Image Processing
INT3404 20
Week 5: Feature extraction

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.

Email: ngocdiep@vnu.edu.vn

Slide & code: https://github.com/chupibk/INT3404_20

Schedule

Week	Content	Homework
1	Introduction	Set up environments: Python 3, OpenCV 3, Numpy, Jupyter Notebook
2	Digital image – Point operations Contrast adjust – Combining images	HW1: adjust gamma to find the best contrast
3	Histogram - Histogram equalization – Histogram-based image classification	Self-study
4	Spatial filtering - Template matching	Self-study
5	Feature extraction Edge, Line, and Texture	Self-study
6	Morphological operations	HW2: Barcode detection → Require submission as mid-term test
7	Filtering in the Frequency domain Announcement of Final project topics	Final project registration
8	Color image processing	HW3: Conversion between color spaces, color image segmentation
9	Geometric transformations	Self-study
10	Noise and restoration	Self-study
11	Compression	Self-study
12	Final project presentation	Self-study
13	Final project presentation Class summarization	Self-study

Recall week 4: Neighborhood

	(x-1, y)	
(x, y – 1)	**(x, y)**	(x, y+1)
	(x+1, y)	

4 - neighbors

(x-1, y-1)	(x-1, y)	(x-1, y+1)
(x, y – 1)	**(x, y)**	(x, y+1)
(x+1, y-1)	(x+1, y)	(x+1, y+1)

8 - neighbors

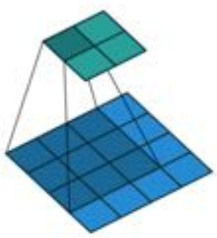
2 pixels *p=(x, y)* and *q=(u,v)*

Euclidean distance: $D_e(p,q) = \left[(x-u)^2 + (y-v)^2\right]^{\frac{1}{2}}$

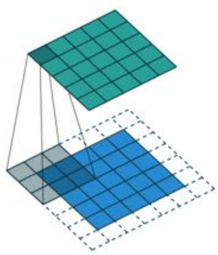
City-block distance: $D_4(p,q) = |x-u| + |y-v|$

Chessboard distance: $D_8(p,q) = \max(|x-u|, |y-v|)$

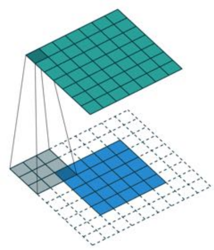
Recall week 4: Convolution/correlation



”valid”



”same”



”full”

Image size: MxN
Kernel size: mxn

Output: (M-m+1) x (N – n + 1) M x N (M + m – 1) x (N +n – 1)

Illustration credit: https://github.com/vdumoulin/conv_arithmetic

Recall week 4: Padding borders

- Pad a constant value (black)
- Wrap around (circulate the image)
- Copy edge (replicate the edges' pixels)
- Reflect across edges (symmetric)




Recall week 4: Spatial filtering

- Average filter (Box filter)
- Gaussian filter
- Mean filter
- Unsharp masking

Week 5: Feature extraction

Edge, Line, Texture

Edge detection

What is an edge?

- An edge = a significant local change in the image intensity

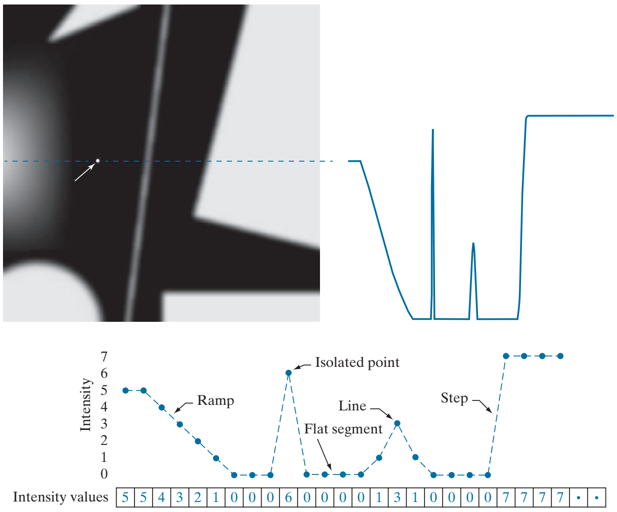


Image credit: Gonzalez et.al., Fig. 10.2

Gradient

- Gradient of a function indicates how strong the function increases.
  - For 1-dimension function: $f(x) = x^2$

$$Grad(x) = \frac{\partial f(x)}{\partial(x)} = 2x$$

  - Grad(2)=4 indicates the the increasing direction of the function is to the right.
  - Grad(-1)=-2 indicates the increasing direction of the function is to the left.

First-order and second-order derivatives

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

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

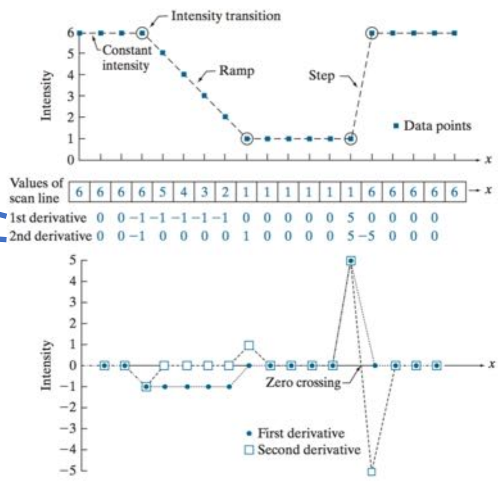


Image credit: Gonzalez, Fig. 3.44

Edge detection using derivatives

(1) Detecting the local maxima or minima of the first derivative

(2) Detecting the zero-crossings of the second derivative

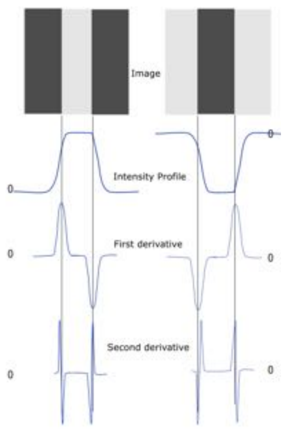


Image credit: MIPAV

Gradient of 2D discrete function

- Gradient of a 2-dimension function is calculated as follows:

$$Grad(x,y) = \frac{\partial f(x,y)}{\partial x}\vec{i} + \frac{\partial f(x,y)}{\partial y}\vec{j}$$

- The gradient is approximated as follows (first-order derivative) :

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y), \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y)$$

Gradient

- The magnitude of gradient indicates the strong of edges:

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

- Gradient computation procedure:
  - Calculate column gradient
  - Calculate row gradient
  - Calculate final gradient by the above function

Various kernels used to compute the gradient

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

A 3x3 region of an image

−1	0		0	−1
0	1		1	0

Roberts

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$
$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

−1	−1	−1		−1	0	1
0	0	0		−1	0	1
1	1	1		−1	0	1

Prewitt

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$
$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

−1	−2	−1		−1	0	1
0	0	0		−2	0	2
1	2	1		−1	0	1

Sobel

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Pixel Difference masks

Column mask

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

Row mask

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Pixel difference example

Original Image

Column edges

Row edges

Final edges

Robert mask

- Roberts masks calculate gradient from two diagonals

Column

$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Row

$$\begin{bmatrix} -1 & & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Robert mask example

Original Image

Column edges

Row edges

Final edges

Prewitt mask

Column

$$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Row

$$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Prewitt mask

Original Image

Row edges

Column edges

Final edges

Sobel mask

Column

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Row

$$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel filter example

Original Image

Column

Row

Combination

Laplace gradient

- Laplace edge in a continuous domain

$$G(x,y) = -\left(\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}\right)$$

- In a discrete domain, Laplace edge is approximated by

$$\begin{aligned} G(x,y) &= [f(x,y) - f(x,y-1)] - [f(x,y+1) - f(x,y)] \\ &\quad + [f(x,y) - f(x+1,y)] - [f(x-1,y) - f(x,y)] \\ &= f(x,y) * H(x,y) \end{aligned}$$

Laplace mask

$$H = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Laplace filter example

Original image I

|I*H|

Highboost filtering with Laplace

- Overall

$$I_{highboost} = c \cdot I_{orginal} + I_{highpass}$$
$$= \left(c \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + H \right) * I_{original}$$

- Using Laplace mask

$$I_{highboost} = \left(c \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \right) * I_{original} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & c+4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * I_{original}$$

Highboost filter example

Original Image

c=0.5

c=1

Gradient comparison

Pixel difference

Robert

Prewitt

Sobel

Laplace

More advanced edge detection

LoG edge detection

1. Applying LoG to the image

$$\nabla^2 G(x,y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

2. Detection of zero-crossings in the image
3. Threshold the zero-crossing to keep only the strong ones (large difference between the positive maximum and the negative minimum)

LoG edge detection example

Original image

LoG filter

LoG edge

Canny edge detection

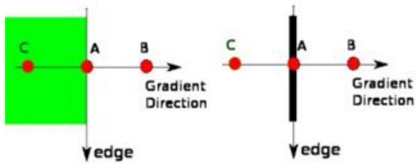
1. Smooth with 5x5 Gaussian kernel

2. Gradient with Sobel kernels

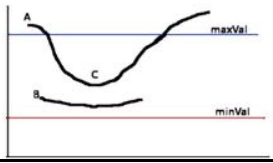
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3. Non-maximum suppression



4. Thresholding



Canny edge detection example

Original image

Canny

Line detection

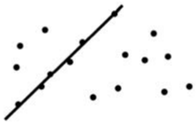
Hough transform

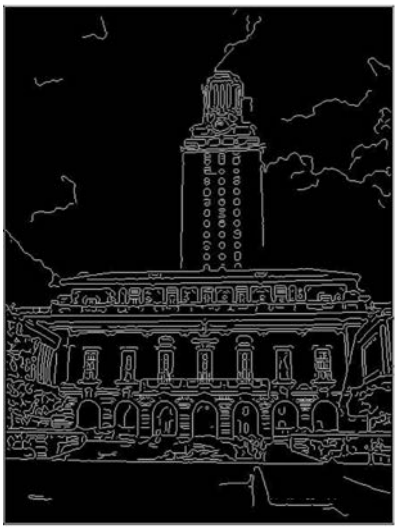
Example: Line fitting

- Many objects characterized by presence of straight lines



Difficulty of line fitting

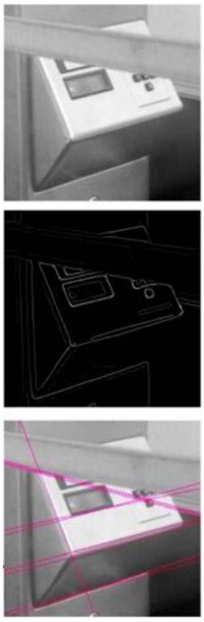




- Extra edge points (clutter), multiple models
  - Which points go with which line, if any?
- Only some parts of each line detected, and some parts are missing:
  - How to find a line that bridges missing evidence?
- Noise in measured edge points, orientations:
  - How to detect true underlying parameters

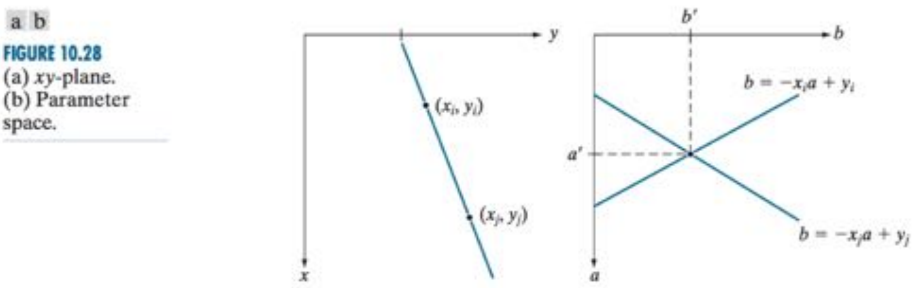
Fitting lines with Hough transform

- 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



Line planes

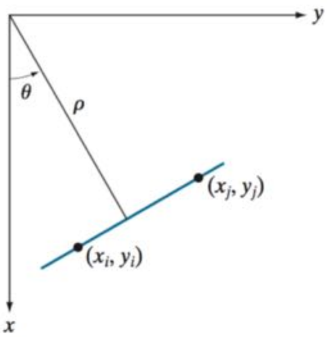
ab-plane or parameter space



y = a'x + b'

What if the line approaches the vertical or horizontal direction?
(i.e., infinity slope)

Polar representation for lines



rho: perpendicular distance from line to origin

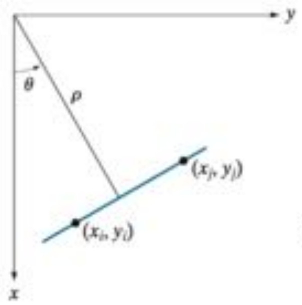
theta: angle the perpendicular makes with the x-axis

Normal presentation of a line:

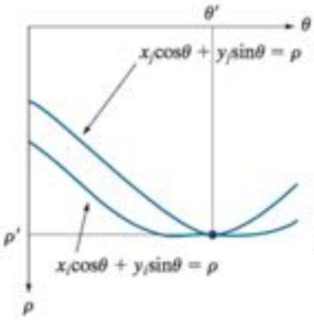
$$\boxed{x\cos\theta + y\sin\theta = \rho}$$

→ Point in image space is now sinusoid segment in Hough space

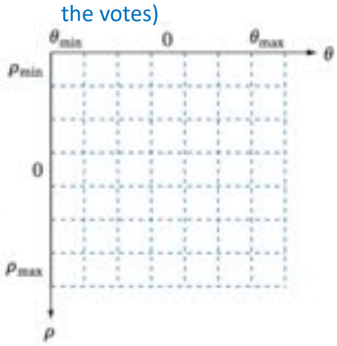
Finding lines in an image: Hough algorithm



1. Using the polar parameterization



2. Create a Hough Accumulator Array (keeps the votes)



- Domain of the parametric space:

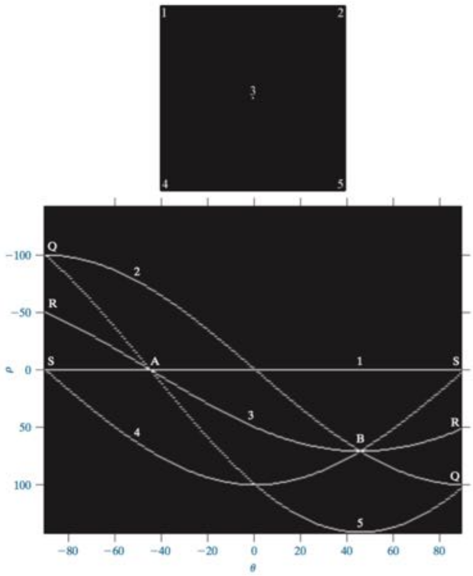
$$r\in\left[-\sqrt{M^2+N^2},\sqrt{M^2+N^2}\right],\theta\in\left[-\frac{\pi}{2},\frac{\pi}{2}\right]$$

M and N image resolution

Example of Hough transform

a
b

FIGURE 10.30 (a) Image of size 101×101 pixels, containing five white points (four in the corners and one in the center). (b) Corresponding parameter space.



Basic Hough transform algorithm

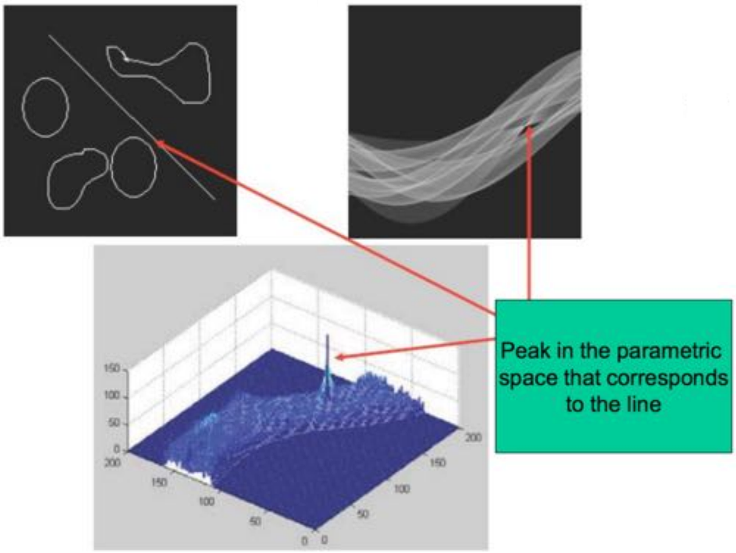
1. Initialize H[d, θ]=0
2. For each ***edge*** point in $E(x, y)$ in the image

    for θ = 0 to 180 // 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 $d = x\cos\theta + y\sin\theta$

Line detection example

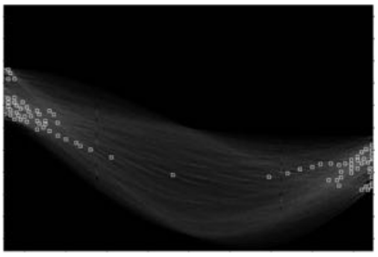


Line detection example

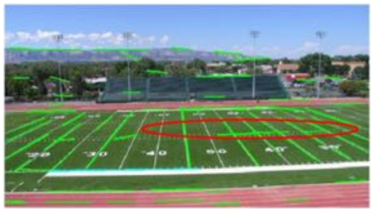
original

Canny edges

Vote space and
top peaks



Longest segments
found



Impact of noise on Hough

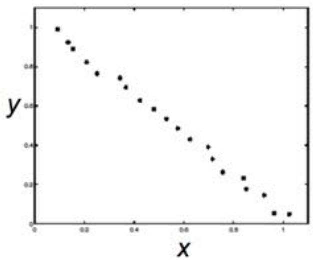
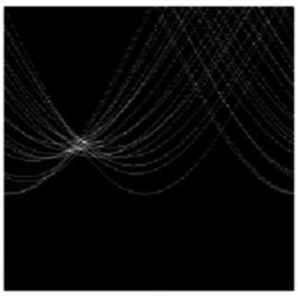


Image space
edge coordinates



Votes

What difficulty does this present for an implementation?

Impact of noise on Hough

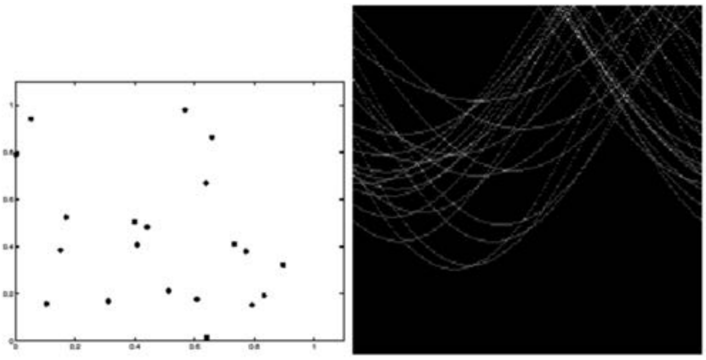


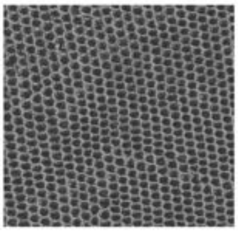
Image space edge coordinates

Votes

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

Texture analysis

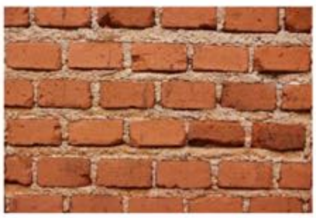
What is texture?


- An image obeying some statistical properties
- Similar structures repeated over and over again
- Often has some degree of randomness

Aspects of texture


- Size/granularity (sand versus pebbles versus boulders)
- Directionality/Orientation
- Random or regular (stucco versus bricks)


Statistical approach to texture

- Characterize texture using statistical measures computed from grayscale intensities (or colors) alone
- Less intuitive, but applicable to all images and computationally efficient
- Can be used for both classification of a given input texture and segmentation of an image into different regions

Some (simple) statistical texture measures

- Edge density and direction

- Use an edge detector as the first step in texture analysis
- The number of edge pixels in a fixed-size region tells us how busy that region is
- The directions of the edges also help characterize the texture

Two edge-based texture measures

1. edgeness per unit area

$\boxed{F_{edgeness} = |\{ p \mid \text{gradient\_magnitude}(p) \geq \text{threshold}\}| / N}$

where N is the size of the unit area

2. edge magnitude and direction histograms

$\boxed{F_{magdir} = (H_{magnitude}, H_{direction})}$

where these are the normalized histograms of gradient magnitudes and gradient directions, respectively.

Example

Original Image

Frei-Chen Edge Image

Thresholded Edge Image

Different F_{edgeness} for different regions

Local binary pattern measure

Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns

Timo Ojala, Matti Pietikäinen and Topi Mäenpää

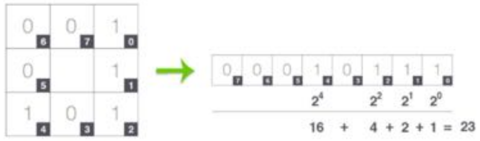


Figure 2: Taking the 8-bit binary neighborhood of the center pixel and converting it into a decimal representation. (Thanks to Bikramjot of Hanzra Tech for the inspiration on this visualization!)

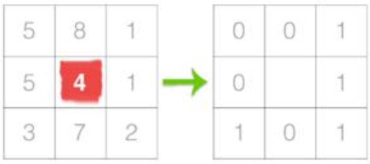


Figure 1: The first step in constructing a LBP is to take the 8 pixel neighborhood surrounding a center pixel and threshold it to construct a set of 8 binary digits.

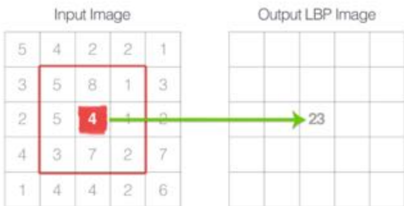
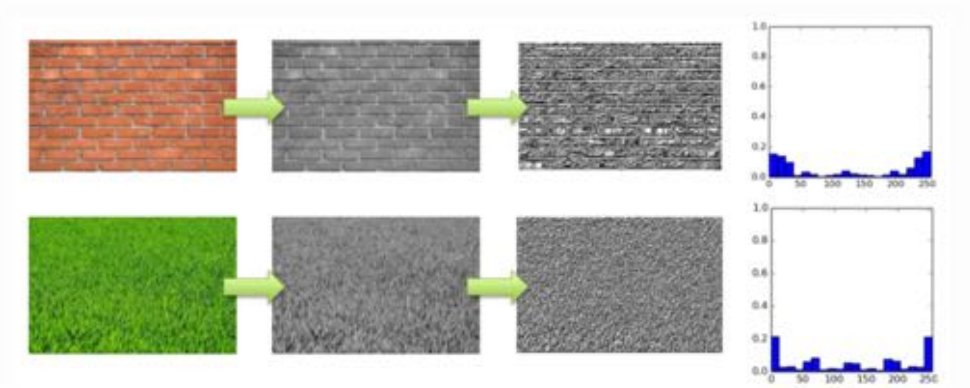


Figure 3: The calculated LBP value is then stored in an output array with the same width and height as the original image.

Image credit: https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

LBP example



Color Image -> Grayscale Image -> LBP Mask -> Normalized LBP Histogram

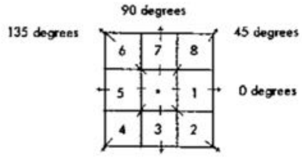
Image credit: http://hanzratech.in/2015/05/30/local-binary-patterns.html

Gray Level Co-occurence Matrix (GLCM)

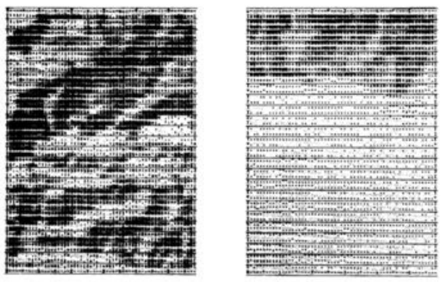
Textural Features for Image Classification

ROBERT M. HARALICK, K. SHANMUGAM, AND ITS'HAK DINSTEIN

- Distribution of co-occurring pixel values (grayscale values, or colors) at a given offset
  - A distance *d*, and an angle θ



GLCM example



	Grassland			Water Body		
Angle	ASM	Contrast	Correlation	ASM	Contrast	Correlation
0°	.0128	3.048	.8075	.1016	2.153	.7254
45°	.0080	4.011	.6366	.0771	3.057	.4768
90°	.0077	4.014	.5987	.0762	3.113	.4646
135°	.0064	4.709	.4610	.0741	3.129	.4650
Avg.	.0087	3.945	.6259	.0822	2.863	.5327

(a) (b)

Fig. 4. Textural features for two different land-use category images.