EE 4211 Computer Vision

Lecture 7: Edge detection

Semester B, 2021-2022

# Schedules

Week	Date	Topics
1	Jan. 11 (face to face)	Introduction/Imaging
2	Jan. 18 (online)	Image enhancement in spatial domain
3	Jan. 25 (online)	Image enhancement in frequency domain (HW1 out) publish online on Jan. 26 noon.
4	Feb. 8 (online)	Morphological processing
5	Feb. 15 (online)	Image restoration (HW1 due) will publish results on today
6	Feb. 22 (online)	Midterm (no tutorials this week) will publish results on Mar. 8
7	Mar. 1 (online)	Edge detection (HW2 out) publish online on Mar. 2 noon.
8	Mar. 8 (online)	Image segmentation
9	Mar. 15 (online)	Face recognition with PCA, LDA (tutorial on segmentation) (HW2 due)
10	Mar. 22 (online)	Face recognition based on deep learning (Quiz) Image segmentation based on deep learning
11	Mar. 29 (online)	Object detection with traditional methods Object detection based on deep learning (tutorial on detection)
12	Apr. 5	Events / Public Holidays
13	Apr. 12 (online)	Invited project presentation and Summary

Lecture outline

- Overview of Image Segmentation

- Discontinuity Detection

    - Point Detection

    - Line Detection

    - Edge Detection

- Edge Linking & Boundary Detection

Image Segmentation

- ## Segmentation:
  - Split/separate/subdivide an image into regions or objects
  - To facilitate recognition, understand region of interest

- ## Challenges of Segmentation:
  - The definition of a region/object is problem-dependent
  - One of the most difficult task in image processing
  - Accuracy determines success or failure of application
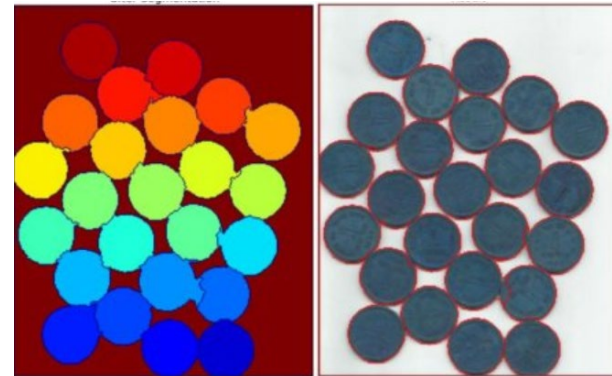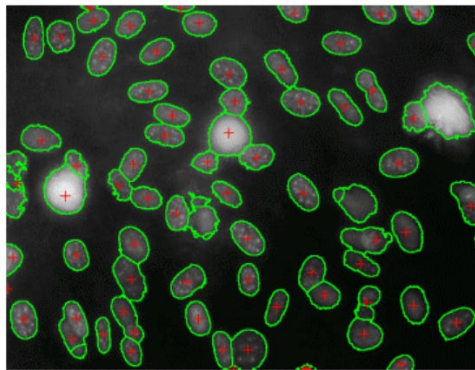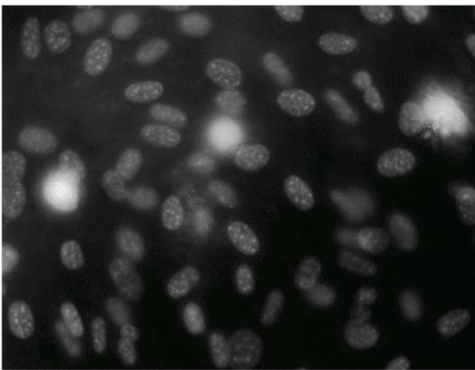
Segmentation – Categories

- Edge-based Segmentation

  - Finding boundary between adjacent regions

- Threshold-based Segmentation

  - Finding regions by grouping pixels of similar gray values

- Region-based Segmentation

  - Finding regions directly using growing or splitting

- Motion-based Segmentation

  - Finding regions by comparing successive frames of a video sequence to identify regions that correspond to moving objects

Edge-based Segmentation

- Finding discontinuities (sharp, local changes in intensity) as boundary of regions
- Discontinuities in digital images:
  - Point
  - Line
  - Edges
- Techniques
  - Point detection
  - Edge (pixel) detection
  - Edge formation from edge pixels – Edge linking, Hough Transform

Idea of Derivatives

- Local changes in intensity – detected using derivatives (1st and 2nd order)

- Derivatives –" terms of differences"

- In discrete domain, there are various ways to approximate these terms of differences

First Derivative Revisited

- Approximation:
  - Must be zero in areas of constant intensity
  - Must be nonzero at the onset of an intensity step/ramp
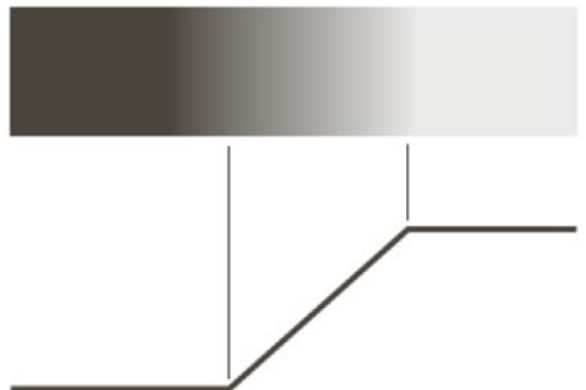 - Must be nonzero at points along an intensity ramp

$$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$$
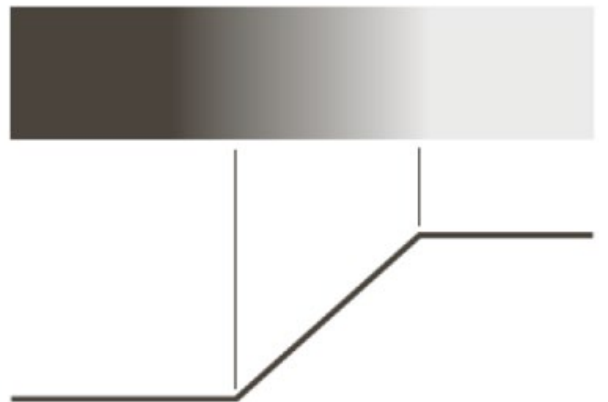
Second Derivative Revisited

- Approximation:
  - Must be zero in areas of constant intensity
  - Must be nonzero at the onset of an intensity step/ramp
  - Must be zero along intensity ramps

$$\frac{\partial^2 f}{\partial x^2} = f''(x) = f(x+1) + f(x-1) - 2f(x)$$

1st order vs. 2nd order

- Conclusions:

  - 1st order derivatives generally produce thicker edges

  - 2nd order derivatives have a stronger response to fine detail, such as thin lines, isolated points, noise

  - 2nd order derivatives produce a double-edge response at ramp transitions in intensity

  - The sign of the 2nd derivative can be used to determine whether a transition into an edge is from light to dark or dark to light

Computing Derivatives

- The simplest approach to compute 1st and 2nd derivatives at every pixel location in an image is to use spatial filters.
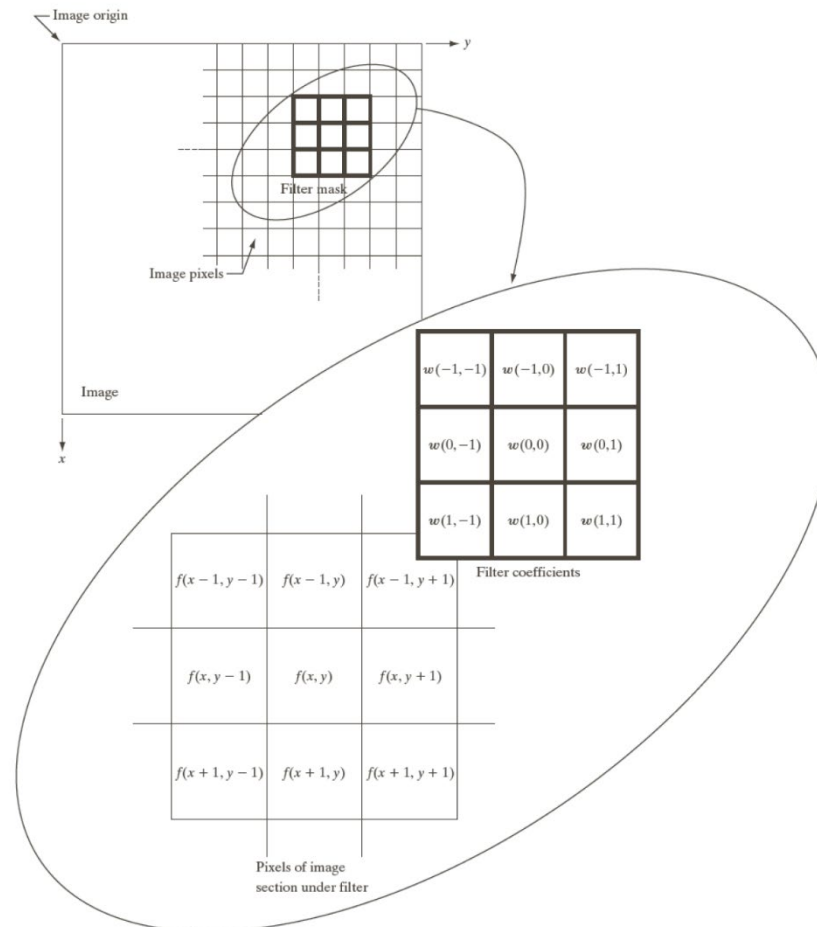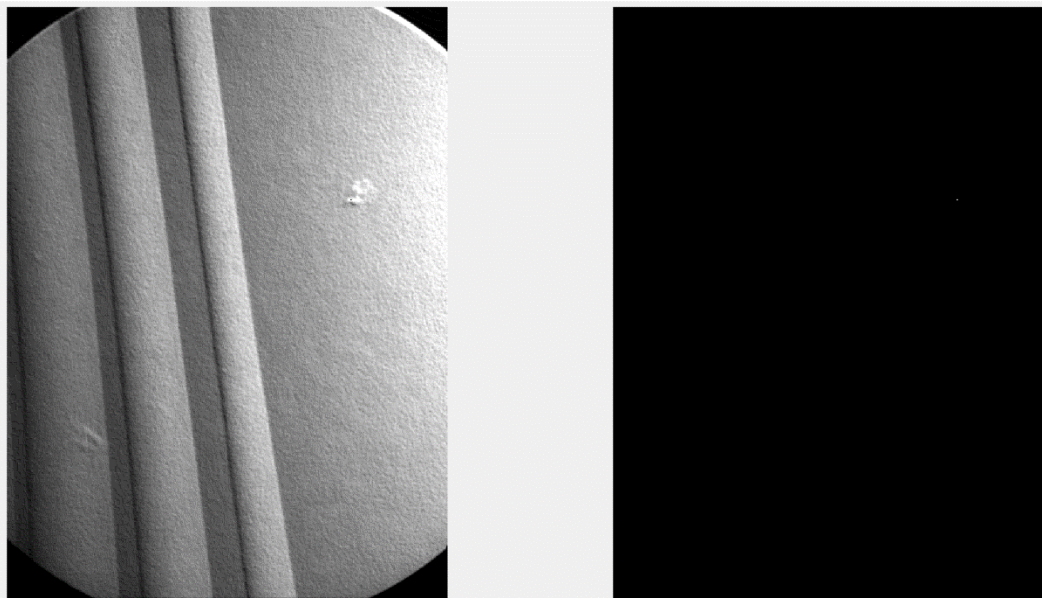
Point detection

- Point detection: detect an isolated point
- Laplacian filter is a good choice
- Thresholding after detection

$$|R| \geq T$$

| -1	-1	-1
-1	8	-1
-1	-1	-1


```
f=imread('1.tif');
w=[-1,-1,-1;-1,8,-1;-1,-1,-1];
g=abs(imfilter(double(f),w));
T=max(g(:));
g=g>=T;
figure;
subplot(1,2,1);imshow(f);
subplot(1,2,2);imshow(g);
```
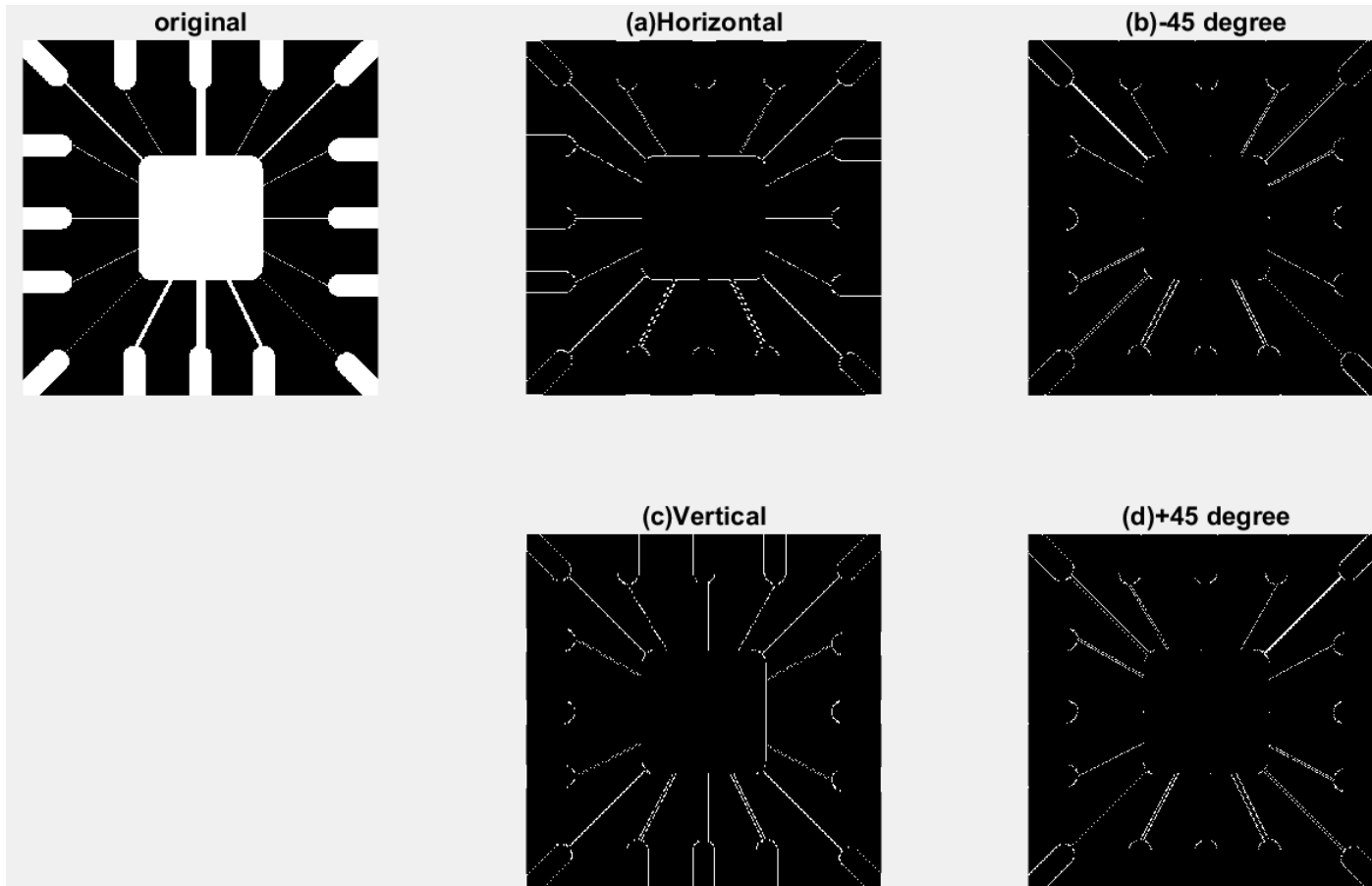
Line Detection

- Line detection: If each mask is moved around an image it would respond more strongly to lines in the mentioned direction.

- Expect 2nd derivatives to result in a stronger response and Produce thinner lines than 1st derivatives

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

2	-1	-1
-1	2	-1
-1	-1	2

- 45°

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

| -1	-1	2
-1	2	-1
2	-1	-1

+ 45°
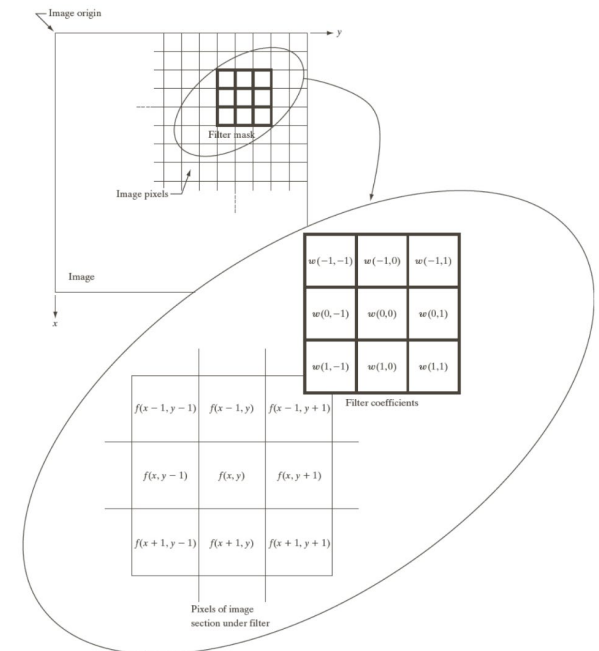
Example: Line Detection


```
f=imread('2.tif');
w1=[-1 -1 -1;2 2 2; -1 -1 -1];
g1=imfilter(double(f),w1);

w2=[2 -1 -1;-1 2 -1; -1 -1 2];
g2=imfilter(double(f),w2);

w3=[-1 2 -1;-1 2 -1; -1 2 -1];
g3=imfilter(double(f),w3);

w4=[-1 -1 2;-1 2 -1; 2 -1 -1];
g4=imfilter(double(f),w4);
```

```
figure;
subplot(2,3,1);imshow(f);title('original');
subplot(2,3,2);imshow(g1);title('(a)Horizontal');
subplot(2,3,3);imshow(g2);title('(b)+45 degree');
subplot(2,3,5);imshow(g3);title('(c)Vertical')
subplot(2,3,6);imshow(g4);title('(d)-45 degree')
```
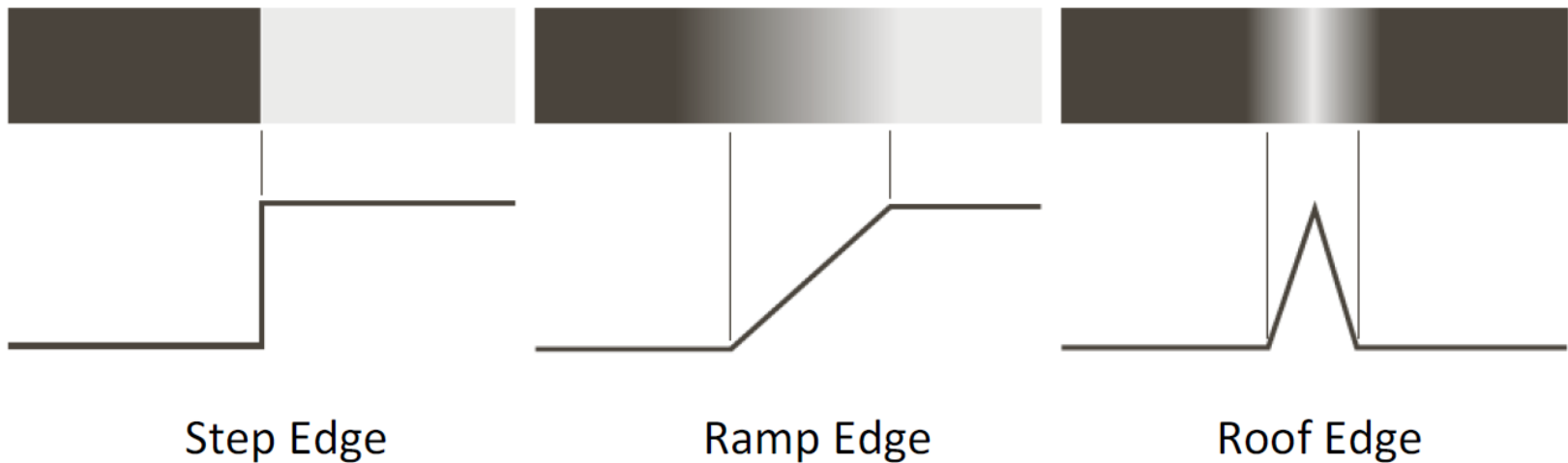
Edge detection

- Edge: The boundary between two regions with relatively distinct gray-level properties

- Points and lines are special cases of edges

- Assumption: The regions are sufficiently homogeneous, so that the transition between two regions can be determined on the basis of gray level discontinuities alone
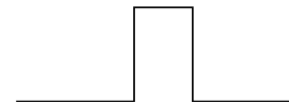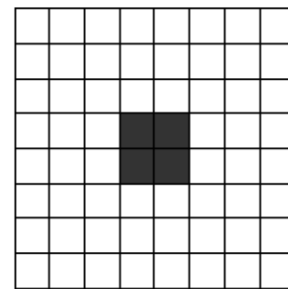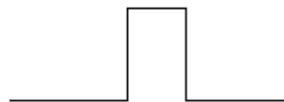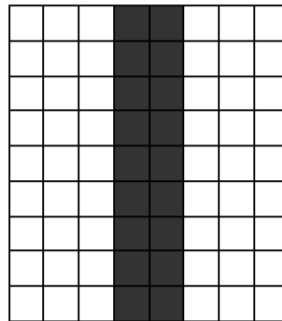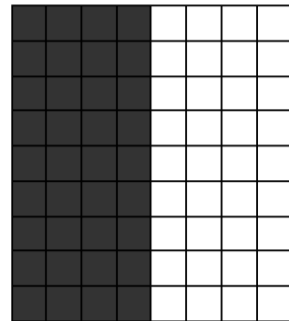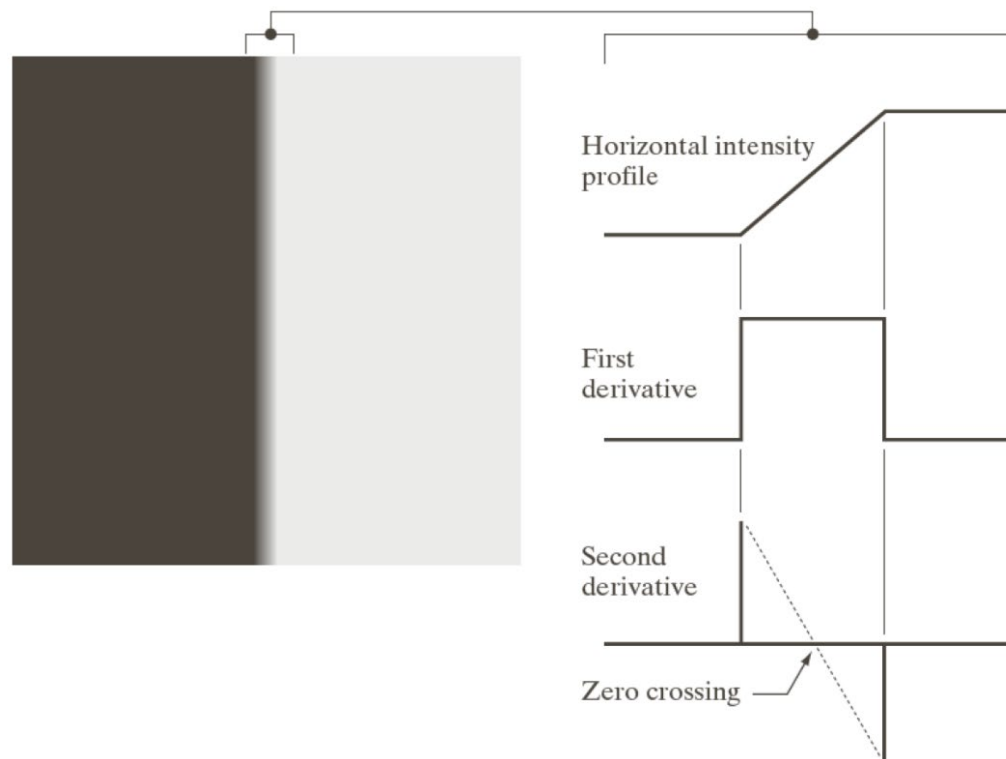
Edge detection

- Edge: boundary between two regions with relatively distinct gray levels
- Edge models

Step Edge   Ramp Edge   Roof Edge

Edge detection

- Magnitude of the 1st derivative can be used to detect presence of an edge
- 2nd derivative properties –
  - It produces two values for every edge
  - Its zero crossings can be used for locating the centers of thick edges

Horizontal intensity profile

First derivative

Second derivative

Zero crossing

Edge detection

- The magnitude of the first derivative can be used to detect an edge

- The sign (zero crossing) of the second derivative can be used to detect an edge.

- The same idea can be extended into 2-D. 2-D derivatives should be used.

Edge Operator Enhance Noise

- Even with very low noise, 1st derivative is useless at this noise level
- Second derivative is useless for even less noise

Original image   first derivative   Second derivative

Increasing noise
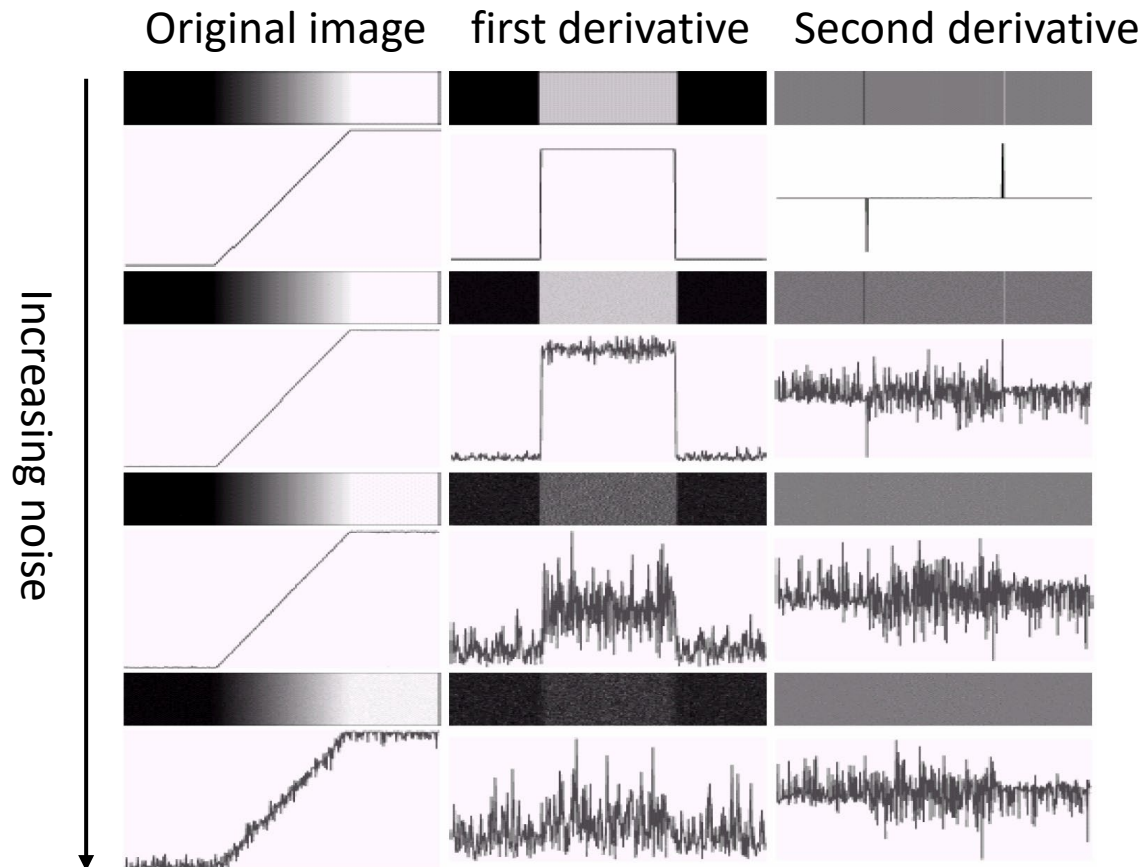
Steps in Edge Detection

- 3 fundamental steps performed in edge detection:

- Image smoothing for noise reduction
  - Noise reduces distinctiveness of edges

- Detection of edge points
  - Local operation that extracts all points that are potential candidates to become edge points

- Edge localization
  - Select from the candidate edge points, only the points that are true members of the set of points comprise an edge

Basic Edge Detection

- Image Gradient (1st order derivative)
  - The tool of choice for finding edge strength and direction at location (x,y) of an image, f, is the gradient, Δf:

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\[2mm] \dfrac{\partial f}{\partial y} \end{bmatrix}$$

  - The magnitude of vector Δf is denoted as:

$$\nabla f = mag(\nabla \mathbf{f}) = \sqrt{\left(G_x^{\,2} + G_y^{\,2}\right)} \qquad \nabla f \approx |G_x| + |G_y|$$

  - The direction of gradient vector is given by the angle,

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Basic Edge Detection

■ Gradient operators

$$g_x = f(x+1, y) - f(x, y)$$
$$g_y = f(x, y+1) - f(x, y)$$

−1	
1	

−1	1

■ The approach frequently used to approximate the magnitude of the gradient by absolute values:

$$M(x, y) \approx \left| g_x \right| + \left| g_y \right|$$

   ■ More attractive computationally, still preserves relative changes in intensity levels

   ■ No longer isotropic (invariant to rotation), but some masks make provision for this problem

Basic Edge Detection

- Roberts (cross-gradient operators)

- Prewitt & Sobel operators (horizontal, vertical and two diagonal masks)

- Prewitt masks are simpler to implement

- Sobel masks have better noise-suppression (smoothing)

Basic Edge Detection

- Prewitt Operator

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H_x^P = \begin{bmatrix} 1 \\ \mathbf{1} \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} 1 & \mathbf{1} & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

Basic Edge Detection

- Sobel Operator

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$H_x^S = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$
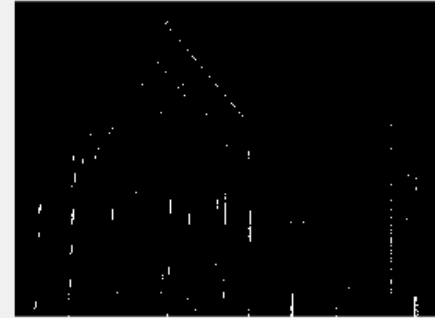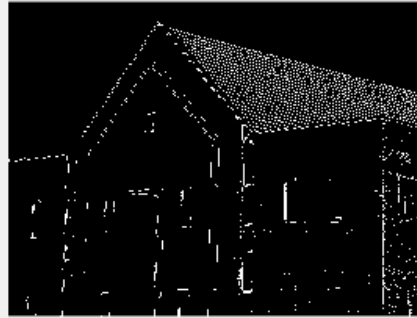
Thresholding Needed

- Output of edge operators do not result in binary edges (1s and 0s), thus thresholding is needed

- How to set threshold?
  - Trial and Error (depends on application)
  - According to edge magnitude distribution
  - E.g. Assuming only 5% pixels should be edge pixels, then threshold should be set at 95% percentile of edge magnitude
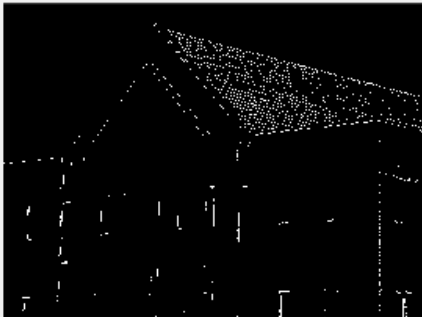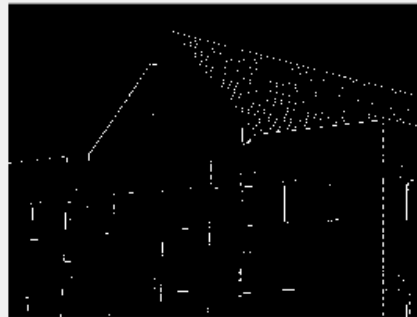
Example: Sobel edge detector

27

Example: Sobel edge detector

```matlab
%% edge detection with sobel  (Page 27 of EE4211_7)
f = imread('3.tif');
g_1 = edge(f,'sobel');
g_2= edge(f,'sobel',0.15,'vertical');
g_3 = edge(f,'sobel',0.15);
w=[-2 -1 0;-1 0 1;0 1 2];
g_4=imfilter(double(f),w,'replicate');
T=0.3*max(abs(g_4(:)));
g_4=g_4>=T;

w1=[0 1 2;-1 0 1;-2 -1 0];
g_5=imfilter(double(f),w1,'replicate');
T=0.3*max(abs(g_5(:)));
g_5=g_5>=T;

figure
subplot(2,3,1);imshow(f),title('a) Original');
subplot(2,3,2);imshow(g_1),title('b) sobel with automated threshold');
subplot(2,3,3);imshow(g_2),title('c) sobel with defined vertical threshold')
subplot(2,3,4);imshow(g_3),title('d) sobel with defined threshold');
subplot(2,3,5);imshow(g_4),title('e) sobel with 45 degree, threshold');
subplot(2,3,6);imshow(g_5),title('f) sobel with -45 degree, threshold');
```
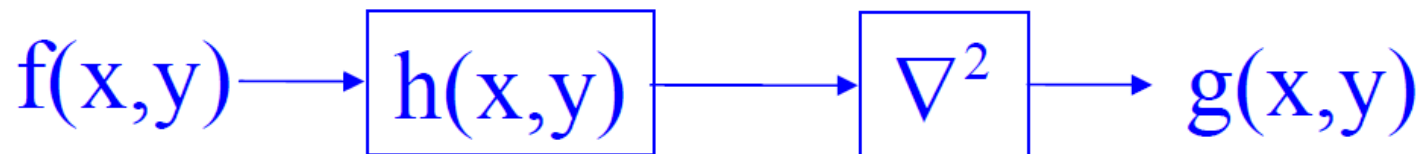
Problems with Basic Edge Detection

- Basic Edge Detection
  - Merely applying filter on image
  - No provision for edge characteristics and noise content

- Problems of previous approaches
  - Cannot locate edges precisely
  - Ramp edges can lead to many edge pixels detected depending on threshold T
  - High T – weak edges missing, Low T – detected edges too thick, noise points falsely detected

Laplacian

- Laplacian in its original form is not used for edge detection because:
  - It is very sensitive to noise
  - It's magnitude produces double edges
  - Unable to detect the direction of an edge

- To solve the first problem, the image is low-pass filtered before using the Laplacian operator.

$$f(x,y) \longrightarrow \boxed{h(x,y)} \longrightarrow \boxed{\nabla^2} \longrightarrow g(x,y)$$

$$h(x,y) = e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}$$

LoG

$$\nabla^2\big(f(x,y) * h(x,y)\big) = f(x,y) * \boxed{[\nabla^2 h(x,y)]}$$
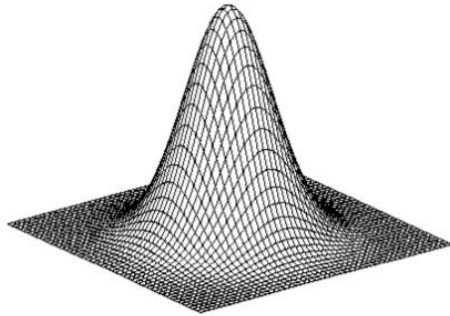
30

Laplacian of Gaussian (LoG) Edge Detector

- Also known as the Marr-Hildreth edge detector

- A Gaussian lowpass function smoothens the image, before the Laplacian if used to find the edges

  - Gaussian Low Pass Filter

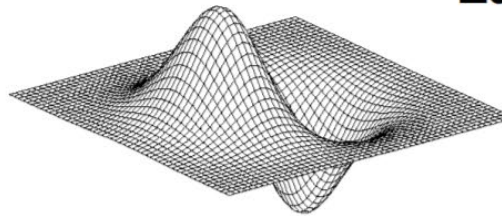$$h(x,y) = e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}$$

  - Taking the Laplacian

LoG

$$\nabla^2\big(f(x,y) * h(x,y)\big) = f(x,y) * \boxed{[\nabla^2 h(x,y)]}$$

$$= \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{\frac{x^2+y^2}{2\sigma^2}}$$

Laplacian of Gaussian Operator

Laplacian of Gaussian

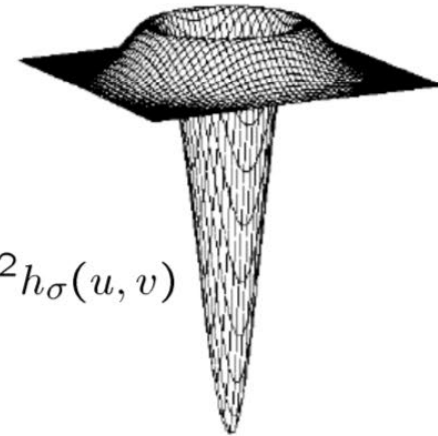**Gaussian**   **derivative of Gaussian**

$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$ $$\nabla h_\sigma(u,v)$$ $$\nabla^2 h_\sigma(u,v)$$
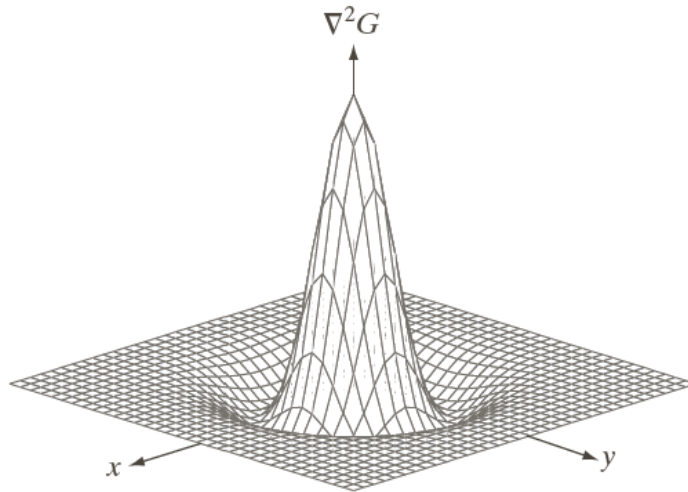
- ∇^2 is the Laplacian operator:

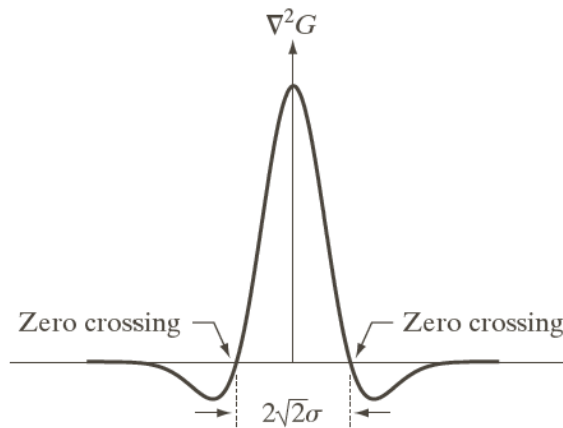$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

LoG Filter

Three dimension plot of the negative of LoG

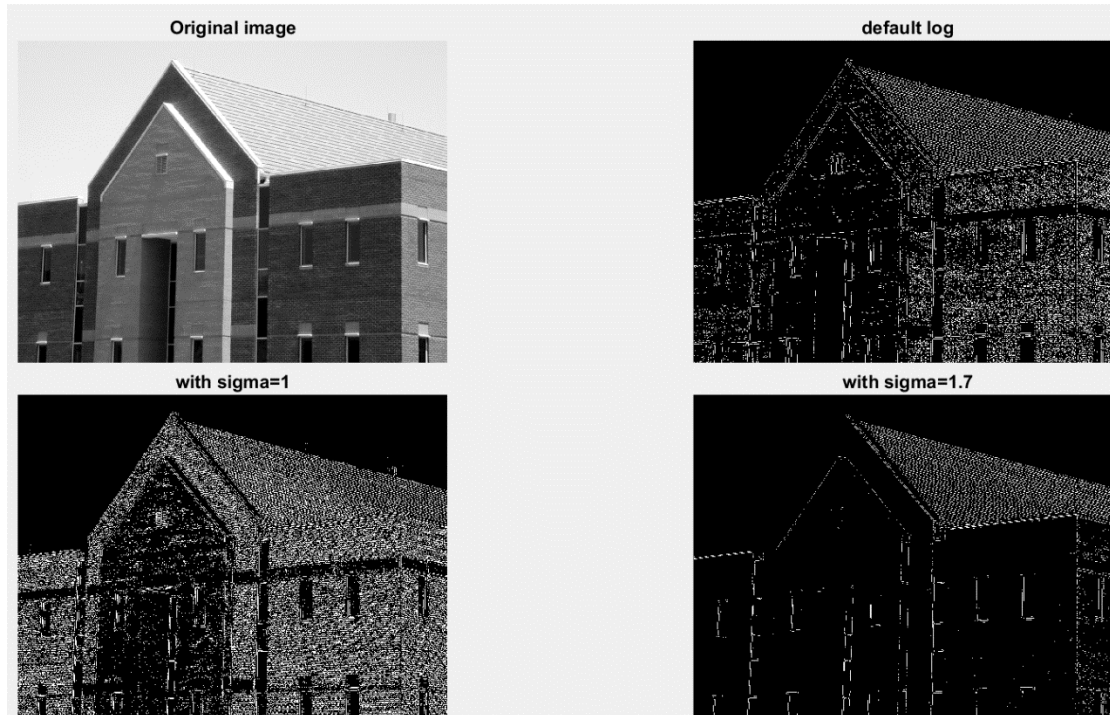Negative LoG displayed as an image

Cross section

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

5*5 mask approximation to the filter

Example: LoG Filter


```
%% edge detection with log

f = imread('5.tif');
[g_log_default,tlog] = edge(f,'log');

g_log_1 = edge(f,'log',0.02,1);%0.02 means threshold
g_log_2 = edge(f,'log',0.02,1.7);
subplot(2,2,1);imshow(f);title('Original image');
subplot(2,2,2);imshow(g_log_default);title('default log');
subplot(2,2,3);imshow(g_log_1);title('with sigma=1');
subplot(2,2,4);imshow(g_log_2);title('with sigma=1.7');
```

Canny Edge Detector

- More complex algorithm, but far superior performance in general to all the edge detectors discussed thus far.

- Approach based on 3 basic objectives:

  - Low error rate – all (or as many) edges should be found

  - Edge points should be well localized – edges as close as possible to true edges

  - Single edge point response – only one true edge point returned

Canny Edge Detector

- Smooth the image with a Gaussian filter with σ

- Compute gradient magnitude and direction at each pixel of the smoothed image

- Apply non-maximal suppression to the gradient magnitude image

- Use double thresholding (T_H and T_L) and connectivity analysis to detect and link edges

Non-maximal suppression

The gradient $M(x, y)$ typically contains wide ridge around local maxima. Next step is to thin those ridges.

Nonmaxima suppression:

Let $d_1, d_2, d_3,$ and d_4 denote the four basic edge directions for a 3×3 region: horizontal, $-45°$, vertical, $+45°$, respectively.

1. Find the direction d_k that is closest to $\alpha(x, y)$.

2. If the value of $M(x, y)$ is less than at least one of its two neighbors along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$

Non-maximal suppression

- Eliminate all but local maxima in magnitude of gradient
- At each pixel, look along direction of gradient: if either neighbor is bigger, set to zero
- In practice, quantize direction to horizontal, vertical and two diagonals
- Result: thinned edge image

Double thresholding ($T_H > T_L$)

- **T**he final operation is to threshold $g_N(x, y)$ to reduce false edge points.

Hysteresis thresholding:

$$g_{NH}(x, y) = g_N(x, y) \geq T_H$$

$$g_{NL}(x, y) = g_N(x, y) \geq T_L$$

and

$$g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y)$$

Double thresholding ($T_H > T_L$)

- Pixels with values greater T_H are strong edge pixels

- Pixels with values greater than T_L are weak edge pixels

- **Edge Linking**
  - Locate a pixel p belonging to a strong edge
  - Find weak pixels that are connected to p
  - Append these weak pixels to p

Example: Canny Filter

Original image | default canny
with sigma=1.2 | with sigma=3
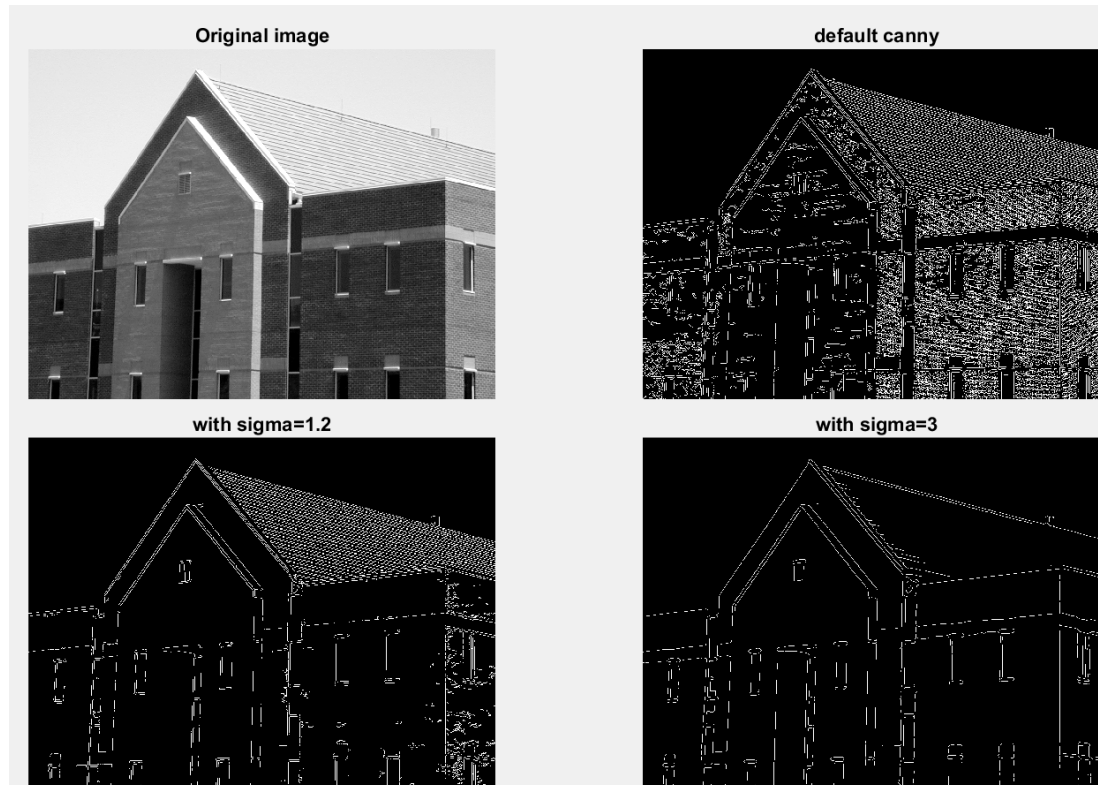
```
%% edge detection with canny
f = imread('3.tif');
[g_canny_default,tlog] = edge(f,'canny');

g_canny_1 = edge(f,'canny',0.2,1.2);%0.2 means threshold
g_canny_2 = edge(f,'canny',0.2,3);
subplot(2,2,1);imshow(f);title('Original image');
subplot(2,2,2);imshow(g_canny_default);title('default canny');
subplot(2,2,3);imshow(g_canny_1);title('with sigma=1.2');
subplot(2,2,4);imshow(g_canny_2);title('with sigma=3');
```

BW = **edge**(I,'canny',THRESH) specifies sensitivity thresholds for the
Canny method. THRESH is a two-element vector in which the first element
is the low threshold, and the second element is the high threshold. If
you specify a scalar for THRESH, this value is used for the high
threshold and 0.4*THRESH is used for the low threshold. If you do not
specify THRESH, or if THRESH is empty ([]), **edge** chooses low and high
values automatically.

Lecture outline

- Overview of Image Segmentation
- Discontinuity Detection
    - Point Detection
    - Line Detection
    - Edge Detection
- Edge Linking & Boundary Detection

Edge Linking and Boundary Detection

- Edge detection typically is followed by linking algorithms designed to assemble edge pixels into meaningful edges and/or region boundaries

- Two approaches to edge linking
    - Local processing
    - The Hough Transform

Local Processing

- Analyze the characteristics of pixels in a small neighborhood about every point (x,y) that has been declared an edge point

- All points that similar according to predefined criteria are linked, forming an edge of pixels.

  - Establishing similarity: (1) the strength (magnitude) and (2) the direction of the gradient vector.

  - A pixel with coordinates (s,t) in Sxy is linked to the pixel at (x,y) if both magnitude and direction criteria are satisfied.

Local Processing

Let S_{xy} denote the set of coordinates of a neighborhood centered at point (x, y) in an image. An edge pixel with coordinate (s, t) in S_{xy} is similar in *magnitude* to the pixel at (x, y) if

$$\left| M(s,t) - M(x,y) \right| \leq E$$

An edge pixel with coordinate (s, t) in S_{xy} is similar in *angle* to the pixel at (x, y) if

$$\left| \alpha(s,t) - \alpha(x,y) \right| \leq A$$

Local Processing: Steps (1)

- 1. Compute the gradient magnitude and angle arrays, M(x,y) and $\alpha(x,y)$, of the input image f(x,y)

- 2 Form a binary image, g, whose value at any pair of coordinates (x,y) is given by

$$g(x, y) = \begin{cases} 1 & \text{if } M(x, y) > T_M \text{ and } \alpha(x, y) = A \pm T_A \\ 0 & \text{otherwise} \end{cases}$$

T_M : threshold A : specified angle direction
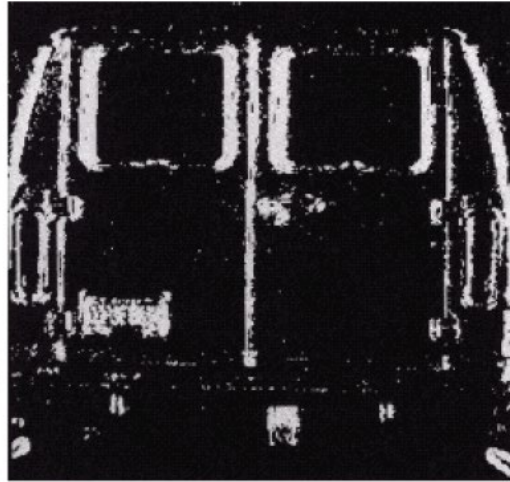
T_A : a "band" of acceptable directions about A

Local Processing: Steps (2)

- 3 Scan the rows of g and fill (set to 1) all gaps (sets of 0s) in each row that do not exceed a specified length, K.

- 4 To detect gaps in any other direction, rotate g by this angle and apply the horizontal scanning procedure in step 3.
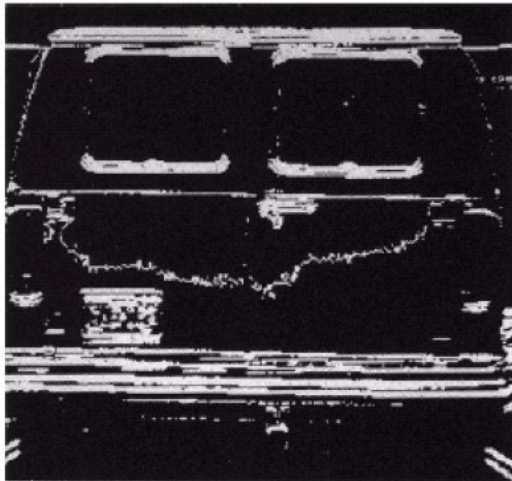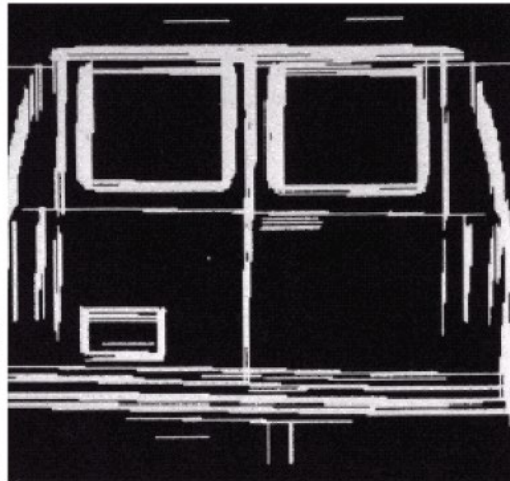
Example

Input image

G_y

G_x

Result of edge linking

The Hough Transform

- **The Hough transform** is a general technique for identifying the locations and orientations of certain types of features in a digital image.

- Developed by Paul Hough in 1962 and patented by IBM, the transform consists of parameterizing a description of a feature at any given location in the original image's space.
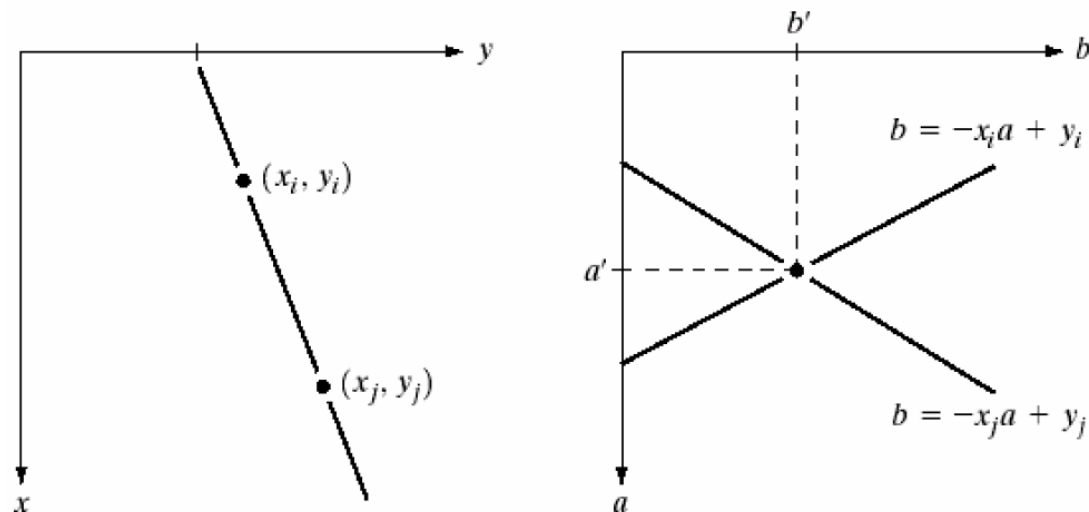
Hough Transform

(x_i, y_i): all the lines passing this point $y_i = ax_i + b$

$b = y_i - ax_i$: point (x_i, y_i) maps to a single line in ab plane.

Another point (x_j, y_j) also has a single line in ab plane

$\quad b = y_j - ax_j$

a' and b' are the slope and intercept of the line containing
both (x_i, y_i) and (x_j, y_j).

- Several points in *x,y* domain with given *a,b* can be represented by one point in a, b domain
- Several lines in the xy space can be represent as a single line in ab space

- If we want to know how many points exit in one line in *x,y* domain, we can accumulate the values in *a b* domain with the corresponding *a', b'*.
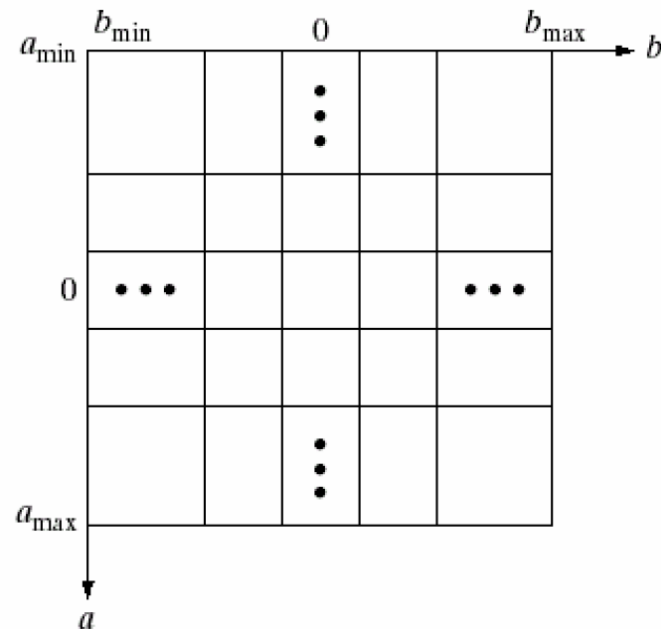
Hough Transform

(a_{min}, a_{max}): expected range of slope
(b_{min}, b_{max}): expected range of intercepts
A(i,j): the number of points in the cell at coordinates (i,j) in the ab plane.

For every point in the image plane, we let the value of a equal each of the allowed subdivisions and find the corresponding b from $b = y_i - ax_i$ If for a_p we get b_q the $A(p,q)$ is incremented.
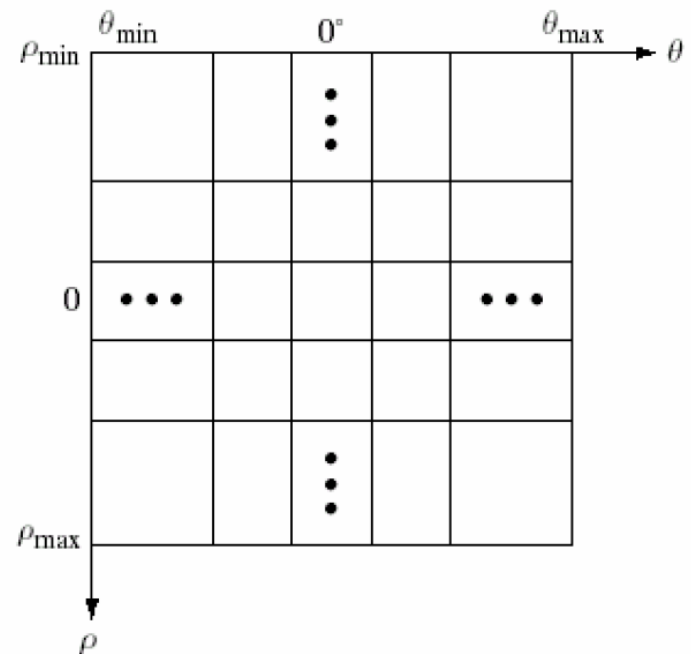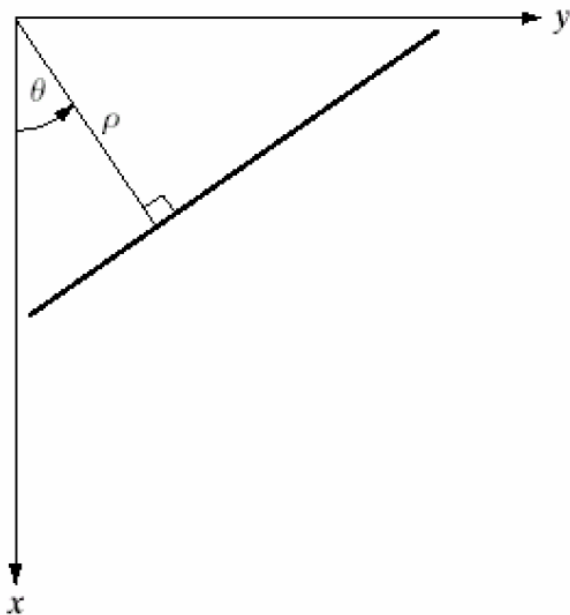
Hough transform

- The cell with the largest value shows the parameters of the line that contains the maximum number of points.

- Problem with this method: *a* approaches infinity as the line gets perpendicular to the *x* axis. (if a line is perpendicular to x axis, then this line is represented x=M. a->infinity)

- Solution: use the representation of the line as:

$$x \cos \theta + y \sin \theta = \rho$$
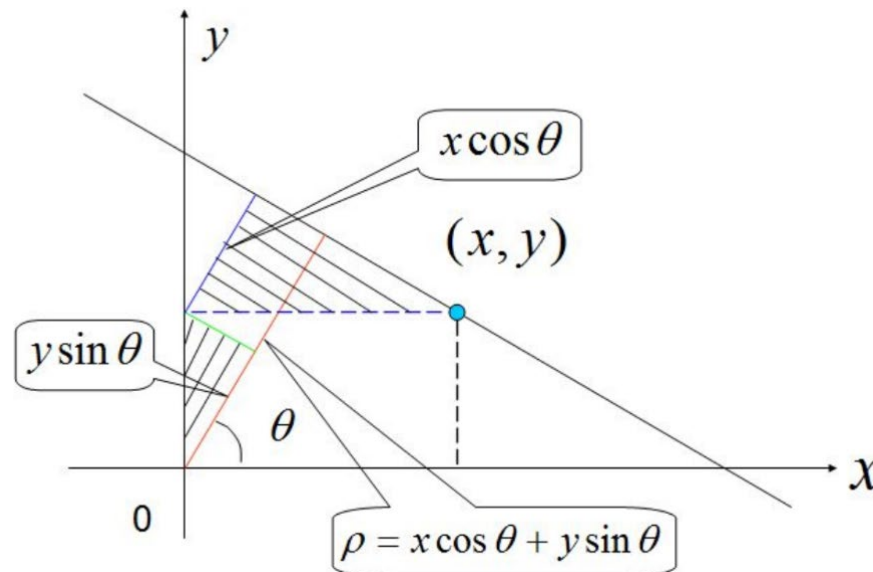
Hough transform

- A point in *xy* plane is mapped into a sinusoidal curve in ρθ plane.

- In that case, we can transfer "finding numbers of points in a line in xy plane" to "the accumulated values in corresponding ρ θ subcell in ρ θ plane"

Edge-linking Based on the Hough Transform

- Obtain a binary edge image using any techniques
- Specify subdivisions in ρ θ plane
- Examine the number of intersects at each cell in the ρ θ plane;
- Bridge the gap based on continuity
  - The gap in a line associate with a given cell is bridged if the length of the gap is less than a threshold

https://www.youtube.com/watch?v=4zHbI-fFIlI

Example: Hough Transform
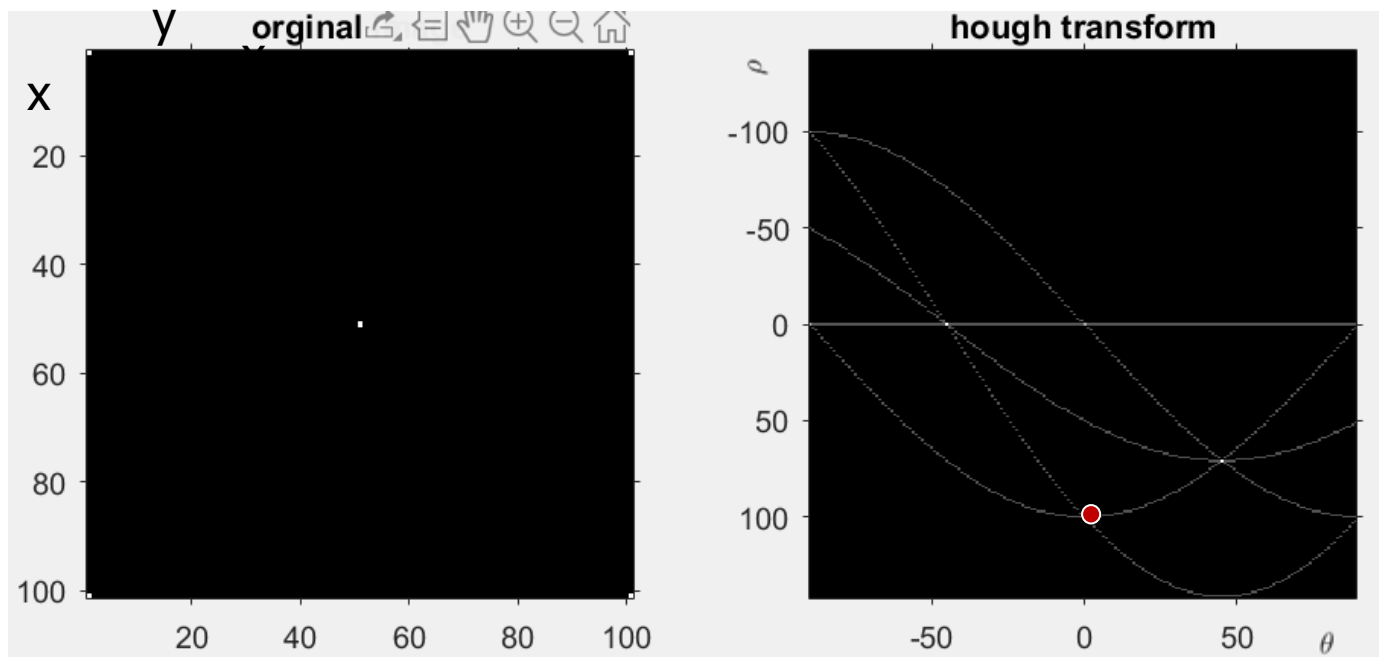

```
%% example for hough transform (Page 55 of EE 4211_7)
f=zeros(101,101);
f(1,1)=1;
f(101,1)=1;
f(1,101)=1;
f(101,101)=1;
f(51,51)=1;
[H,T,R]=hough(f);
figure;
subplot(1,2,1);imshow(f);title('orginal image');axis on;axis square;
subplot(1,2,2);imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit')
axis on,axis square;
```

Example: Hough Transform

- In order to link a line in the original image, we use Hough transform and identify the largest values in the accumulator, then identify the corresponding lines
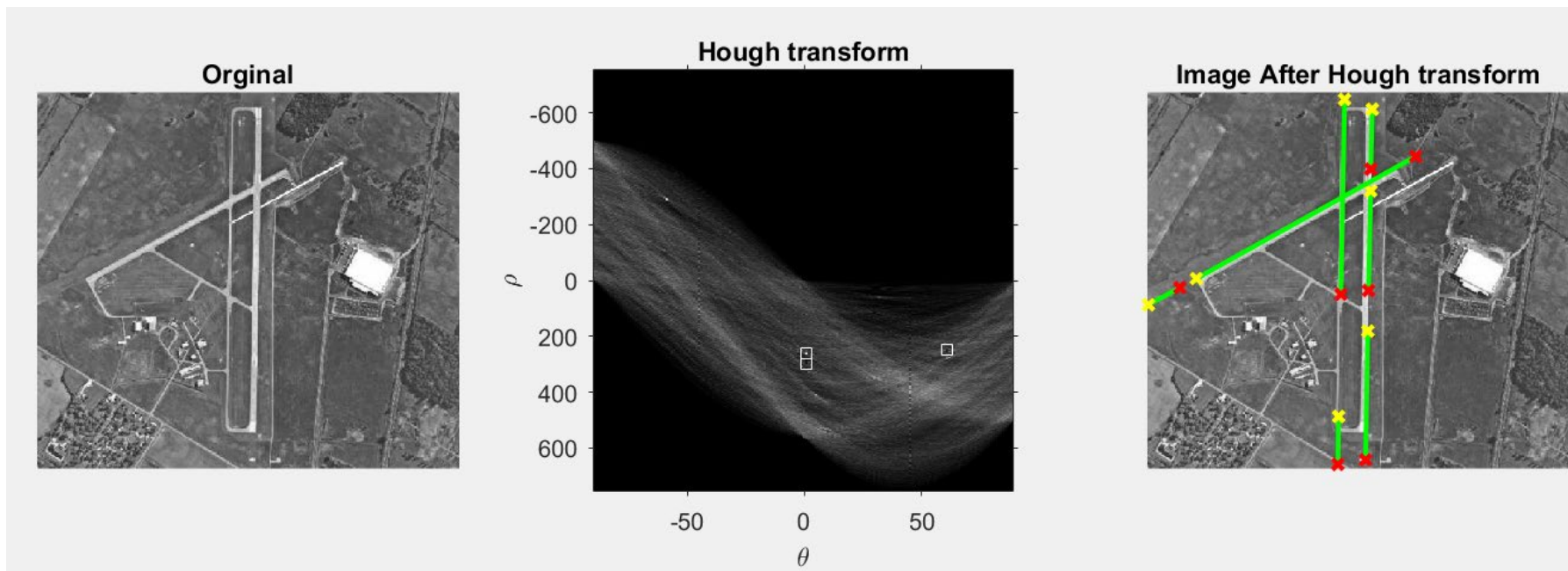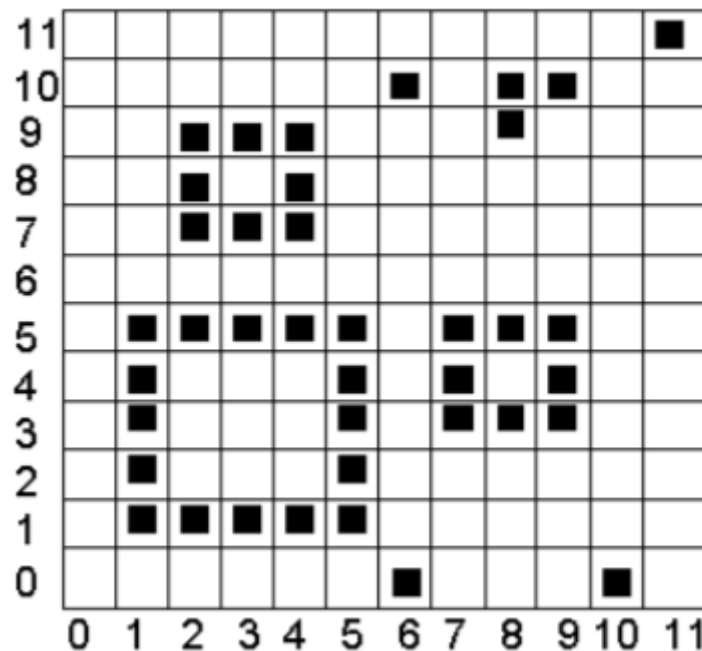
Example: Hough Transform

```matlab
clear all;
I  = imread('6.tif');
figure;
subplot(1,3,1);
imshow(I); title('Orginal');
BW = edge(I,'canny');%Canny detect boundary
[H,T,R] = hough(BW);%H hough tranform matrix,I,R angle and radius
subplot(1,3,2);
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit');%hough transform
xlabel('\theta'), ylabel('\rho');title('Hough transform');
axis on,axis square,hold on;
P  = houghpeaks(H,3);
x = T(P(:,2));
y = R(P(:,1));
plot(x,y,'s','color','white');
lines=houghlines(BW,T,R,P);
subplot(1,3,3);
imshow(I), title('Image After Hough transform');hold on;
for k = 1:length(lines)
xy = [lines(k).point1; lines(k).point2];
 plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
 end
```

57

Example: Hough

▪ If we apply the Hough transform on the image below, what would be the maximum value for the accumulator cell in the (ρ, θ) space? What is the corresponding (ρ, θ) value?

The maximum value is 8, corresponding to the horizontal line with vertical coordinate 5, because there are 8 points lying on this line. This line has the parameter ρ = 5, θ = pi/2.

Example

■ If we apply the Hough transform on the image below, what would be the maximum values for the accumulator cell in the (ρ, θ) space? What are the corresponding (ρ, θ) values.
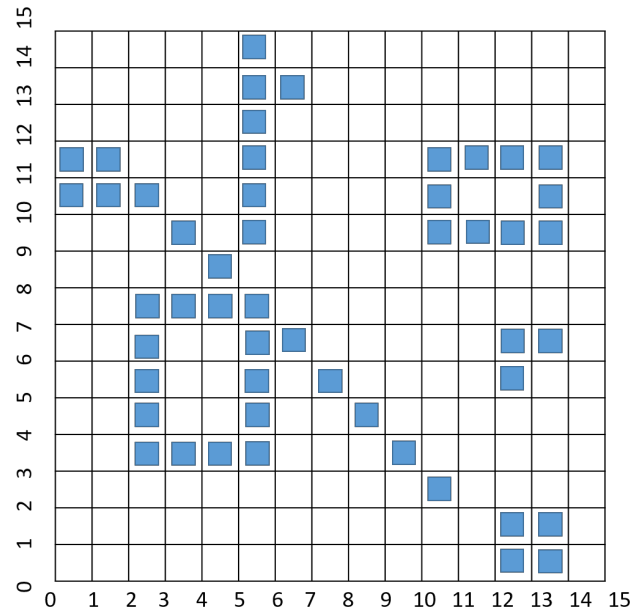
Solution:
The maximum value is 11. There are two lines correspond to this values, with ρ = 12/sqrt(2), θ = pi/4 or ρ = 5, θ = 0.

Review: Edge-based Segmentation

- Finding discontinuities (sharp, local changes in intensity) as boundary of regions

- Discontinuities in digital images (Spatial filters)
  - Point (Laplacian)
  - Line (horizontal, -45 degree, vertical, 45 degree)
  - Edges (Roberts, Prewitt, Sobel, LOG, Canny)

- Techniques
  - Point detection
  - Edge (pixel) detection
  - Edge formation from edge pixels – Edge linking, Hough Transform

Project information(Illustrated on Lecture 3)

- Topics: Image segmentation and object detection
- Four students form a group (file sharing, finalized before Mar. 8)

https://docs.google.com/spreadsheets/d/1b9LiFO4XwFJCpn-eZOOUU18IEQRobP4cRcJid1JtZlI/edit?usp=sharing

- Images: will be provided through Kaggle and Kaggle link will be published Mar. 8
- Codes:
  - Basic codes including the traditional ones and deep learning ones will be provided.
  - Please modify these codes to achieve better performance
- Submission and Evaluation:
  - Codes and reports should be submitted
  - Report writing(will illustrate the writing in tutorial session), results
  - Excellent ones will be invited to present their work in the last lecture and get extra bonus for the marks

Project Information

- Updated project information can be found in

https://docs.google.com/spreadsheets/d/1b9LiFO4XwFJCpn-eZOOUU18IEQRobP4cRcJid1JtZlI/edit#gid=0

- As mentioned on Lecture 3, we will finalize the project group information on Mar. 8. Please specify your topics in the excel file. Please choose detection or segmentation, first come, first served. (8 groups for detection and 8 groups for segmentation)
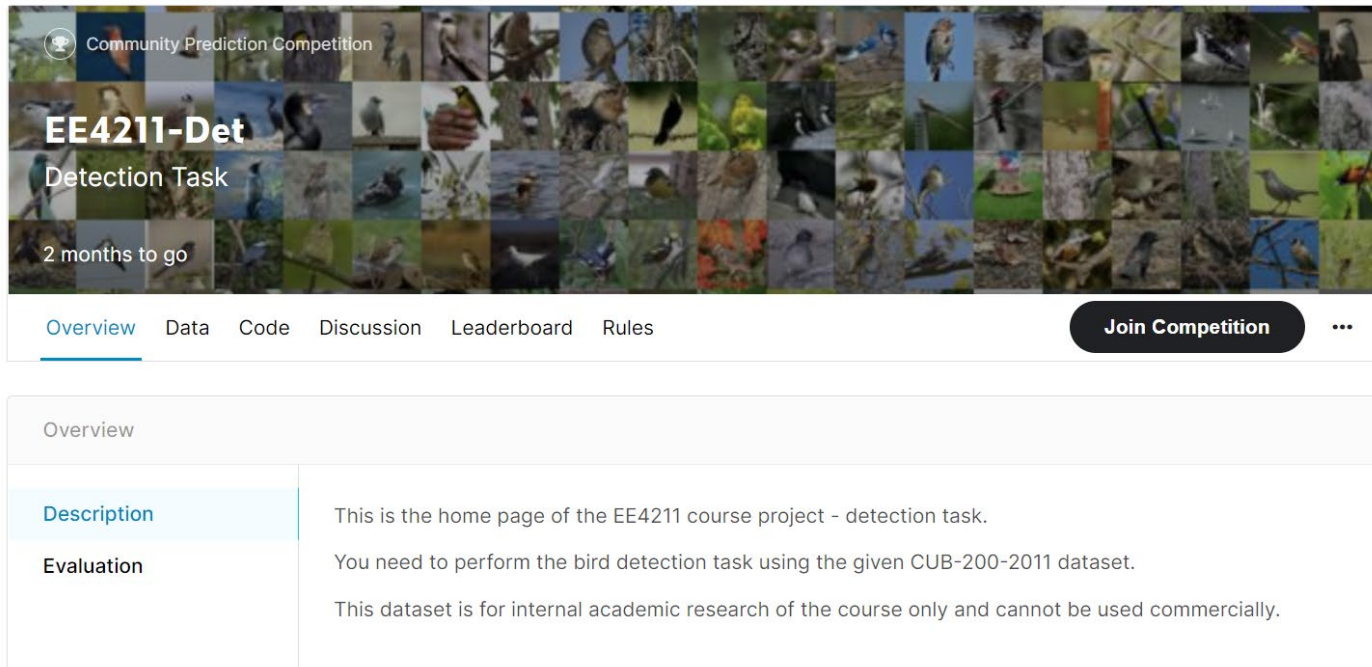
Project Information

- Segmentation https://www.kaggle.com/c/ee4211-seg/overview

- Based on the schedule, TA will illustrate the segmentation codes for you to modify (both traditional one and deep learning one), illustrate how to upload the results and rules on Mar. 15

- You can download the datasets to have a try by yourself before the tutorial

63

Project Information

- Detection https://www.kaggle.com/c/ee4211-object-detection/overview

- Based on the schedule, TA will illustrate the segmentation codes for you to modify (both traditional one and deep learning one), illustrate how to upload the results and rules on Mar. 29

Workflow

- With the given data and codes, please get familiar with the data and task
- Run the provided codes with the data and see the results
- Modification of the methods to improve the performance
- Re-evaluate the methods (TA will illustrate it the tutorials)
- Write the reports (will illustrate it later)