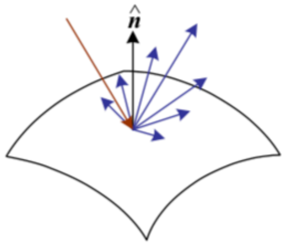
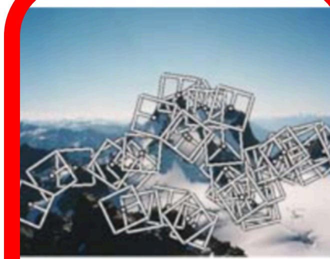
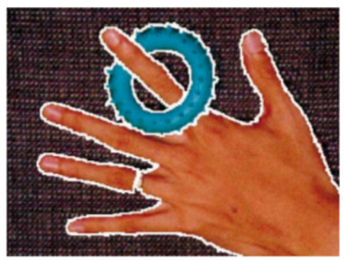


2. Image Formation

3. Image Processing

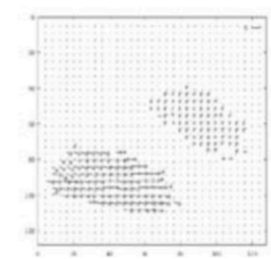


4. Features

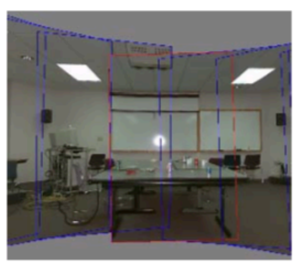


5. Segmentation

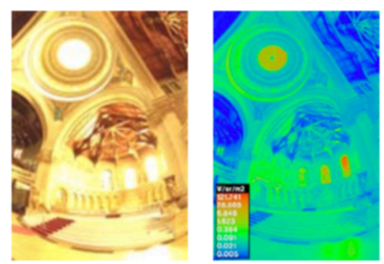
6-7. Structure from Motion



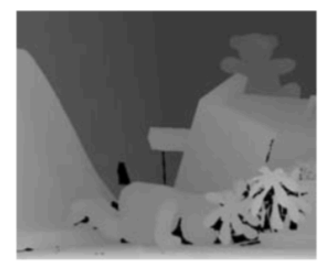
8. Motion



9. Stitching

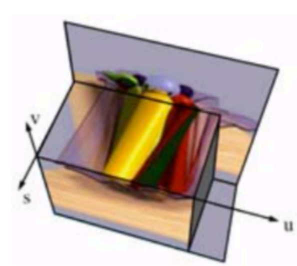


10. Computational Photography



11. Stereo

12. 3D Shape



13. Image-based Rendering

14. Recognition

Edge detection

- **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
- **Why?**

Figure from D. Lowe

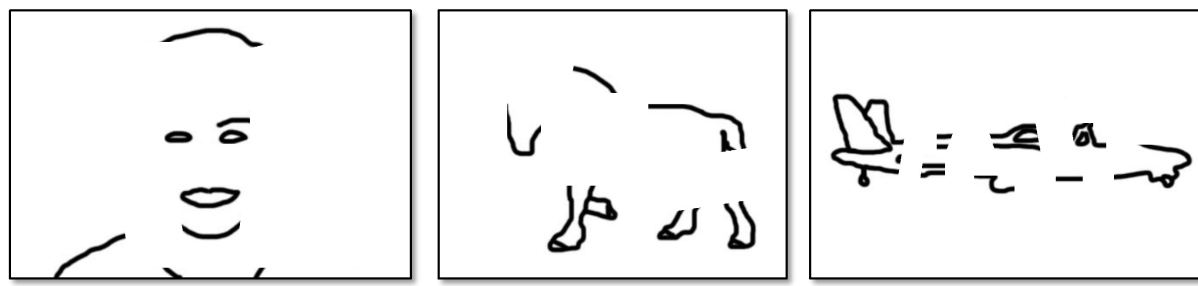


Figure from J. Shotton et al., PAMI 2007

- **Main idea**: look for strong **gradients**, post-process

Slide credit: Kristen Grauman

Gradients → edges

Primary edge detection steps:

1. Smoothing: suppress noise
2. Edge enhancement: filter for contrast
3. Edge localization

   Determine which local maxima from filter output are actually edges vs. noise

   - Threshold, Thin

Slide credit: Kristen Grauman

Thresholding

- Choose a threshold value t
- Set any pixels less than t to zero (off)
- Set any pixels greater than or equal to t to one (on)

Slide credit: Kristen Grauman

Original image

Slide credit: Kristen

Gradient magnitude image

Slide credit: Kristen Grauman

Thresholding gradient with a lower threshold

Slide credit: Kristen Grauman

Thresholding gradient with a higher threshold

Slide credit: Kristen Grauman

Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin wide “ridges” down to single pixel width
- **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’);`**
- **`>>help edge`**

Slide credit: David Lowe, Fei-Fei Li

The Canny edge detector

original image (Lena)

Slide credit: Steve Seitz

The Canny edge detector

norm of the gradient

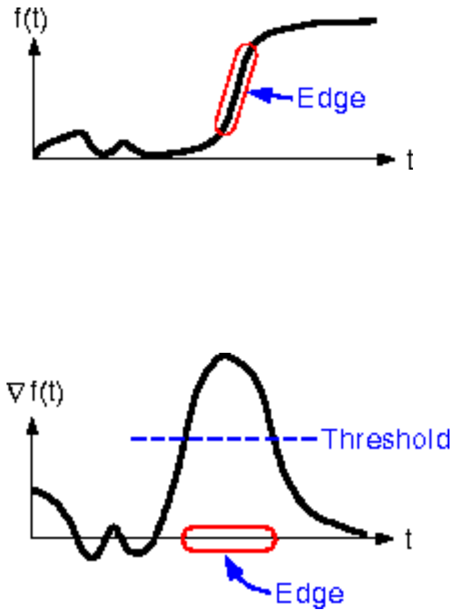
Slide credit: Kristen

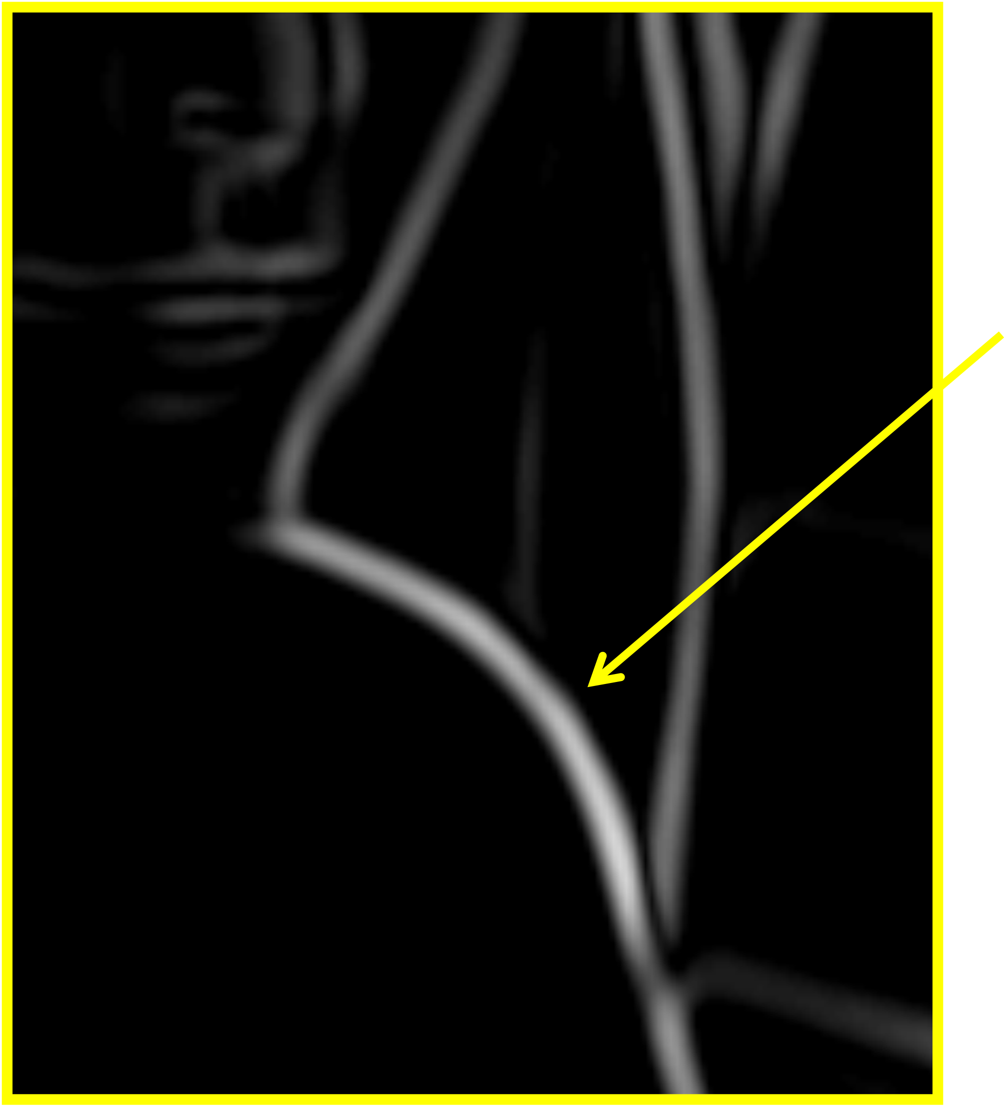
The Canny edge detector

thresholding

Slide credit: Kristen

The Canny edge detector

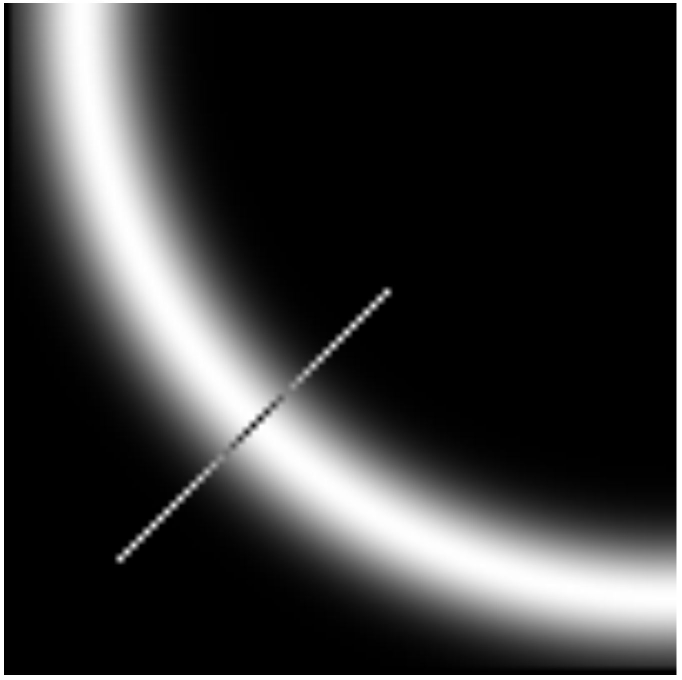


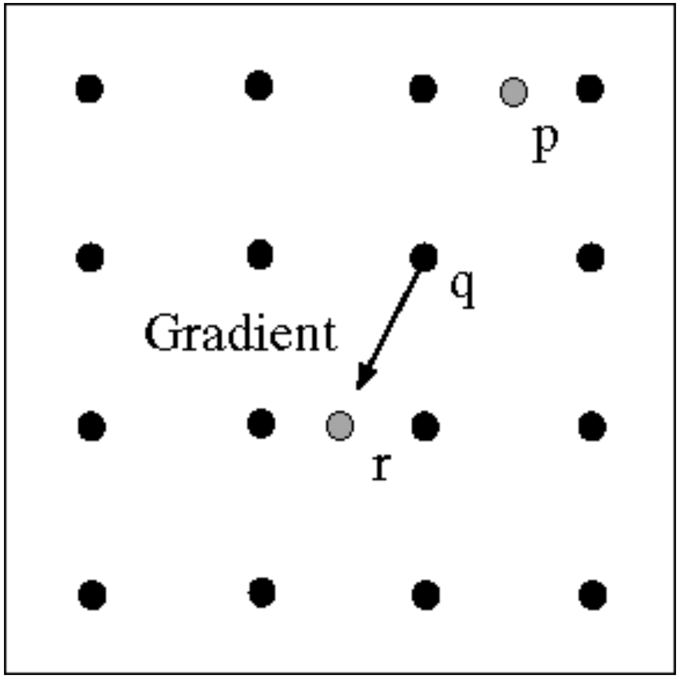


How to turn these thick regions of the gradient into curves?

Slide credit: Kristen

Non-maximum suppression





Check if pixel is local maximum along gradient direction

Select single max across width of the edge

Requires checking interpolated pixels p and r

Slide credit: Kristen Grauman

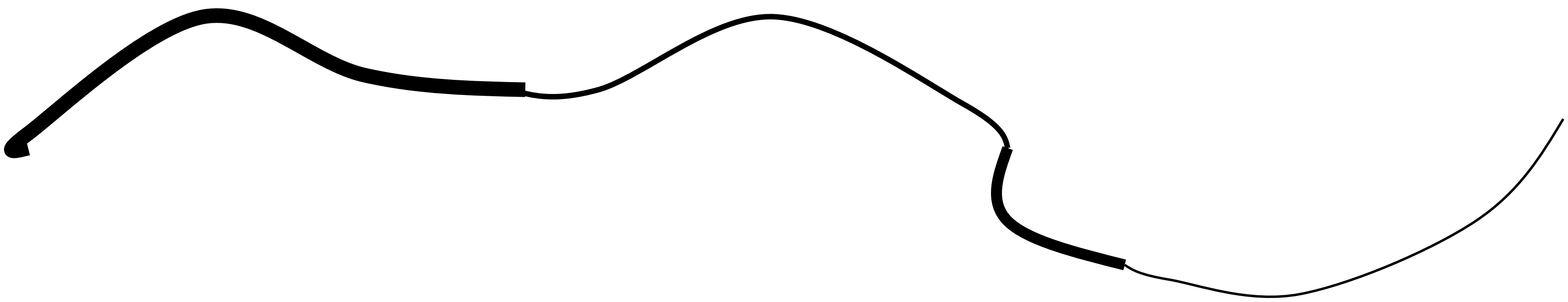
The Canny edge detector

Problem: pixels along this edge didn't survive the thresholding

thinning
(non-maximum suppression)

Hysteresis thresholding

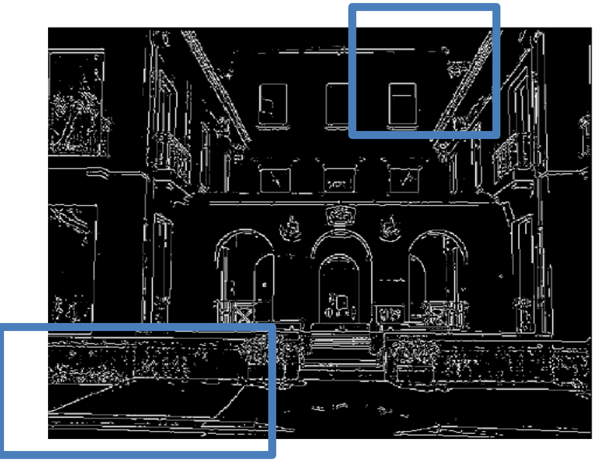
- Use a high threshold to start edge curves, and a low threshold to continue them.



Slide credit: Steve Seitz

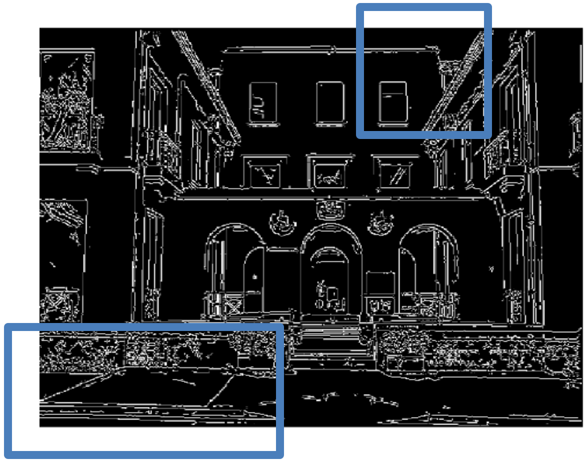
Hysteresis thresholding

original image



high threshold
(strong edges)

low threshold
(weak edges)



hysteresis threshold

Slide credit: Fei-Fei Li

Recap: Canny edge detector

- Filter image with derivative of Gaussian
- Find magnitude and orientation of gradient
- **Non-maximum suppression**:
  - Thin wide "ridges" down to single pixel width
- **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');`**
- **`>>help edge`**

Slide credit: David Lowe, Fei-Fei Li

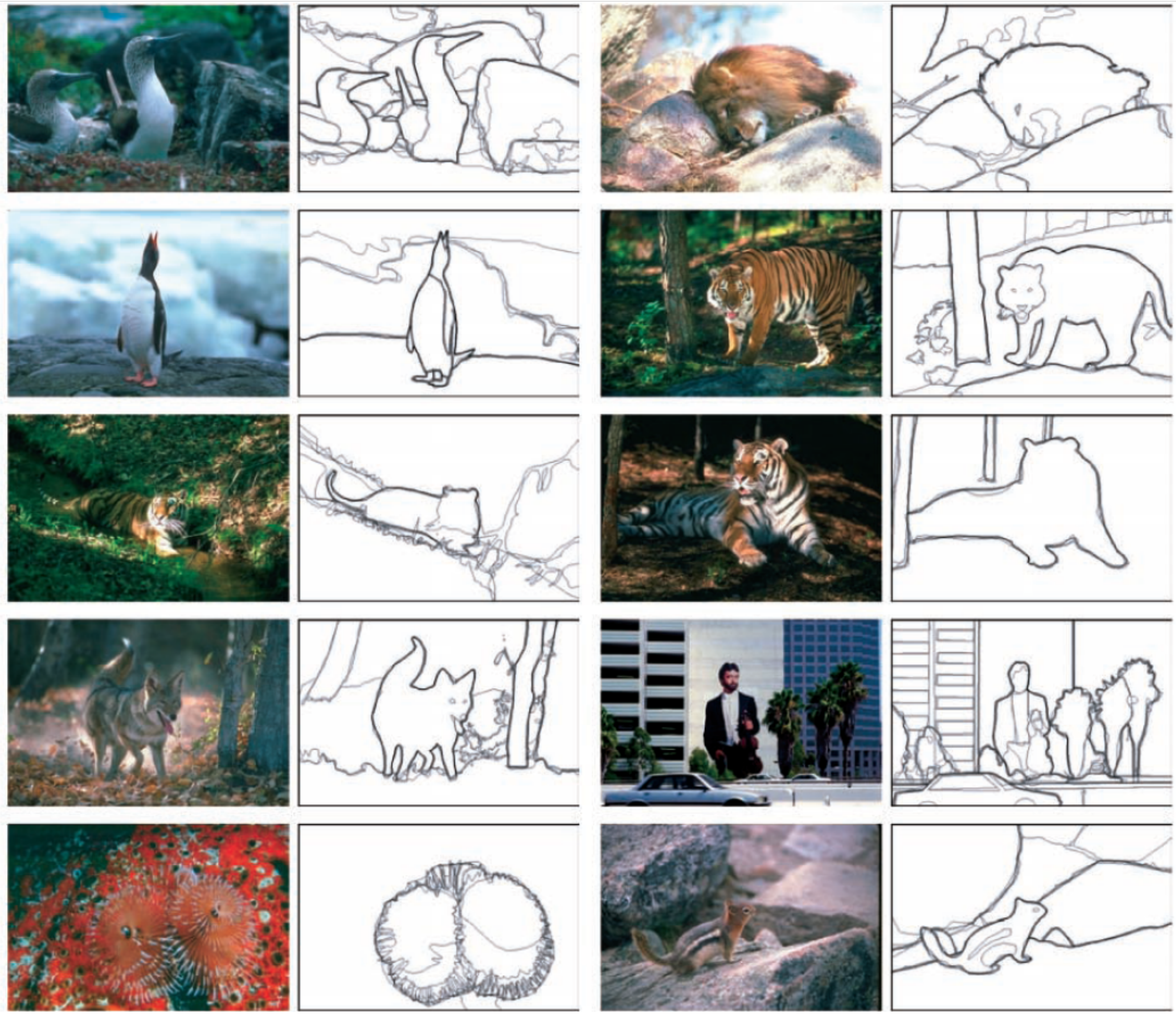
Low-level edges vs. perceived contours

image **human segmentation** **gradient magnitude**

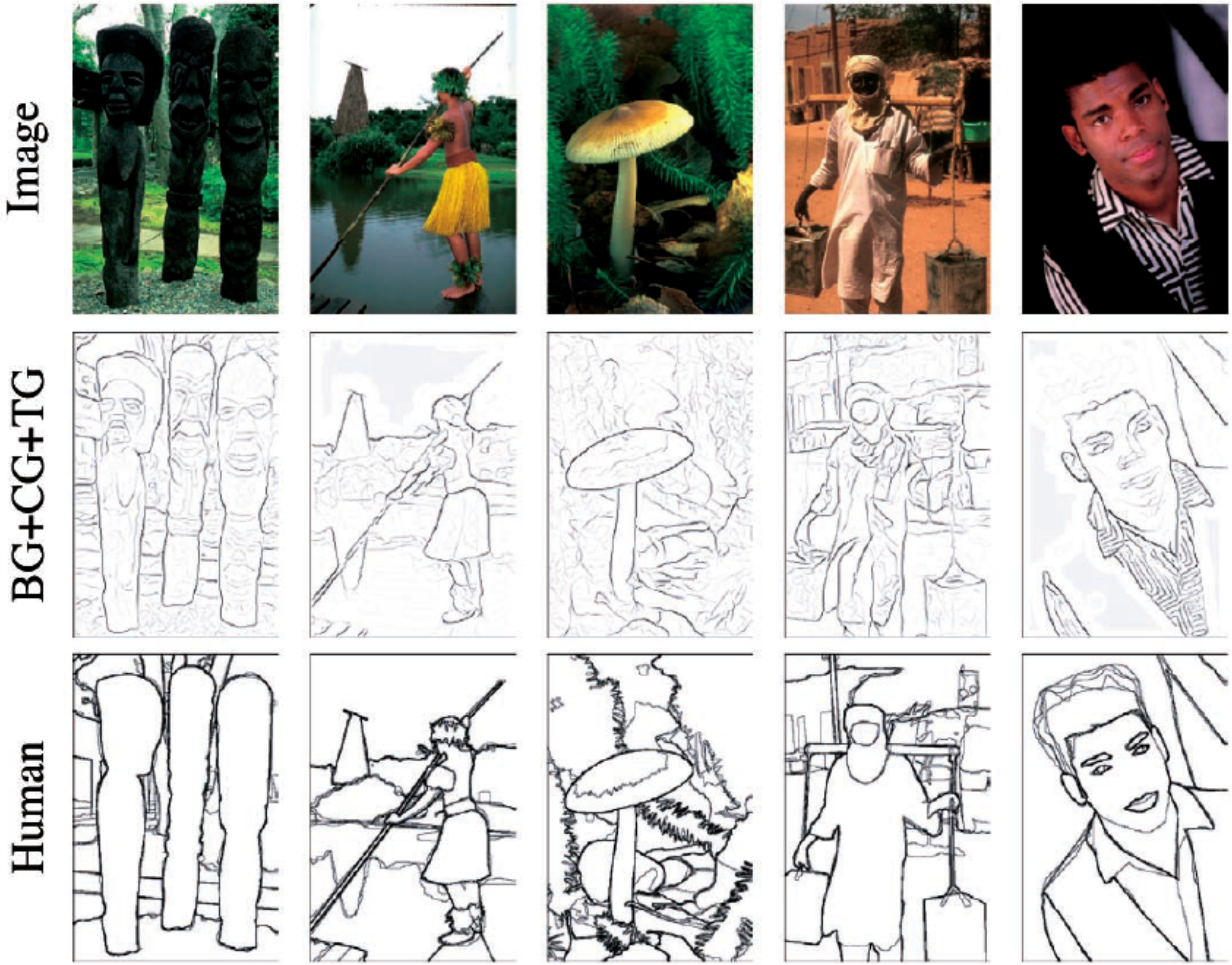
- Berkeley segmentation database: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Learn from humans which combination of features is most indicative of a “good” contour?

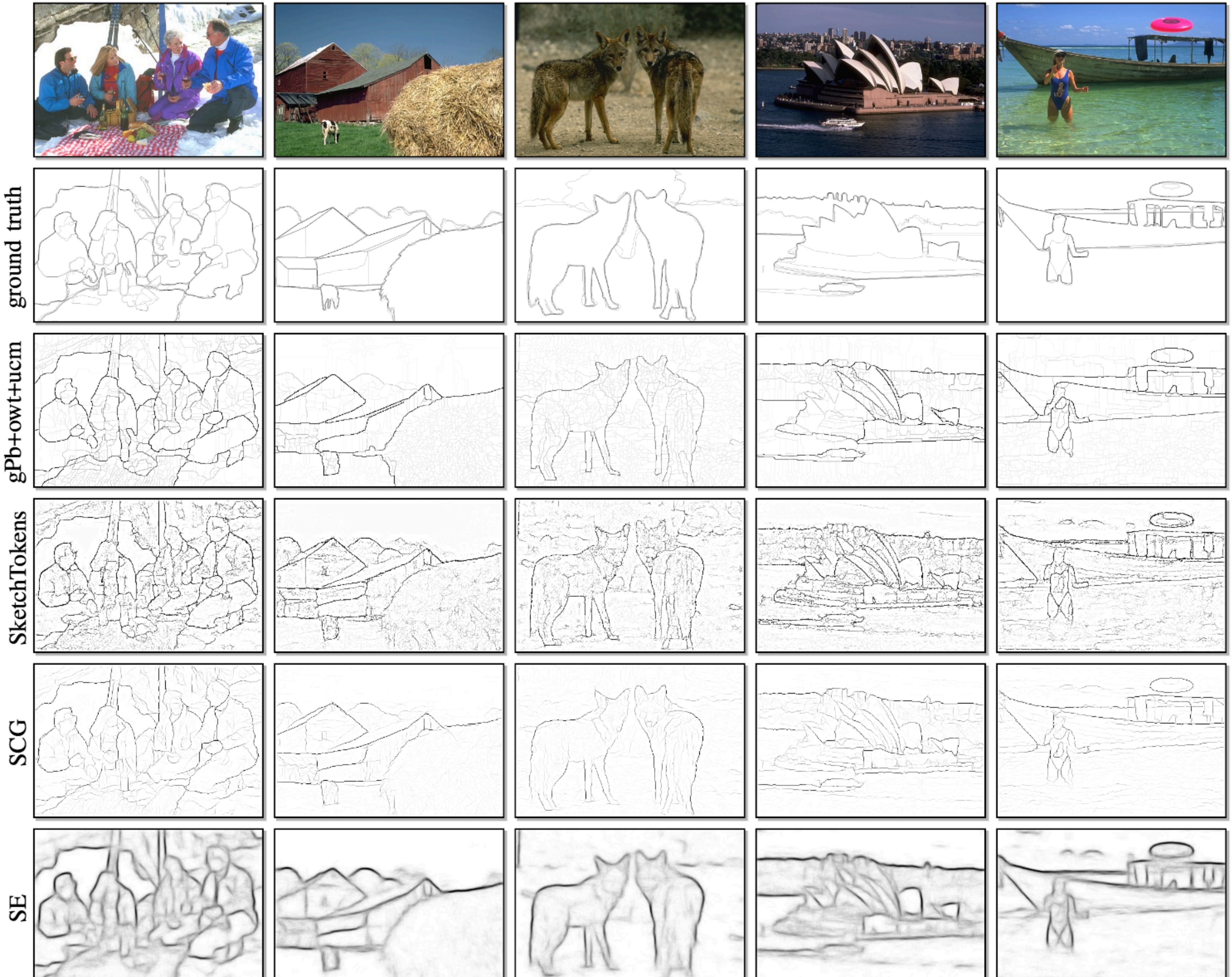
[D. Martin et al. PAMI 2004]



Human-marked segment boundaries



[D. Martin et al. PAMI 2004]



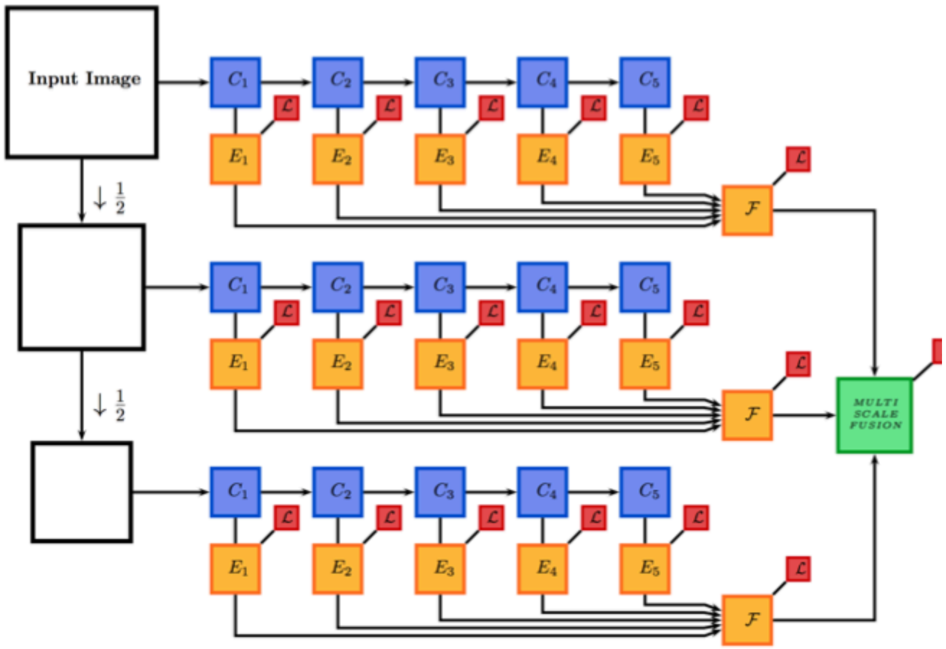
Slide credit: Devi Parikh

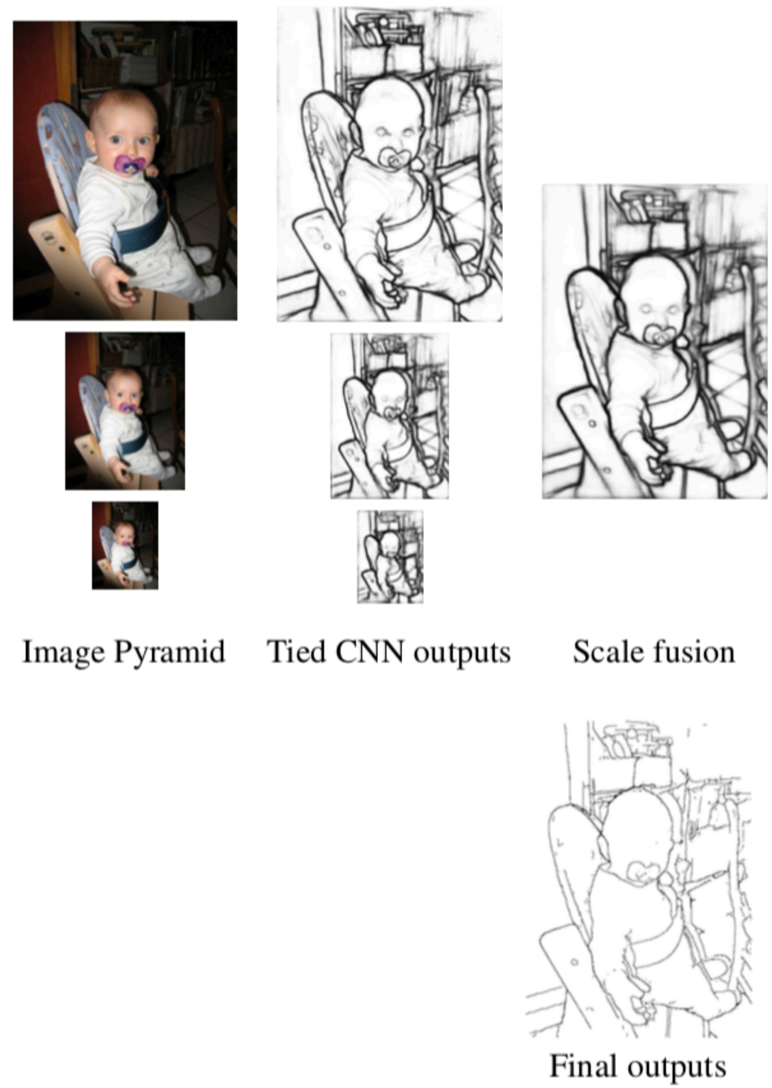
Figure from: Dollar and Zitnick, PAMI 2015

Pushing the Boundaries of Boundary Detection using Deep Learning

ICLR 2016

Iasonas Kokkinos





Richer Convolutional Features for Edge Detection

Yun Liu[1] Ming-Ming Cheng[1] Xiaowei Hu[1] Kai Wang[1] Xiang Bai[2]
[1]Nankai University [2]HUST
https://mmcheng.net/rcfEdge/

CVPR 2017

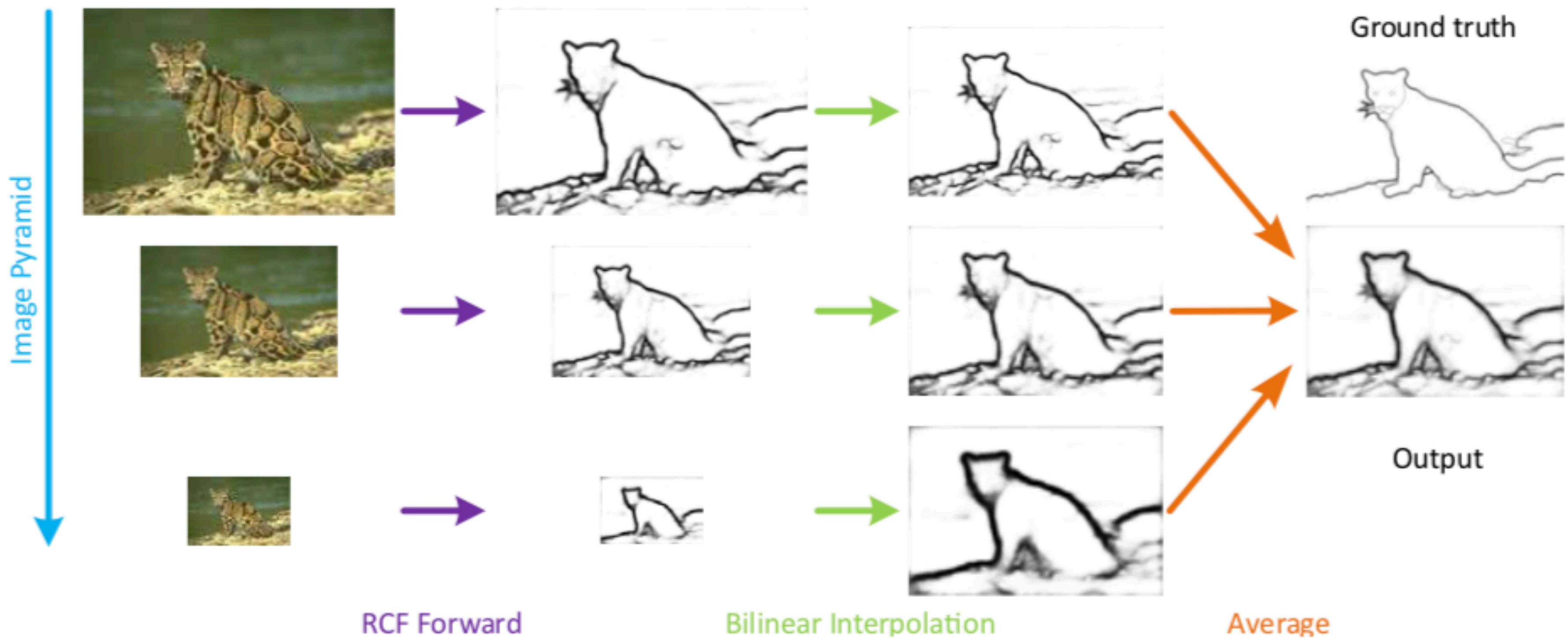
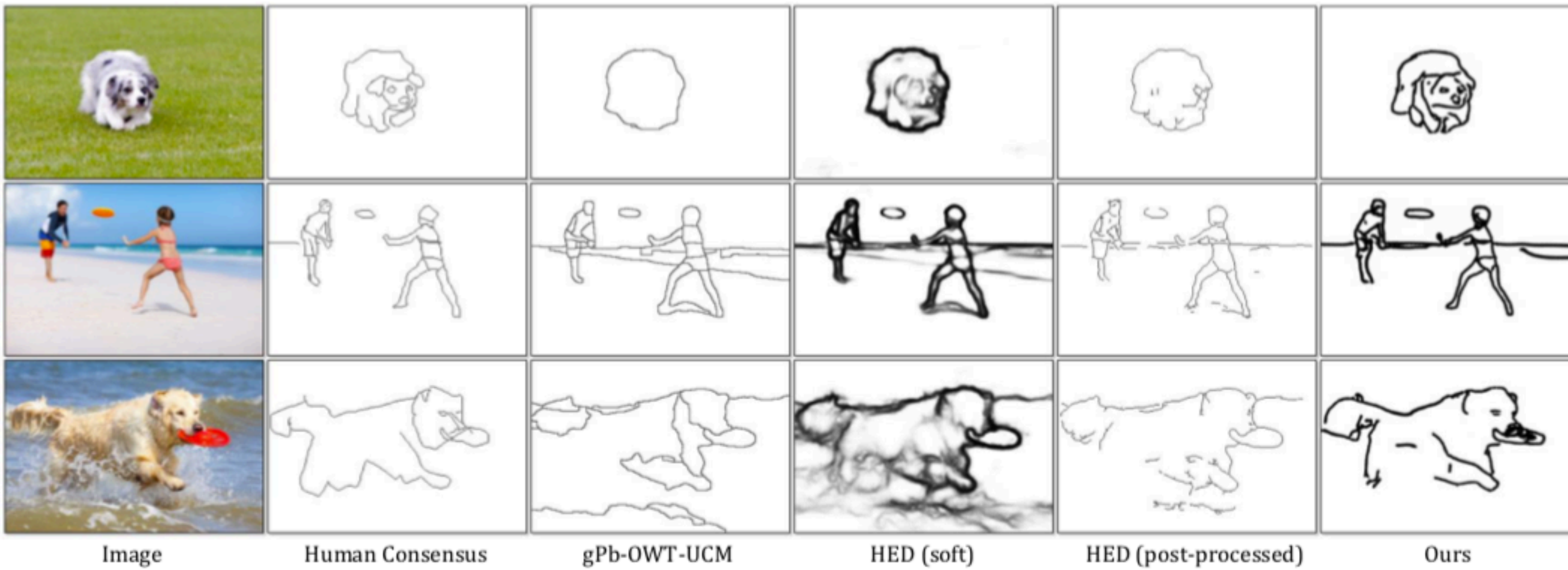


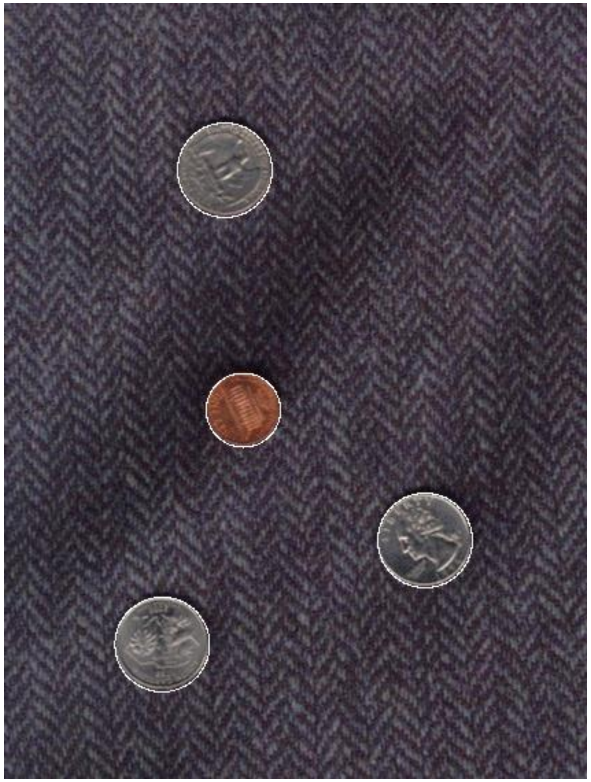
Photo-Sketching: Inferring Contour Drawings from Images

WACV 2019

Mengtian Li[1] Zhe Lin[2] Radomír Měch[2] Ersin Yumer[3] Deva Ramanan[1,4]
[1]Carnegie Mellon University [2]Adobe Research [3]Uber ATG [4]Argo AI



Uses fairly advanced deep net technique (GANs), which we'll discuss only later in the course.



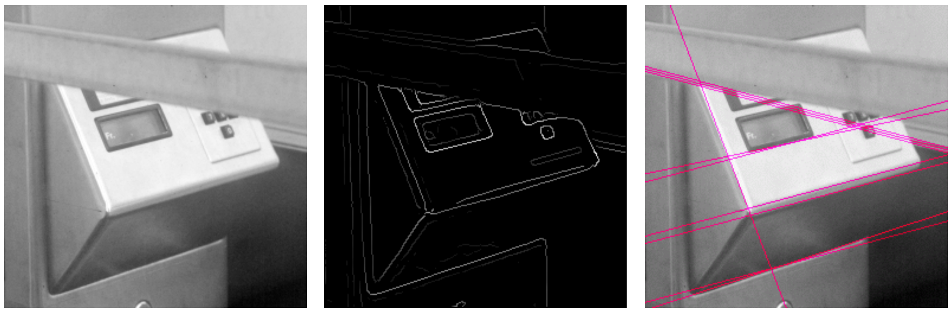
Voting and the Hough Transform

Disclaimer: Many slides have been borrowed from Devi Parikh and/or Kristen Grauman, who may have borrowed from others.

Slide credit: Kristen Grauman

Fitting

- 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.

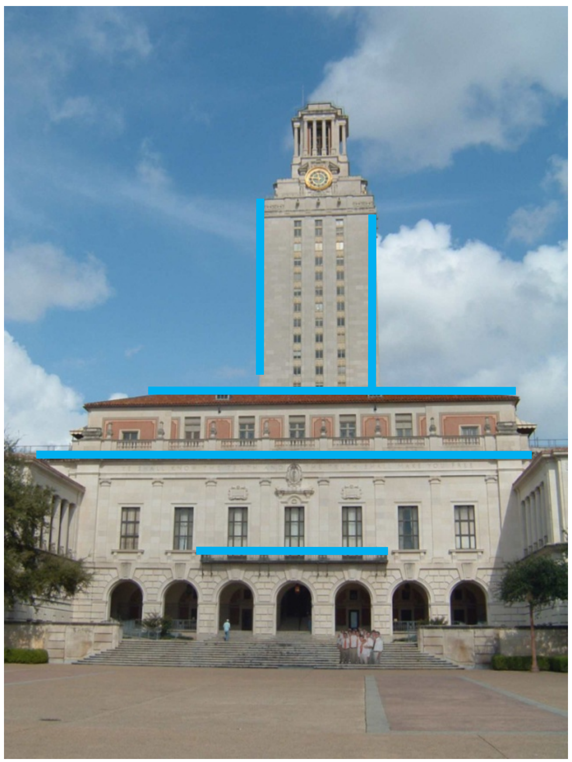
Fitting: Main idea

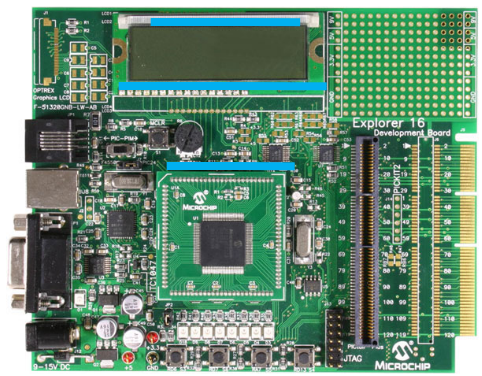
- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can’t tell whether a point belongs to a given model just by looking at that point
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which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

Slide credit: L. Lazebnik

Example: Line fitting

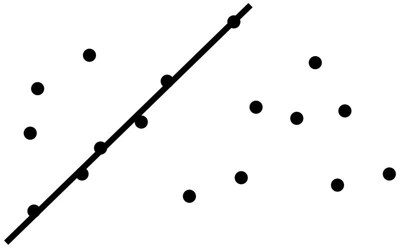
- Why fit lines?
  Many objects characterized by presence of straight lines

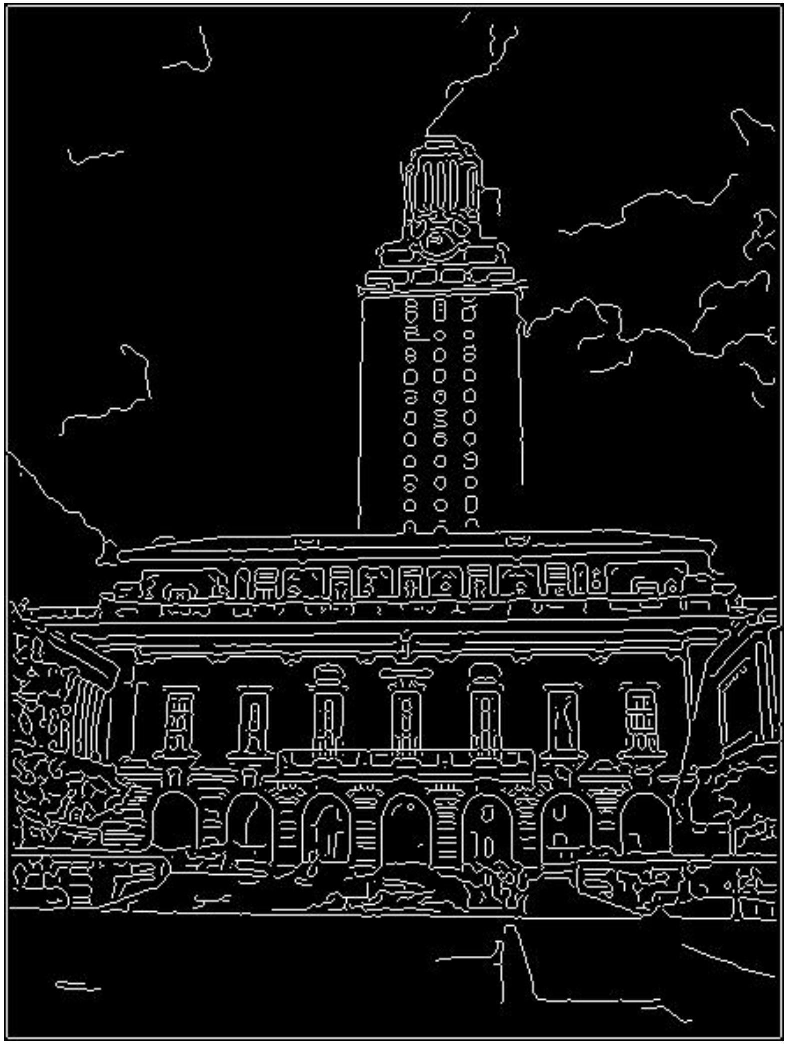




- Wait, why aren’t we done just by running edge detection?

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?

Kristen Grauman

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.

Kristen Grauman

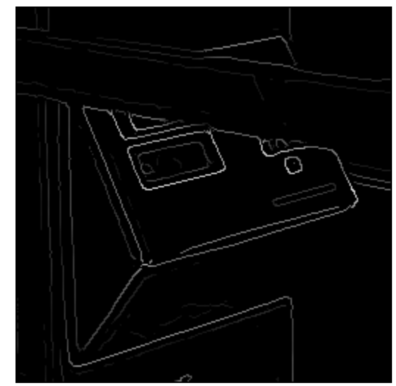
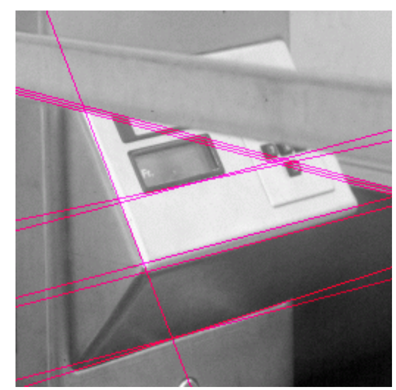
Fitting lines: Hough transform

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

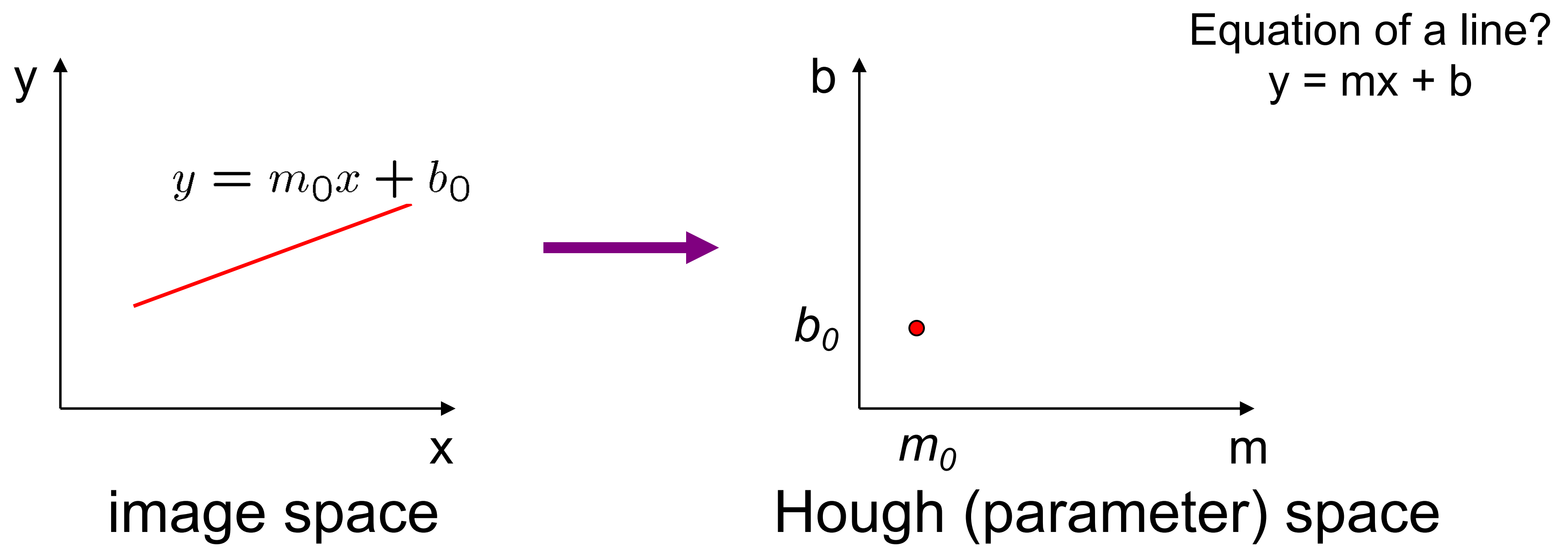
- **Hough Transform** is a voting technique that can be used to answer all of these questions.

  Main idea:

  1. Record vote for each possible line on which each edge point lies.
  2. Look for lines that get many votes.

Finding lines in an image: Hough space



Connection image (x,y) and Hough (m,b) spaces:

- Line in 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

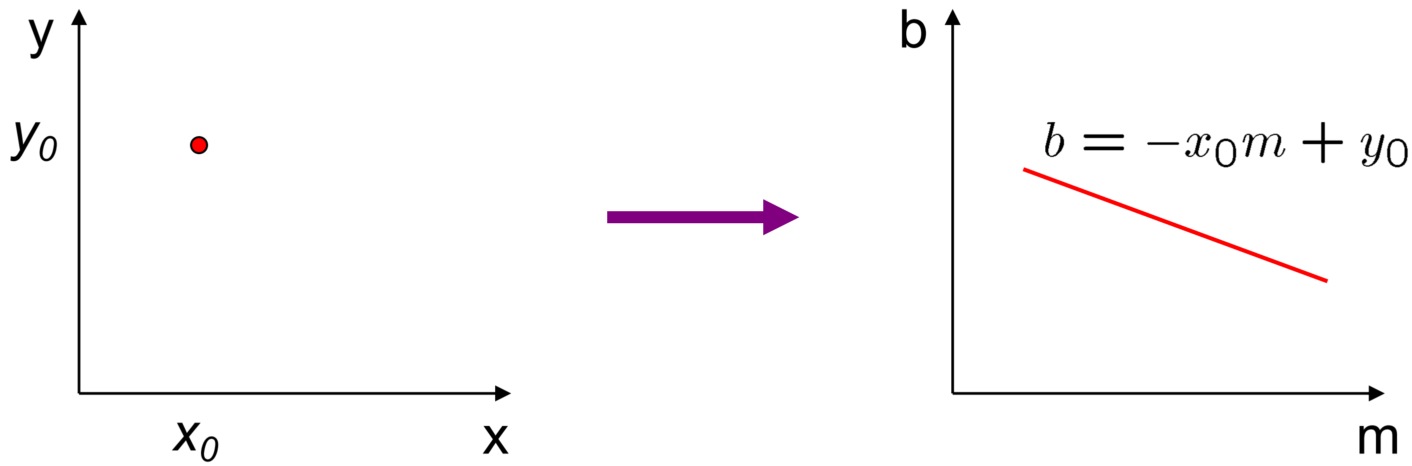


image space

Hough (parameter) space

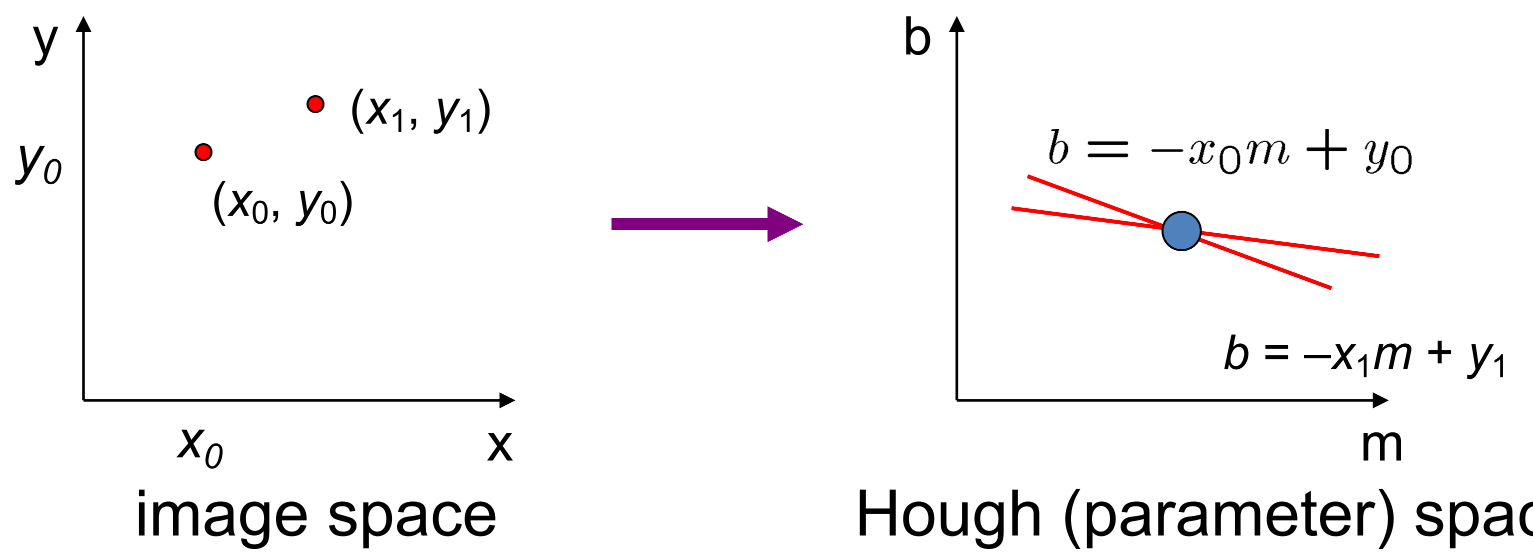
Connection between image (x,y) and Hough (m,b) spaces

- 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 $b = -x_0 m + y_0$ and $b = -x_1 m + y_1$

Slide credit: Kristen Grauman

Finding lines in an image: Hough algorithm

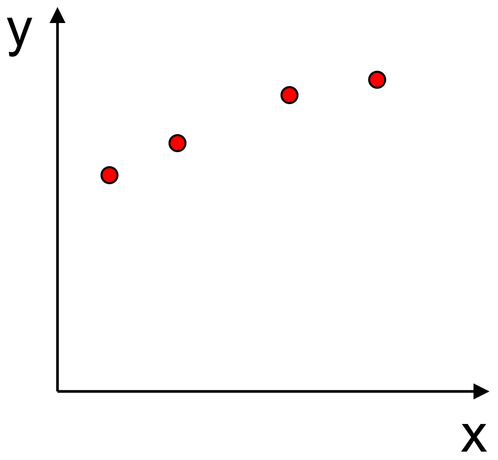
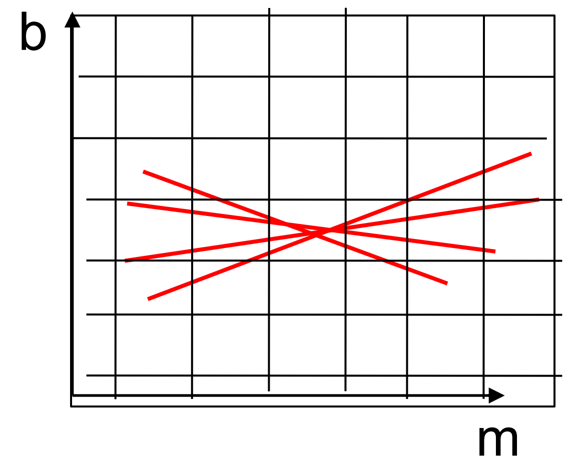


image space



Hough (parameter) space

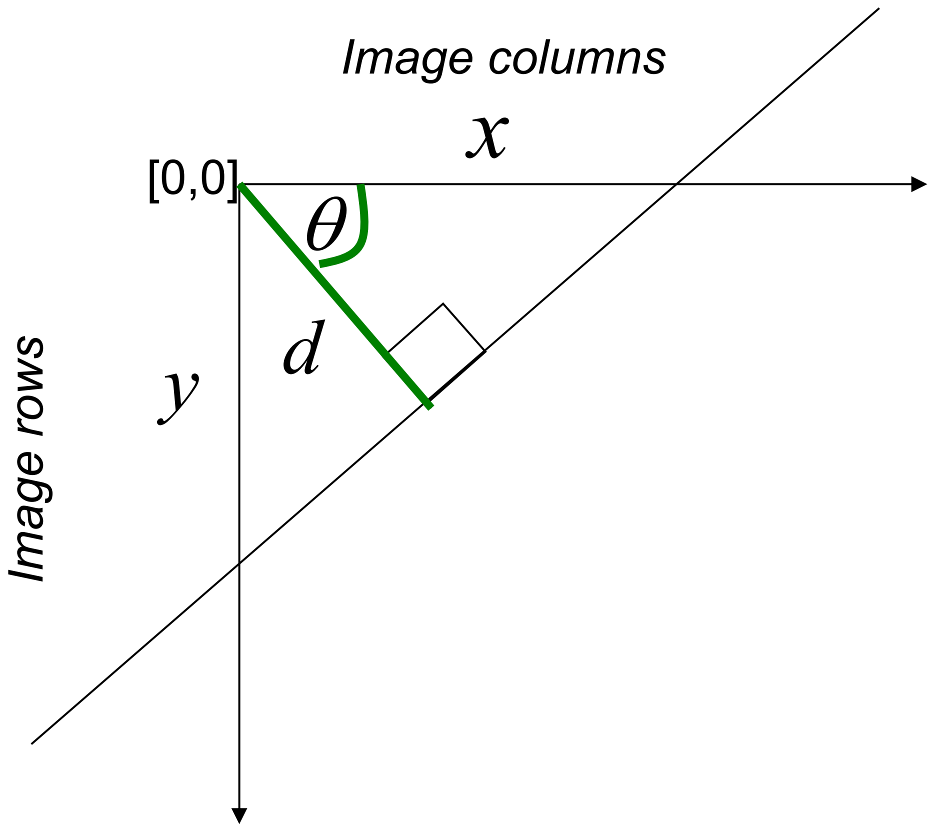
How can we use this to find the most likely parameters (m,b) for the most prominent line in the image space?

- 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.

Slide credit: Kristen Grauman

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 makes with the x-axis

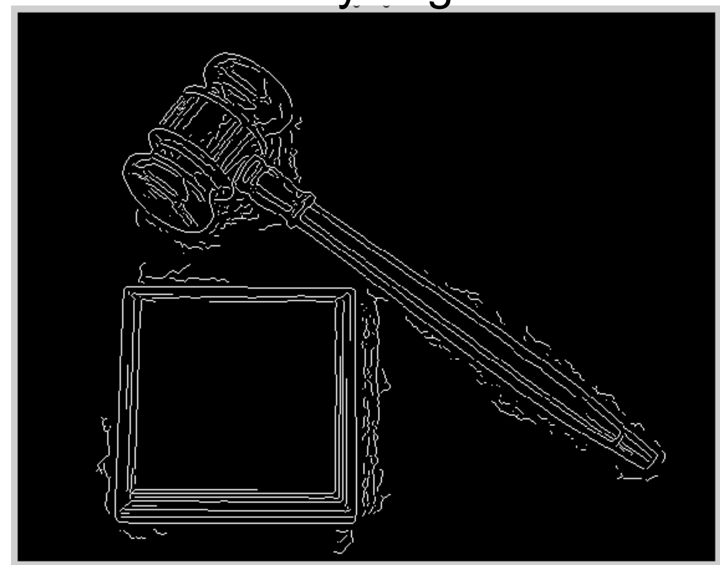
$$x\cos\theta - y\sin\theta = d$$

Point in image space → sinusoid segment in Hough space

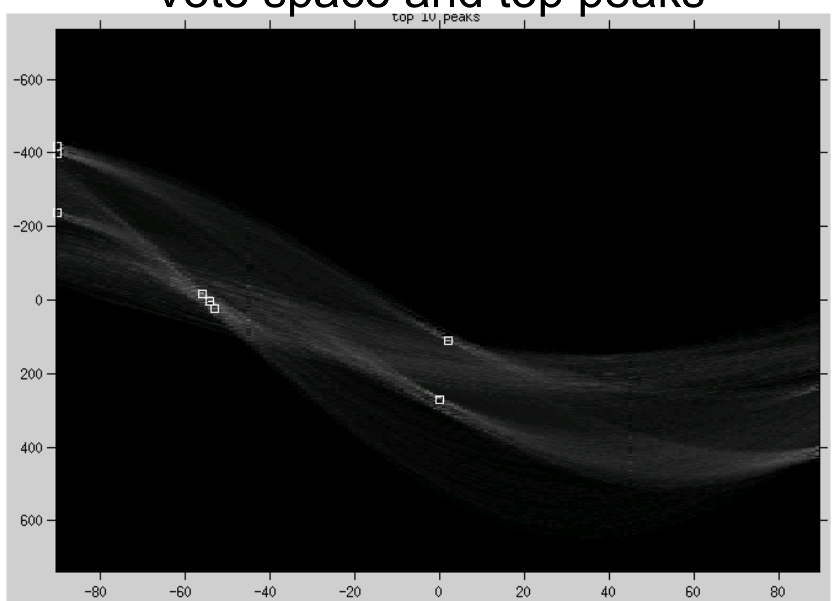
Kristen Grauman

Original image

Canny edges



Vote space and top peaks



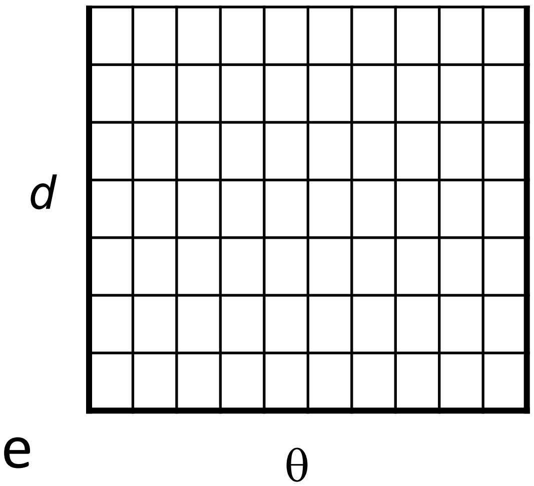
Kristen Grauman

Hough transform algorithm

Using the polar parameterization:

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

H: accumulator array (votes)



Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. for each edge point I[x, y] in the image

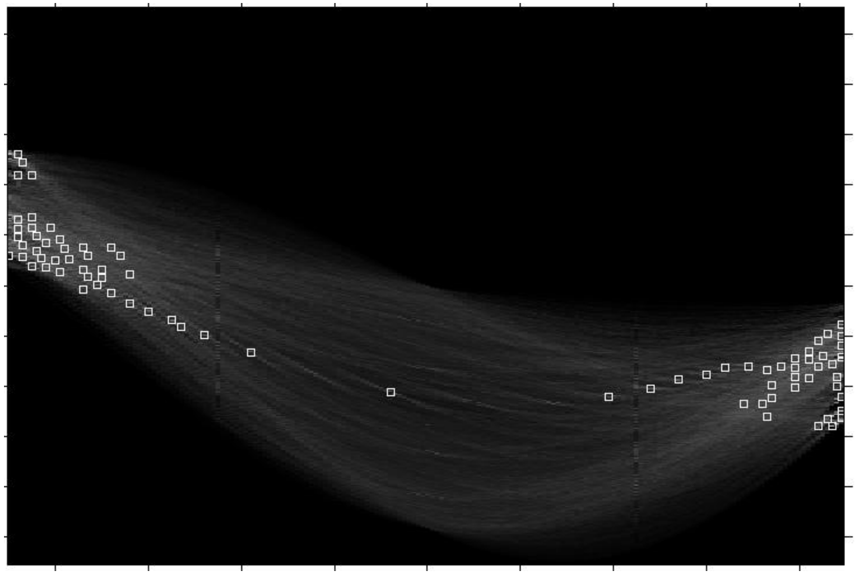
   for $\theta = [\theta_{min}$ to $\theta_{max}]$ // some quantization

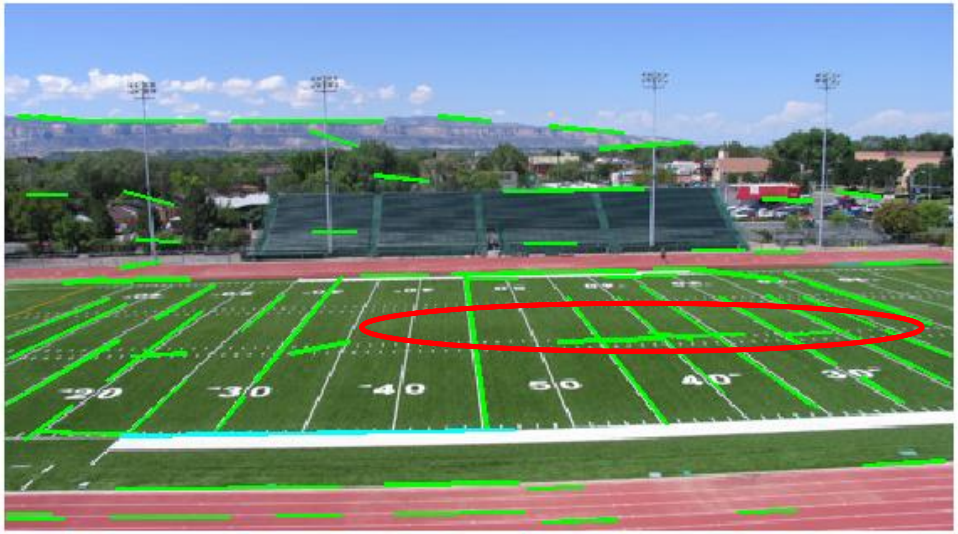
   $$d = x\cos\theta - y\sin\theta$$

   H[d, θ] += 1
3. Find the value(s) of (d, θ) where H[d, θ] is maximum
4. The detected line in the image is given by

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

Source: Steve Seitz





Showing longest segments found

Kristen Grauman

Impact of noise on Hough

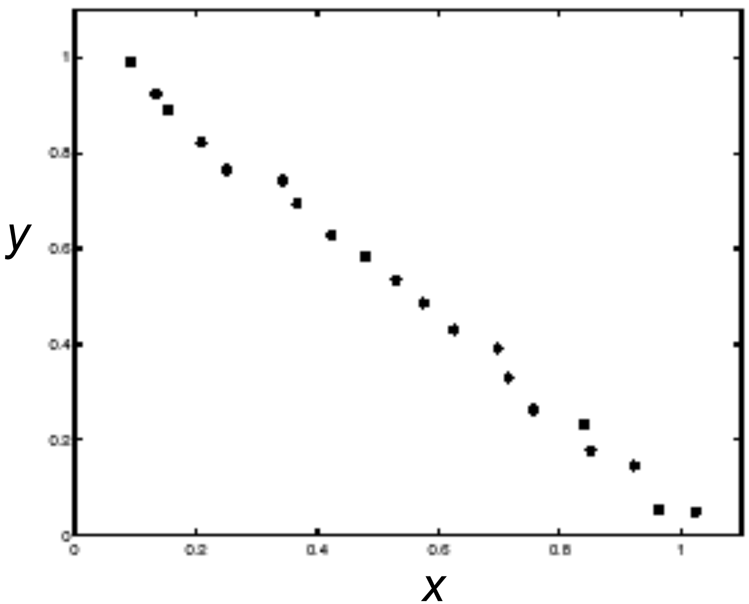
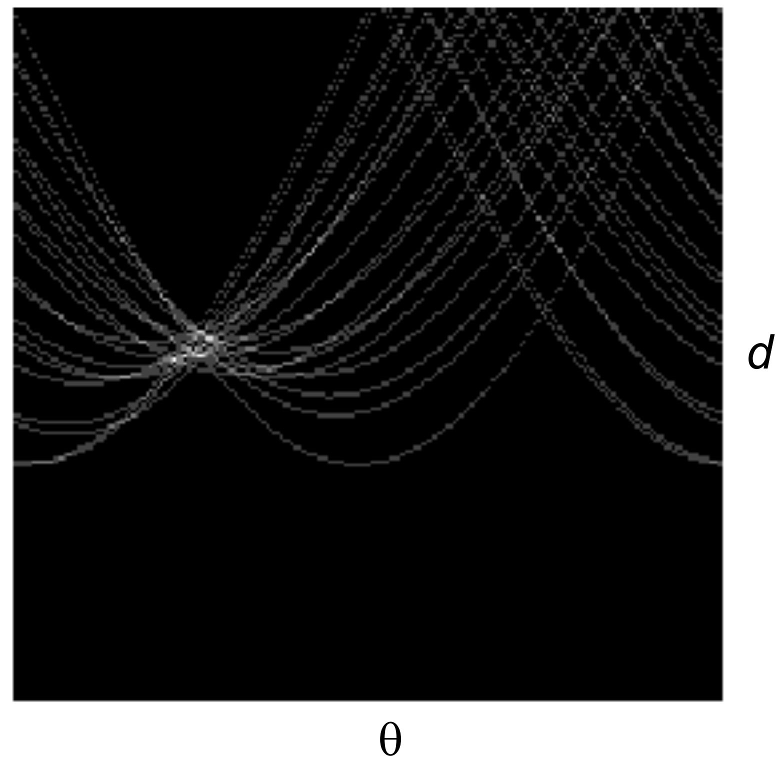


Image space
edge coordinates



Votes

What difficulty does this present for an implementation?

Slide credit: Kristen Grauman

Impact of noise on Hough

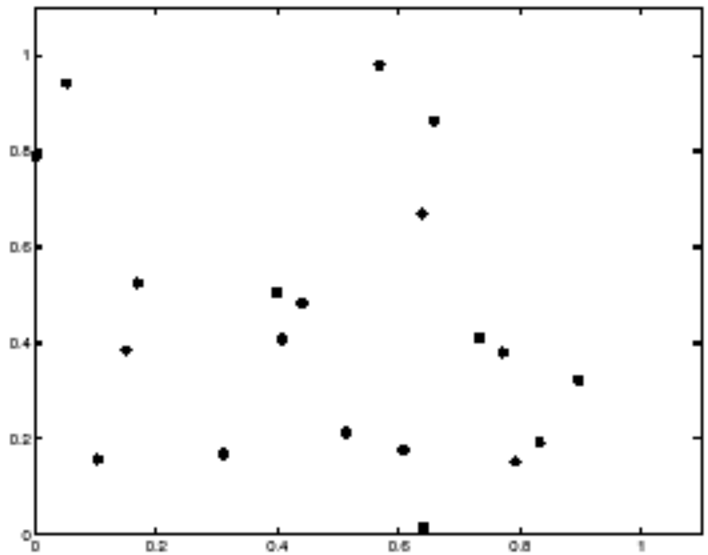
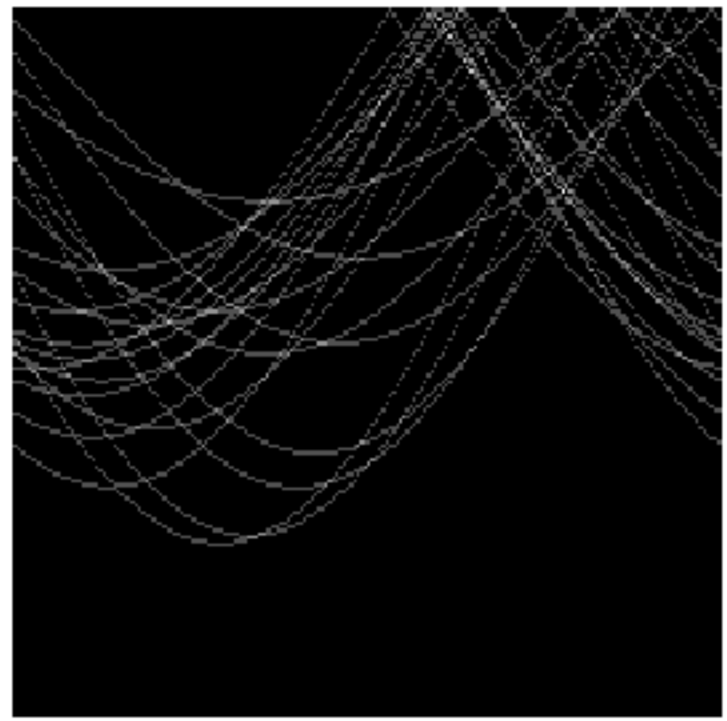


Image space edge coordinates



Votes

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

Hough transform for circles

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

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

Equation of circle?

- For a fixed radius r

Equation of set of circles that all pass through a point?

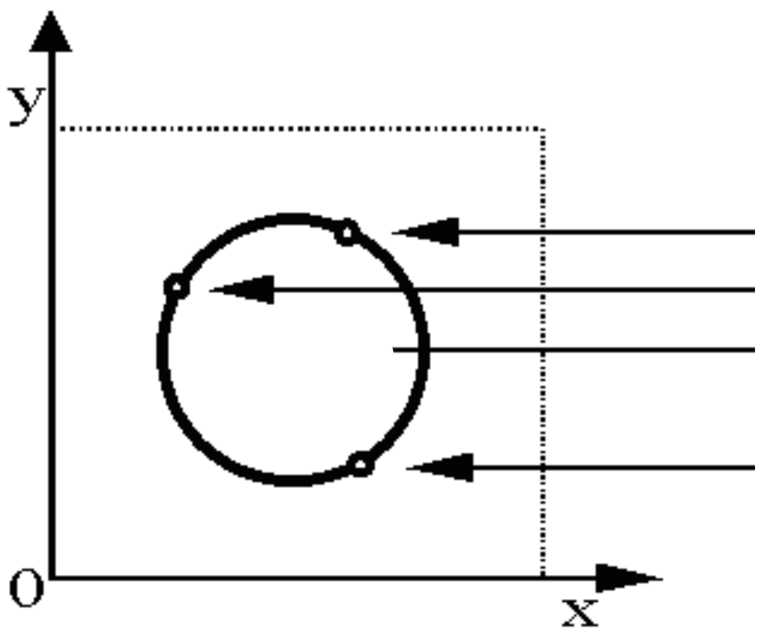
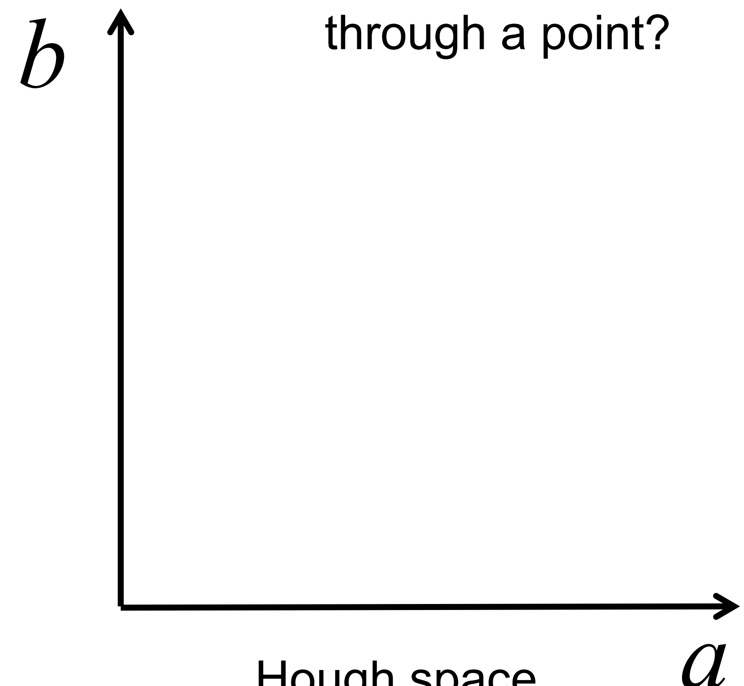


Image space



Hough space

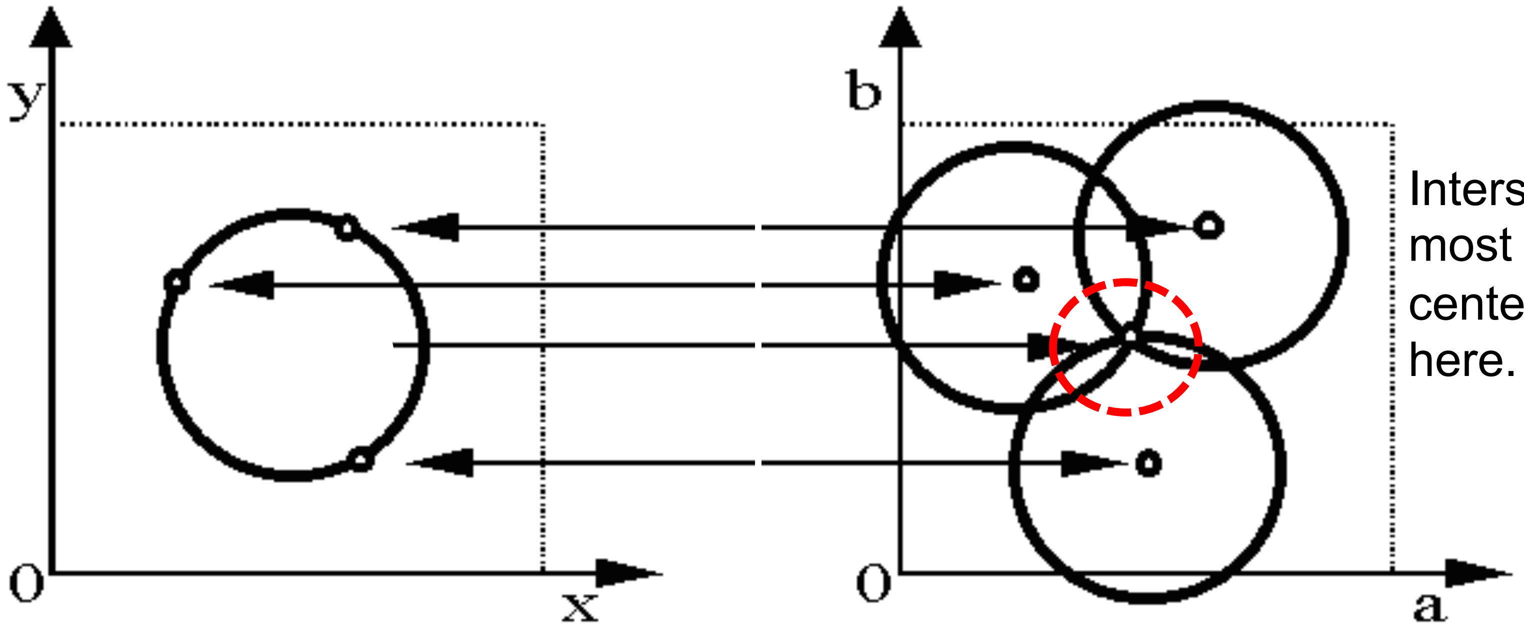
Adapted by Devi Parikh from: Kristen Grauman

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



Intersection: most votes for center occur here.

Image space

Hough space

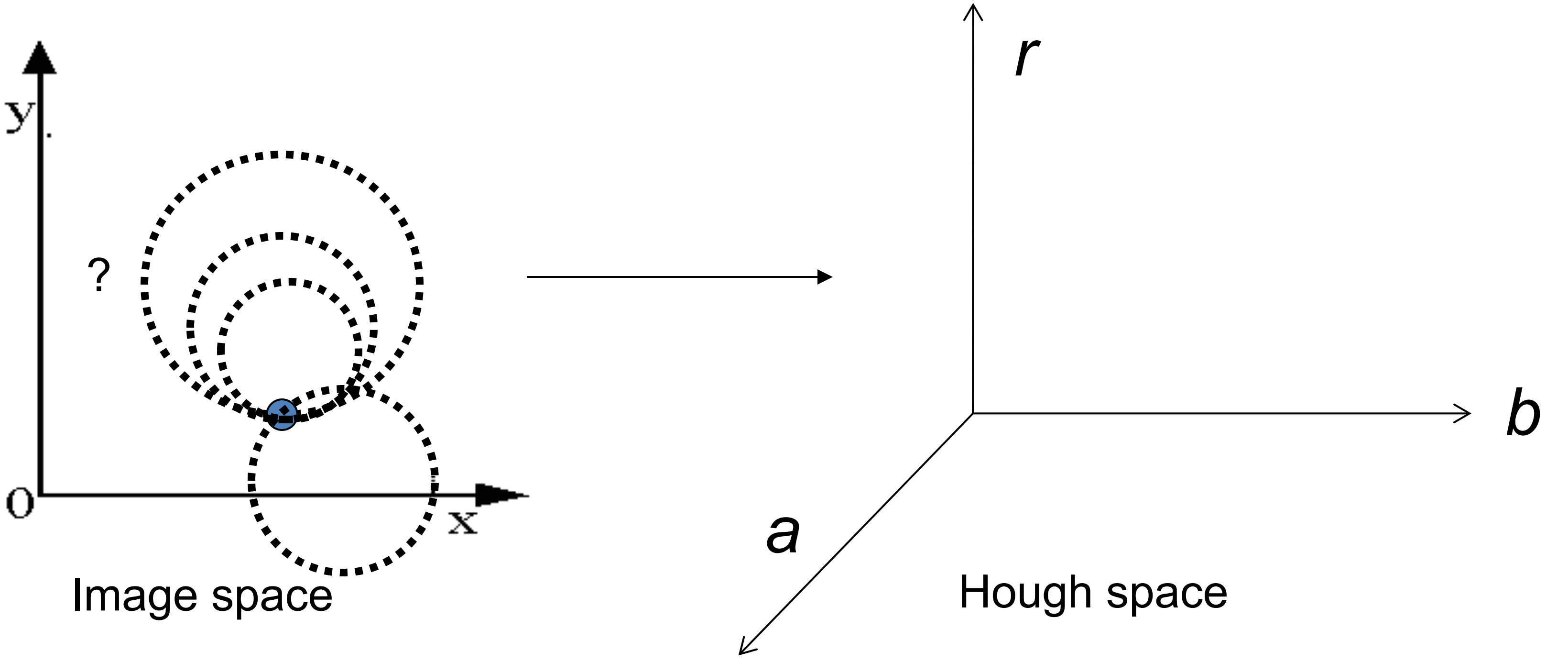
Kristen Grauman

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



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

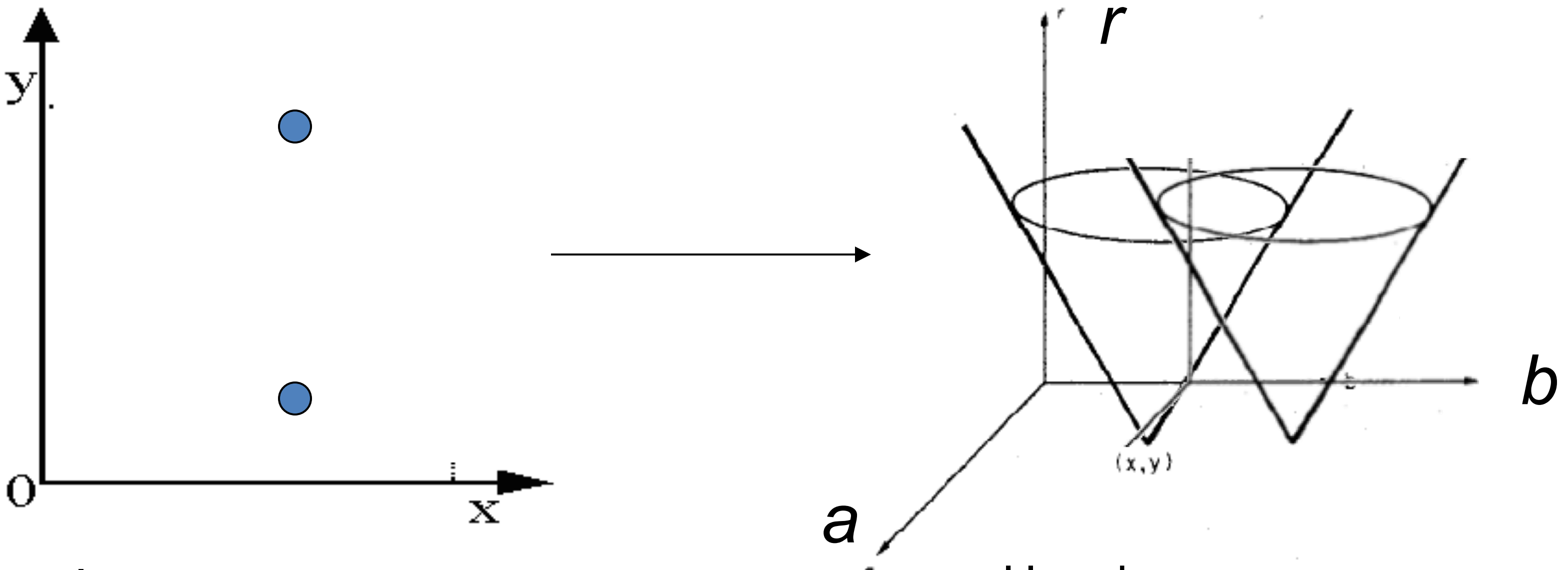


Image space

Hough space

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

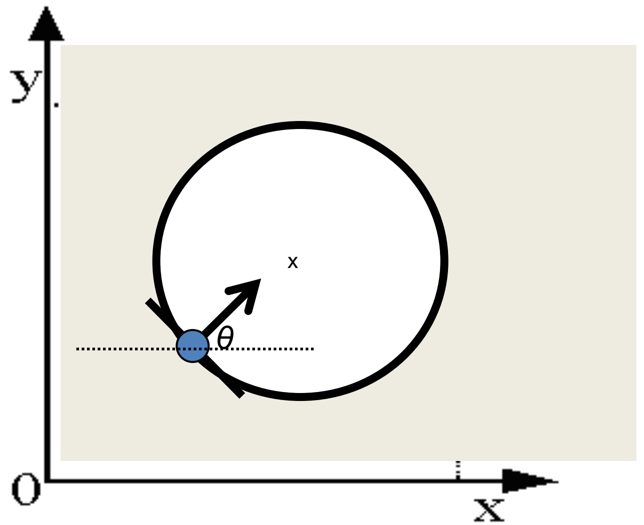
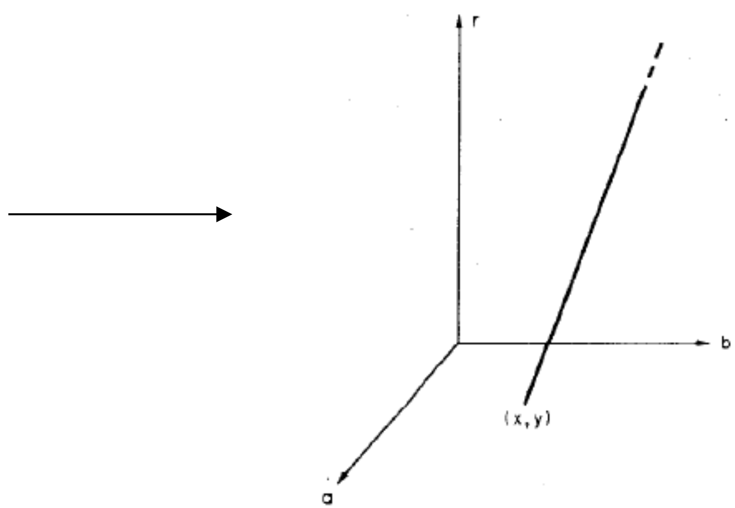


Image space



Hough space

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 at (x,y)

$a = x - r\cos(\vartheta)$ // column

$b = y + r\sin(\vartheta)$ // row

H[a,b,r] += 1

end

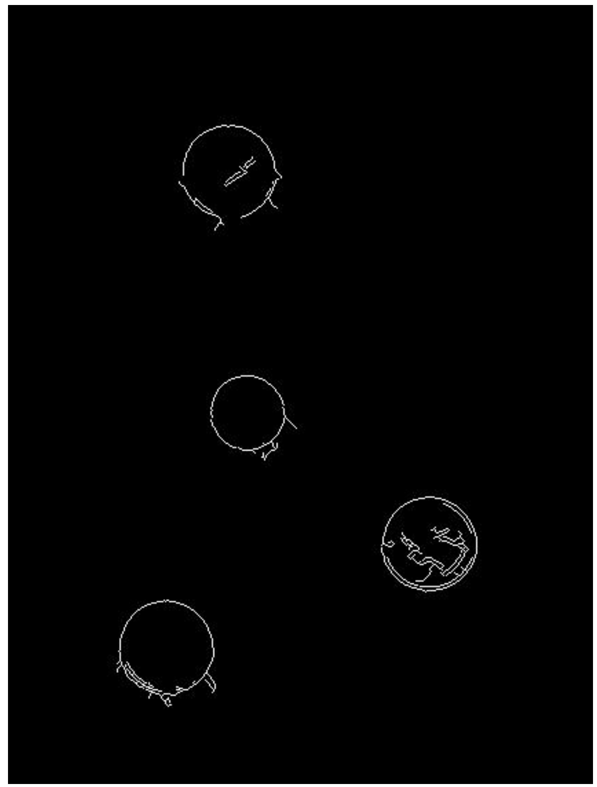
end

Kristen Grauman

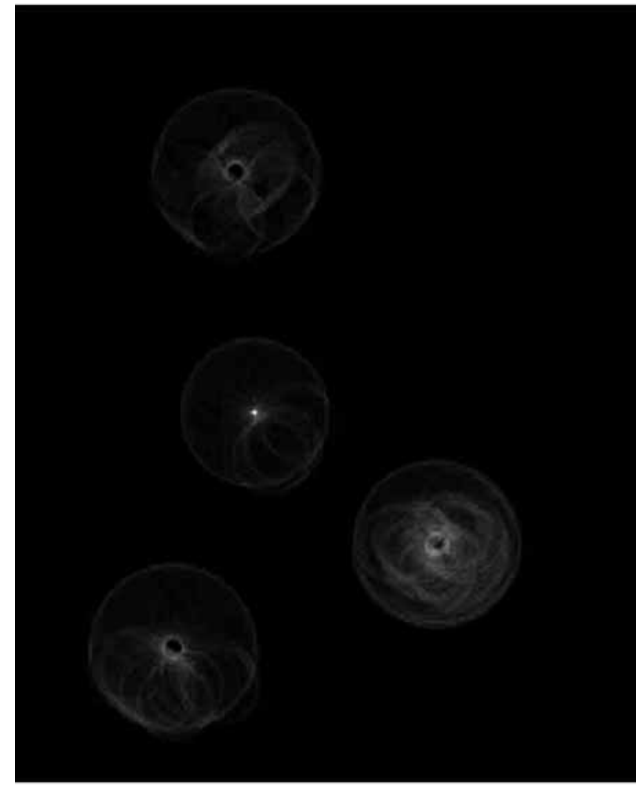
Example: detecting circles with Hough

Original

Edges



Votes: Penny

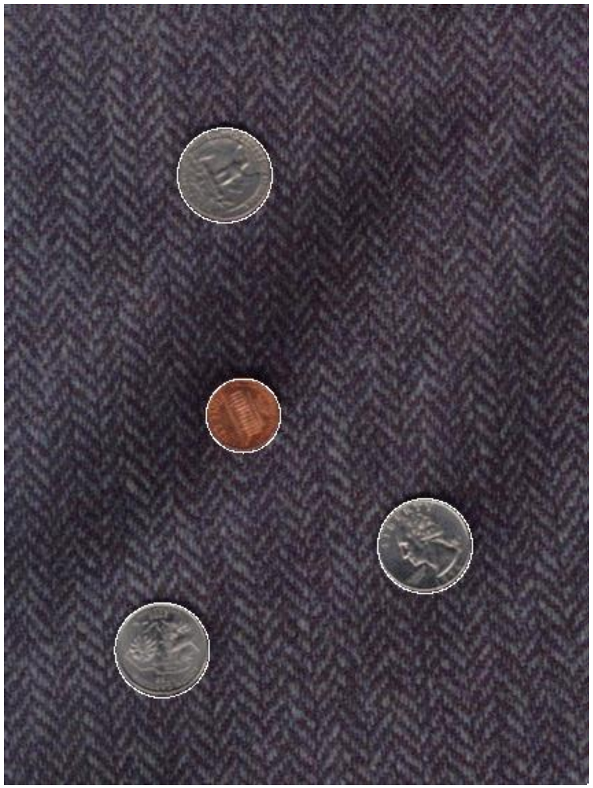


Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

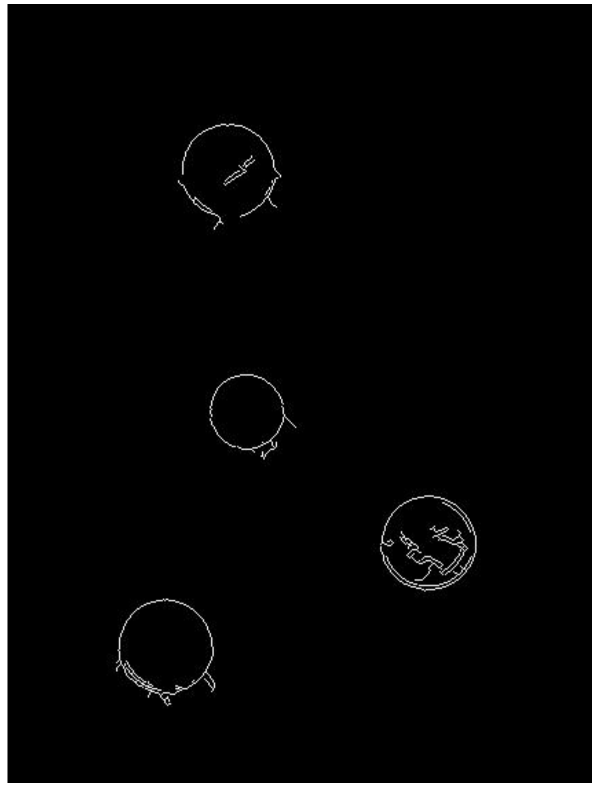
Slide credit: Kristen Grauman

Example: detecting circles with Hough

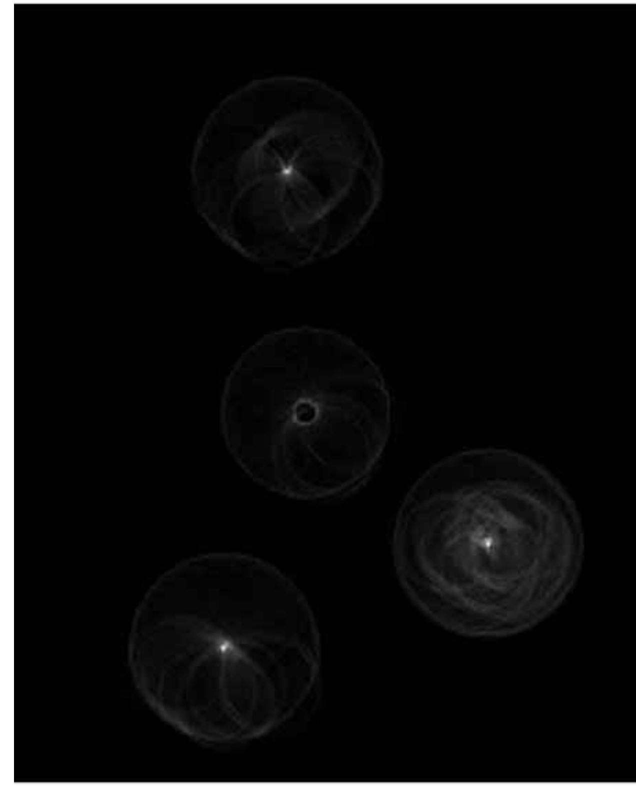
Combined detections



Edges



Votes: Quarter

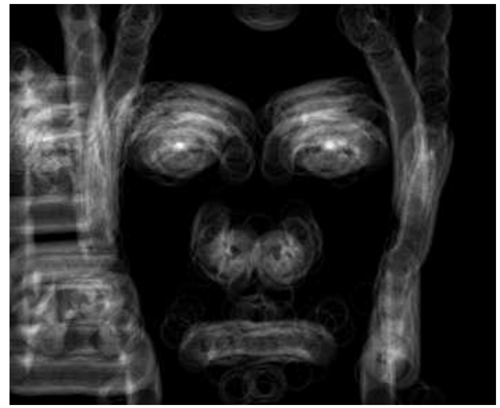


Slide credit: Kristen Grauman

Coin finding sample images from: Vivek Kwatra

Example: iris detection

Gradient+threshold



Hough space
(fixed radius)

Max detections

- Hemerson Pistori and Eduardo Rocha Costa
  http://rsbweb.nih.gov/ij/plugins/hough-circles.html

Kristen Grauman

Example: iris detection


Figure 2. Original image


Figure 3. Distance image


Figure 4. Detected face region

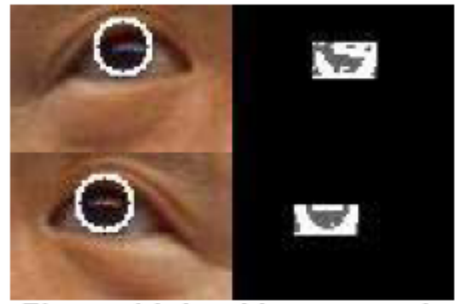


Figure 14. Looking upward

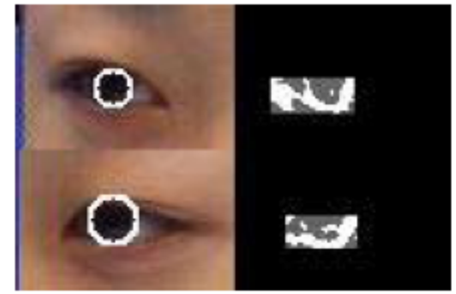


Figure 15. Looking sideways

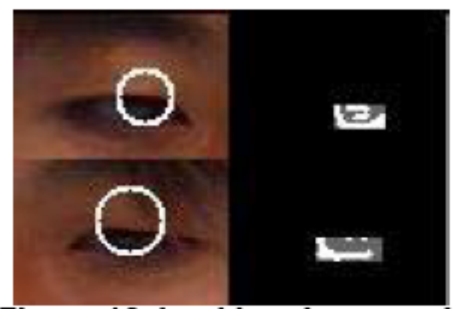
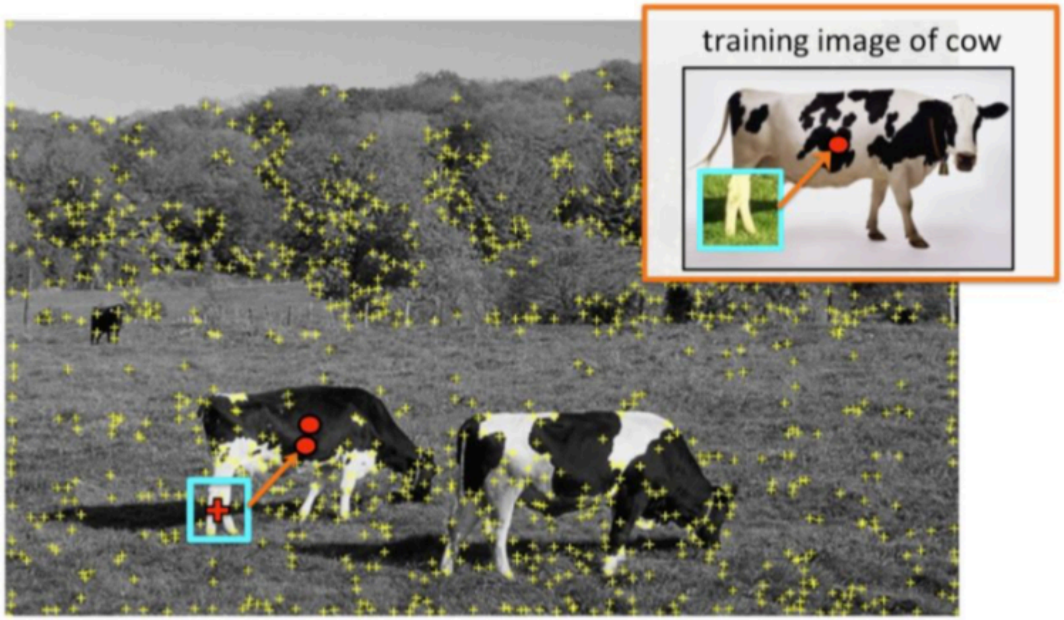


Figure 16. Looking downward

- An Iris Detection Method Using the Hough Transform and Its Evaluation for Facial and Eye Movement, by Hideki Kashima, Hitoshi Hongo, Kunihito Kato, Kazuhiko Yamamoto, ACCV 2002.

Hough Voting for Object recognition



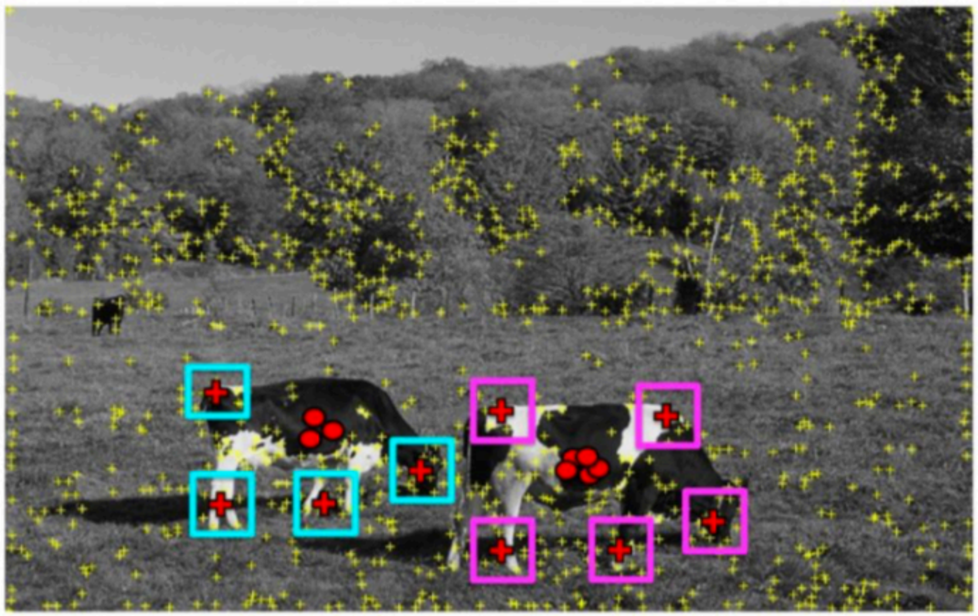
vote for center of object

Hough voting pipeline (in 2D):

- Select interest points
- Match patch around each interest point to a training patch (codebook)
- Vote for object center given that training instance

From U. Toronto CSC420

Hough Voting for Object recognition



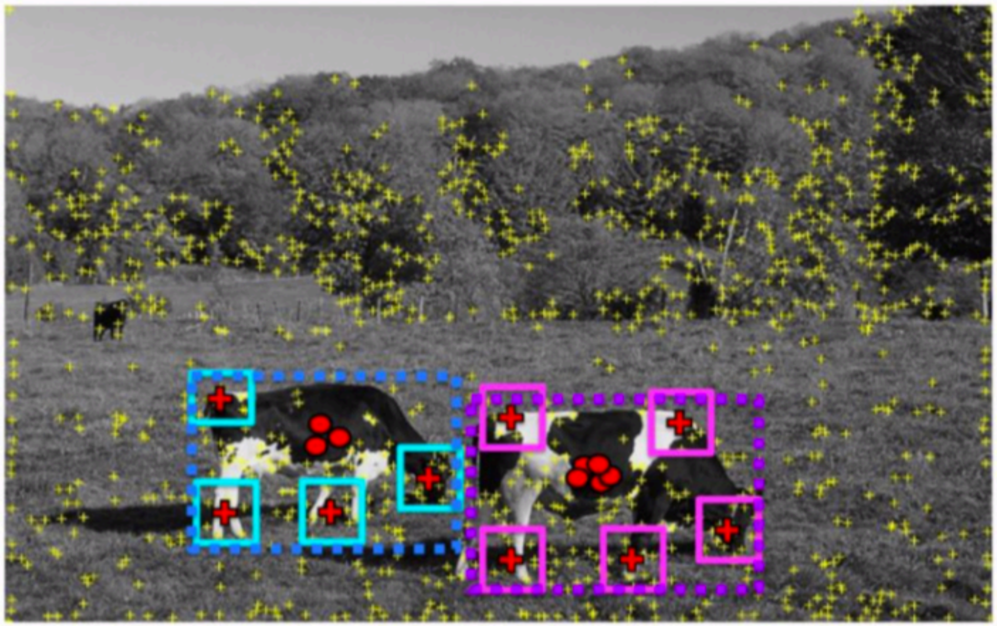
Find patches that voted for the peaks (back-projection).

Hough voting pipeline (in 2D):

- Select interest points
- Match patch around each interest point to a training patch (codebook)
- Vote for object center given that training instance
- Votes clustering to find peaks
- **Find patches that voted for the peaks back-projection**

From U. Toronto CSC420

Hough Voting for Object recognition



Find full objects based on the back-projected patches.

Hough voting pipeline (in 2D):

- Select interest points
- Match patch around each interest point to a training patch (codebook)
- Vote for object center given that training instance
- Votes clustering to find peaks
- Find patches that voted for the peaks back-projection
- **Find full objects based on back-projected patches**

From U. Toronto CSC420

Hough transform: pros and cons

Pros

- All points are processed independently, so can cope with occlusion, gaps
- 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: can be tricky to pick a good grid size

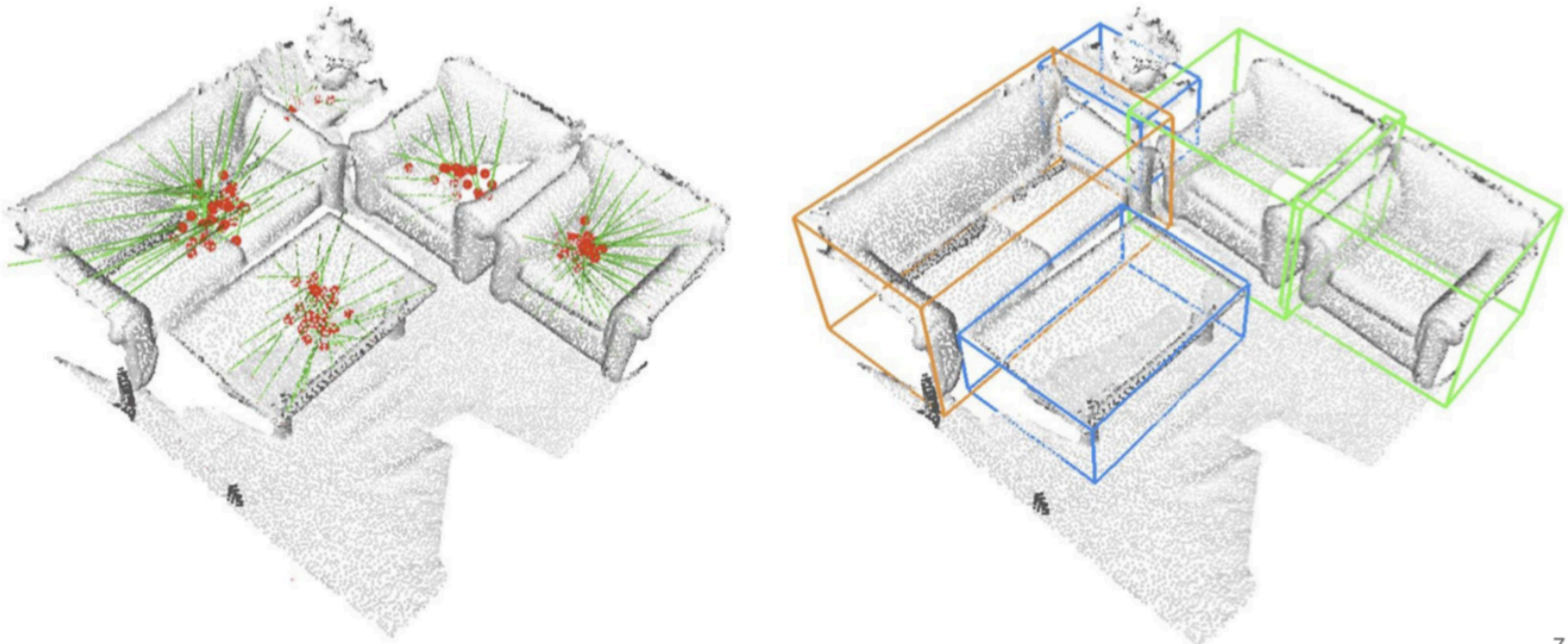
Kristen Grauman

Deep Hough Voting for 3D Object Detection in Point Clouds

Charles R. Qi[1] Or Litany[1] Kaiming He[1] Leonidas J. Guibas[1,2]
[1]Facebook AI Research [2]Stanford University

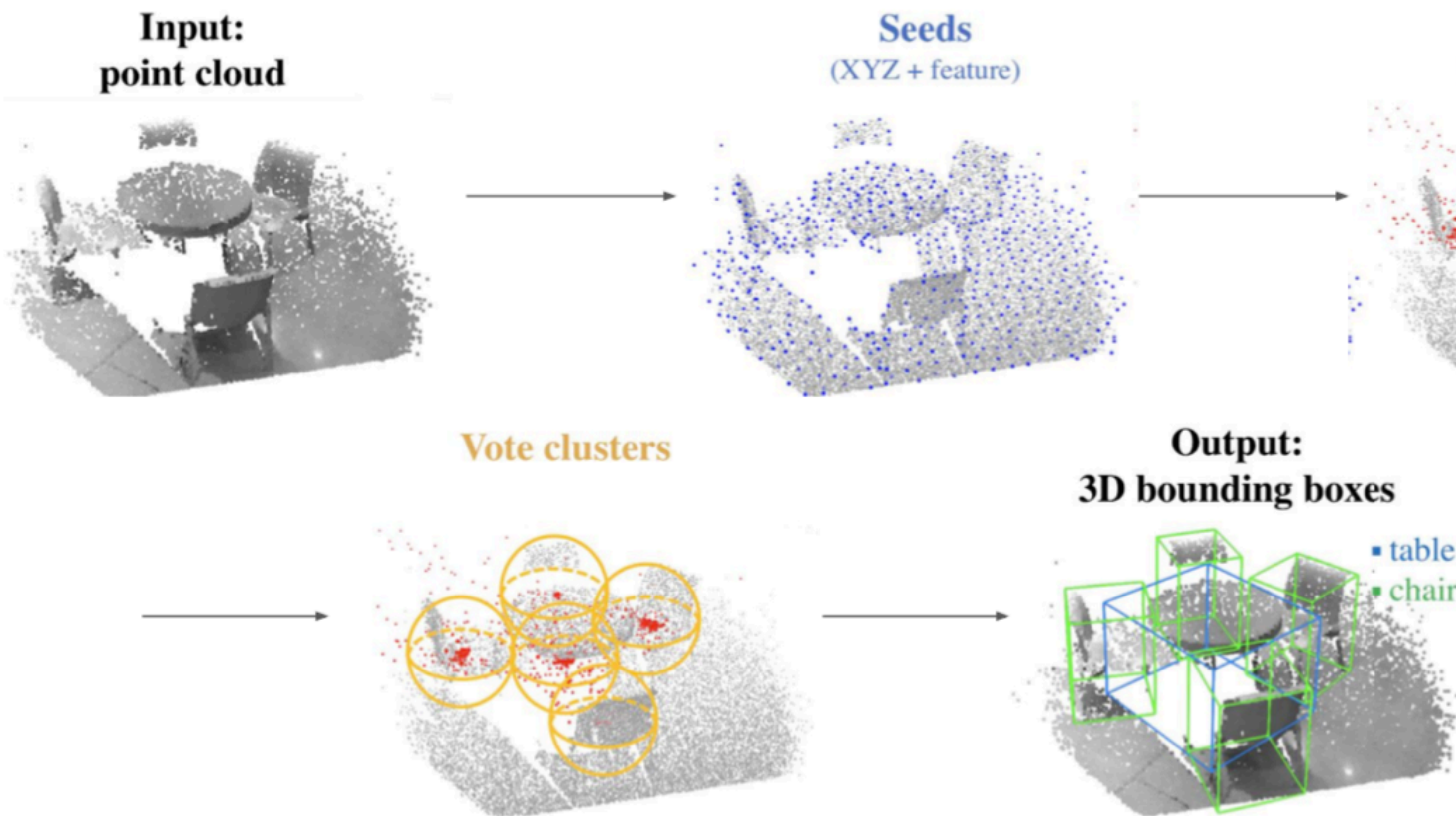
ICCV 2019

Deep Hough Voting: 3D Object Detection in Point Clouds



Or Litany

Deep Hough voting:



Or Litany