

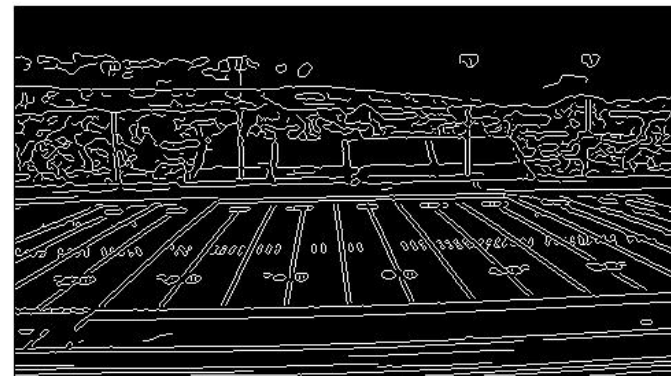
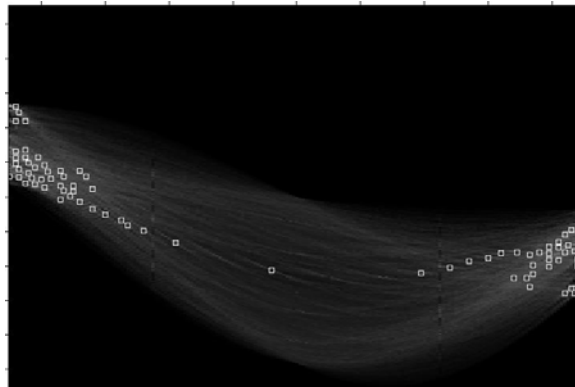
# CS 4495 Computer Vision

## *Finding 2D Shapes and the Hough Transform*

---

Aaron Bobick

School of Interactive  
Computing



# Administrivia

- Today: Modeling Lines and Finding them
- Problem set 1 is posted.
  - You **can** use Matlab edge operators
  - You **cannot** use Matlab Hough methods.
  - Due Sunday, Sept 8<sup>th</sup> 11:55pm.

## Now some “real” vision...

- So far, we applied operators/masks/kernels to images to produce new image

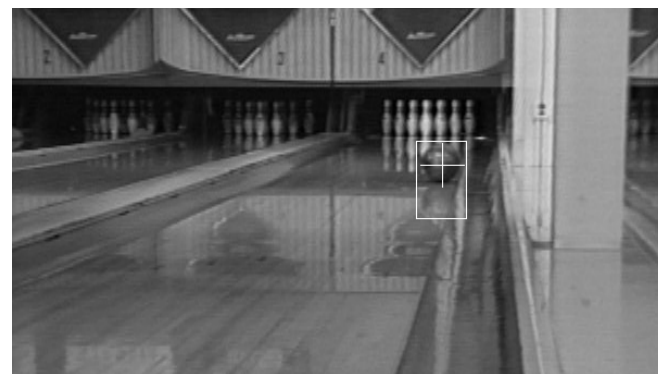
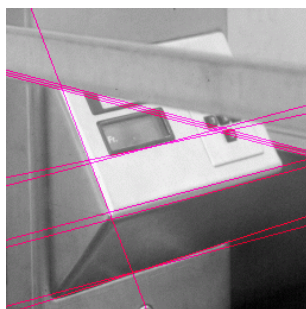
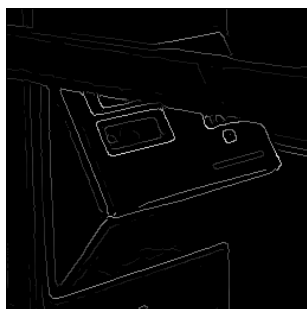
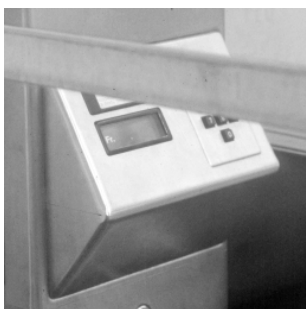
Image processing:  $F : I(x, y) \longrightarrow I'(x, y)$

- Now real vision:

$$F : I(x, y) \longrightarrow \text{good stuff}$$

# Fitting a model

- Want to associate a model with observed features



[Fig from Marszalek & Schmid, 2007]

For example, the model could be a line, a circle, or an arbitrary shape.

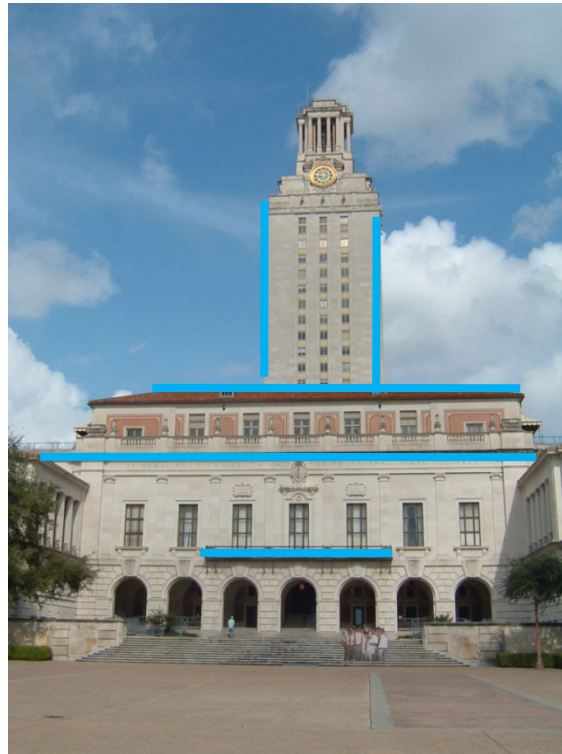
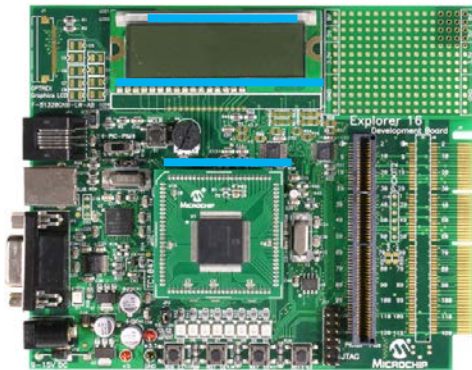
# Fitting

- Choose a ***parametric model*** to represent a set of features
- Membership criterion is not local
  - Can't tell whether a point in the image belongs to a given model just by looking at that point
- Three main questions:
  1. What model represents this set of features best?
  2. Which of several model instances gets which feature?
  3. 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?

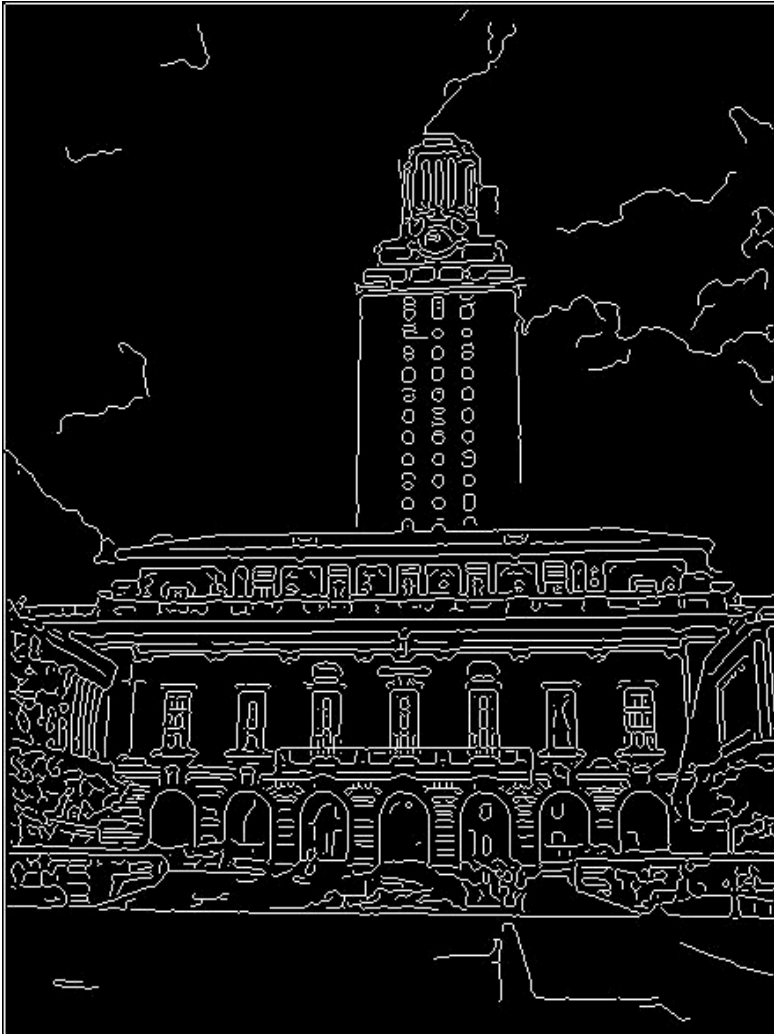
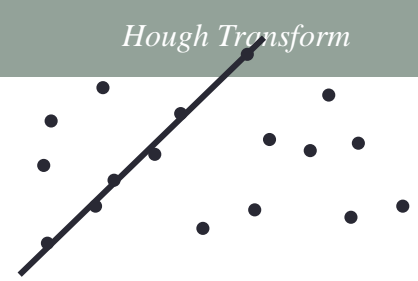
*Many objects characterized by presence of straight lines*



- 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?

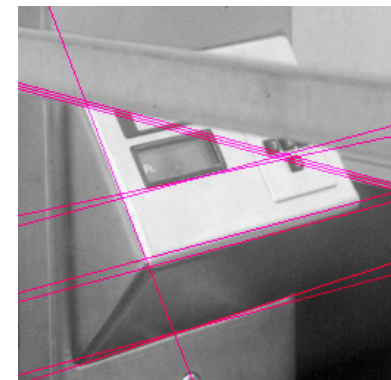
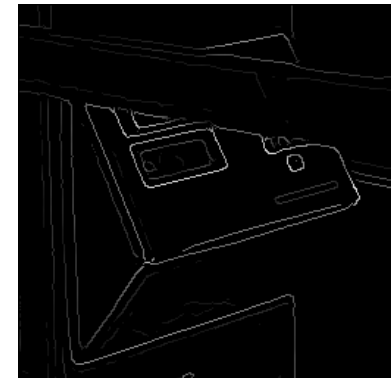
# Voting

- 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.

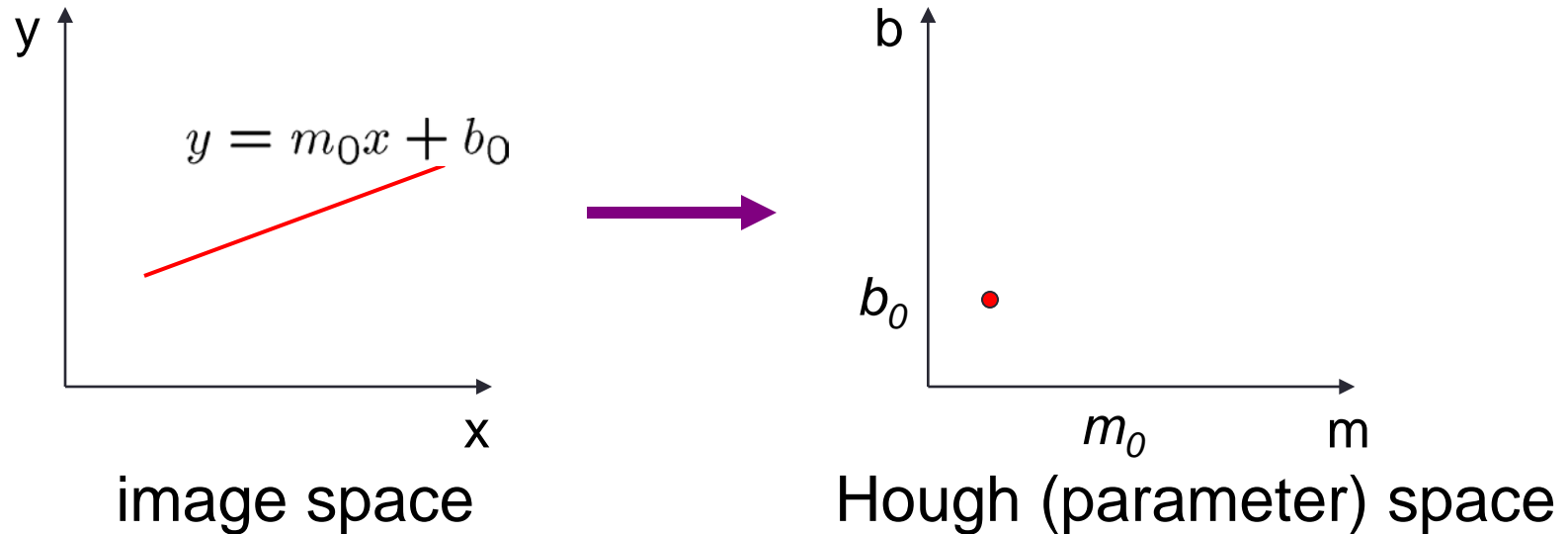


# Fitting lines

- 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.



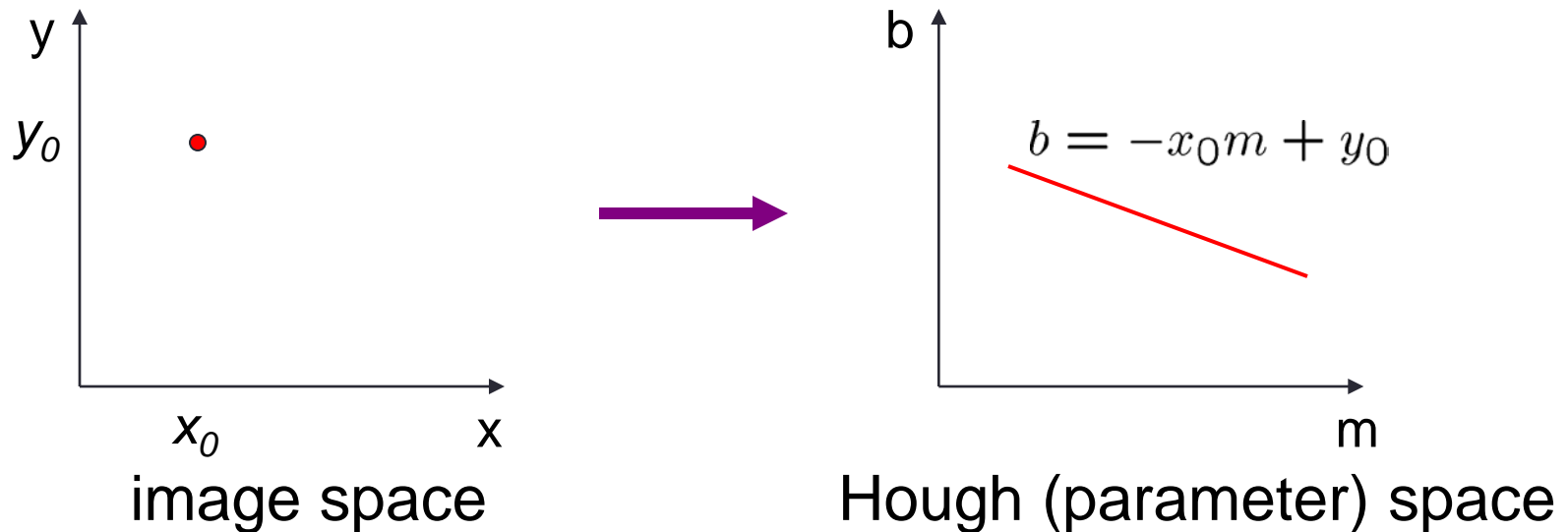
# Finding lines in an image: Hough space



## 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  $y = mx + b$

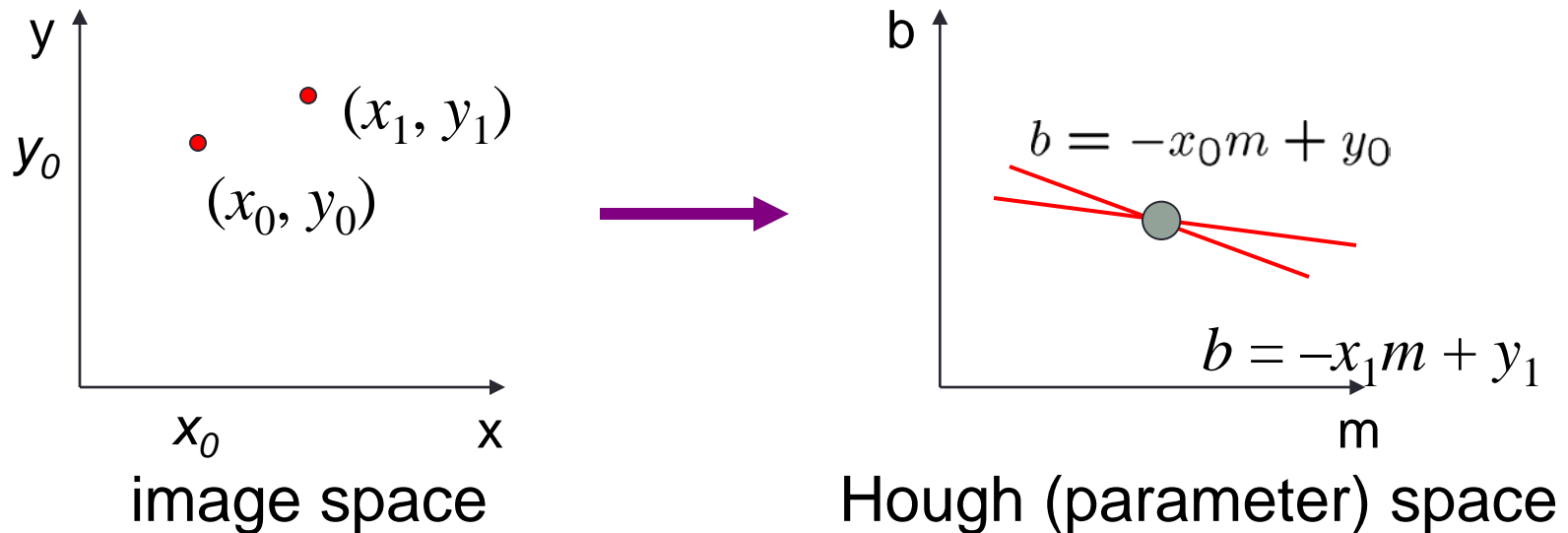
# Finding lines in an image: Hough space



## 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  $b = -x_0m + y_0$
  - this is a line in Hough space

# Finding lines in an image: Hough *transform*



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: Hough algorithm

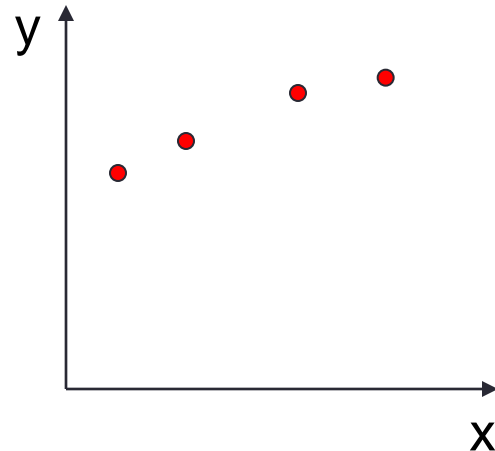
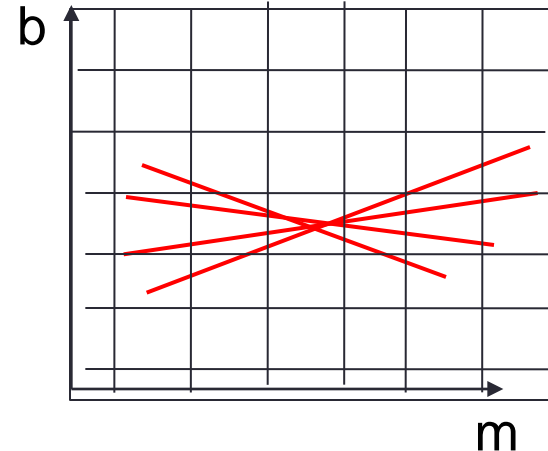


image space

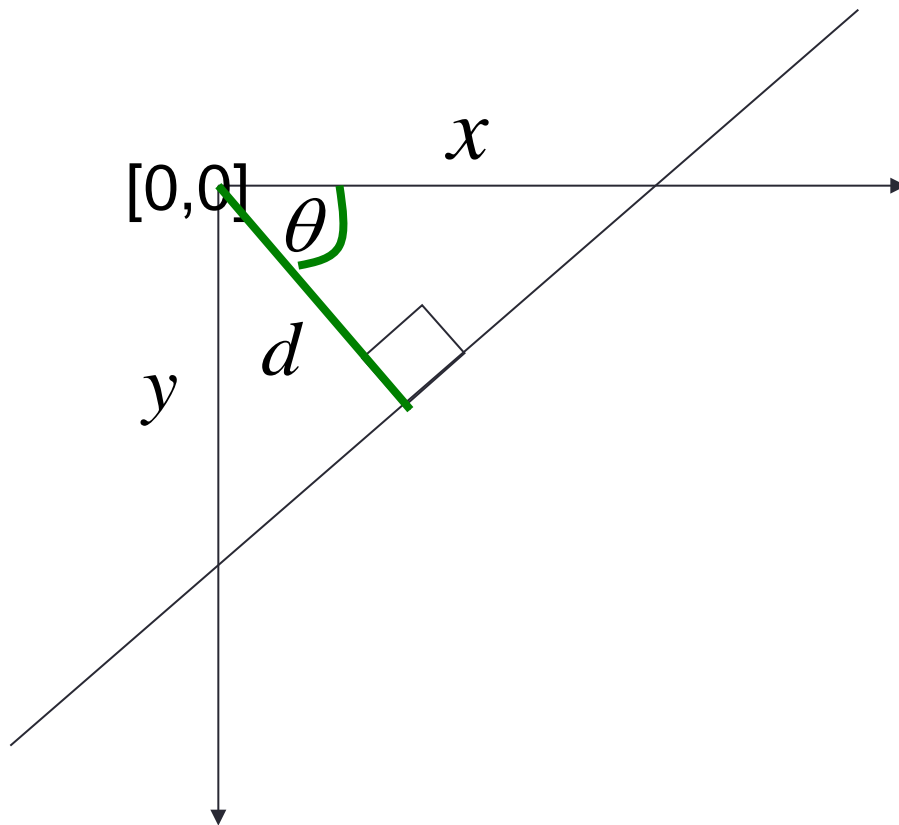


Hough (parameter) space

- 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: can take on infinite values, undefined for vertical lines.



$d$  : perpendicular distance from line to origin

$\theta$  : angle the perpendicular makes with the x-axis

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

Point in image space  $\rightarrow$  sinusoid segment in Hough space

# Hough transform algorithm

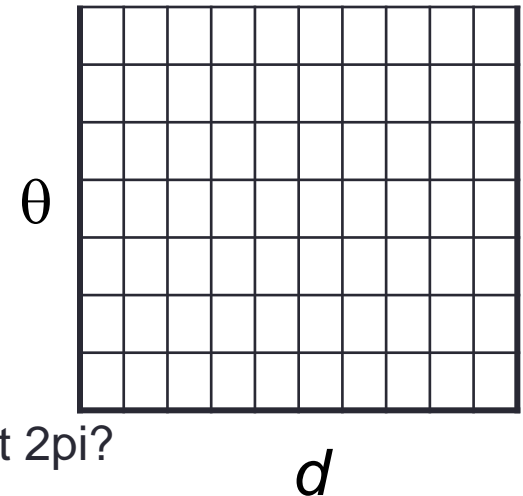
Using the polar parameterization:

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

## Basic Hough transform algorithm

1. Initialize  $H[d, \theta] = 0$
2. for each edge point  $I[x, y]$  in the image
  - for  $\theta = 0$  to  $180$  // some quantization; not  $2\pi$ ?
  - $d = x \cos \theta - y \sin \theta$  // maybe negative
  - $H[d, \theta] += 1$
3. Find the value(s) of  $(d, \theta)$  where  $H[d, \theta]$  is maximum
4. The detected line in the image is given by  $d = x \cos \theta - y \sin \theta$

H: accumulator array (votes)

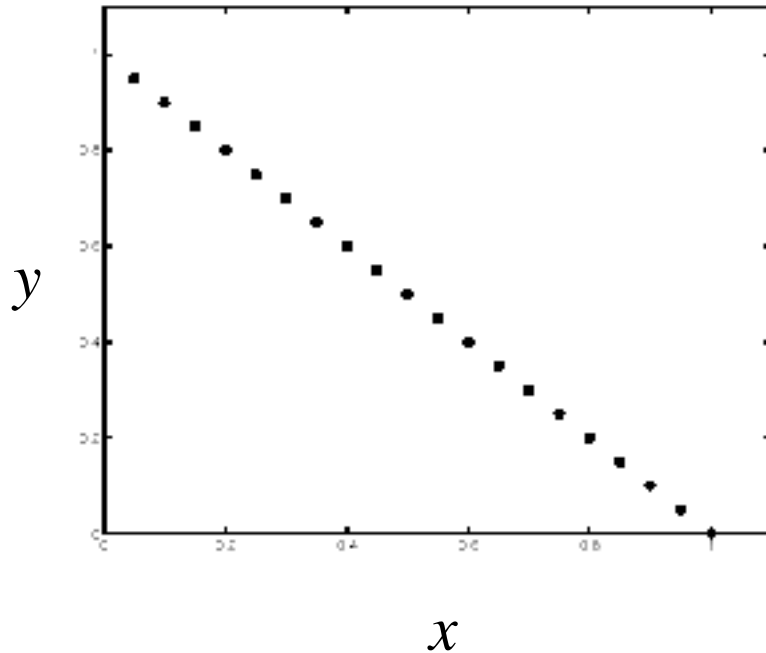


Space complexity?  $k^n$  (n dimensions, k bins each)

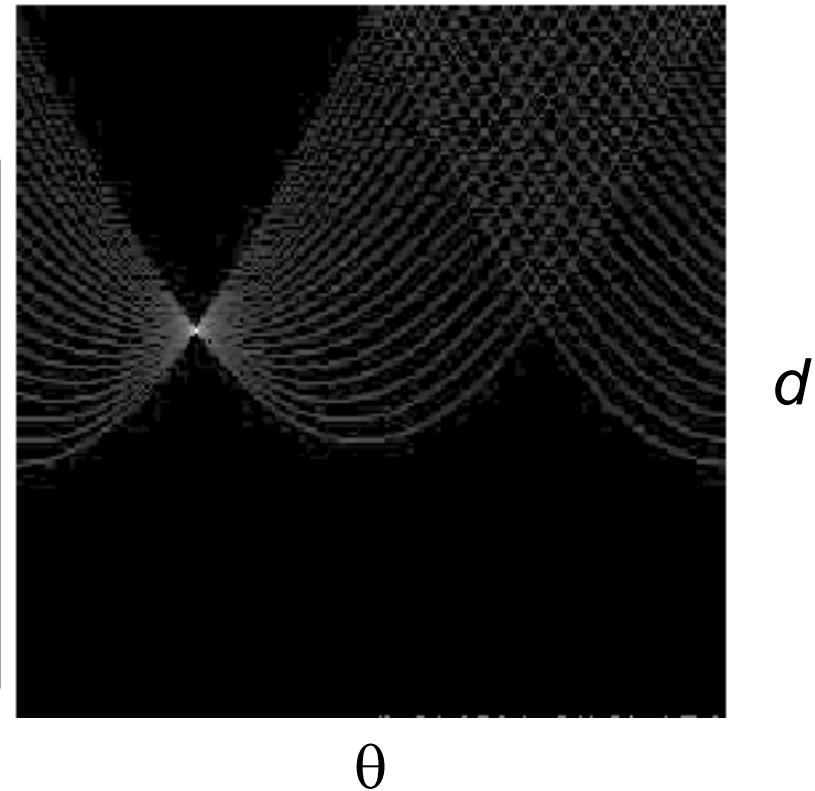
Time complexity (in terms of number of voting elements)?



# Example: Hough transform for straight lines



**Image space  
edge coordinates**

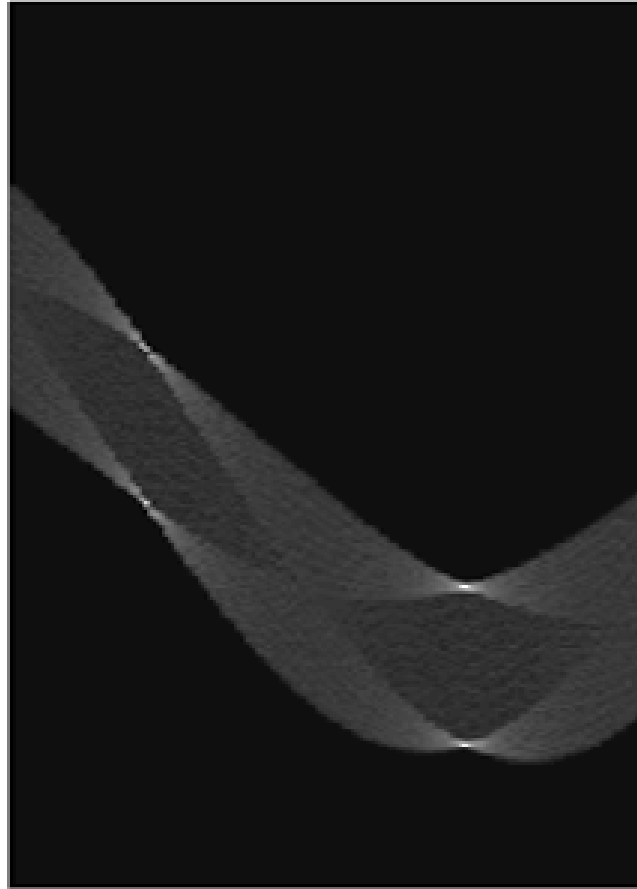


**Votes**

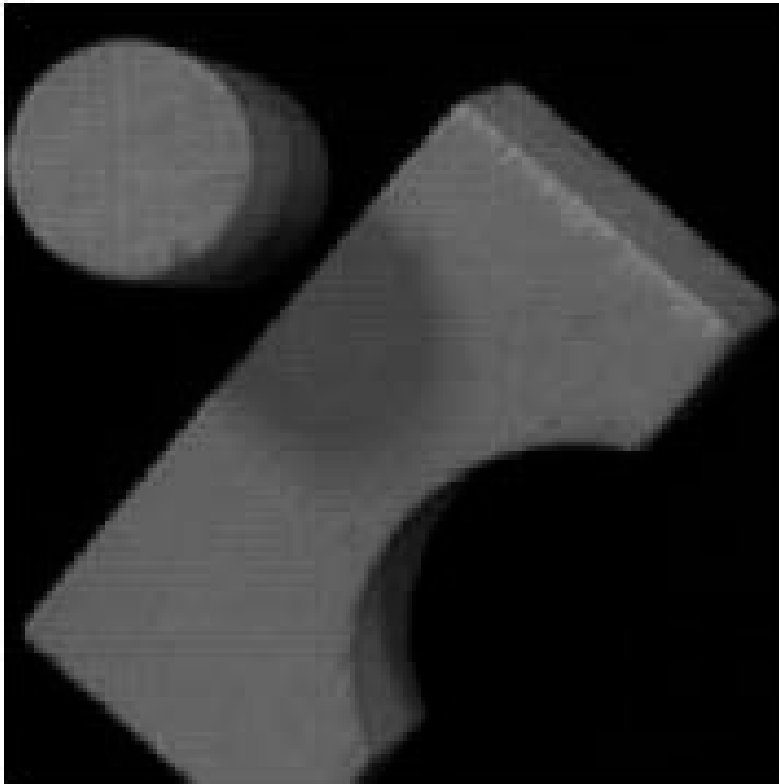
Bright value = high vote count  
Black = no votes

# Example: Hough transform for straight lines

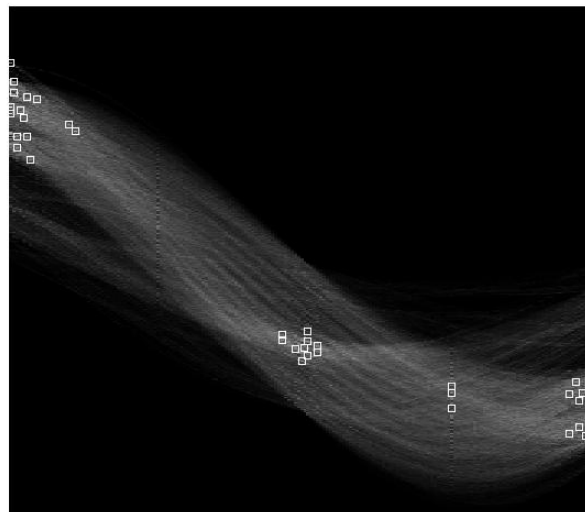
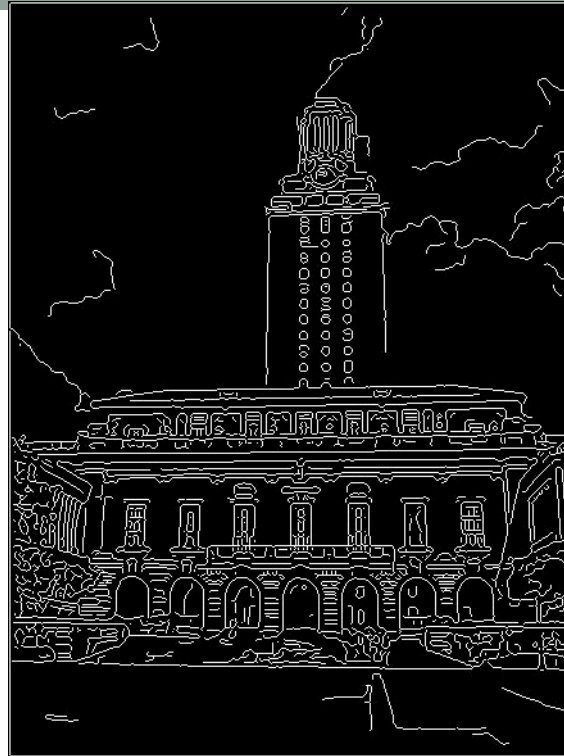
Square :

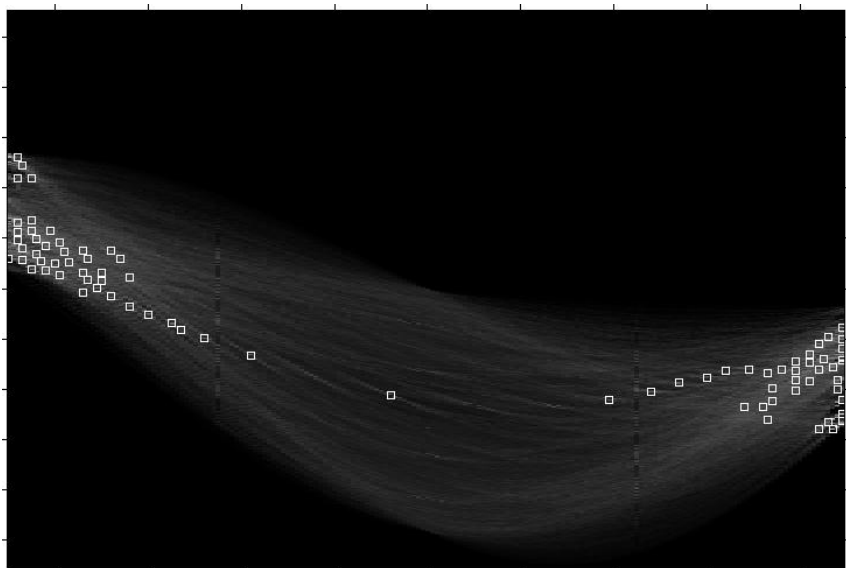
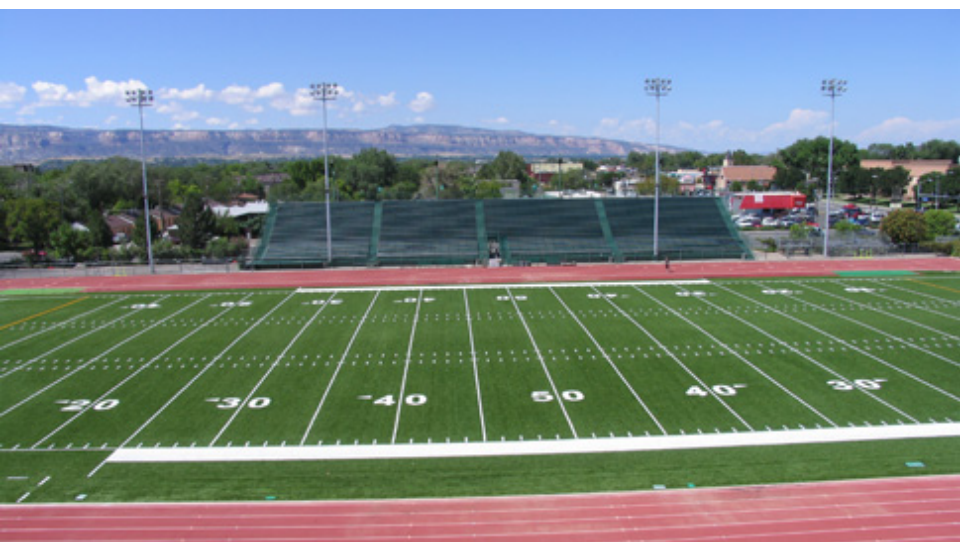


# Example: Hough transform for straight lines



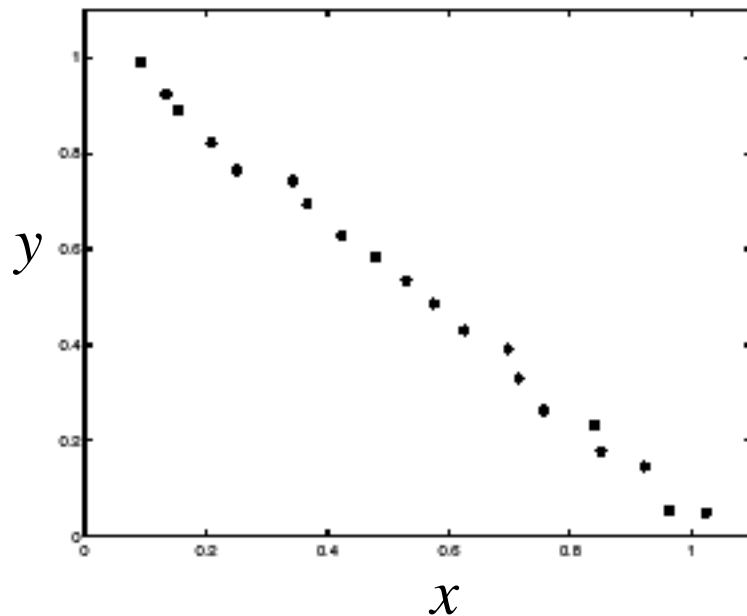
# Hough demo..



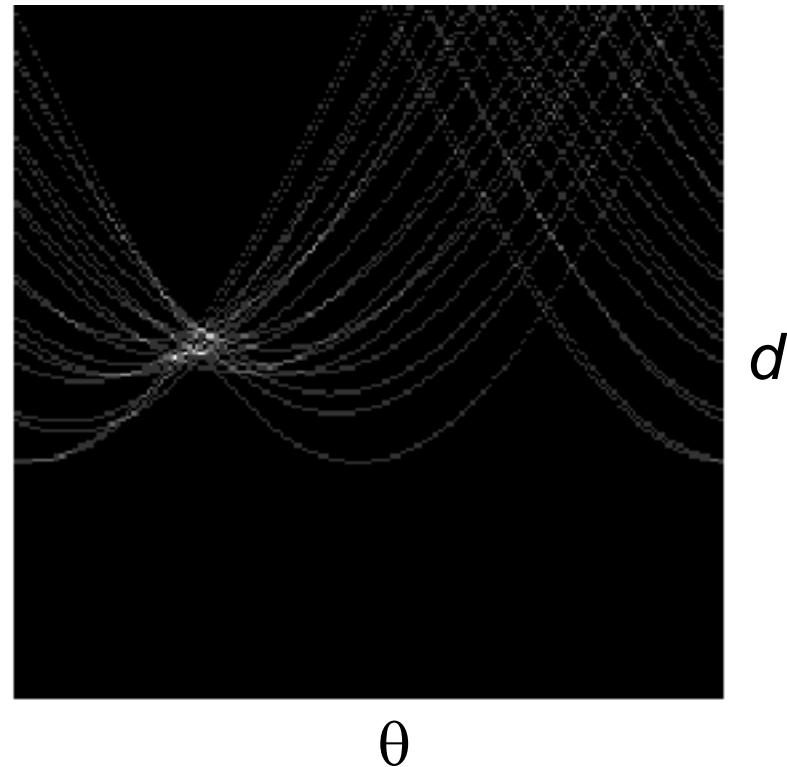


Showing longest segments found

# Impact of noise on Hough



**Image space  
edge coordinates**

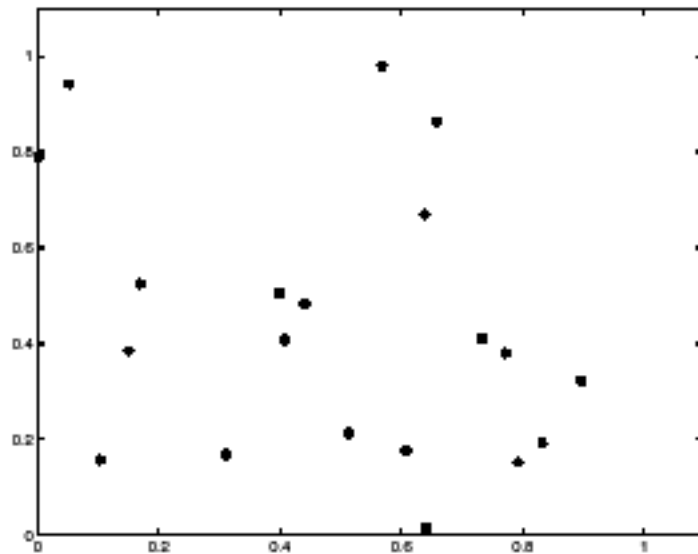


**Votes**

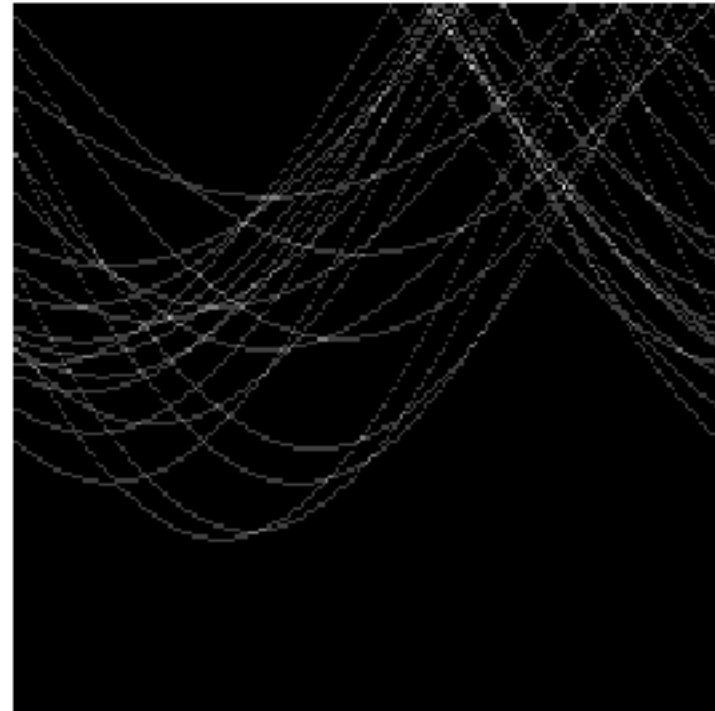
What difficulty does this present for an implementation?



# Impact of noise on Hough



**Image space  
edge coordinates**



**Votes**

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

# Extensions

- **Extension 1:** Use the image gradient

- same
- for each edge point  $I[x,y]$  in the image
  - $\theta$  = gradient at  $(x,y)$

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

- $H[d, \theta] += 1$
  - same
  - same
- (Reduces degrees of freedom)

- Extension 2

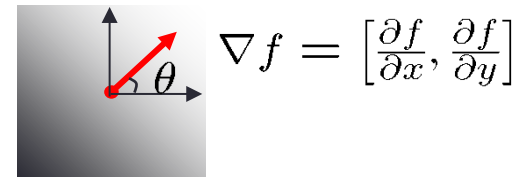
- give more votes for stronger edges

- Extension 3

- change the sampling of  $(d, \theta)$  to give more/less resolution

- Extension 4

- The same procedure can be used with circles, squares, or any other shape



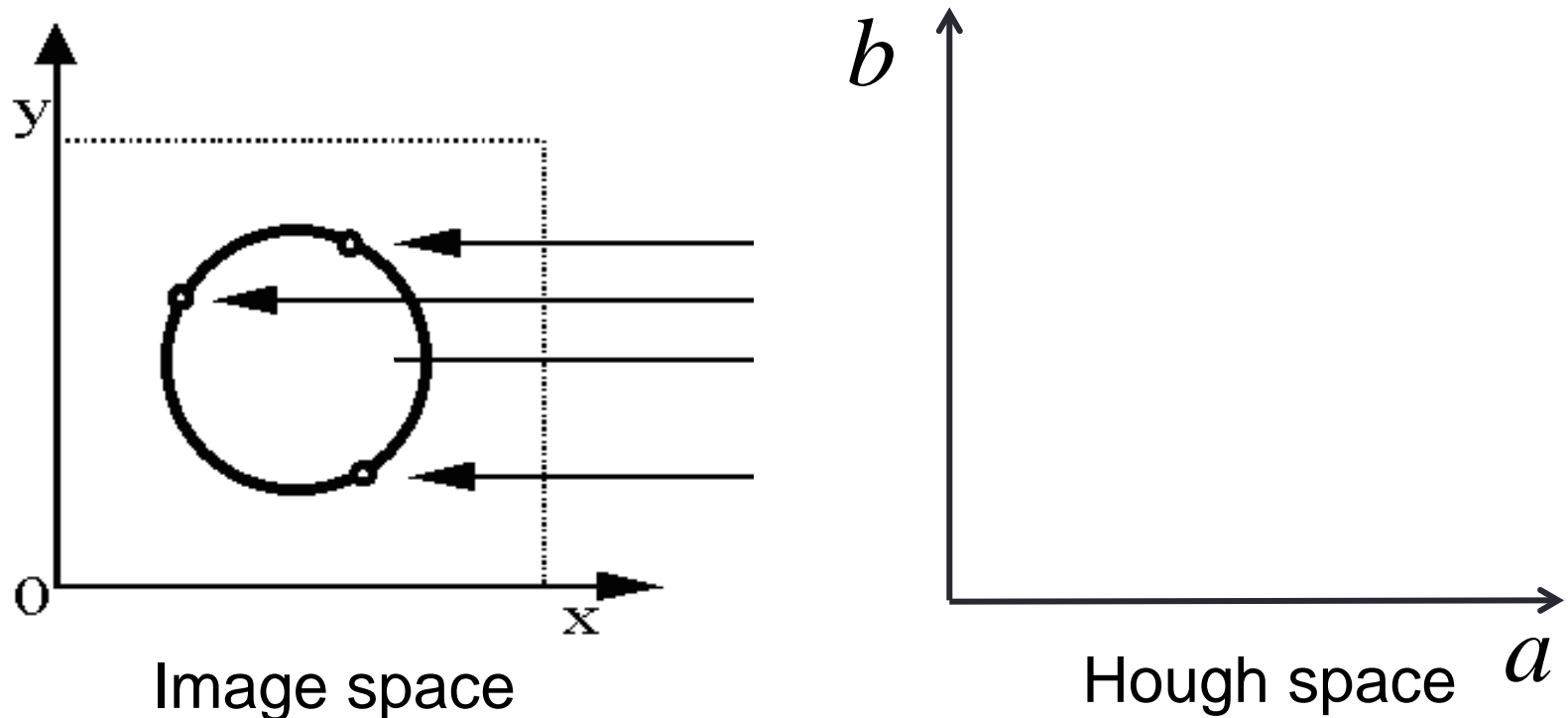
$$\theta = \tan^{-1} \left( \frac{\partial f / \partial y}{\partial f / \partial x} \right)$$

# 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

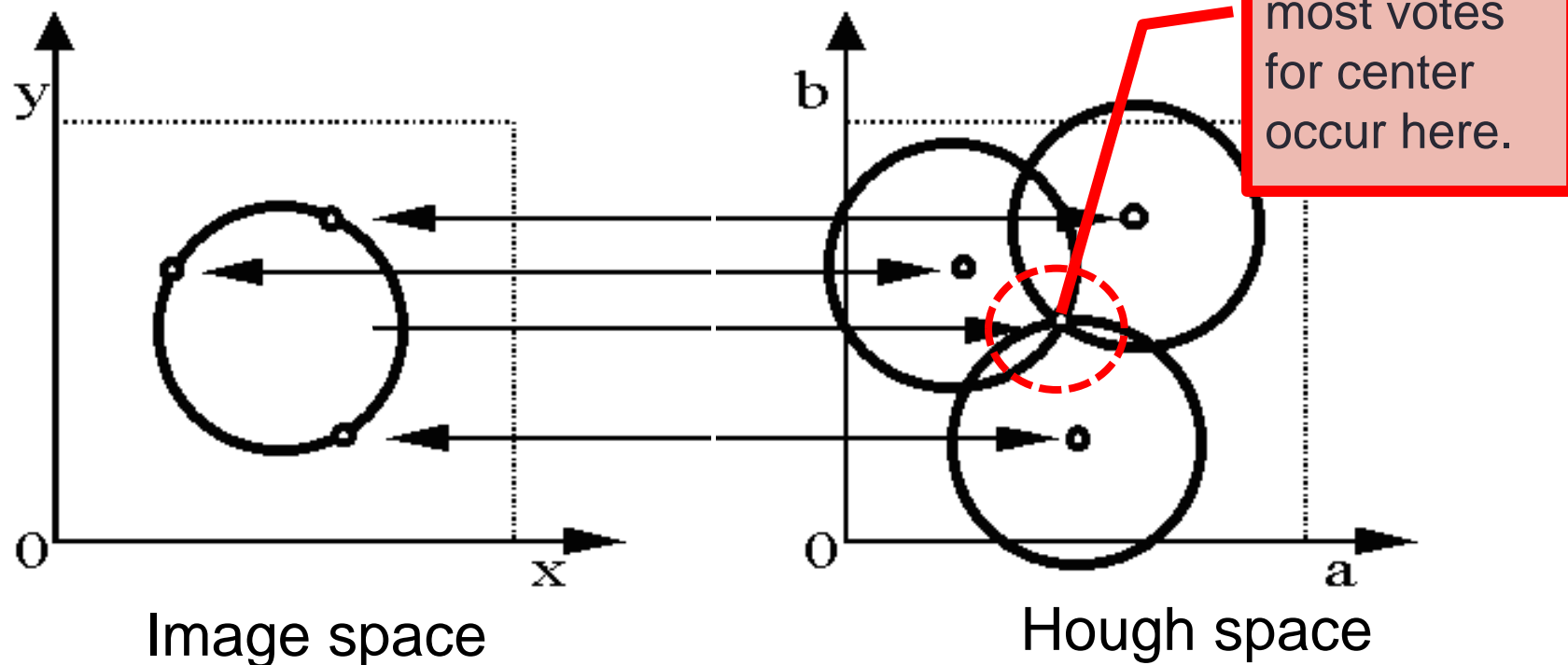


# Hough transform for circles

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

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

- For a fixed radius  $r$ , unknown gradient direction

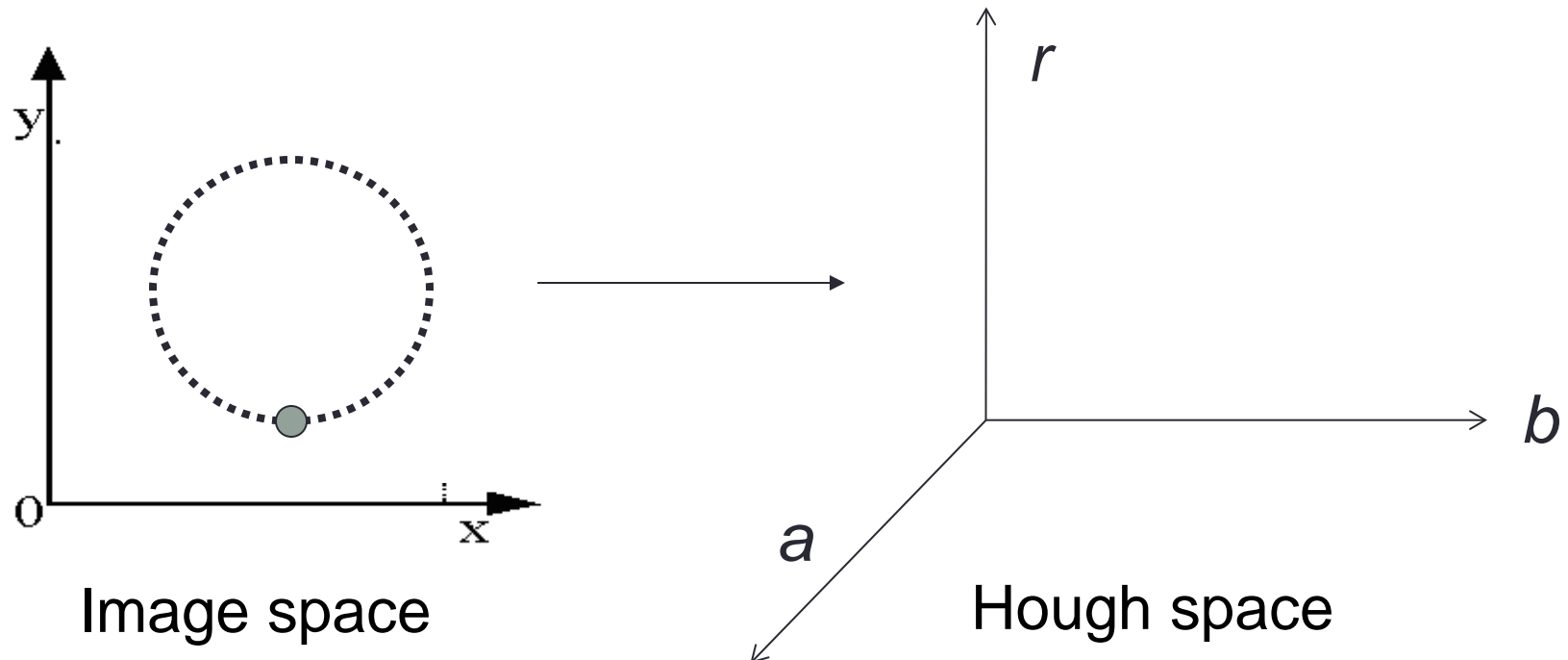


# Hough transform for circles

- 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



# 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

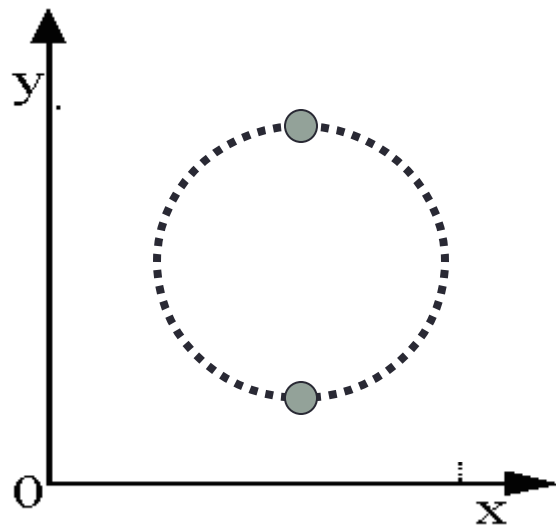
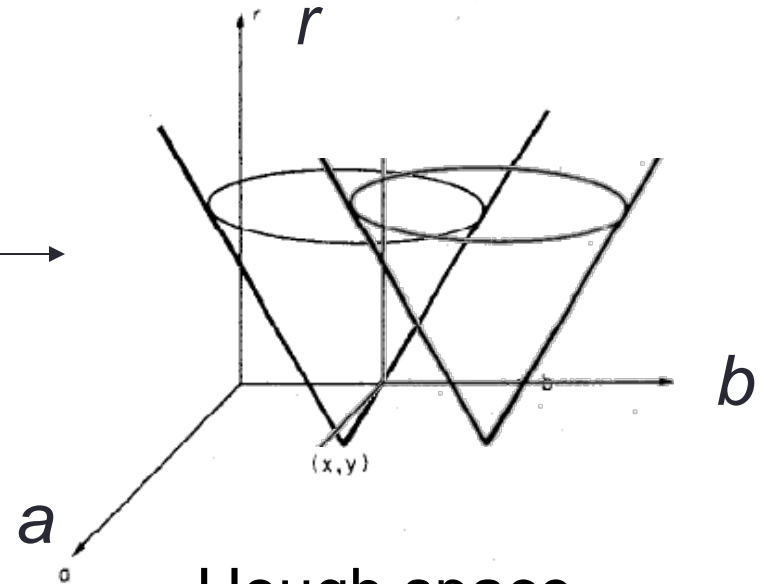


Image space



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

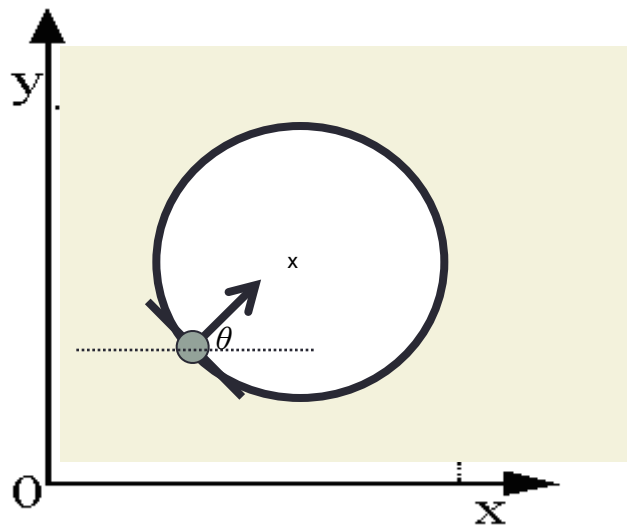
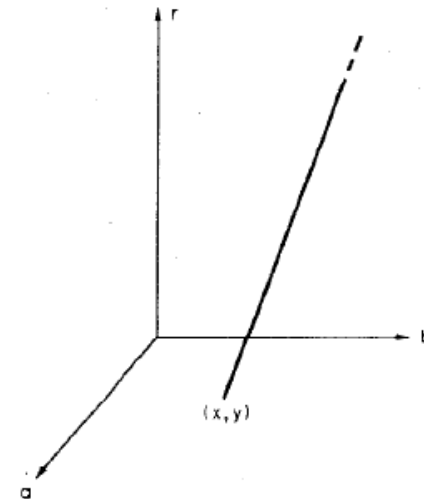


Image space



Hough space



# Hough transform for circles

For every edge pixel  $(x,y)$  :

For each possible radius value  $r$ .

For each possible gradient direction  $\theta$ :  
*%% or use estimated gradient*

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

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

$$H[a,b,r] += 1$$

end

end

# Example: detecting circles with Hough



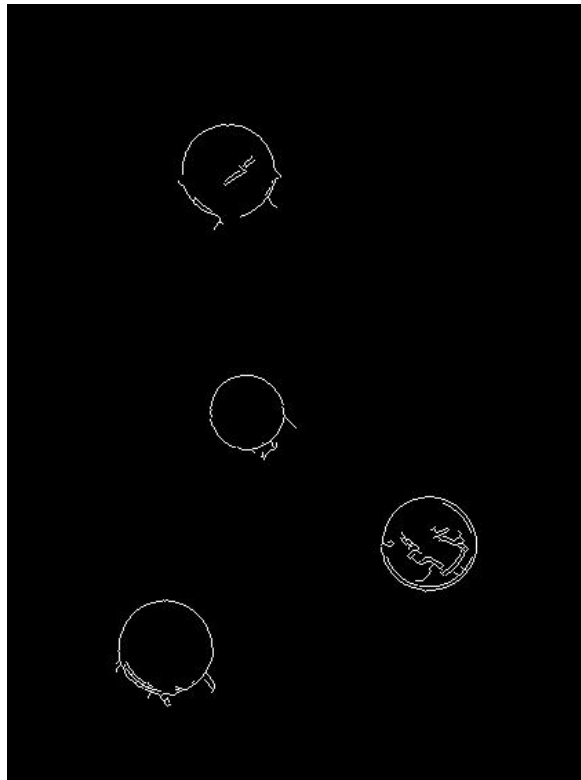
Crosshair indicates results of Hough transform, bounding box found via motion differencing.

# Example: detecting circles with Hough

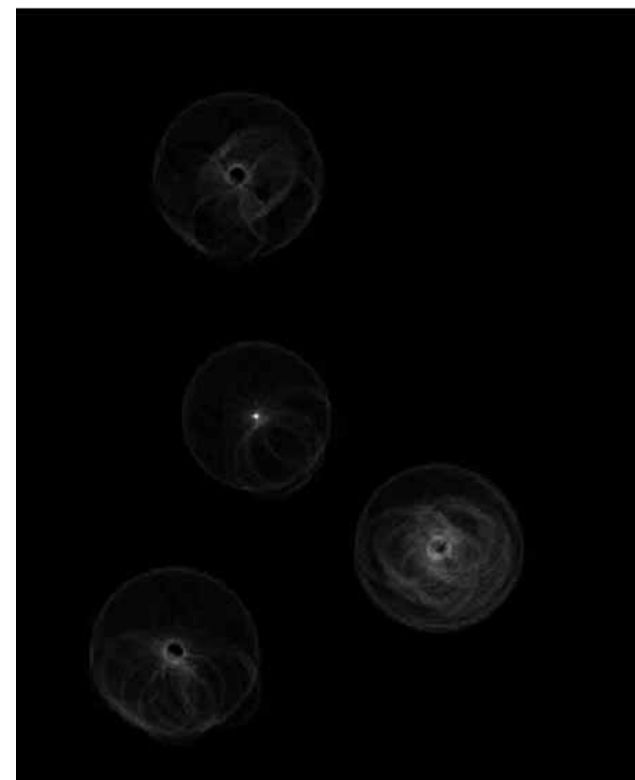
Original



Edges



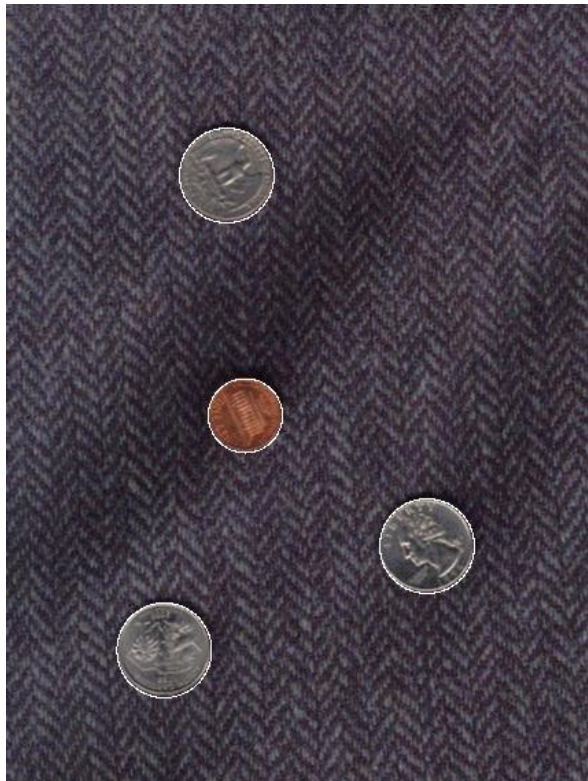
Votes: Penny



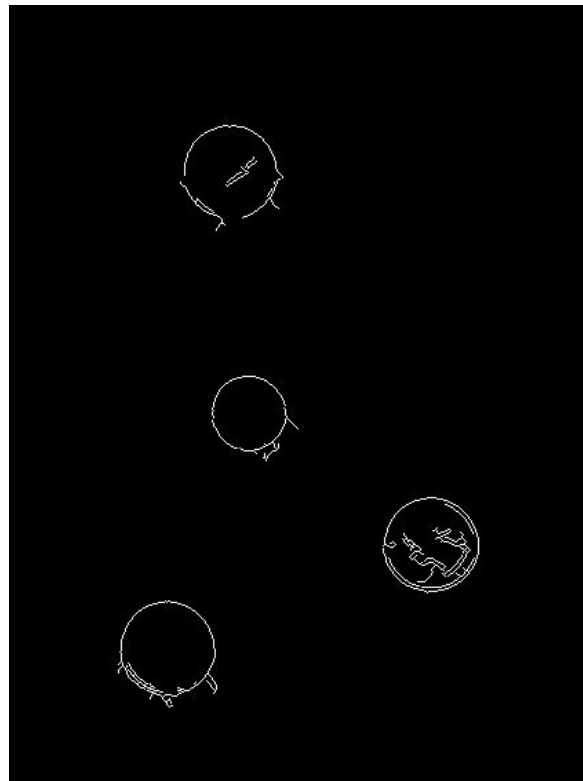
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

# Example: detecting circles with Hough

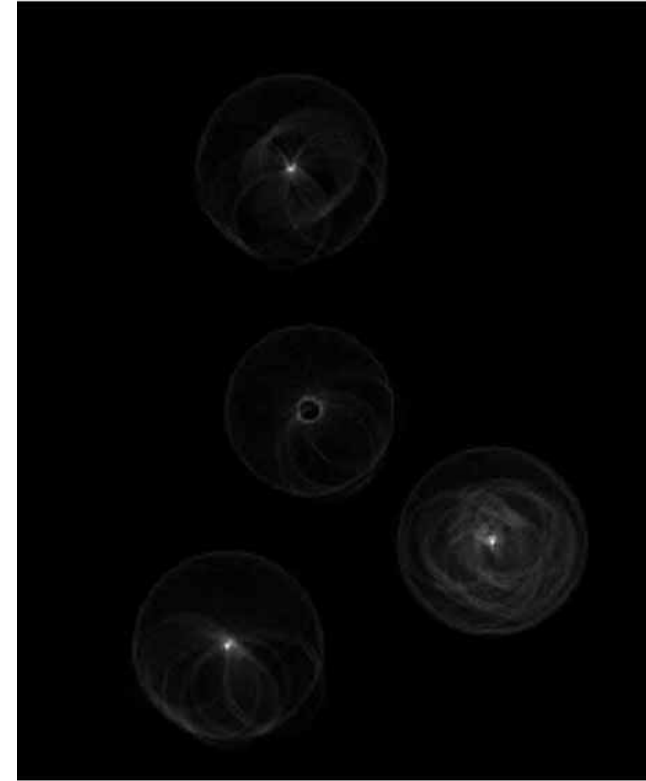
Original



Edges



Votes: Quarter



# 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

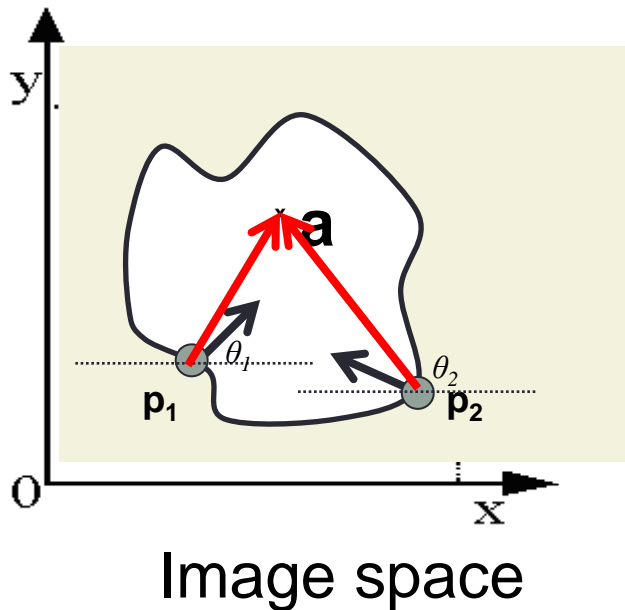
- All points are processed independently, so can cope with occlusion
- Some robustness to noise: noise points unlikely to contribute consistently to any single bin
- Can detect multiple instances of a model in a single pass

- 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

# Generalized Hough transform

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



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  $\theta$ .

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

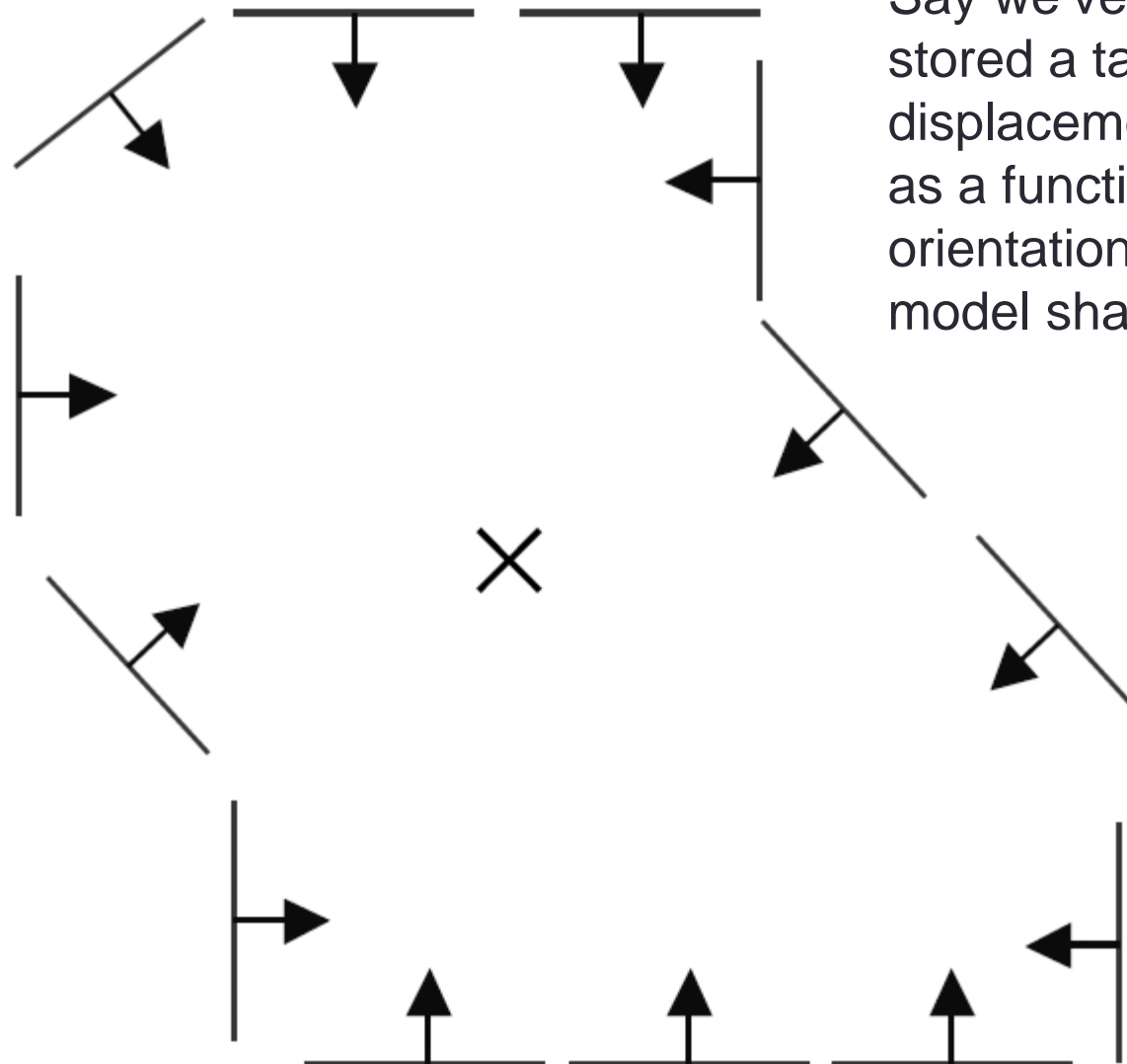


# Generalized Hough transform

- To detect the model shape in a new image:
- For each edge point
  - For each possible  $\theta^*$  master orientation
    - Index into table with its gradient orientation  $\theta - \theta^*$
    - Use retrieved  $r$  vectors to vote for position of reference point
- Peak in this Hough space is reference point with most supporting edges

*(In next slides we'll assume only translation is unknown, i.e., orientation and scale are fixed)*

# Example

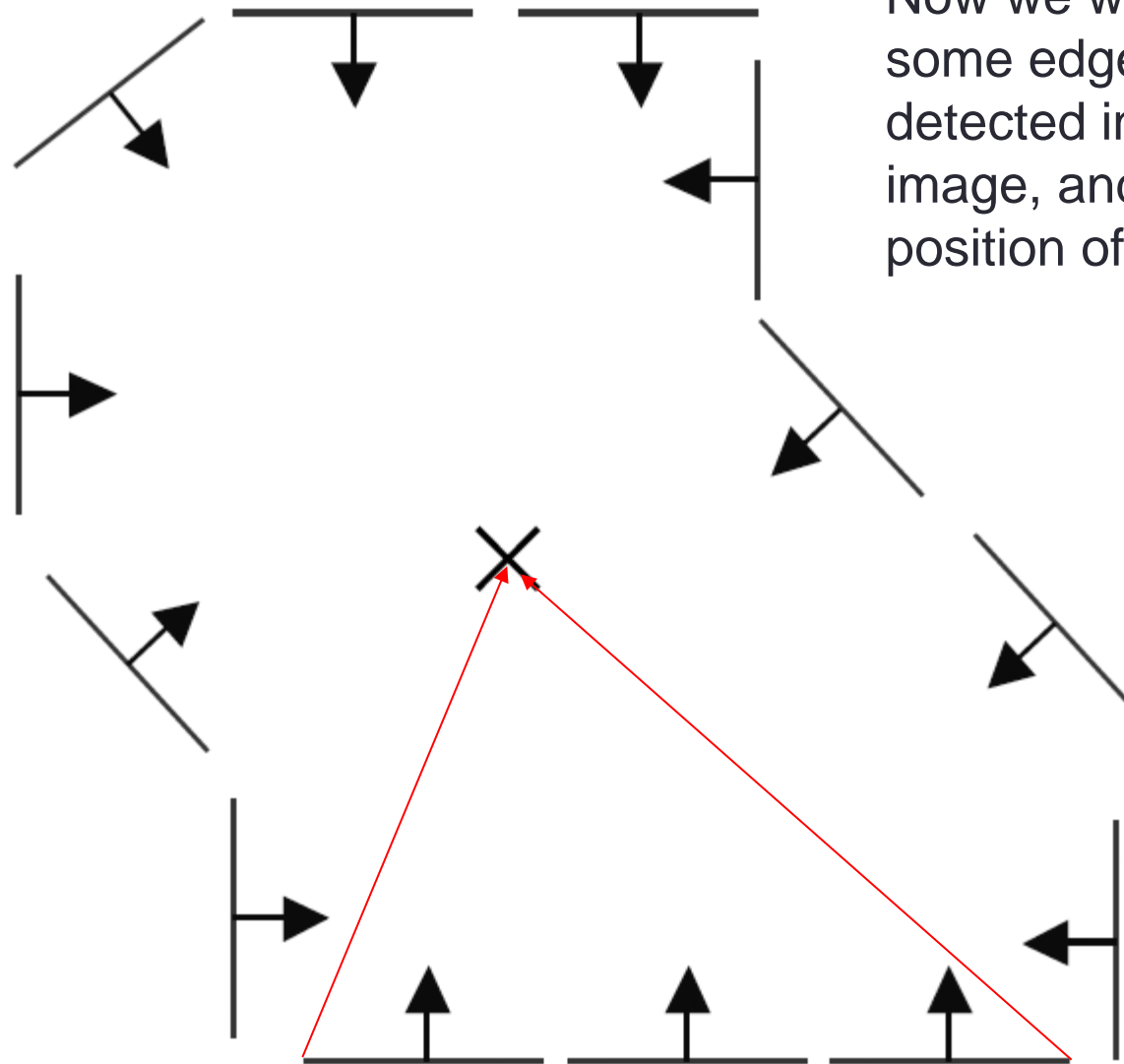


Say we've already stored a table of displacement vectors as a function of edge orientation for this model shape.

model shape

Source: L. Lazebnik

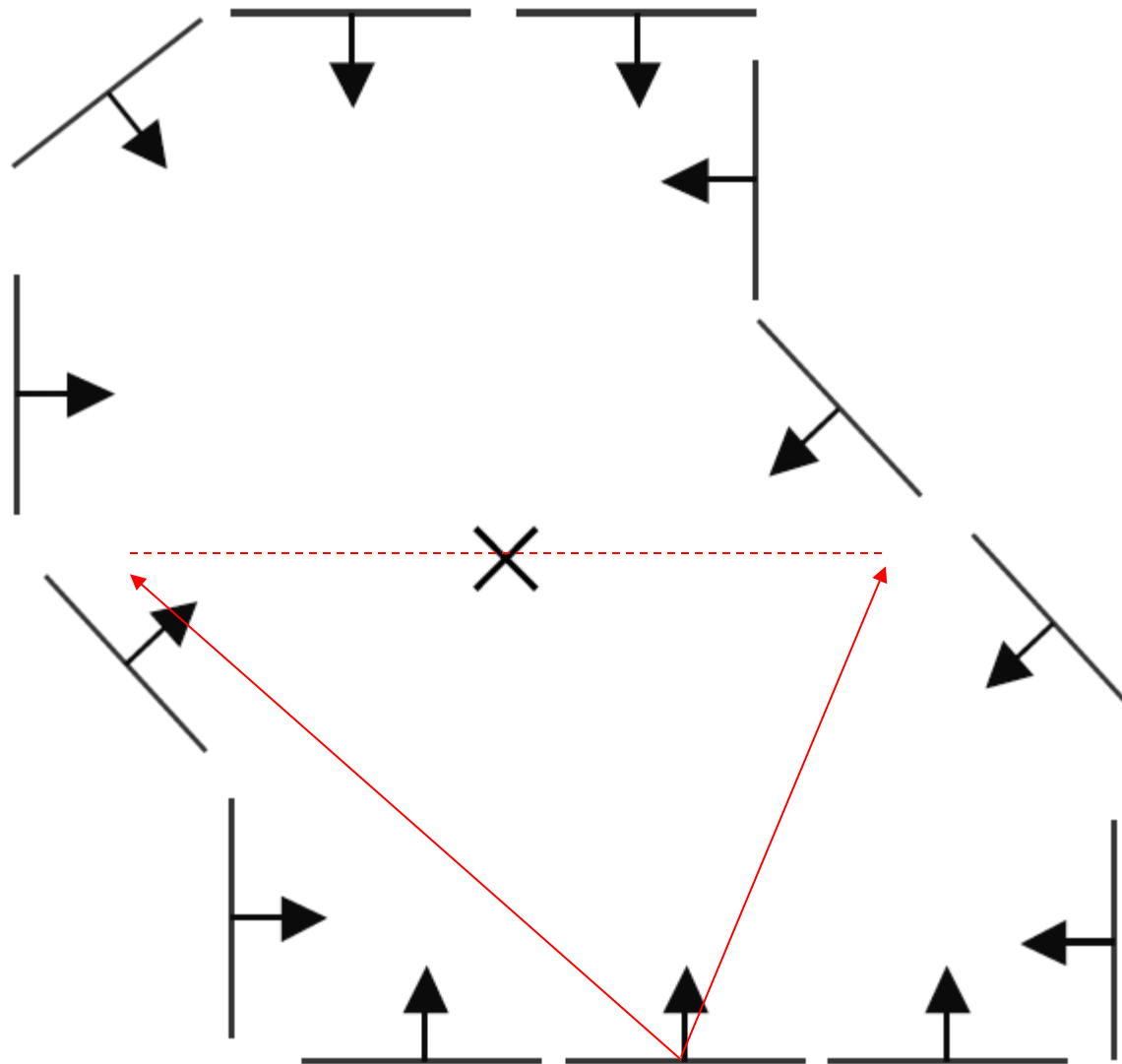
# Example



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

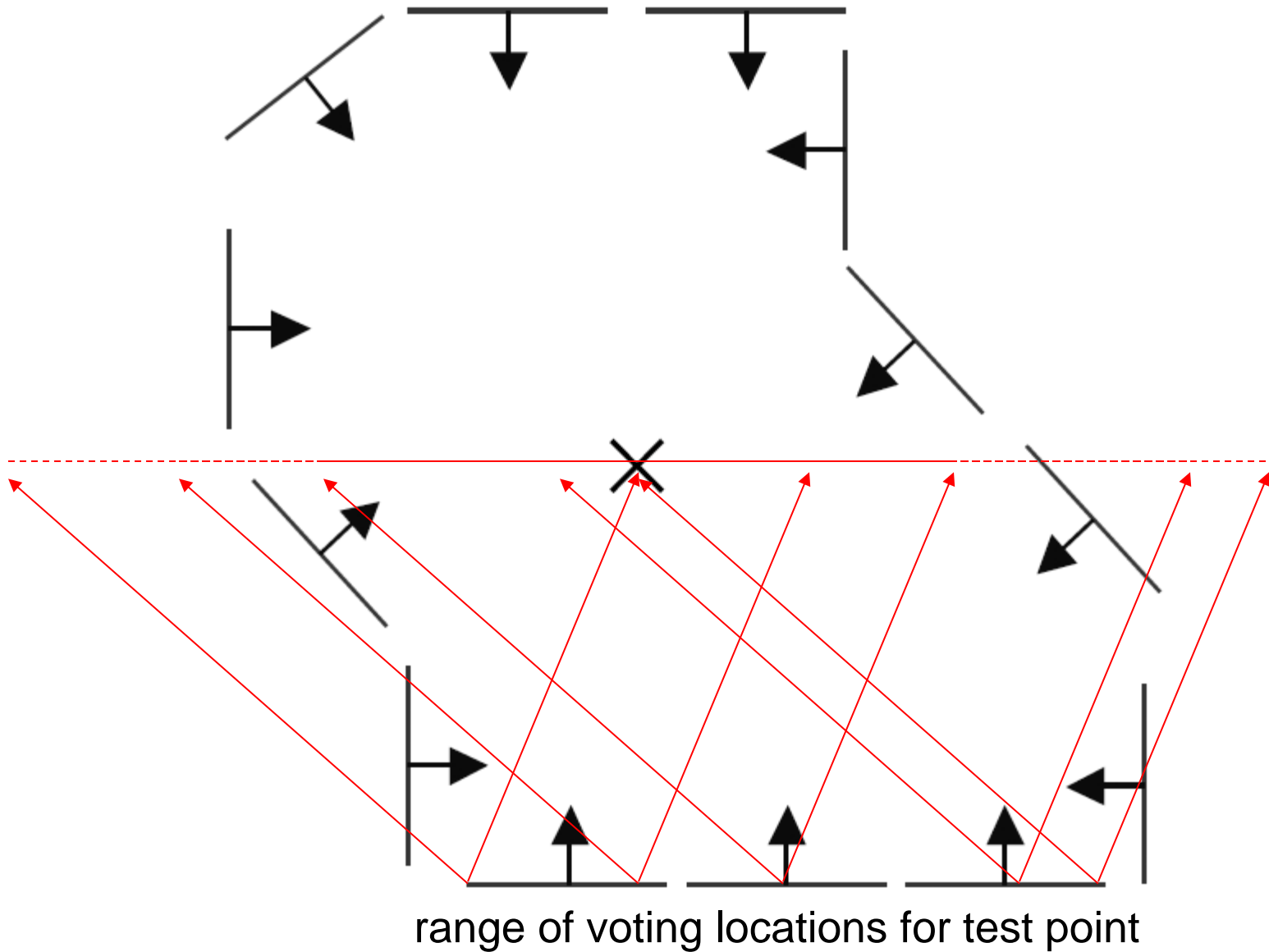
displacement vectors for model points

# Example

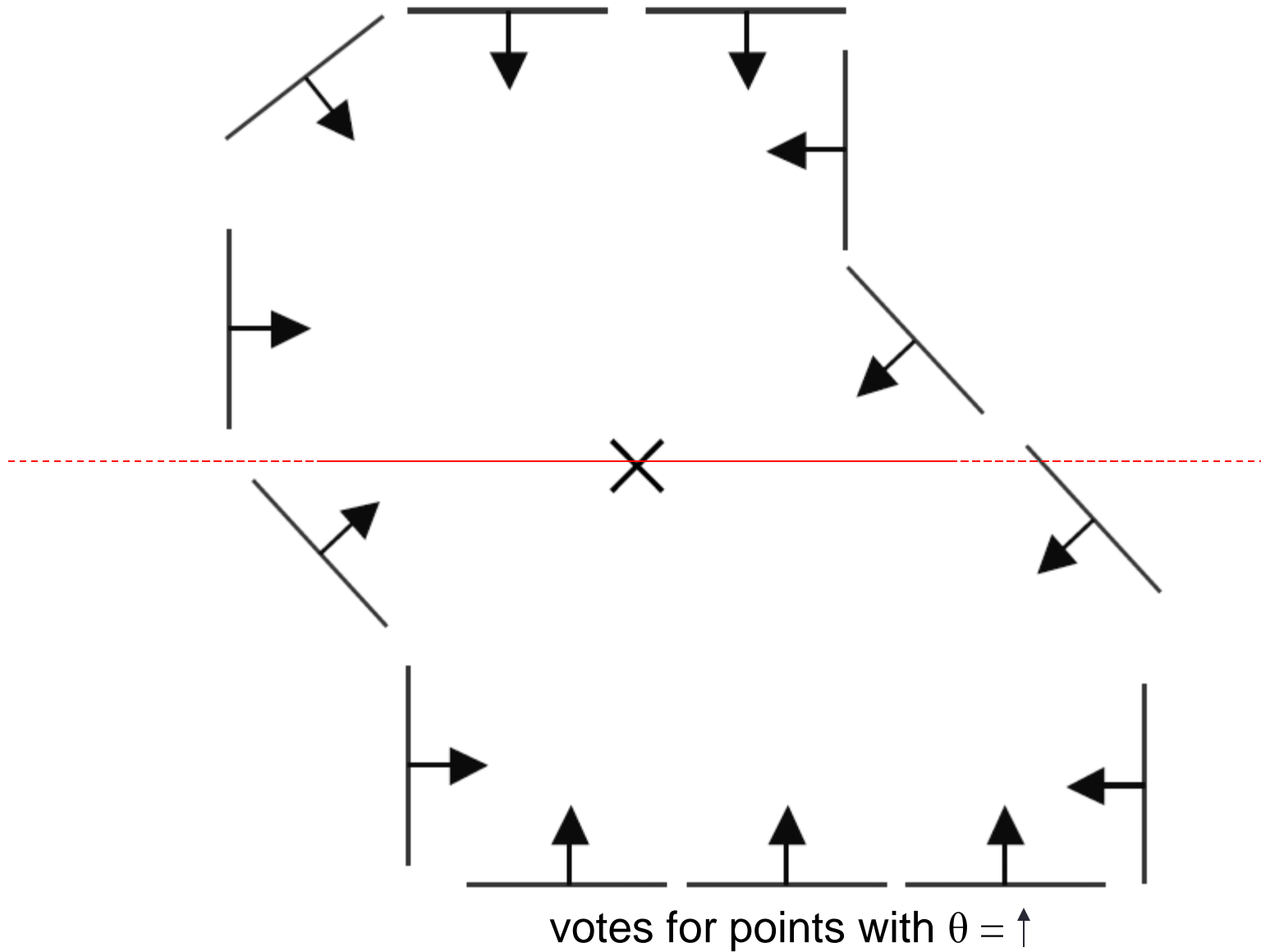


range of voting locations for test point

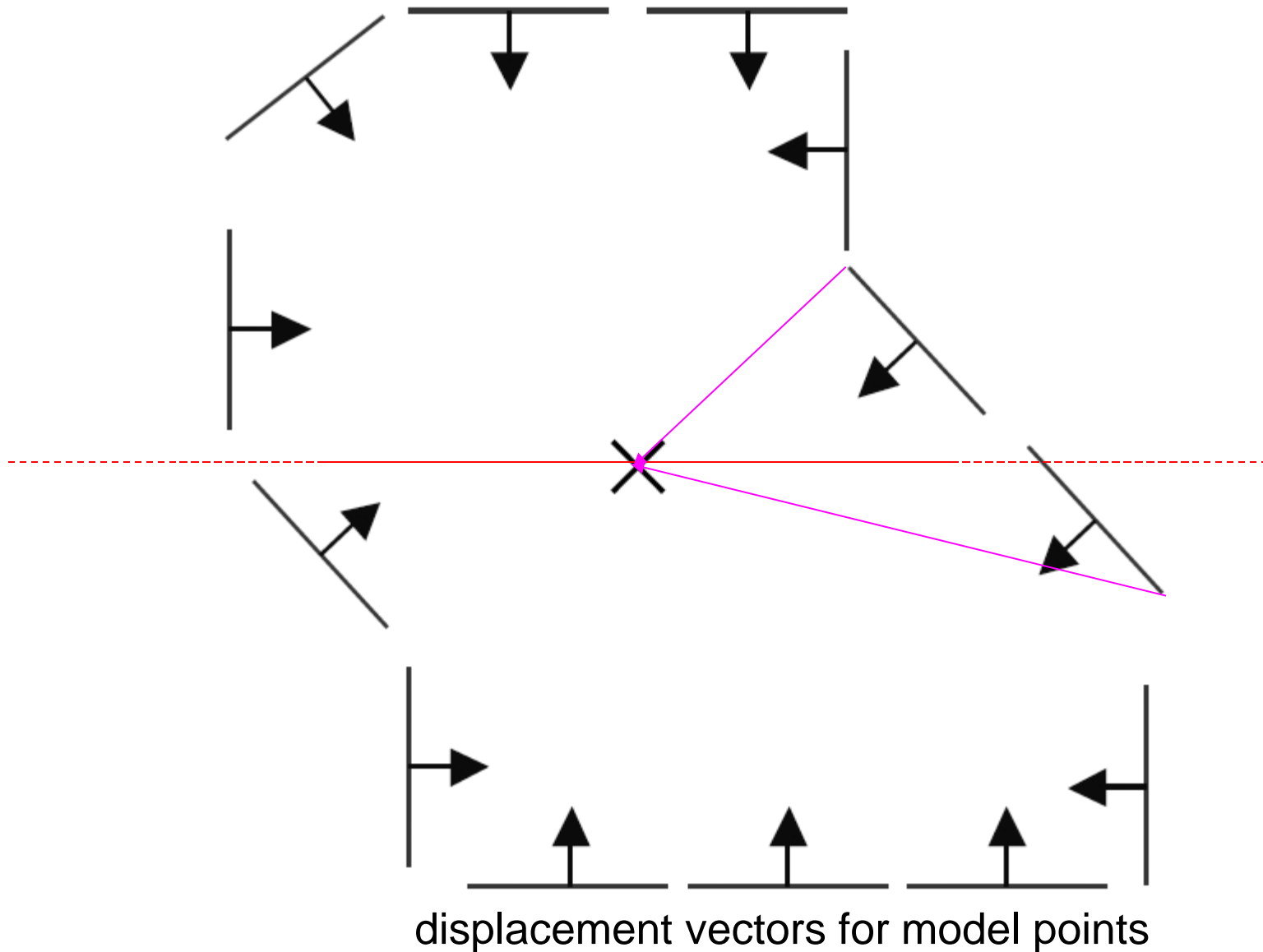
# Example



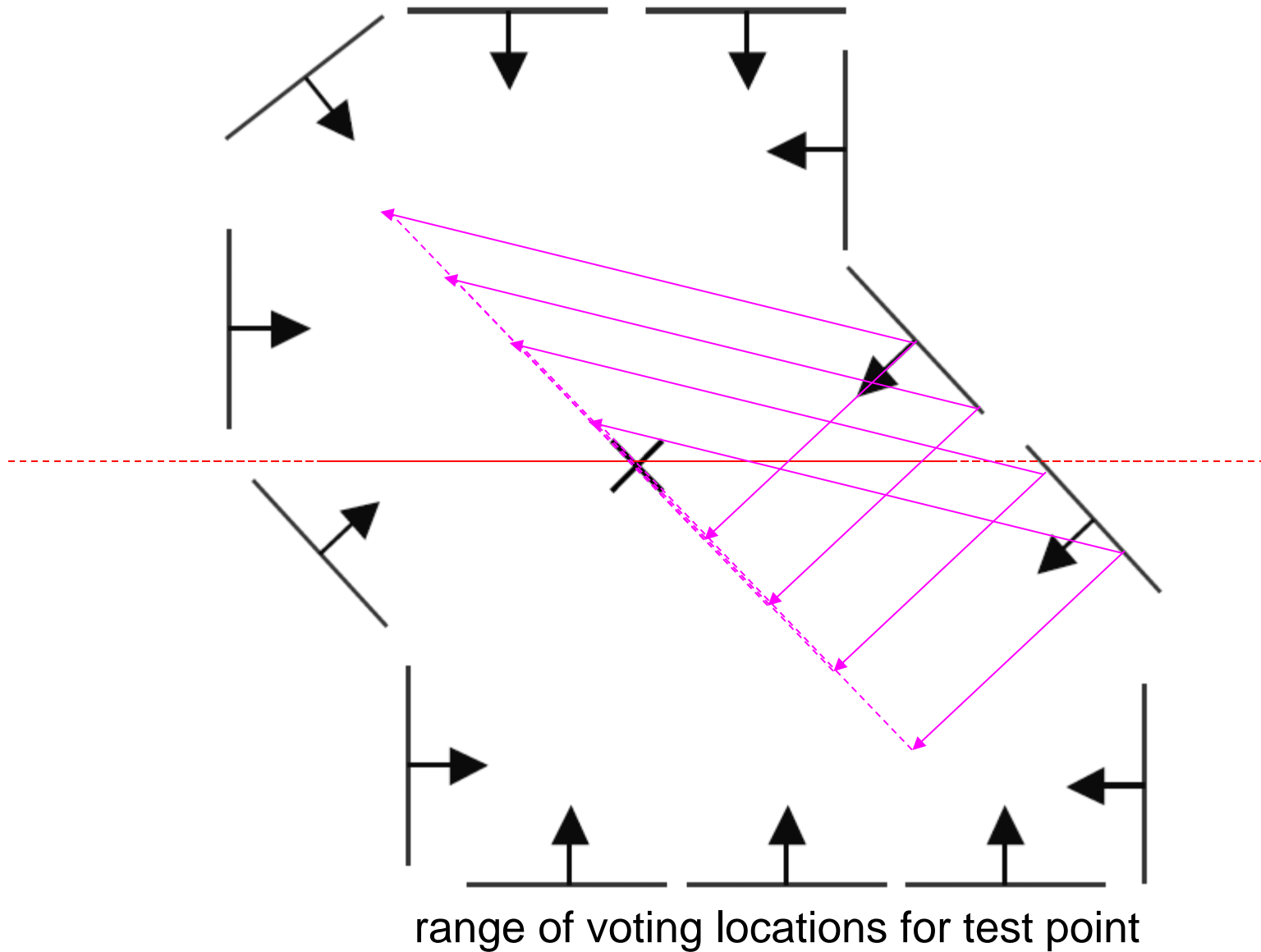
# Example



# Example

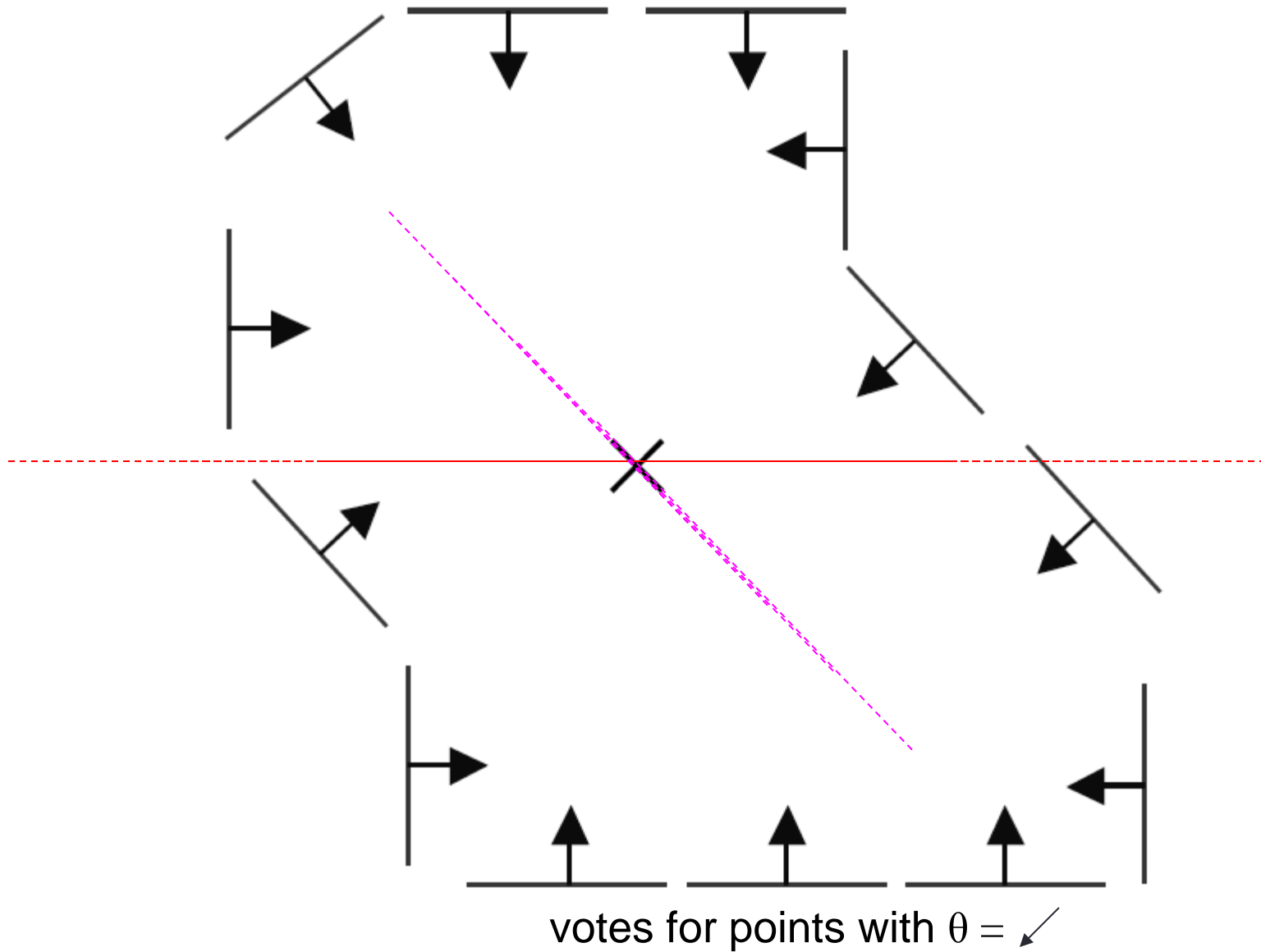


# Example



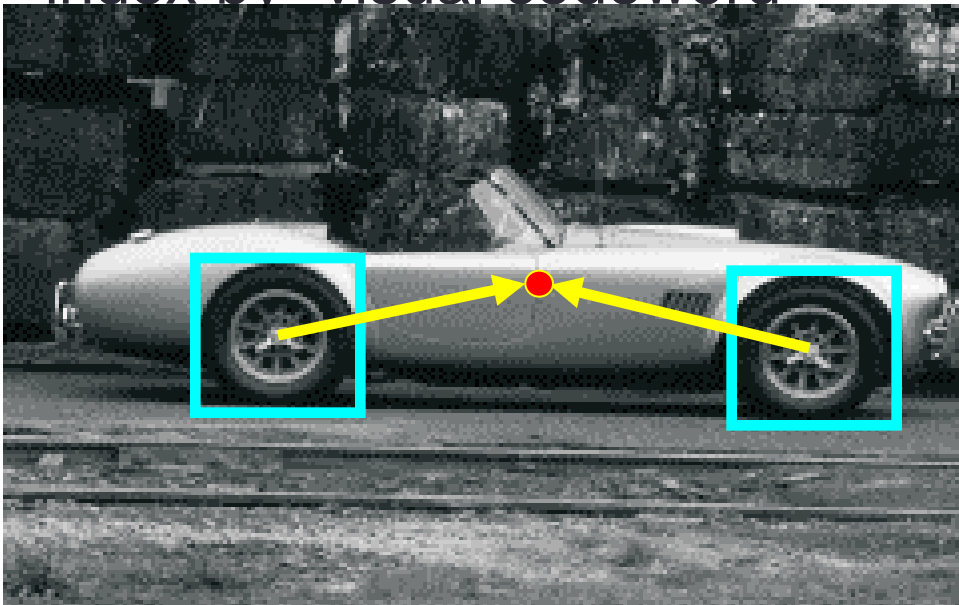


# Example

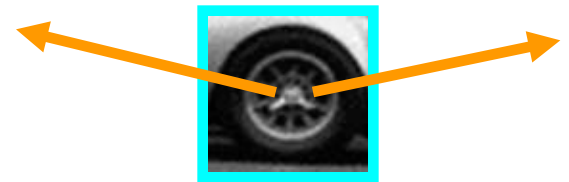


# Application in recognition

- 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, [Combined Object Categorization and Segmentation with an Implicit Shape Model](#), ECCV Workshop on Statistical Learning in Computer Vision 2004

# Application in recognition

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



test image

B. Leibe, A. Leonardis, and B. Schiele, [Combined Object Categorization and Segmentation with an Implicit Shape Model](#), ECCV Workshop on Statistical Learning in Computer Vision 2004

# Summary

- Fitting problems require finding any supporting evidence for a model, even within clutter and missing features.
  - associate features with an explicit model
- Voting approaches, such as the Hough transform, make it possible to find likely model parameters without searching all combinations of features.
  - Hough transform approach for lines, circles, ..., arbitrary shapes defined by a set of boundary points, recognition from patches.
- It's Labor Day Weekend – enjoy!