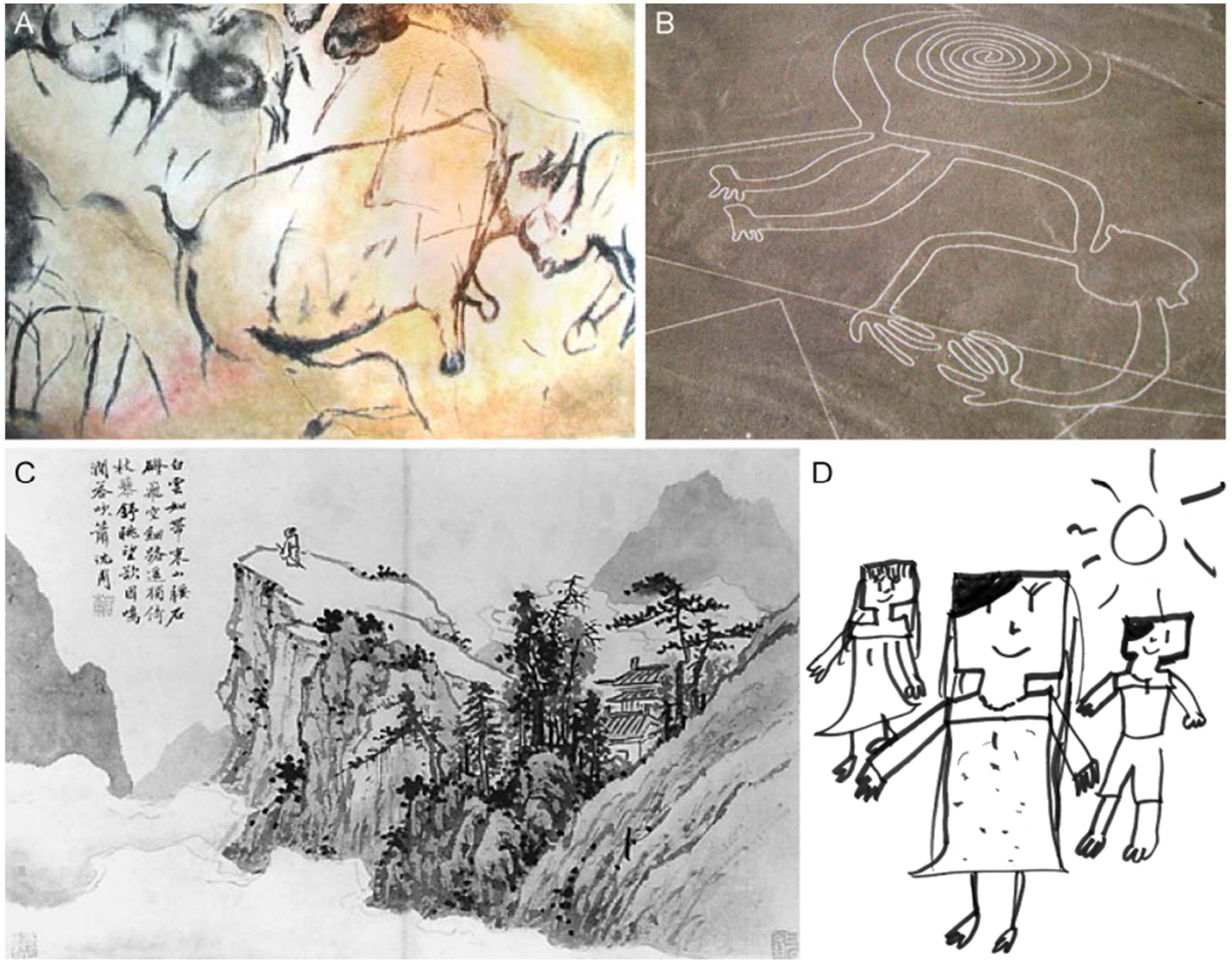
Lecture 4: Finding lines: from detection to model fitting

Professor Fei-Fei Li

Stanford Vision Lab

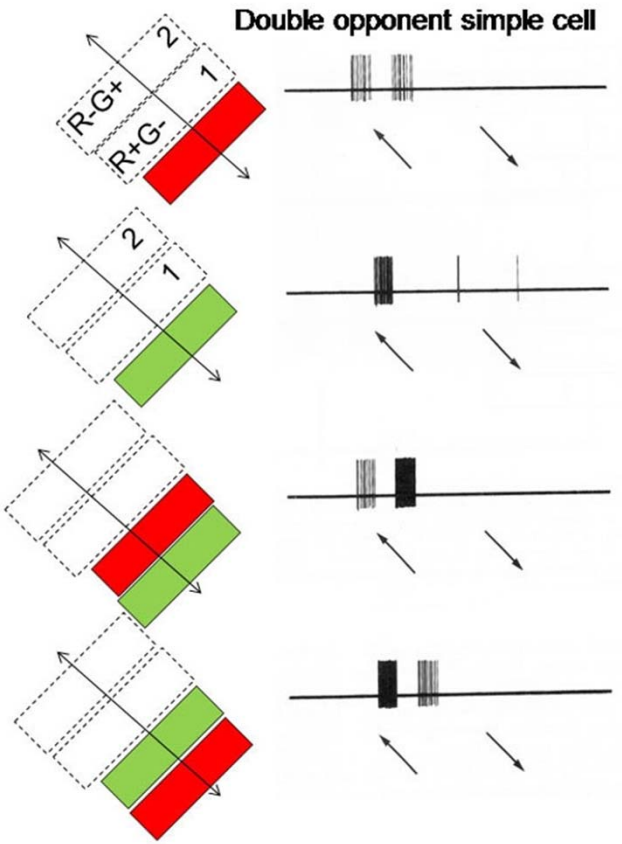
What we will learn today

- Edge detection
  - Canny edge detector
- Line fitting
  - Hough Transform
  - RANSAC (Problem Set 2 (Q5))



(A) Cave painting at Chauvet, France, about 30,000 B.C.;
(B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoplyphs, Peru, about 700 – 200 B.C.;
(C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
(D) Line drawing by 7-year old I. Lleras (2010 A.D.).

We know edges are special from human (mammalian) vision studies



Hubel & Wiesel, 1960s

We know edges are special from human (mammalian) vision studies

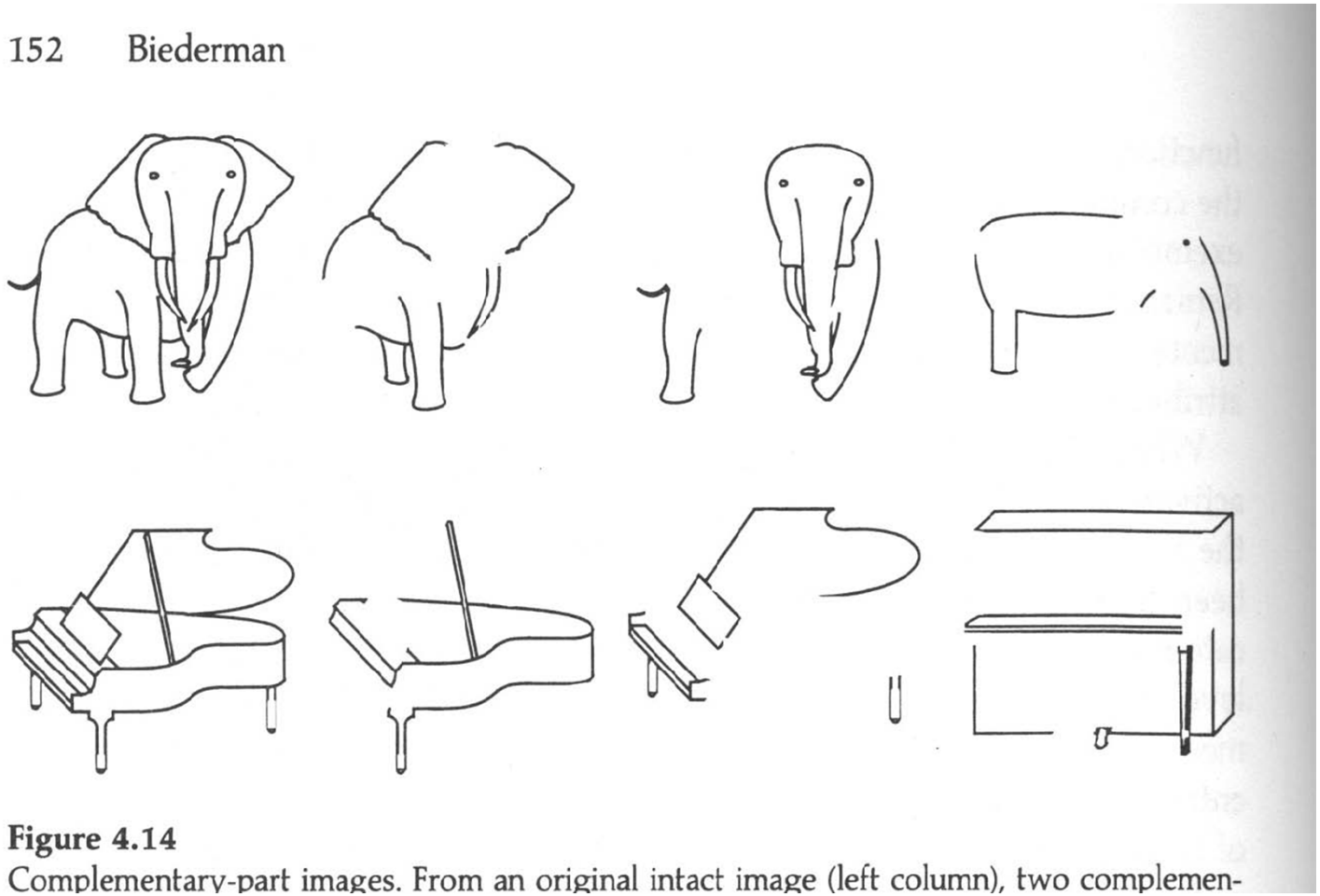
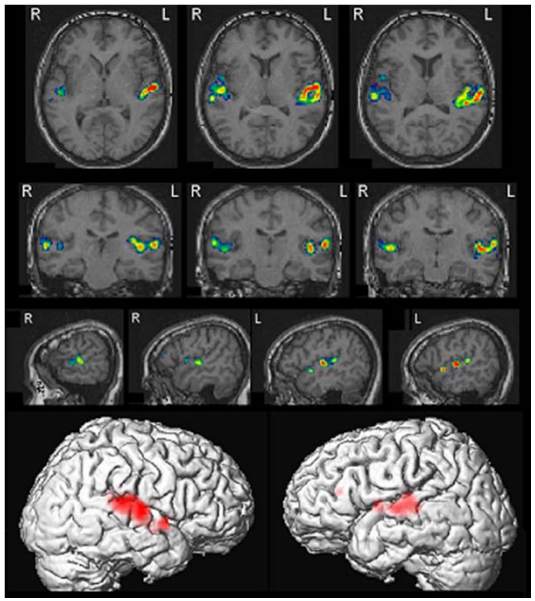
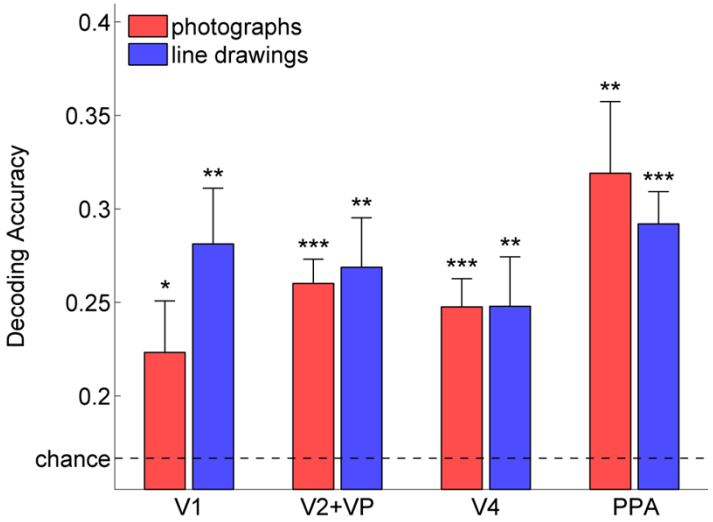


Figure 4.14
Complementary-part images. From an original intact image (left column), two complemen-





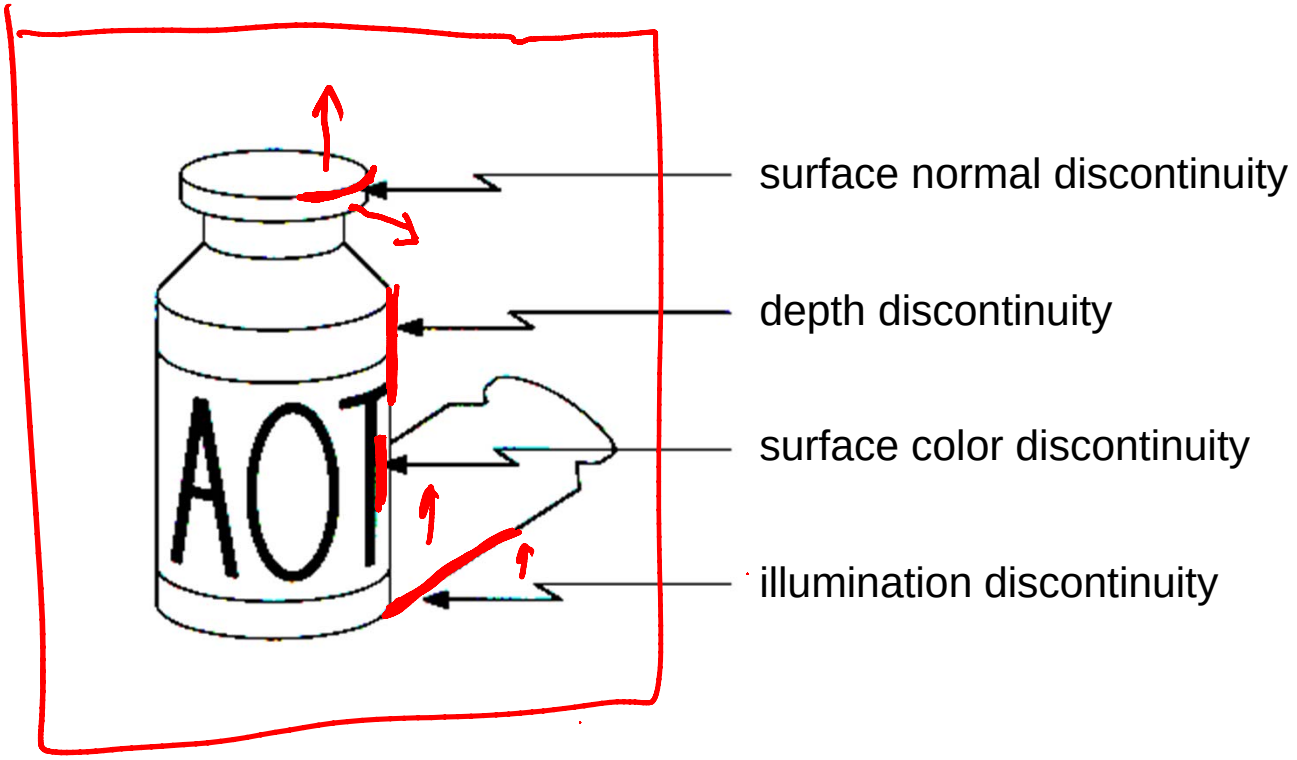
Walther, Chai, Caddigan, Beck & Fei-Fei, *PNAS, 2011*

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

Origin of Edges



- Edges are caused by a variety of factors

Source: Steve Seitz

What we will learn today

- Edge detection
  - Canny edge detector
- Line fitting
  - Hough Transform
  - RANSAC

Characterizing edges

- An edge is a place of rapid change in the image intensity function

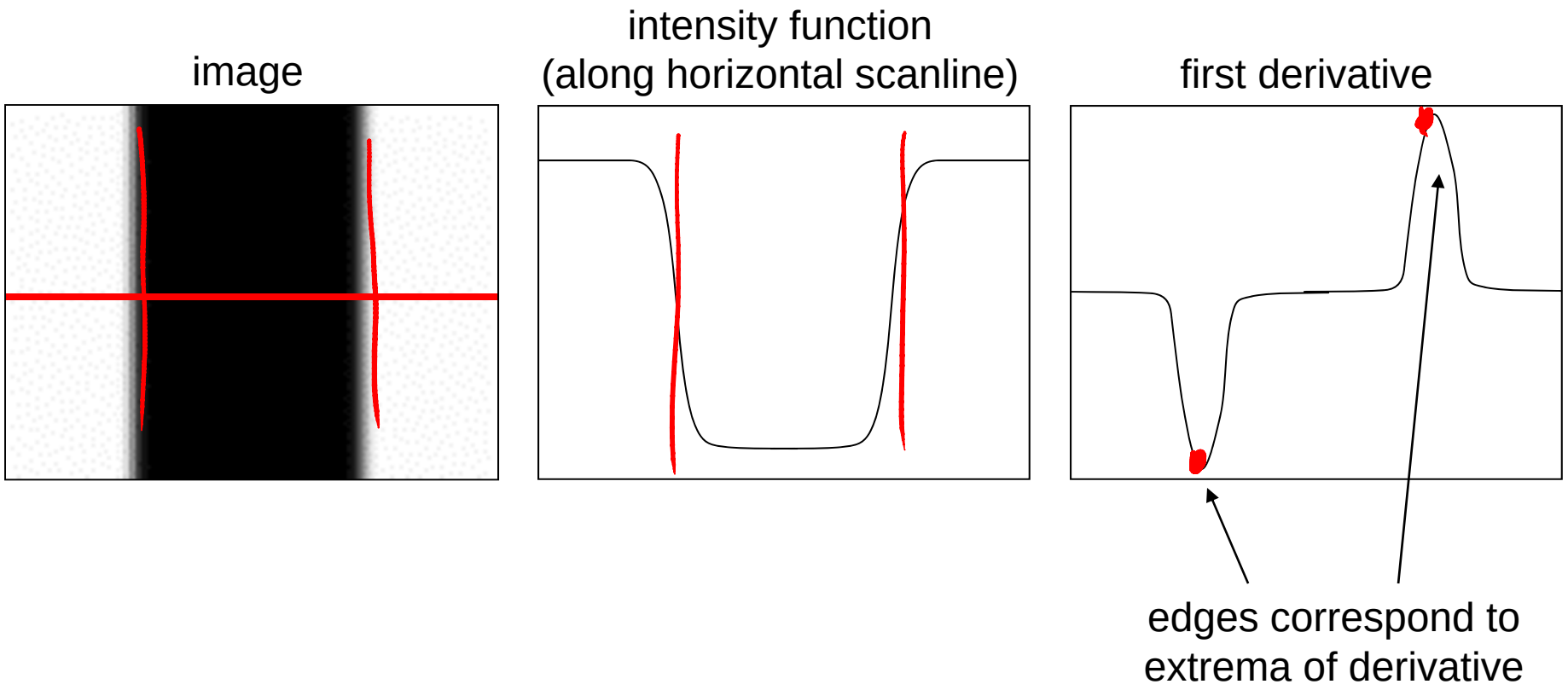
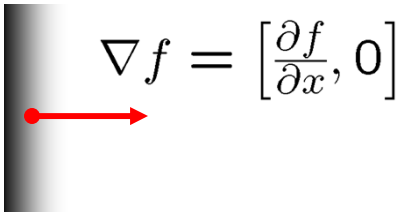
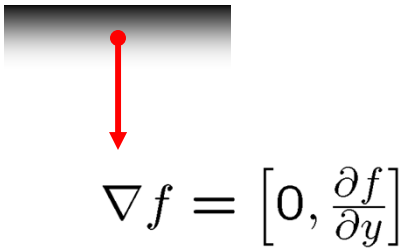
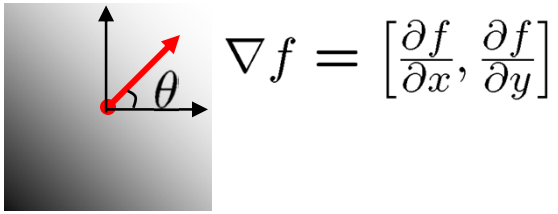


Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$







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

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

- how does this relate to the direction of the edge? Orthogonal

The *edge strength* is given by the gradient magnitude

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

Source: Steve Seitz

Differentiation and convolution

•Recall, for 2D function, f(x,y):

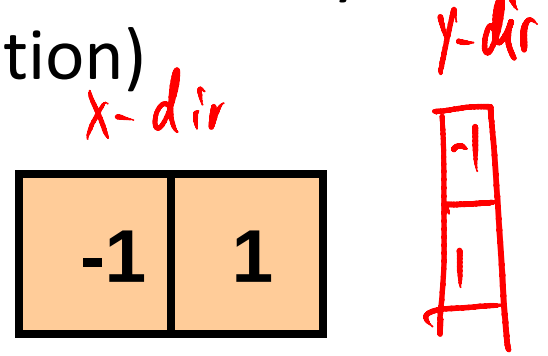
$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon} \right)$$

•We could approximate this as

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

•This is linear and shift invariant, so must be the result of a convolution.

•(which is obviously a convolution)



Source: D. Forsyth, D. Lowe

Finite difference filters

- Other approximations of derivative filters

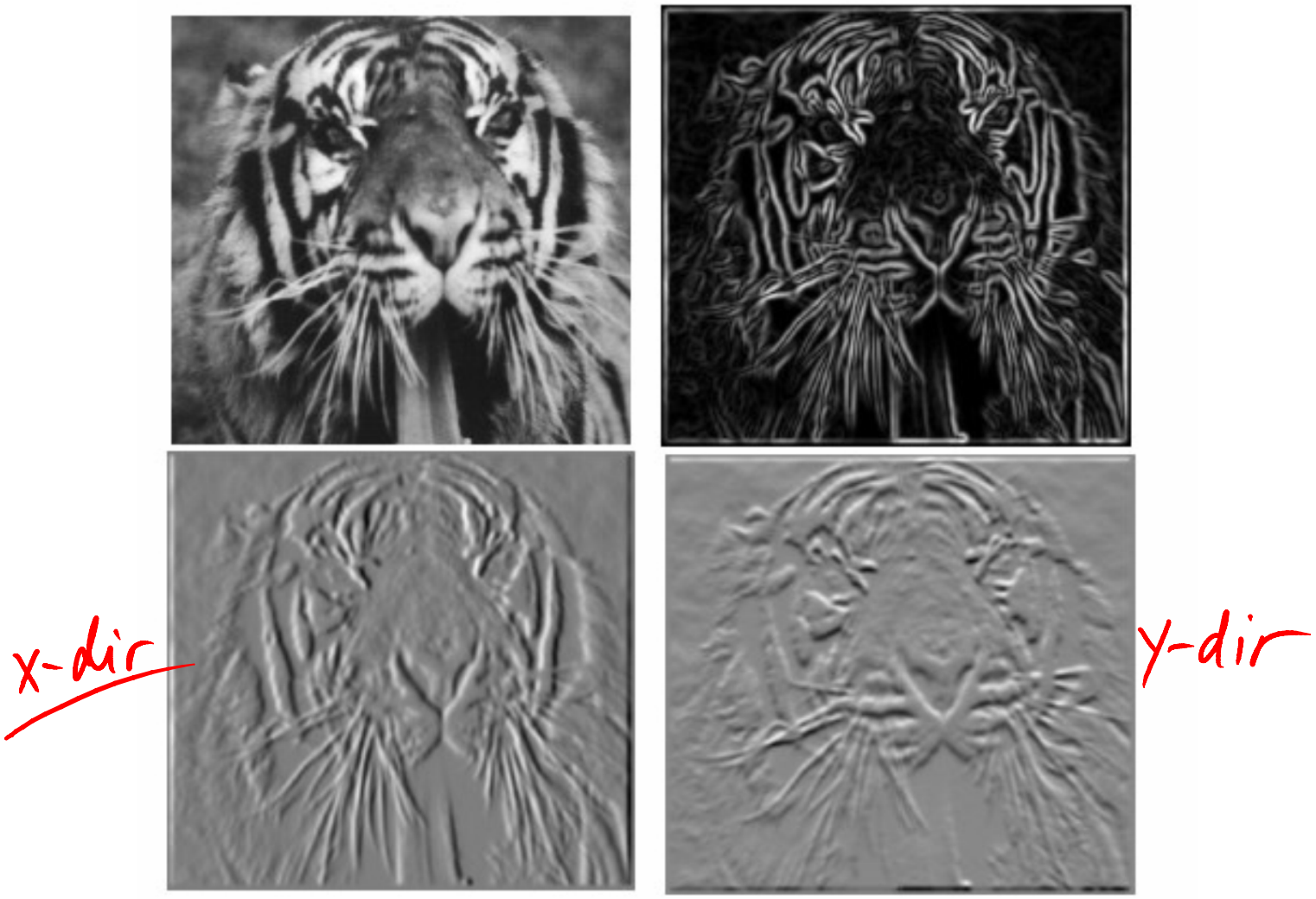
Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

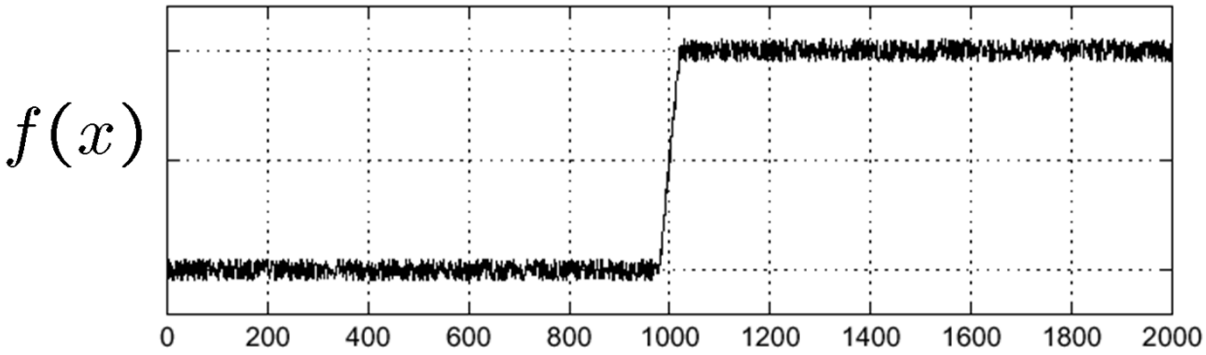
Finite differences: example

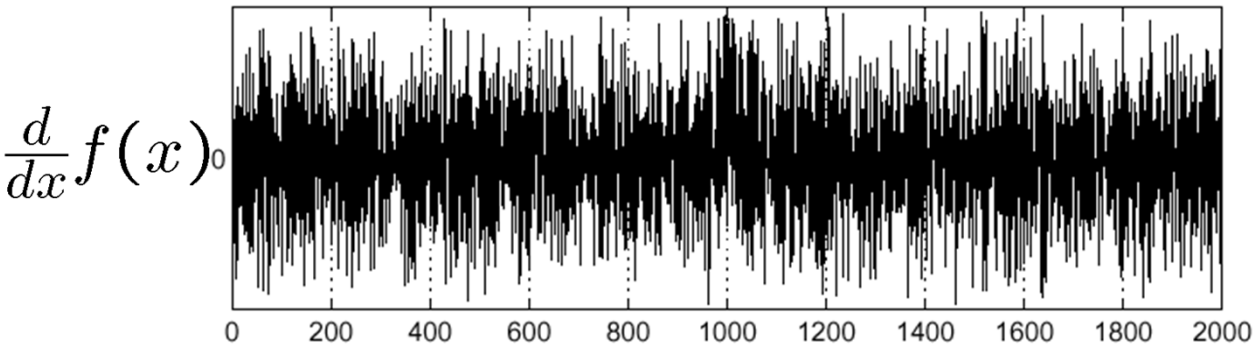


- Which one is the gradient in the x-direction? How about y-direction?

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

Source: S. Seitz

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?

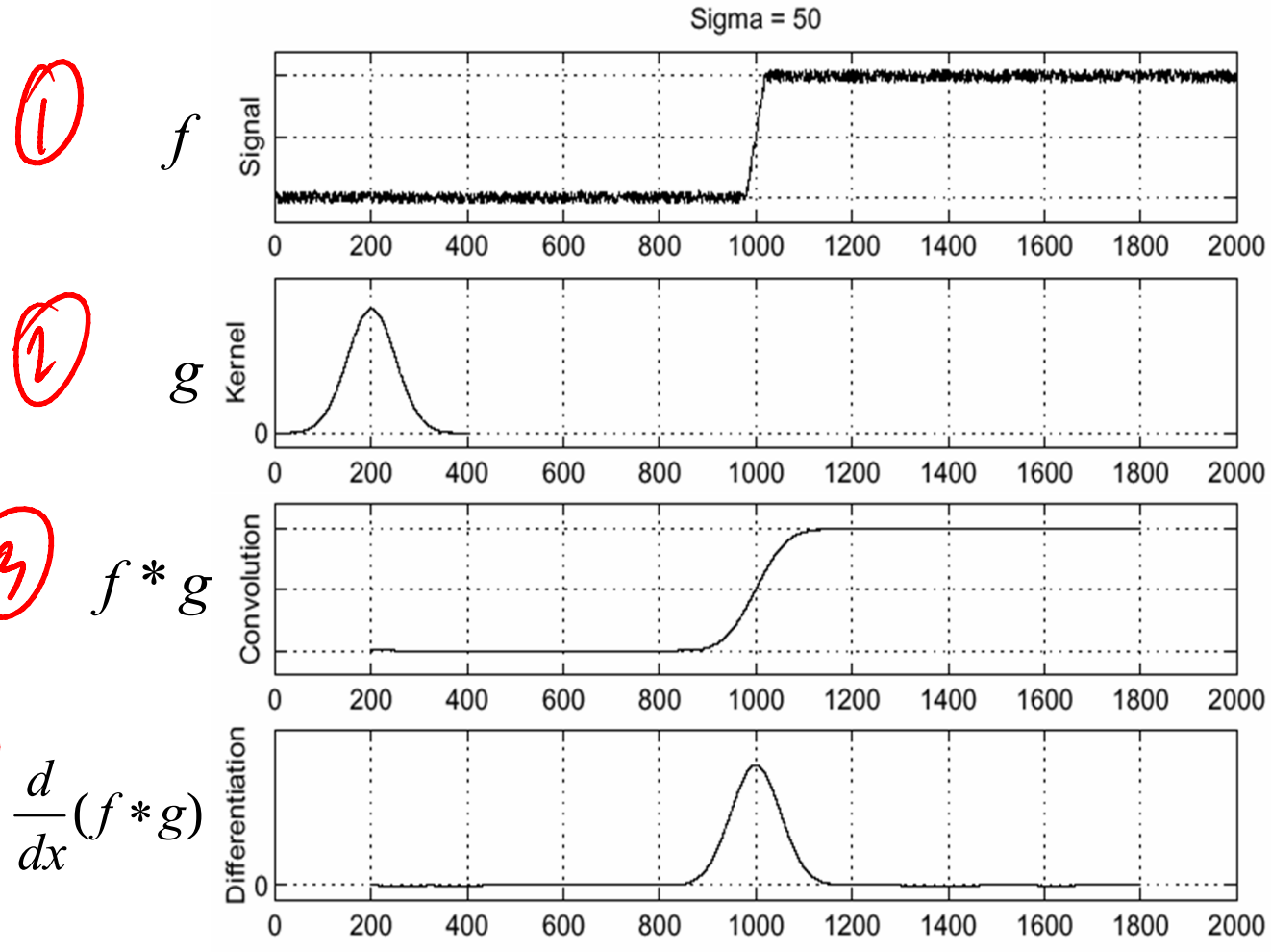
Source: D. Forsyth

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

Solution: smooth first



1. f
2. g
3. $f * g$
4. $\frac{d}{dx}(f * g)$

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

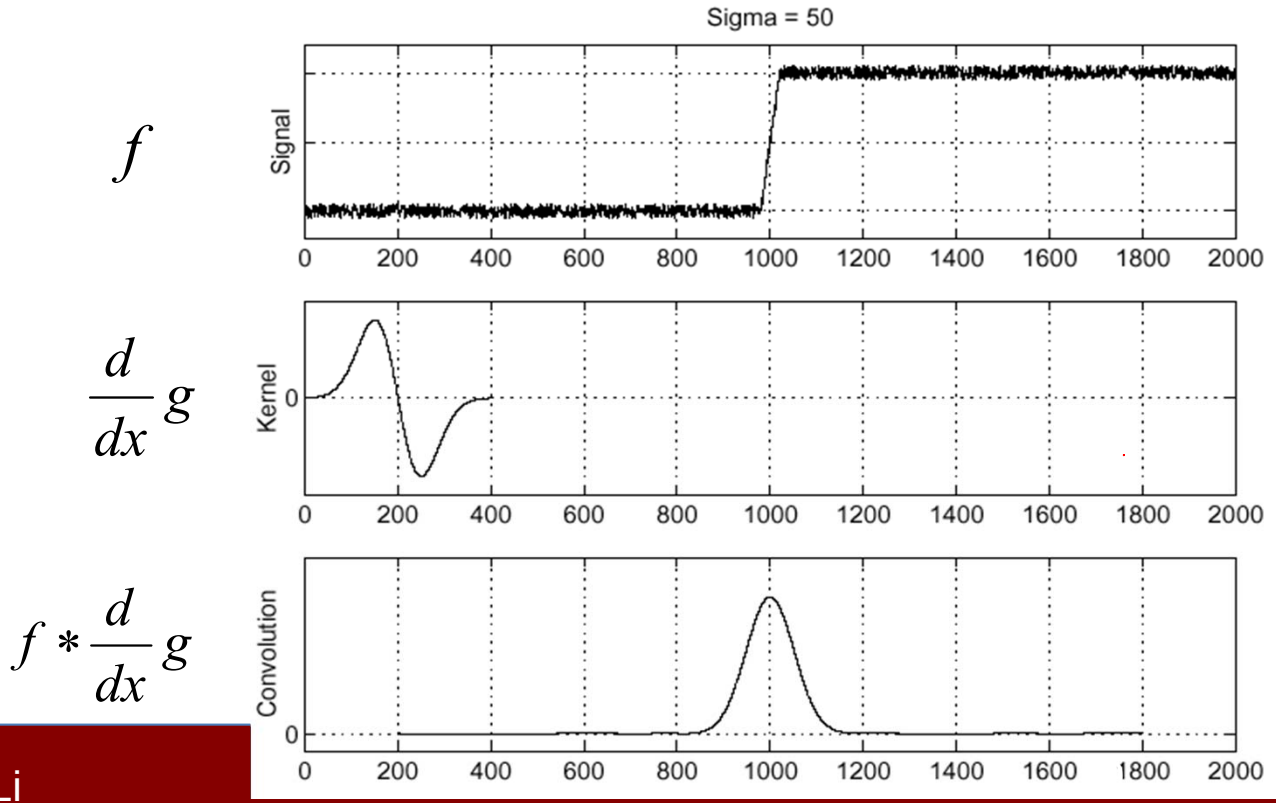
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

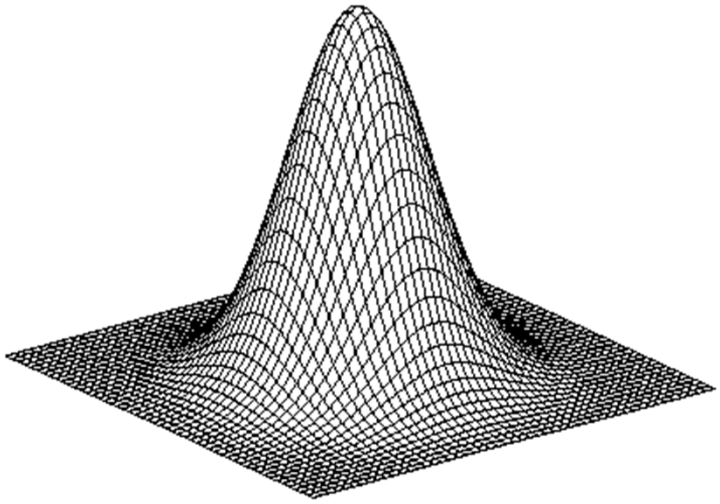
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

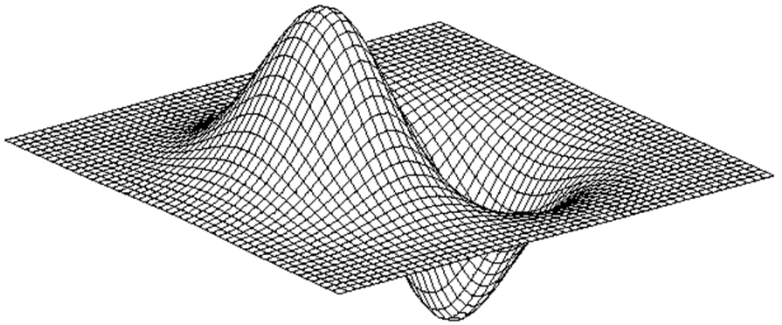


Source: S. Seitz

Derivative of Gaussian filter

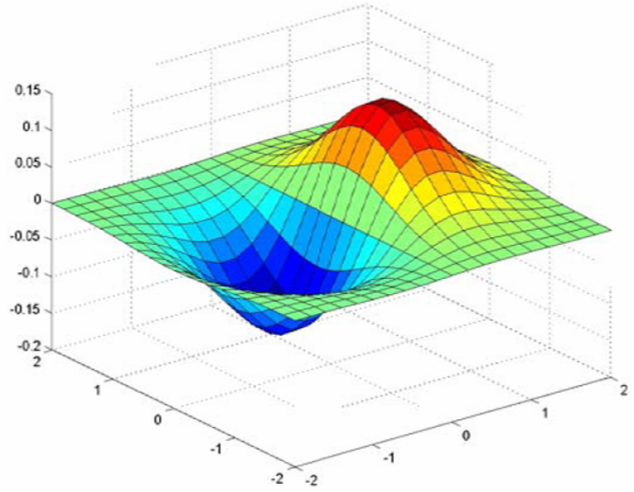


* [1 -1] =

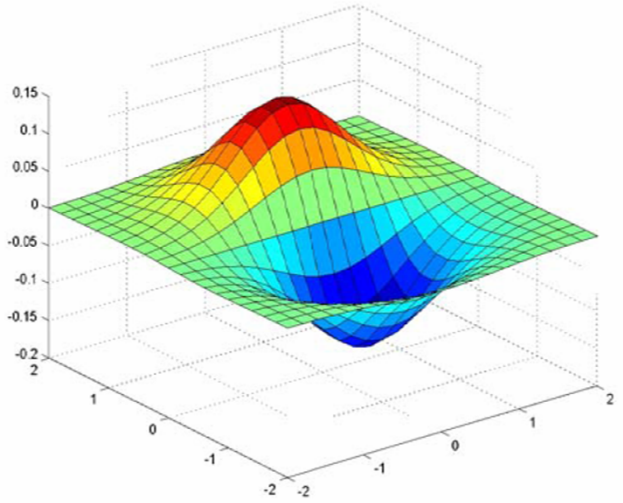


- Is this filter separable?

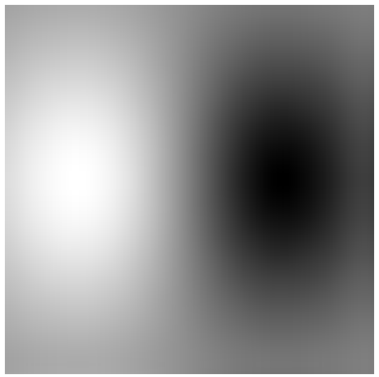
Derivative of Gaussian filter

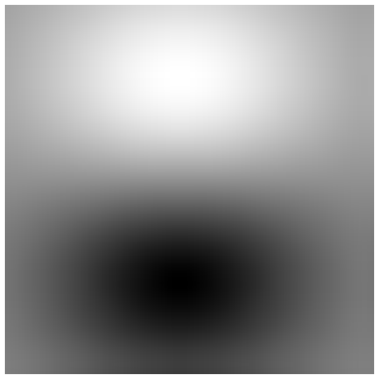


x-direction



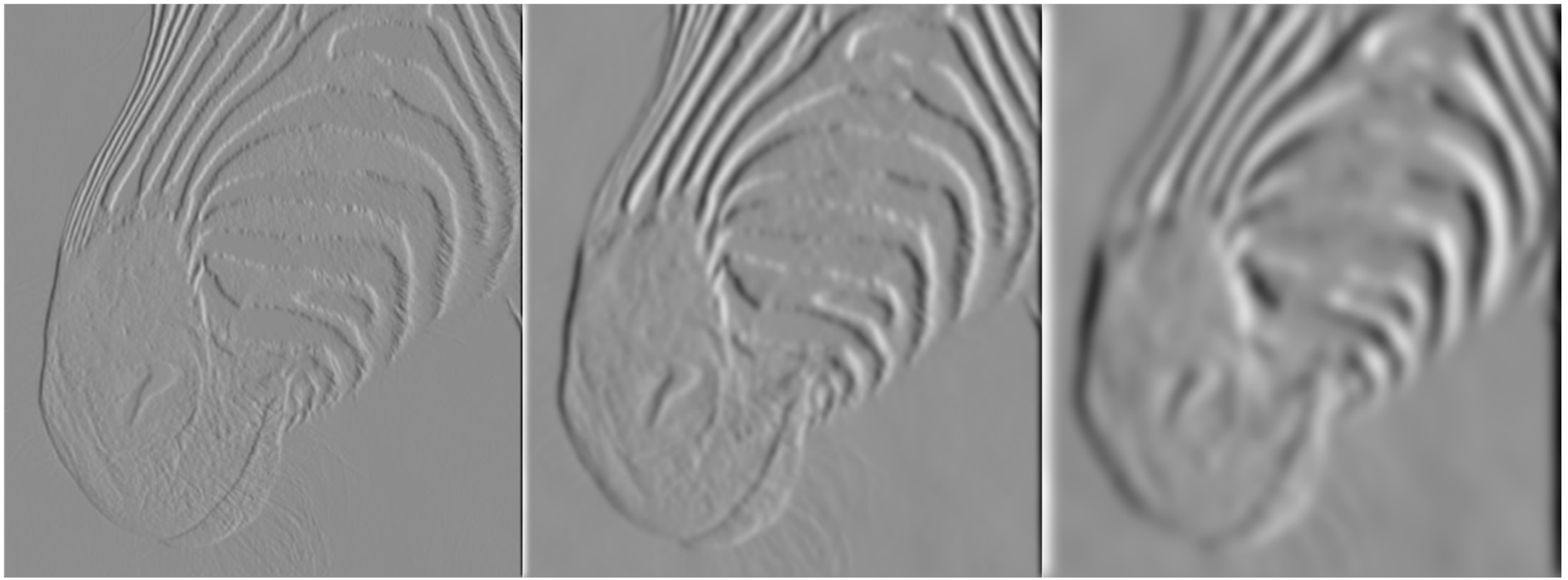
y-direction





- Which one finds horizontal/vertical edges?

Tradeoff between smoothing and localization



1 pixel 3 pixels 7 pixels

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

Source: D. Forsyth

Implementation issues

- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

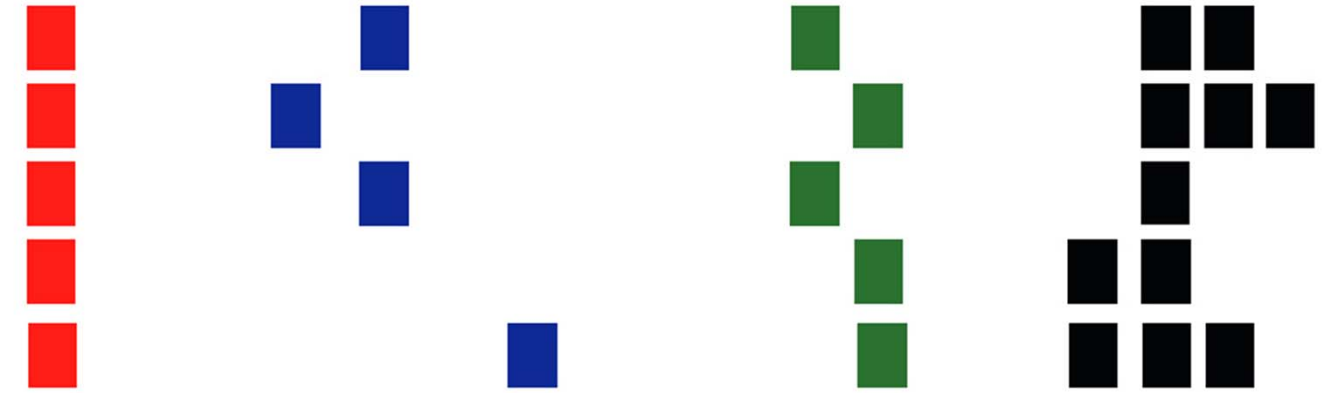
Source: D. Forsyth

Designing an edge detector

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

Designing an edge detector

- Criteria for an "optimal" edge detector:
  - **Good detection**
  - **Good localization**
  - **Single response**



Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

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

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

- MATLAB: edge(image, ‘canny’)

Example

- original image (Lena)

Example

norm of the gradient

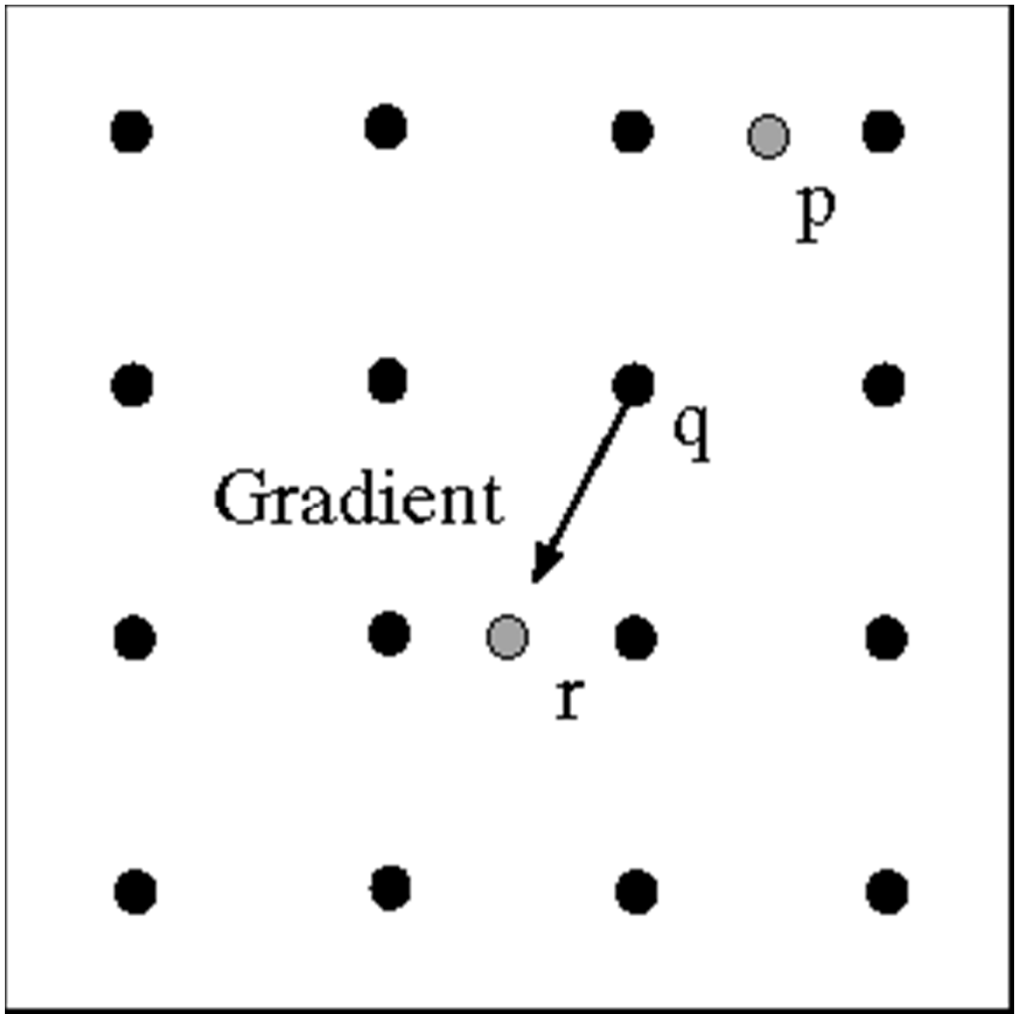
Example

thresholding

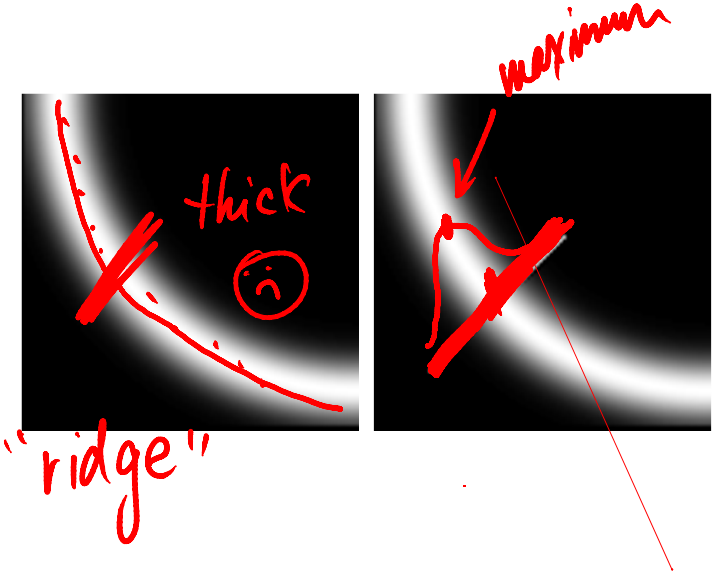
Example

Thinning (non-maximum suppression)

Non-maximum suppression

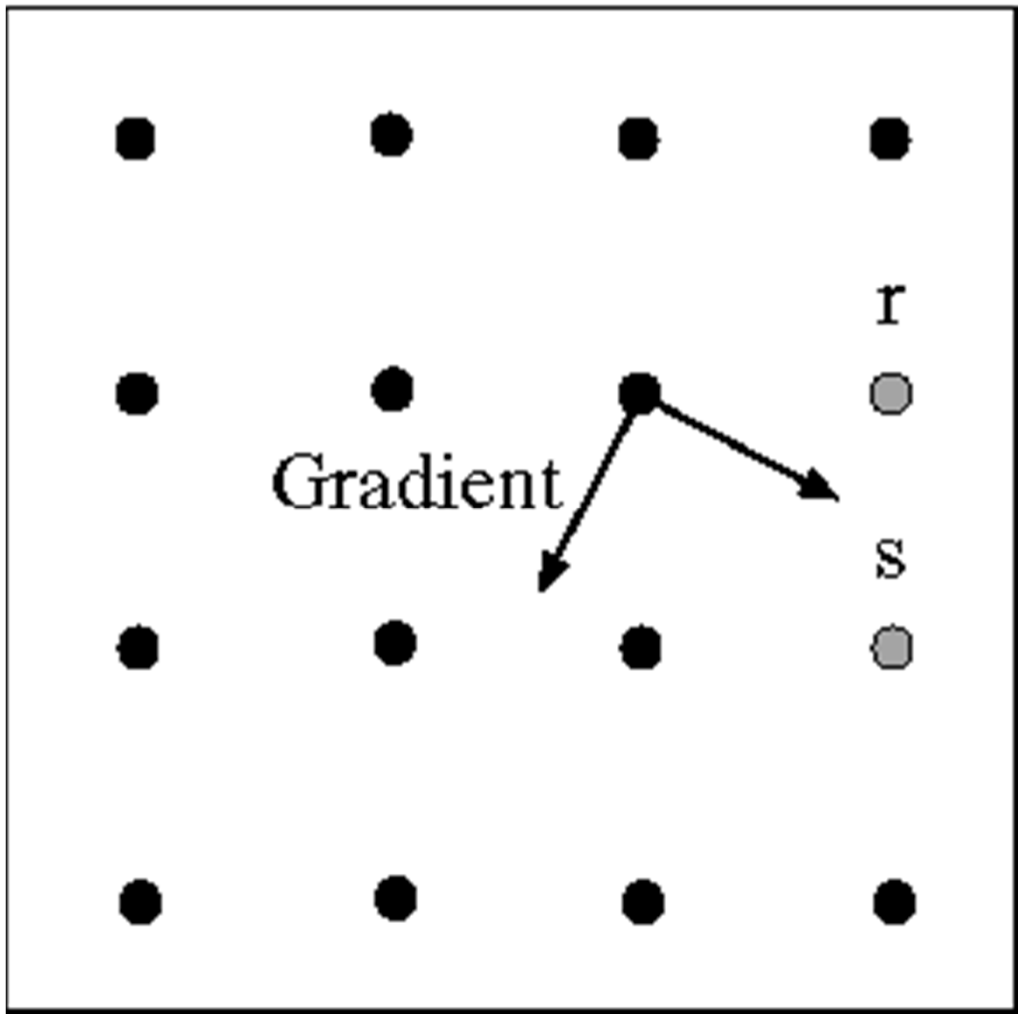


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

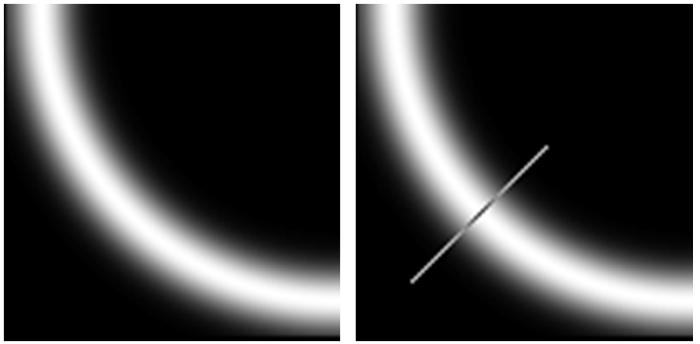


Source: D. Forsyth

Edge linking



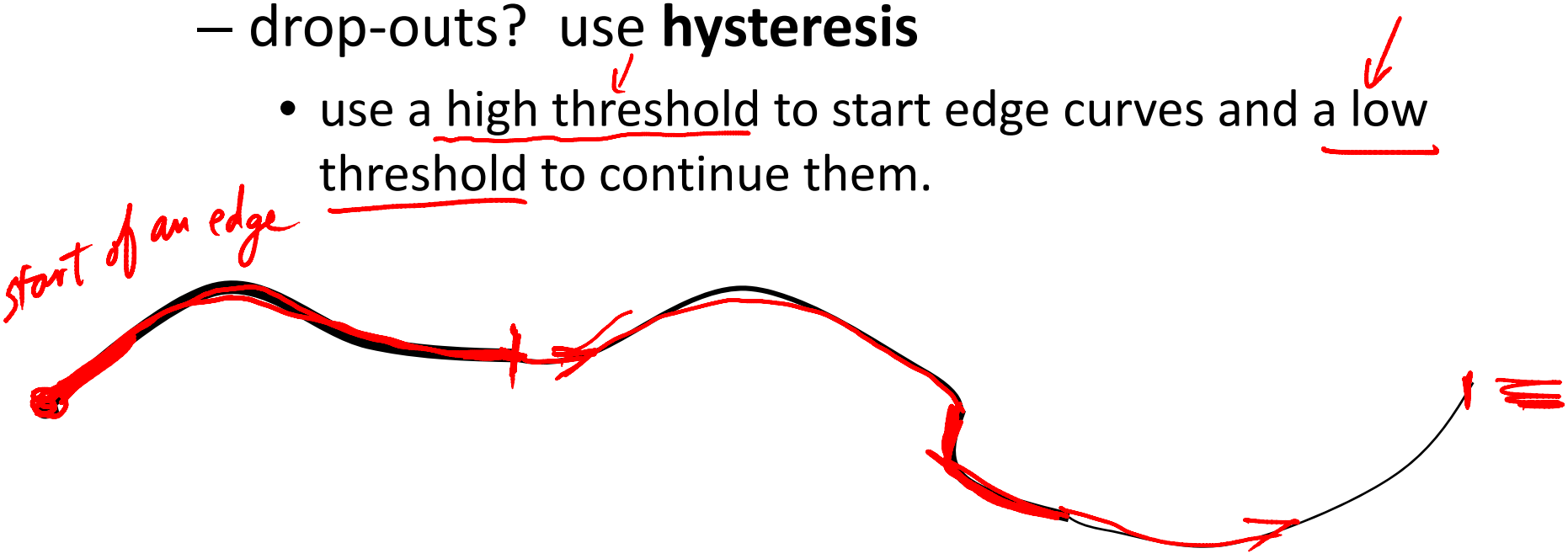
Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Source: D. Forsyth

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



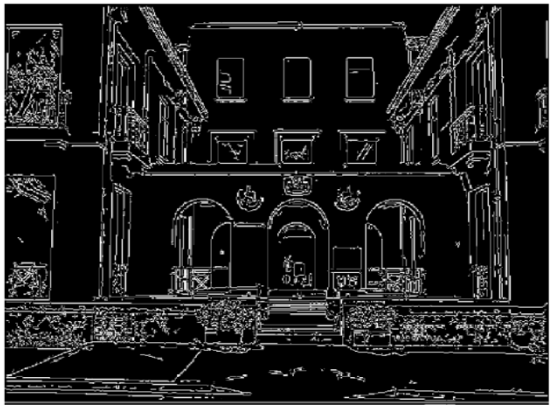
Source: S. Seitz

Hysteresis thresholding

original image

high threshold
(strong edges)

low threshold
(weak edges)



hysteresis threshold

Effect of σ (Gaussian kernel spread/size)

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

The choice of σ depends on desired behavior

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

Source: S. Seitz

Edge detection is just the beginning…

image | human segmentation | gradient magnitude

- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

What we will learn today

- Edge detection
  - Canny edge detector
- Line fitting
  - Hough Transform
  - RANSAC (Problem Set 2 (Q5))

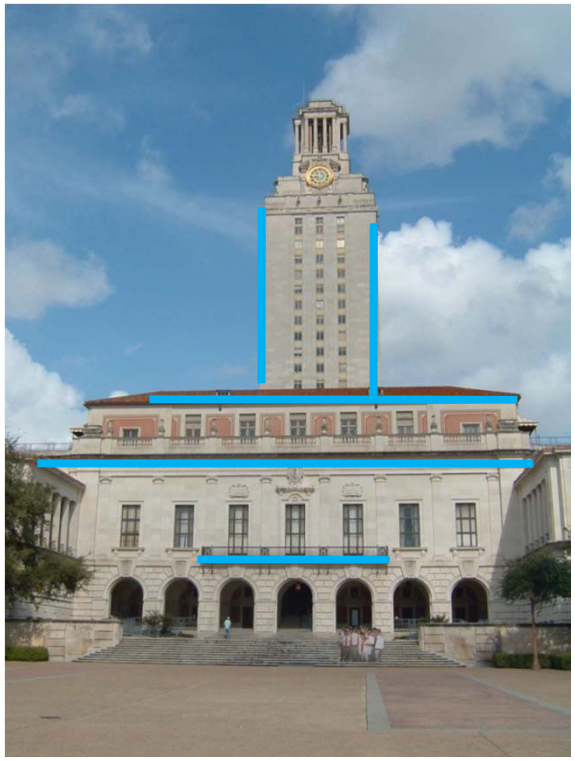
Fitting as Search in Parametric Space

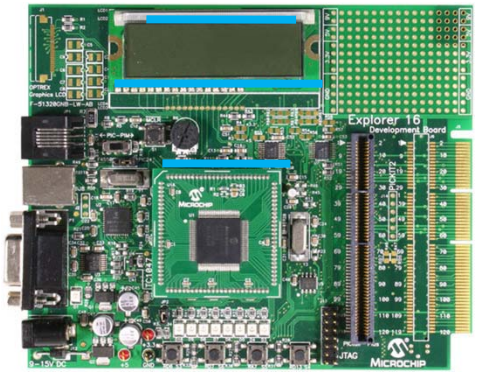
- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can't tell whether a point belongs to a given model just by looking at that point.
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets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

Source: L. Lazebnik

Example: Line Fitting

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

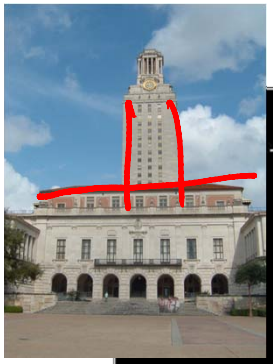


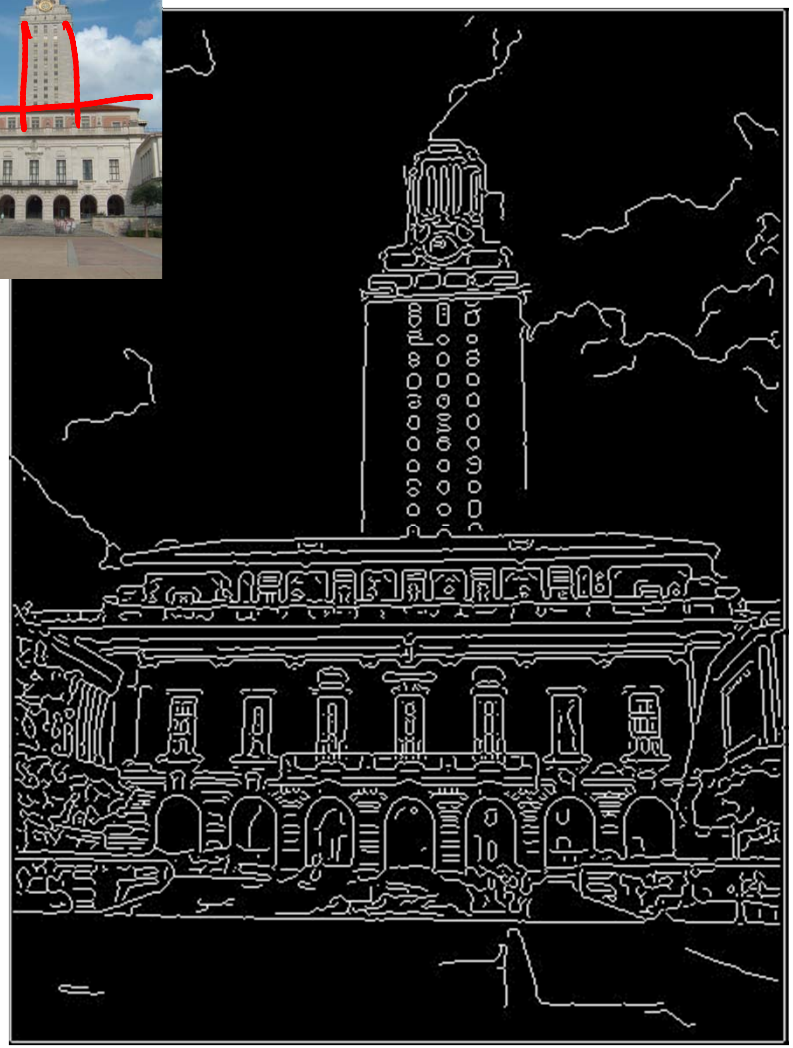


- Wait, why aren't we done just by running edge detection?

Slide credit: Kristen Grauman

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?

Slide credit: Kristen Grauman

Voting in parameter space

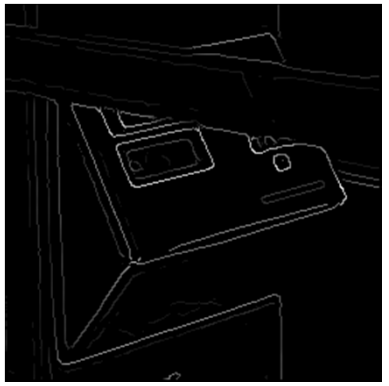
- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- Voting is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that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

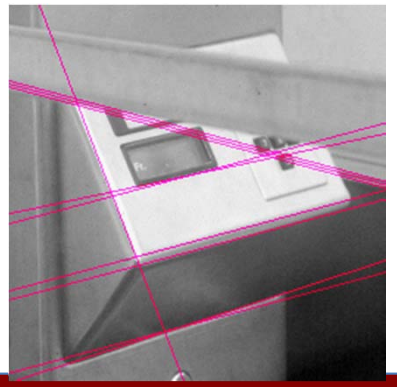
Slide credit: Kristen Grauman

Fitting Lines

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

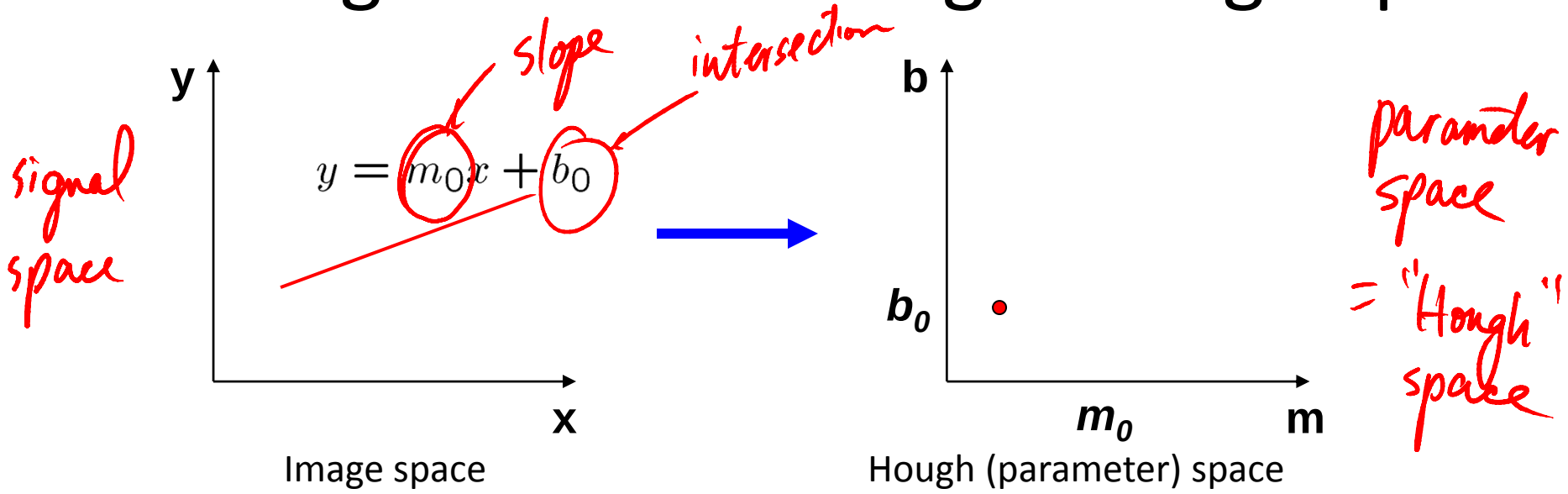
- *Hough Transform* is a voting technique that can be used to answer all of these
- Main idea:
  1. Record all possible lines on which each edge point lies.
  2. Look for lines that get many votes.





Slide credit: Kristen Grauman

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 $y = mx + b$

Slide credit: Steve Seitz

Finding Lines in an Image: Hough Space

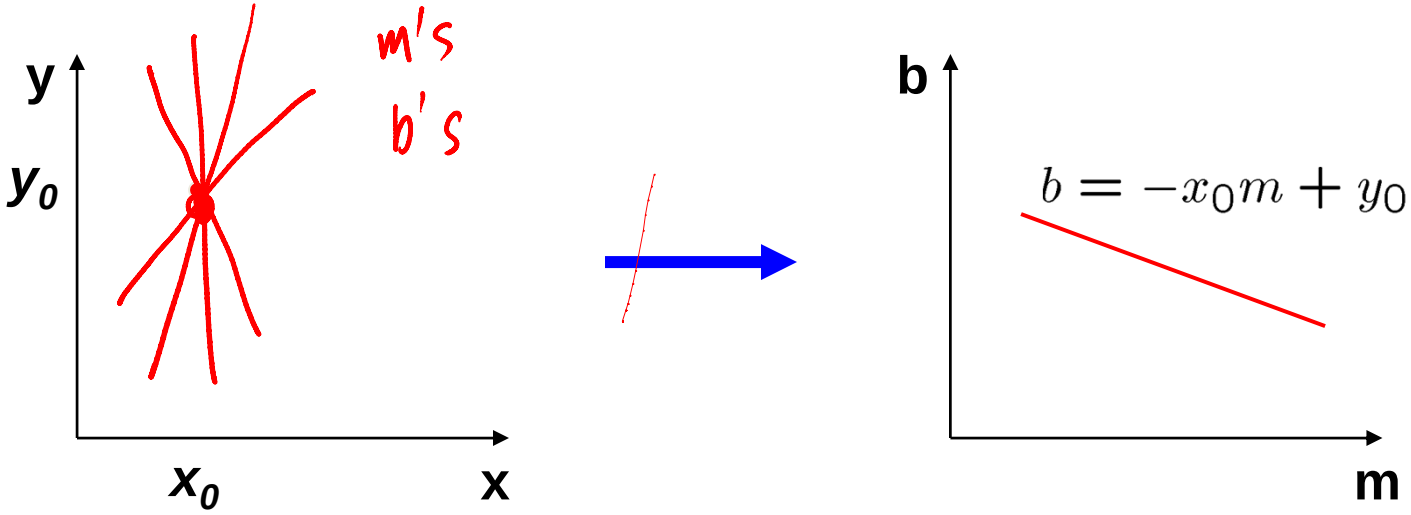


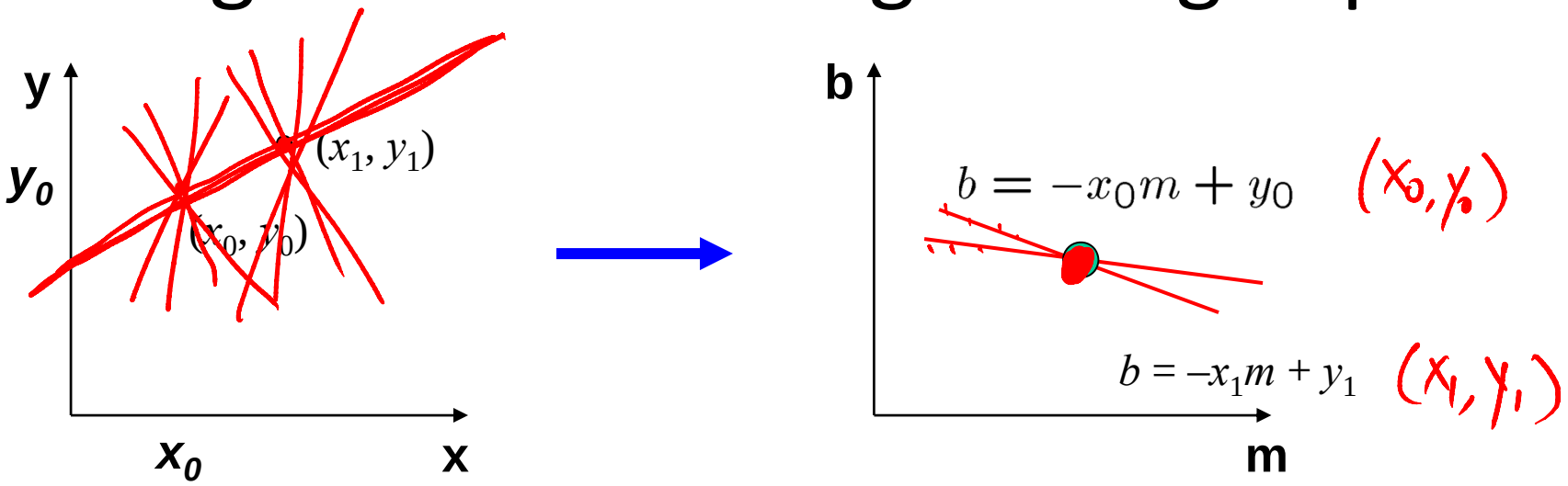
Image space

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 $y = mx + b$
  - 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

Slide credit: Steve Seitz

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 $b = -x_0m + y_0$ and $b = -x_1m + y_1$

Slide credit: Steve Seitz

Finding Lines in an Image: Hough Space

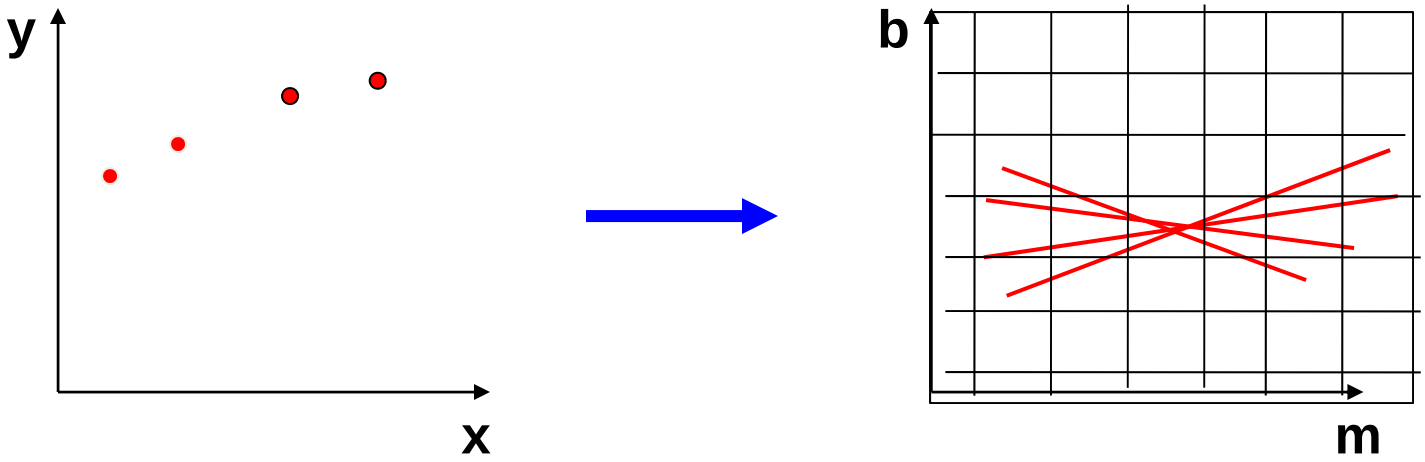


Image space

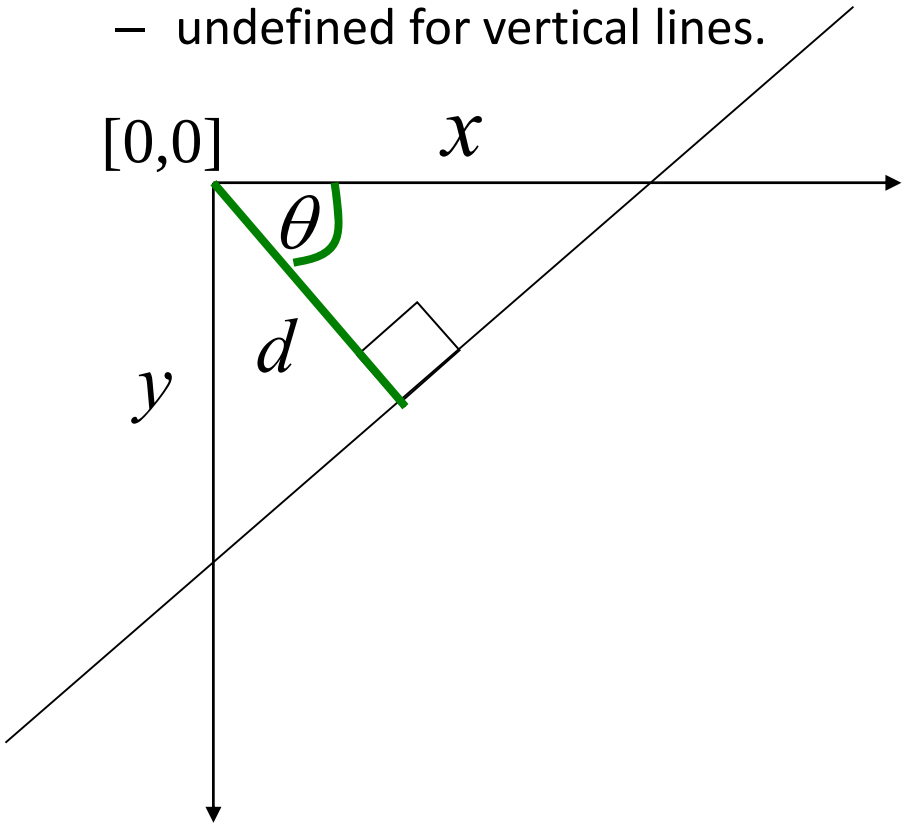
Hough (parameter) space

- 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.

Slide credit: Steve Seitz

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

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

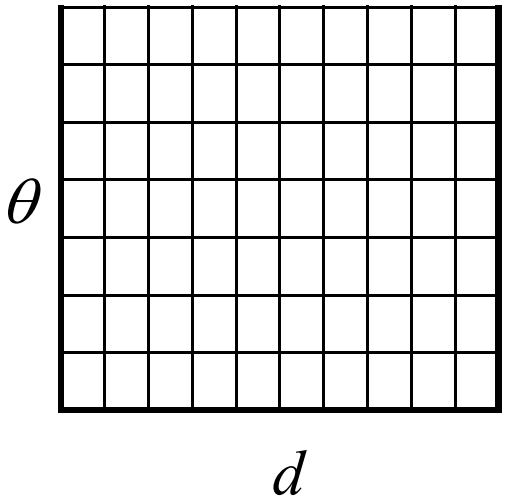
where $\theta \in [0,\pi)$ and d ∈ **R**

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

Slide credit: Steve Seitz

Hough Transform Algorithm

H: accumulator array (votes)



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 $\theta \in [0,\pi)$ // some quantization
   H[d, θ] += 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)?

Hough line demo

Slide credit: Steve Seitz

Example: HT for Straight Lines

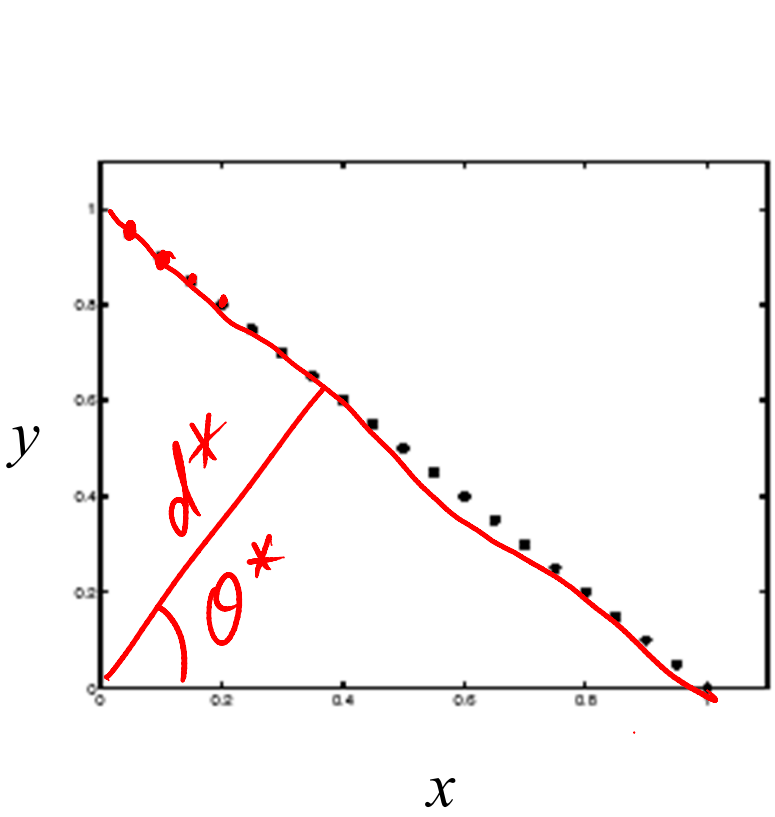
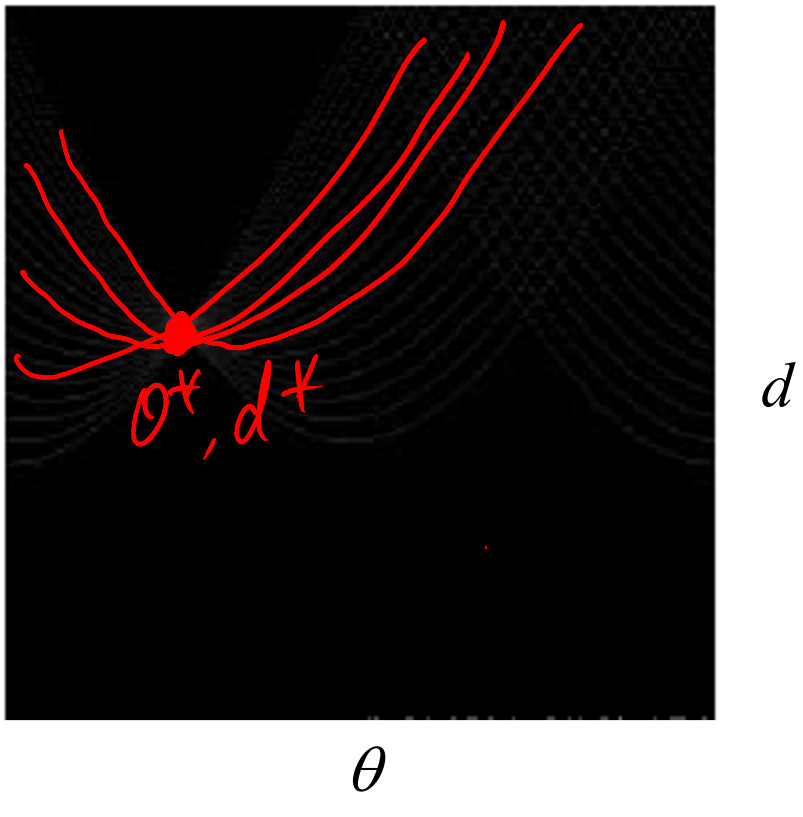


Image space
edge coordinates



Votes

Bright value = high vote count
Black = no votes

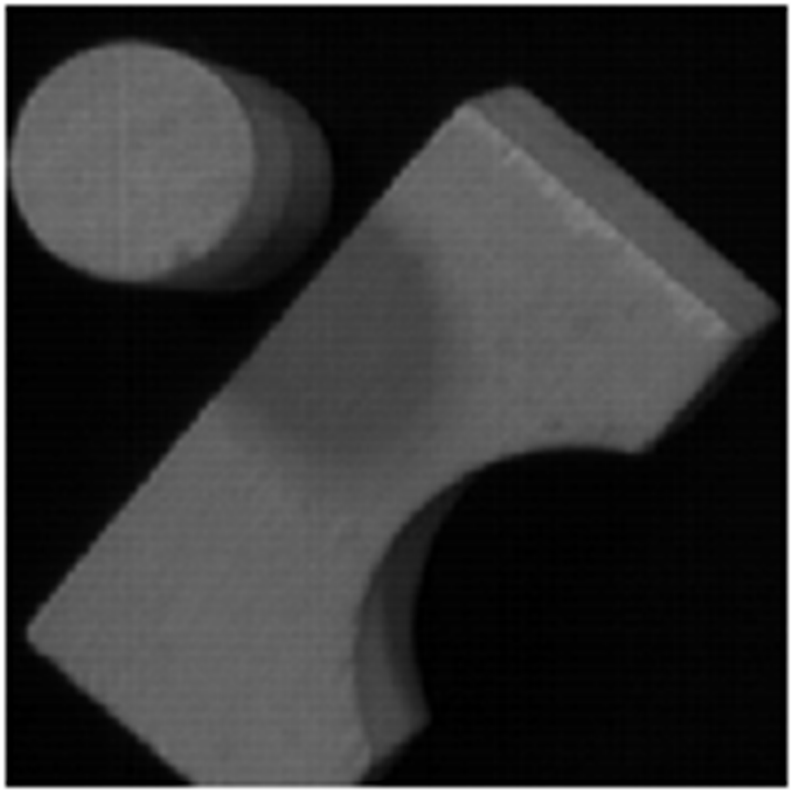
Slide credit: Steve Seitz

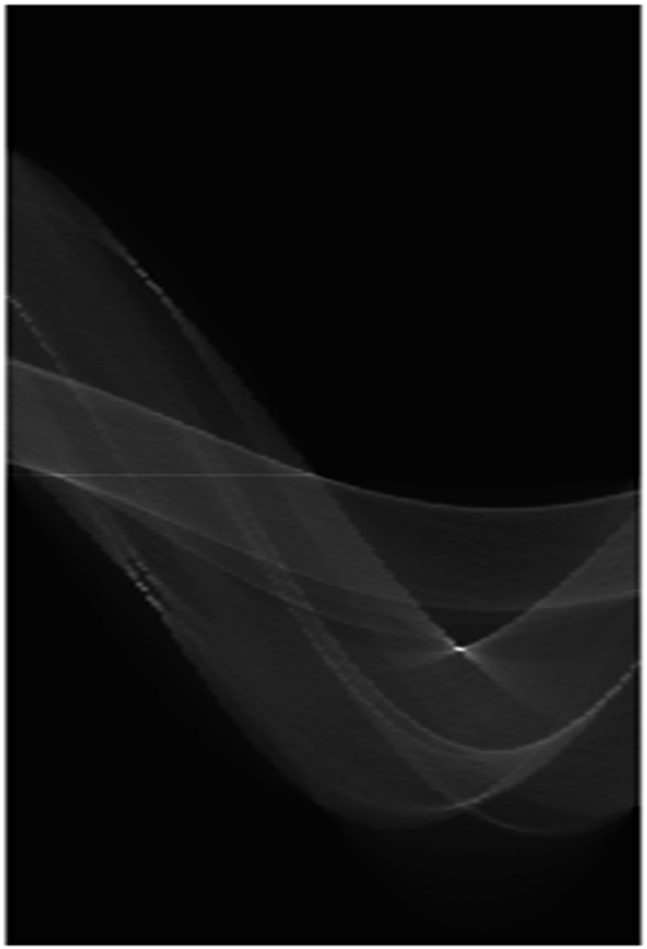
Example: HT for Straight Lines

Square:

Slide credit: David Lowe

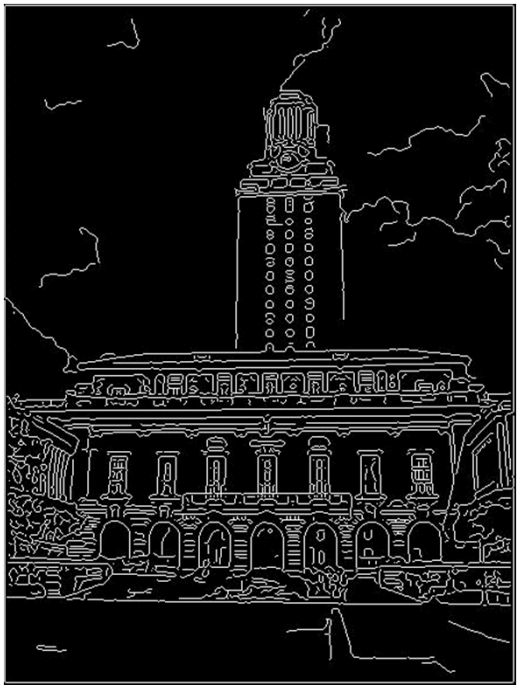
Example: HT for Straight Lines

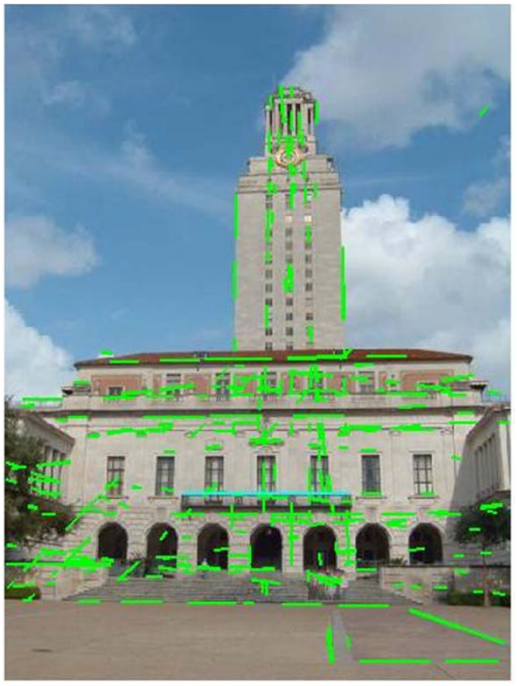


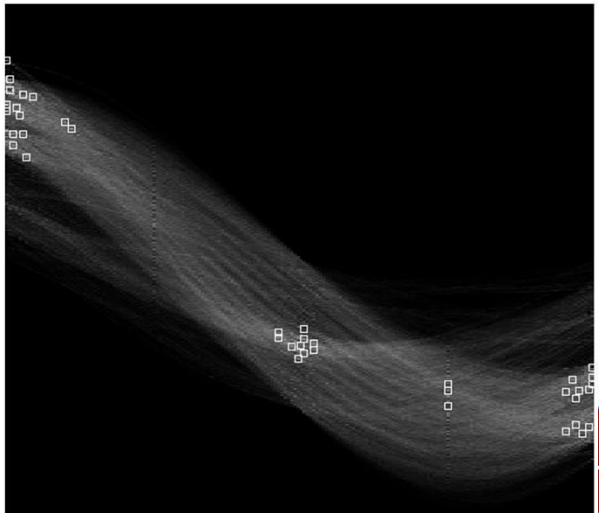


Slide credit: David Lowe

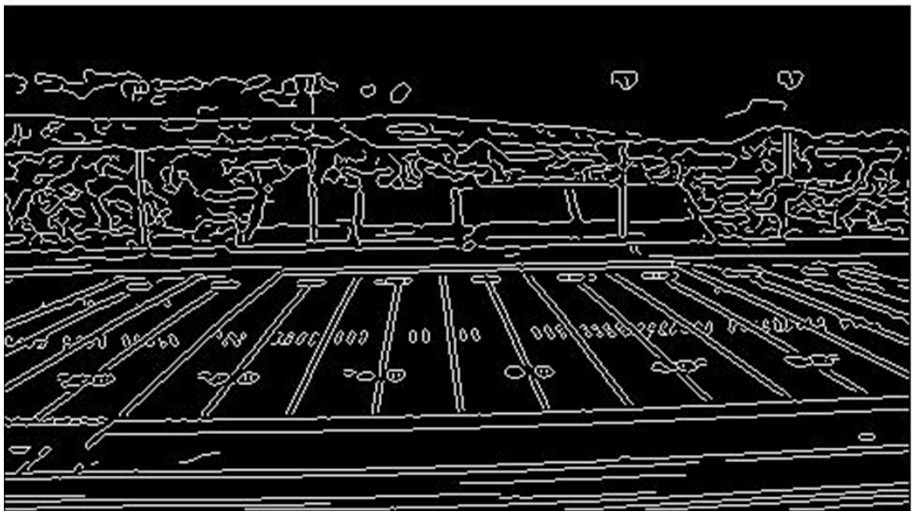
Real-World Examples

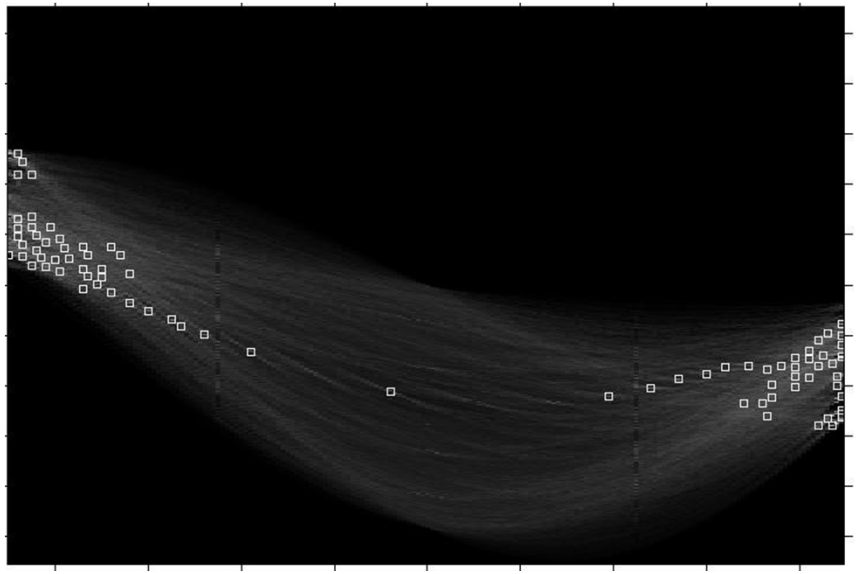


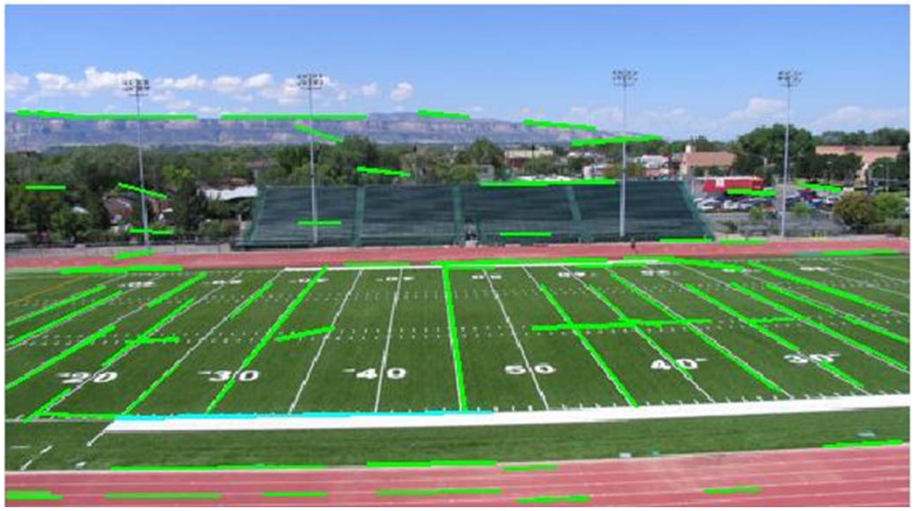




Slide credit: Kristen Grauman







Showing longest segments found

Slide credit: Kristen Grauman

Impact of Noise on Hough Transform

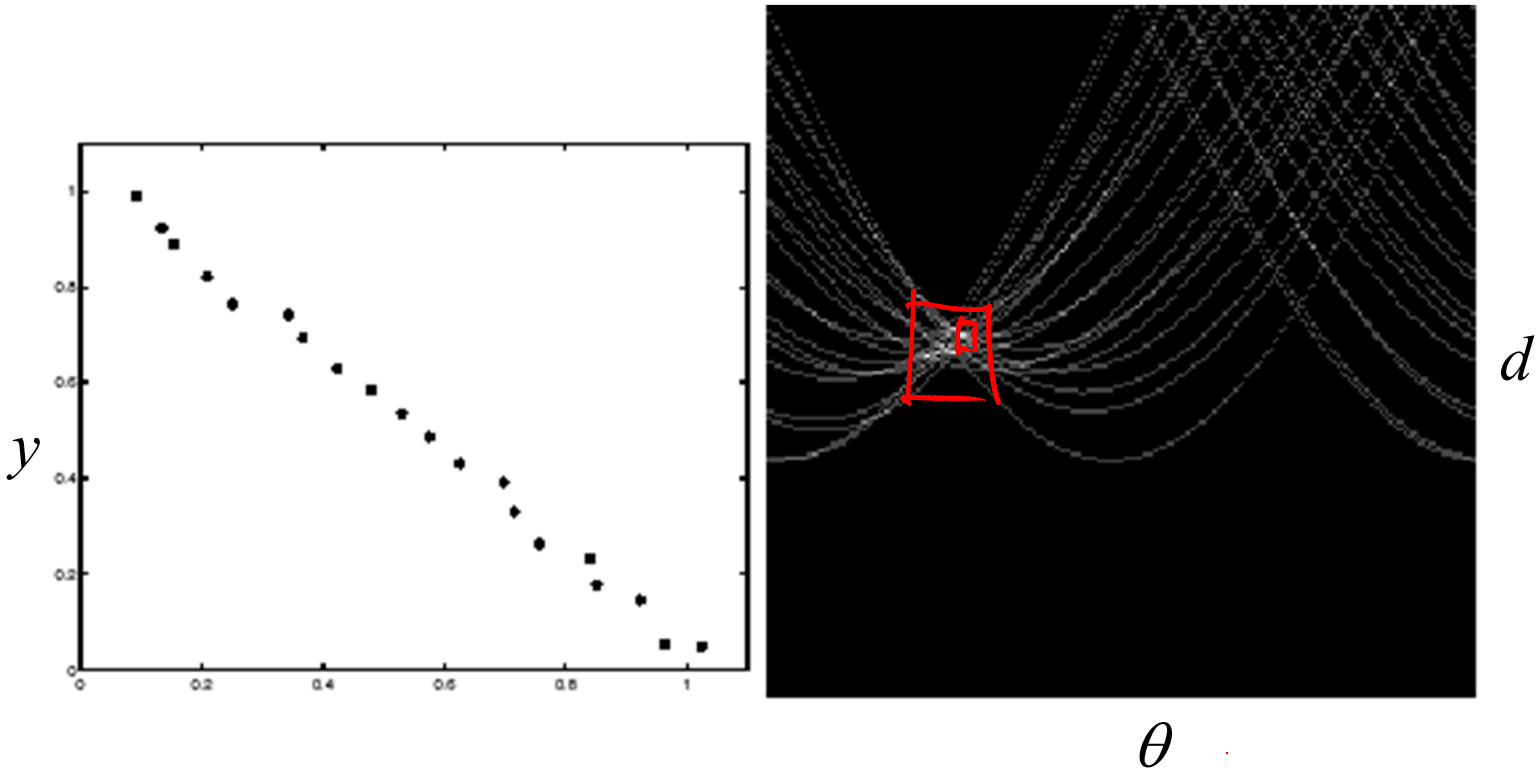


Image space
edge coordinates

Votes

What difficulty does this present for an implementation?

Slide credit: David Lowe

Impact of Noise on Hough Transform

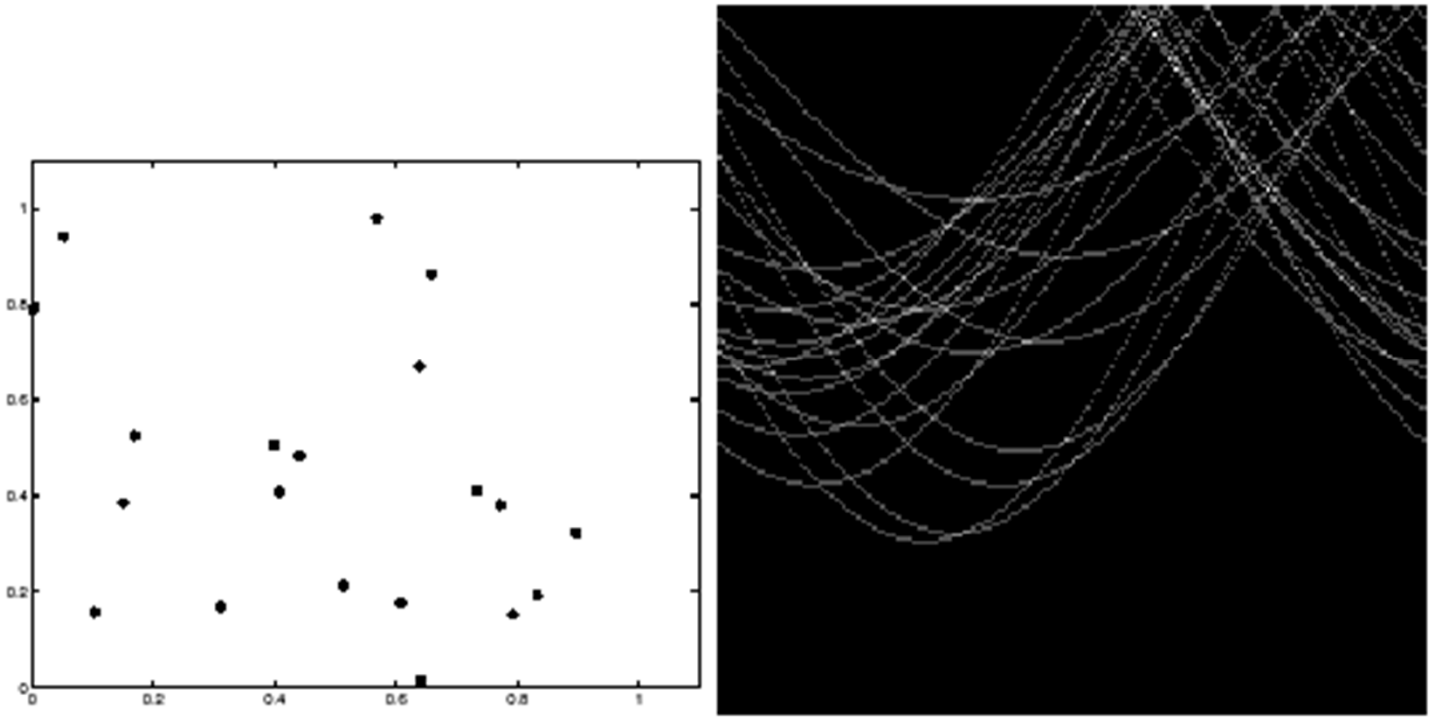


Image space
edge coordinates

Votes

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

Slide credit: David Lowe

Generalized Hough Transform

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

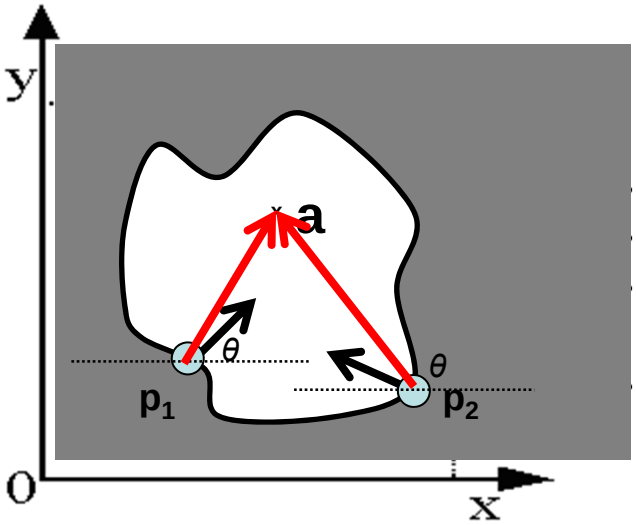


Image space

At each boundary point, compute displacement vector: $r = a - p_i$.

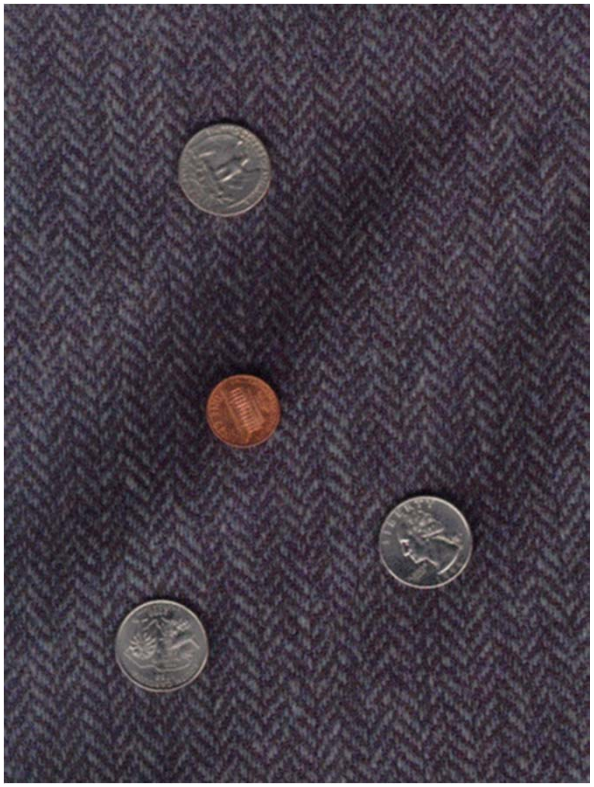
For a given model shape: store these vectors in a table indexed by gradient orientation θ.

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

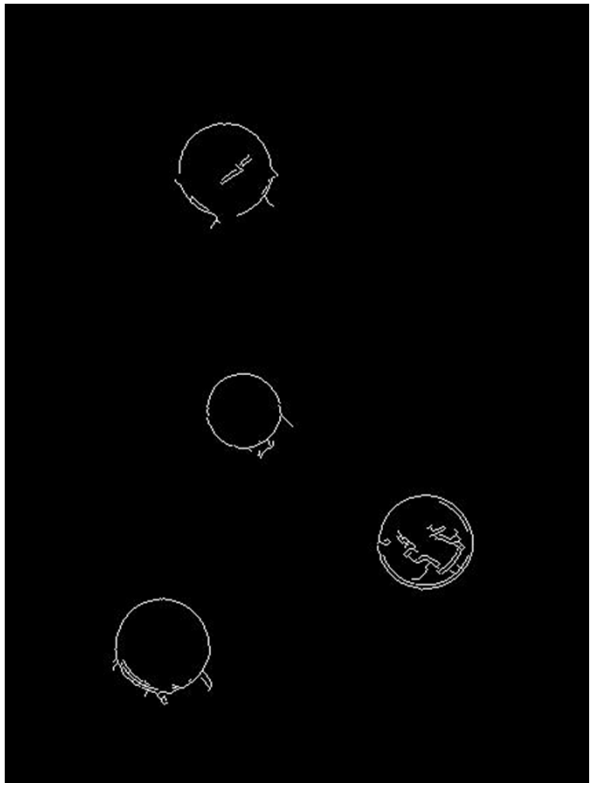
Slide credit: Kristen Grauman

Example: Detecting Circles with Hough

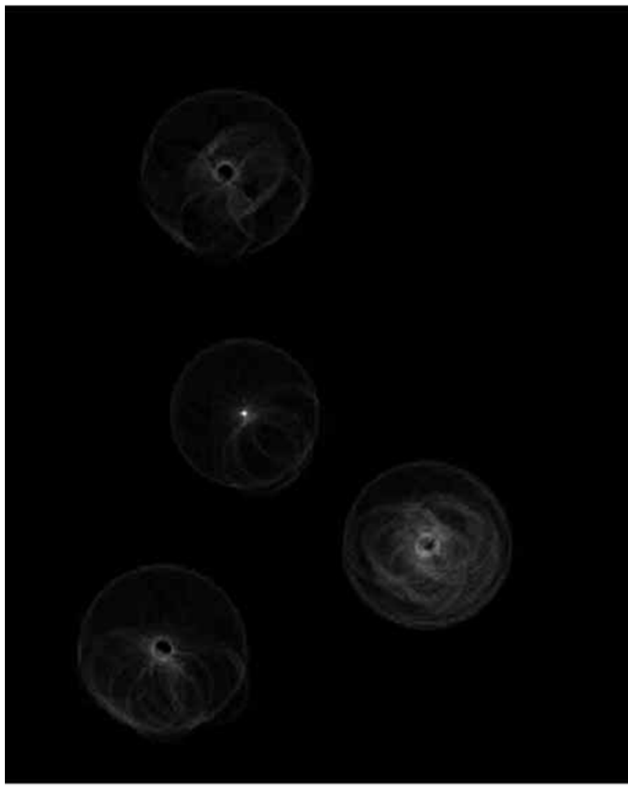
Original



Edges



Votes: Penny



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

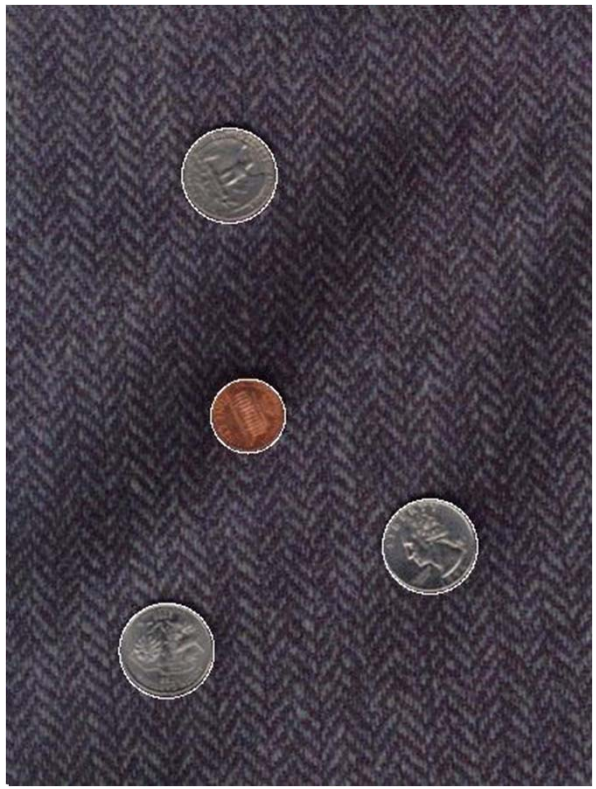
Slide credit: Kristen Grauman

Coin finding sample images from: Vivek Kwatra

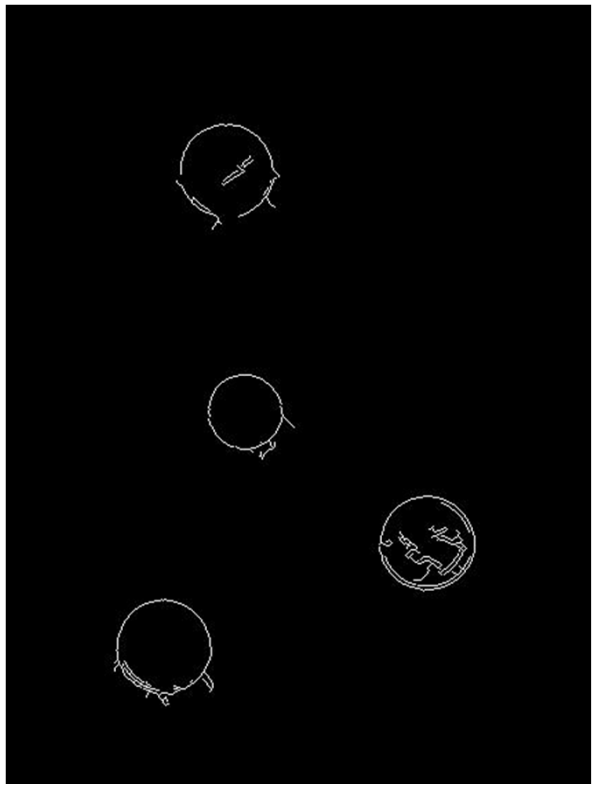
Example: Detecting Circles with Hough

Combined detections

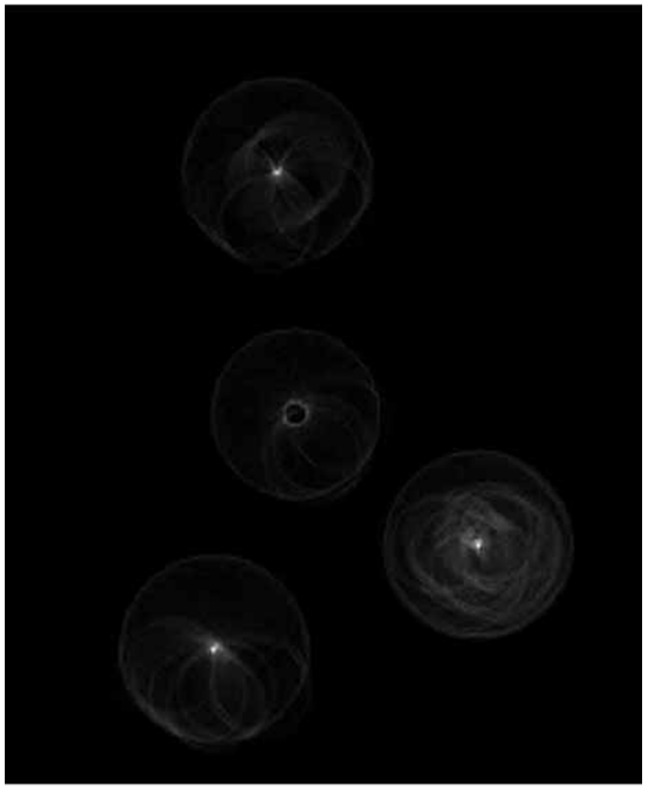
Original



Edges



Votes: Quarter



Slide credit: Kristen Grauman

Coin finding sample images from: Vivek Kwatra

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.

Slide credit: Kristen Grauman

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

Slide credit: Kristen Grauman

Another model fitting strategy: RANSAC [Fischler & Bolles 1981]

- RANdom SAmple Consensus
- 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.

Slide credit: Kristen Grauman

RANSAC

RANSAC loop:

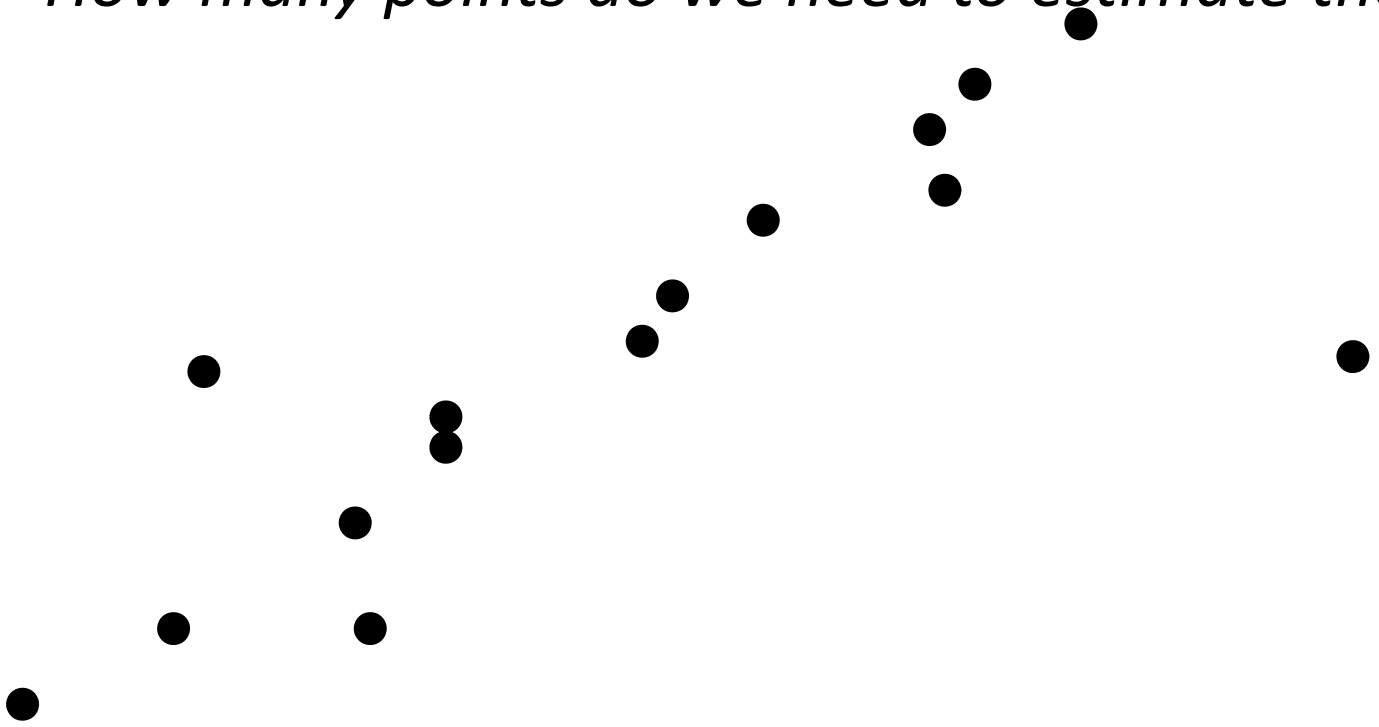
1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on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

- Keep the transformation with the largest number of inliers

Slide credit: Kristen Grauman

RANSAC Line Fitting Example

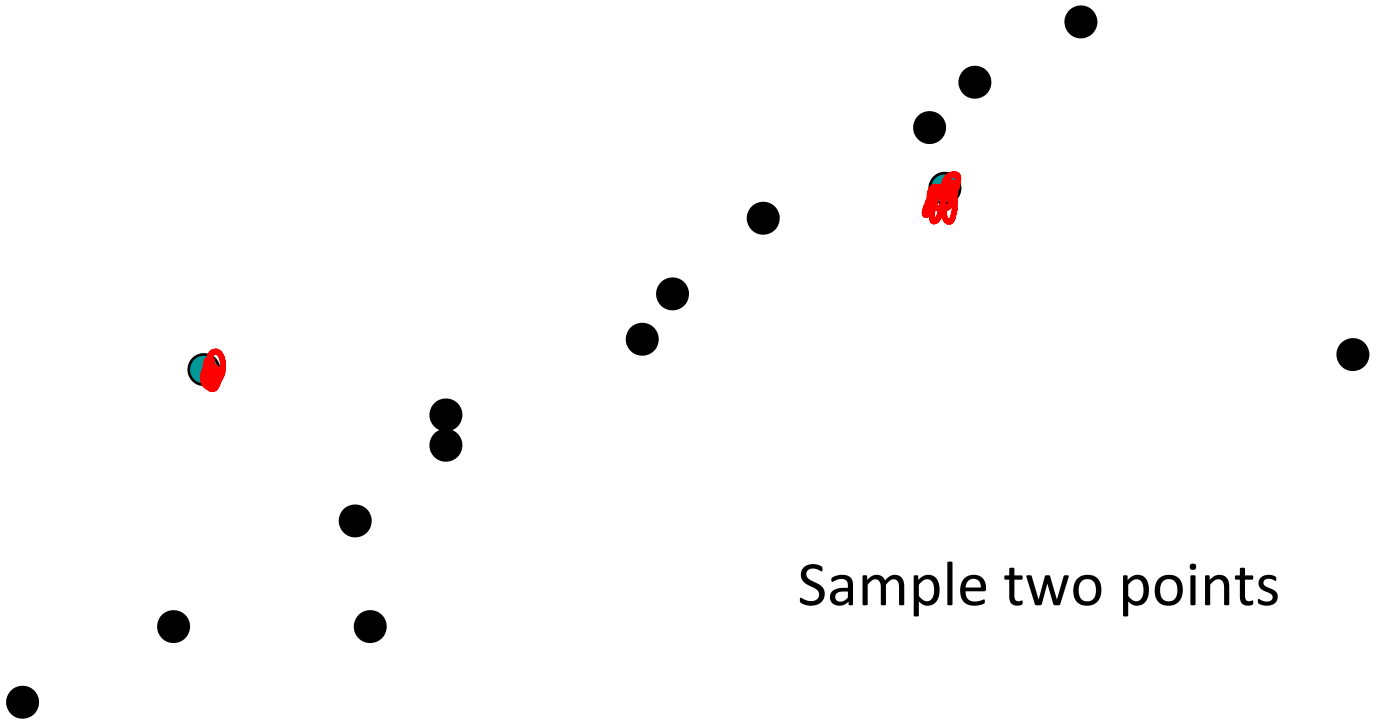
- Task: Estimate the best line
  - *How many points do we need to estimate the line?*



Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

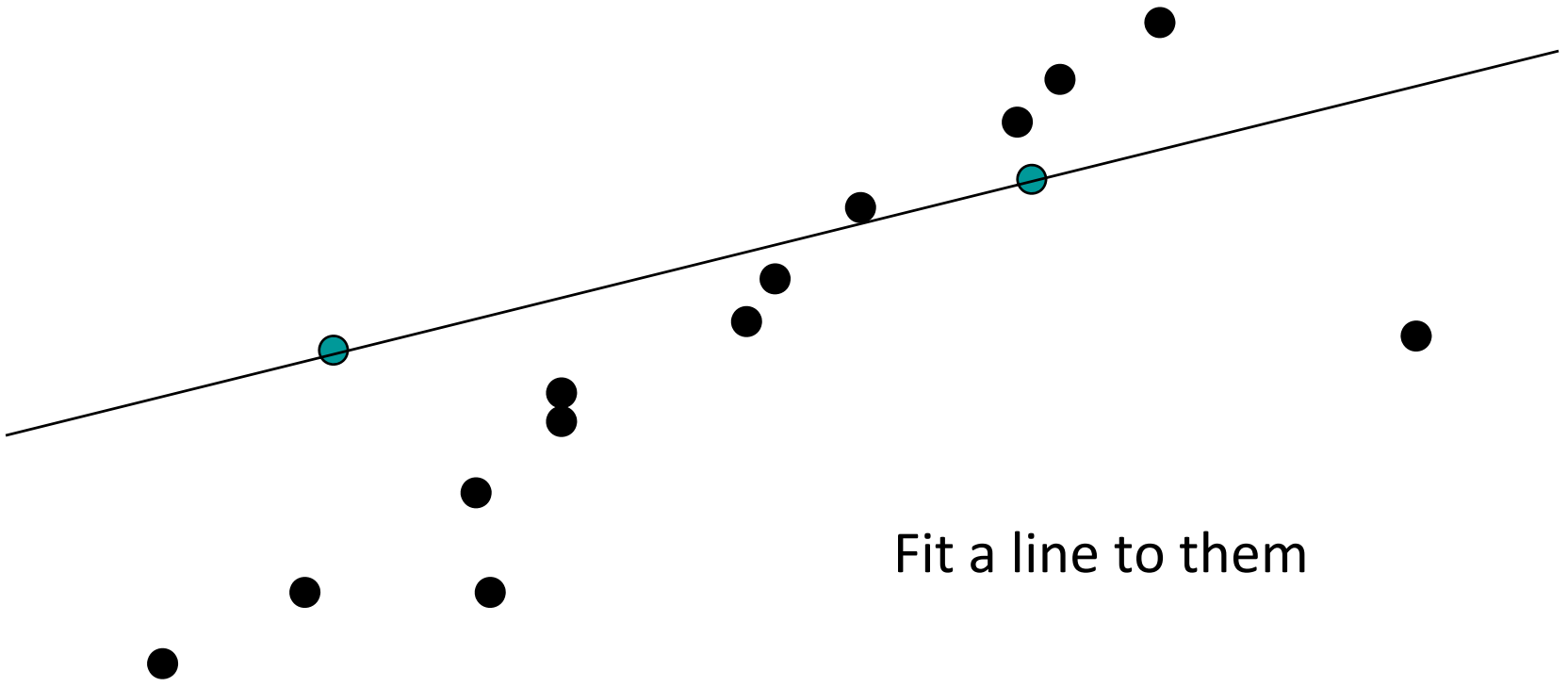
- Task: Estimate the best line



Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

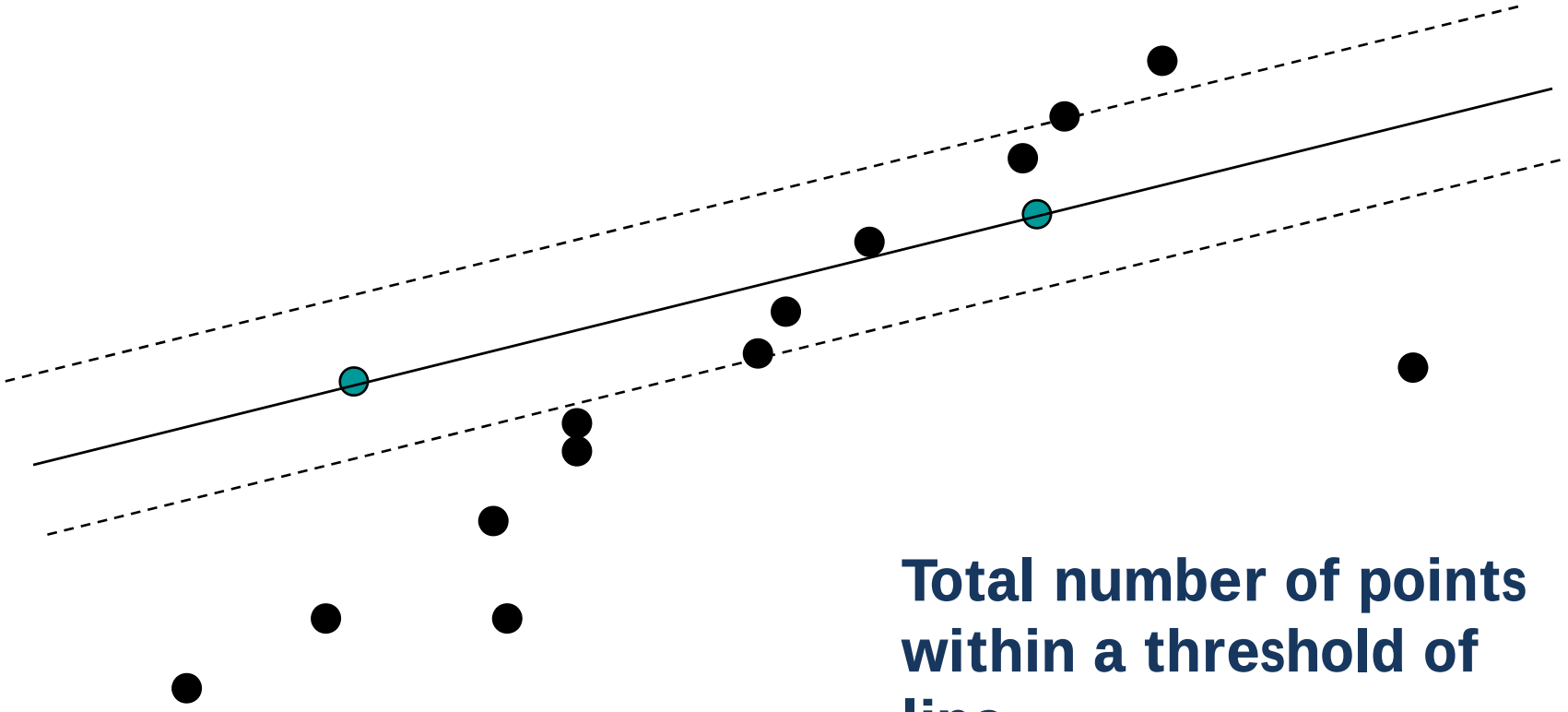
- Task: Estimate the best line



Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

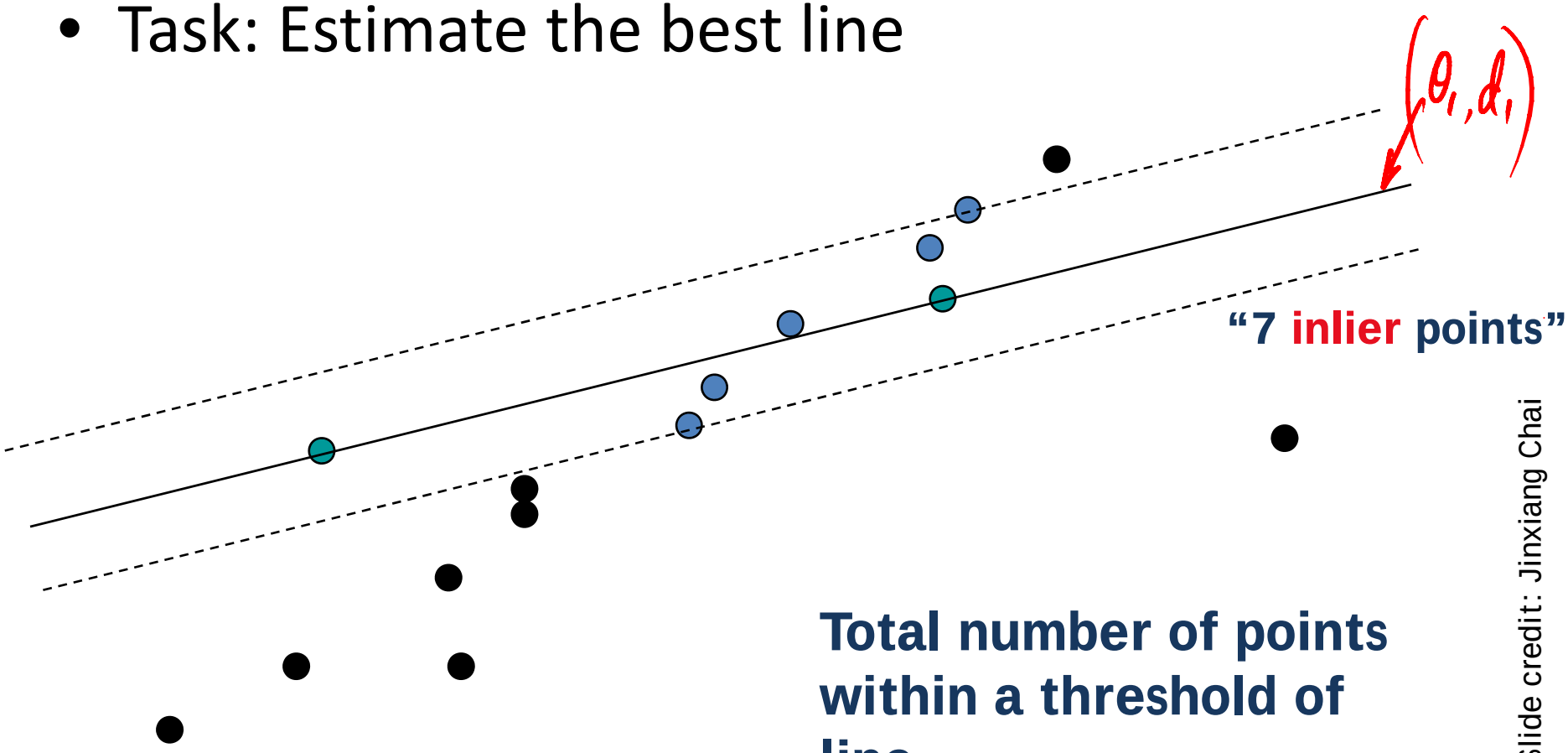


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

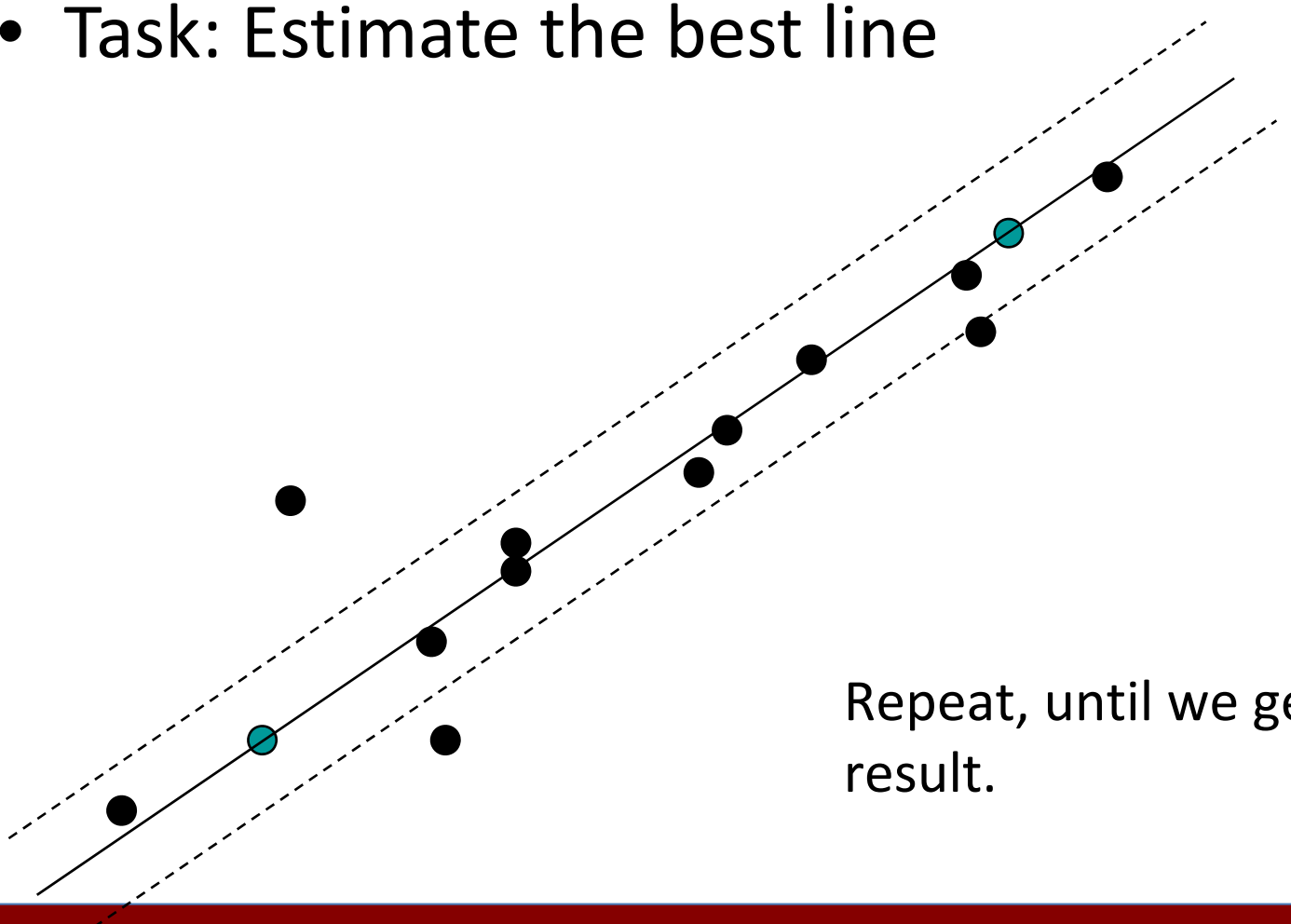


Total number of points within a threshold of line.

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line

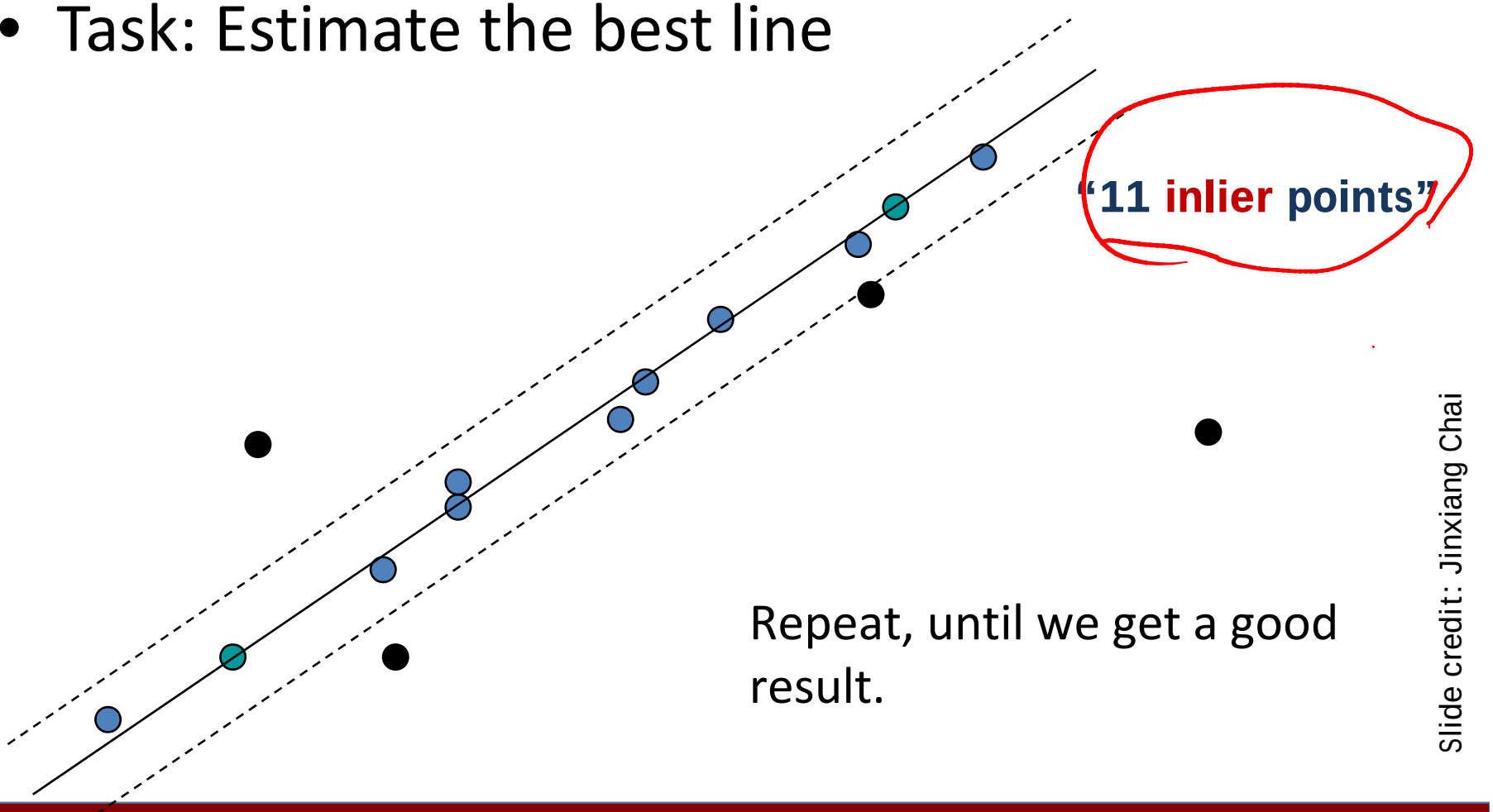


Repeat, until we get a good result.

Slide credit: Jinxiang Chai

RANSAC Line Fitting Example

- Task: Estimate the best line



Repeat, until we get a good result.

Slide credit: Jinxiang Chai

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
  1. Draw a sample of n points from the data
    uniformly and at random
  2. Fit to that set of n points
  3. 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
  4. 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: $(1-w^n)^k$

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

Slide credit: David Lowe

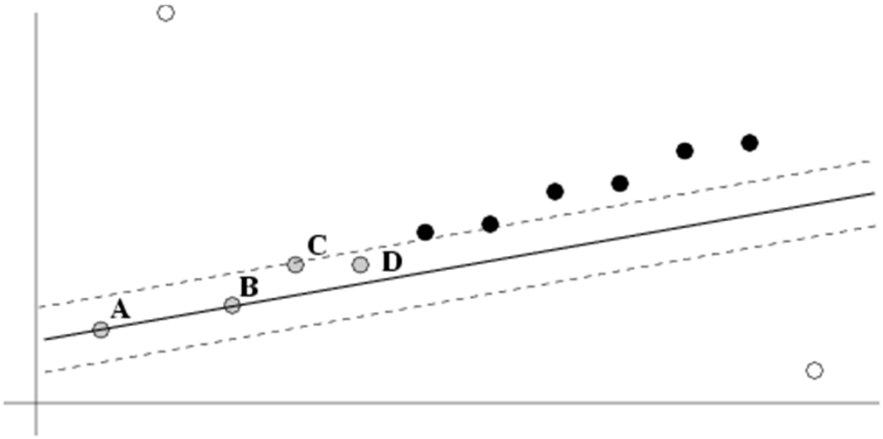
RANSAC: Computed k (p=0.99)

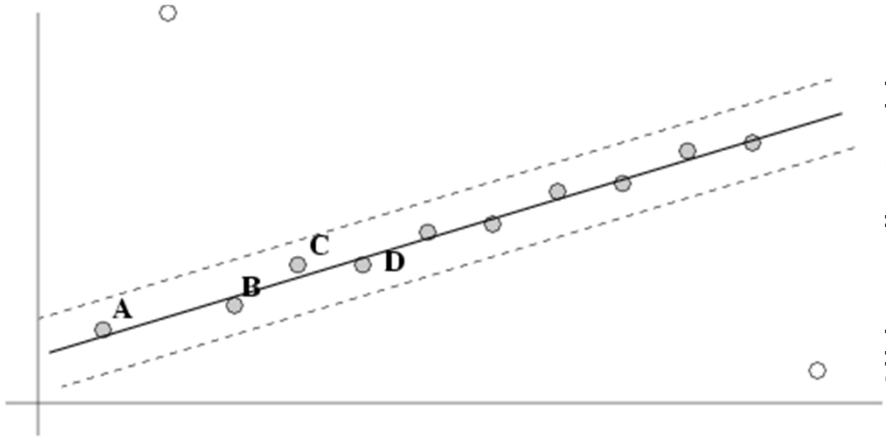
Sample size n	Proportion of outliers 5%	10%	20%	25%	30%	40%	50%
2	**2**	**3**	**5**	**6**	**7**	**11**	**17**
3	**3**	**4**	**7**	**9**	**11**	**19**	**35**
4	**3**	**5**	**9**	**13**	**17**	**34**	**72**
5	**4**	**6**	**12**	**17**	**26**	**57**	**146**
6	**4**	**7**	**16**	**24**	**37**	**97**	**293**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

Slide credit: David Lowe

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.





Slide credit: David Lowe

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

What we have learned today

- Edge detection
  - Canny edge detector
- Line fitting
  - Hough Transform
  - RANSAC (Problem Set 2 (Q5))

Supplementary materials

Generalized Hough Transform

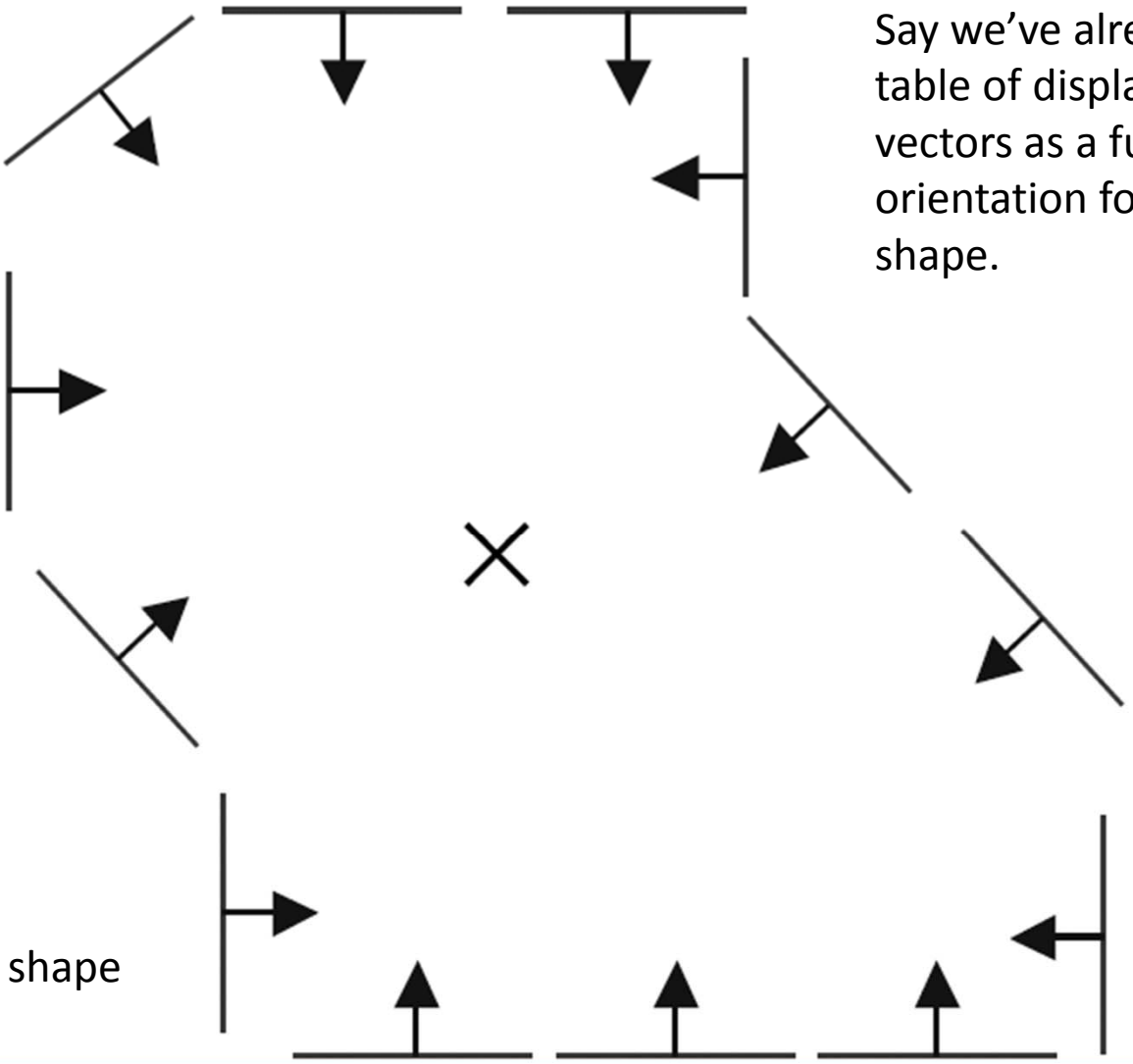
To *detect* the model shape in a new image:

- For each edge point
  - Index into table with its gradient orientation θ
  - Use retrieved 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.

Slide credit: Kristen Grauman

Example: Generalized Hough Transform

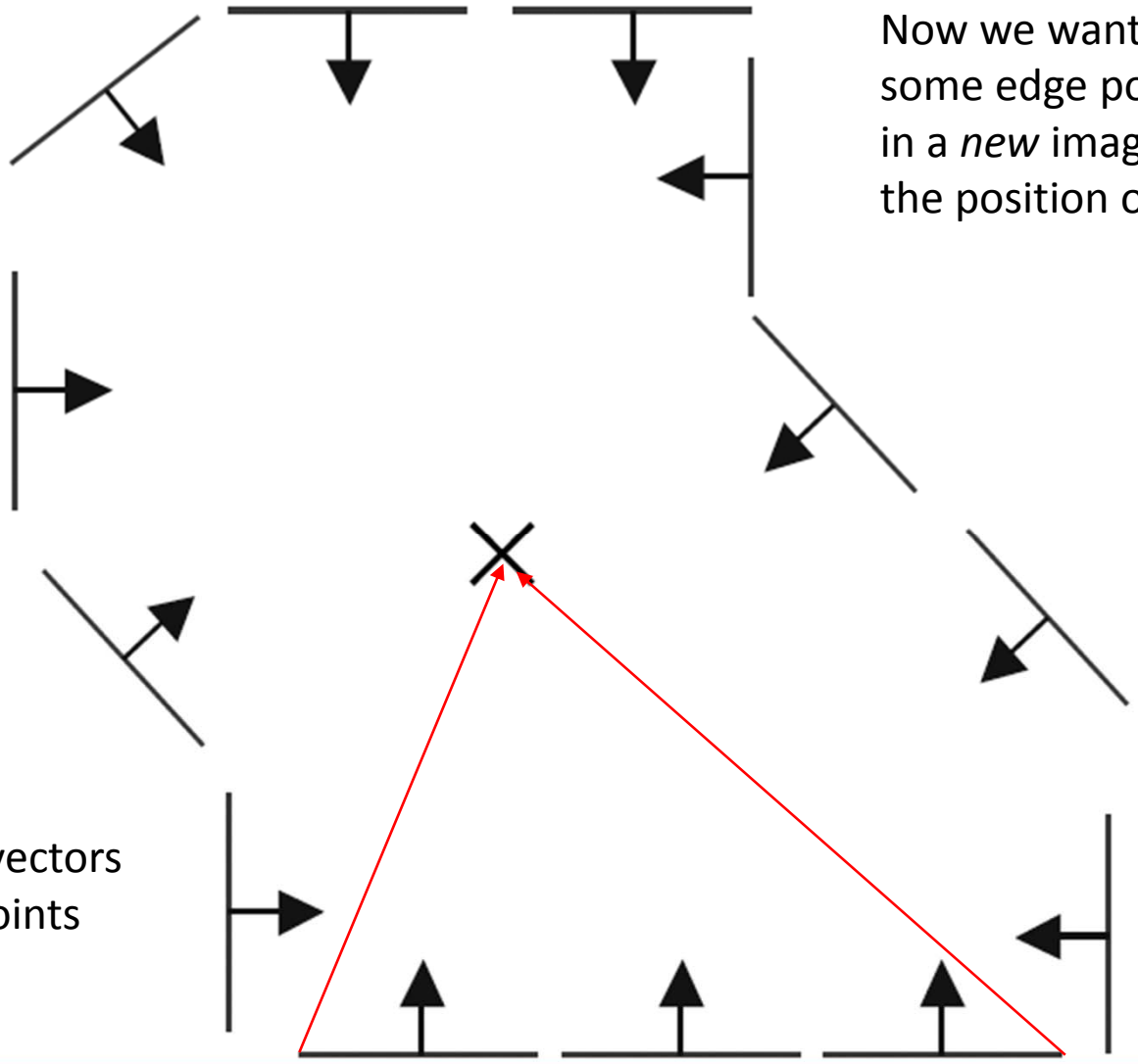


Model shape

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

Slide credit: Svetlana Lazebnik

Example: Generalized Hough Transform

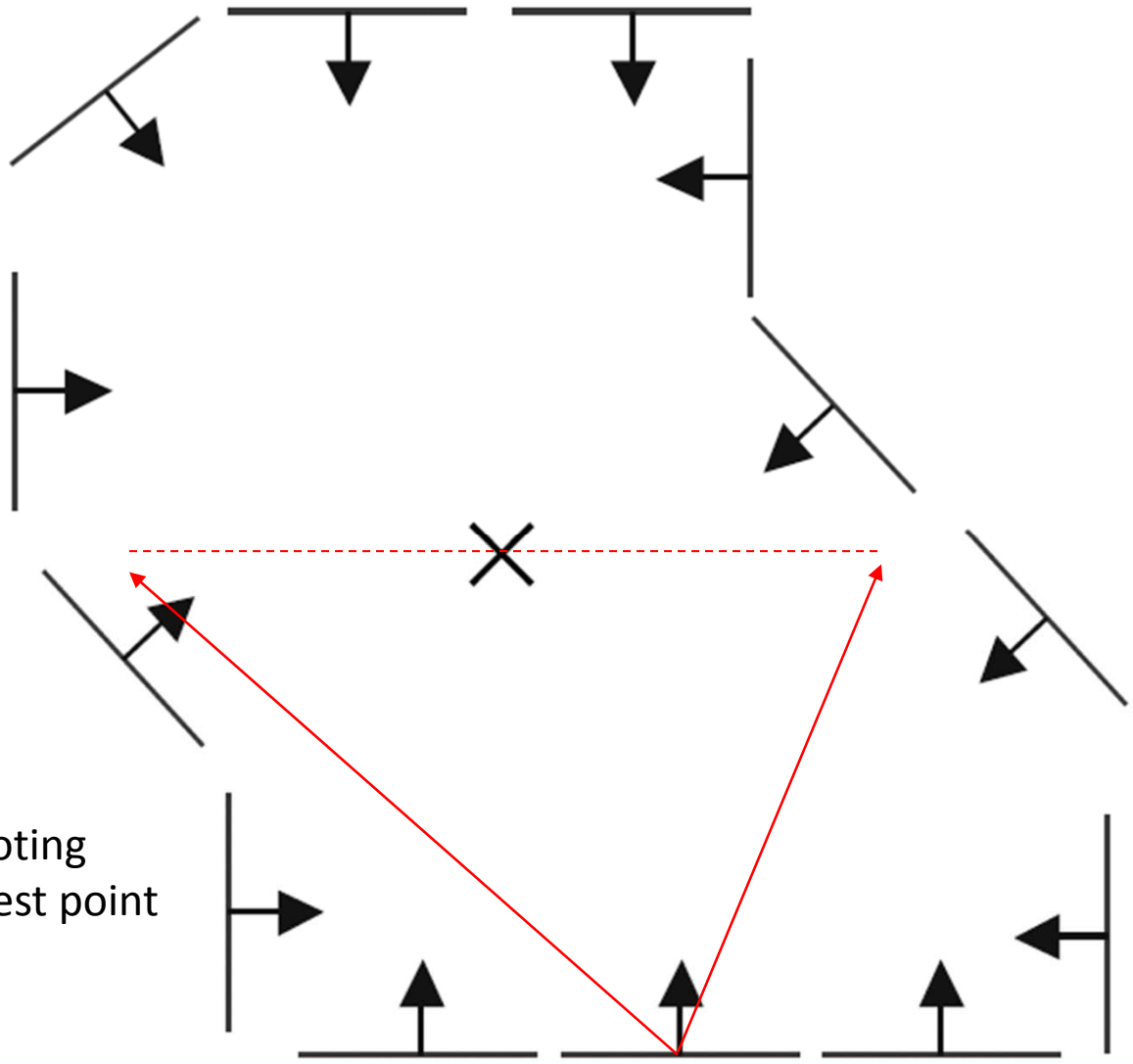


Now we want to look at some edge points detected in a *new* image, and vote on the position of that shape.

Displacement vectors for model points

Slide credit: Svetlana Lazebnik

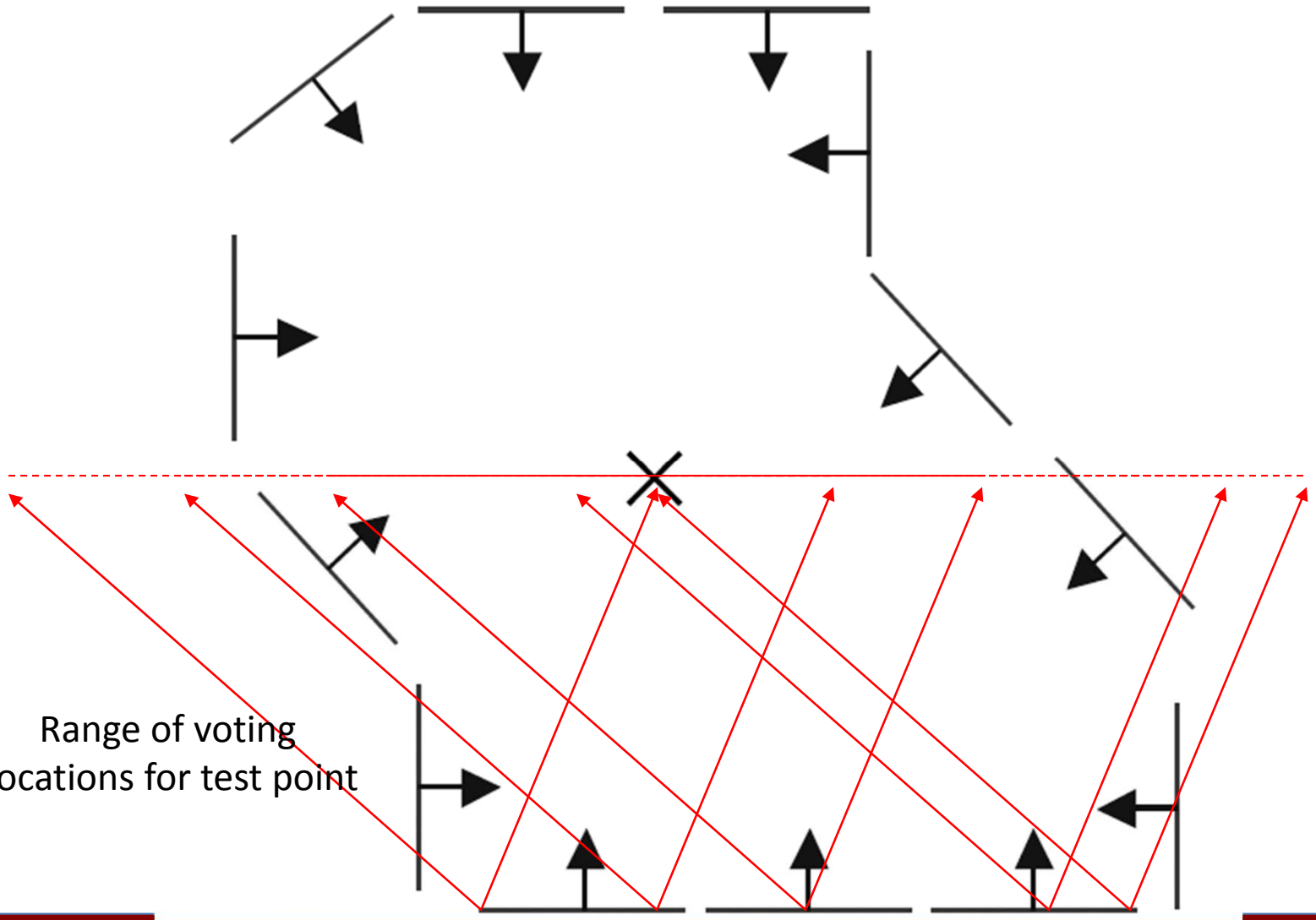
Example: Generalized Hough Transform



Range of voting locations for test point

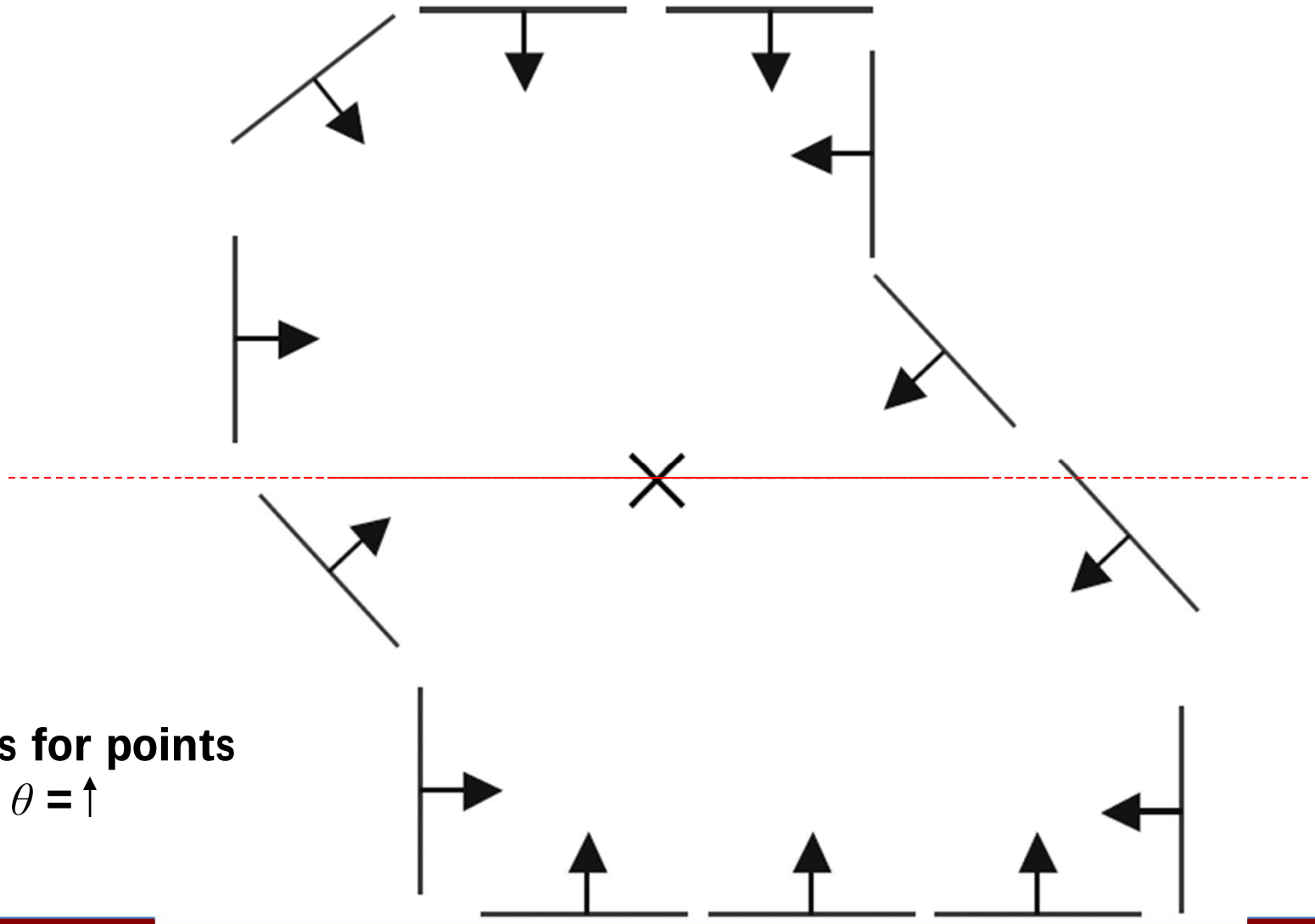
Slide credit: Svetlana Lazebnik

Example: Generalized Hough Transform



Slide credit: Svetlana Lazebnik

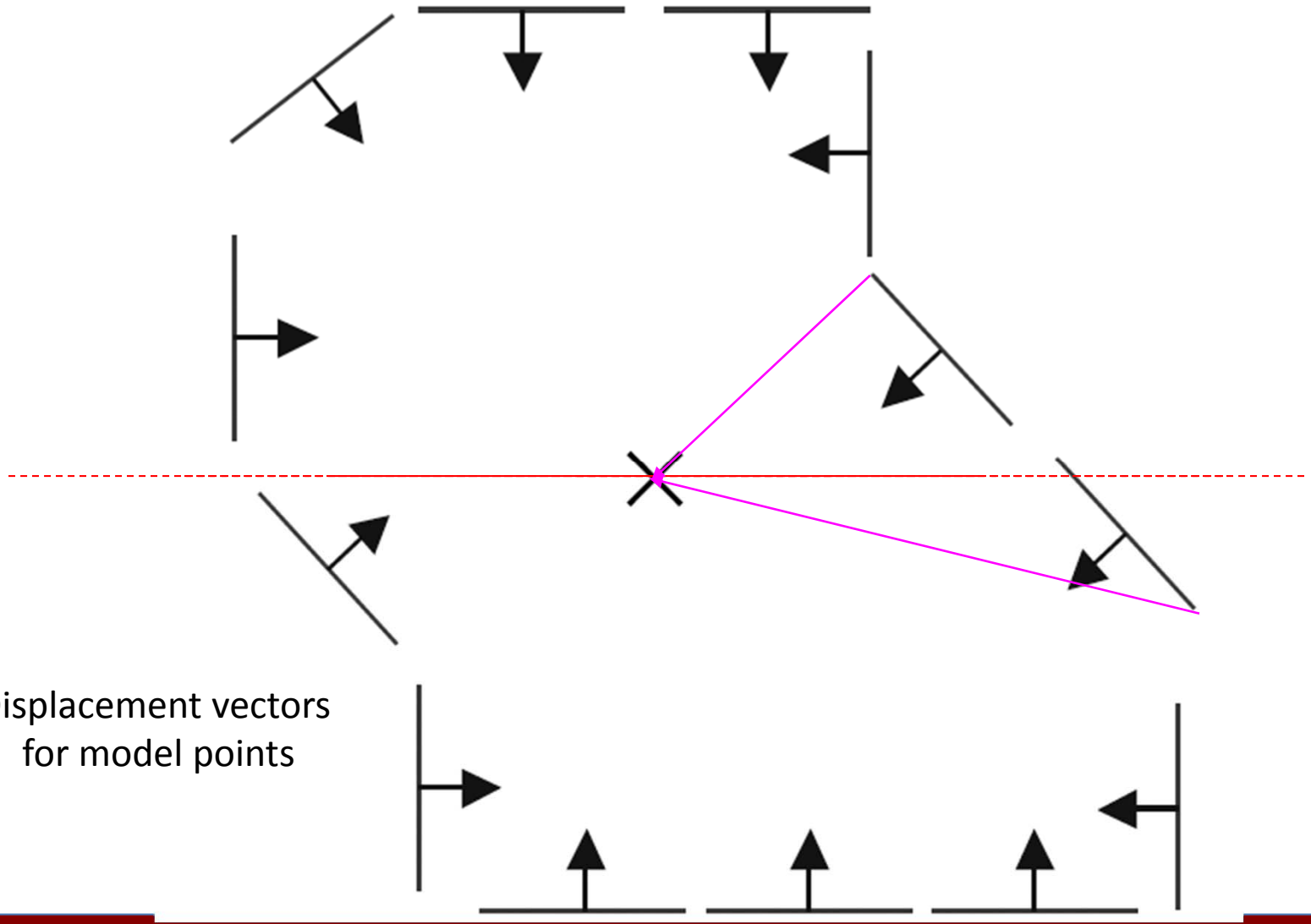
Example: Generalized Hough Transform



Votes for points with θ = ↑

Slide credit: Svetlana Lazebnik

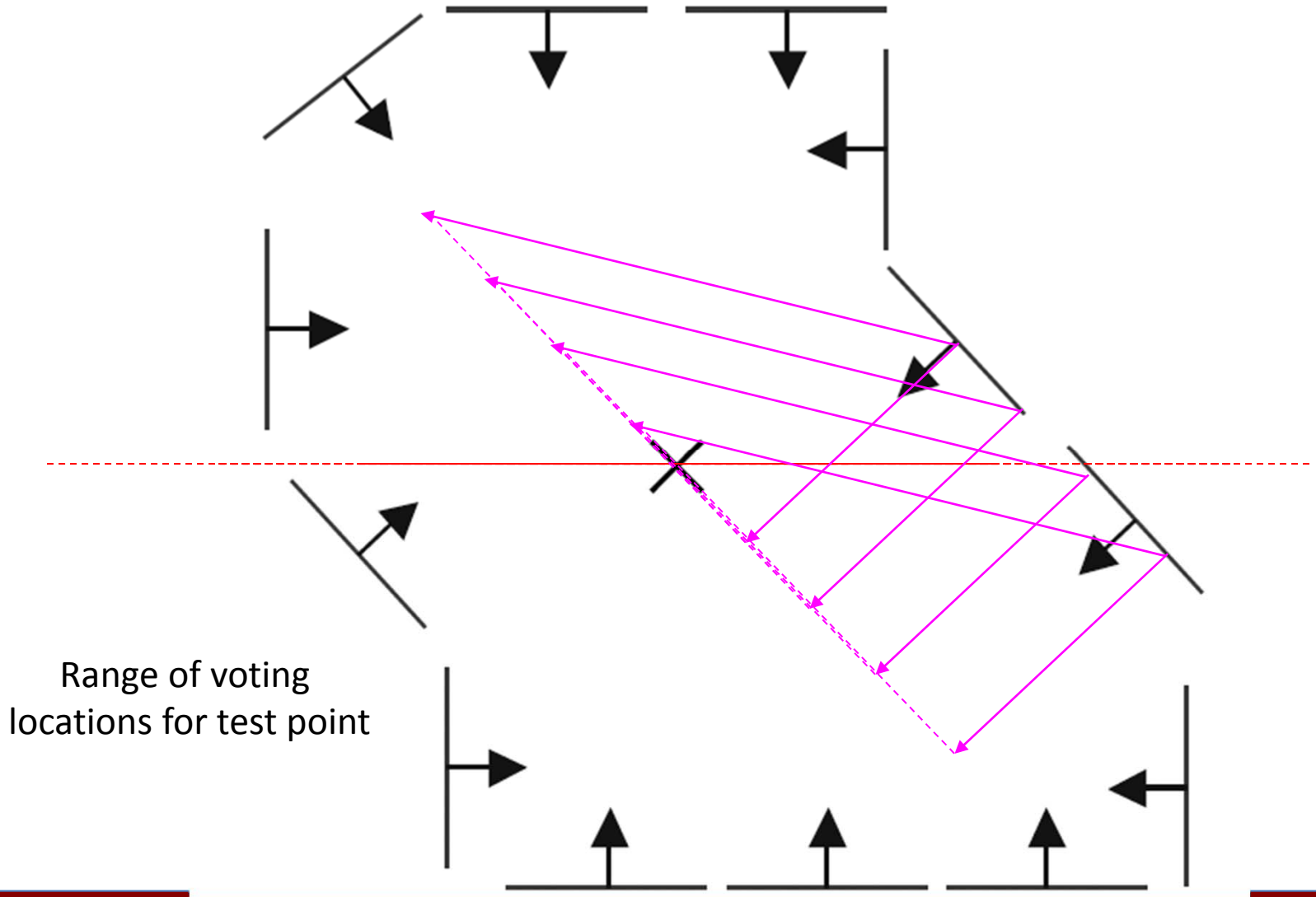
Example: Generalized Hough Transform



Displacement vectors for model points

Slide credit: Svetlana Lazebnik

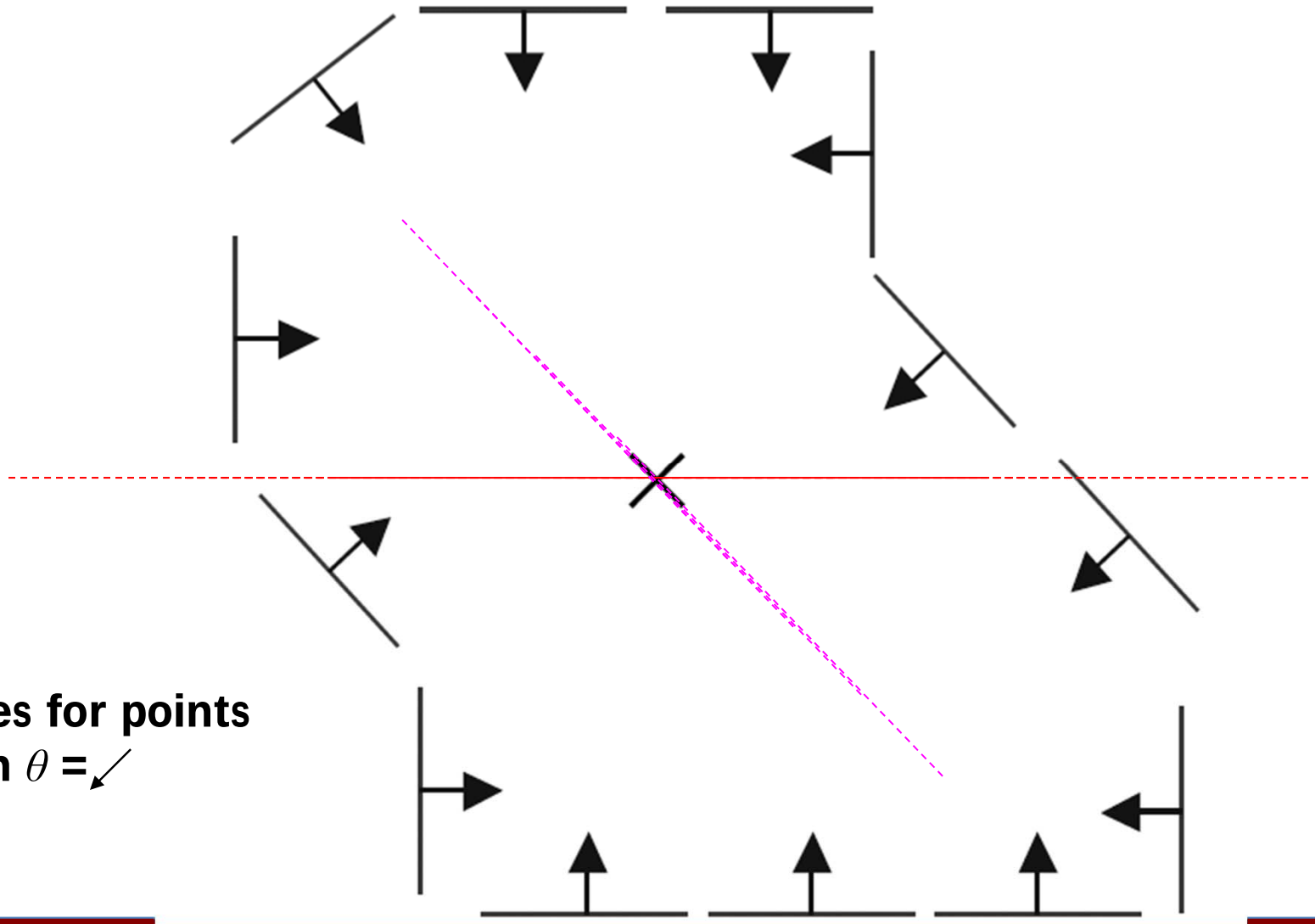
Example: Generalized Hough Transform



Range of voting locations for test point

Slide credit: Svetlana Lazebnik

Example: Generalized Hough Transform

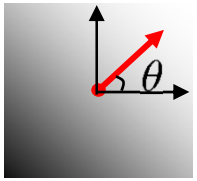


Votes for points with θ =↙

Slide credit: Svetlana Lazebnik

Extensions to Hough Transform

Extensions



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

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

Extension 1: Use the image gradient

1. same
2. for each edge point $\mathrm{I}[x,y]$ in the image

   θ = gradient at (x,y)

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

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

(Reduces degrees of freedom)

Slide credit: Kristen Grauman

Extensions

Extension 1: Use the image gradient

1. same
2. for each edge point $\mathrm{I}[x,y]$ in the image

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

   $\mathrm{H}[d,\theta]$ += 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…

Slide credit: Kristen Grauman

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

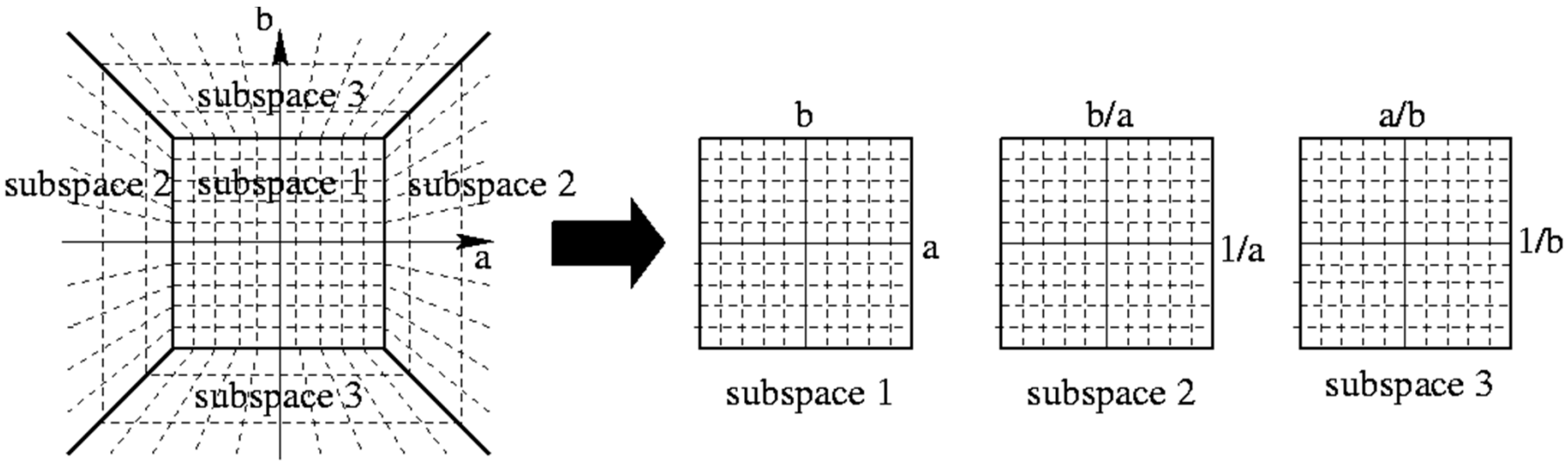
T. Tuytelaars, M. Proesmans, L. Van Gool *"The cascaded Hough transform", ICIP'97.*

Slide credit: Svetlana Lazebnik

Finding Vanishing Points

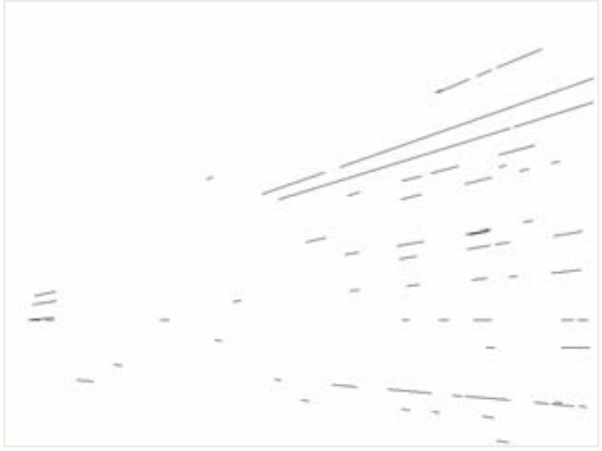
Raphael, *The School of Athens*, 1511

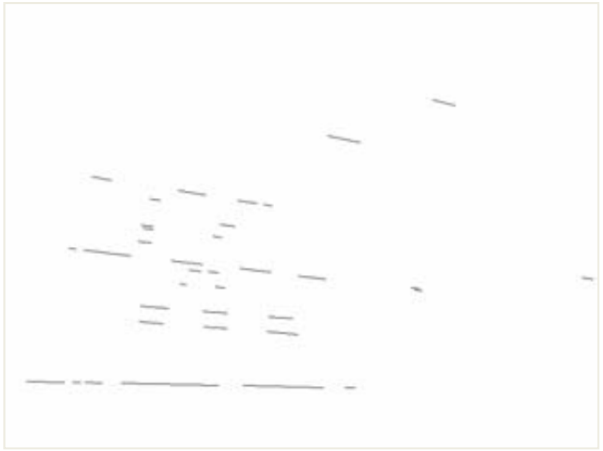
Cascaded Hough Transform



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

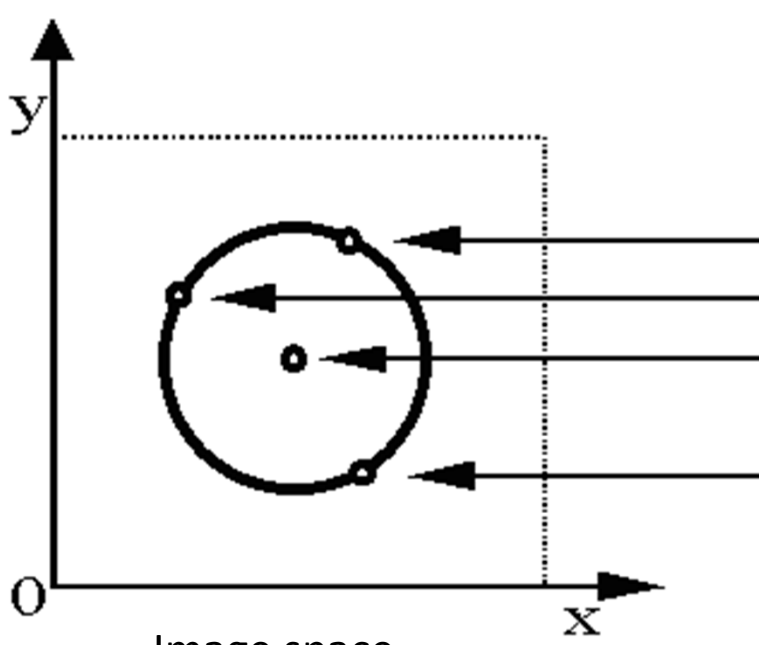
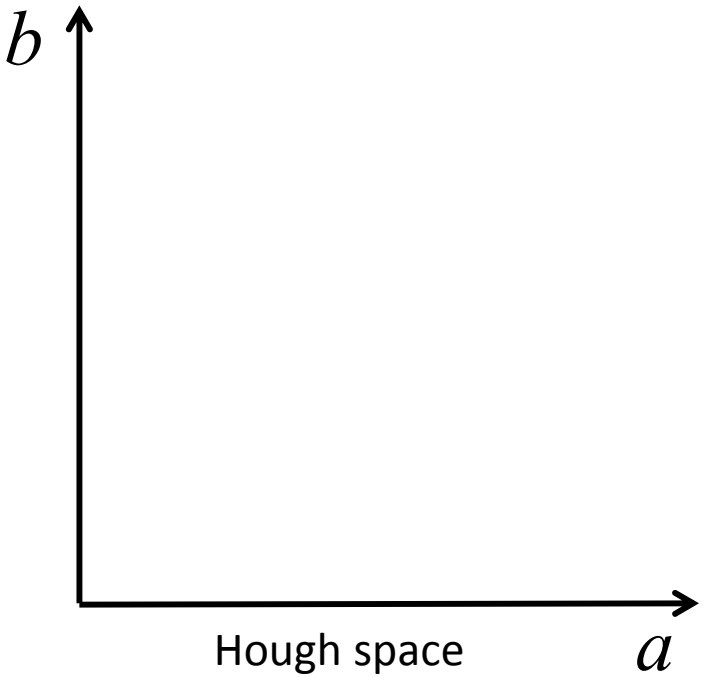


Image space



Hough space

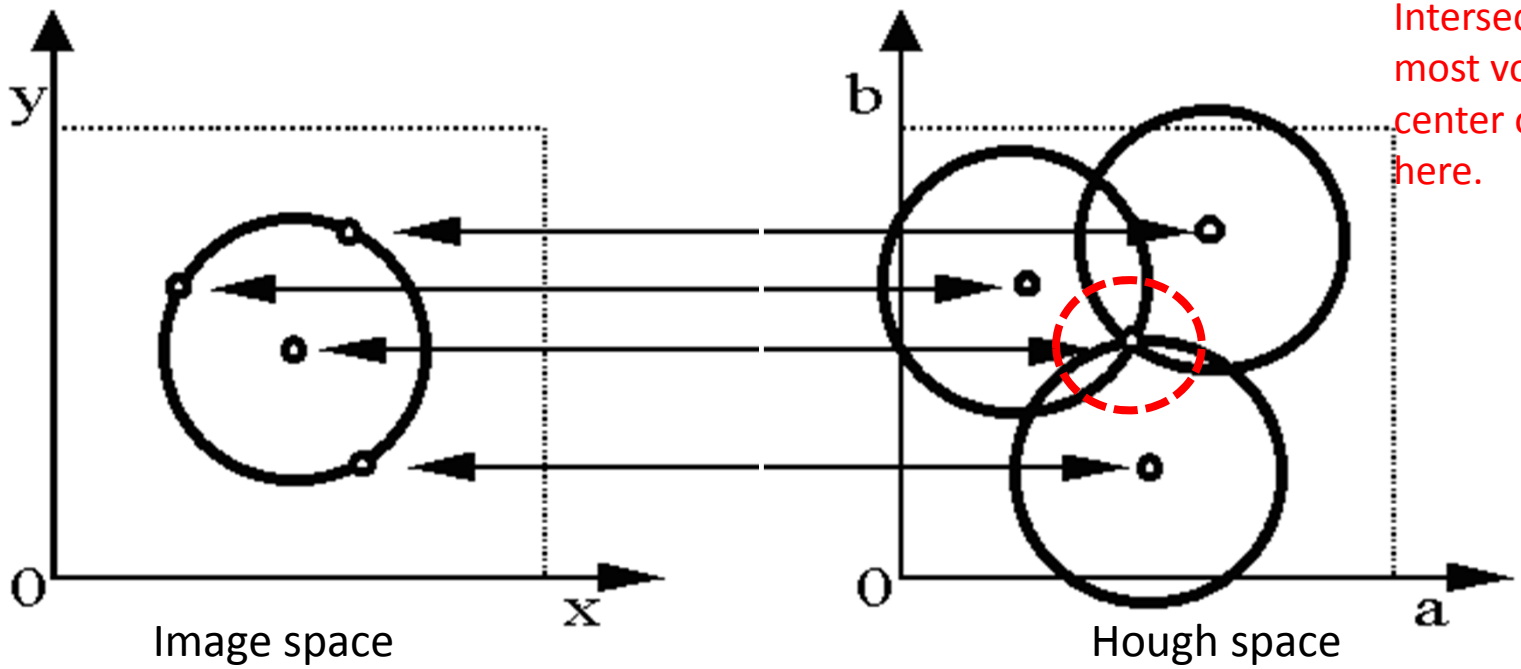
Slide credit: Kristen Grauman

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



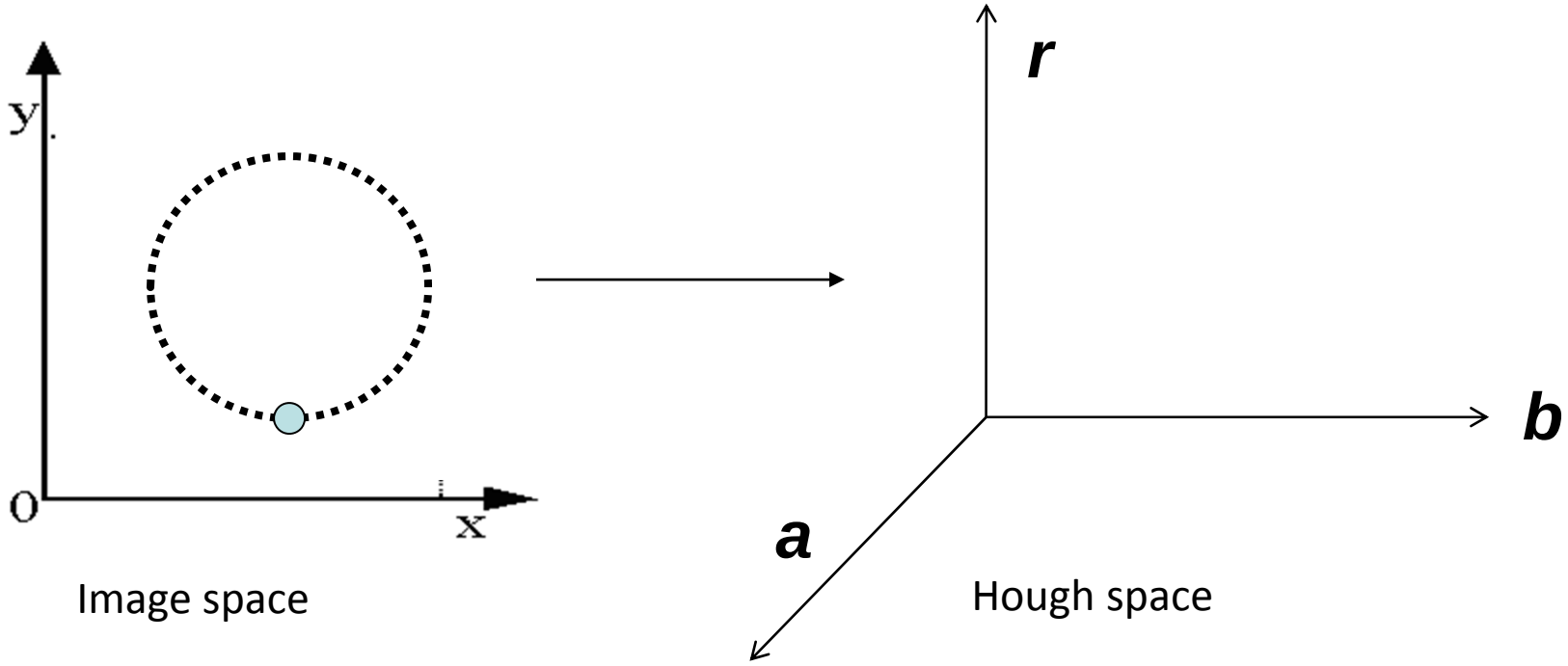
Slide credit: Kristen Grauman

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



Slide credit: Kristen Grauman

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

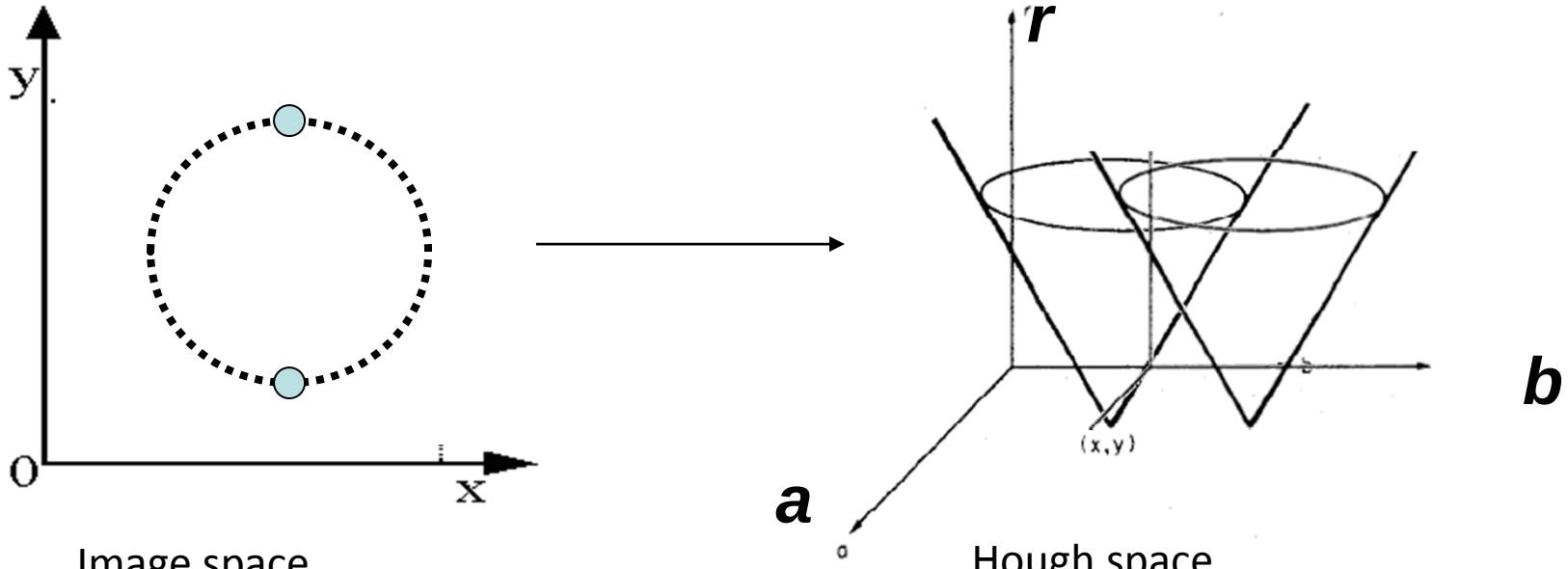


Image space

Hough space

Slide credit: Kristen Grauman

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

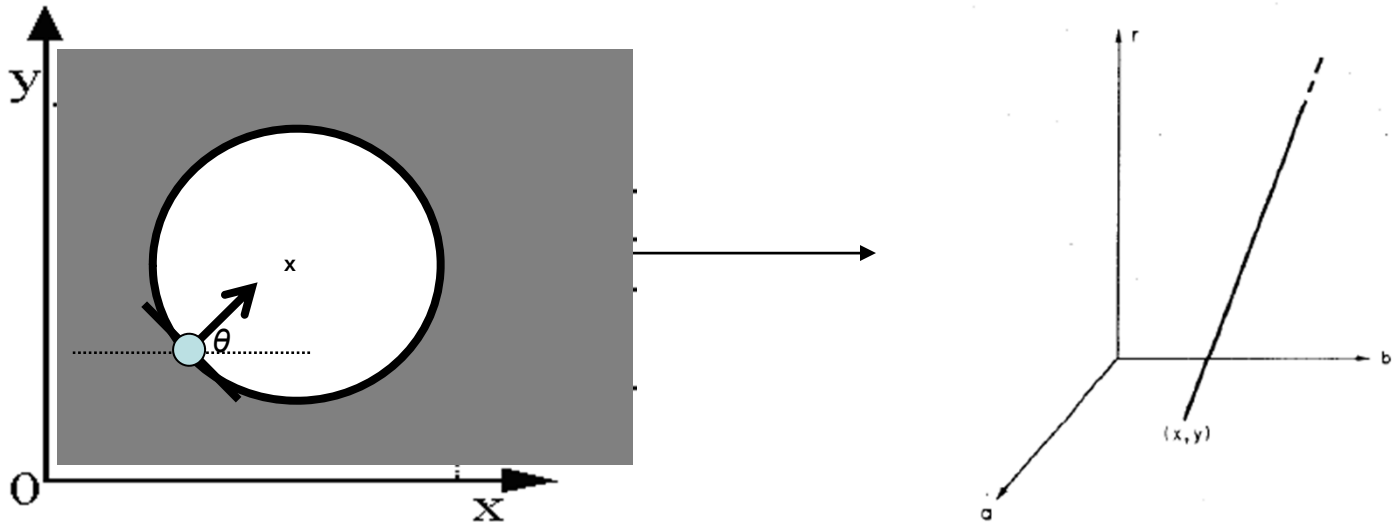


Image space

Hough space

Slide credit: Kristen Grauman

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

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

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

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

end

end

Slide credit: Kristen Grauman

Example: Detecting Circles with Hough

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

Slide credit: Kristen Grauman