Chapter 8

Fitting: Voting and Hough Transform

Prof. Fei-Fei Li, Stanford University

Contents

Line fitting

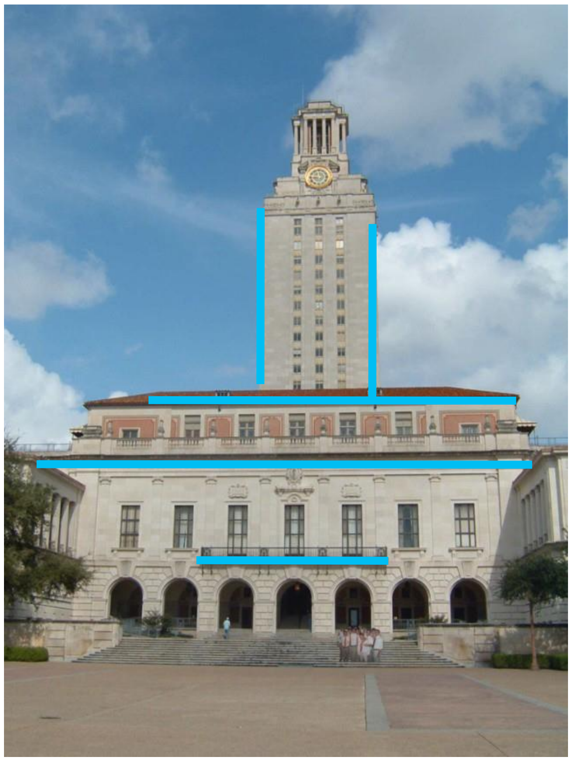
- Hough Transform
- RANSAC (**RANdom SAmple Consensus**)

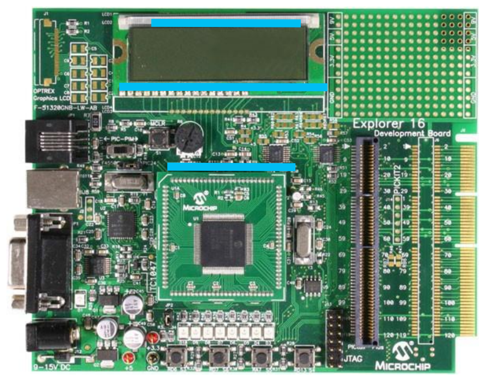
Fitting as Search in Parametric Space

- *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*

Example: Line Fitting

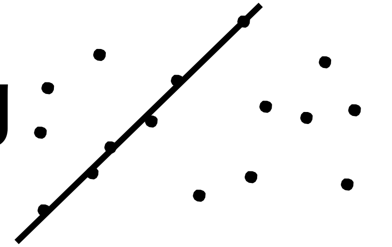
- ***Why fit lines? Many objects characterized by presence of straight lines***

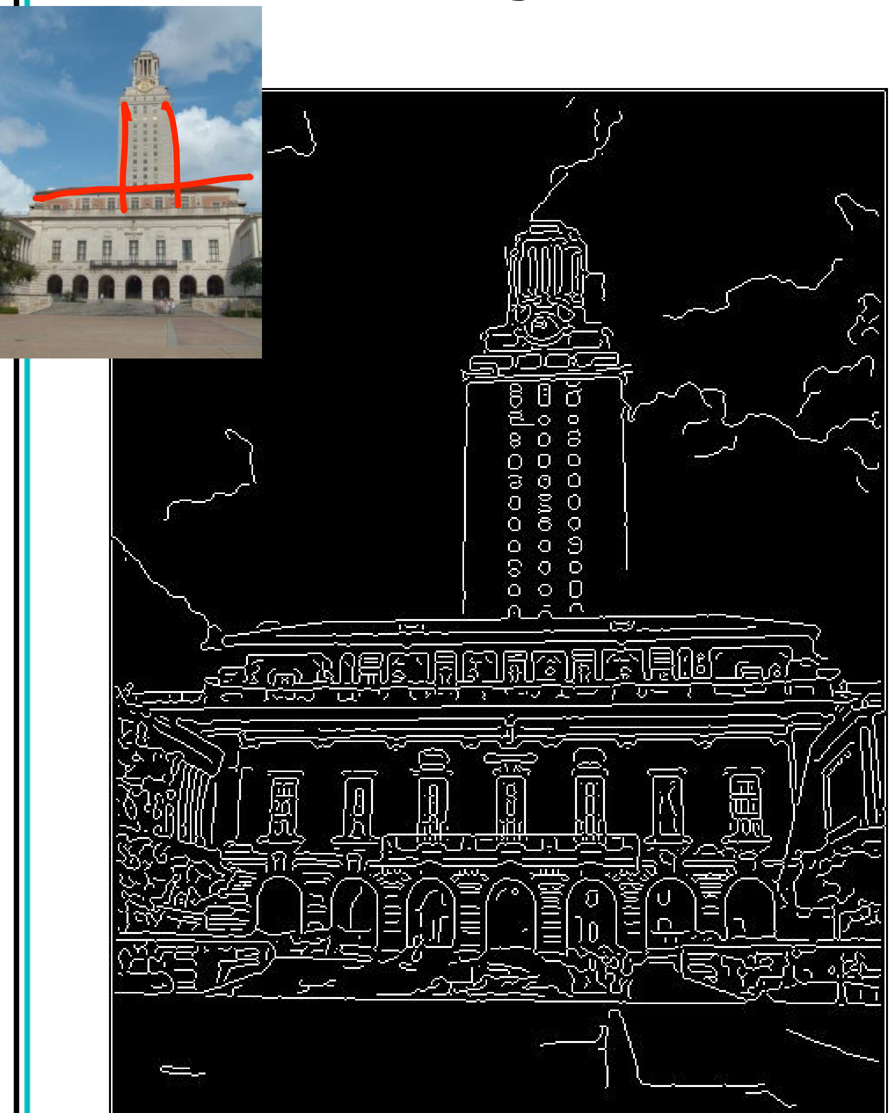




- ***Wait, why aren't we done just by running edge detection?***

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?***

Voting

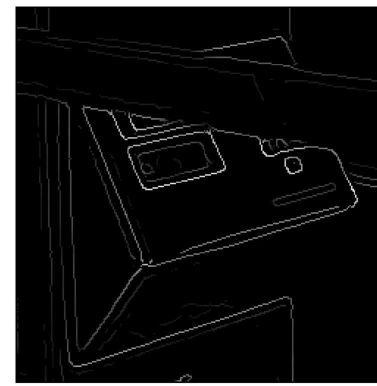
- *It's not feasible to check all combinations of features by fitting a model to each possible subset.*
- *Voting is a general technique where we let the features vote for all models that are compatible with it.*
  - *Cycle through features, cast votes for model parameters.*
  - *Look for model parameters 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.*
- *Ok if some features not observed, as model can span multiple fragments.*

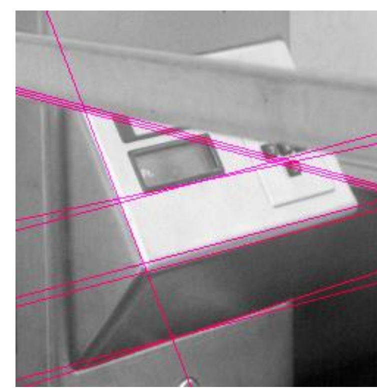
Slide credit: Kristen Grauman

Fitting Lines

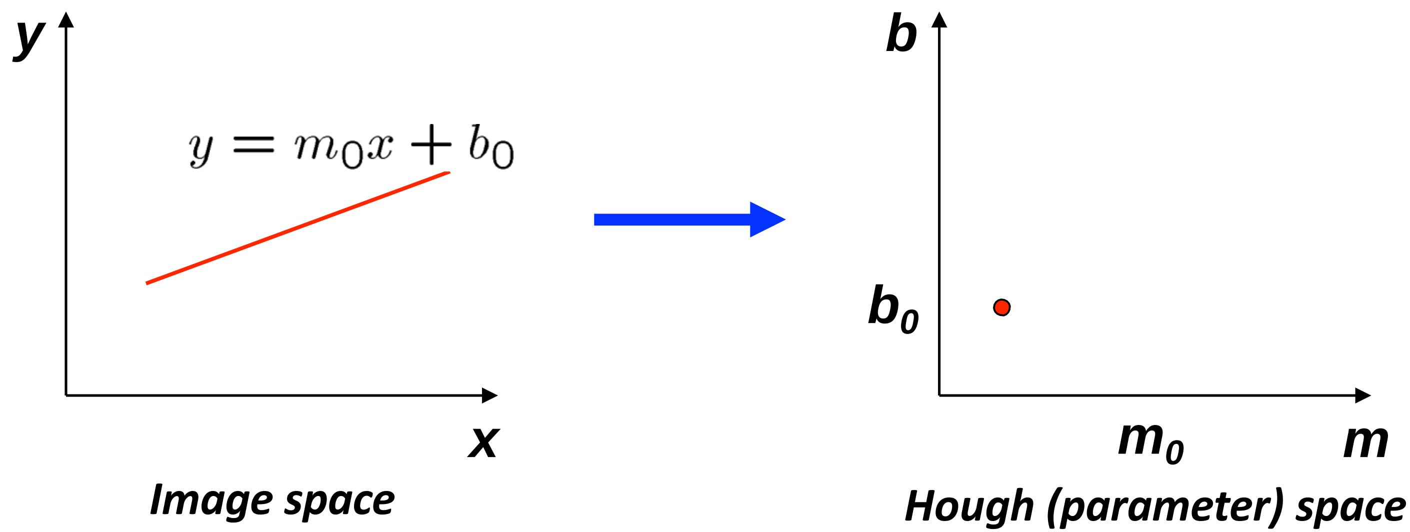
- ***Given points that belong to a line, what is the line?***
- ***How many lines are there?***
- ***Which points belong to which lines?***

- ***Hough Transform is a voting technique that can be used to answer all of these***
- ***Main idea:***
  1. ***Record all possible lines on which each edge point lies.***
  2. ***Look for lines that get many votes.***



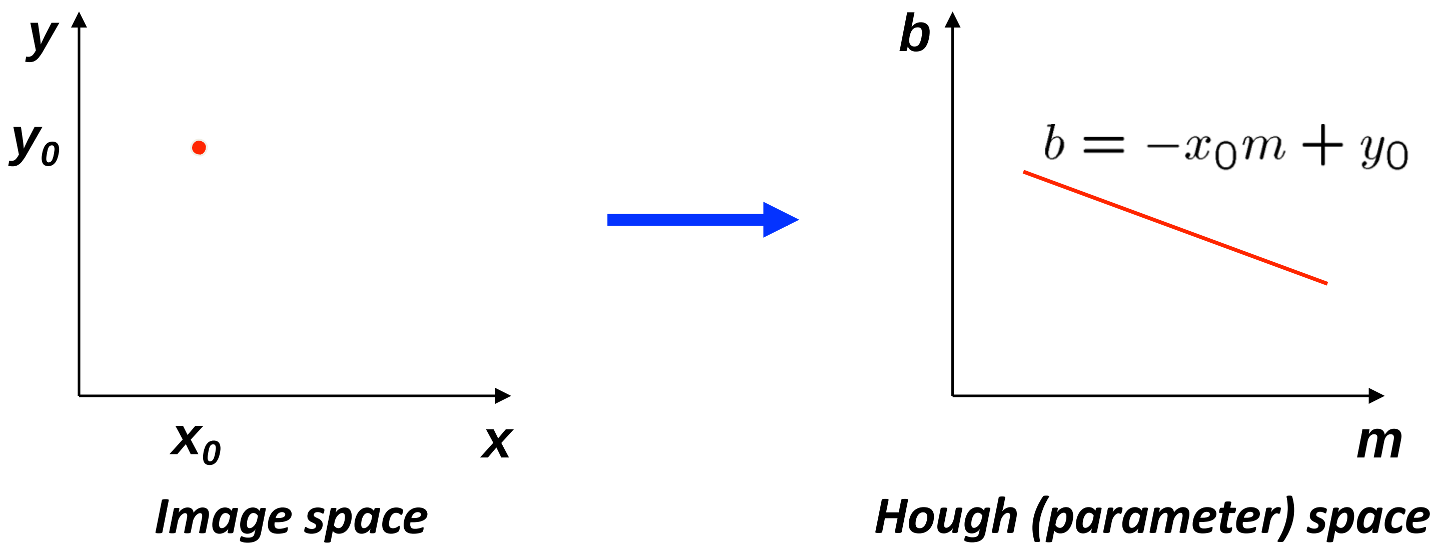


Finding Lines in an Image: Hough Space



- ***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 $y = mx + b$*

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 $b = -x_0 m + y_0$*
    - *This is a line in Hough space*

Finding Lines in an Image: Hough Space

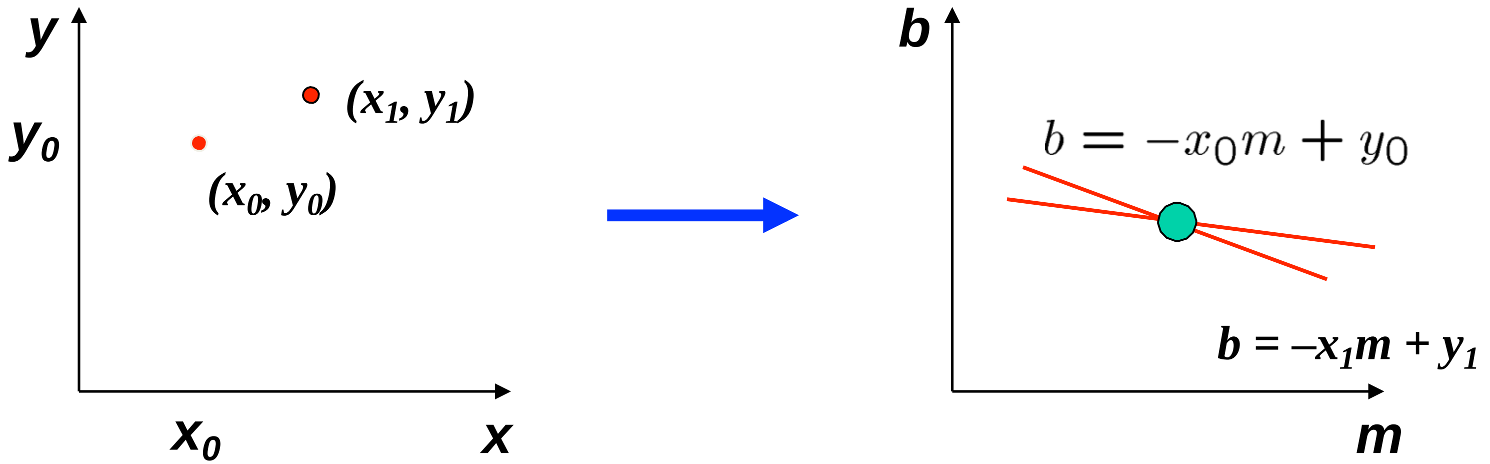
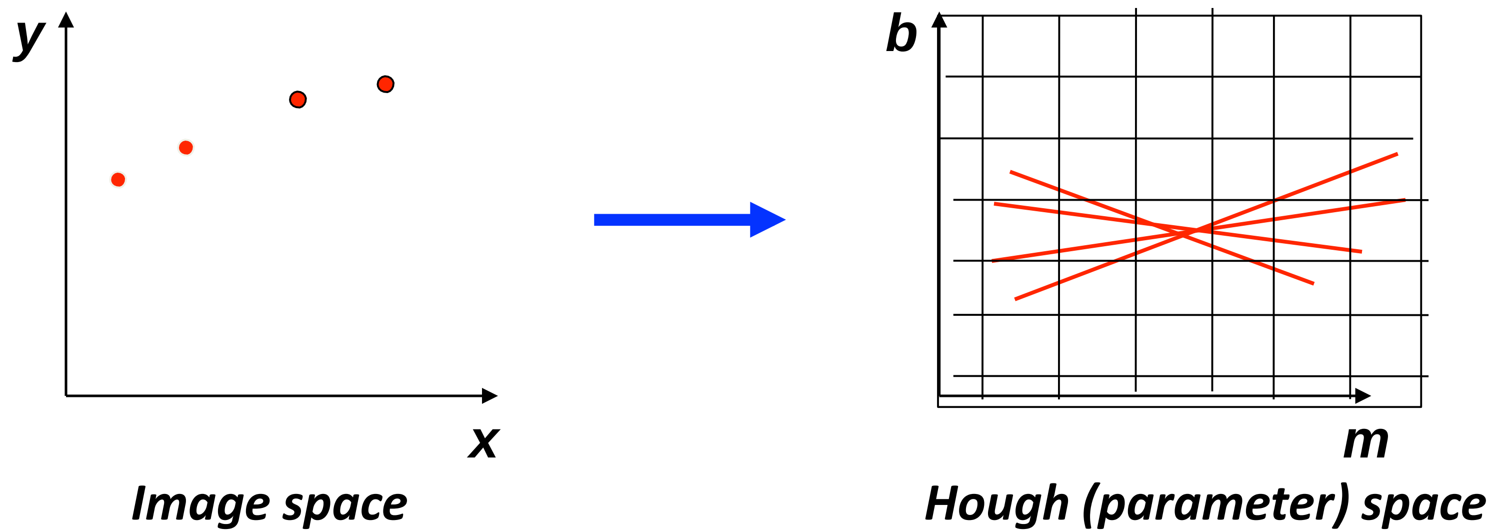


Image space

Hough (parameter) 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** $b = -x_1 m + y_1$*

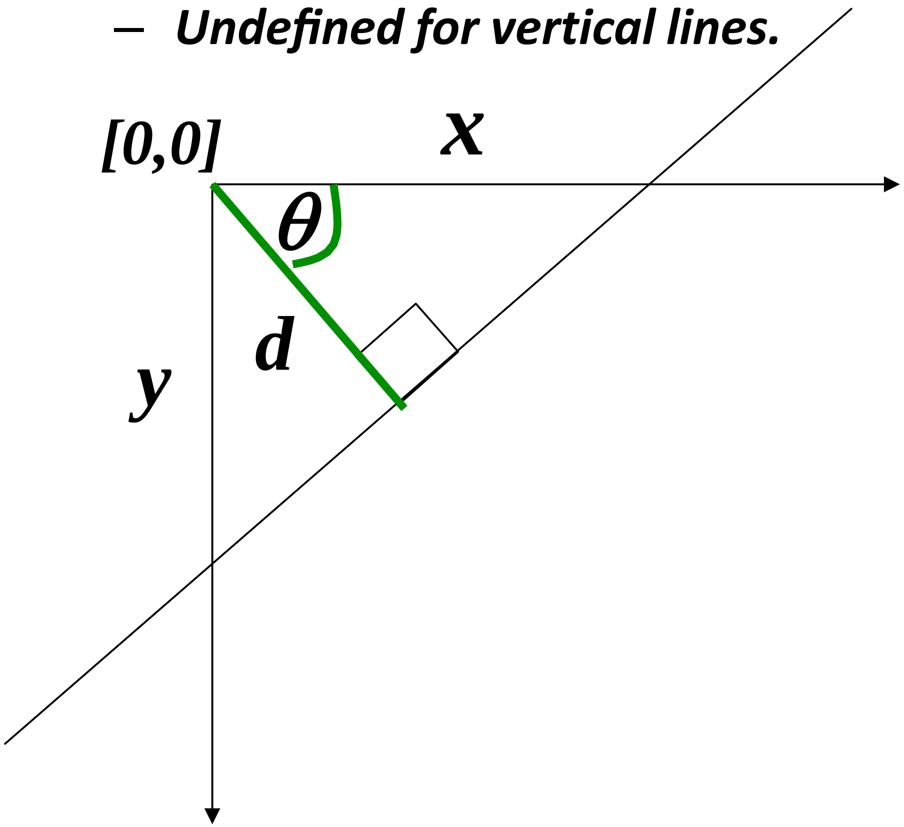
Finding Lines in an Image: Hough Space



- ***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:***
  - *Can take on infinite values;*
  - *Undefined for vertical lines.*



***d** : perpendicular distance from line to origin*

θ : angle the perpendicular line makes with the x-axis

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

where $\theta \in [0, \pi)$ and $d \in R$

- ***Point in image space $\Rightarrow$ sinusoid segment in Hough space***

Hough Transform Algorithm

Using the polar parameterization:

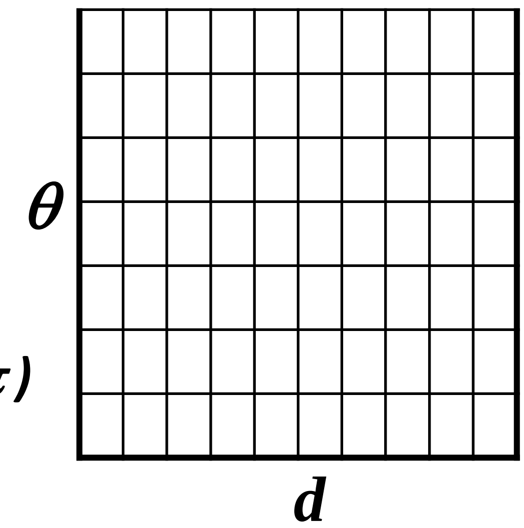
$$x\cos\theta + y\sin\theta = d$$

Basic Hough transform algorithm

1. *Initialize $H[d,\theta] = 0$.*
2. *For each edge point (x,y) in the image for $\theta \in [0,\pi)$*
   $H[d,\theta]$ += 1
3. *Find the value(s) of (d,θ) where $H[d,\theta]$ is maximum*
4. *The detected line in the image is given by*

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

- ***Time complexity (in terms of number of votes)?***



Example: HT for Straight Lines

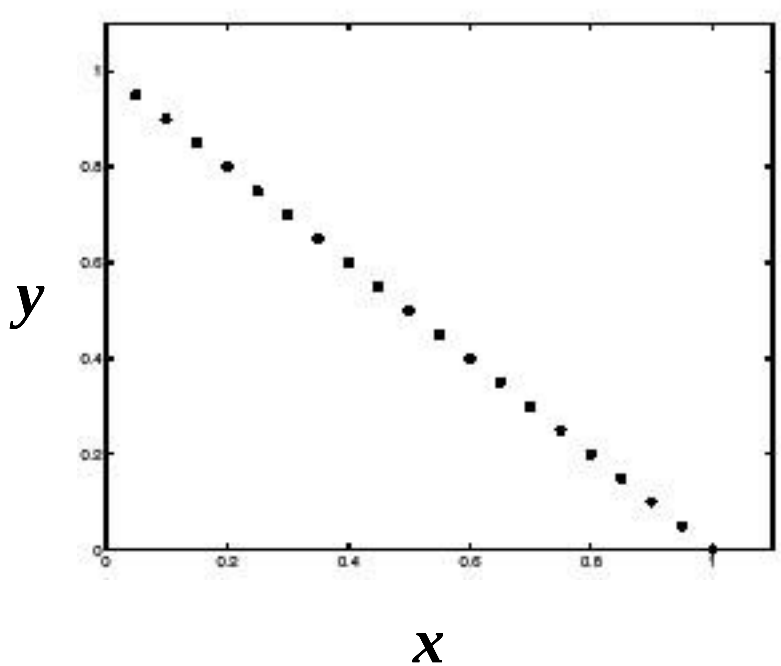
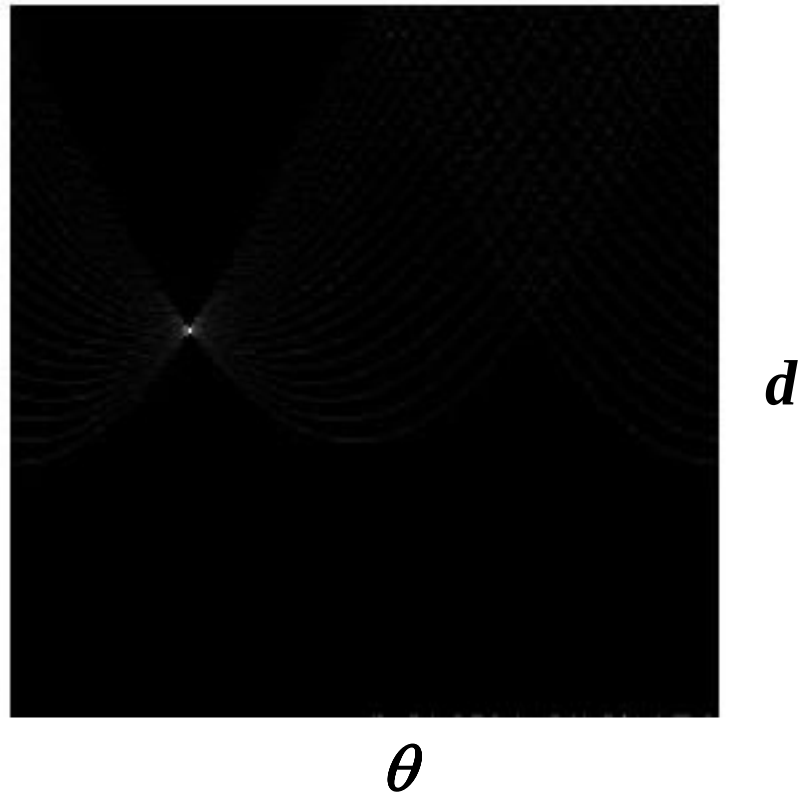


Image space edge coordinates



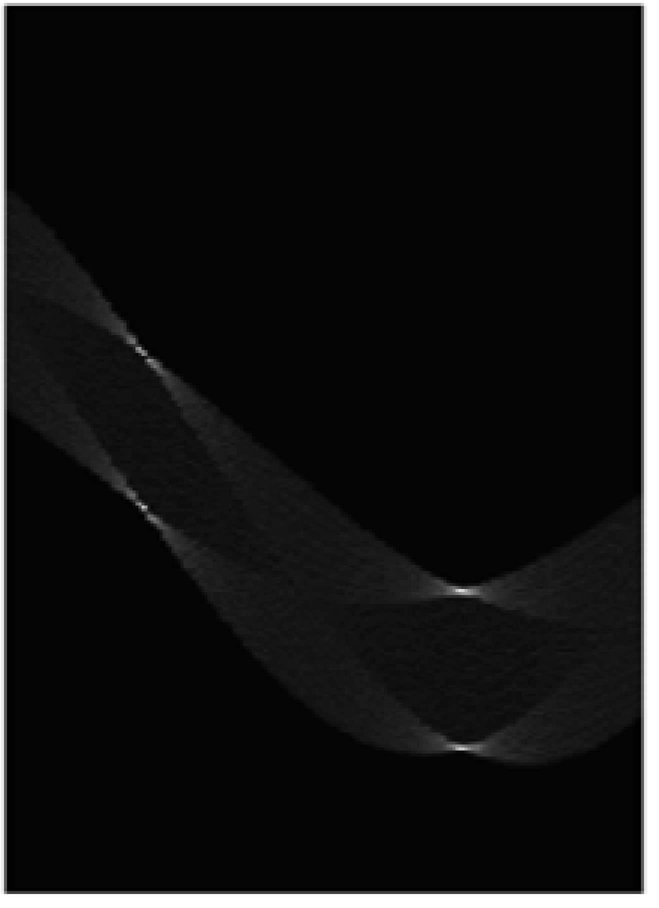
Votes

Bright value = high vote count

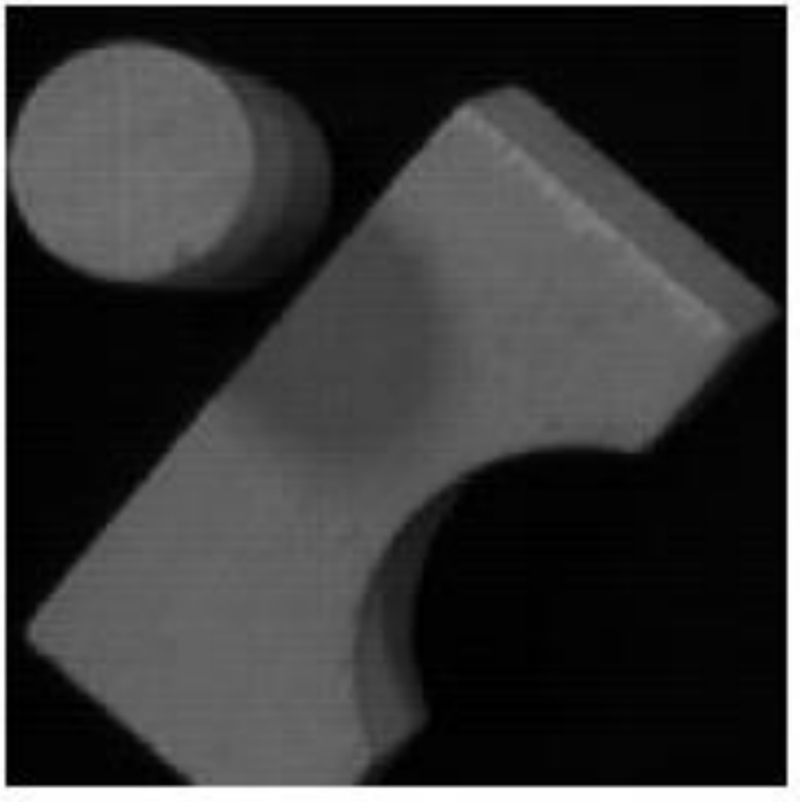
Black = no votes

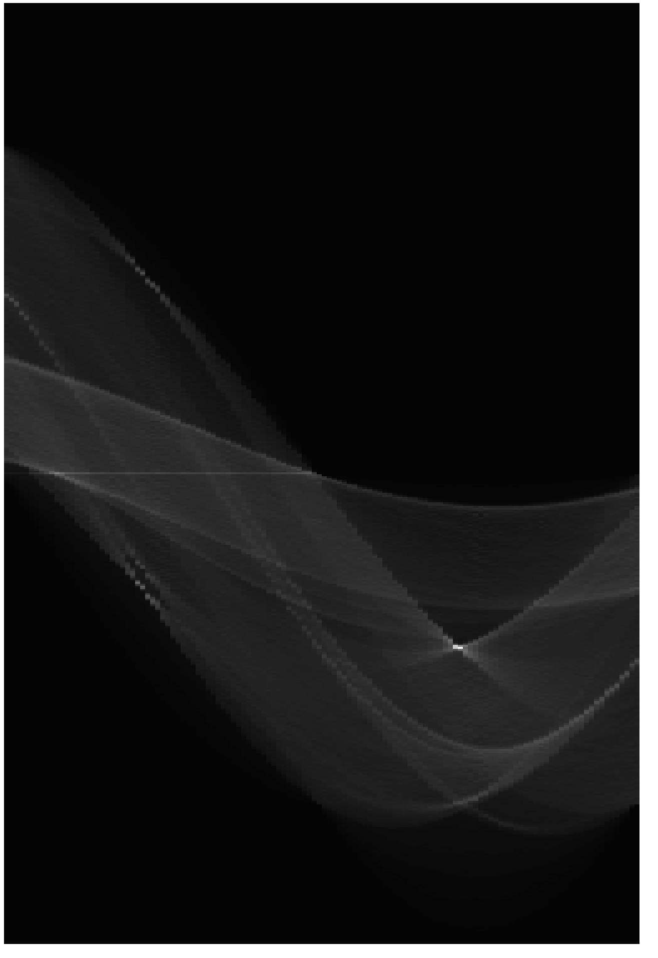
Example: HT for Straight Lines

Square:

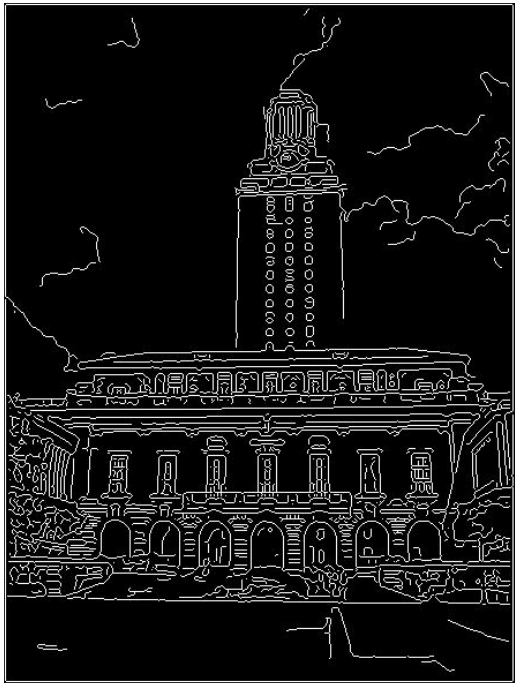


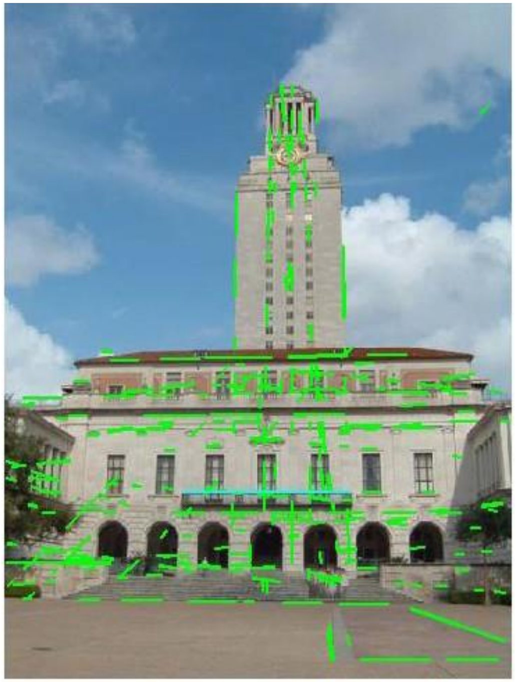
Example: HT for Straight Lines

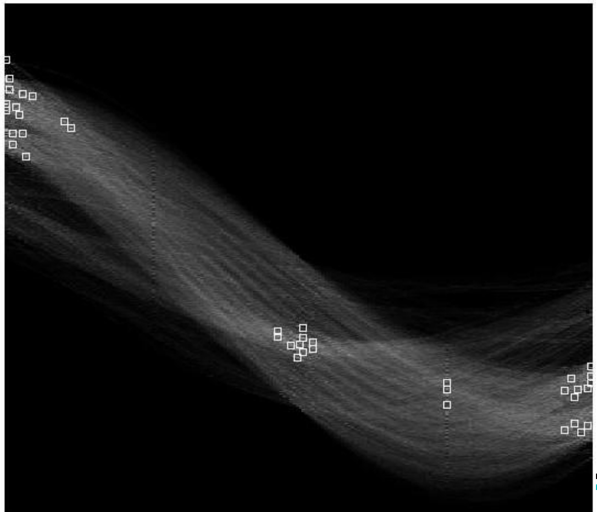


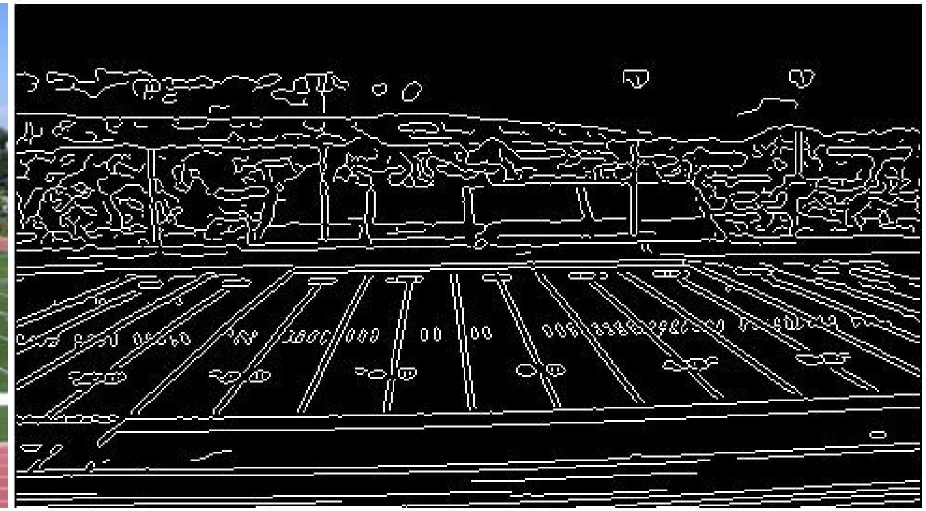


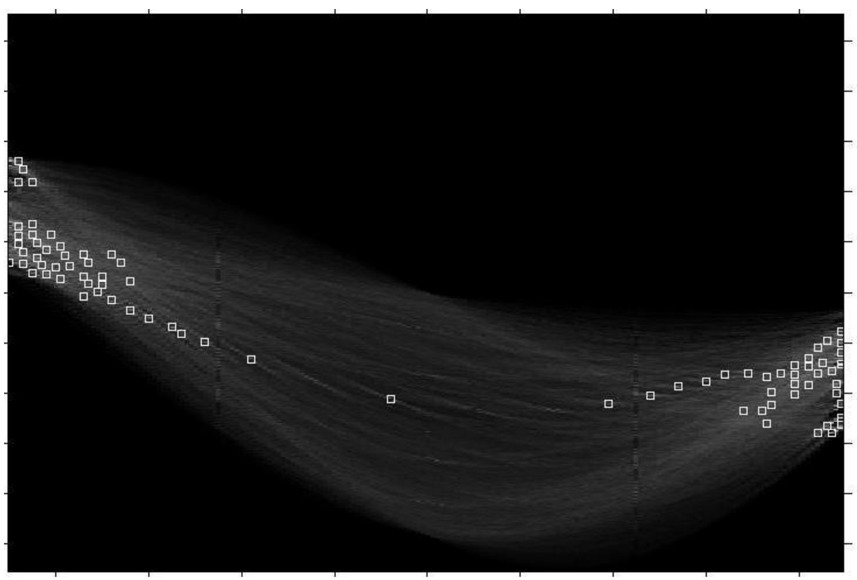
Real-World Examples

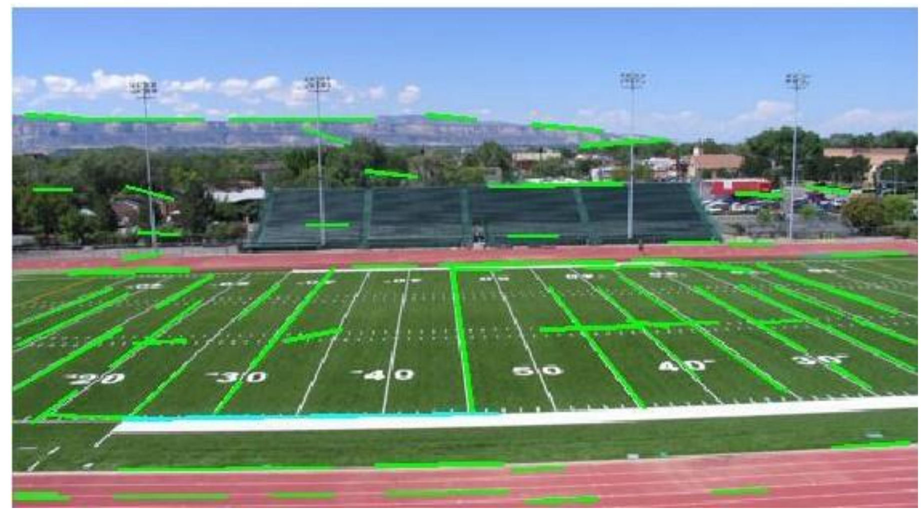












Showing longest segments found

Impact of Noise on Hough Transform

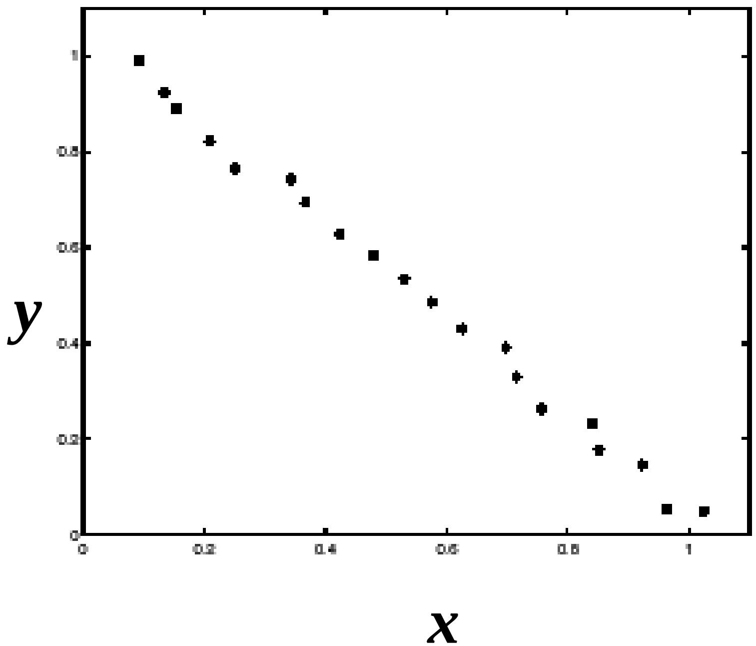
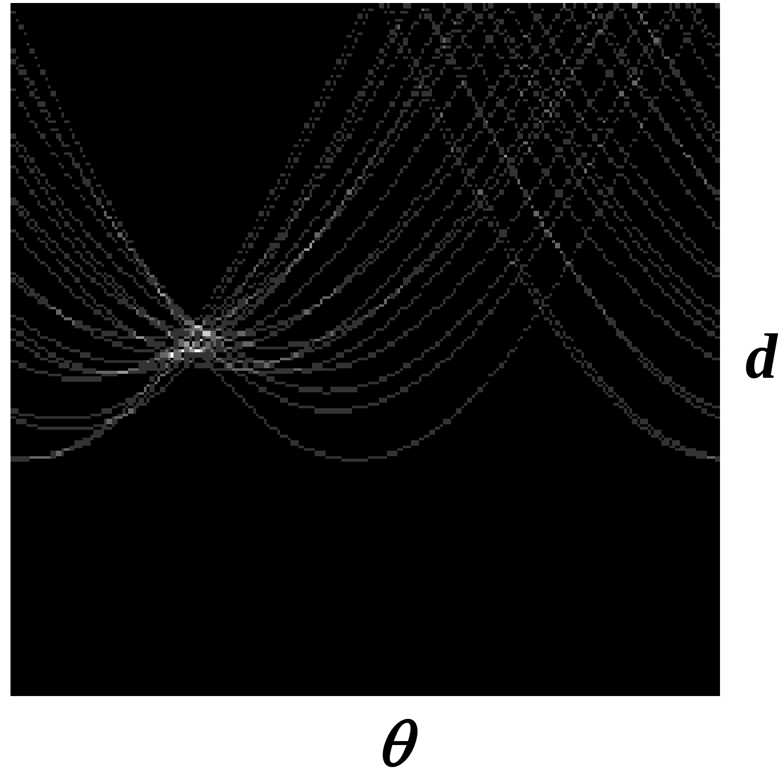


Image space edge coordinates



Votes

What difficulty does this present for an implementation?

Impact of Noise on Hough Transform

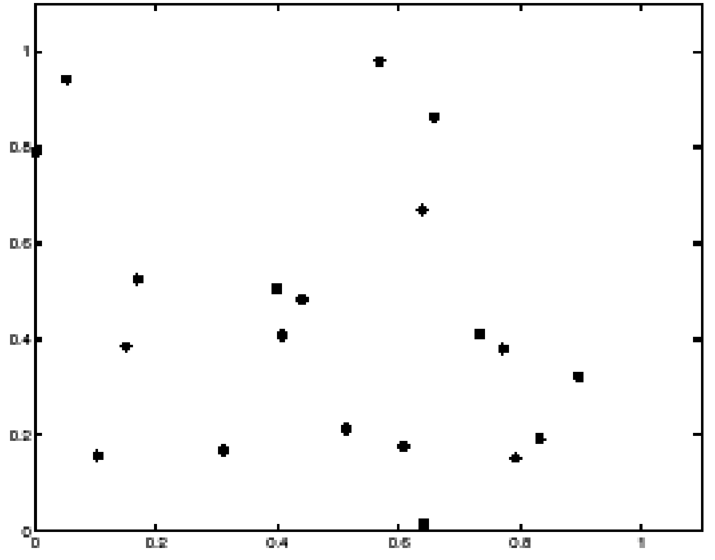
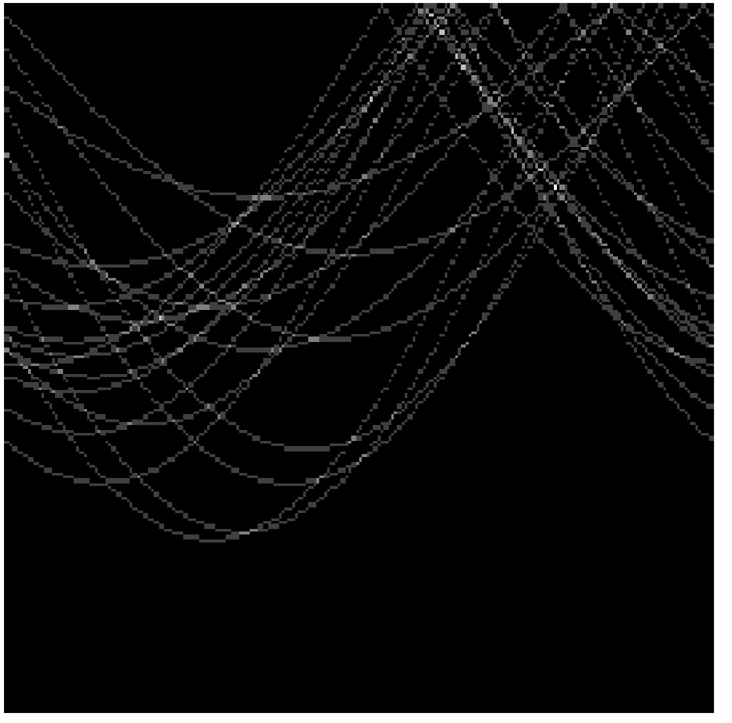


Image space edge coordinates



Votes

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

Generalized Hough Transform

- ***What if want to detect arbitrary shapes defined by boundary points and a reference point?***

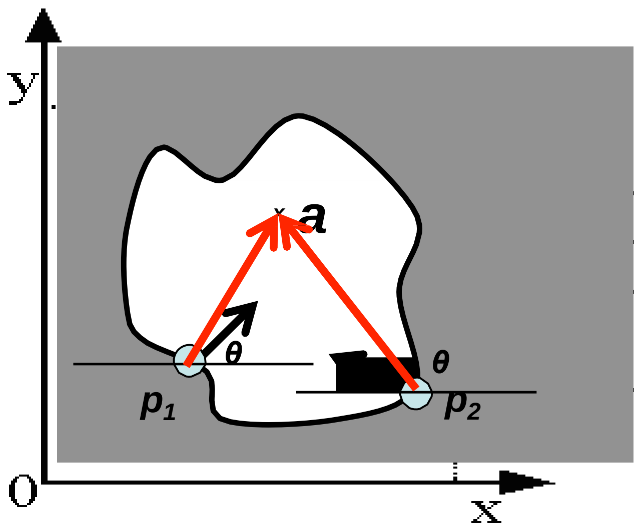


Image space

At each boundary point, compute displacement vector: $\mathbf{r} = \mathbf{a} - \mathbf{p}_i$.

For a given model shape: store these vectors in a table indexed by gradient orientation θ.

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

Voting: Practical Tips

- *Minimize irrelevant tokens first (take edge points with significant gradient magnitude)*
- *Choose a good grid / discretization*
  - *Too coarse: large votes obtained when 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 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***

Another model fitting strategy: RANSAC

[Fischler & Bolles 1981]

- ***RAN**dom **SA**mple **C**onsensus*

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

- ***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

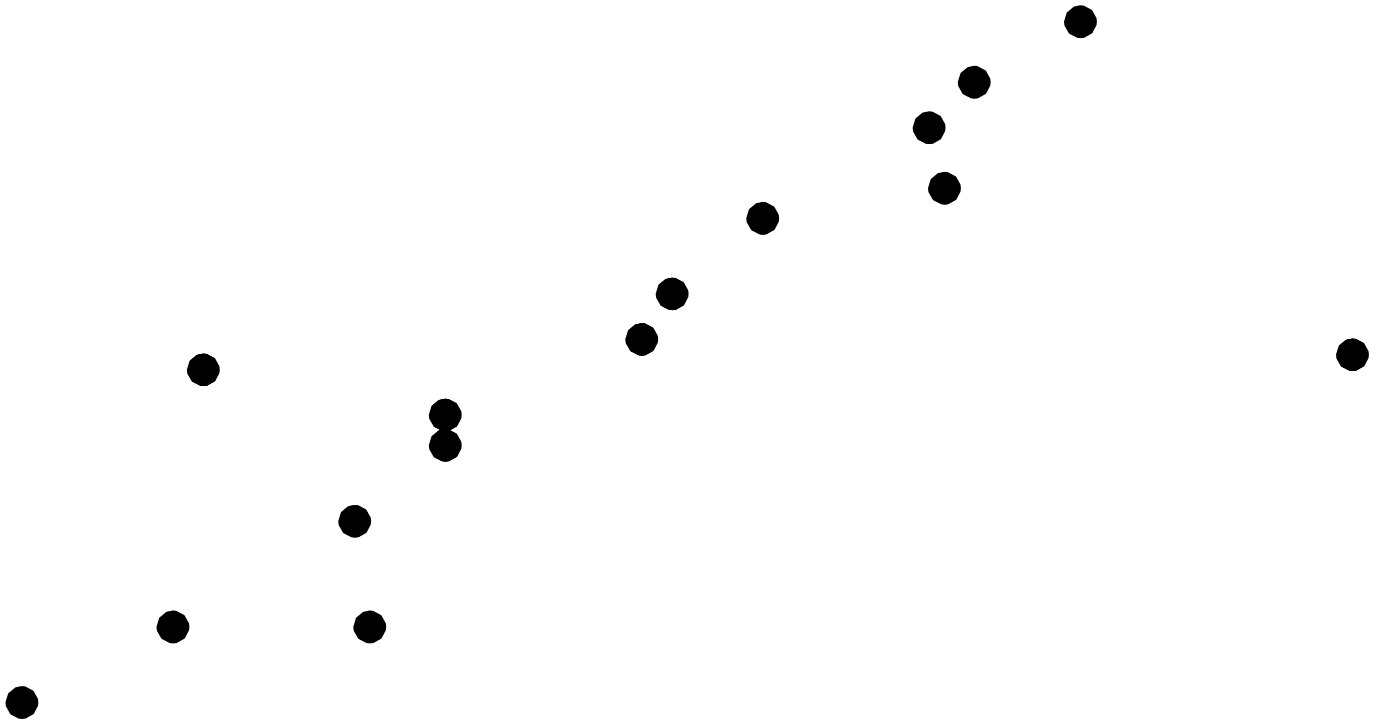
<u>RANSAC loop</u>:

1. ***Randomly select a seed group of points on which to base transformation estimate (e.g., a group of matches)***
2. ***Compute transformation from seed group***
3. ***Find inliers to this transformation***
4. ***If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers***

- ***Keep the transformation with the largest number of inliers***

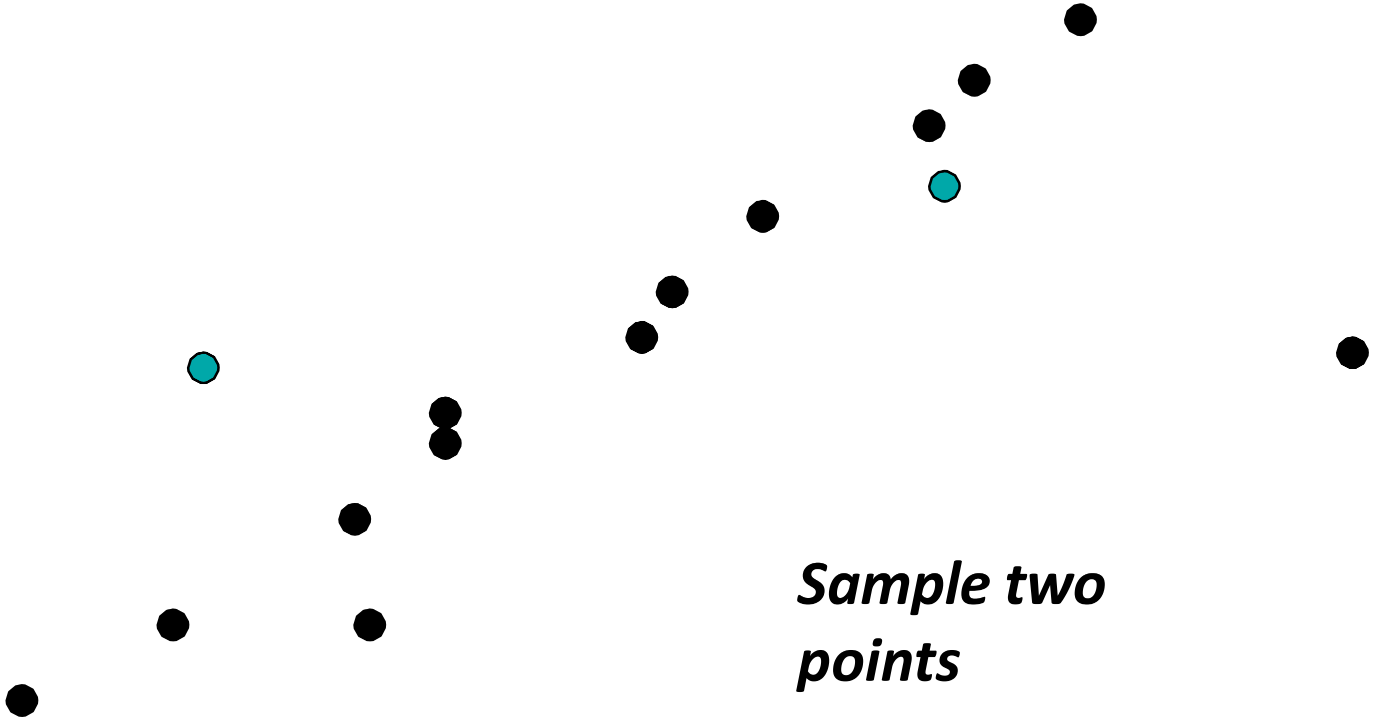
RANSAC Line Fitting Example

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



RANSAC Line Fitting Example

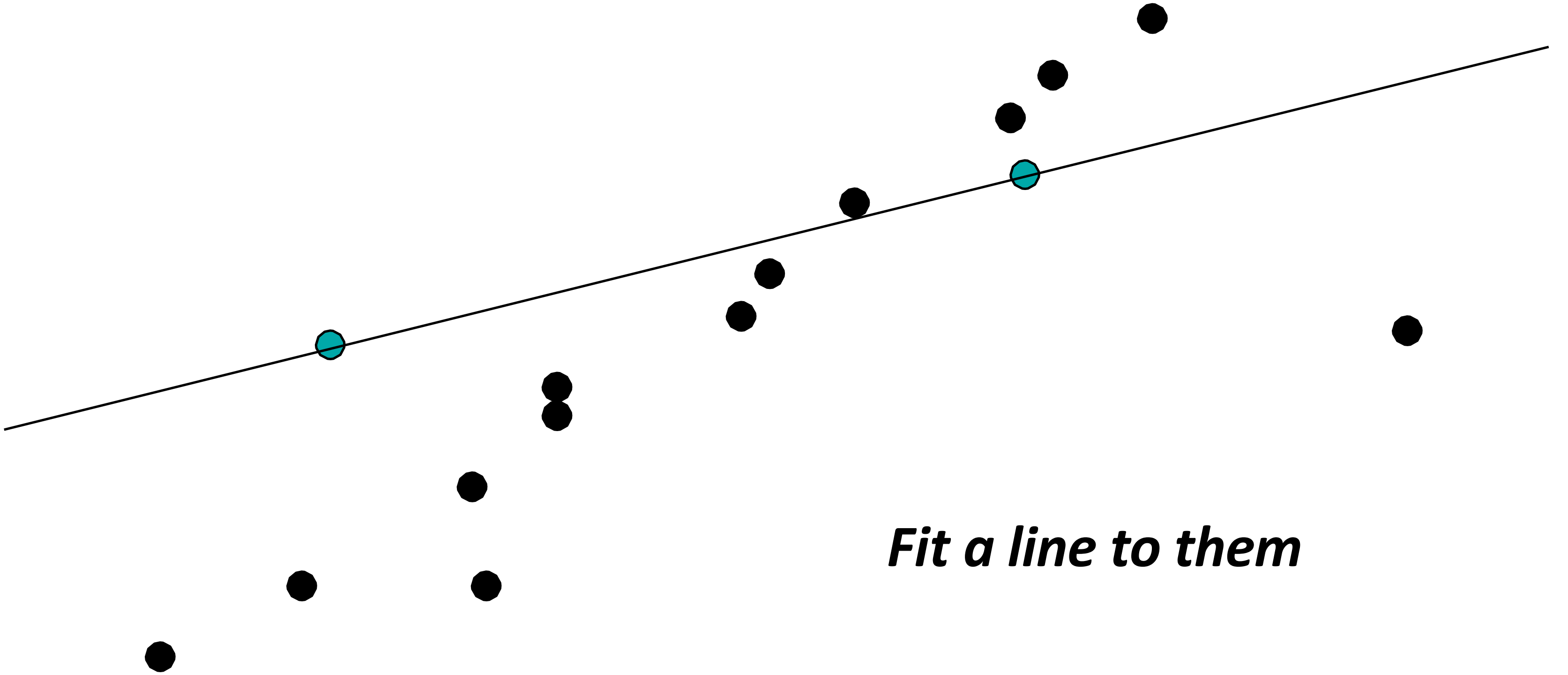
- *Task: Estimate the best line*



Sample two points

RANSAC Line Fitting Example

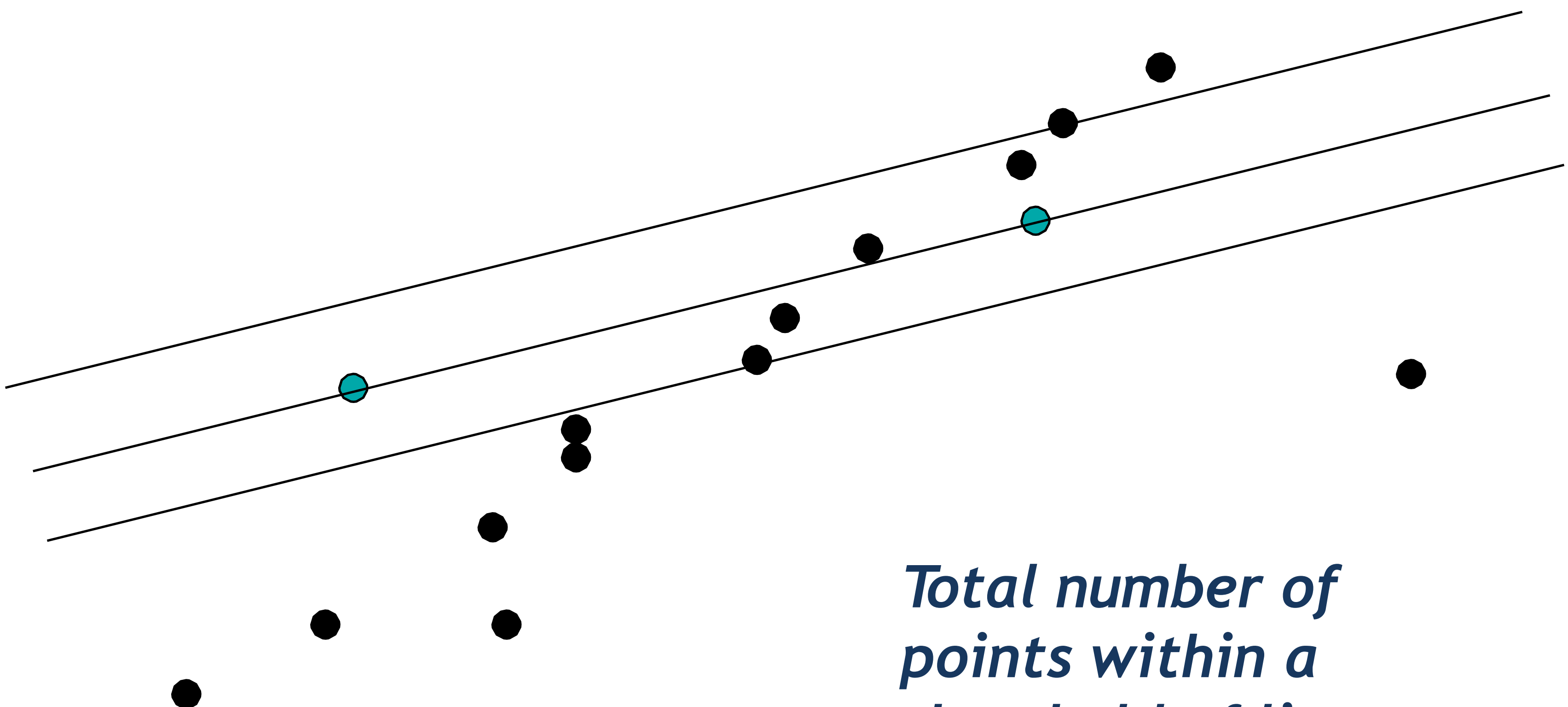
- *Task: Estimate the best line*



Fit a line to them

RANSAC Line Fitting Example

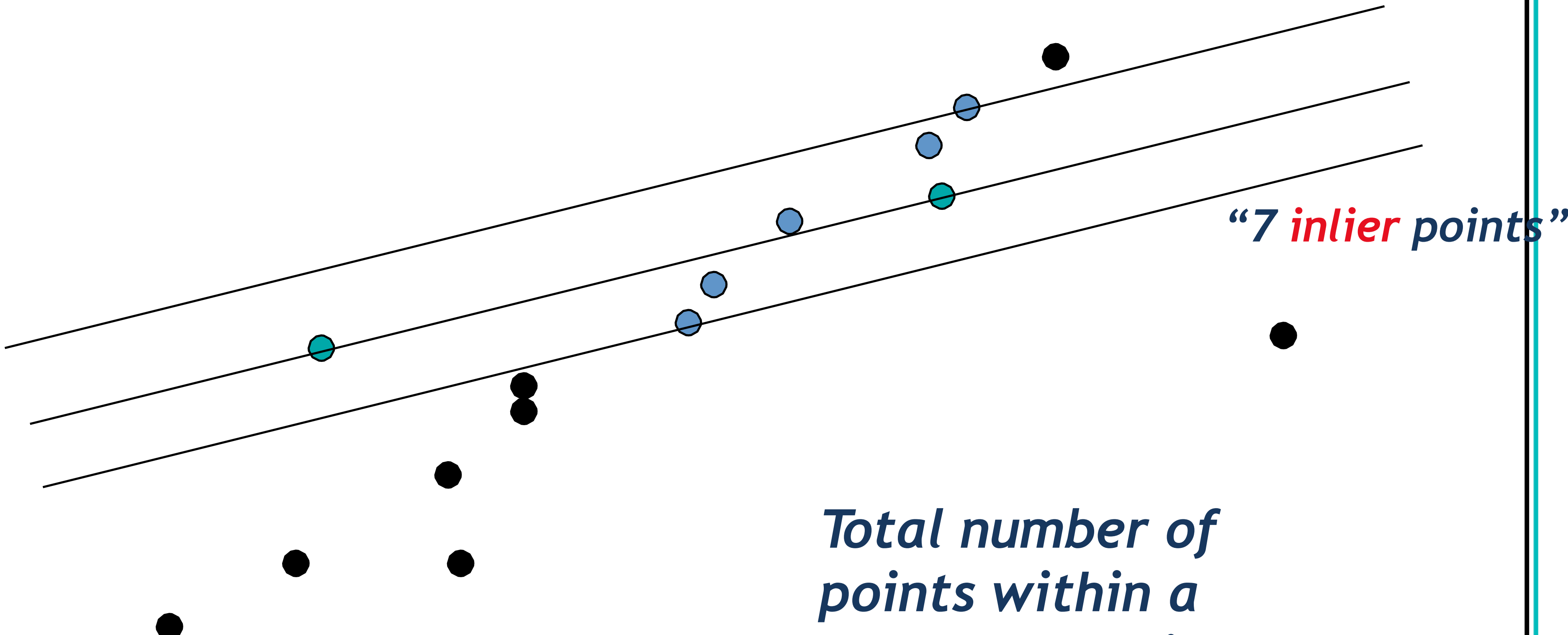
- ***Task: Estimate the best line***



Total number of points within a threshold of line.

RANSAC Line Fitting Example

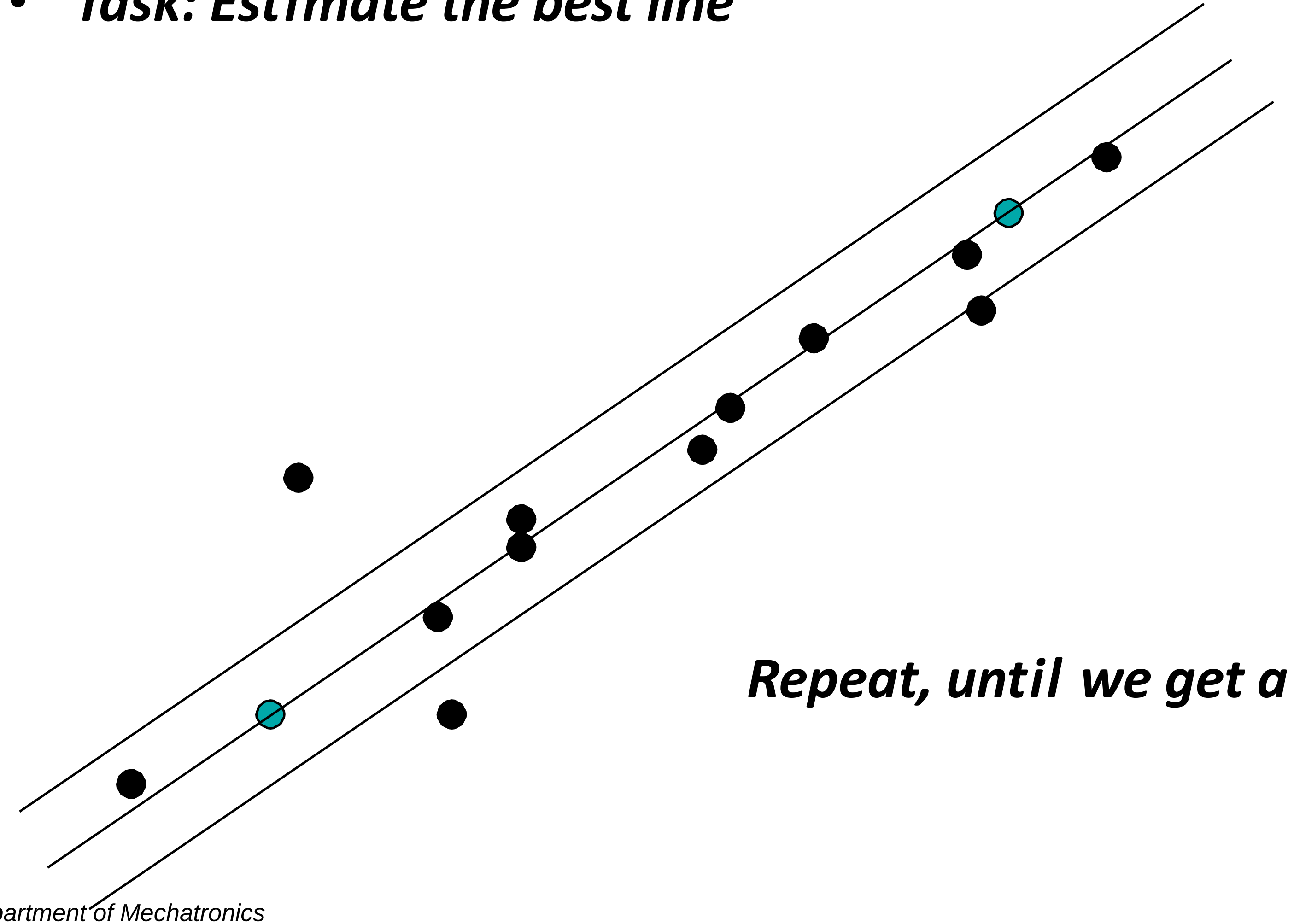
- ***Task: Estimate the best line***



Total number of points within a threshold of line.

RANSAC Line Fitting Example

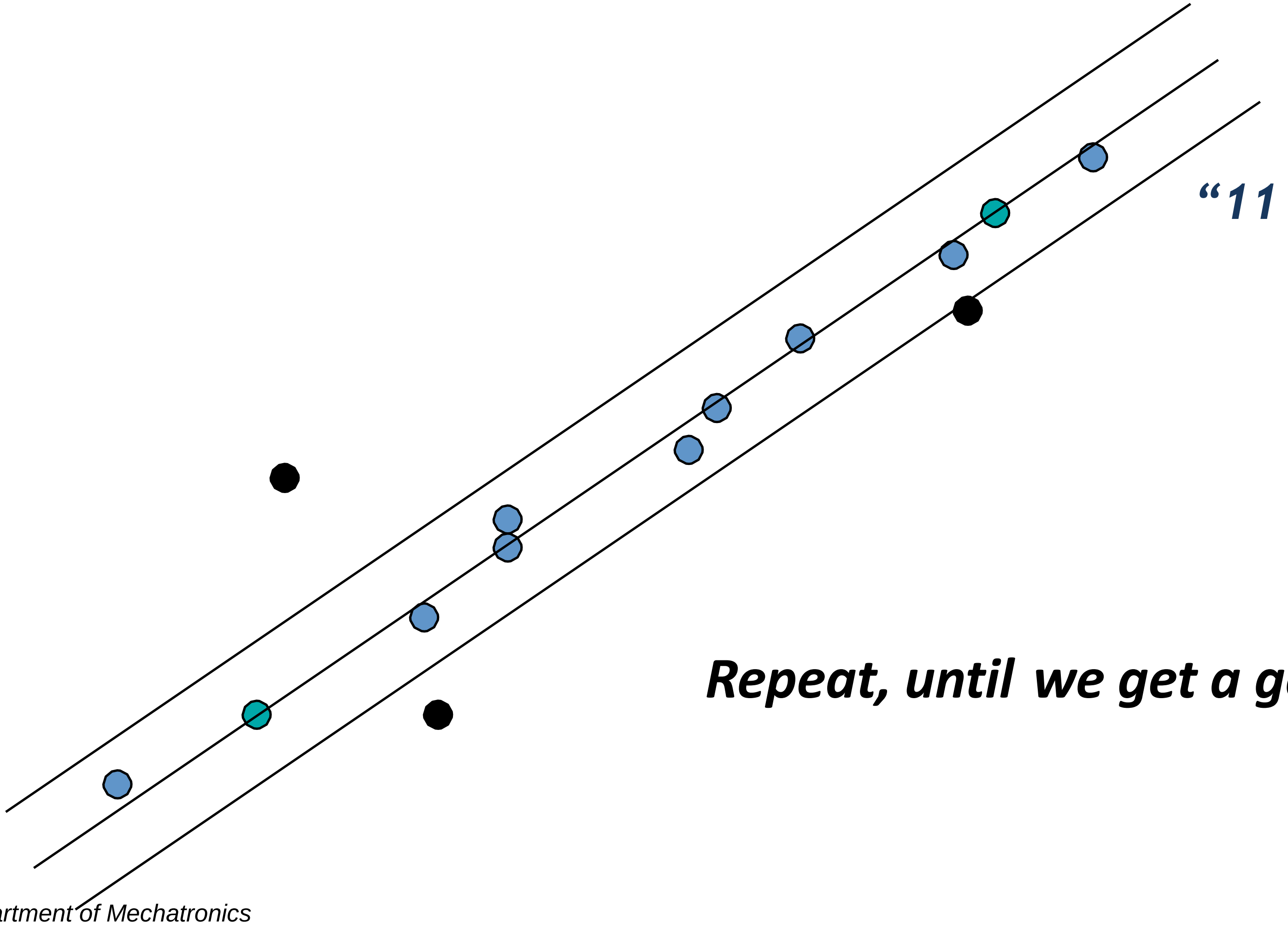
- ***Task: Estimate the best line***



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.

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 samples?

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

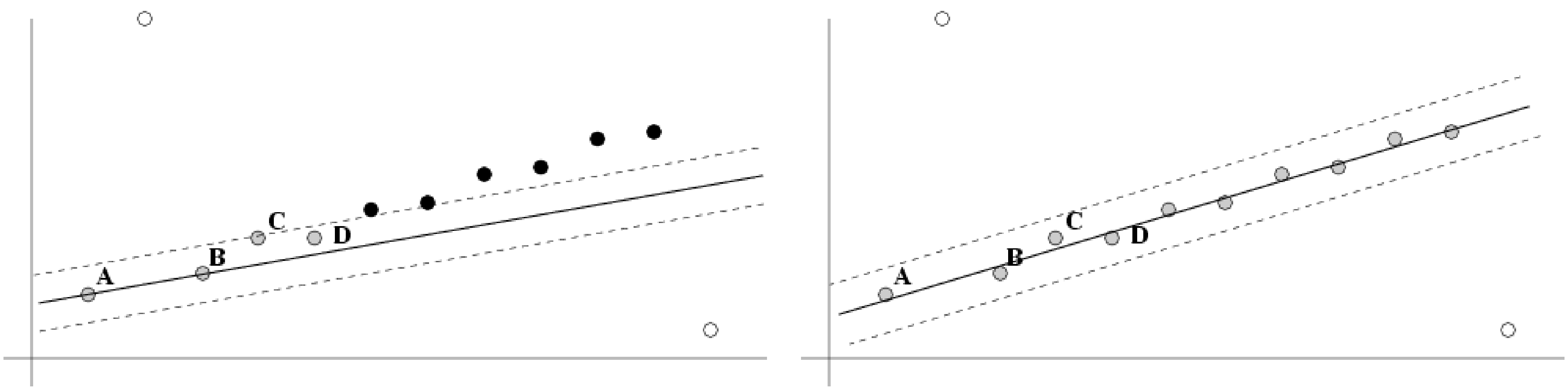
Choose k high enough to keep this below desired failure rate.

RANSAC: Computed k (p=0.99)

Sample size n	5%	10%	20%	25%	30%	40%	50%
			Proportion of outliers				
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

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



RANSAC: Pros and Cons

- ***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)***
- ***The Hough transform can handle high percentage of outliers***

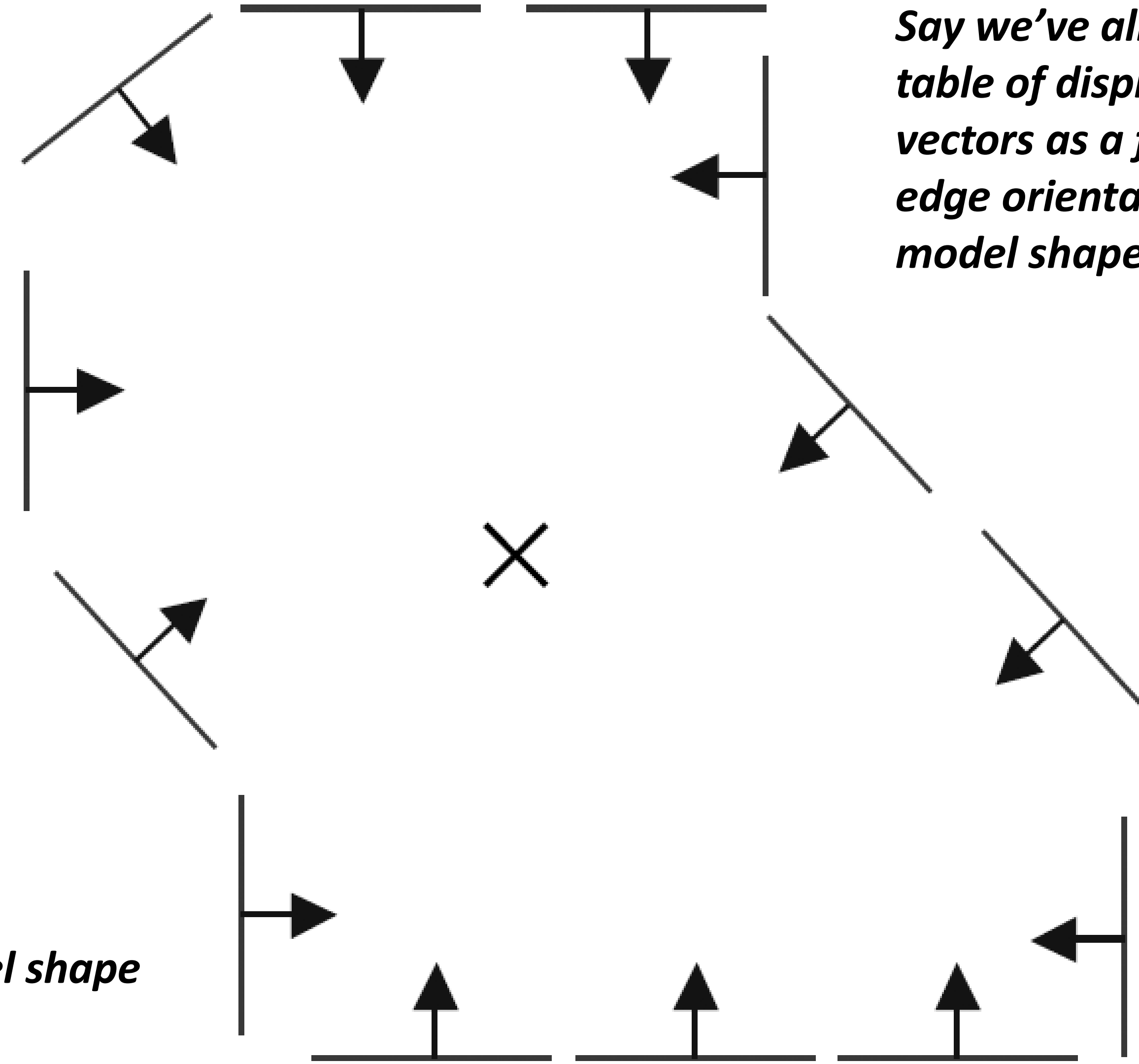
Generalized Hough Transform

To detect the model shape in a new image:

- *For each edge point*
  - *Index into table with its gradient orientation θ*
  - *Use retrieved $\mathbf{r}$ vectors to vote for position of reference point*
- *Peak in this Hough space is reference point with most supporting edges*

Assuming translation is the only transformation here, i.e., orientation and scale are fixed.

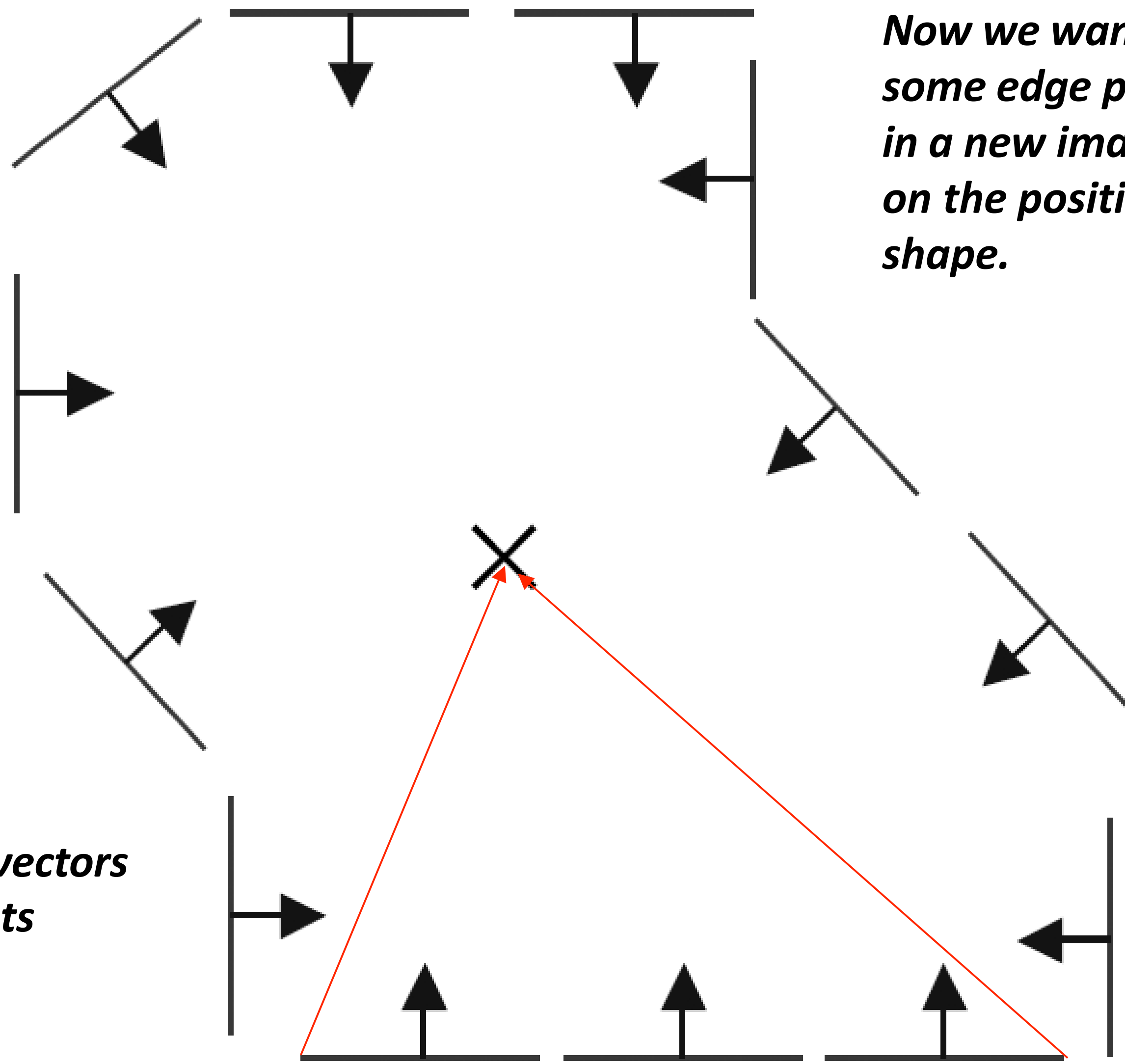
Example: Generalized Hough Transform



Say we've already stored a table of displacement vectors as a function of edge orientation for this model shape.

Model shape

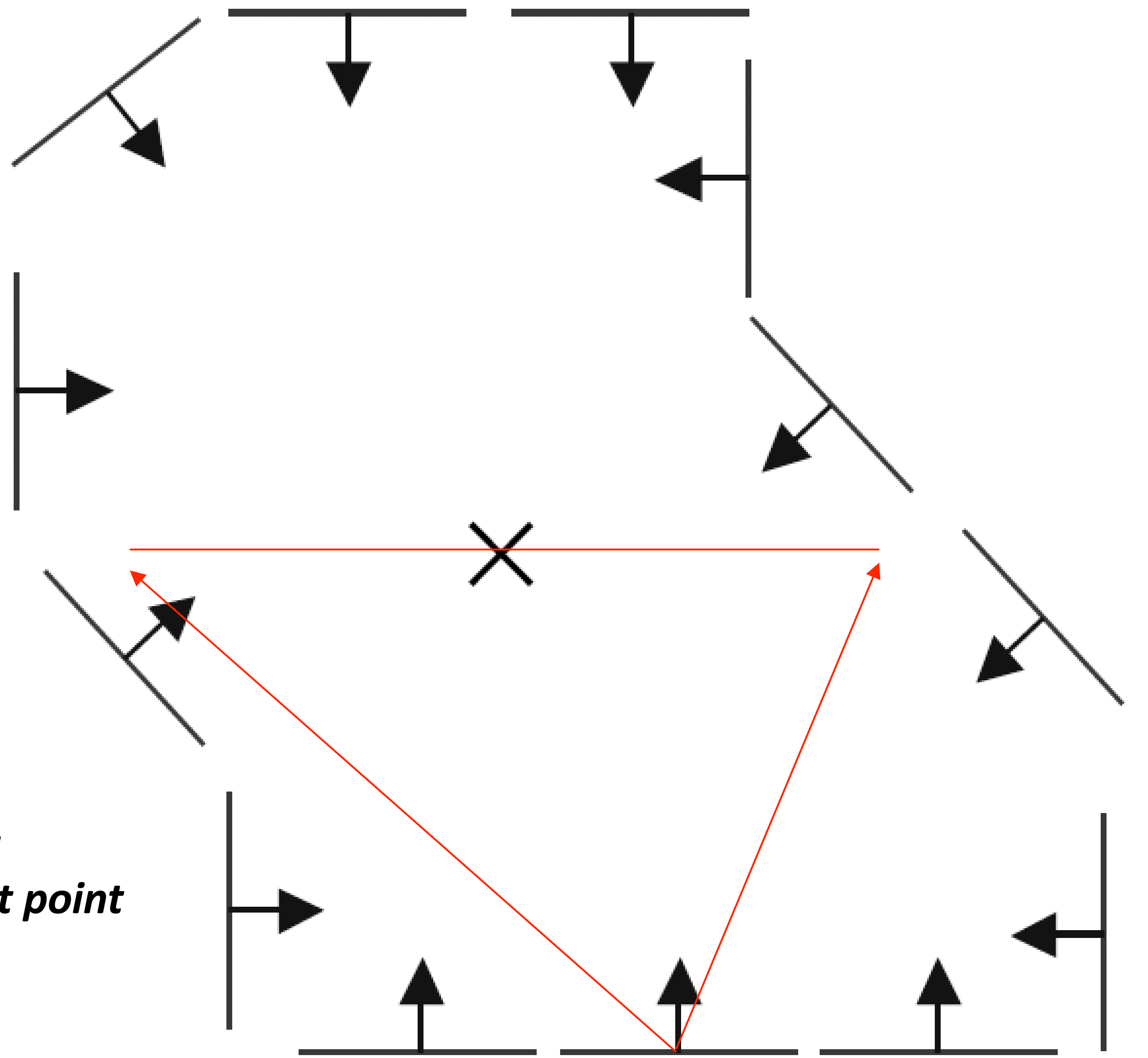
Example: Generalized Hough Transform



Now we want to look at some edge points detected in a new image, and vote on the position of that shape.

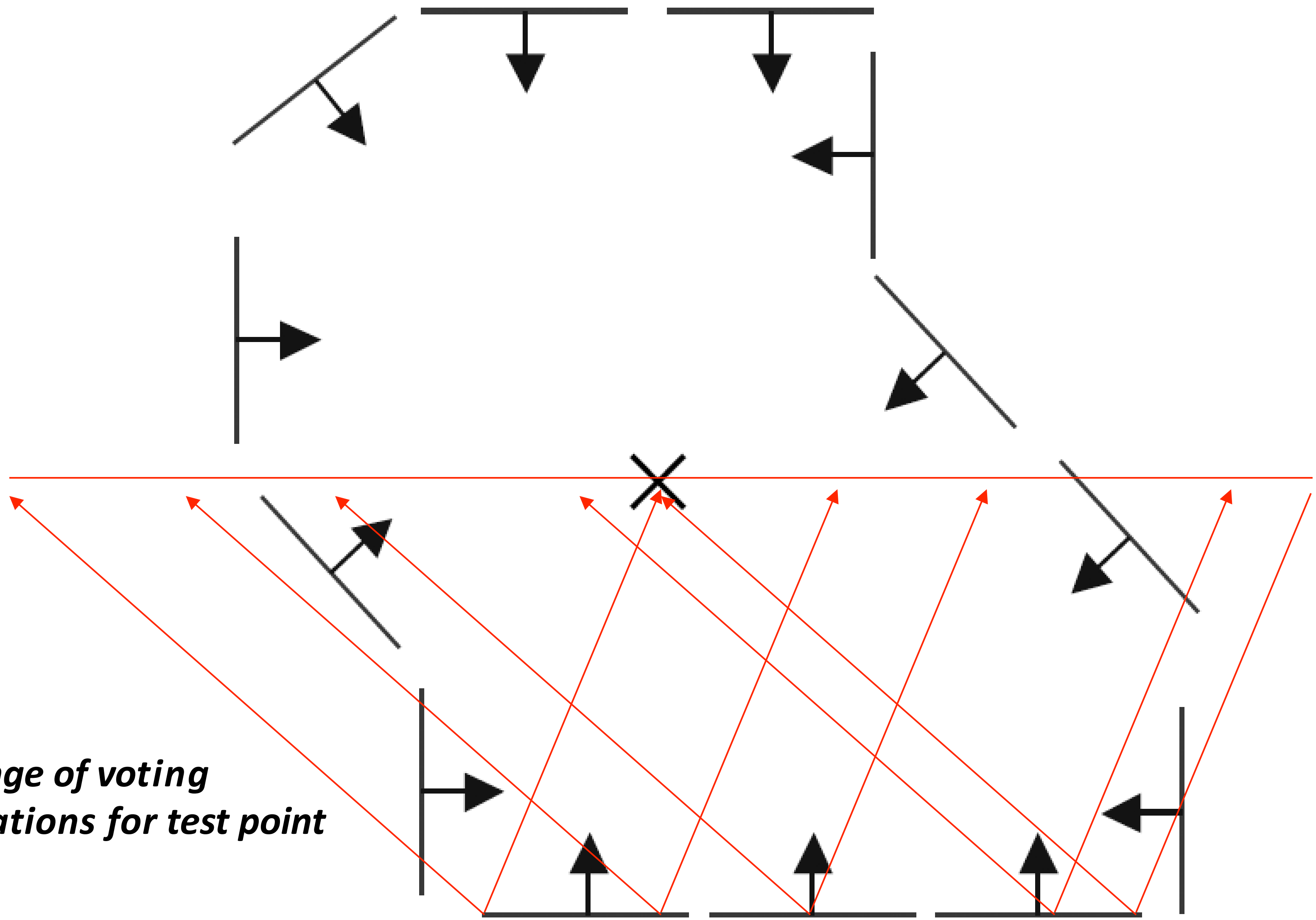
Displacement vectors for model points

Example: Generalized Hough Transform



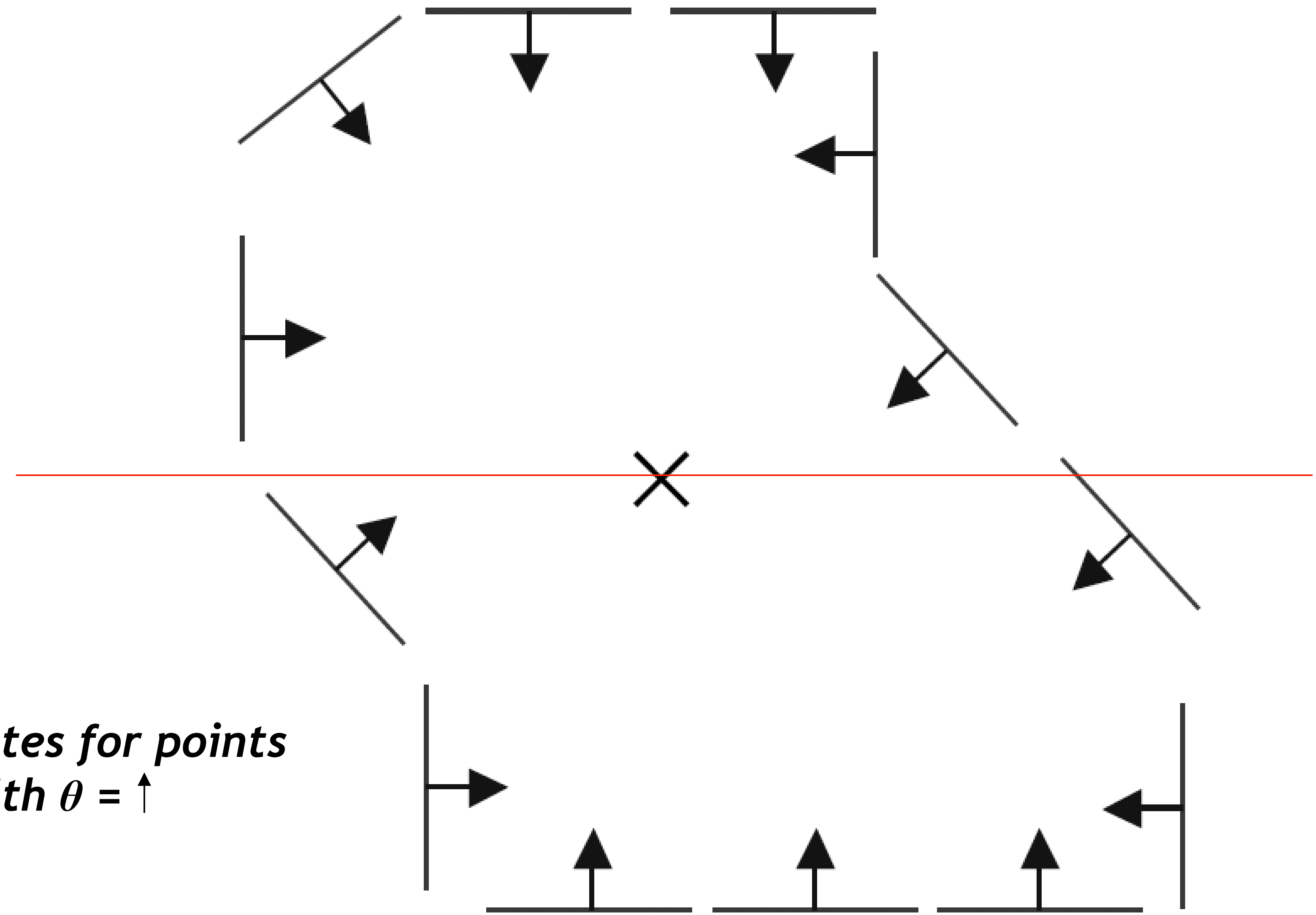
Range of voting locations for test point

Example: Generalized Hough Transform



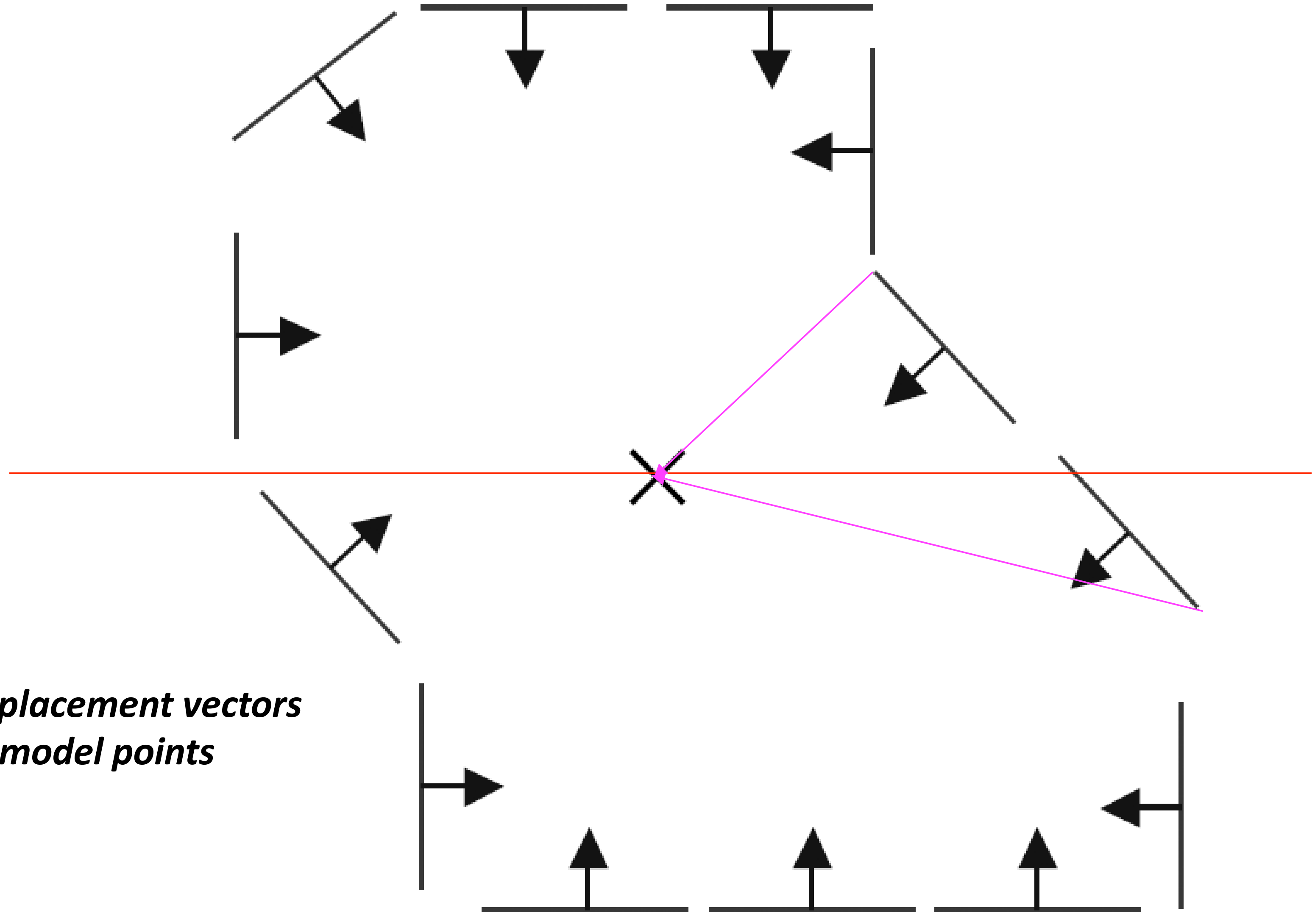
Range of voting locations for test point

Example: Generalized Hough Transform



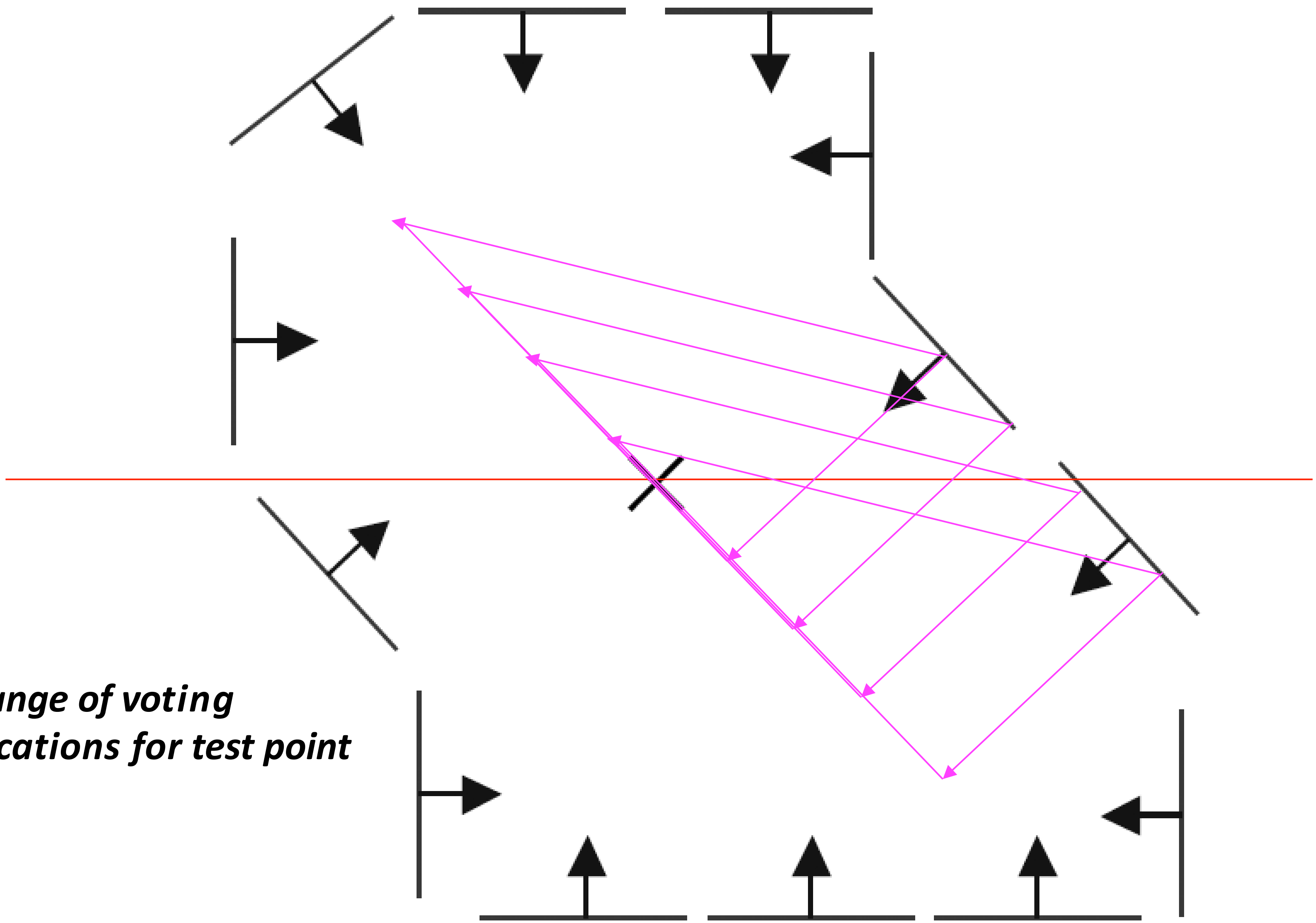
Votes for points with θ = ↑

Example: Generalized Hough Transform



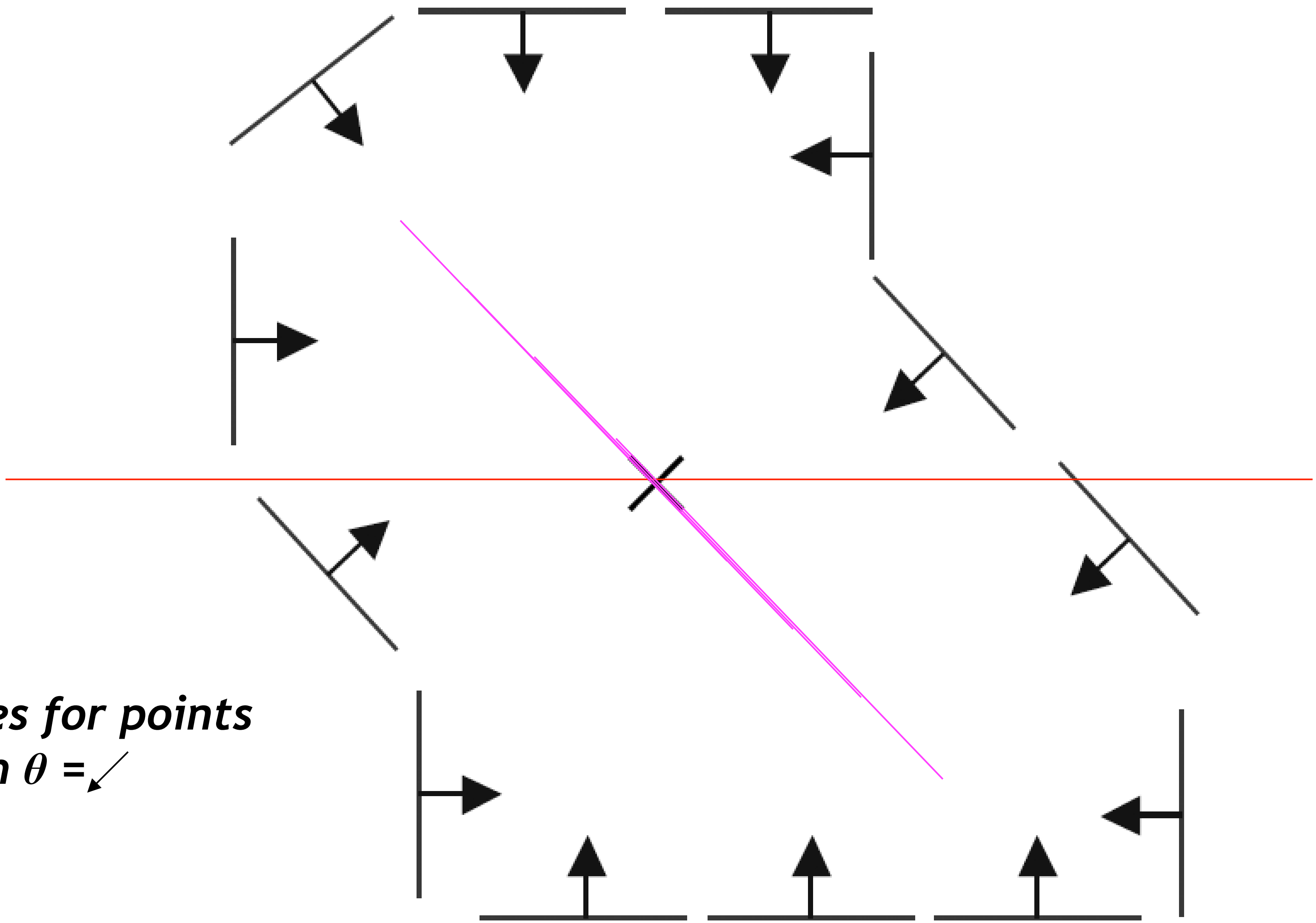
Displacement vectors for model points

Example: Generalized Hough Transform



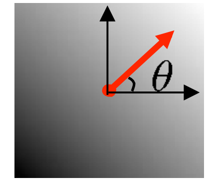
Range of voting locations for test point

Example: Generalized Hough Transform



Votes for points with θ = ↙

Extensions



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

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

Extension 1: Use the image gradient

1. ***same***
2. ***for each edge point I[x,y] in the image***

   θ = gradient at (x,y)

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

   $H[d,\theta]$ += 1
3. ***same***
4. ***same***

(Reduces degrees of freedom)

Extensions

Extension 1: Use the image gradient

1. *same*
2. *for each edge point $I[x,y]$ in the image*

   compute unique (d,θ) based on image gradient at (x,y)

   $H[d,\theta] \mathrel{+}= 1$
3. *same*
4. *same*

(Reduces degrees of freedom)

Extension 2

– *Give more votes for stronger edges (use magnitude of gradient)*

Extension 3

– *Change the sampling of (d,θ) to give more/less resolution*

Extension 4

– *The same procedure can be used with circles, squares, or any other shape…*

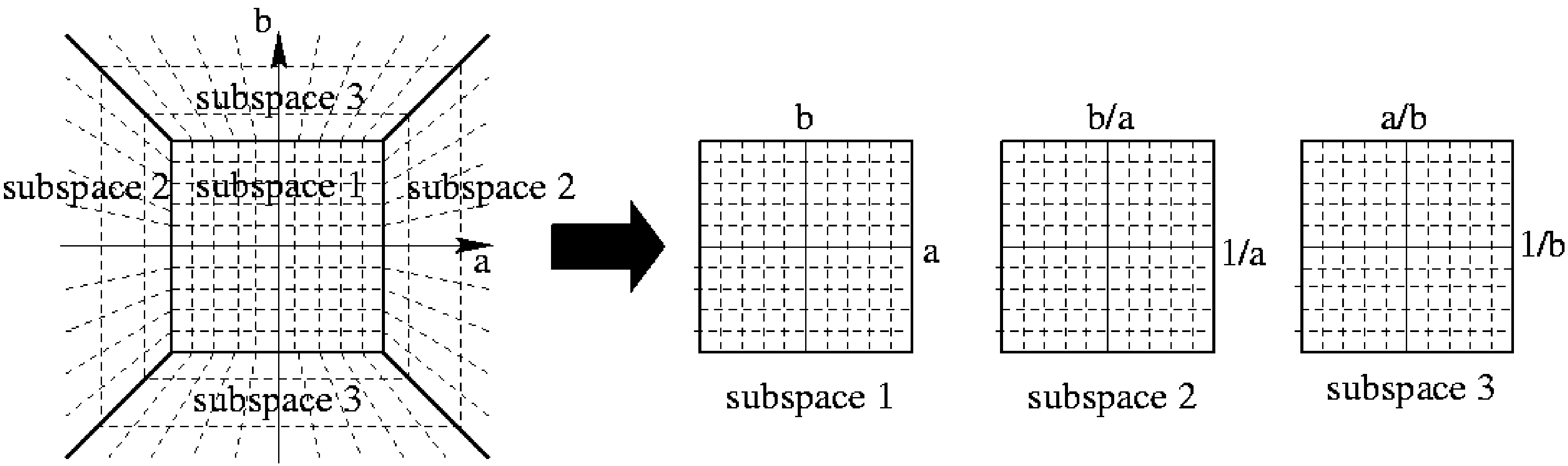
Extension: Cascaded Hough Transform

- **Let's go back to the original (m,b) parametrization**
- **A line in the image maps to a pencil of lines in the Hough space**
- **What do we get with parallel lines or a pencil of lines?**
  - **Collinear peaks in the Hough space!**
- **So we can apply a Hough transform to the output of the first Hough transform to find vanishing points**

- **T. Tuytelaars, M. Proesmans, L. Van Gool** *"The cascaded Hough transform", ICIP'97.*

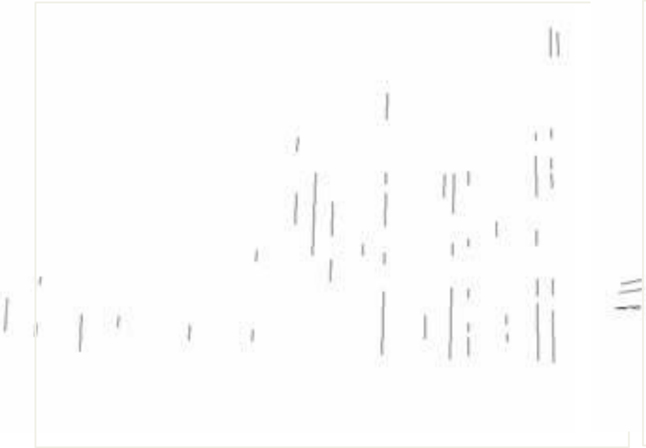
Finding Vanishing Points

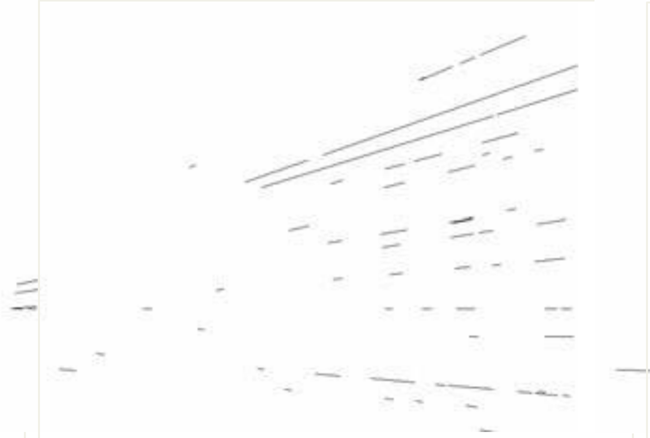
Cascaded Hough Transform

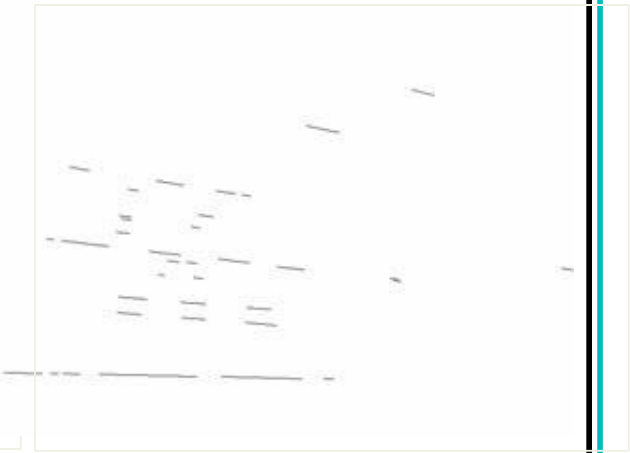


T. Tuytelaars, M. Proesmans, L. Van Gool "The cascaded Hough transform", ICIP'97.

Cascaded Hough Transform







T. Tuytelaars, M. Proesmans, L. Van Gool "The cascaded Hough transform", ICIP'97.

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***

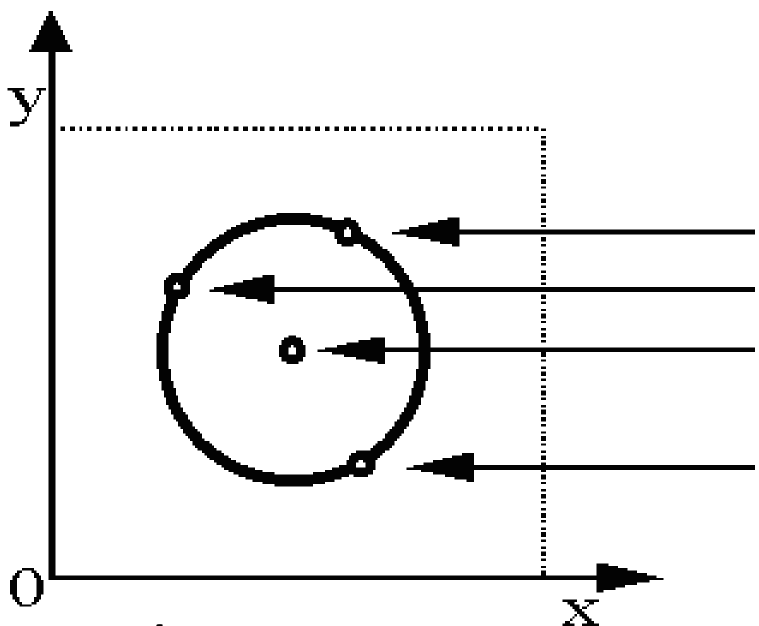
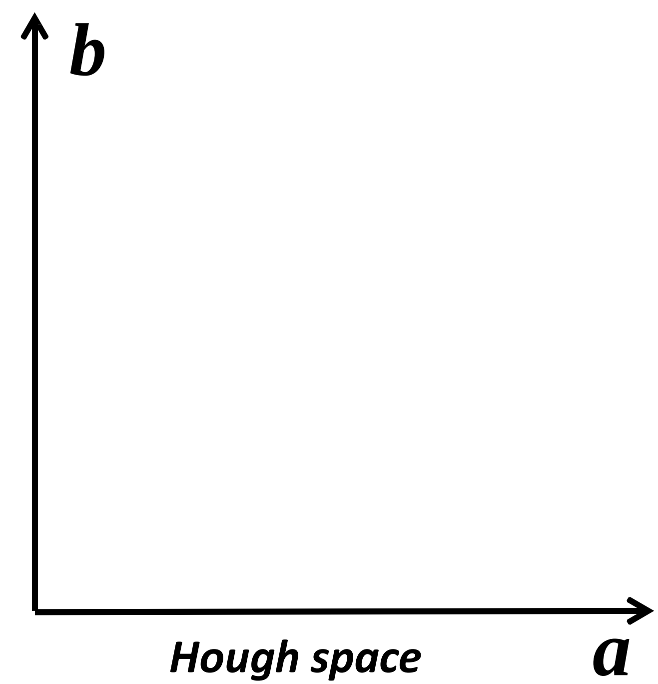


Image space



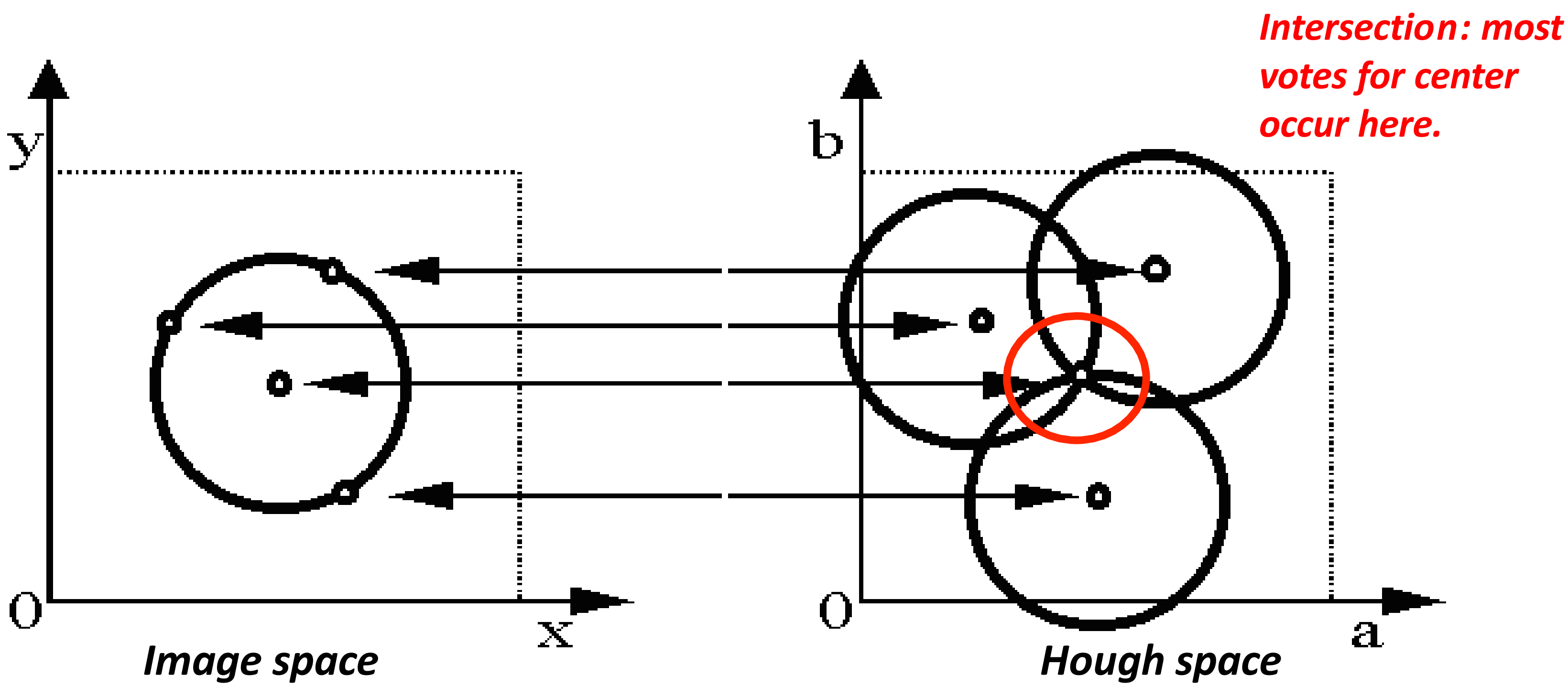
Hough space

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***



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, unknown gradient direction***

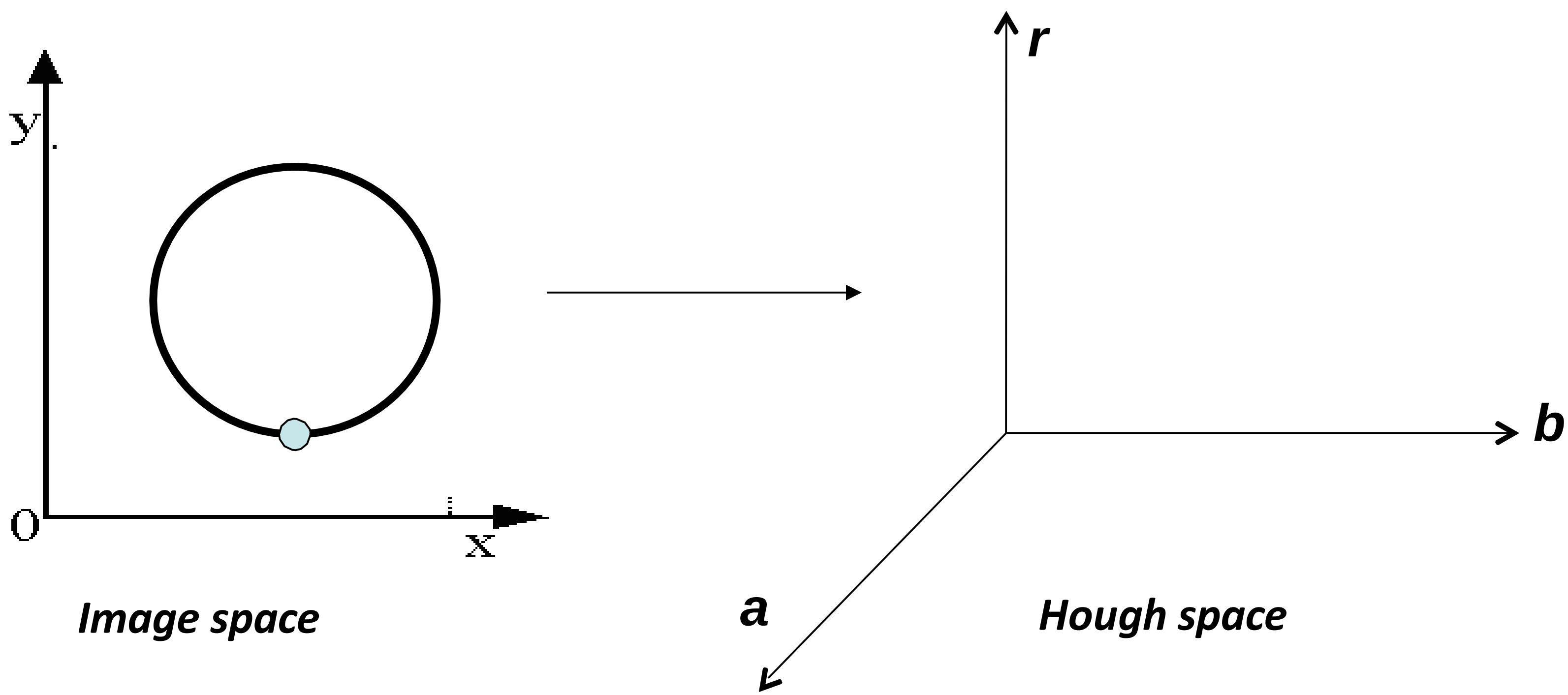


Image space

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, unknown gradient direction***

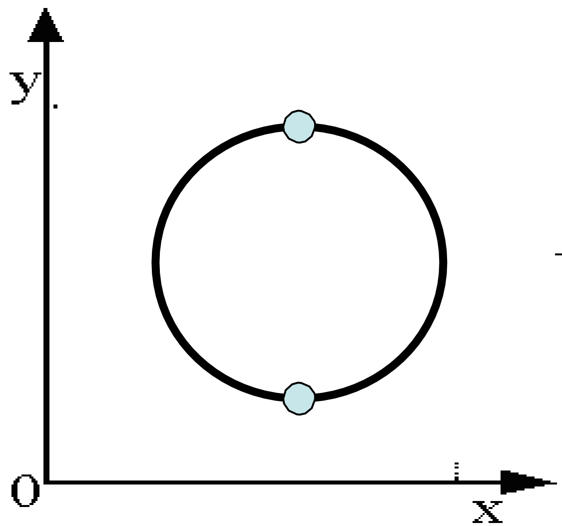
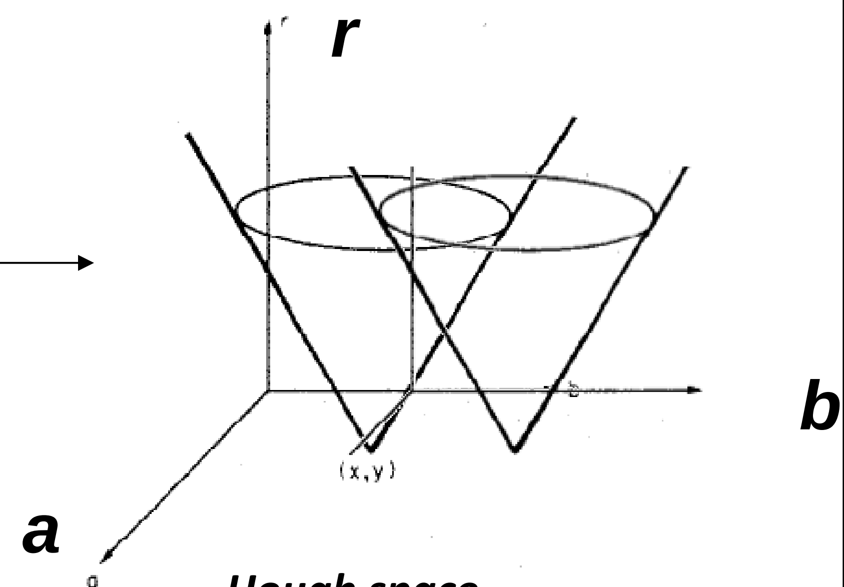


Image space



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*****,** ***known gradient direction***

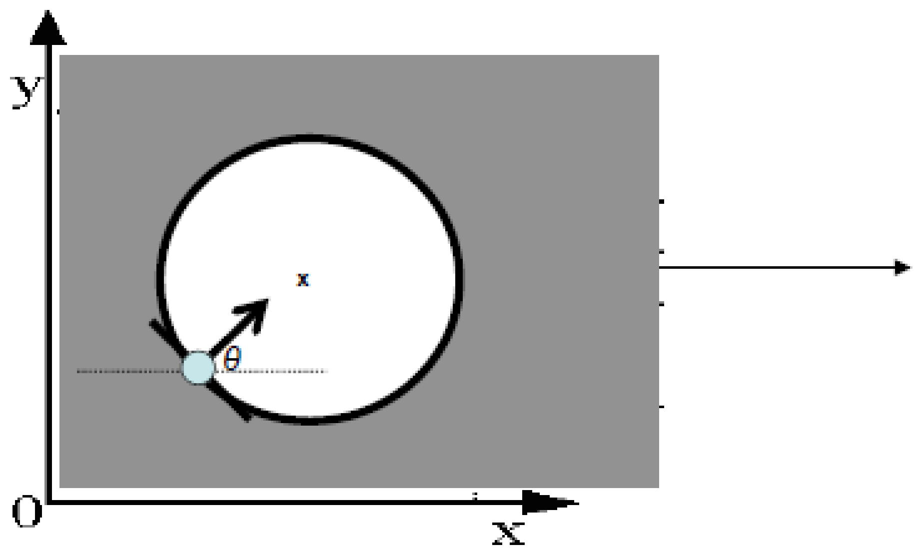
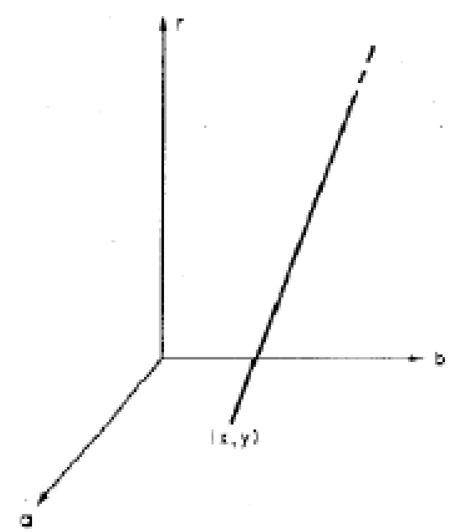


Image space



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

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

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

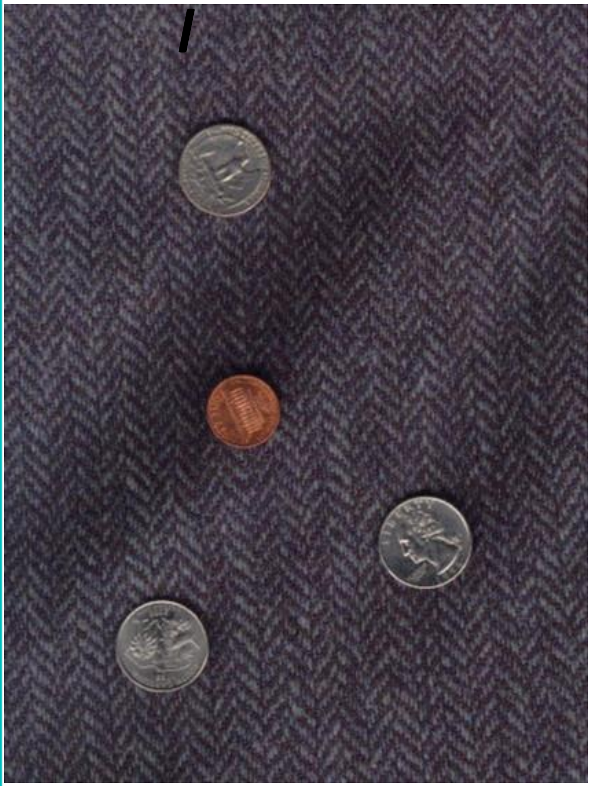
$H[a,b,r] \mathrel{+}= 1$

end

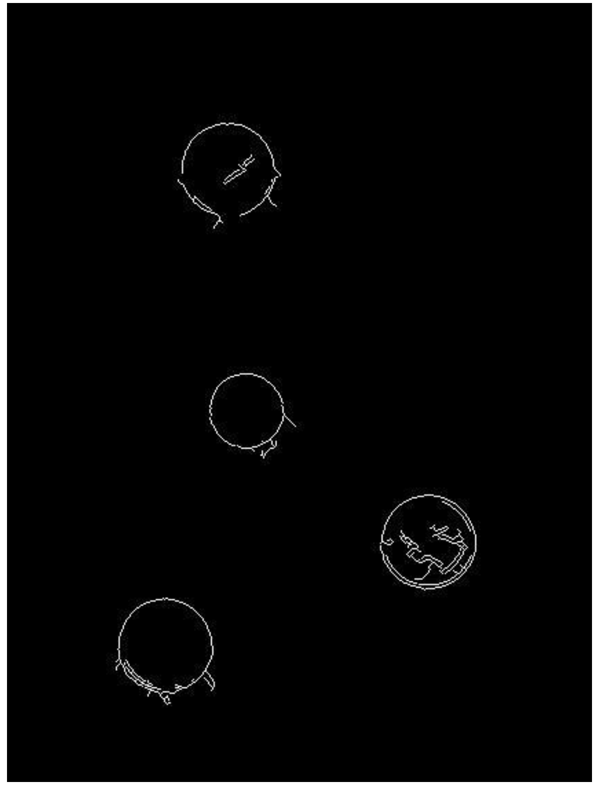
end

Example: Detecting Circles with Hough

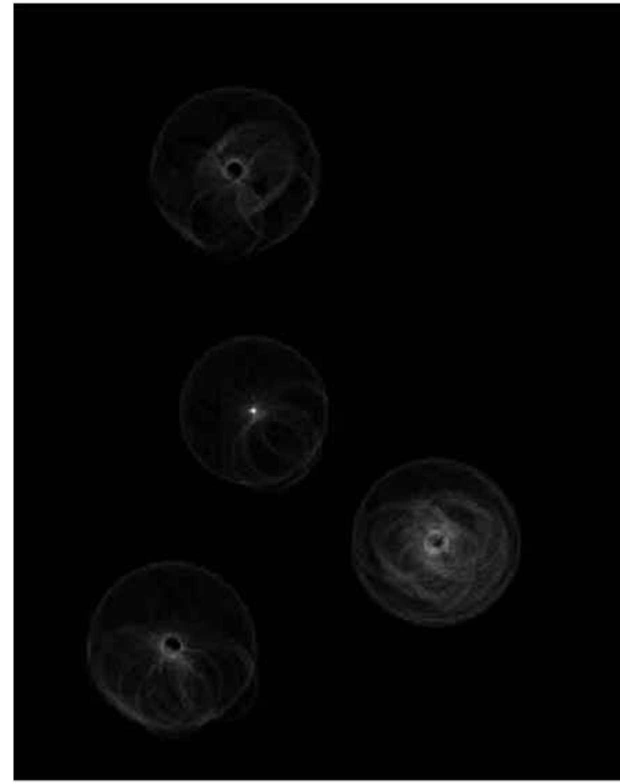
Origina
l



Edges



Votes: Penny

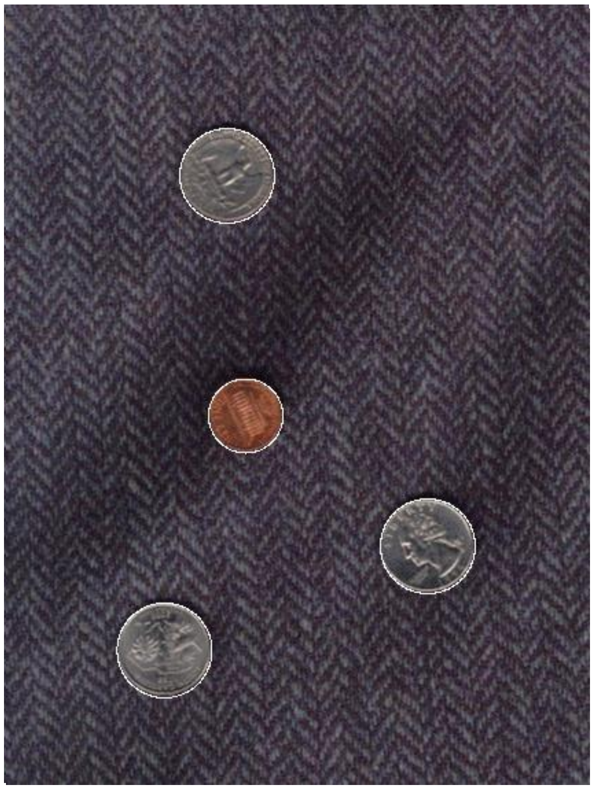


Note: a different Hough Transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

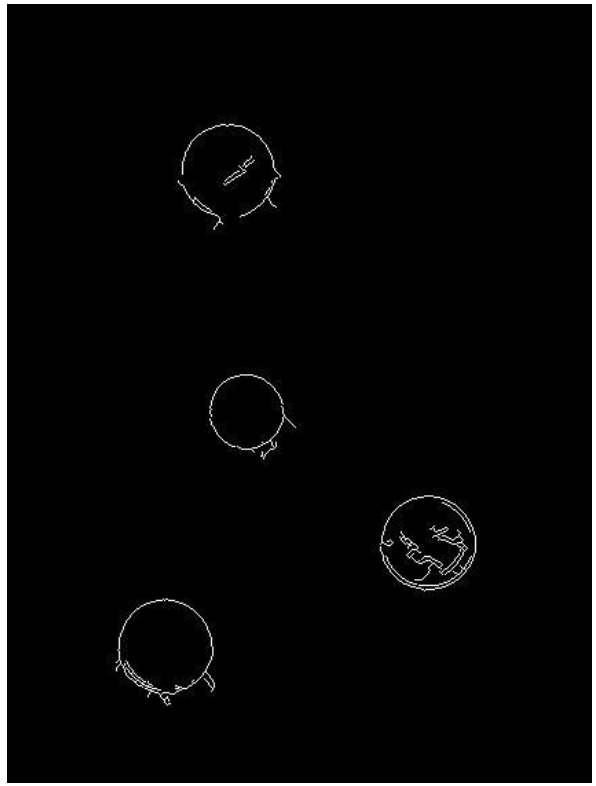
Example: Detecting Circles with Hough

Combined detections

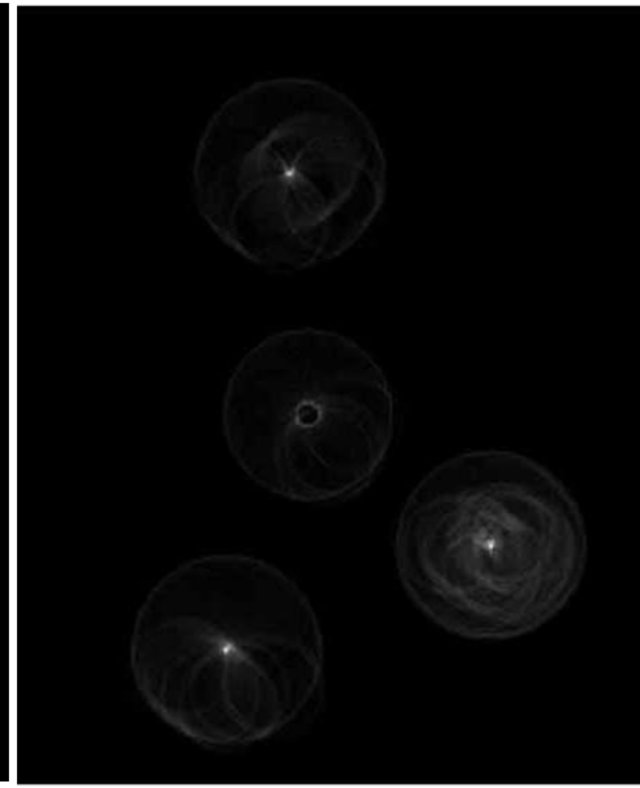
Origina



Edges



Votes: Quarter



Coin finding sample images from: Vivek Kwatra

Example: Detecting Circles with Hough

Crosshair indicates results of Hough transform, bounding box found via motion differencing.