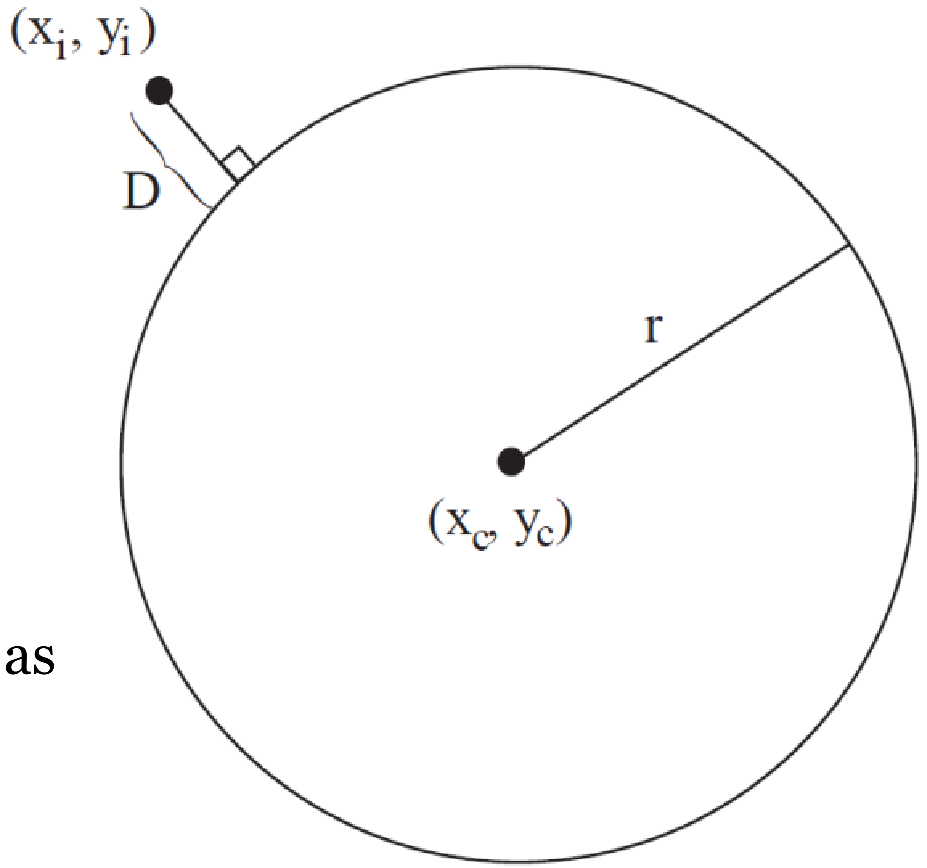
- Hough transform can be used for other parametric shapes
  - For example, an ellipse is defined by 5 points so 5 D parameter space.
  - Templates can also be searched in the images.
    - A template is a set of edge points of the shape.
    - The accumulator space is incremented by aligning the template with each possible image point for all possible transformations (translations and rotations) of the template.

RANSAC – Circle

- The algorithm steps are the same as those for line RANSAC
- For a circle, center (x_c, y_c) and radius r
- The distance D of a point $p_i = (x_i, y_i)$ from the circle center is calculated as:

$$D = \left|\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - r\right|$$

- Points that are within the distance are accumulated as score for the quality of particular model
- Define termination conditions



(x_i, y_i)
D
r
(x_c, y_c)