CS4495/6495
Introduction to Computer Vision

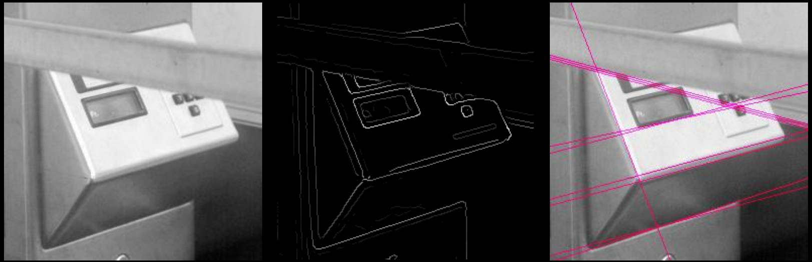
2B-L1 *Hough transform: Lines*

Now some “real” vision…

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

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

Fitting a model



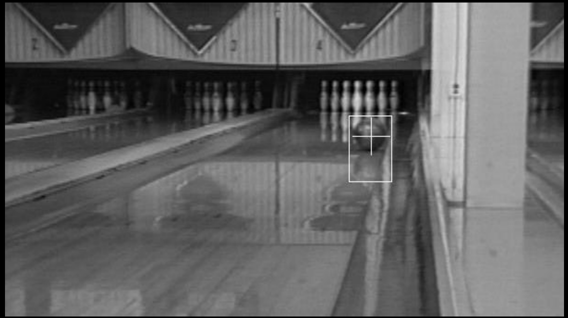


Figure from Marszalek & Schmid, 2007

Parametric model

- A ***parametric*** model can represent a class of instances where each is defined by a value of the parameters.
- Examples include lines, or circles, or even a parameterized template.

Fitting a parametric model

- Choose a parametric model to represent a set of features
- Membership criterion is not local:

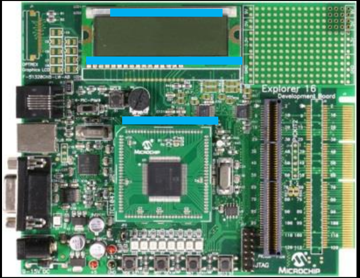
  Can't tell whether a point in the image belongs to a given model just by looking at that point

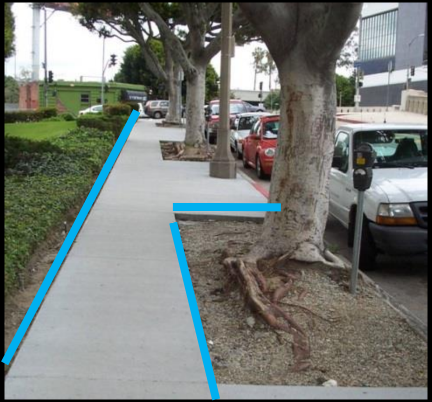
- Computational complexity is important

  Not feasible to examine possible parameter setting

Source: S. Lazebnik

Example: Line fitting





Source: K. Grauman

Difficulty of line fitting

- Extra edge points (clutter), multiple models.
- Only some parts of each line detected, and some parts are missing.
- Noise in measured edge points, orientations.

Voting

It’s not feasible to check all possible models or all combinations of features (e.g. edge pixels) 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.

1. Cycle through features, each casting votes for model parameters.
2. Look for model parameters that receive a lot of votes.

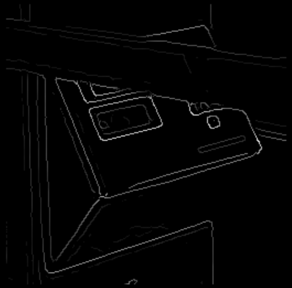
Voting – why it works

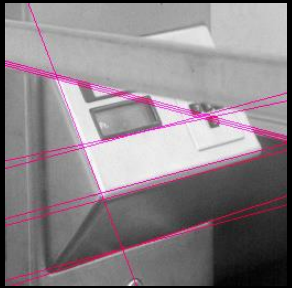
- Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of “good” features.
- Ok if some features not observed, as model can span multiple fragments.

Fitting lines

To fit lines we need to answer a few questions:

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



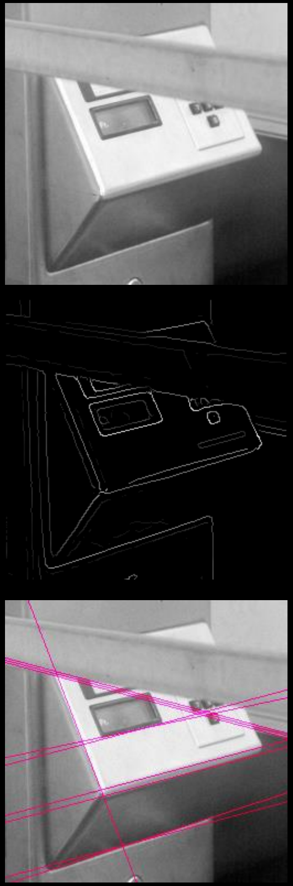


Fitting lines

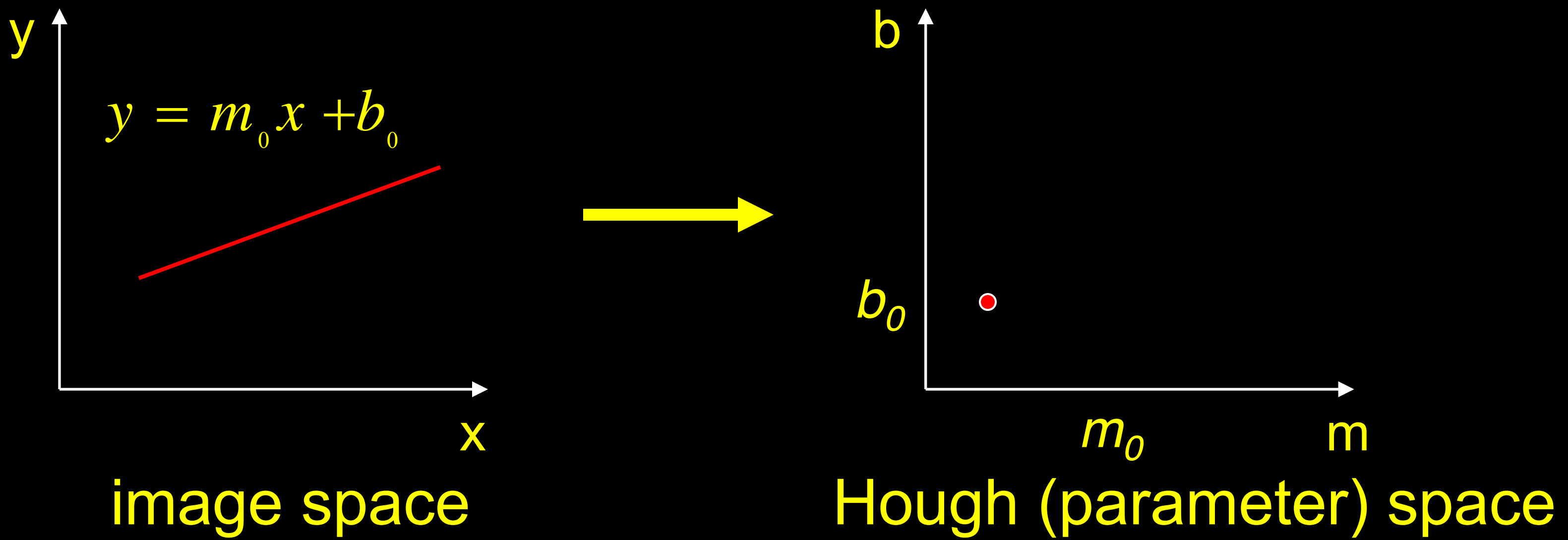
Hough Transform is a voting technique that can be used to answer all of these

Main idea

1. Each edge point votes for compatible lines.
2. Look for lines that get many votes.



Hough space



A line in the image corresponds to a point in Hough space

Slide credit: Steve Seitz

Hough space

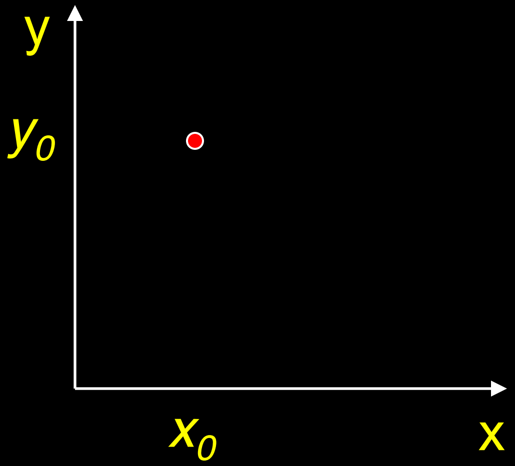
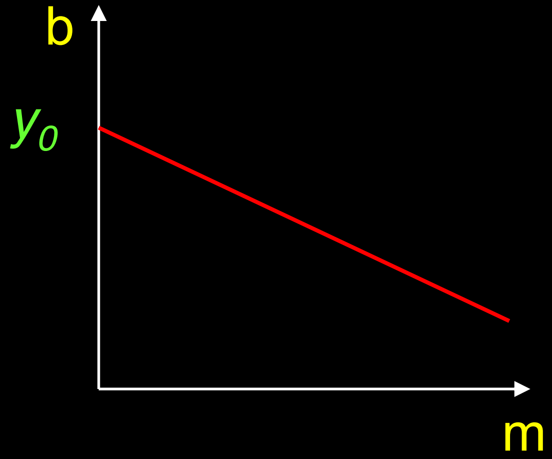


image space



Hough (parameter) space

$$y_0 = m x_0 + b \longrightarrow b = -x_0 m + y_0$$

Hough space

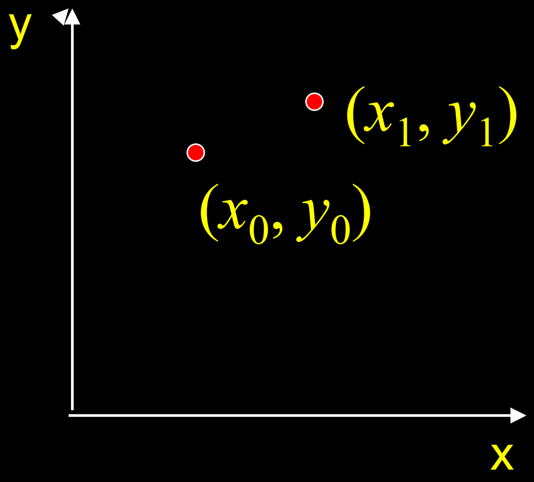
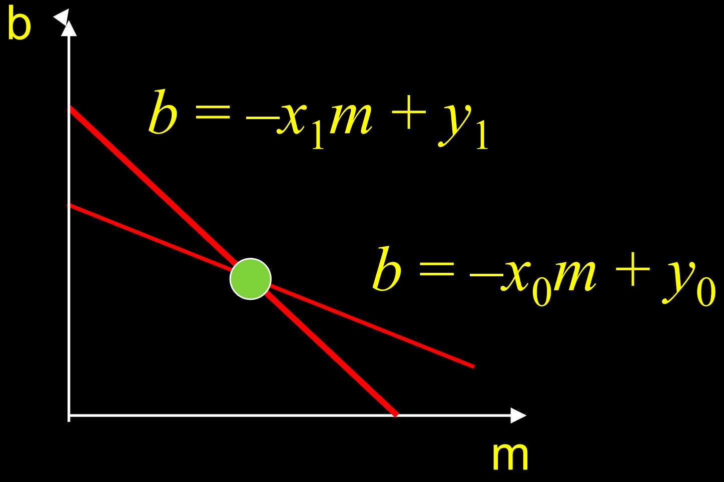
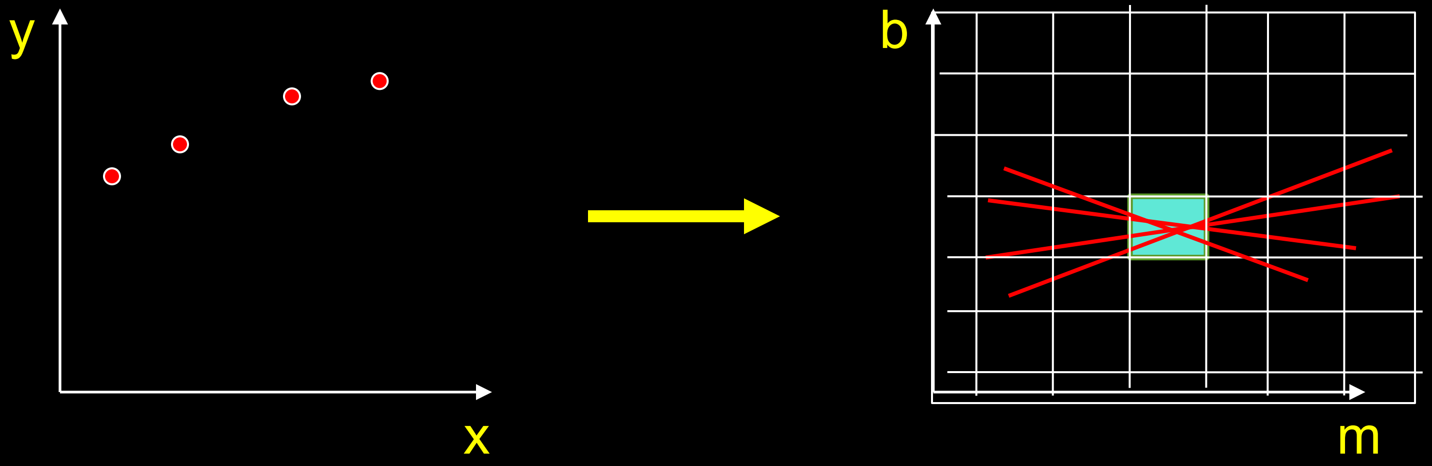


image space



Hough (parameter) space

Hough algorithm

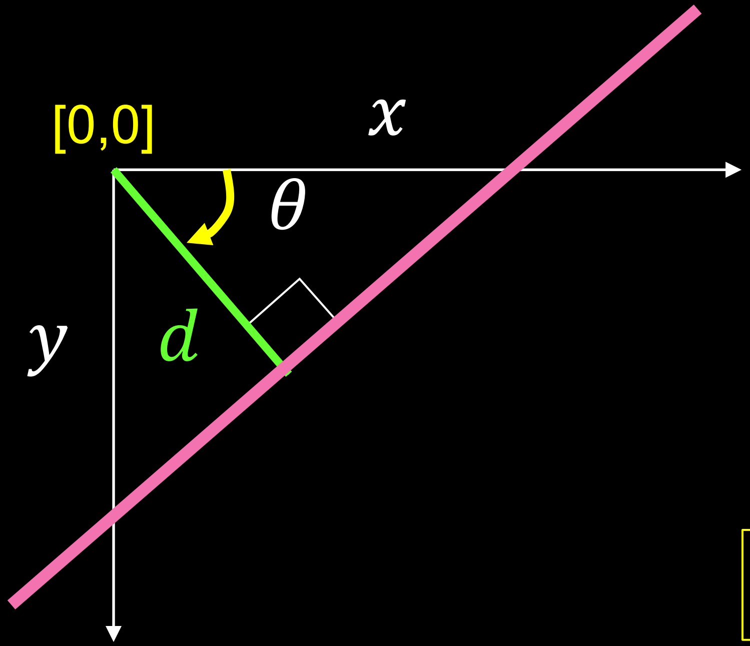


- Let each edge point in image space vote for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Line representation issues

- Before we implement this we need to rethink our representations of lines.
- As you may remember, there are issues with the $y – mx + b$ representation of line.
- In particular, undefined for vertical lines with m being infinity. So we use a more robust polar representation of lines.

Polar representation for lines

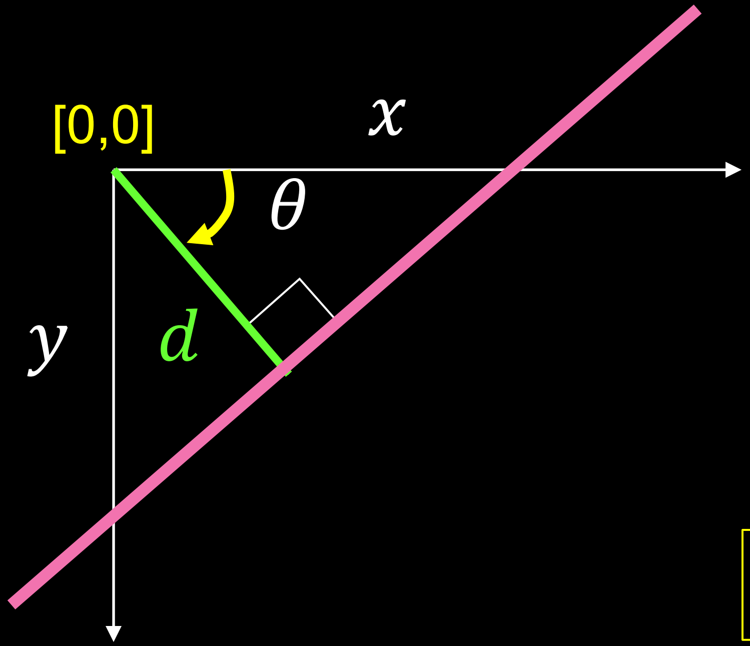


$\boldsymbol{d}$: perpendicular distance from line to origin

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

$$\boxed{x \cos\theta + y \sin\theta = d}$$

Polar representation for lines



$\boldsymbol{d}$: perpendicular distance from line to origin

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

$$\boxed{x\cos\theta + y\sin\theta = d}$$

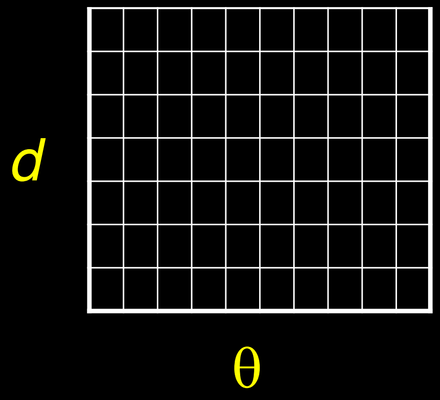
Point in image space is now sinusoid segment in Hough space

Hough transform algorithm

Using the polar parameterization:

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

And a *Hough Accumulator Array (keeps the votes)*



Source: Steve Seitz

Basic Hough transform algorithm

1. Initialize H[d, θ]=0
2. For each ***edge*** point in $E(x, y)$ in the image

    for θ = -90 to +90 // some quantization; why not 2pi?

    $d = x\cos\theta + y\sin\theta$ // maybe negative

    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

Complexity of the Hough transform

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

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

Hough example

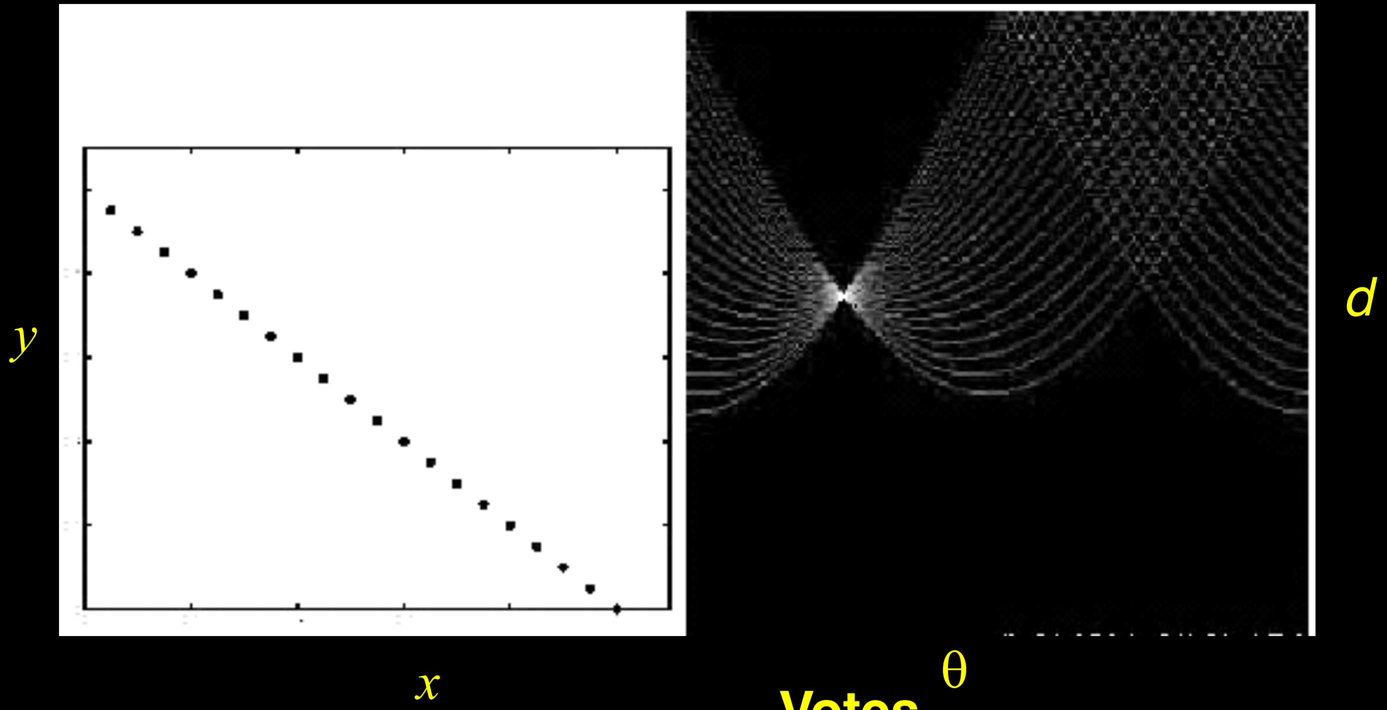


Image space edge coordinates

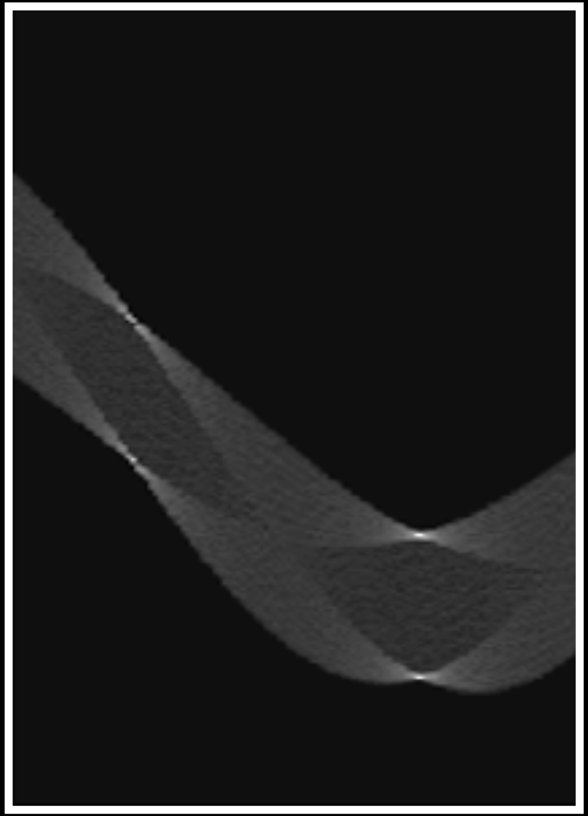
Votes

Bright value = high vote count

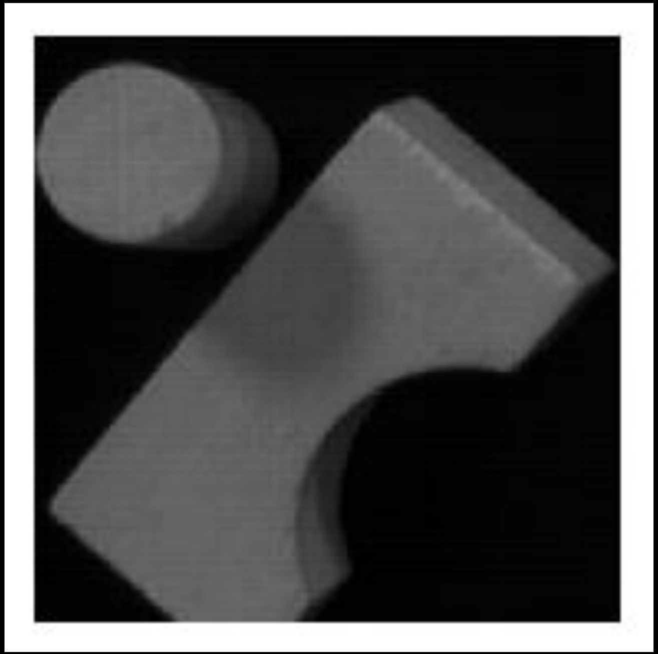
Black = no votes

Example: Hough transform of a square

Square :



Hough transform of blocks scene

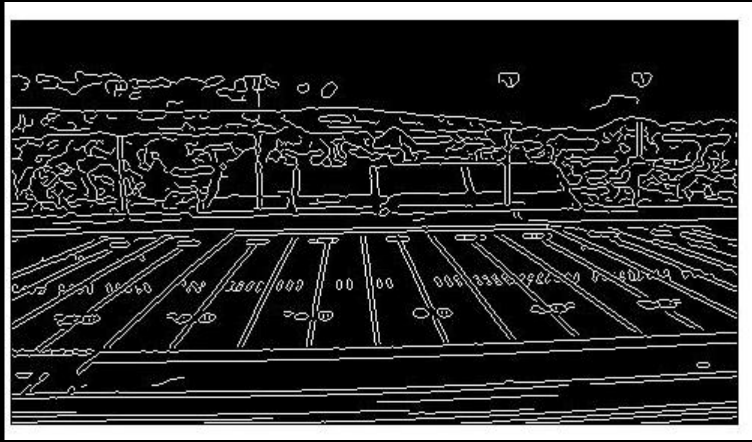


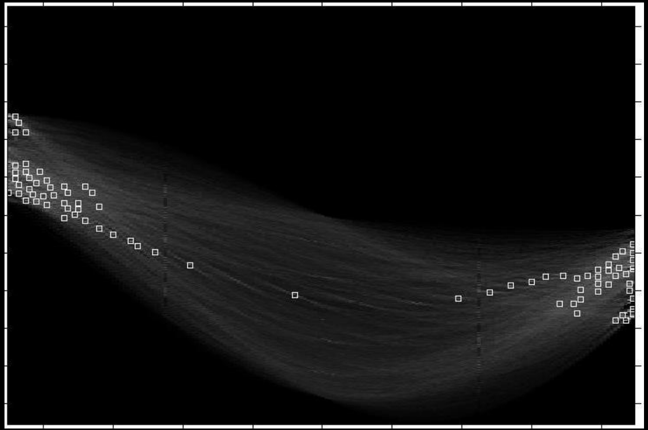
Hough Demo

```
% Hough Demo
pkg load image; % Octave only

%% Load image, convert to grayscale and apply Canny operator to find edge pixels
img = imread('shapes.png');
grays = rgb2gray(img);
edges = edge(grays, 'canny');

%% Apply Hough transform to find candidate lines
[accum theta rho] = hough(edges); % Matlab (use houghtf in Octave)
figure, imagesc(accum, 'XData', theta, 'YData', rho), title('Hough accumulator');

%% Find peaks in the Hough accumulator matrix
peaks = houghpeaks(accum, 100); % Matlab (use immaximas in Octave)
hold on; plot(theta(peaks(:, 2)), rho(peaks(:, 1)), 'rs'); hold off;
```

Hough Demo (contd.)

```
%% Find lines (segments) in the image
line_segs = houghlines(edges, theta, rho, peaks); % Matlab
figure, imshow(img), title('Line segments');
hold on;
for k = 1:length(line_segs)
   endpoints = [line_segs(k).point1; line_segs(k).point2];
   plot(endpoints(:, 1), endpoints(:, 2), 'LineWidth', 2, 'Color', 'green');
end
hold off;

%% Play with parameters to get more precise lines
peaks = houghpeaks(accum, 100, 'Threshold', ceil(0.6 * max(accum(:))), 'NHoodSize', [5 5]);
line_segs = houghlines(edges, theta, rho, peaks, 'FillGap', 50, 'MinLength', 100);
```





Showing longest segments found

Impact of noise on Hough

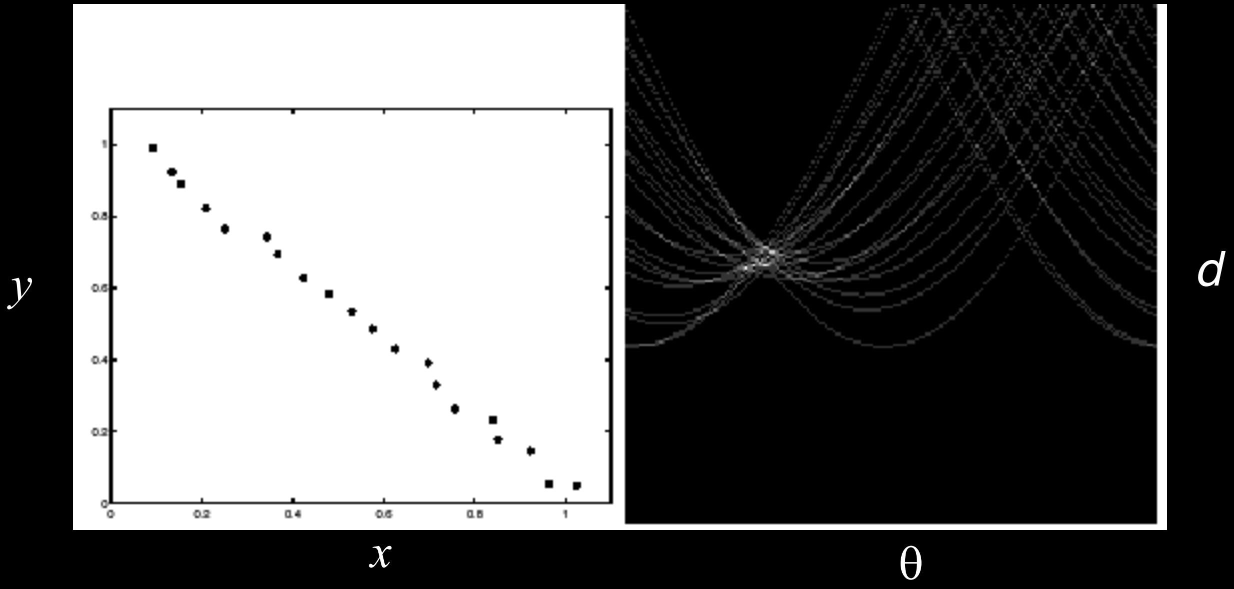


Image space edge coordinates

Votes

Impact of more noise on Hough

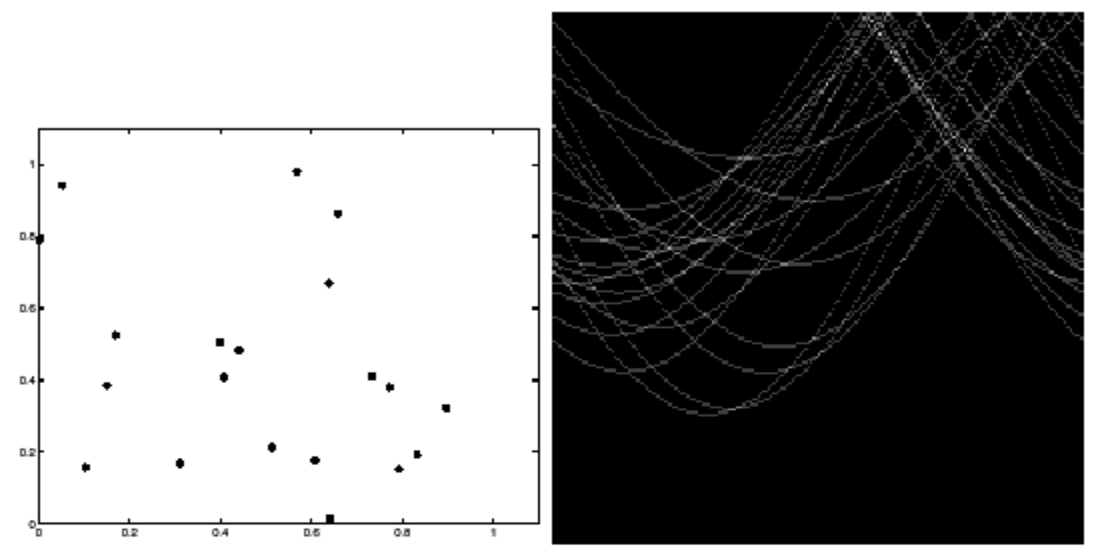


Image space edge coordinates

Votes

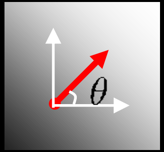
Extensions – using the gradient

1. Initialize H[d, θ]=0
2. For each ***edge*** point in $E(x, y)$ in the image

   θ = gradient at (x,y)

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

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



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

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

Extensions

Extension 2

Give more votes for stronger edges

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