Sensing, Processing and Automatic Decision Making for Security and Surveillance Applications

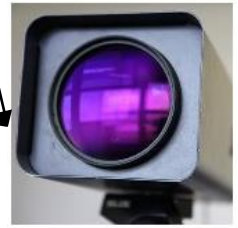
Vijayan K. Asari
Professor and Endowed Chair
Electrical and Computer Engineering
Director, UD Vision Lab
University of Dayton
Dayton, Ohio, USA

Research Theme

Sensor Data Acquisition → Sensor Data Exploitation → Decision and Feedback

Canon Cameras

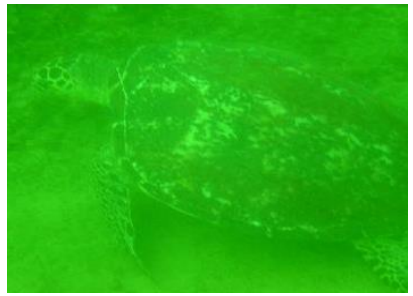
Hyperspectral Camera



Long-range Cameras

Scene Visibility Improvement

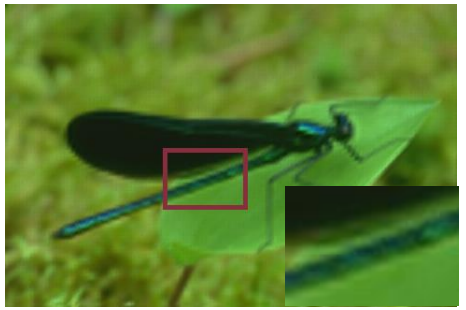
Image Enhancement (*low lighting , underwater, hazy images*)

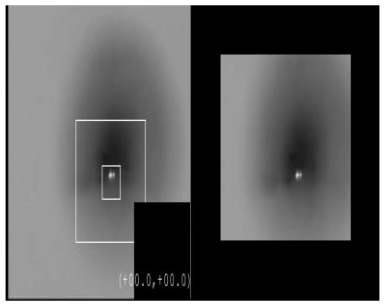

Rain Removal


Wide Area Moving Imagery (CLIF Data)


Image Super-resolution
(*long distance video*)

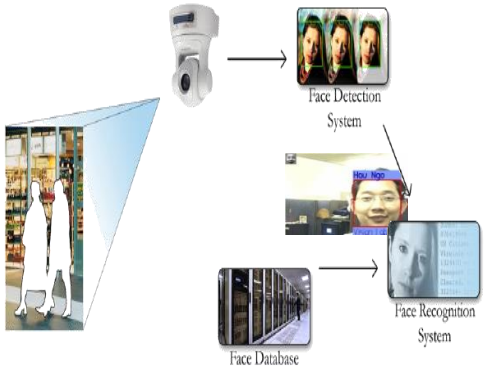
Video Stabilization
(*mobile platform*)



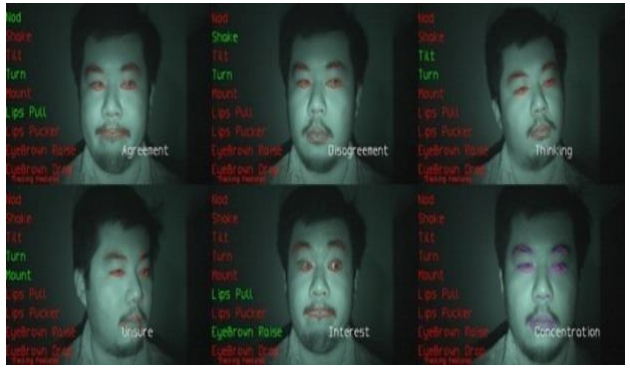
Scene Analysis and Understanding

Face Detection

Human Identification

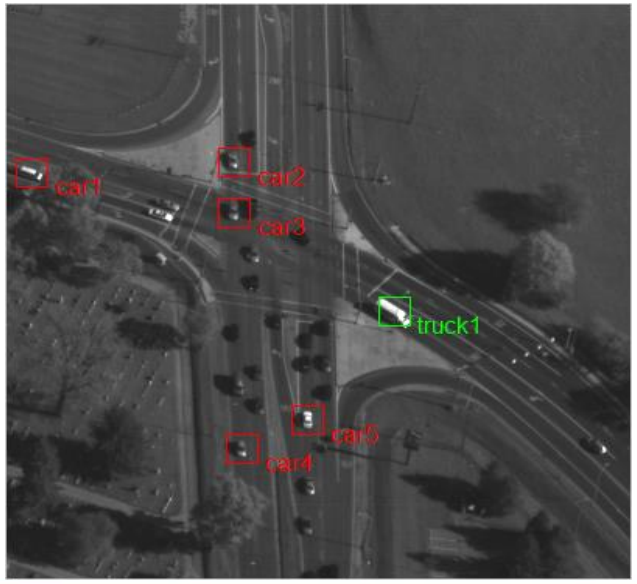


Human Action and Expression Recognition



3D Scene Reconstruction

Long Range Object Tracking



Whale Blow Detection in IR Video

Processing Pipeline

Distortion correction, enhancement, haze/fog removal, rain removal, stabilization.

Pre-processing

Spatial domain features
Spectral domain features
Statistical features
Phase features

Feature Extraction

Classification

Classical and neural network based approaches.

Scene Analysis and Understanding

Wide Area Surveillance

Biometric Identification

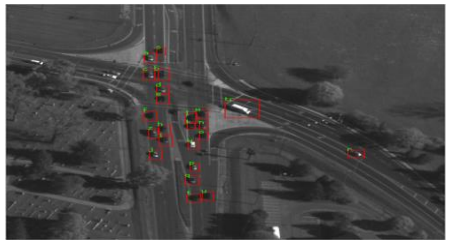
Human Activity Recognition

Brain Activity Analysis

Hardware Acceleration

Decision

Face recognition.
Human action and activity recognition.
Object detection/tracking on WAMI data.
Emotion recognition by EEG analysis.
Brain machine interface.



Data Acquisition

Outline

- **Introduction**
- **Image preprocessing**: noise removal, enhancement, distortion correction
- **Feature extraction**: textural features, shape features, region description
- **Classification**: classical methods, neural networks, deep learning
- **Applications**: human detection, feature tracking, face recognition

Pre-processing: Image Enhancement

Image Processing

$f(x, y)$

$$g(x, y) = T[f(x, y)])$$

$f(x, y)$: input image

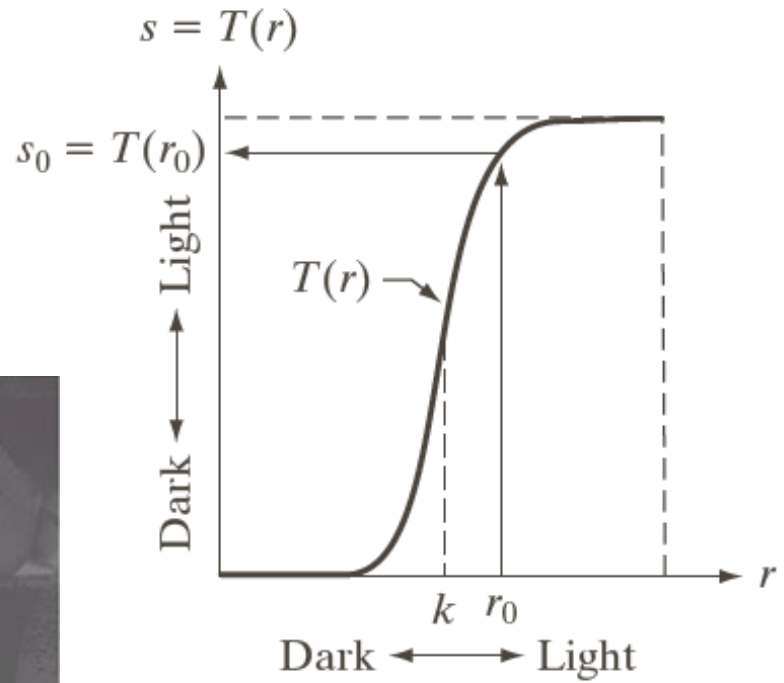
$g(x, y)$: output image

T : an operator on f defined over a neighborhood of point (x, y)

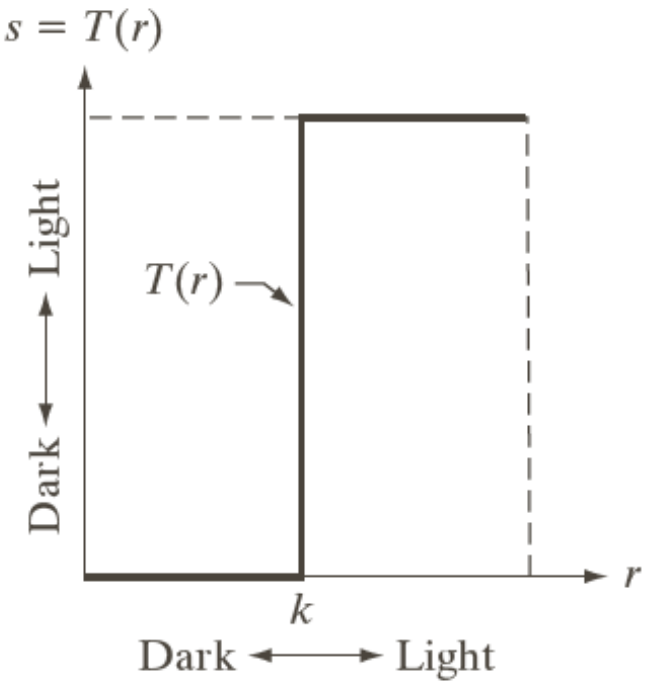
$g(x, y)$

Intensity Transformation Functions

f (x, y)



Contrast Stretching

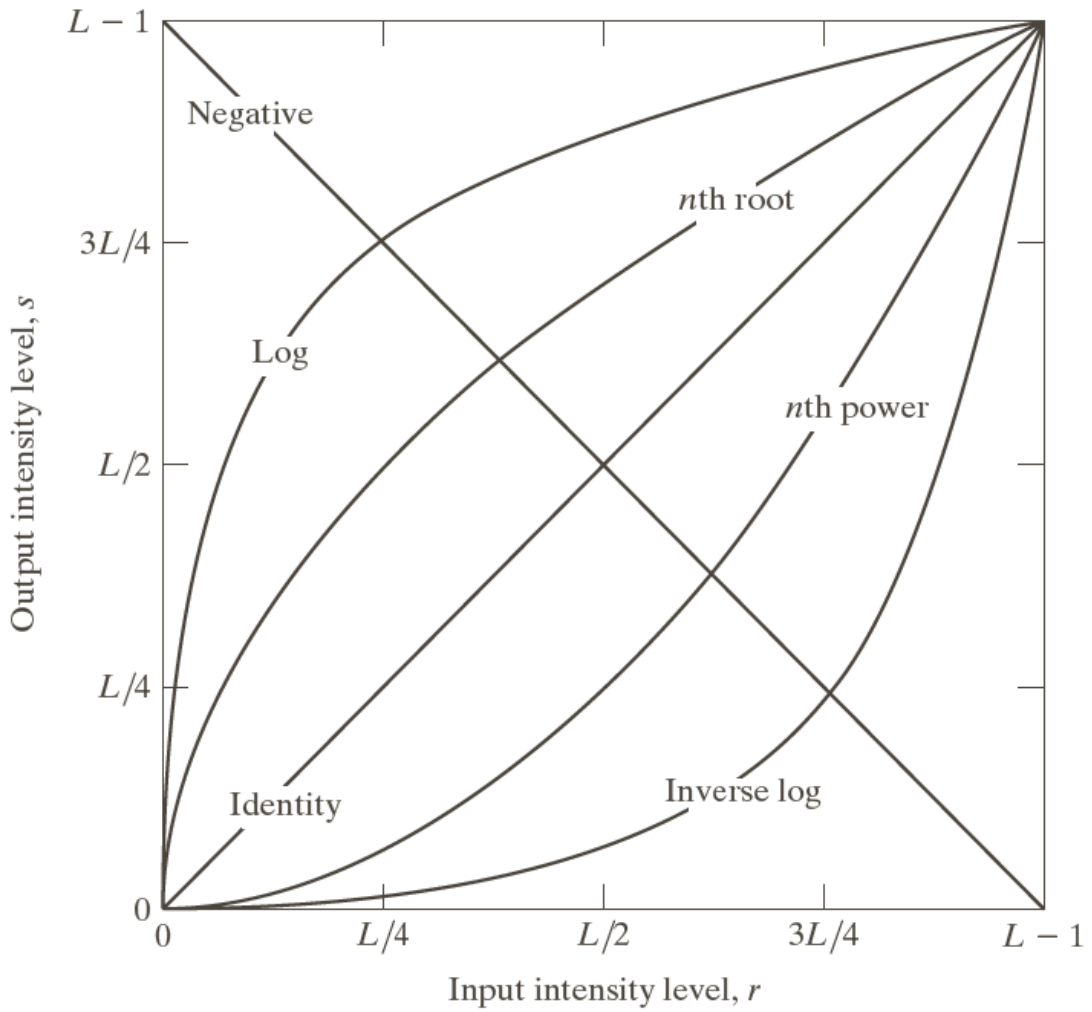


Thresholding

r = f (x, y)
s = g (x, y)

g (x, y)

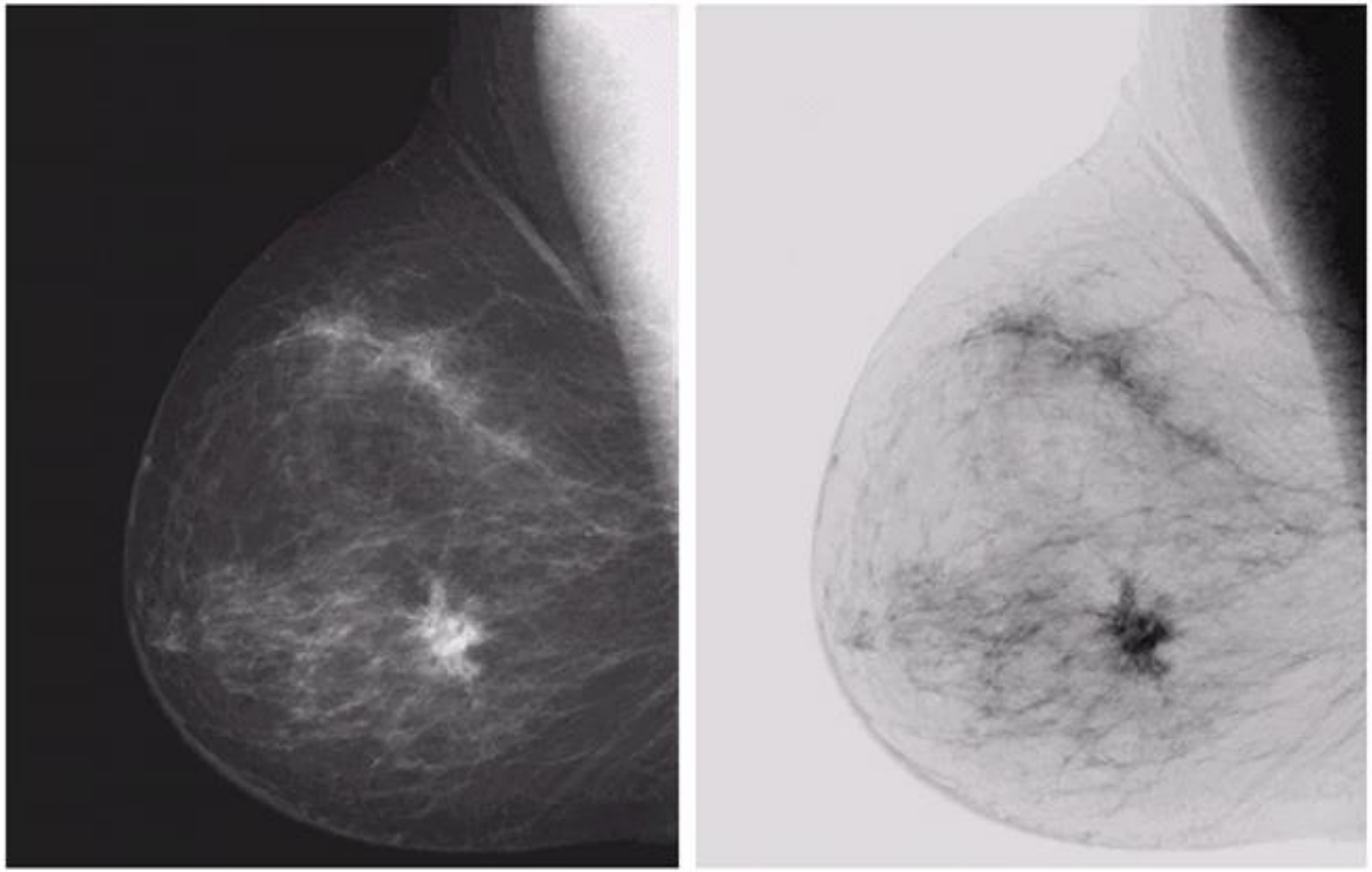
Intensity Transformation Functions

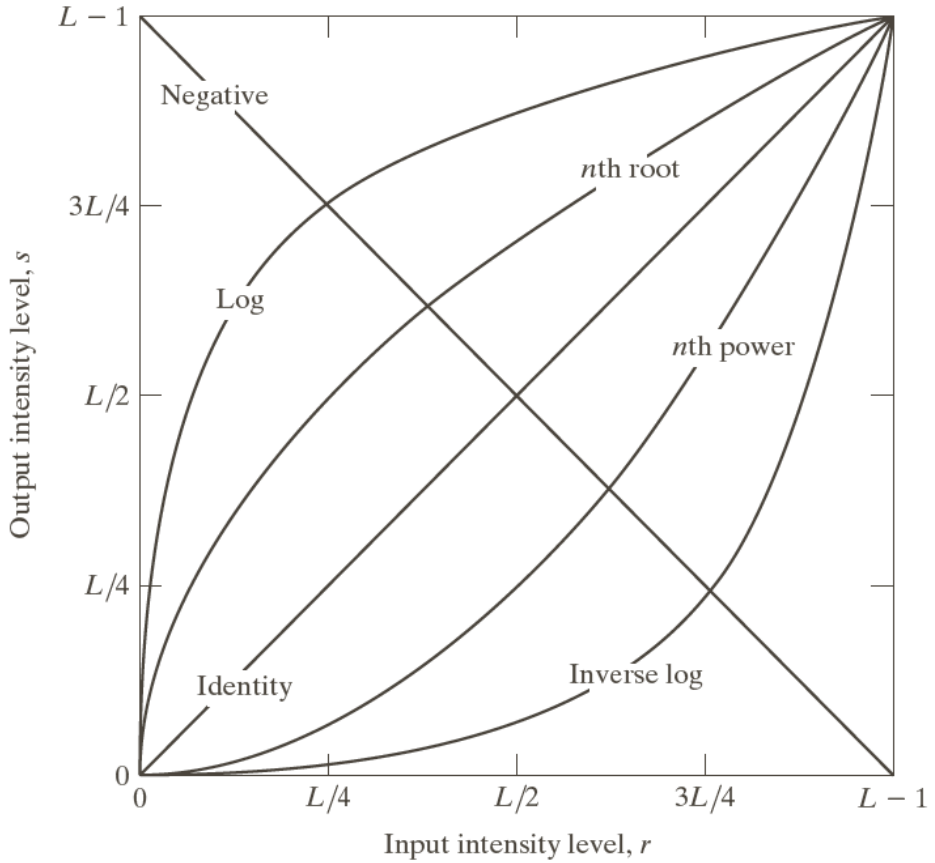


- Linear: Negative, Identity
- Logarithmic: Log, Inverse Log
- Power-Law: *n*th power, *n*th root

Intensity Transformation Functions

Image Negative: $s = (L - 1) - r$

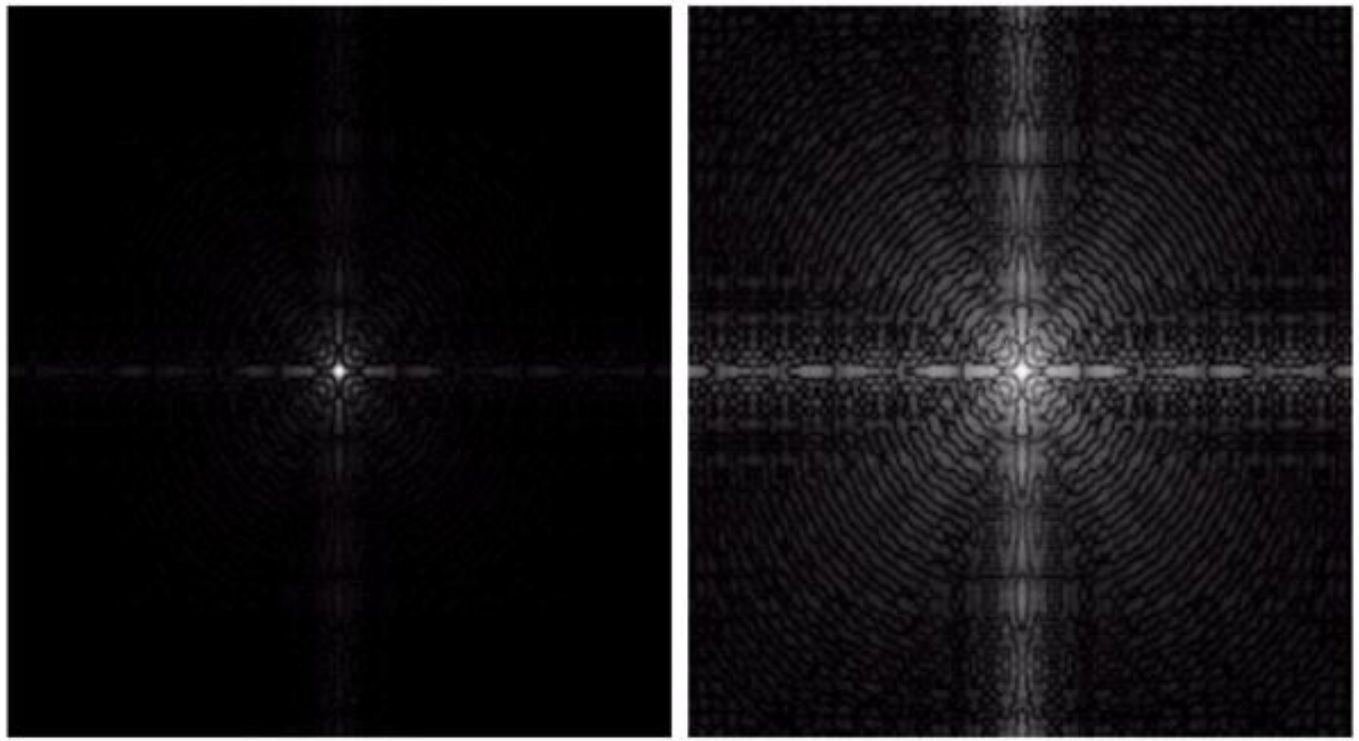


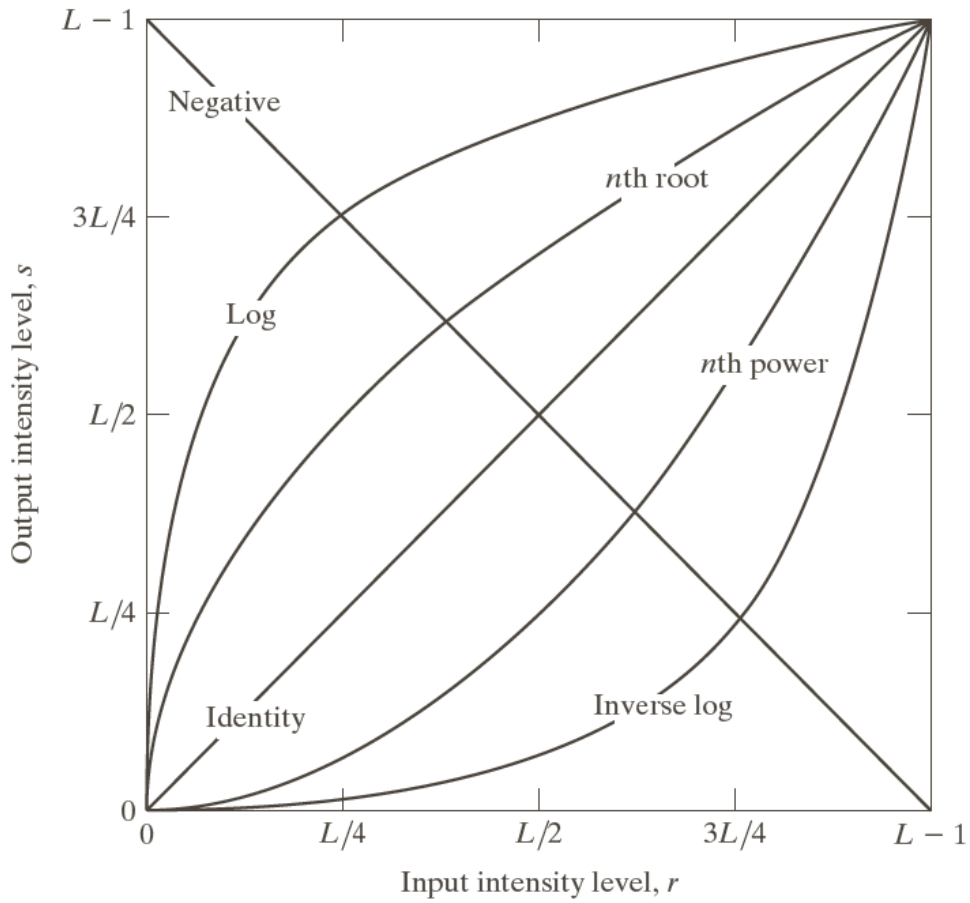


Log Transformation

$s = c \log(1 + r)$, c is a constant

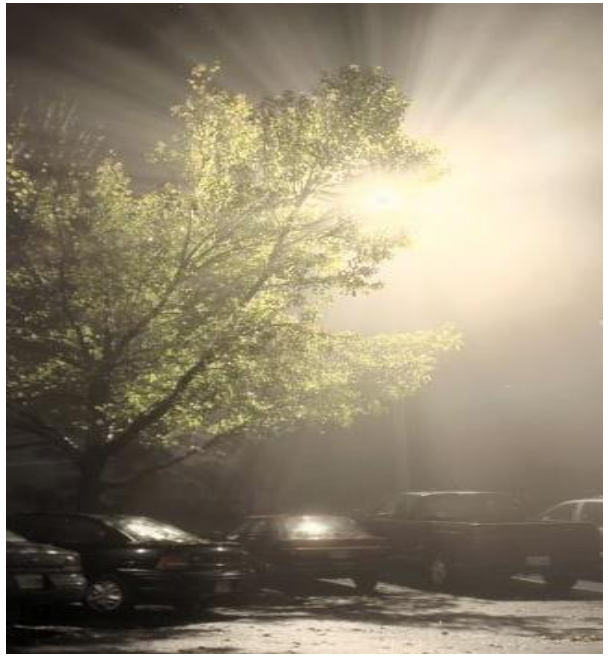
Compresses the dynamic range of images with large variations in pixel values

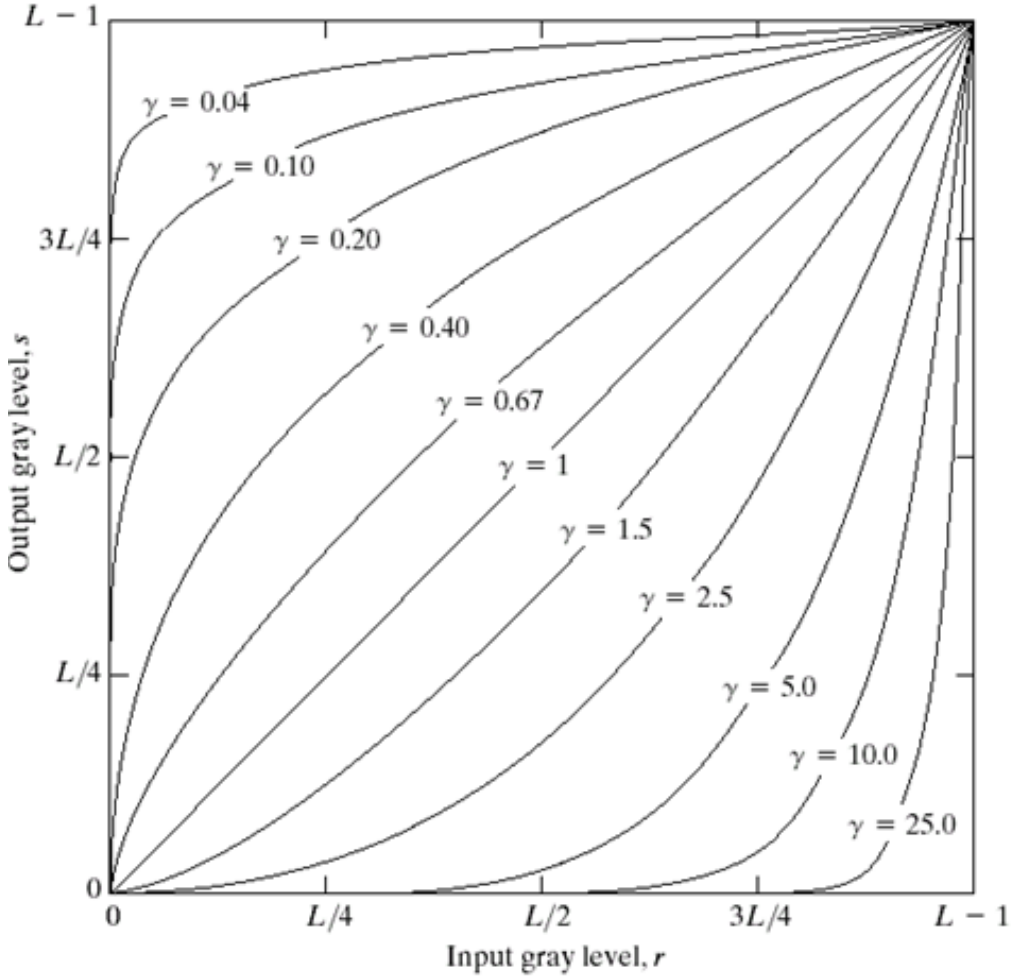




Power Law Transformation: Gamma Correction

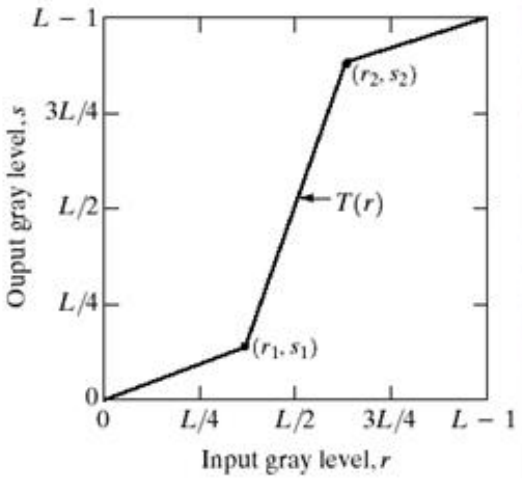
$s = c\, r^{\gamma}$ where c, γ : positive constants

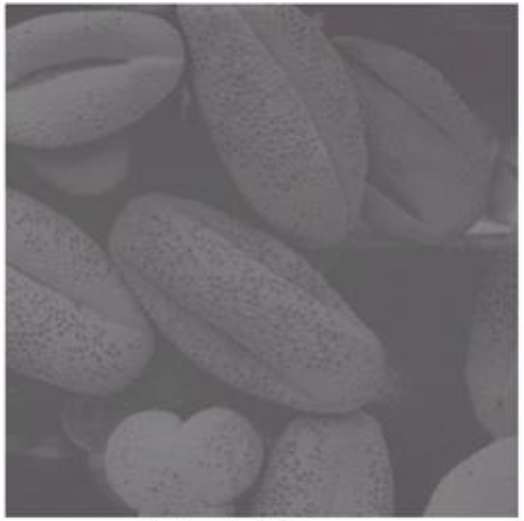




Contrast Stretching

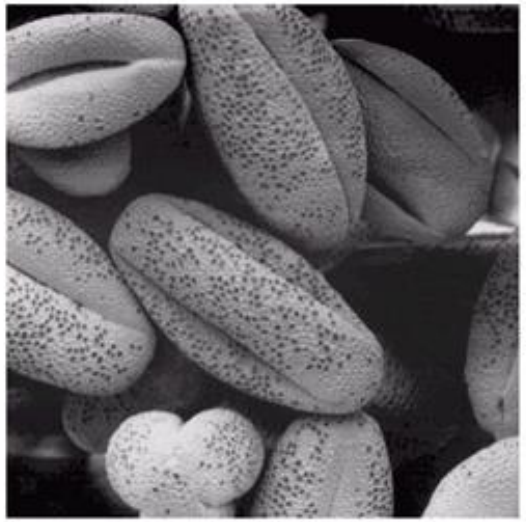
Transformation Function





Low Contrast Image

Contrast Stretching Result



Thresholding Result

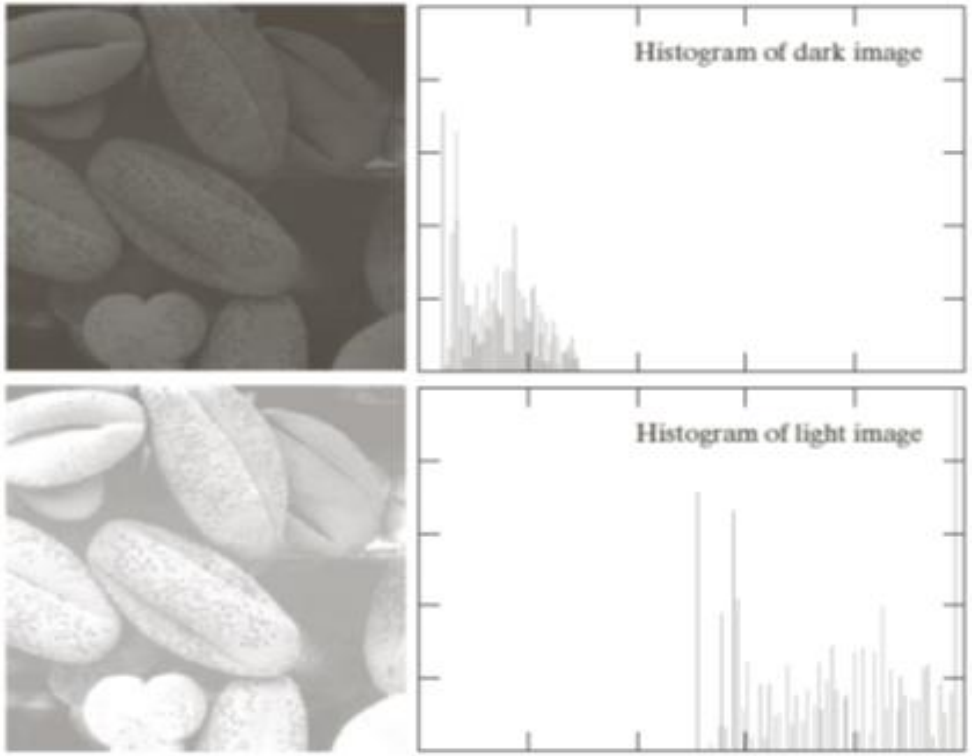
Image Histogram

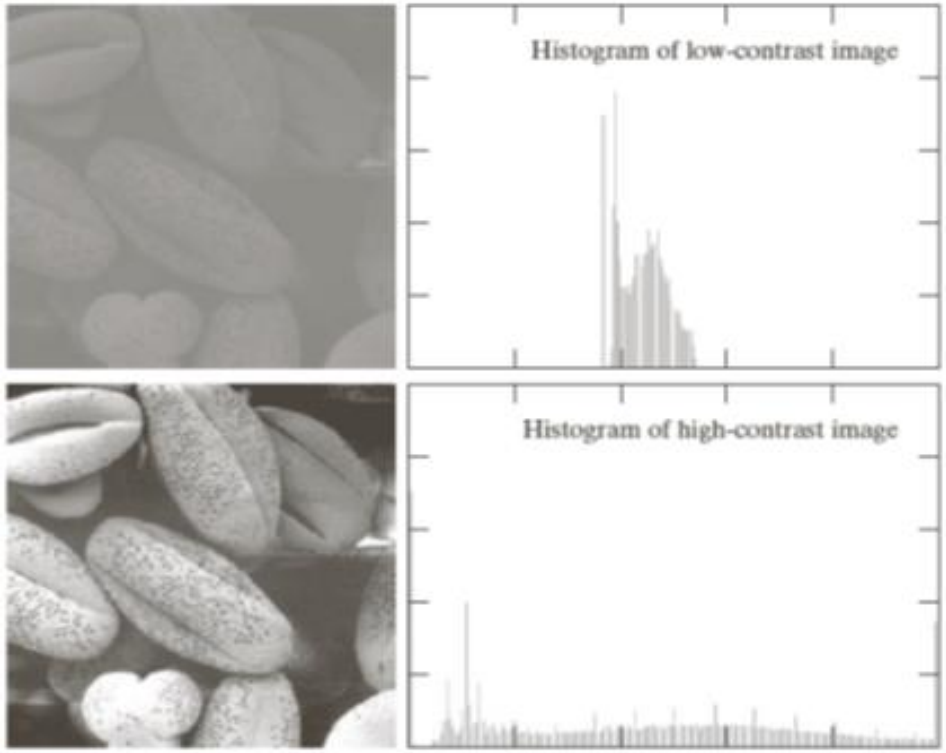
Histogram $h(r_k) = n_k$

Normalized histogram $p(r_k) = \frac{n_k}{MN}$

r_k is the k^{th} intensity value

n_k is the number of pixels in the image with intensity r_k

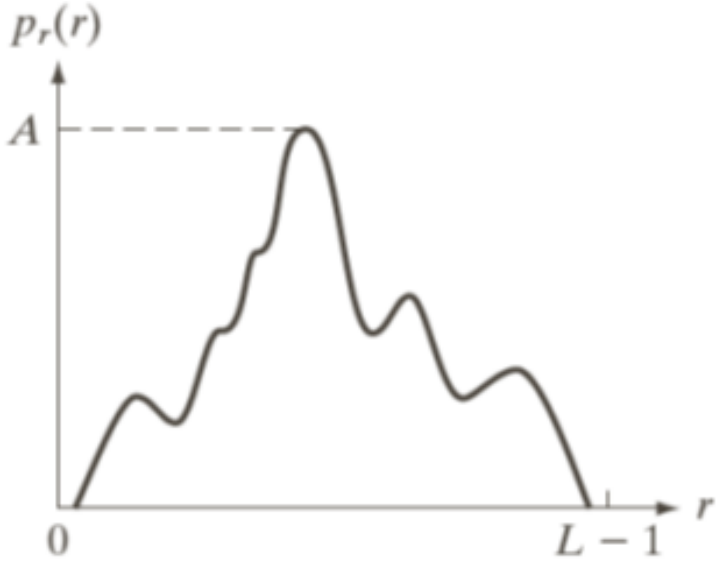


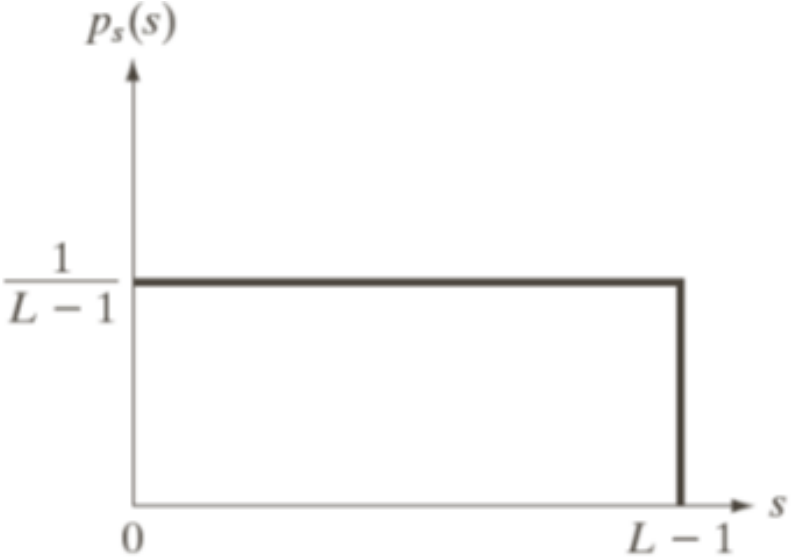


Histogram Equalization

The intensity levels in an image may be viewed as random variables in the interval [0, L-1].

Let $p_r(r)$ and $p_s(s)$ denote the probability density function (PDF) of random variables r and s.





Histogram Equalization

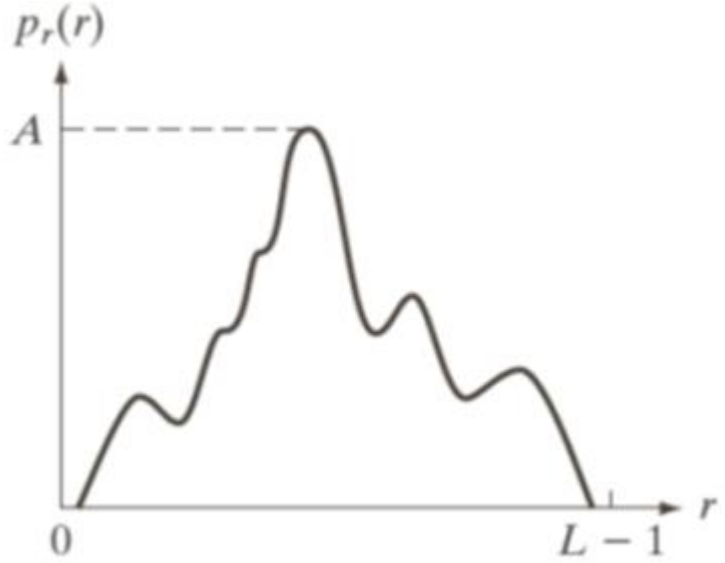
$$s = T(r) \qquad 0 \le r \le L-1$$

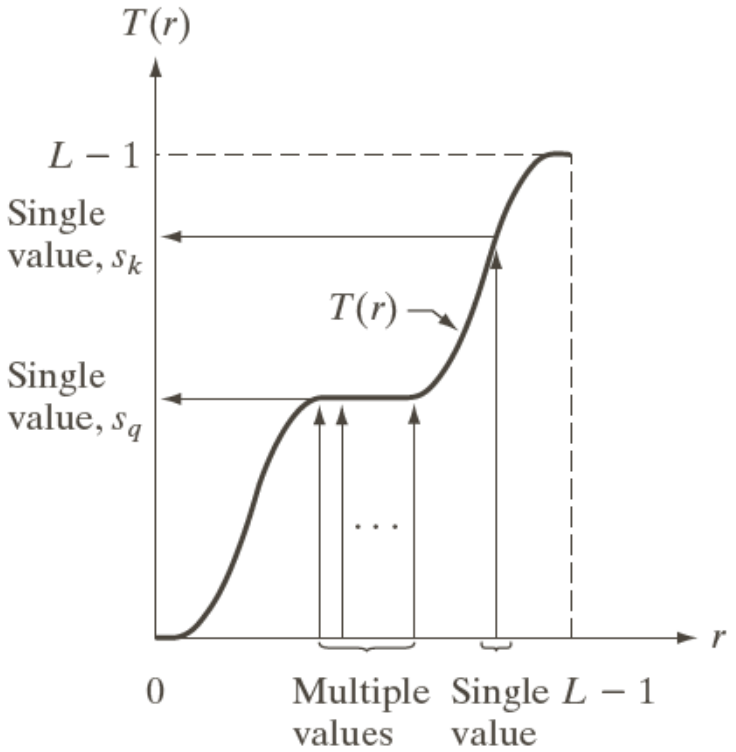
$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

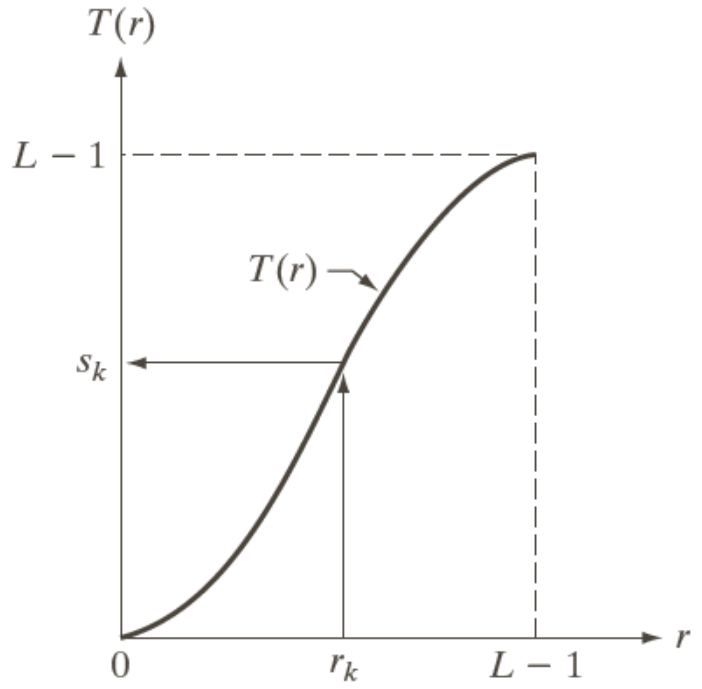
T(r) is a strictly monotonically increasing function in the interval $0 \le r \le L\text{-}1$;

$0 \le T(r) \le L\text{-}1$ for $0 \le r \le L\text{-}1$.

$T(r)$ is continuous and differentiable.







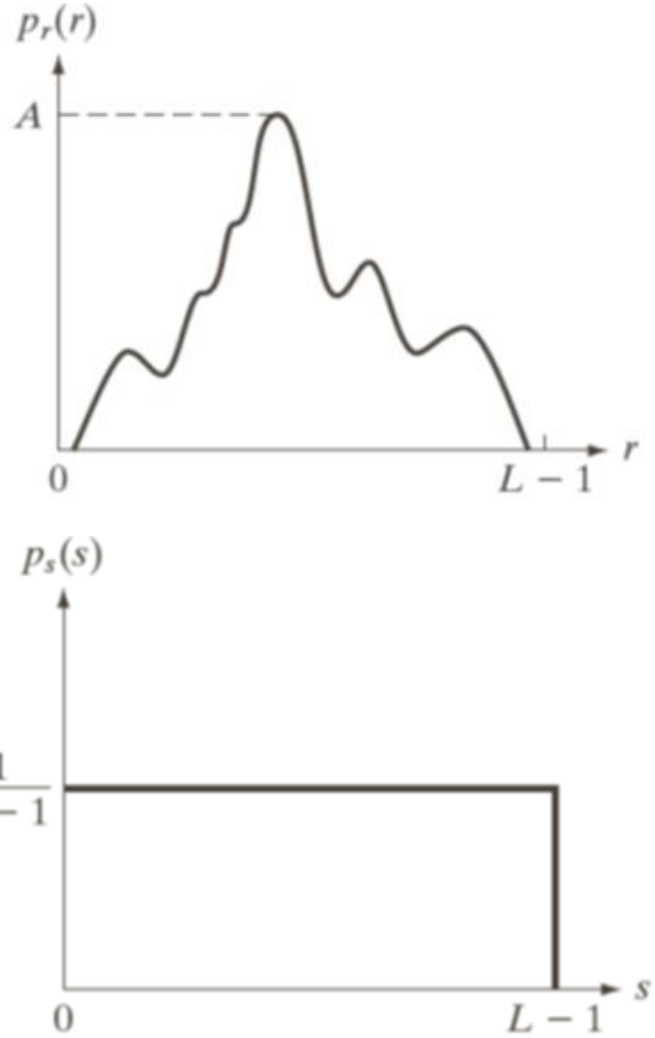
Histogram Equalization

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] = (L-1)p_r(r)$$

$$p_s(s)ds = p_r(r)dr$$

$$p_s(s) = \frac{p_r(r)dr}{ds} = p_r(r)\Big/\left(\frac{ds}{dr}\right) = p_r(r)\Big/\left((L-1)p_r(r)\right) = \frac{1}{L-1}$$



Histogram Equalization

Suppose that a 3-bit image (L=8) of size 64 × 64 pixels (MN = 4096) has the intensity distribution.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

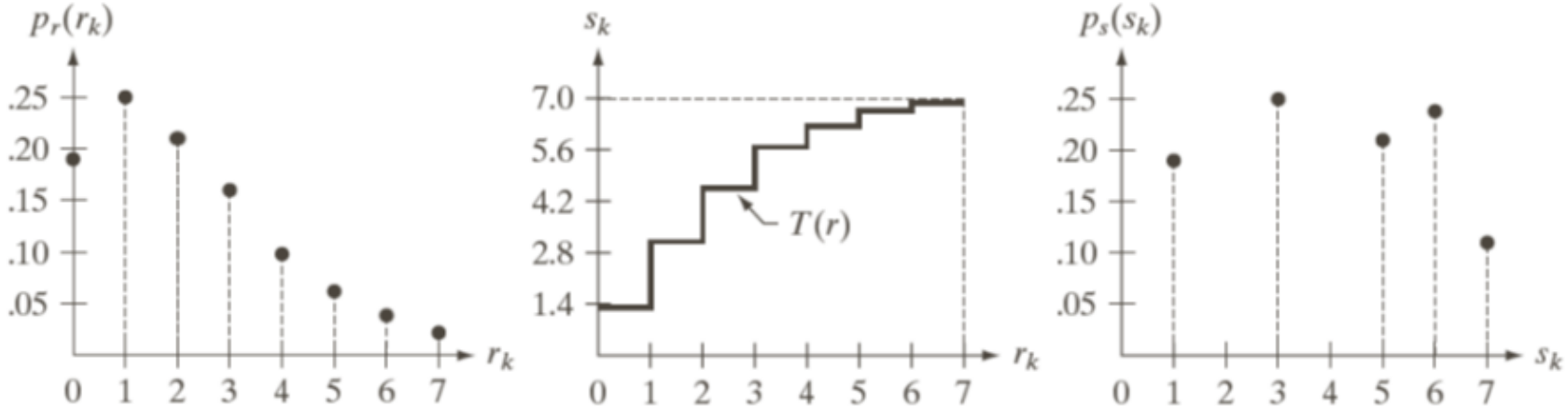
$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7 \times 0.19 = 1.33 \rightarrow 1$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7 \times (0.19 + 0.25) = 3.08 \rightarrow 3$$

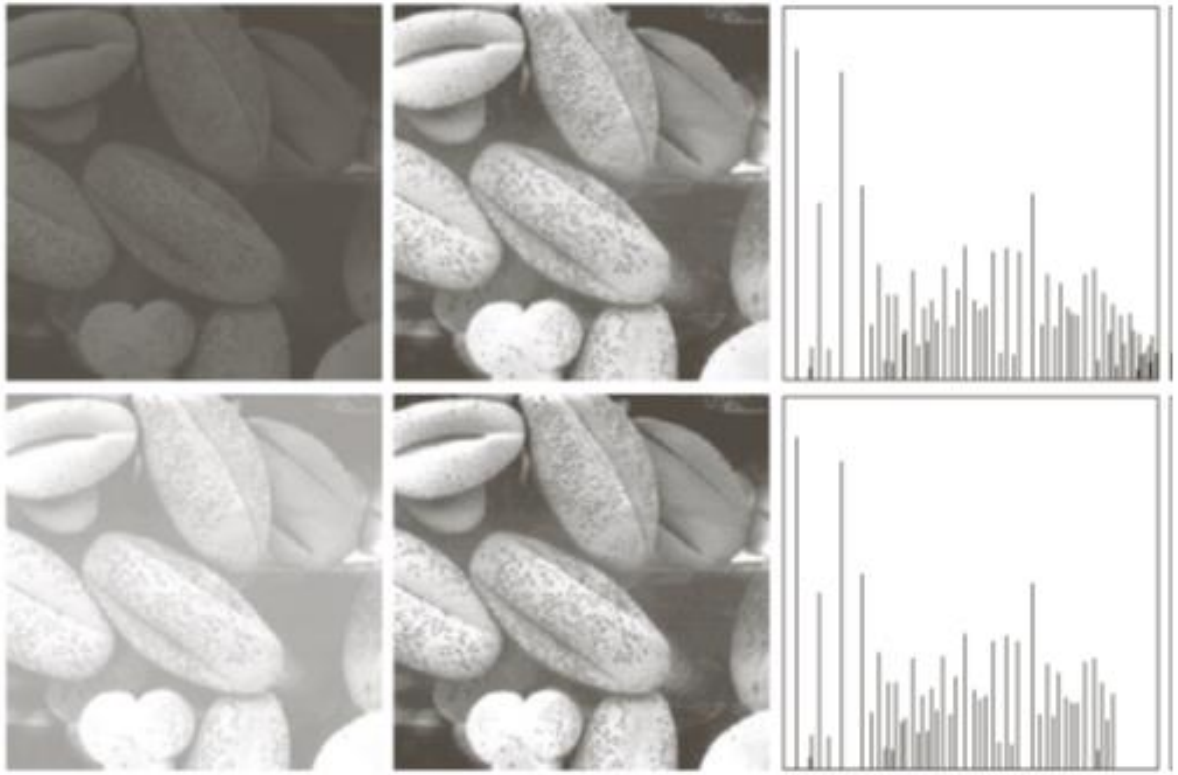
$$s_2 = 4.55 \rightarrow 5 \qquad s_3 = 5.67 \rightarrow 6$$

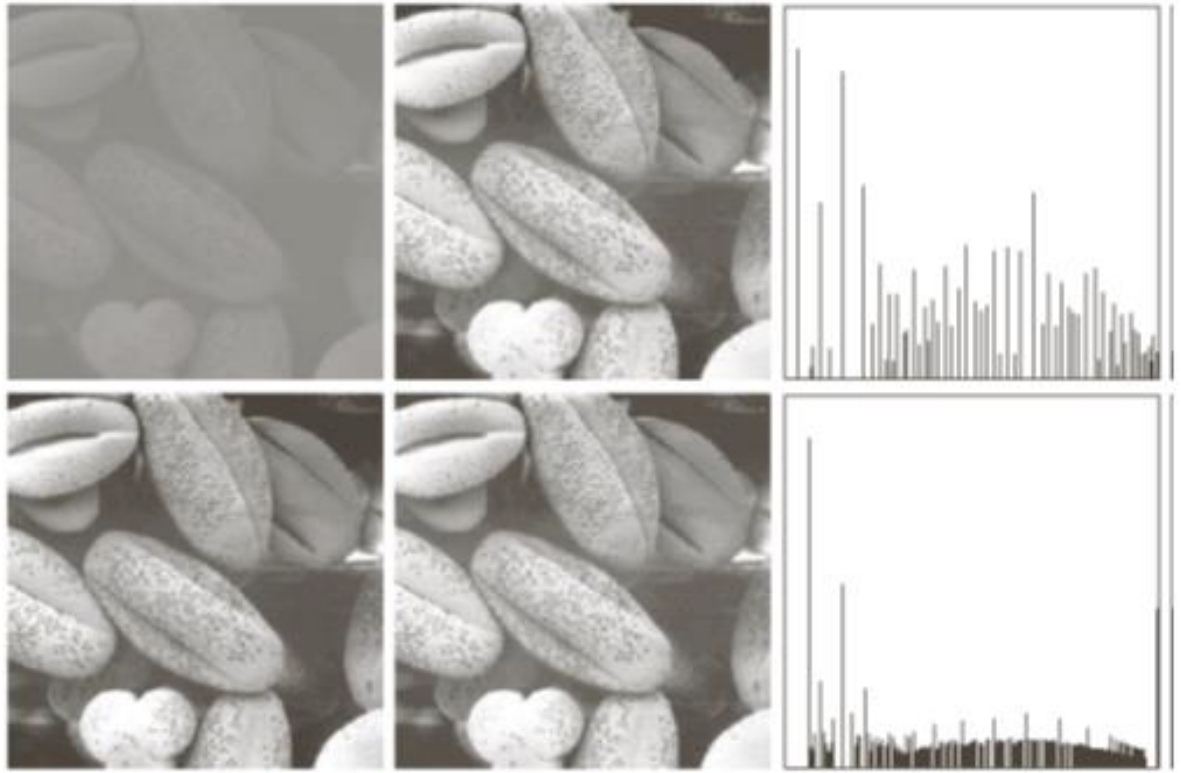
$$s_4 = 6.23 \rightarrow 6 \qquad s_5 = 6.65 \rightarrow 7$$

$$s_6 = 6.86 \rightarrow 7 \qquad s_7 = 7.00 \rightarrow 7$$

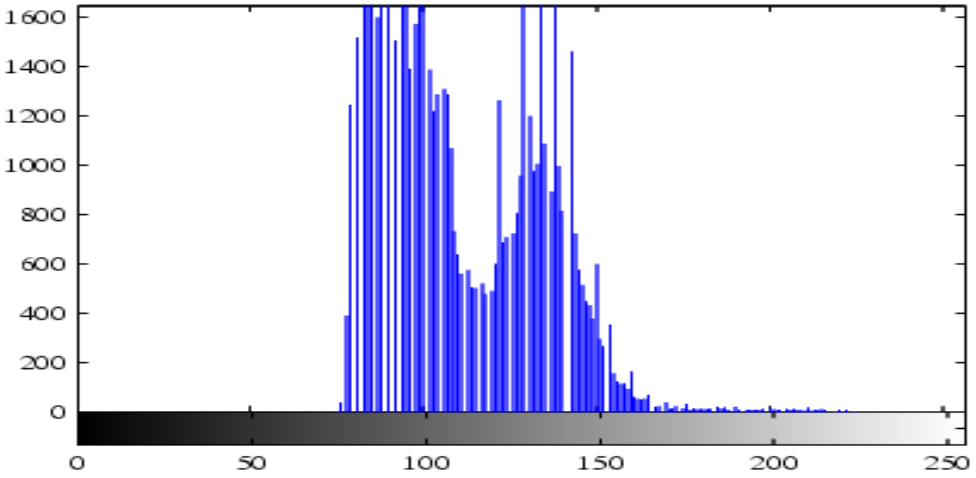


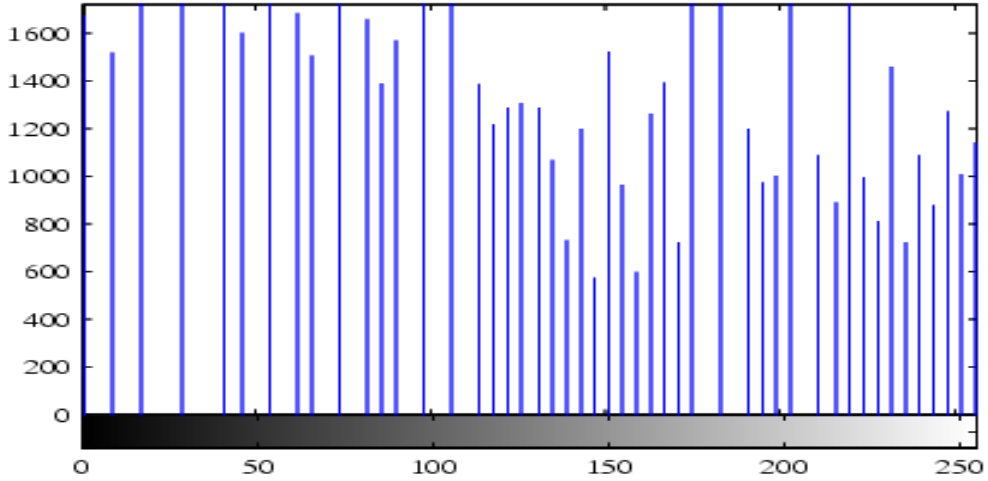
Histogram Equalization





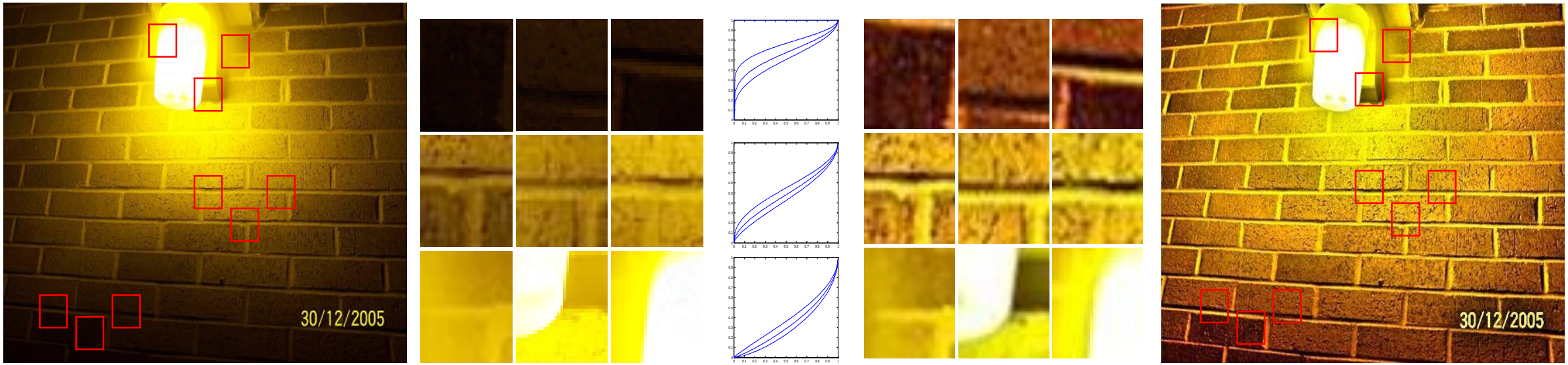
Histogram Equalization

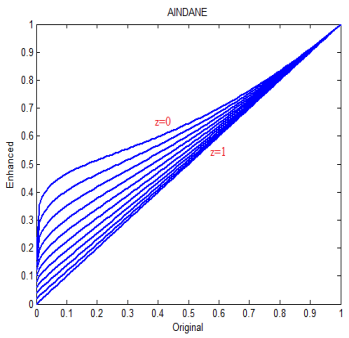




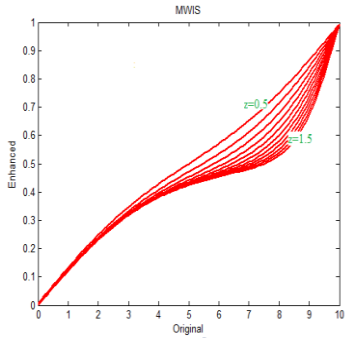
Enhancement of Low Lighting and Over Exposed Images

Underexposed, dark, dark and bright (shadows), bright, overexposed regions

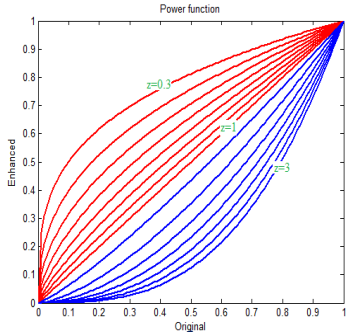




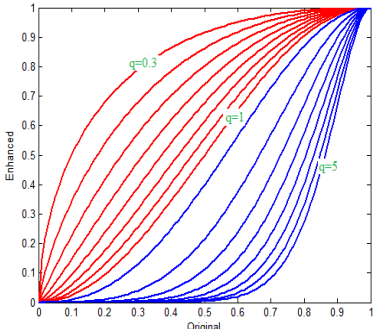
AINDANE



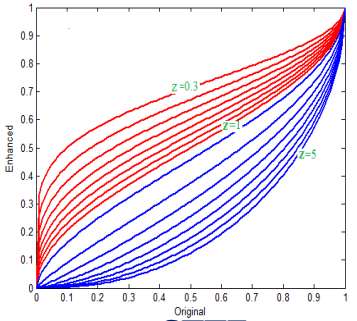
MWIS



Power function



LTSN



STTF

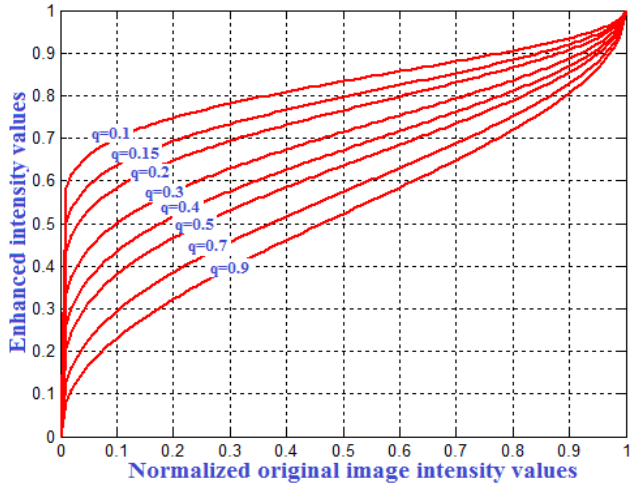
Dynamic Range Compression

Intensity computation (NTSC)

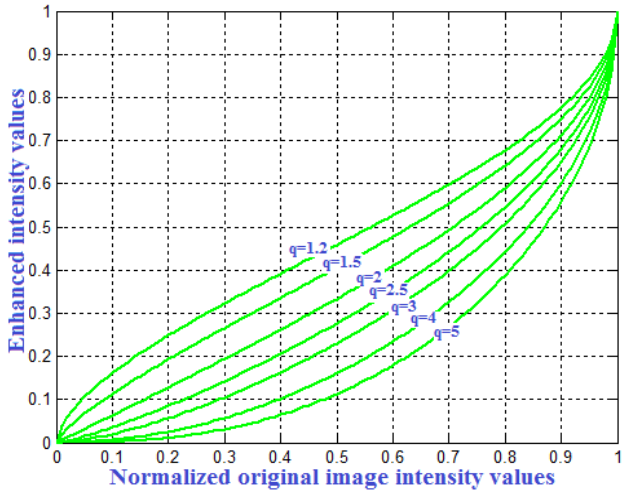
$$I(x,y) = 0.2989 \times I_{Rh}(x,y) + 0.5867 \times I_{Gh}(x,y) + 0.114 \times I_{Bh}(x,y)$$

Nonlinear function

$$I_{enh}(x,y) = (2/\pi) ArcSin(I_n(x,y)^{q/2})$$



Dark pixels



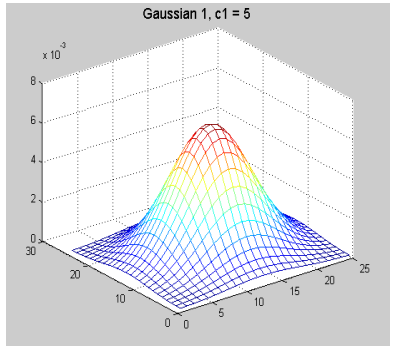
Bright pixels

Adaptive Estimation of Control Parameter

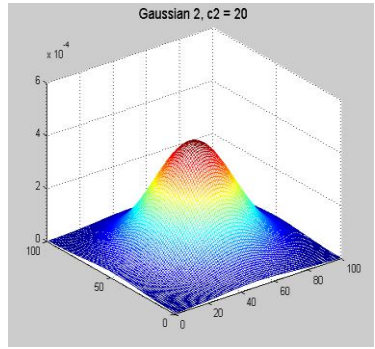
$q < 1$ provides various nonlinear curves if the pixels are dark.

$q = 1$ provides a curve if the pixel has sufficient intensity.

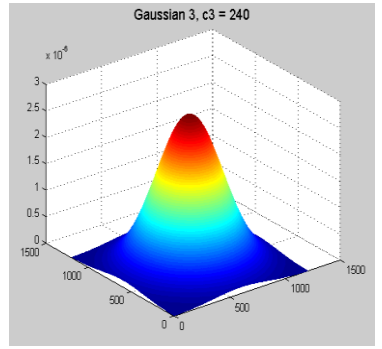
$q > 1$ provides various nonlinear curves if the pixels are bright.



$w_1 = 5$



$w_2 = 20$



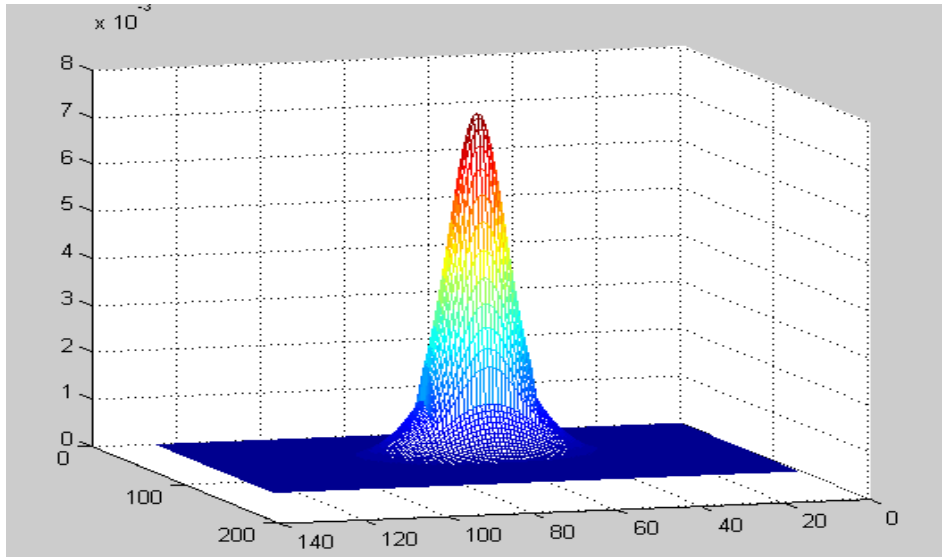
$w_3 = 240$

Depending on the mean value of its neighborhood

$$I_{M_i}(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(m,n)G_i(m+x,n+y)$$

$$G_i(x,y) = K.e^{\left(\frac{-(x^2+y^2)}{w_i^2}\right)}$$

Window size depends on the resolution and object size in an image.



Multi-level Gaussian function

Adaptive Estimation of Control Parameter

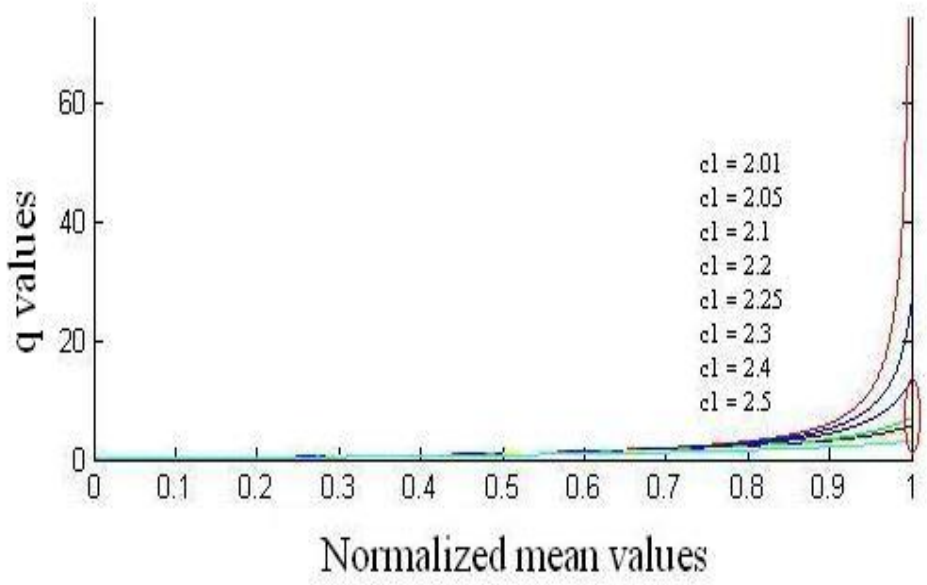
Criteria for estimation of *q*

$$q = \begin{cases} <1, & if \quad I_{M_n} < 0.5 \\ =1, & if \quad I_{M_n} = 0.5 \\ >1, & if \quad I_{M_n} > 0.5 \end{cases}$$

The function for the q value can be designed as

$$q = Tan\left(I_{M_n}(x,y) * (\pi / c_1)\right) + c_2$$

c_1 and c_2 are empirically determined. $c_1 = 2.25$ $c_2 = 0.0085$



For q values which are closer to 0 the noise in the extreme dark regions will also be enhanced.

Hence, the q values corresponding to the mean value below 0.2 is considered as extreme dark regions and q for those pixels can be calculated as

$$q = \log\left(\sqrt{2I_{M_n}(x,y) + 2}\right)$$

Nonlinear Enhancement Module

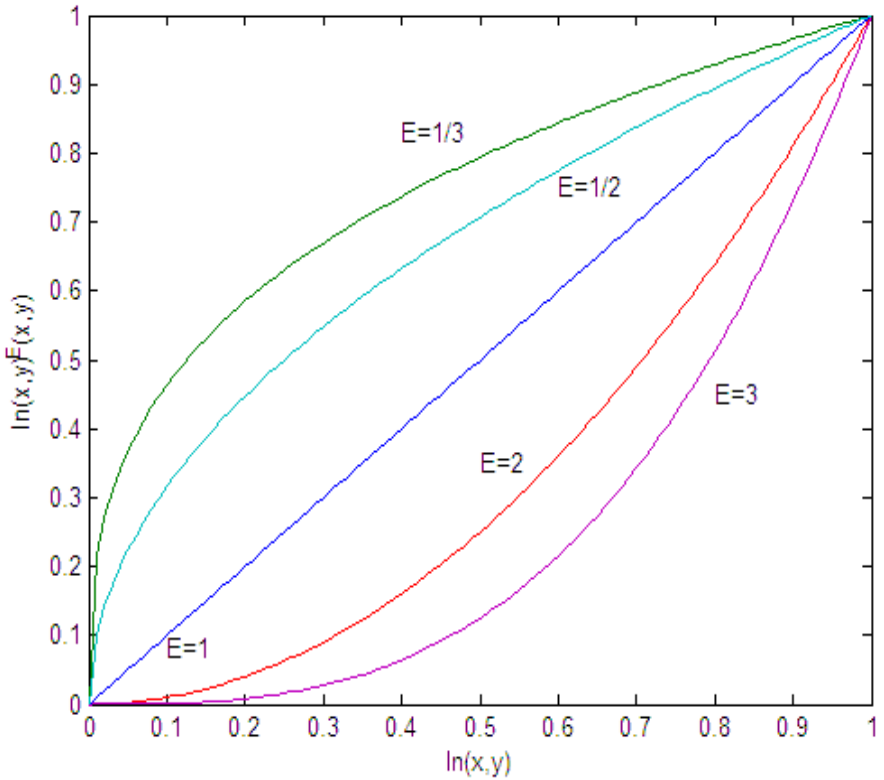
Contrast Enhancement

$$S(x,y) = 255 x I_{enh}(x,y)^{E(x,y)} \qquad E(x,y) = \left[\frac{I_{conv}(x,y)}{I(x,y)} \right]$$

Color restoration

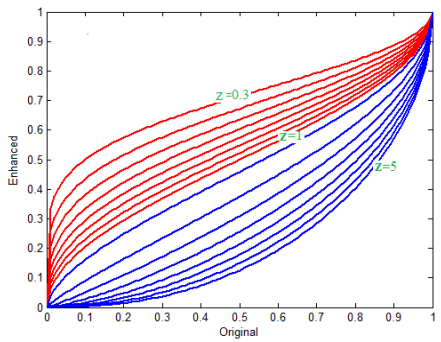
$$I_{enh,i} = I_i(x,y) \left(I_{enh}(x,y) \Big/ (I_n(x,y) \right)$$

where *i* represents red, green, blue spectral band



Enhancement of Low Lighting and Over Exposed Images

Input image



Enhanced image

Video Enhancement

Image Filtering

Spatial Filtering

A spatial filter consists of (a) **a neighborhood**, and (b) **a predefined operation.**

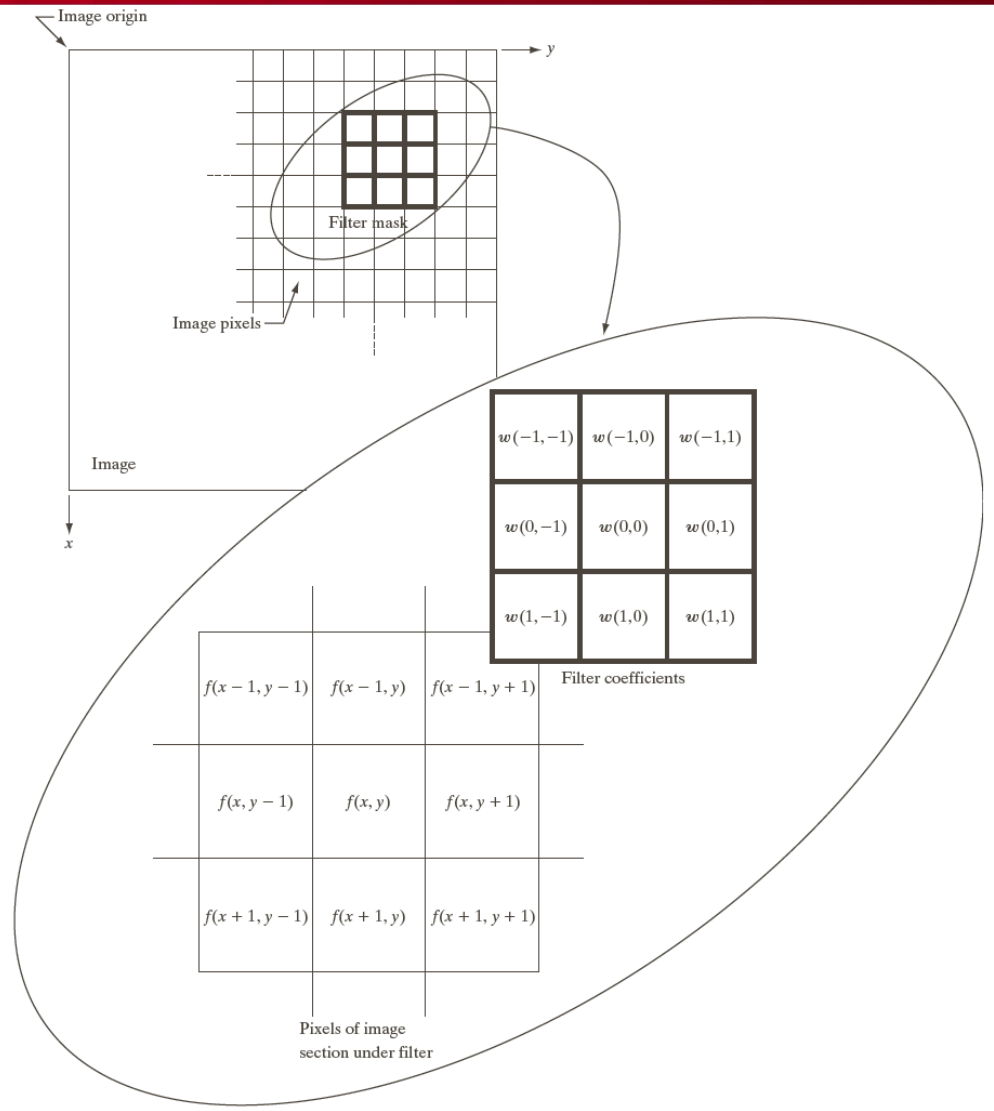
Linear spatial filtering of an image of size M x N with a filter of size m x n.

The correlation of a filter $w(x, y)$ of size $m \times n$ with an image $f(x, y)$, denoted as $w(x, y) \star f(x, y)$

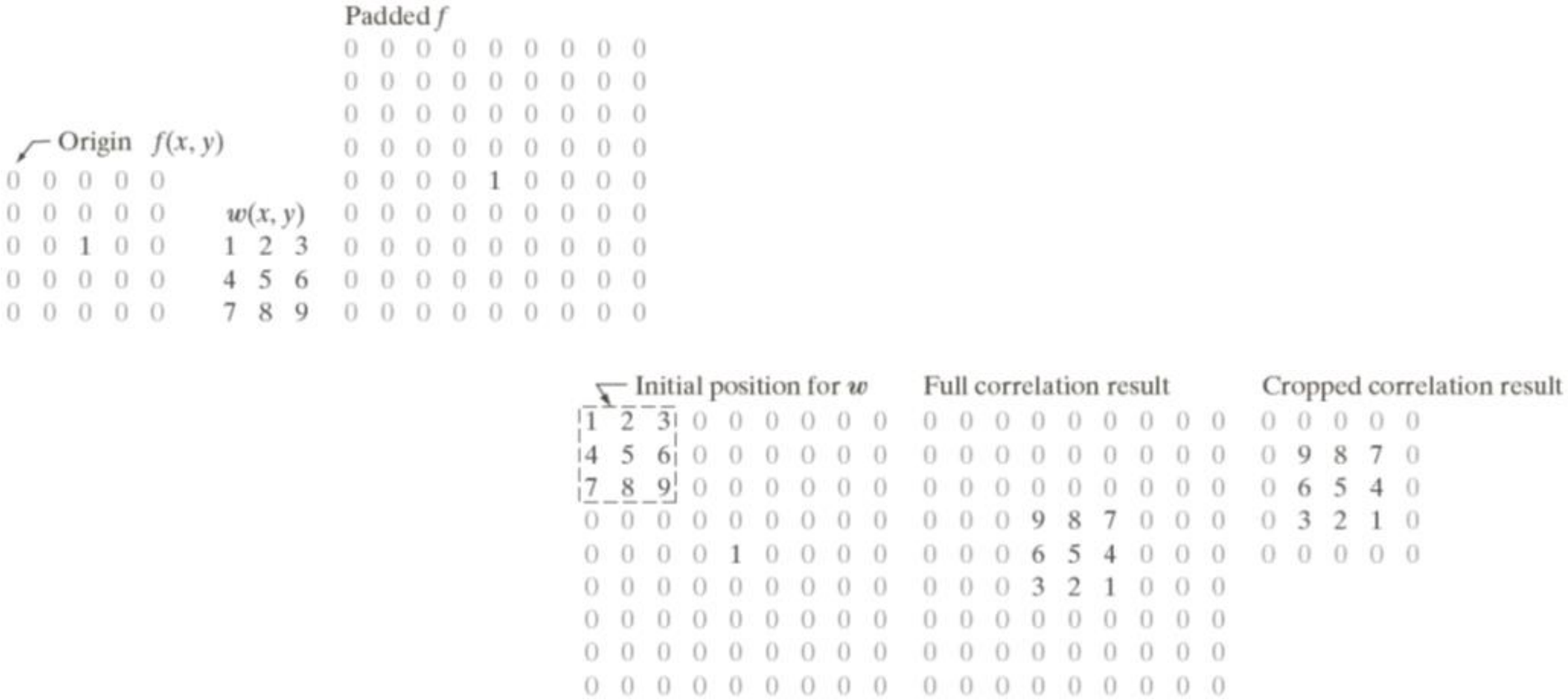
$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$m = 2a+1; \quad n = 2b+1$$

Shifting
Multiplication
Addition



Spatial Correlation



Spatial Filtering

A spatial filter consists of (a) **a neighborhood**, and (b) **a predefined operation.**

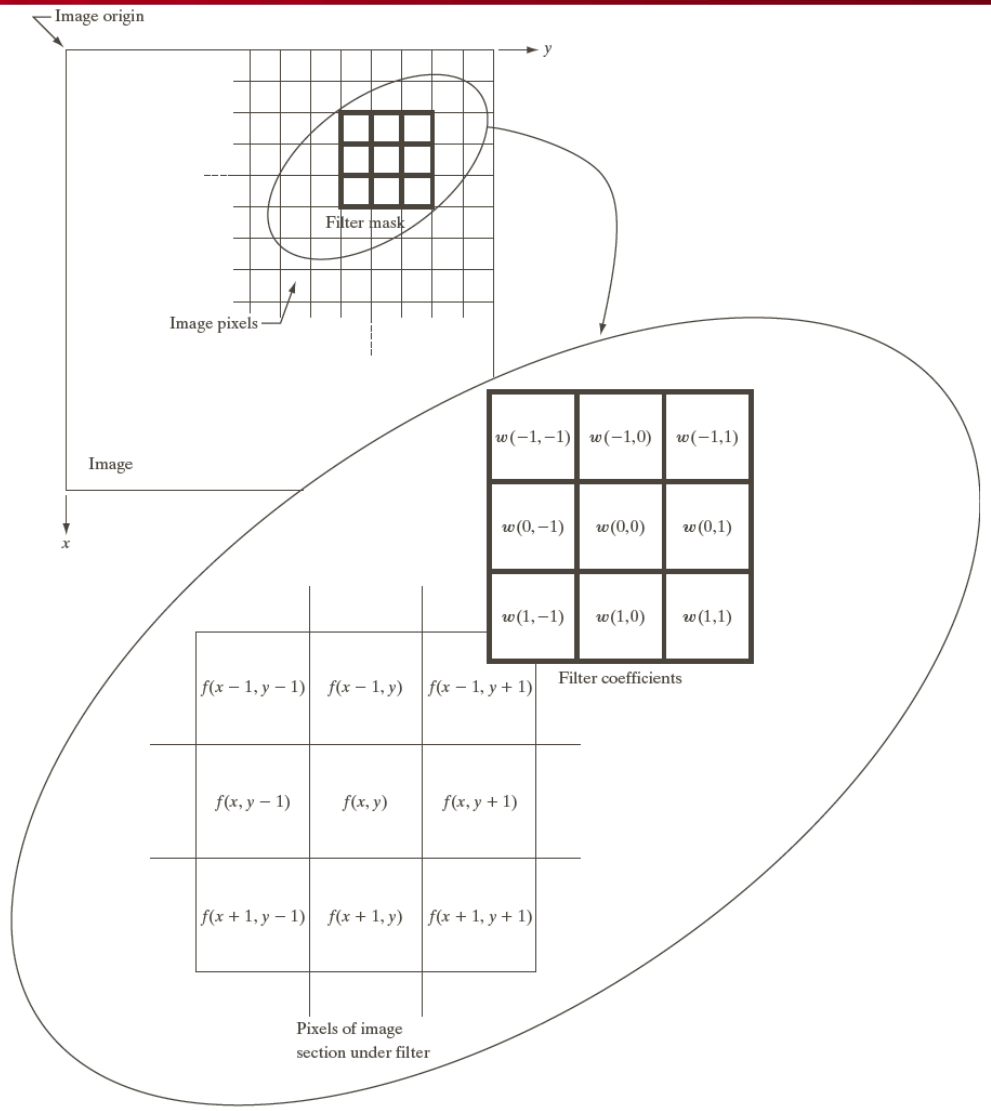
Linear spatial filtering of an image of size M x N with a filter of size m x n.

The convolution of a filter $w(x, y)$ of size $m \times n$ with an image $f(x, y)$, denoted as $w(x, y) \star f(x, y)$

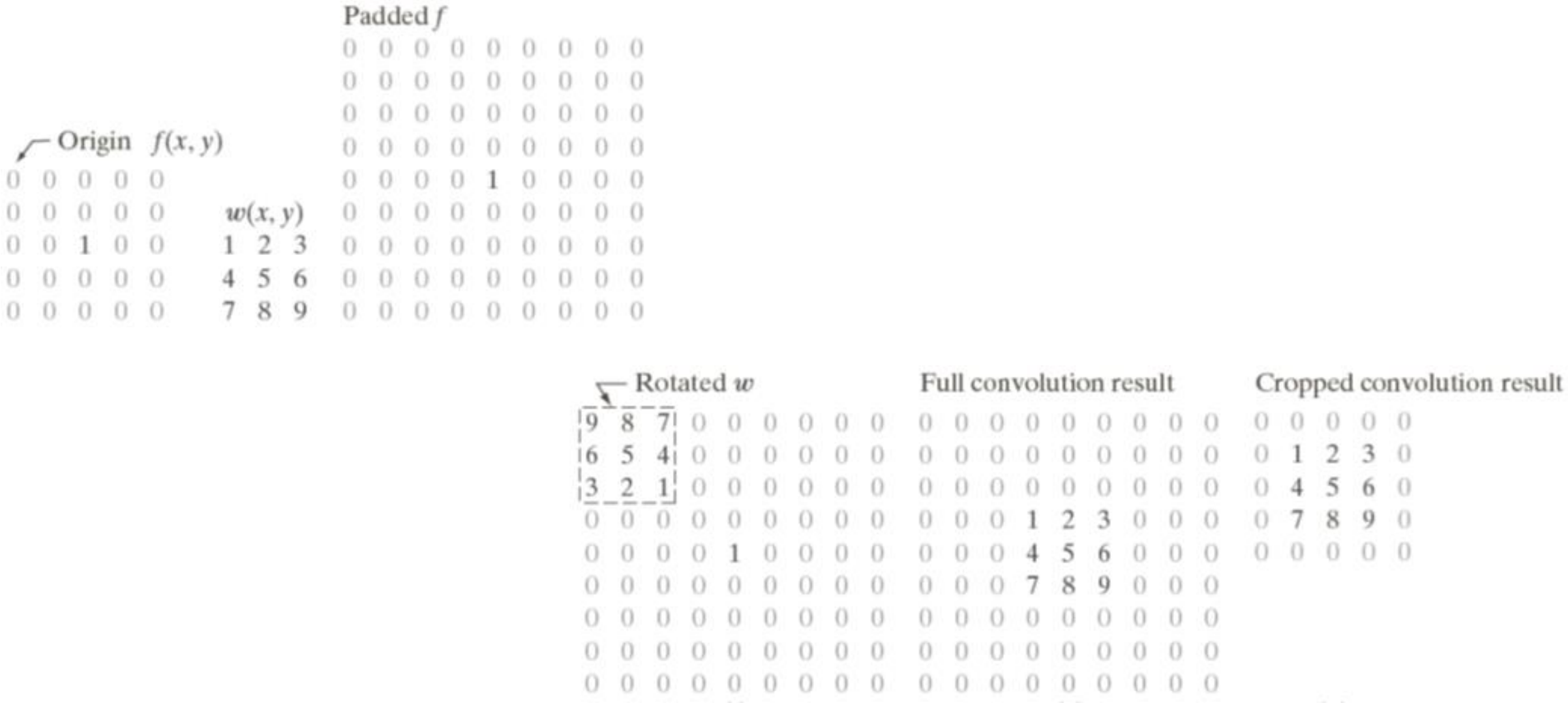
$$w(x,y) \star f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Folding
Shifting
Multiplication
Addition

$$m = 2a+1; \quad n = 2b+1$$



Spatial Convolution



Smoothing Spatial Filters

Smoothing filters are used for blurring and for noise reduction.

Blurring is used in removal of small details and bridging of small gaps in lines or curves.

Smoothing spatial filters include linear filters and nonlinear filters.

The general implementation for filtering an $M \times N$ image with a weighted averaging filter of size $m \times n$

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

where $m = 2a+1, \quad n = 2b+1.$

$\frac{1}{9} \times$

1	1	1
1	1	1
1	1	1

$\frac{1}{16} \times$

1	2	1
2	4	2
1	2	1

Smoothing Averaging Filter Masks

Smoothing Spatial Filters

Original image of size 500 × 500 pixels

Smoothed with a filter mask of size 3 × 3

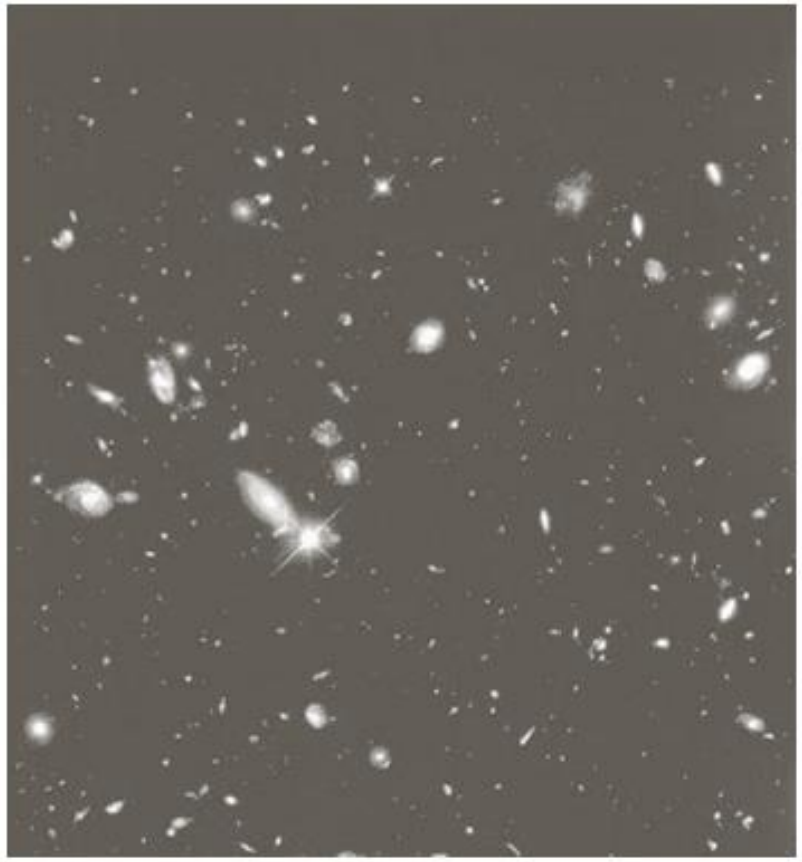
Smoothed with a filter mask of size 5 × 5

Smoothed with a filter mask of size 9 × 9

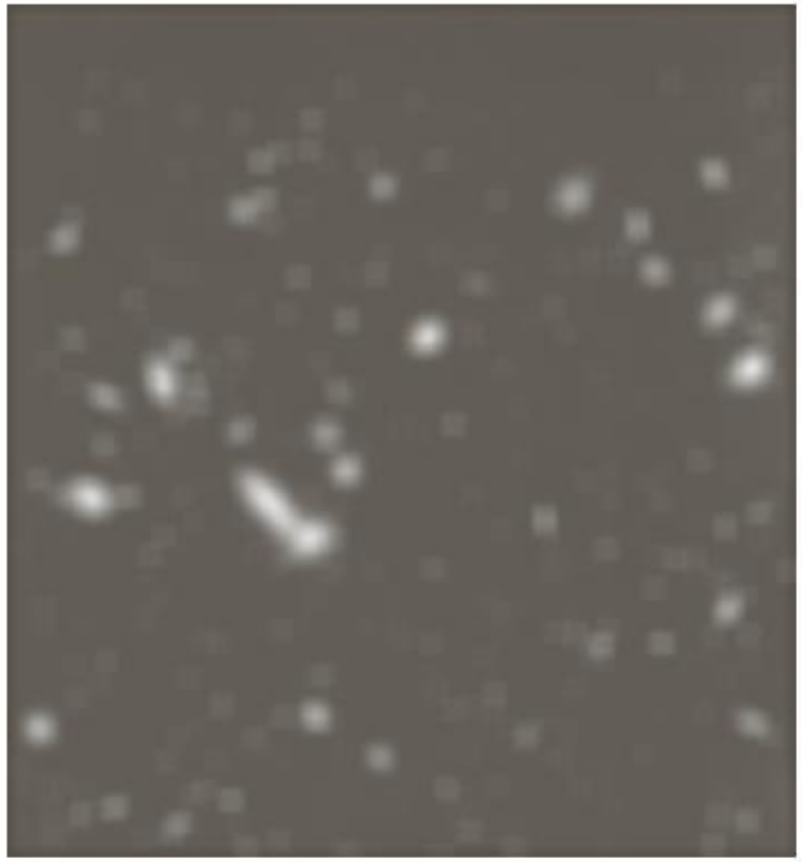
Smoothed with a filter mask of size 15 × 15

Smoothed with a filter mask of size 35 × 35

Image Smoothing and Thresholding



Original image of size 525 × 485 pixels
(*Original telescopic image courtesy of* NASA)



Filtered with averaging mask of size 15 × 15

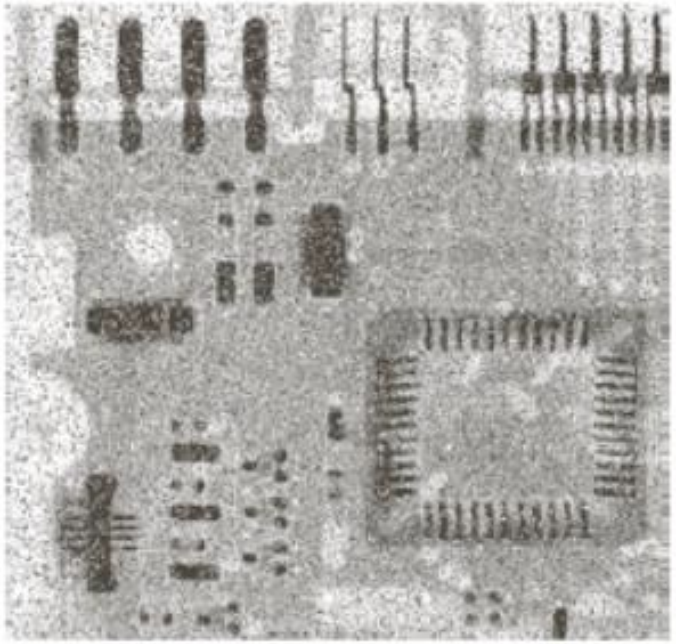
Thresholded image

Order Statistic (Nonlinear) Filters

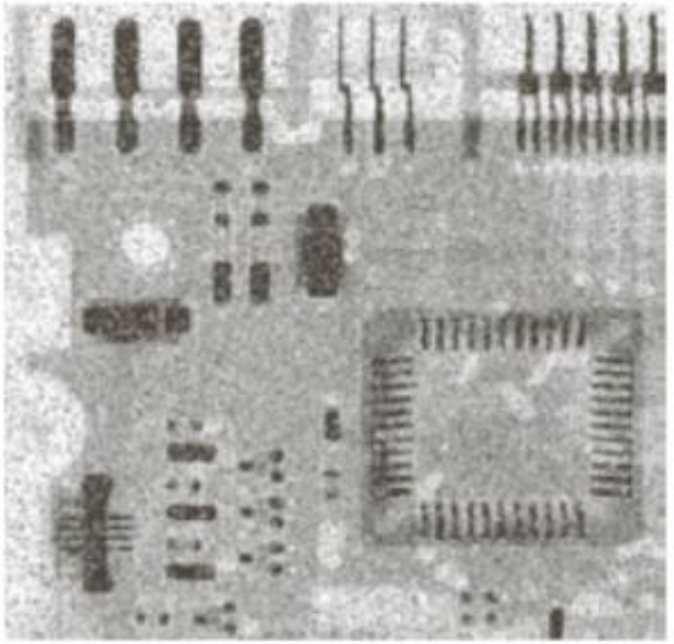
— Based on ordering (ranking) the pixels contained in the filter mask

— Replacing the value of the center pixel with the value determined by the ranking result

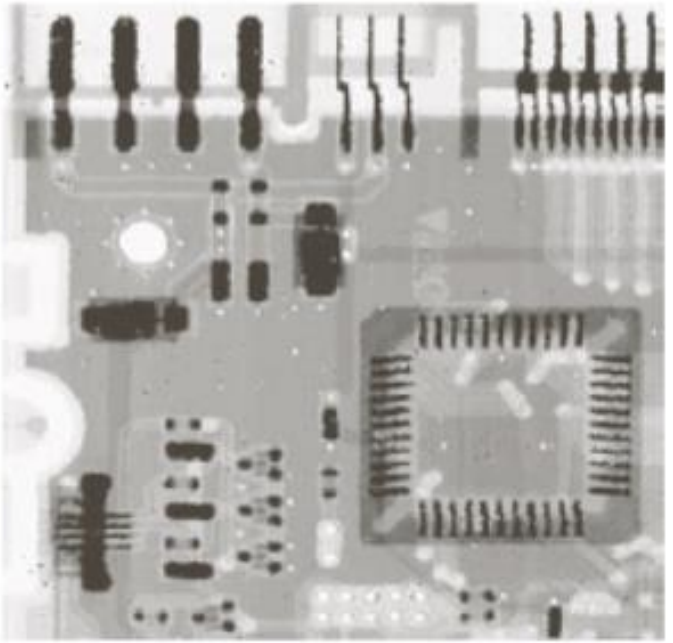
E.g., median filter, max filter, min filter



X-ray image of circuit board corrupted by salt and pepper noise



Noise reduction by a 3×3 averaging mask



Noise reduction by a 3×3 median filter

Derivatives

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

Produces thicker edges in the image

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

Produces sharper edges in the image

First Order Derivatives: Gradient Image

For function $f(x, y)$, the gradient of f at coordinates (x, y) is defined as

$$\nabla f \equiv \text{grad}(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The *magnitude* of vector ∇f, denoted as $M(x, y)$

$$M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^{\,2} + g_y^{\,2}}$$

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

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$M(x, y) = |z_8 - z_5| + |z_6 - z_5|$$

Gradient Image: Sobel Operators

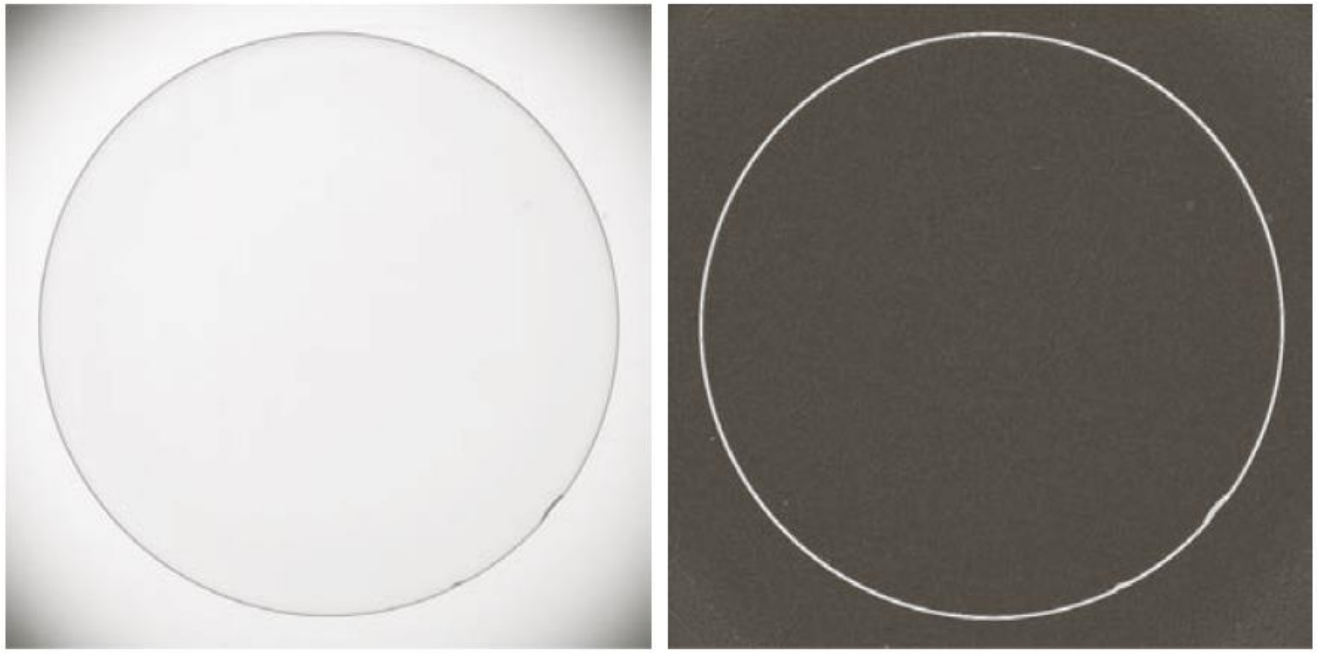
z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Sobel Operators

$$M(x,y) \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1



Sharpening Filters

Laplacian Operators

0	1	0	1	1	1	
1	−4	1	1	−8	1	$c = -1$
0	1	0	1	1	1	

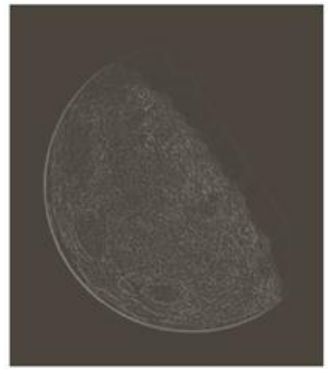
0	−1	0	−1	−1	−1	
−1	4	−1	−1	8	−1	$c = 1$
0	−1	0	−1	−1	−1	

Image sharpening in the way of using the Laplacian

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

$f(x,y)$ is input image
$g(x,y)$ is sharpenend image


Blurred image


Laplacian without scaling


Laplacian with scaling


Sharpened image


Sharpened image

General Structure of Object Detection

Input Dataset

Feature Extraction

Classification

Class1 (*Positive*)

Class2 (*Negative*)

Feature Extraction

Feature Extraction

Local Binary Patterns

g1	g2	g3
g8	gc	g4
g7	g6	g5

- Obtains the local neighborhood differences even in difficult lighting conditions
- Differences between the neighbor pixels (g_1 to g_8) and central pixel (g_c)

  $d_i = g_c - g_i$

  where d_i is the difference between the central pixel and the i[th] neighbor pixel; i = 1,2,…,8
- Compare d_i with 0

$$s_i = \begin{cases} 1 & if\ d_i \geq 0 \\ 0 & else \end{cases}$$

- Sum the thresholded value weighted by powers of two

$$LBP_c = \sum_{i=1}^{8} s_i \cdot 2^i$$

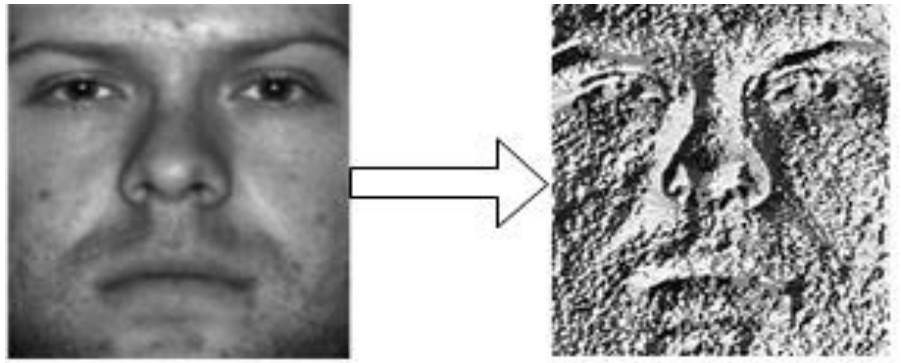
- The new value of the central pixel is 224.

150	150	121
93	121	93
93	93	53

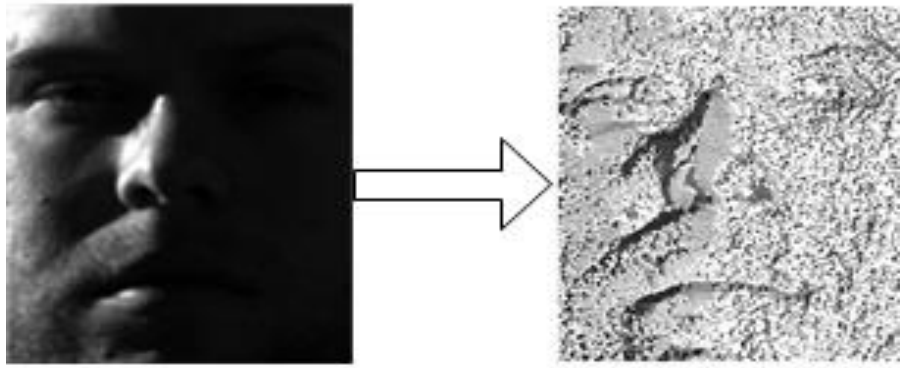
1	1	1
0		0
0	0	0

$(11100000)_2 = (224)_{10}$

Local Binary Patterns (LBP)



Original Image LBP Image



Original Image LBP Image

Local Edge/Corner Feature Integration (LFI)

- LFI is a nonparametric method
- It extracts local structures of images efficiently by:
  - Convolving the image with Frei-Chen masks,
  - Comparing each pixel with its neighboring pixels from edge/corner responses separately, and
  - Combining these thresholding responses to form the final code.
- The proposed LFI technique can be summarized into three stages:
  - Edge/corner detection
  - Binary encoding and decoding
  - Feature integration

Corner/Edge Detection

- Using Frei-Chen edge detector
  - It is nine convolution masks
  - Works on a 3×3 window size
  - The first four masks k_i , $i = 1, \dots, 4$ are used to find the edges' subspace.
  - To detect the corners, the second four kernels are utilized k_i , $i = 5, \dots, 8$
  - The last one k_9 is used to compute the mean to use as a normalization factor

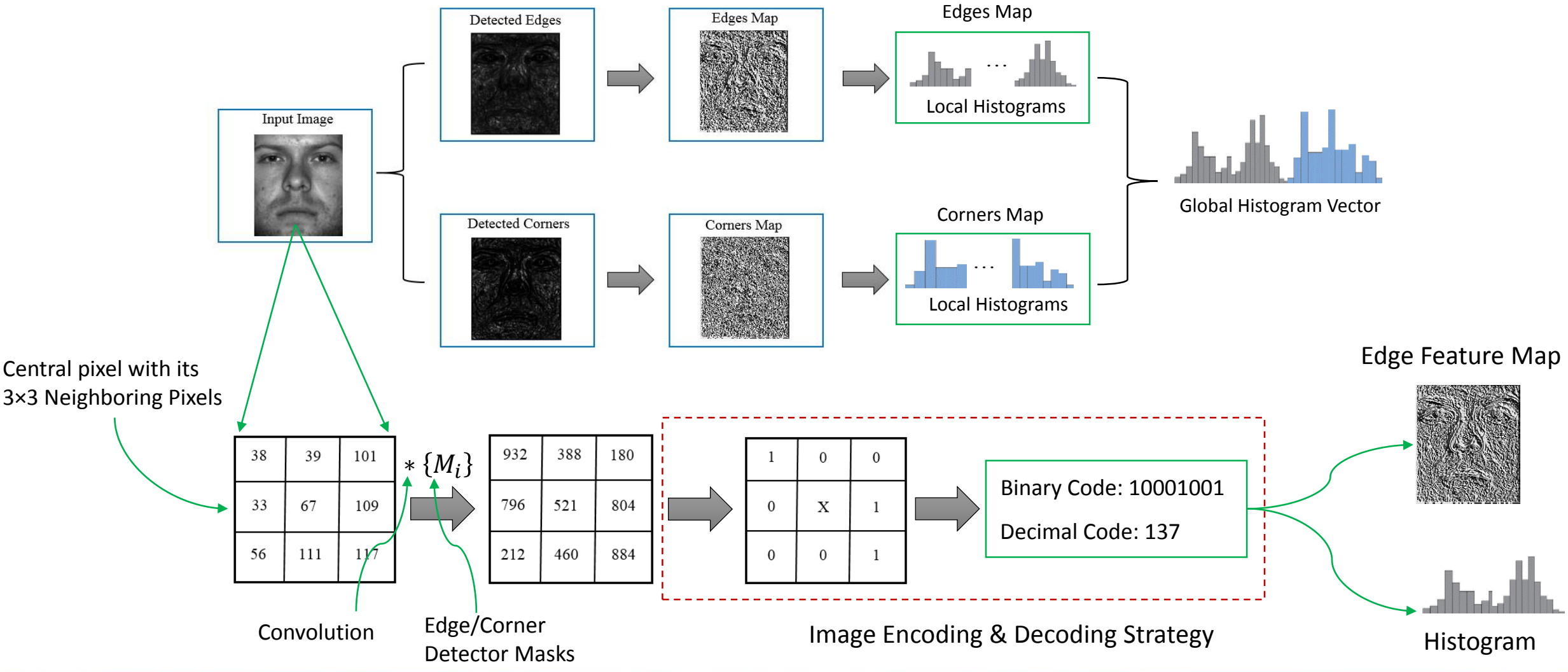
$$K_1 = \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \quad K_2 = \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix} \quad K_3 = \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}$$

$$K_4 = \begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix} \quad K_5 = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad K_6 = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

$$K_7 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad K_8 = \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \quad K_9 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The nine Frei-Chen masks

Overview of LFI Technique



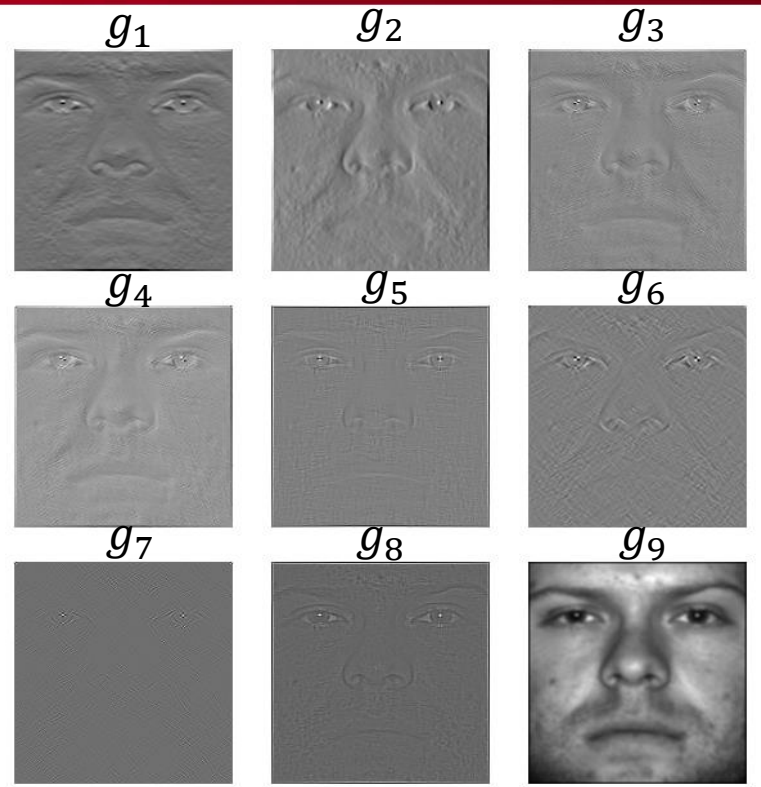
Corner/Edge Detection

- Given an input image $I(x, y)$
  - The nine different edge, corner, and mean responses g_i can be computed by

$$g_i = I(x, y) * K_i, \qquad i = 1,2, \dots, 9$$

  - The projection equations for edge detection (E) and corner detection (C) given as

$$E = \sqrt{\frac{\sum_{i=1}^{4} g_i^2}{\sum_{i=1}^{9} g_i^2}} \qquad C = \sqrt{\frac{\sum_{i=5}^{8} g_i^2}{\sum_{i=1}^{9} g_i^2}}$$



Input Image | Detected Edges | Detected Corners

Image Encoding and Decoding

- Once the edges and corners are detected separately, a binary coding strategy is applied by:
  - Comparing the center pixel value in each 3×3 neighborhood regions
  - If a neighbor pixel has a higher edge/corner value than the center pixel (or the same value) then a 1 is assigned to that pixel, which is otherwise a 0.
  - Finally, retrieve the edge and corner feature map by

$$LFI = \sum_{p=1}^{8} f(d_p - d_c) \times 2^{p-1}$$

$$f(x) = \begin{cases} 1 & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases}$$

where d_p and d_c denote the edge or corner values of the central pixel and its neighbors respectively

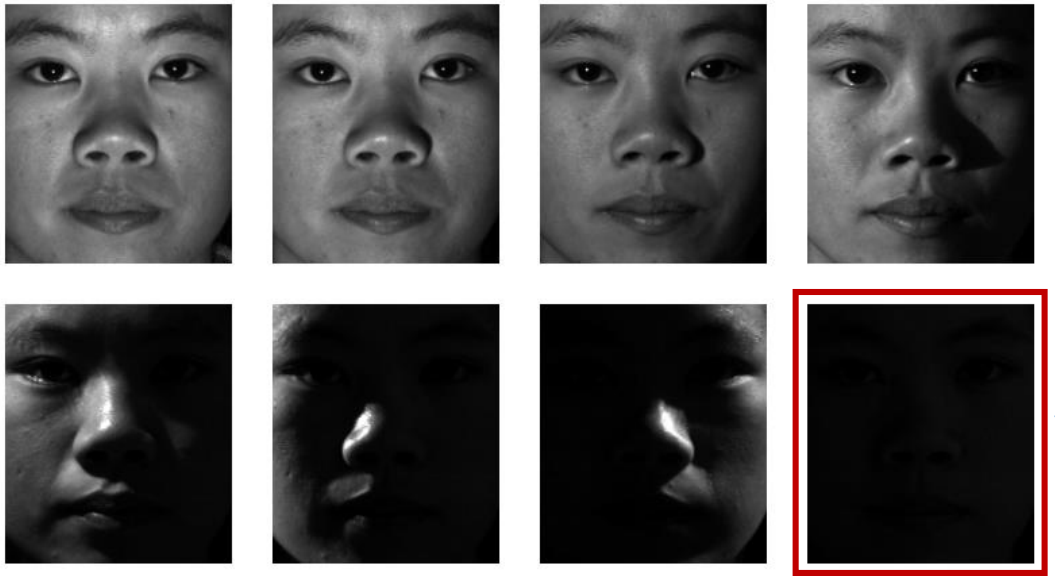
Input Image

Detected Edges Map

Detected Corners Map

Test Results

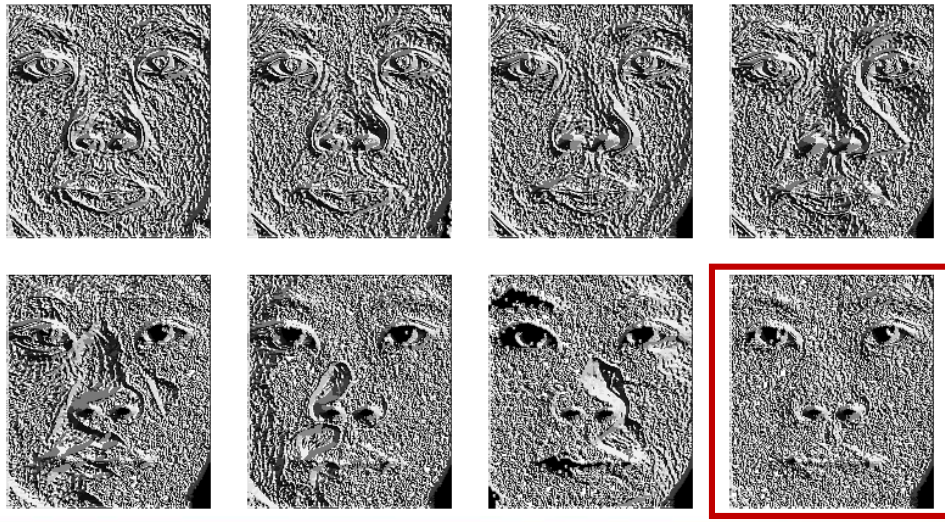
Samples of one subject from the Extended Yale B database

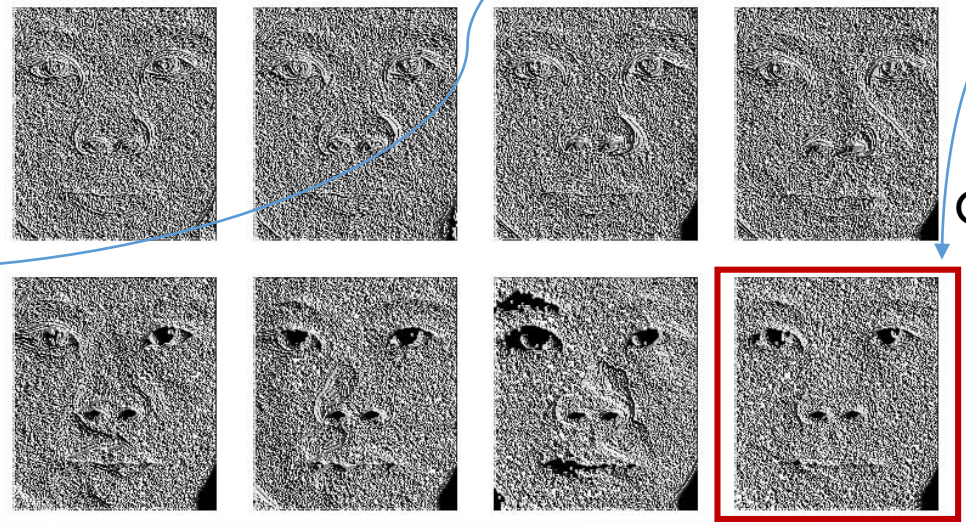


Under extremely dark illumination condition

LFI has the ability to get face information

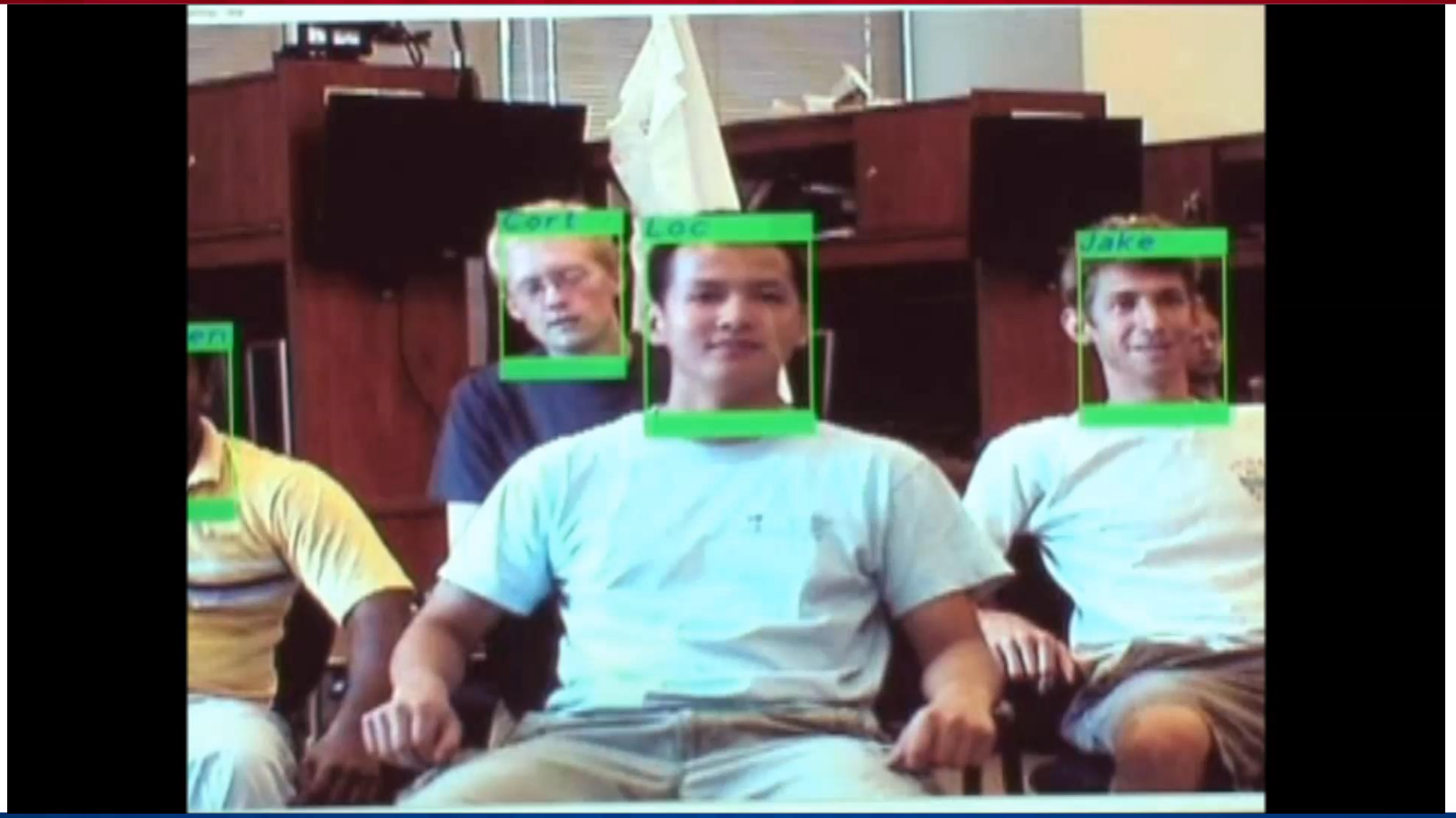
Edge Maps





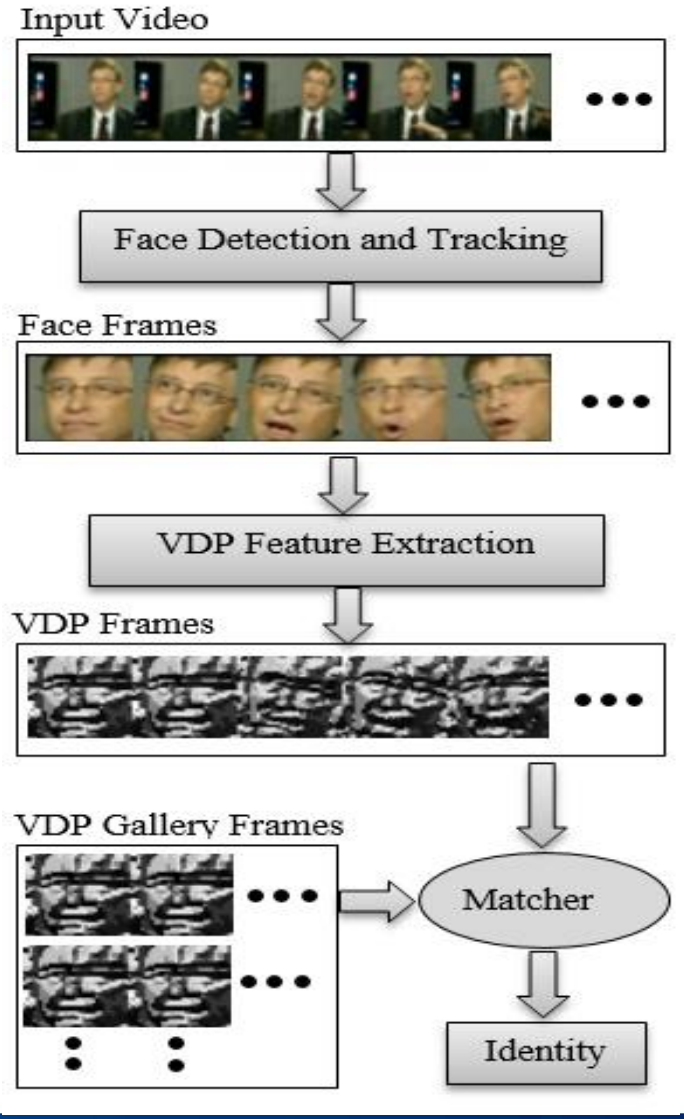
Corner Maps

Face Recognition



Video-to-Video Face Recognition: Volumetric Directional Pattern

- Volumetric Directional Pattern (VDP) is a local appearance based feature set.
  - Basically it is a fine-scale descriptor that captures small textural details.
- VDP is a gray-scale pattern that characterizes and fuses the temporal structure (dynamic information) of three consecutive frames in a video stream.
- VDP computes the edge response values in different directions at each pixel position, and uses the relative strength (magnitude) to encode the image texture.
- VDP is an eight bit binary code assigned to each pixel of an input frame.



VDP Concept and Algorithm

- VDP can be calculated by comparing the relative edge response value of a particular pixel from three consecutive frames in different directions by using Kirsch masks in eight different orientations ($M_0 - M_7$).

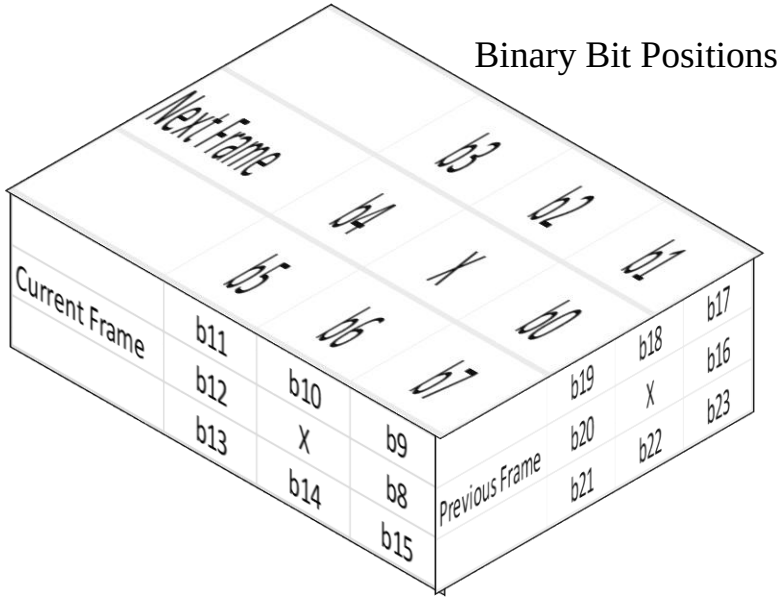
Kirsch edge masks in all eight directions

$$M_0 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad M_1 = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix} \quad M_2 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad M_3 = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad M_5 = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix} \quad M_6 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad M_7 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

VDP Concept and Algorithm

- Given a central pixel in the current frame (middle frame) of three consecutive frames:
  - The eight different directional edge response values are:
    - $c_i(i = 8, 9, \ldots, 15)$ is a binary code for each pixel in the current frame.
    - $p_i(i = 16, 17, \ldots, 23)$ is a binary code of the previous frame.
    - $n_i(i = 0, 1, \ldots, 7)$ is a binary code of the next frame.



$$p_i = \sum_{i=16}^{23} dot(I_{3\times3}, M_{i-16})$$

Dot product operation

$$c_i = \sum_{i=8}^{15} dot(I_{3\times3}, M_{i-8})$$

3×3 Neighbors

$$n_i = \sum_{i=0}^{7} dot(I_{3\times3}, M_i)$$

Mask

$b_{19}b_{11}b_3$	$b_{18}b_{10}b_2$	$b_{17}b_9b_1$
$b_{20}b_{12}b_4$	X	$b_{16}b_8b_0$
$b_{21}b_{13}b_5$	$b_{22}b_{14}b_6$	$b_{23}b_{15}b_7$

Fusing Three Binary bits

VDP Concept and Algorithm

- In order to generate the VDP-feature:
  - We need to know the most prominent Dynamic Features (*DF*):
    - Where *DF* is the mean of 3×3 neighbors of each frame.

- Finally, the VDP code can be derived by:

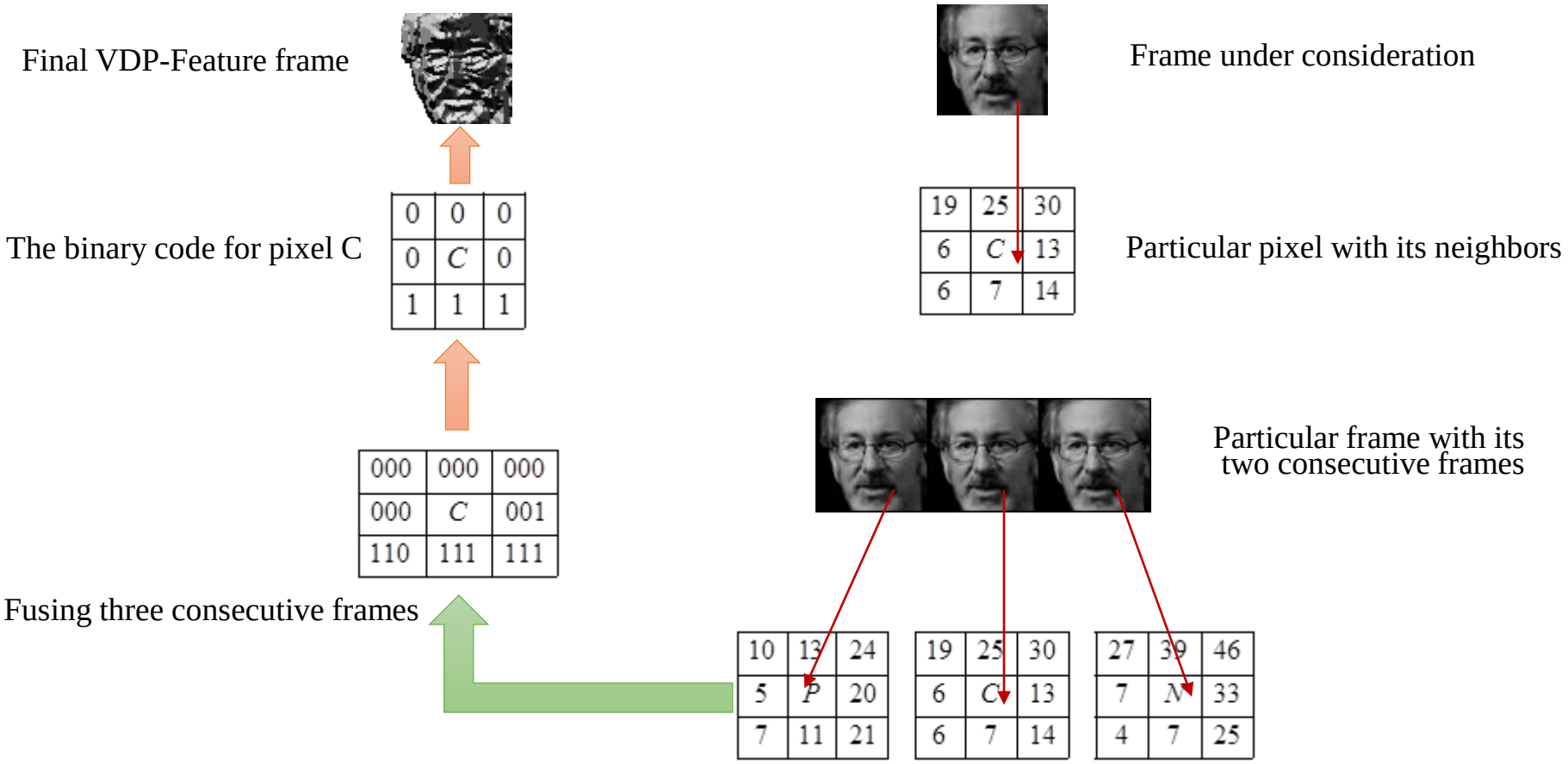
$$VDP = \sum_{i=0}^{7} bitmajority_i \{b_i(n_i - n_{DF}) \times 2^i || b_{(i+8)}(c_{(i+8)} - c_{DF}) \times 2^i \ || \ b_{(1+16)}(p_{(i+16)} - p_{DF}) \times 2^i\}$$

where: $b_i(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$

p_{DF}, c_{DF} *and* n_{DF} are the DF^{th} most prominent dynamic features for all three consecutive frames.

The 8-bit VDP code is obtained by selecting the majority bits of each concatenated binary bits estimated for three consecutive frames.

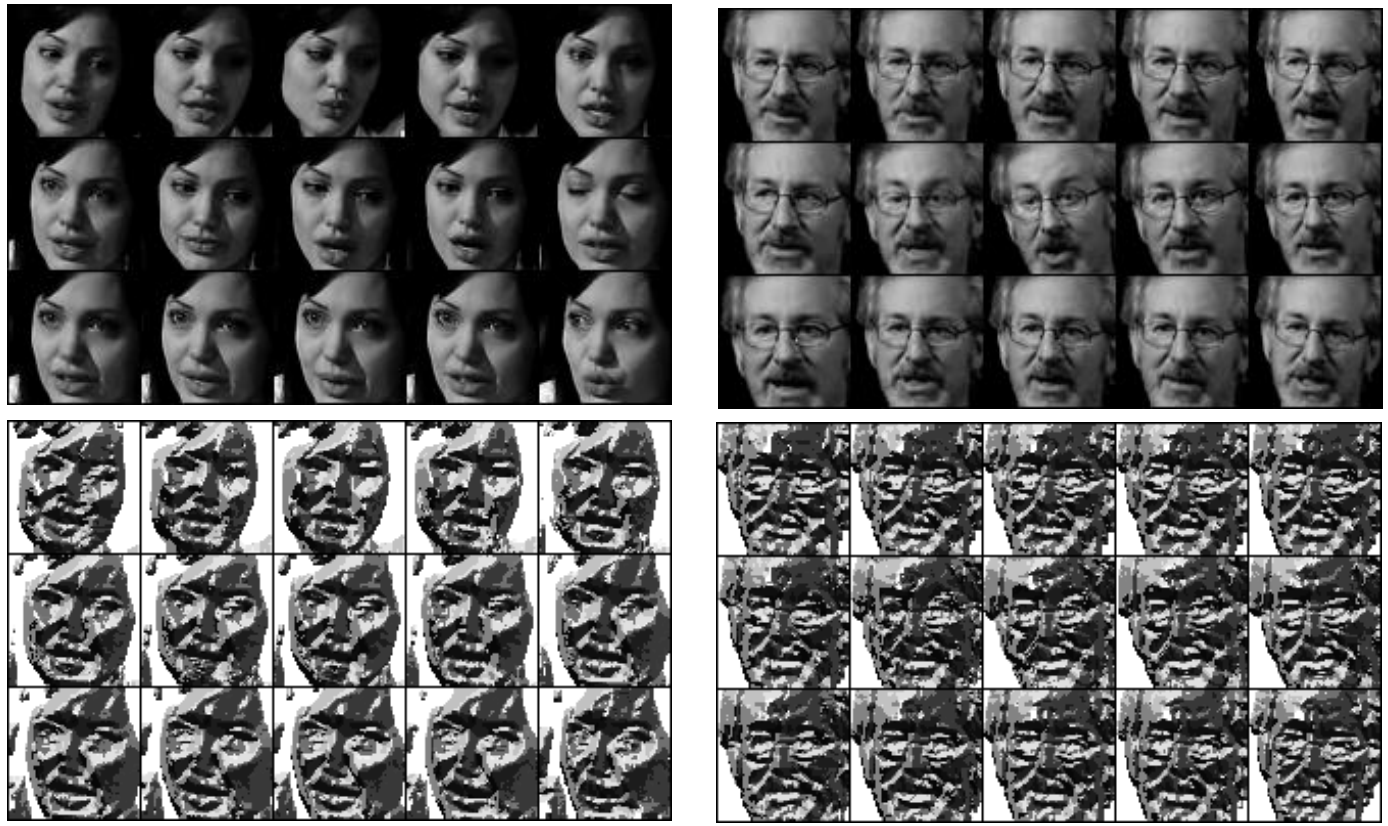
An Example of VDP Code Computation



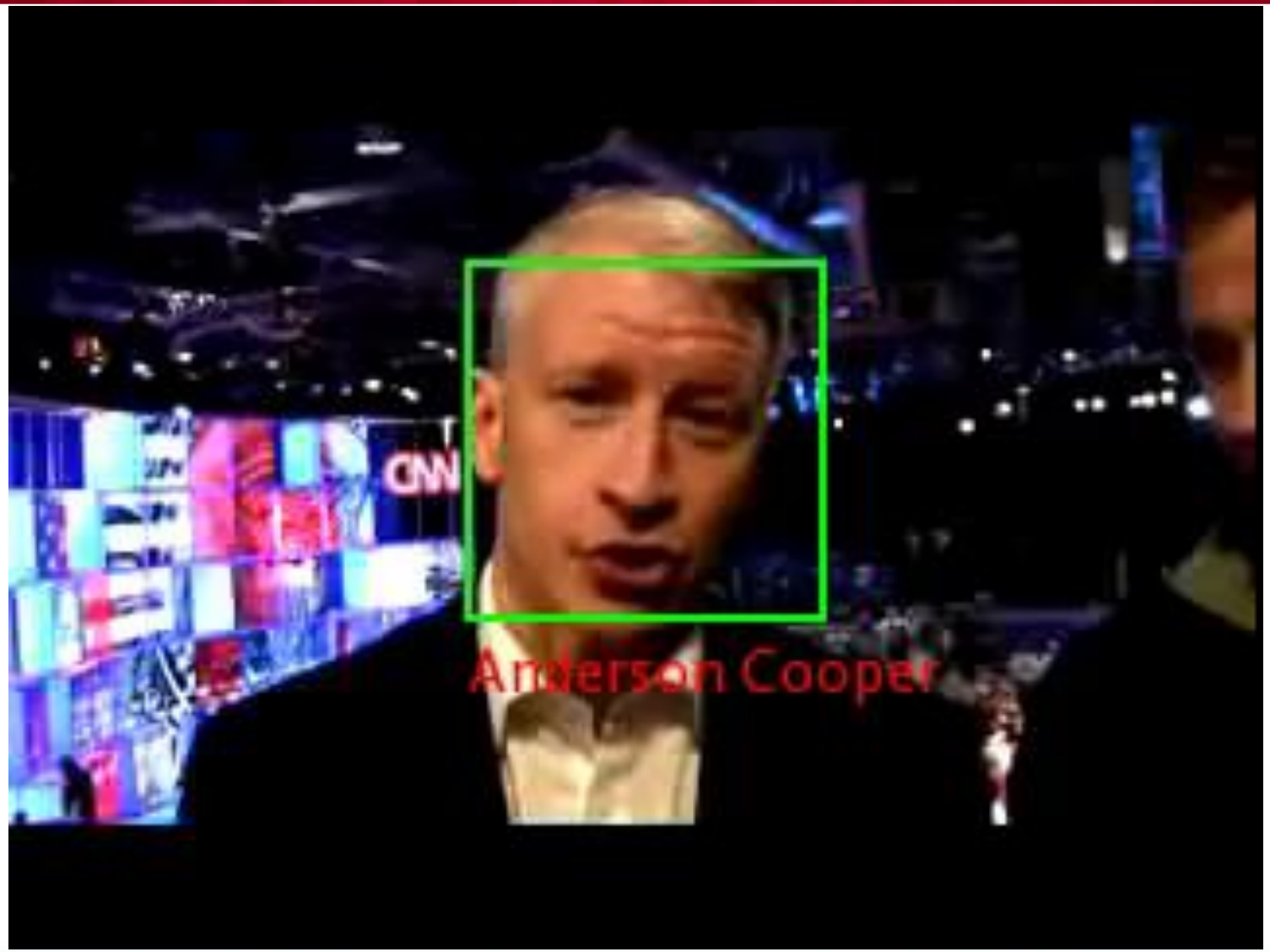
YouTube Celebrities Dataset

- It is a video dataset which contains 1910 video sequences of 47 different subjects (celebrities).

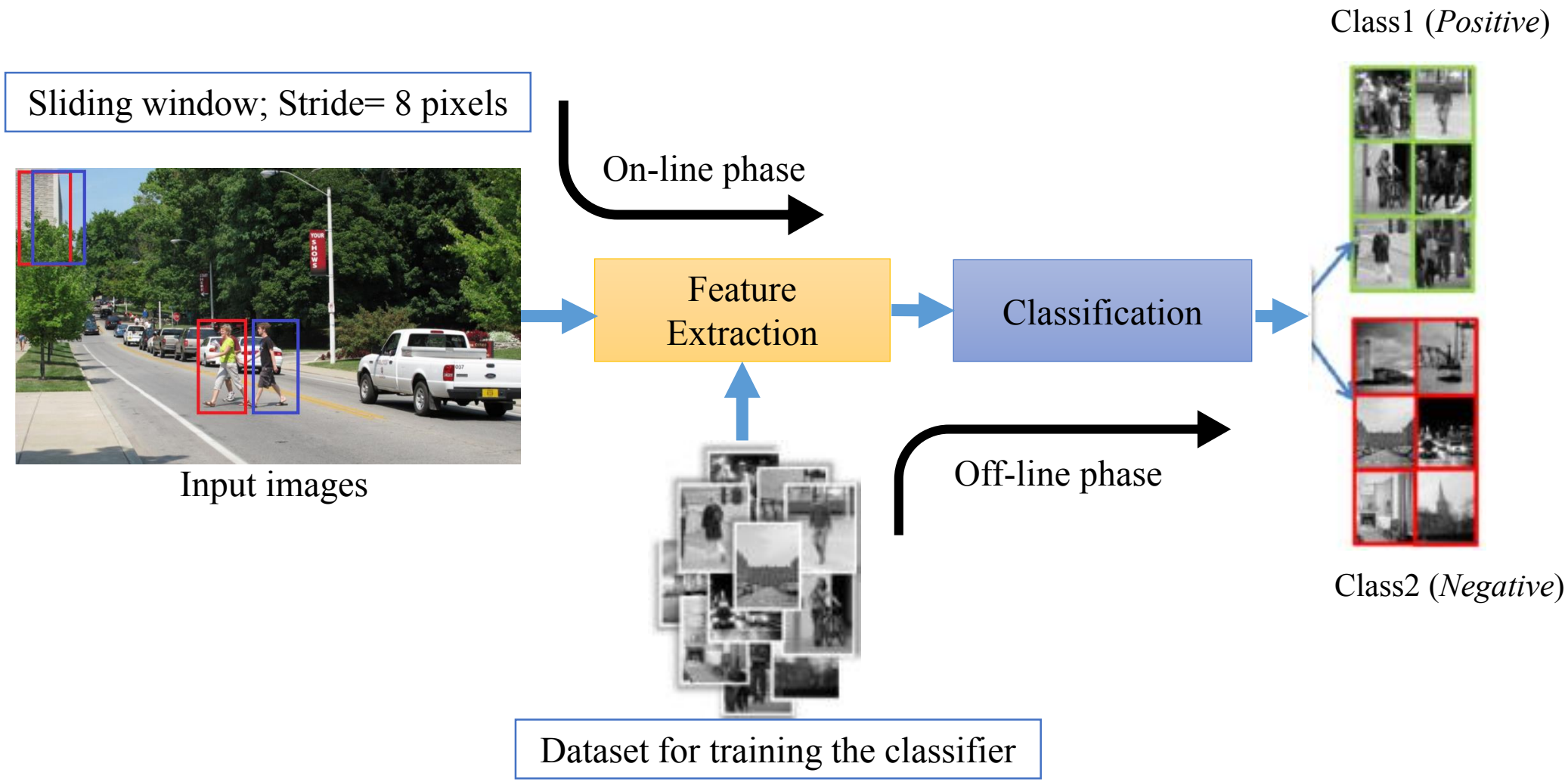
Some samples of YouTube celebrities' dataset and their VDP features



YouTube Celebrities Face Recognition Demo



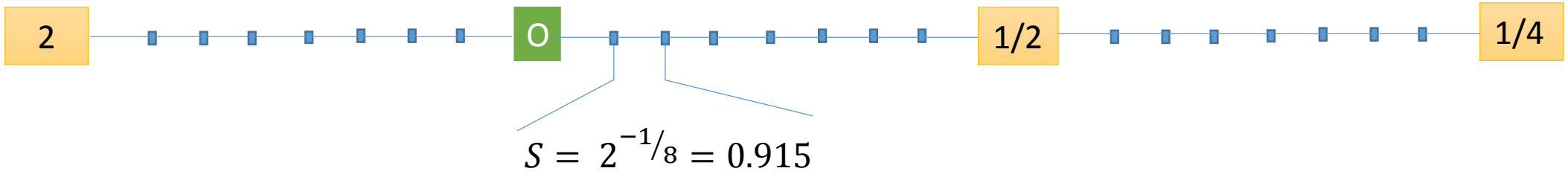
General Structure of Human Detection



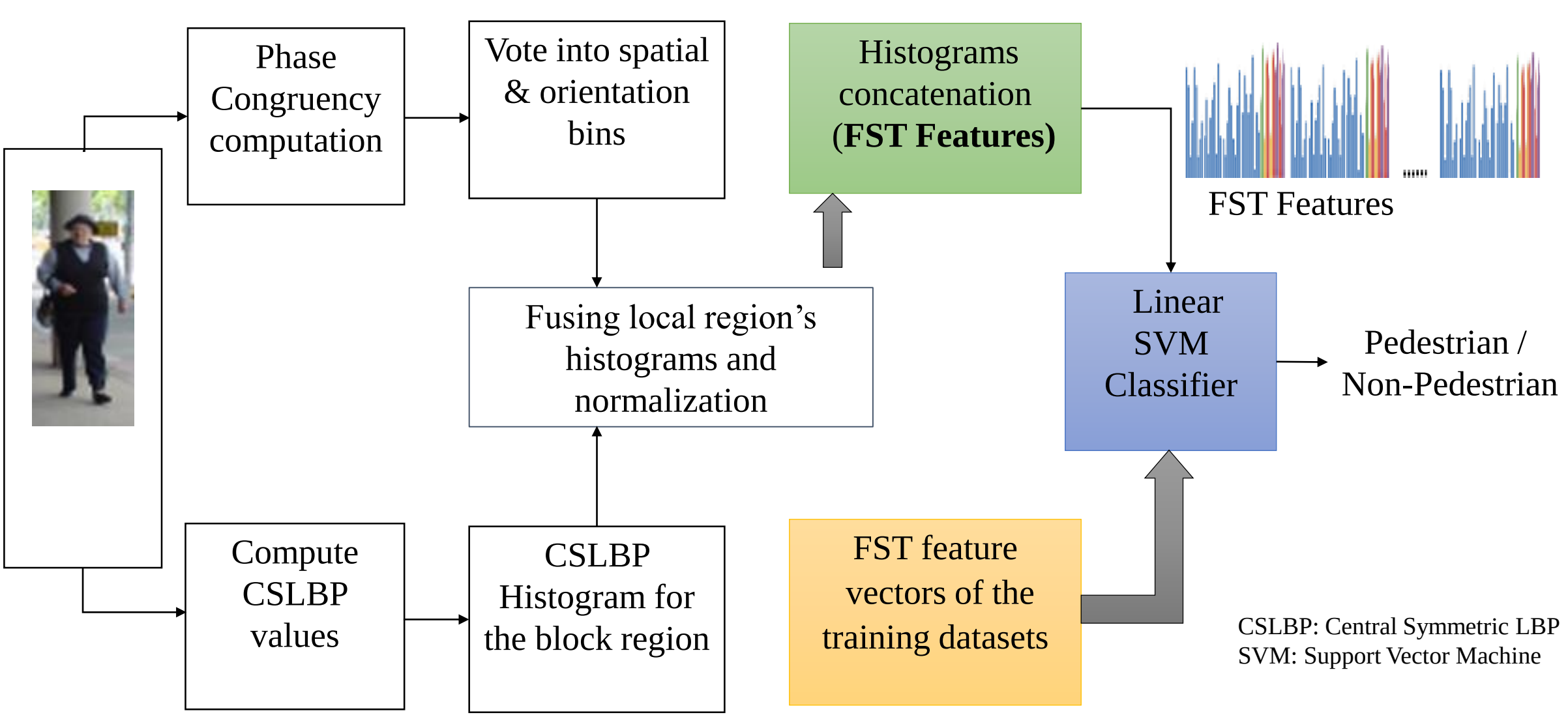
Multi-scale Human Detection

1-model , N- image scales

N-model , 1- image scales



Fused Shape and Textural (FST) Features



CSLBP: Central Symmetric LBP
SVM: Support Vector Machine

Importance of Phase in Images

Phase of the signal conveys more information regarding signal structure than the magnitude.

Image 1 (Img1)

Image 2 (Img2)

Amplitude (Img1), Phase (Img2)

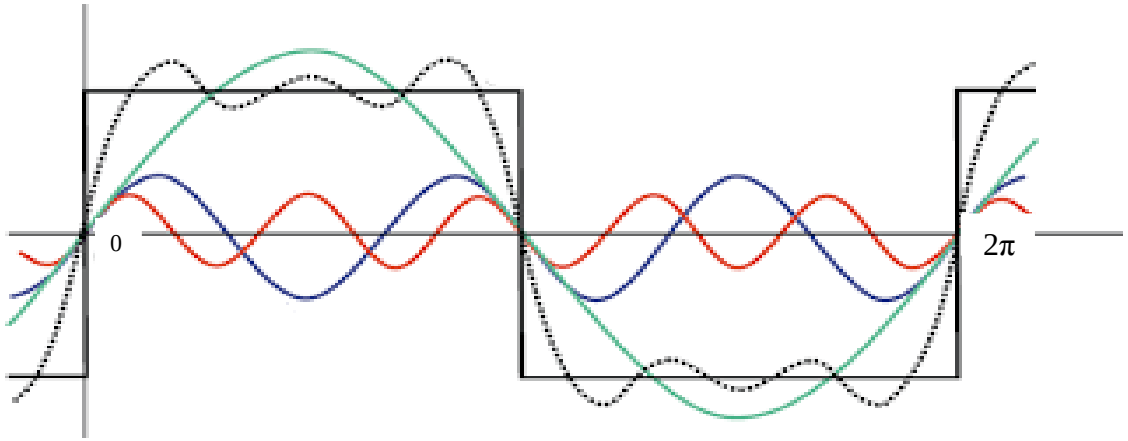
Amplitude (Img2), Phase (Img1)

Phase Congruency Computation

- Let us start with a signal $I(x)$ defined in the range $[0,2\pi]$.
  - All the Fourier components meet at the edges where the phase congruence PC is maximum (have the same phase).
  - Since the Fourier components meet at the edges, the local energy $E(x)$ would be also maximum at these points
  - Therefor the phase congruency is directly proportional to the energy.
  - The phase congruency becomes a dimensionless quantity by the normalization with the sum of all Fourier components amplitudes A_n.
  - Hence, the local energy is given by:

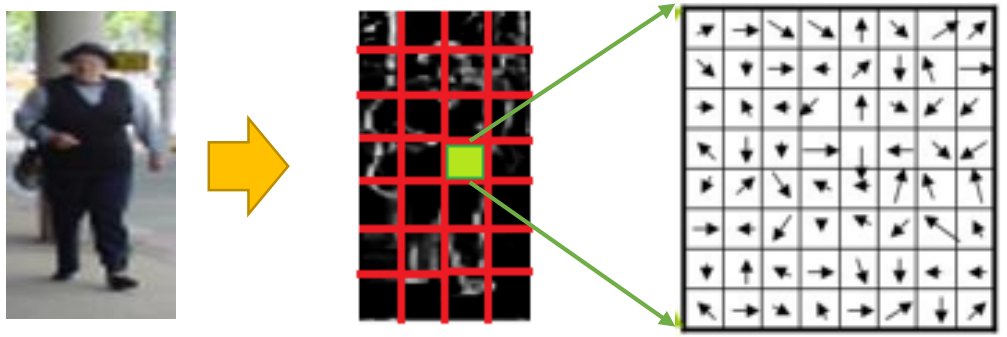
$$E(x) = PC(x)\sum_n A_n \quad \Rightarrow \quad PC(x) = \frac{E(x)}{\varepsilon + \sum_n A_n}$$

where ε is a small number to avoid division by zero.



The Fourier components of a step signal.

Phase Congruency Computation



Input image and its corresponding Phase congruency

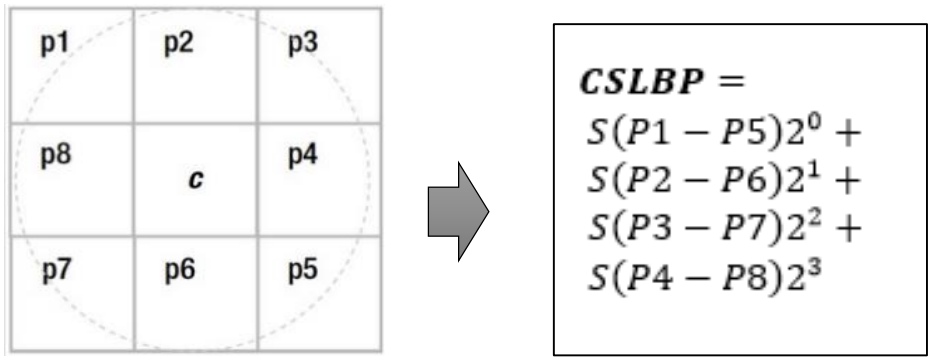
Phase congruency of the image at various illumination levels

Center Symmetric Local Binary Pattern (CSLBP)

- CSLBP is used to capture the texture features of the image.
- The texture information is obtained by replacing each pixel in the image by the pixel's CSLBP value.
- The CSLBP features can be computed by:

$$CSLBP = \sum_{i=1}^{N/2} S(P_i - P_{i+(N/2)})\, 2^i \qquad S(z) = \begin{cases} 1 & if\ z \geq t \\ 0 & otherwise. \end{cases}$$

where, P_i is the gray values of the neighbor pixel, N is the number of the neighbors (the figure below shows $N = 8$), C is the gray level of the center pixel. t is threshold value ($t = 0.1\ is\ selected$),



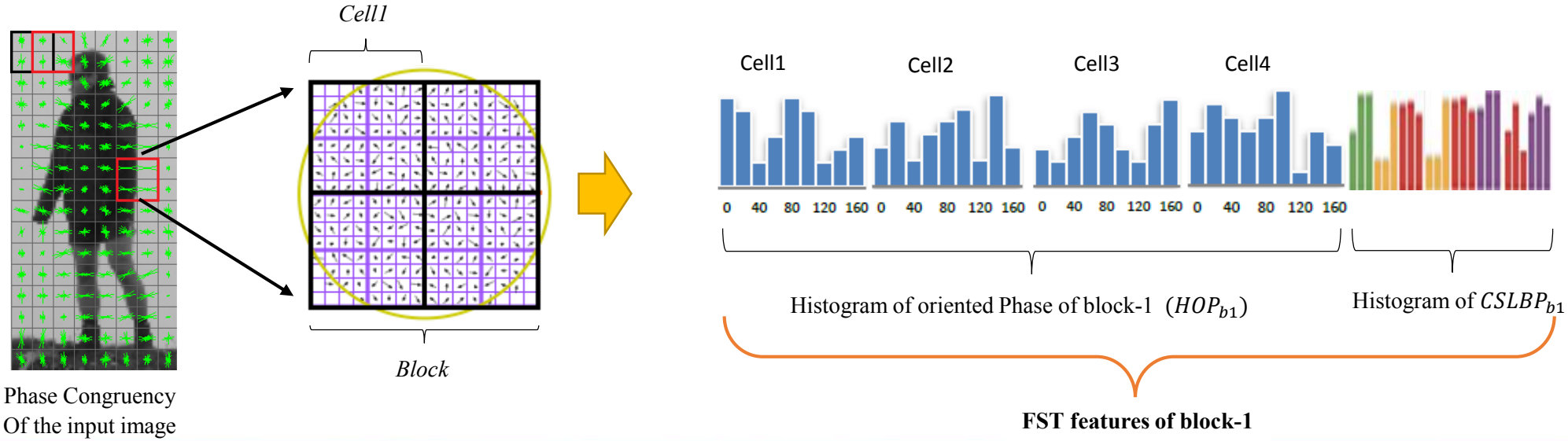
$$\boldsymbol{CSLBP} = S(P1 - P5)2^0 + S(P2 - P6)2^1 + S(P3 - P7)2^2 + S(P4 - P8)2^3$$

Example of CSLBP Images

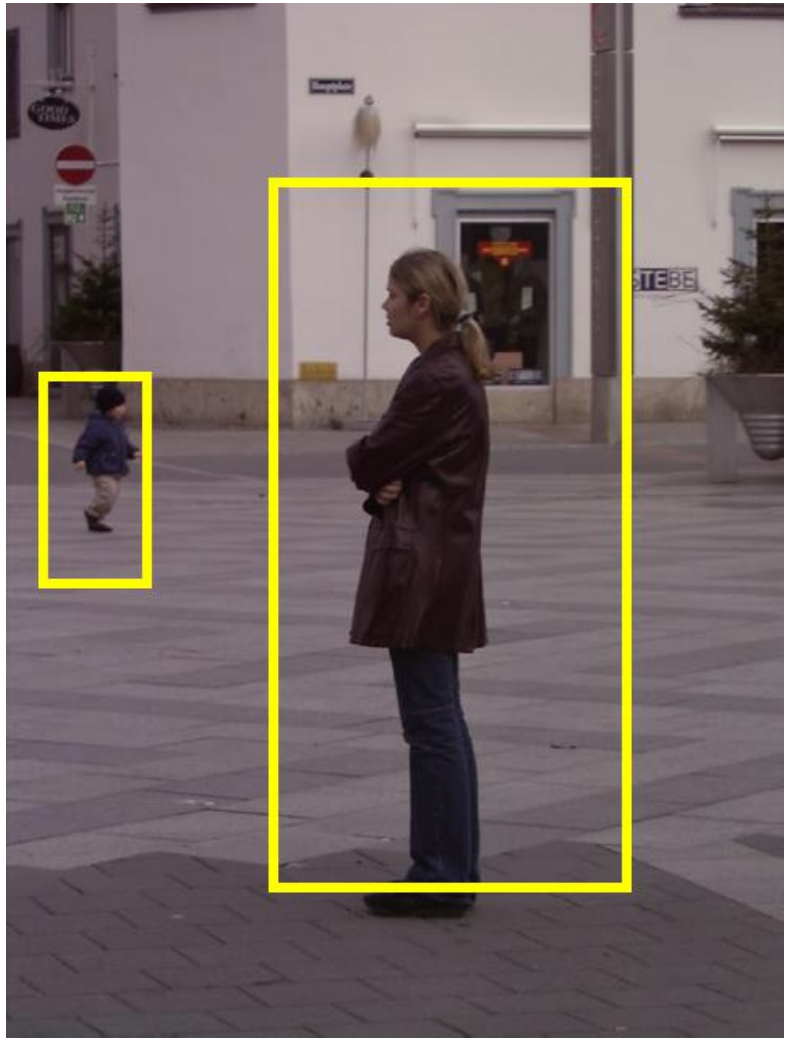



FST Features of a Block Region

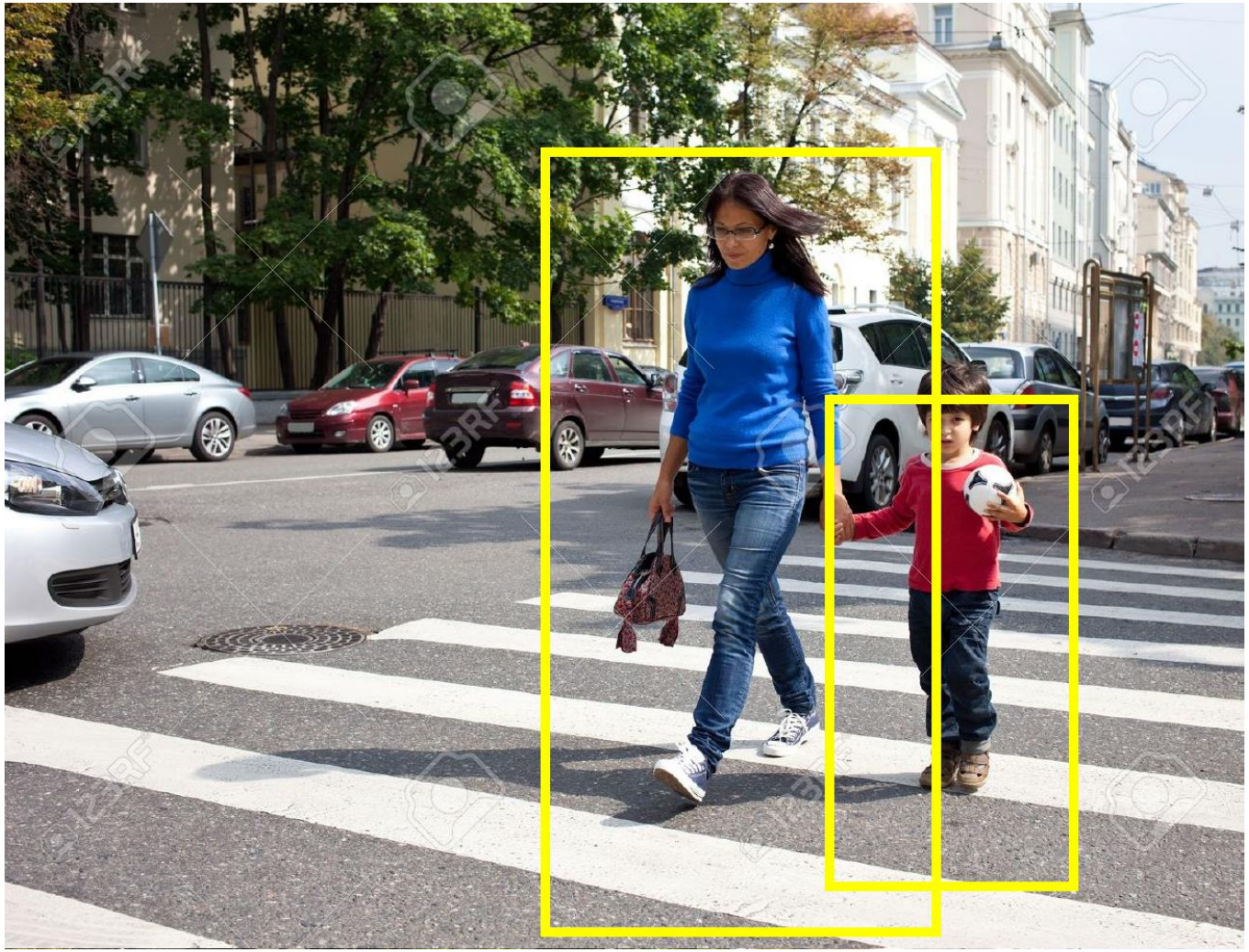
- For an image of the size 128×64 pixels.
  - *Divide the image* into *blocks* (size 16×16 pixels), 50% overlap.
  - Therefore, total No. of *blocks*= 7×15= 105 *blocks*.
  - Each *block* should consists of 2×2 *cells* with size 8×8 pixels.
  - Quantize the gradient orientation into 9-bins
    - **Vote:** The phase congruency magnitude value of each pixel in the cell is contributing to the histogram by adding its value to the corresponding orientation bin.
    - Interpolate votes bi-linearly between neighbor bin center.
  - CSLBP histogram is computed for the *block* region (16×16 pixels)



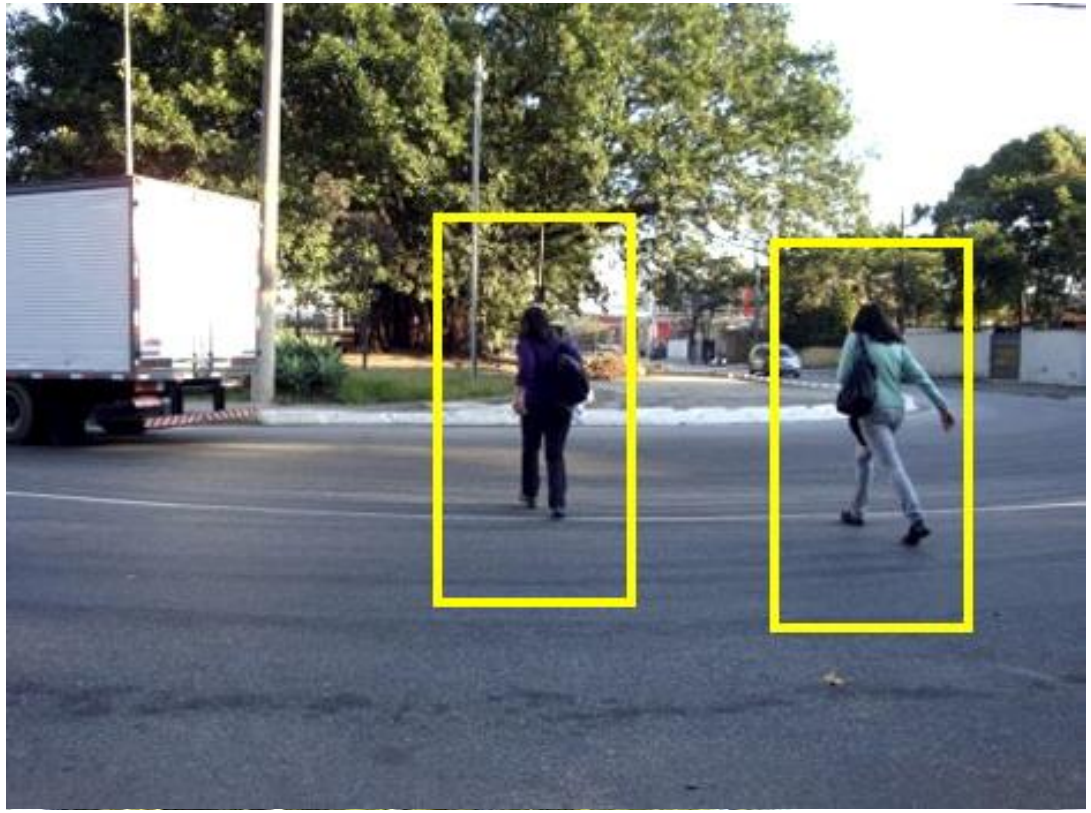
Multi-scale Detection using FST Detector



Multi-scale Detections using FST Detector

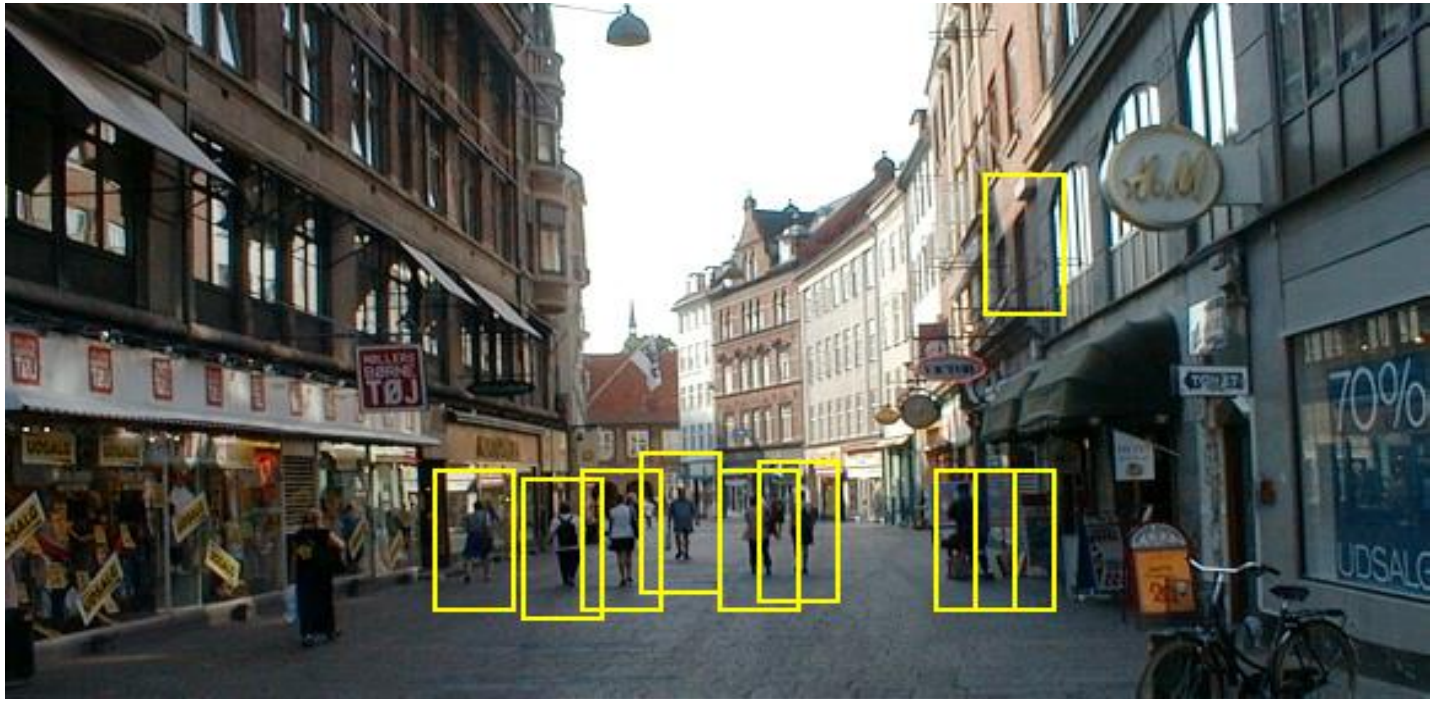


Pedestrian Detection in Low Contrast Images

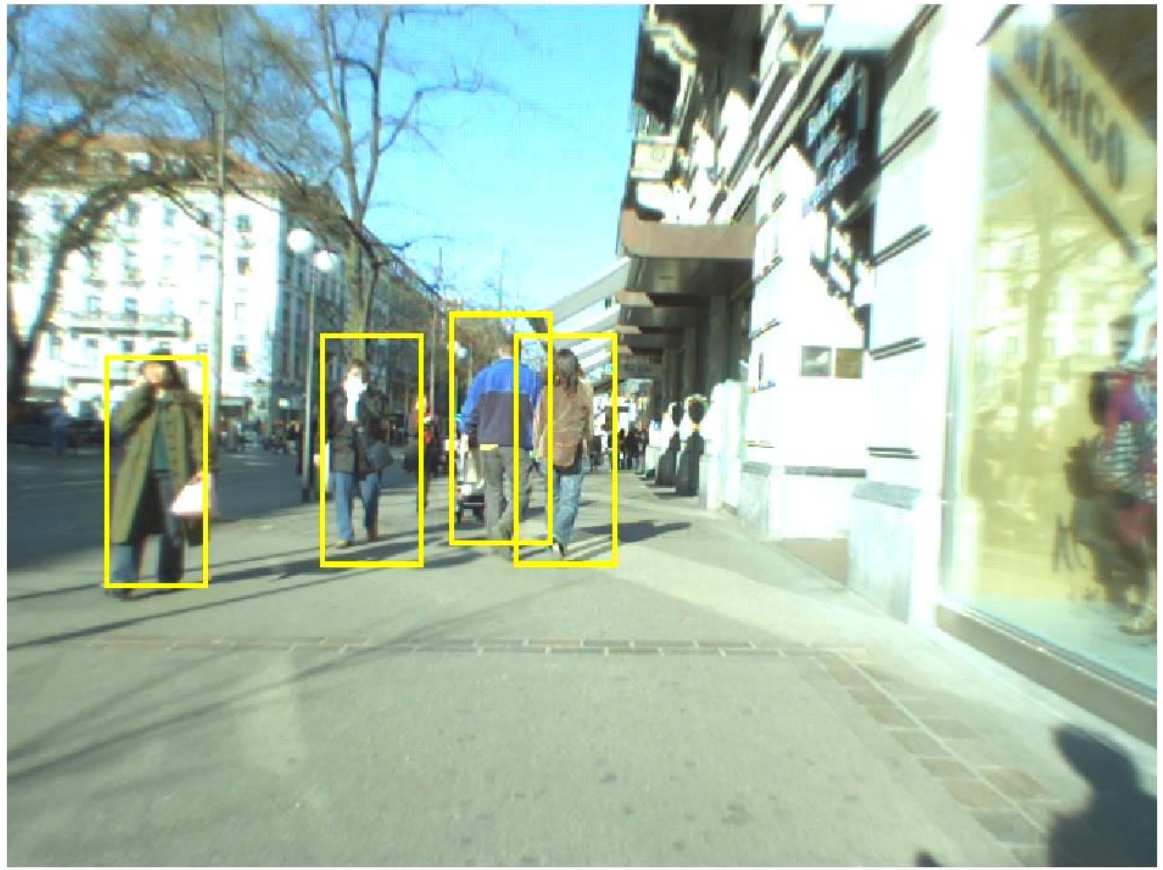


Pedestrian Detection in Low Contrast Images

Pedestrian Detection in Crowded Regions using FST Detector

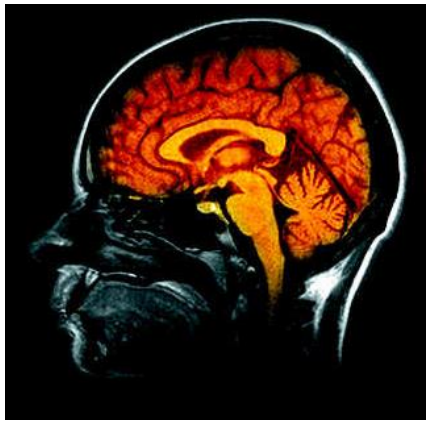


Pedestrian Detection in Multiple Frames using FST Detector

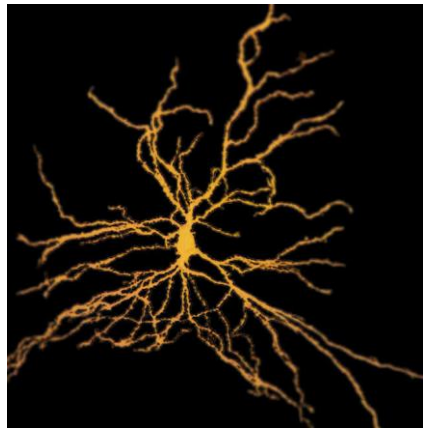


Classification: Machine Learning

Neural Network Model



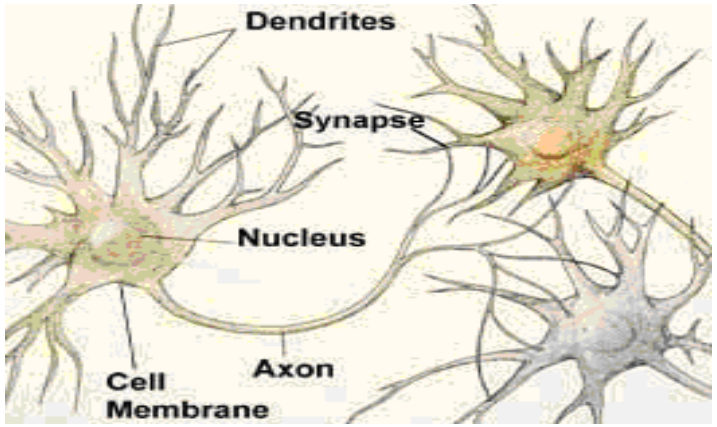
The Brain



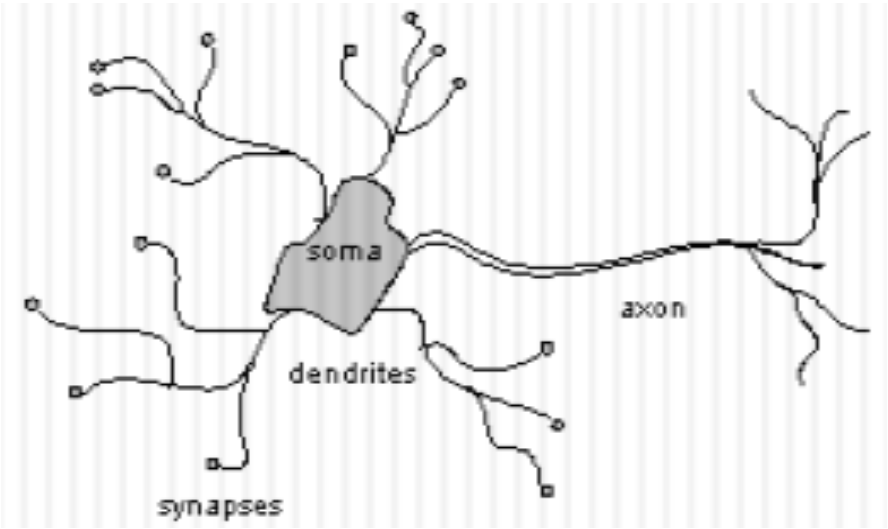
Nervous system

10^{11} *neurons*

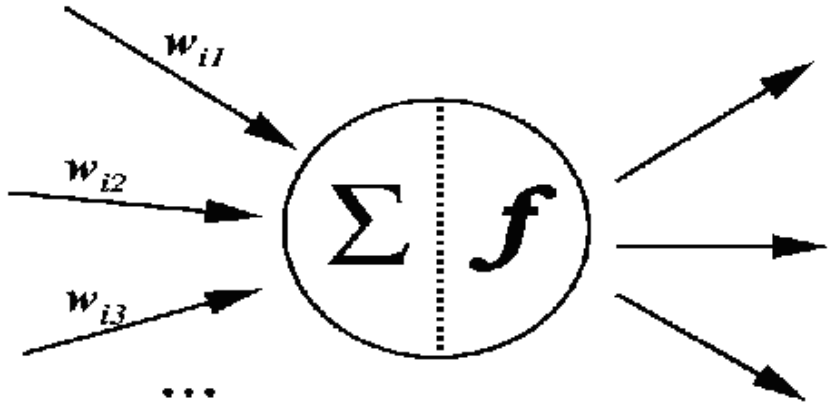
10^{15} *synaptic connections*



Neural system



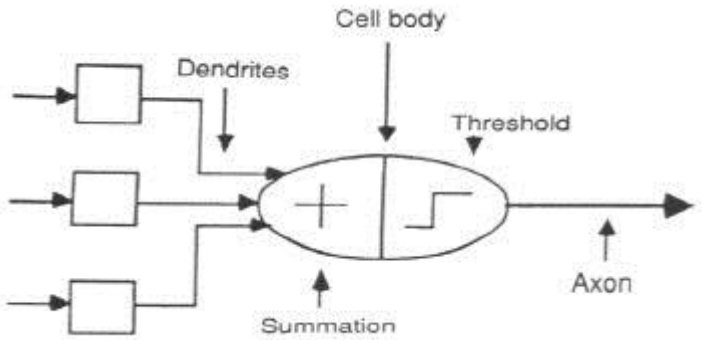
Biological neuron

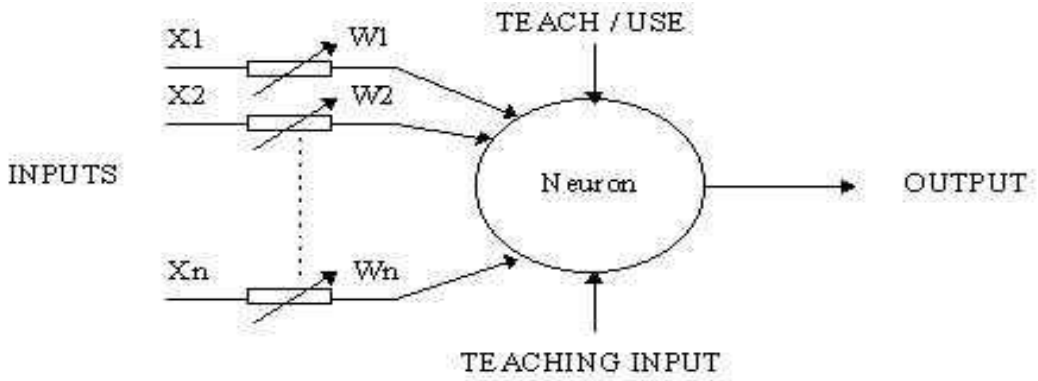


$$y_i = f(net_i)$$

Artificial neuron

A Neuron Model: Perceptron

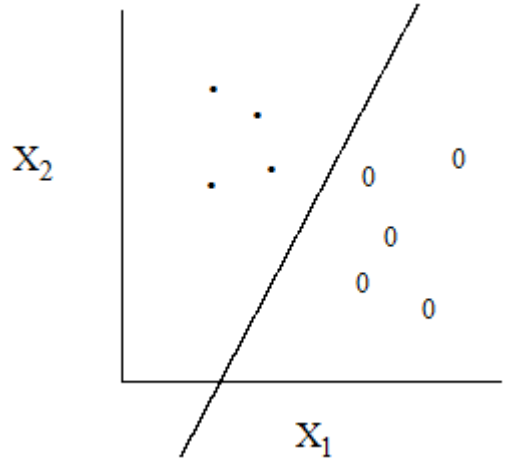




$$Net = \sum_{i=1}^{n} w_i \, x_i$$

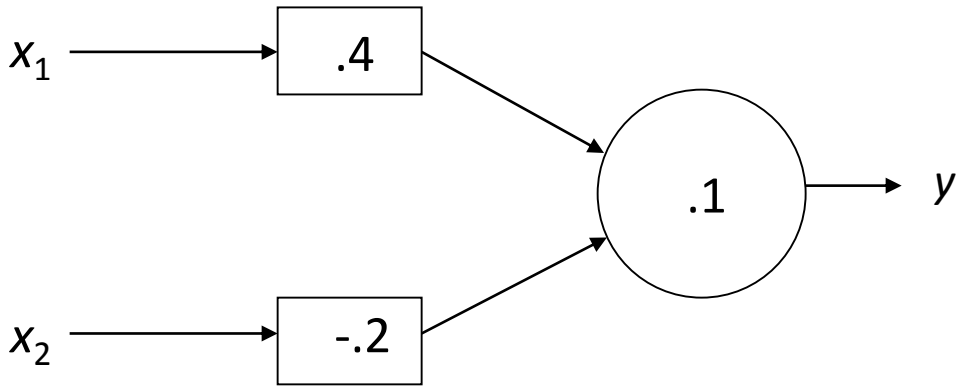
$$y = f\left(Net\right)$$

$$y = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & \text{if} \quad \sum_{i=1}^{n} w_i x_i < \theta \end{cases}$$



Perceptron Learning

x_1	x_2	t
.8	.3	1
.4	.1	0



<u>Delta Rule</u>

$w_i = w_i + \Delta w_i$

$\Delta w_i = \eta\ (t - y)\ x_i$

t is the target value

y is the perceptron output

η Is a small constant (e.g. 0.1) called *learning rate*

net = .8*.4 + .3*-.2 = .26

y = 1

net = .4*.4 + .1*-.2 = .14

y = 1

$\Delta w_i = \eta\ (t - y)\ x_i$

$\Delta w_1 = 0.25\ (0 - 1)\ 0.4 = -0.1$

$\Delta w_2 = 0.25\ (0 - 1)\ 0.1 = -0.025$

$w_1 = 0.4 - 0.1 = 0.3$

$w_2 = -0.2 - 0.025 = -0.225$

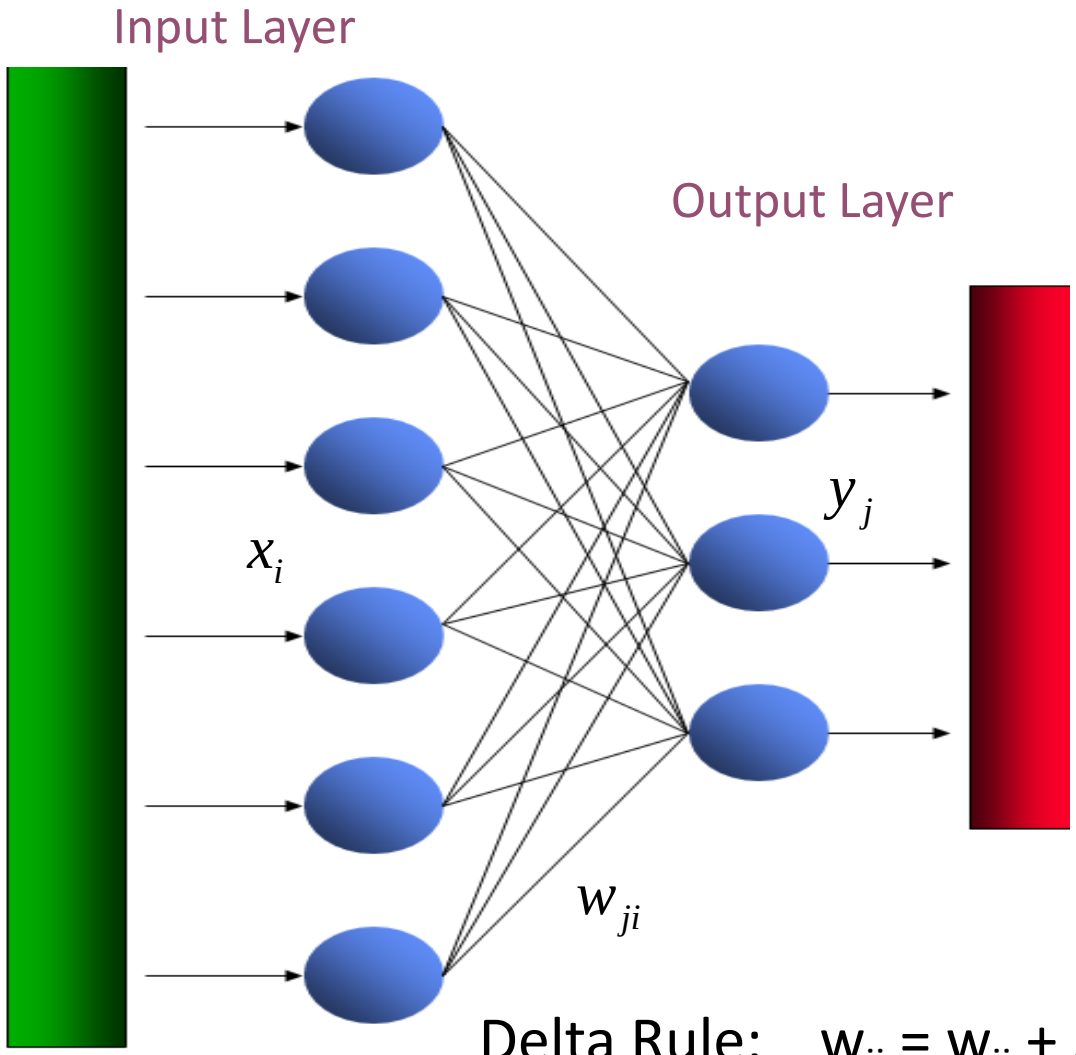
net = .8*.3+ .3*-.225 = 0.1725

y = 1

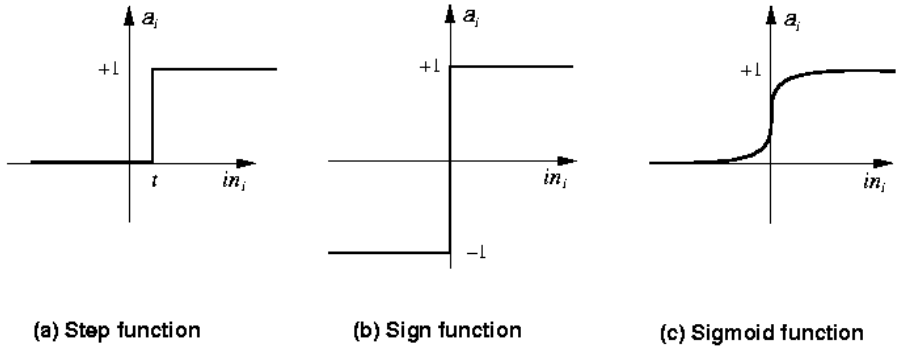
net = .4*.3 + .1*-.225 = 0.0975

y = 0

Single Layer Perceptron (SLP)



$$y_j = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_{ji} x_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^{n} w_{ji} x_i < \theta \end{cases}$$

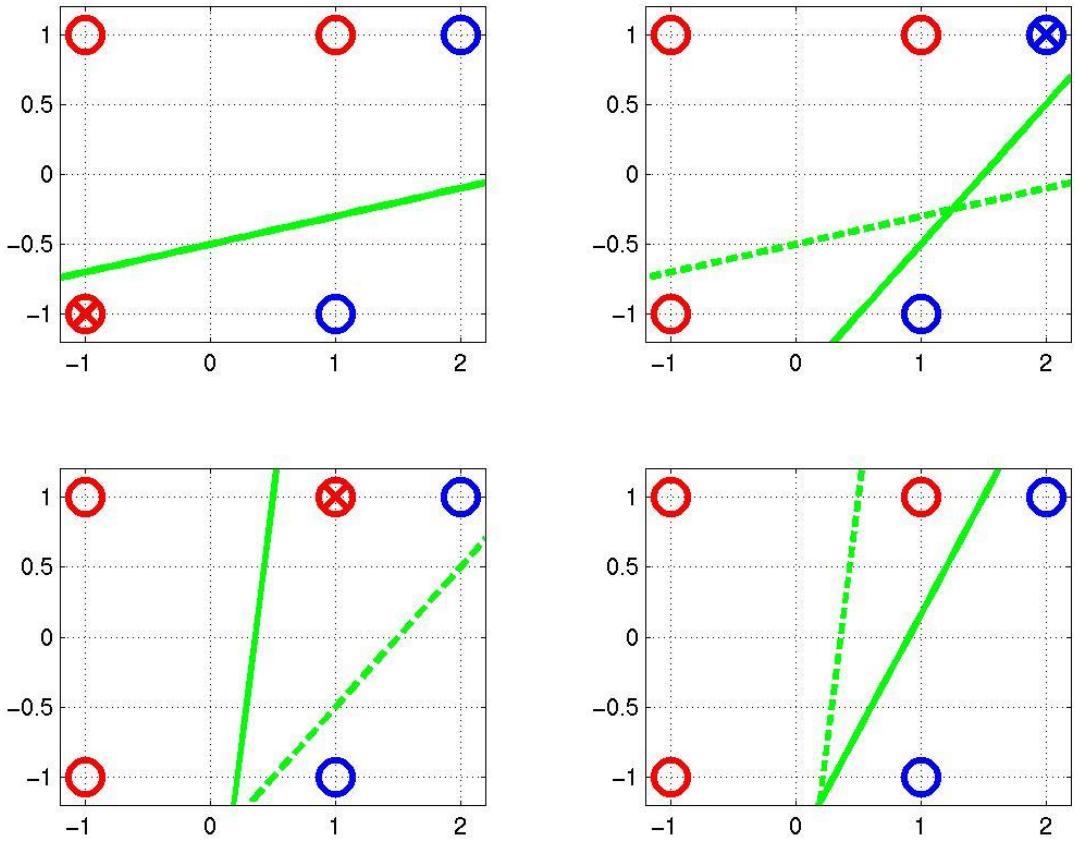


(a) Step function (b) Sign function (c) Sigmoid function

$$y_j = \frac{1}{1 + e^{-a\,Net_j}}$$

Delta Rule: $w_{ji} = w_{ji} + \Delta w_{ji}$ $\Delta w_{ji} = \eta\,(t_j - y_j)\,x_i$

Perceptron Convergence



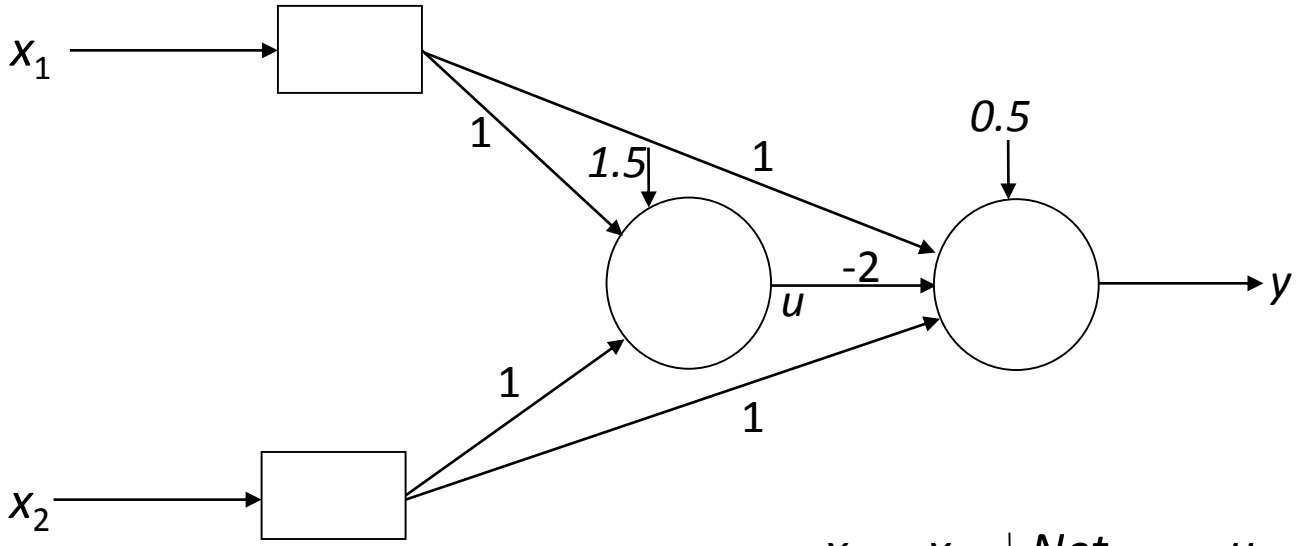
The algorithm converges to the correct classification if

- the training data is linearly separable
- and η is sufficiently small

- If two classes of vectors $\mathbf{X}_1$ and $\mathbf{X}_2$ are linearly separable, the application of the perceptron training algorithm will eventually result in a weight vector $\mathbf{w_0}$, such that $\mathbf{w_0}$ defines a decision hyper-plane separates $\mathbf{X}_1$ and $\mathbf{X}_2$

- Solution $\mathbf{w_0}$ is not unique
- $\mathbf{w_0}\,\mathbf{X} = 0$ defines a hyper-plane

Perceptron Convergence



$$y_j = \sum_i w_{ji}\, x_i + w_0$$

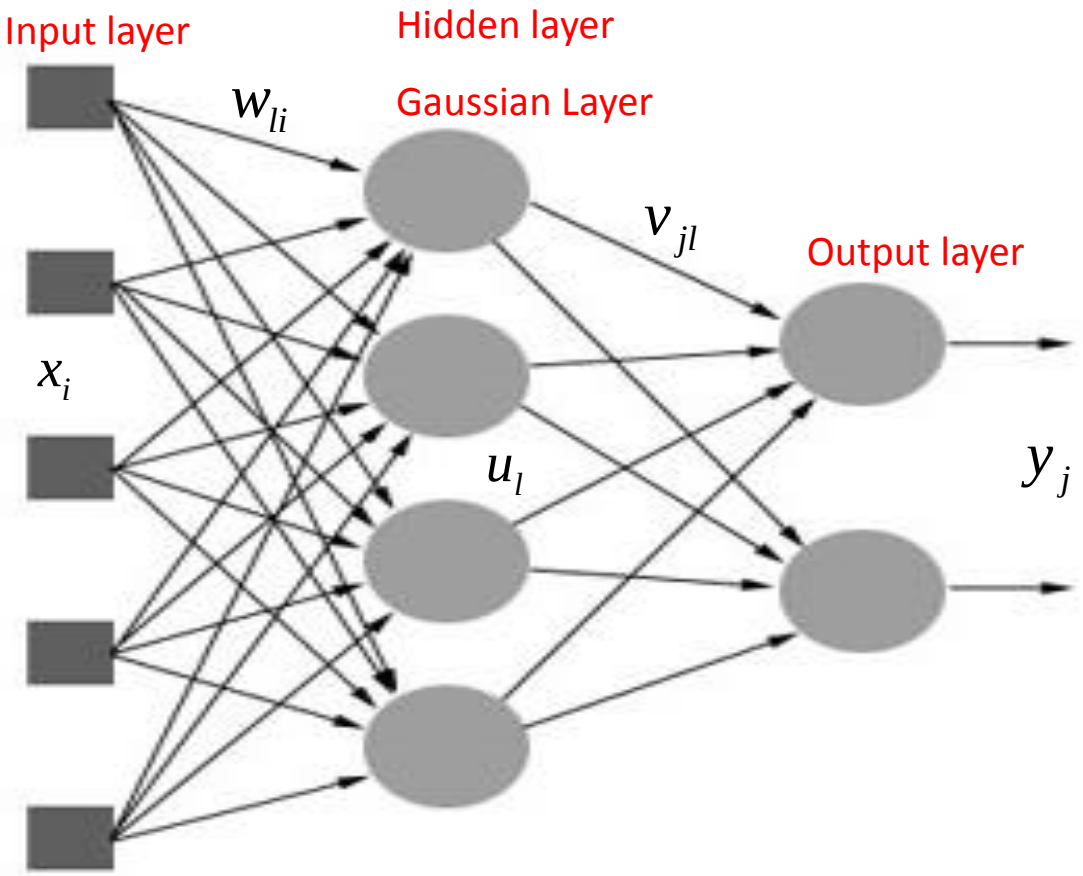
$$u = x_1 + x_2 - 1.5$$

$$y = x_1 + x_2 - 2\,u - 0.5$$

x_1	x_2	*Net*	u	**AND**
0	0	-1.5	0	
0	1	-0.5	0	
1	0	-0.5	0	
1	1	0.5	1	

x_1	x_2	u	*Net*	y	**X-OR**
0	0	0	-0.5	0	
0	1	0	0.5	1	
1	0	0	0.5	1	
1	1	1	-0.5	0	

Multilayer Perceptron (MLP)



Feed-forward neural network architecture

$$Net_l = \sum_i w_{li}\, x_i \qquad u_l = f\left(Net_l\right)$$

$$Net_j = \sum_l v_{jl}\, u_l \qquad y_j = f\left(Net_j\right)$$

$$v_{jl}^{new} = v_{jl} + \Delta v_{jl}$$

$$\Delta v_{jl} = \eta\,(t_j - y_j)\,u_l = \eta\,\partial_j\,u_l$$

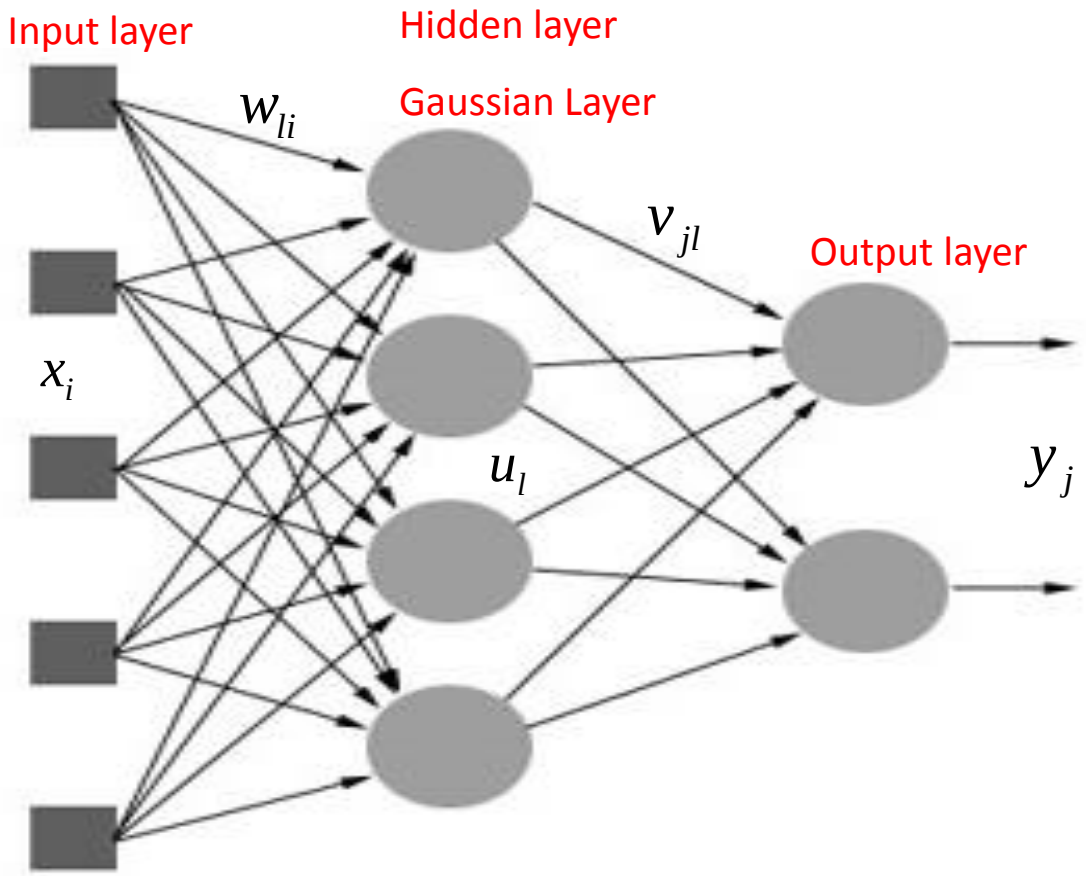
$$w_{li}^{new} = w_{li} + \Delta w_{li}$$

$$\Delta w_{li} = \eta\,\partial_l\,x_i$$

Error backpropagation

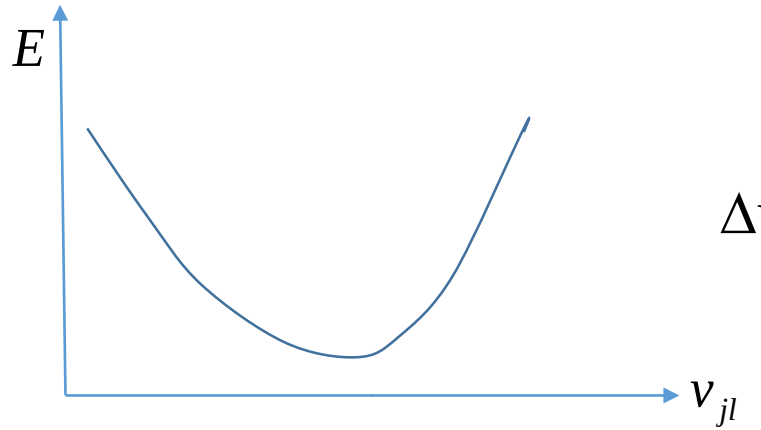
$$\partial_l = \sum_j v_{jl}\,\partial_j$$

Multilayer Perceptron (MLP)



Feed-forward neural network architecture

Gradient Descent Algorithm



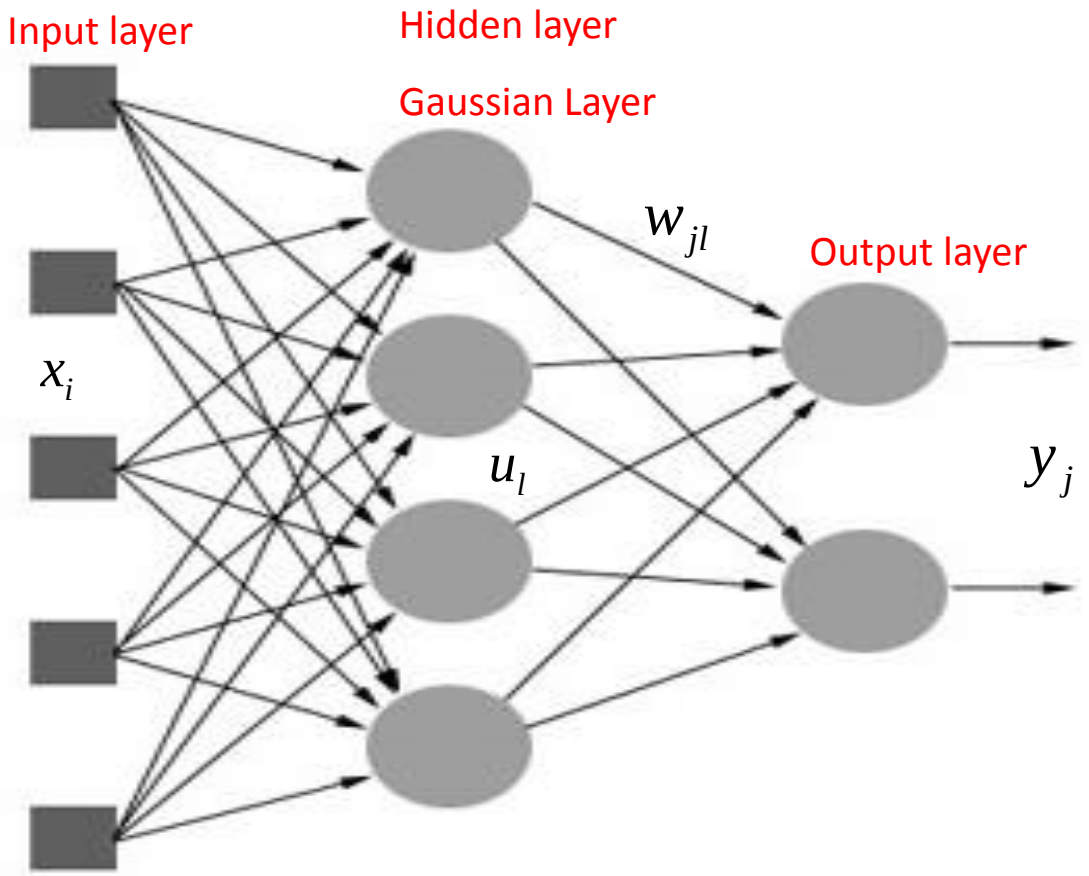
$$\Delta v_{jl} \propto -\frac{\partial E}{\partial v_{jl}}$$

$$E = \frac{1}{M}\sum_{j=1}^{M}\left(t_j - y_j\right)^2 = \frac{1}{M}\sum_{j=1}^{M}\left(t_j - f\left(\sum_{l=1}^{L} v_{jl}\, u_l\right)\right)^2$$

$$\frac{\partial E}{\partial v_{jl}} = \frac{\partial E}{\partial Net_j}\frac{\partial Net_j}{\partial v_{jl}} = -\frac{2}{M}\left(t_j - y_j\right) u_l = -\frac{2}{M}\,\partial_j\, u_l$$

$$\Delta v_{jl} = \eta\,\partial_j\, u_l$$

Radial Basis Function (RBF) Neural Networks



Feed-forward neural network architecture

$$u_l = f_l(x) = e^{\left[\frac{-\|x-\mu_l\|^2}{2\sigma^2}\right]}$$

$$Net_j = \sum_l w_{jl}\, u_l$$

$$y_j = f\left(Net_j\right)$$

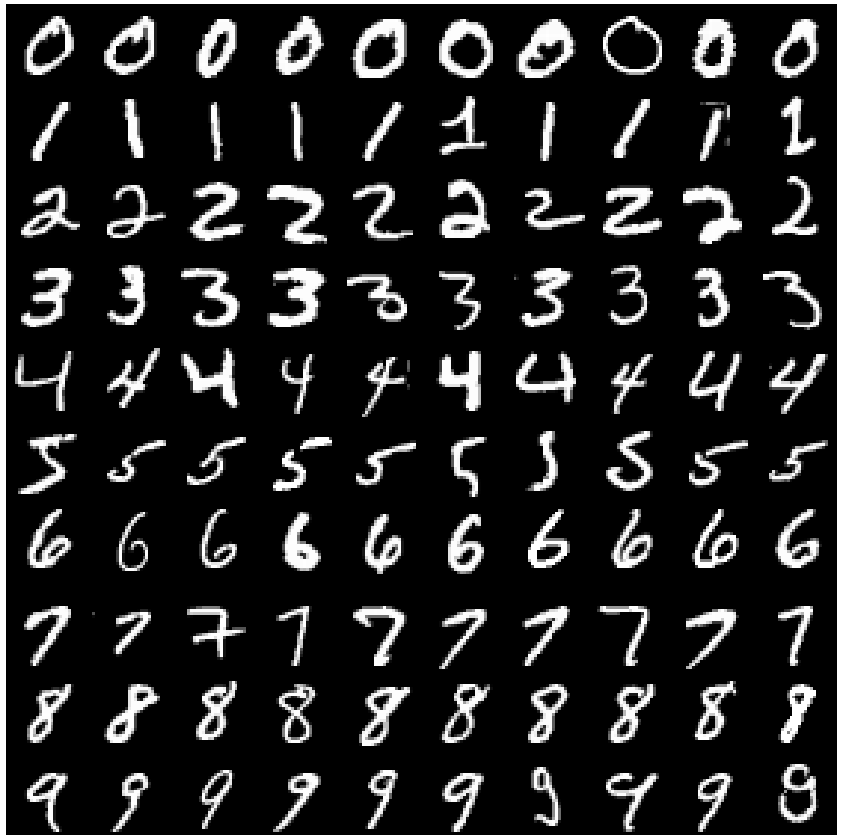
$$w_{jl}^{new} = w_{jl} + \Delta w_{jl}$$

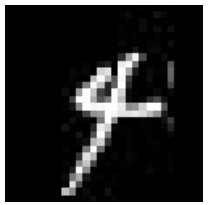
$$\Delta w_{jl} = \eta\,(t_j - y_j)\,u_l = \eta\,\partial_j\,u_l$$

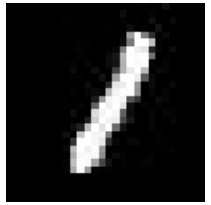
Experimental Evaluation: Training Images

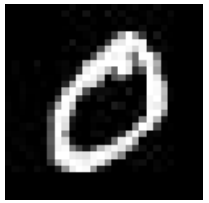
MNIST database of handwritten digits
60,000 training images, 10,000 testing images

Input image size: 28 × 28
10 classes









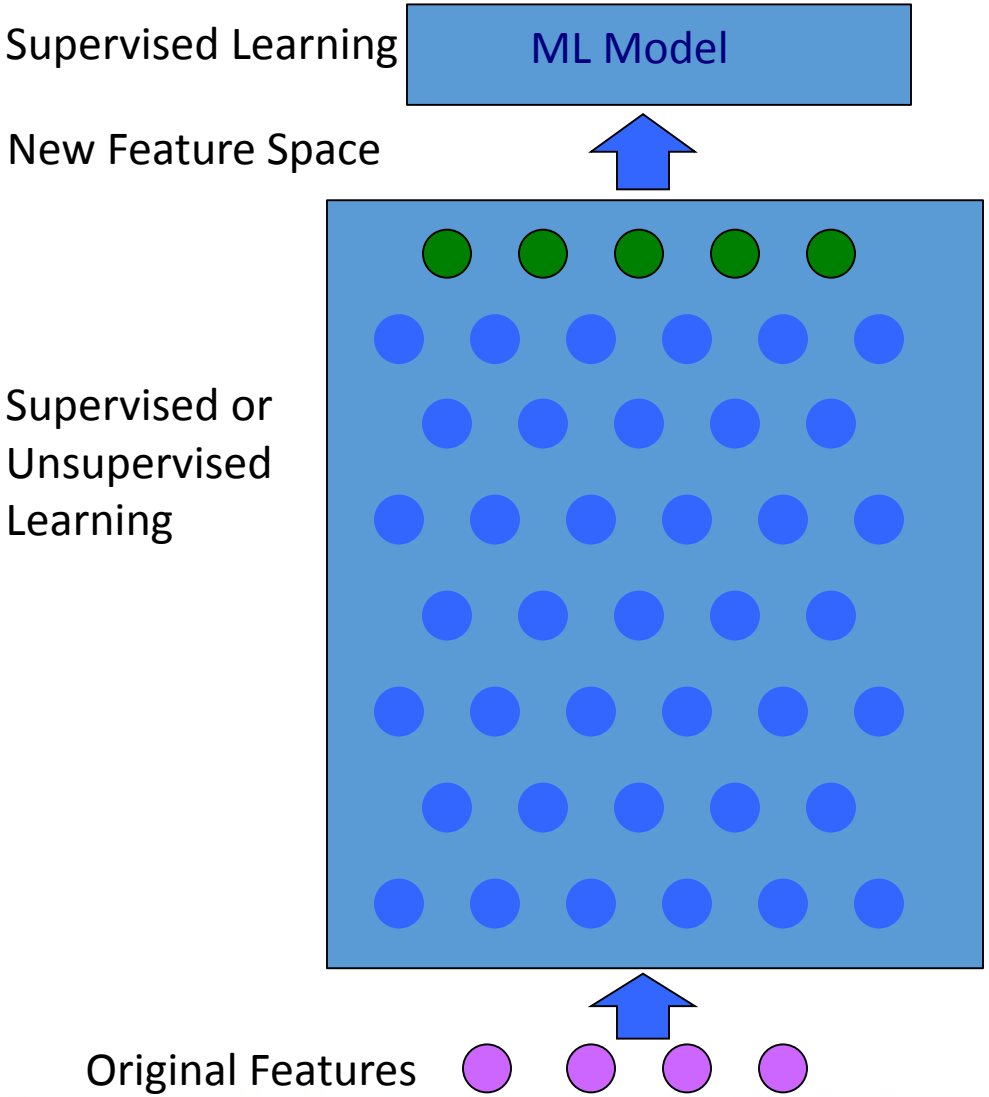
Deep Learning

- Current Trend
- Deep Backpropagation
- CNN – Convolutional Neural Networks
- Deep CNN
- Unsupervised Preprocessing Networks
  - Stacked Auto Encoders
  - Deep Belief Networks
- Deep Supervised Networks with managed gradient approaches
- Deep Reinforcement Learning

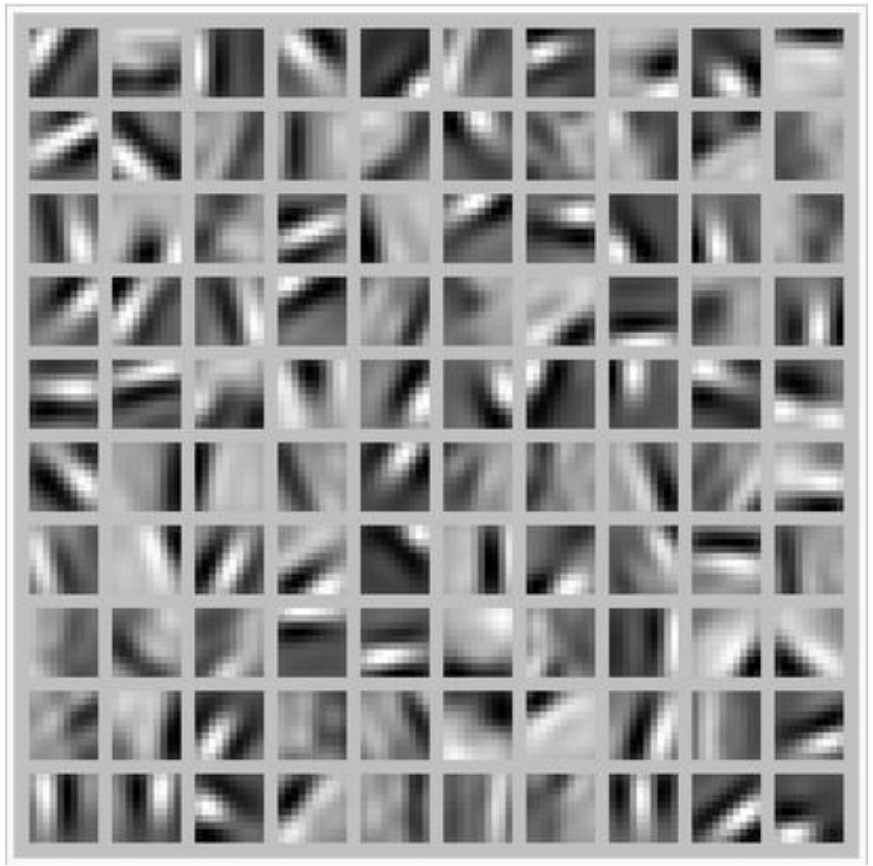
Deep Learning

- Train networks with several layers
- Multiple layers work to build an improved feature space
  - First layer learns 1st order features (*e.g. edges…*)
  - 2nd layer onwards learn higher order features (*combinations of first layer features, combinations of edges, etc.*)
  - Some models learn in an unsupervised mode and discover general features of the input space – serving multiple tasks related to the unsupervised instances (*image recognition, etc.*)
  - Final layer of transformed features are fed into a supervised layer or layers
    - The entire network is often subsequently tuned using supervised training, using the initial weights learned in the unsupervised phase

Deep Learning



- View of a learned feature layer
- Each square in the figure shows the input image that maximally activates one of the 100 units



Deep Learning

- Biological Plausibility – e.g. Visual Cortex
- Problems can be represented with a large number of nodes with *k* layers
- Highly varying functions can be efficiently represented with deep architectures
  - Less weights/parameters to update than a less efficient shallow representation
- Sub-features created in deep architecture can potentially be shared between multiple tasks
  - Type of Transfer/Multi-task learning

Convolutional Neural Network

- Niche networks built specifically for problems with low dimensional (e.g. 2-*d*) grid-like local structure
  - Character recognition – where neighboring pixels will have high correlations and local features (edges, corners, etc.), while distant pixels (features) are un-correlated
  - CNNs enforce that a node receives only a small set of features which are spatially or temporally close to each other called *receptive fields* from one layer to the next (e.g. 3x3, 5x5), thus enforcing ability to handle local 2-D structure.
    - Can find edges, corners, endpoints, etc.

Convolutional Neural Network

- Nodes do a scalar dot product (*convolution*) from the previous layer, with only a small portion (*receptive field*) of the nodes in the previous layer – *Sparse representation.*
- Every node has the exact same weight values from the preceding layer – *Shared parameters*, tied weights, a less number of unique weight values.
- Regularization by having same weights looking at more input situations.
- Each node has it's shared weight convolution computed on a receptive field slightly shifted, from that of it's neighbor, in the previous layer – *Translation invariance*.
- Each node's convolution scalar is then passed through a non-linear activation function.

Convolutional Neural Network

First convolutional layer: $5 \times 5 = 25$; output size: 24×24 (*no zero padding*)

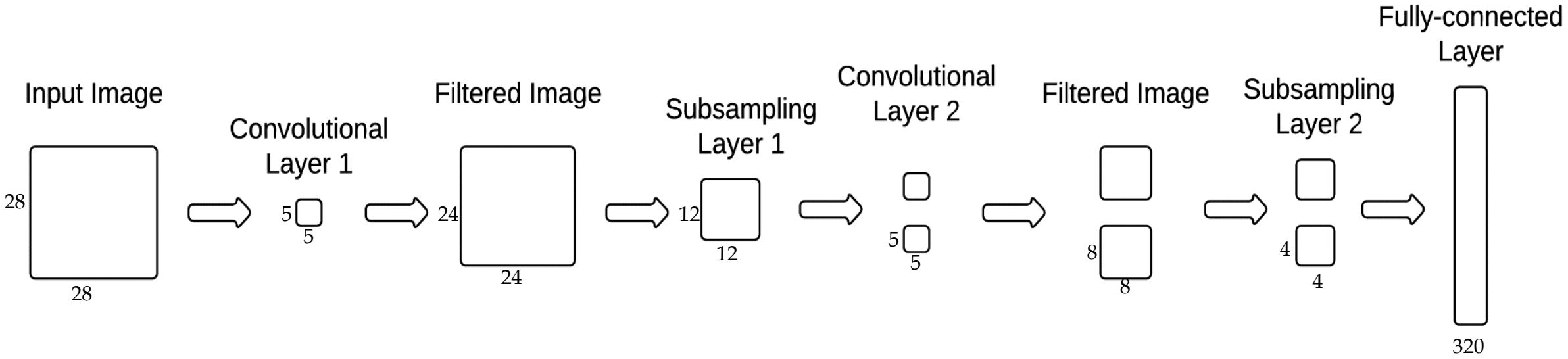
First subsampling/pooling layer: down-sampled to 12×12

Second convolutional Layer: $5 \times 5 \times 2 = 50$; output size: 8×8

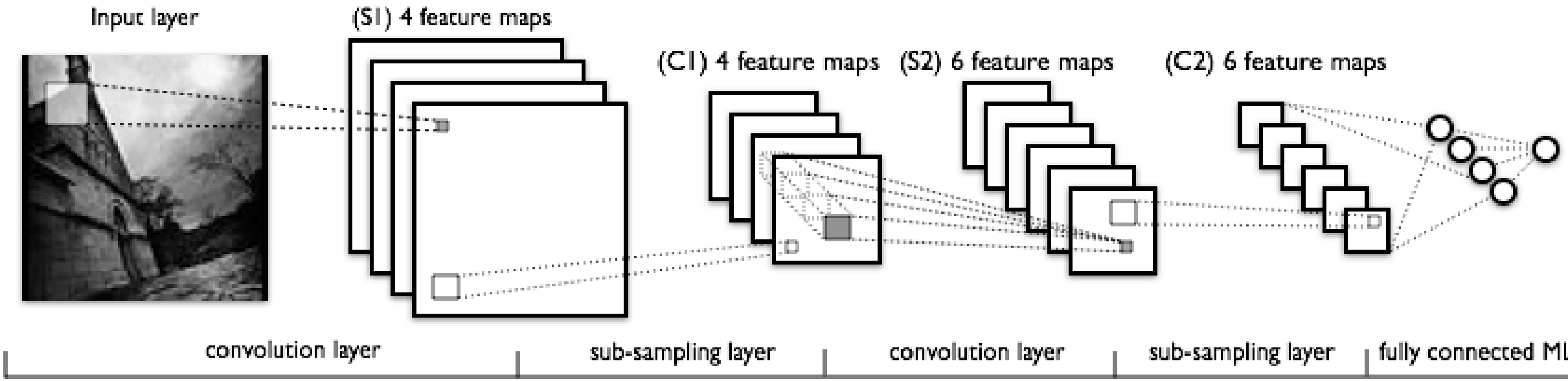
Second subsampling/pooling layer: down-sampled to $4 \times 4 \times 2$

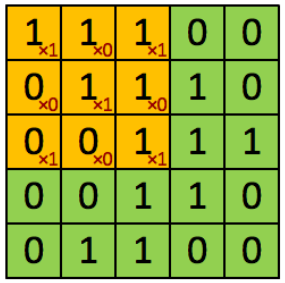
Fully-connected layer: $4 \times 4 \times 2 \times 10 = 320$

Total number of weights: $25 + 50 + 320 = 395$

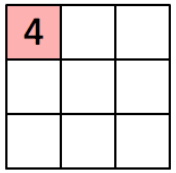


Convolutional Neural Network

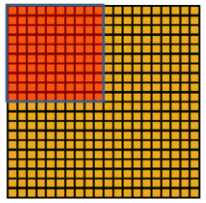




Image

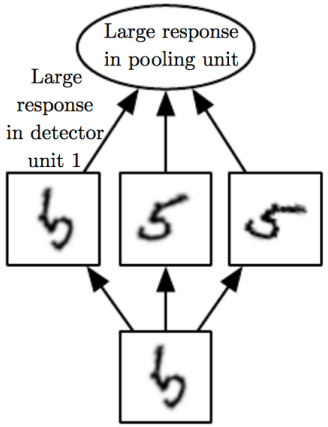


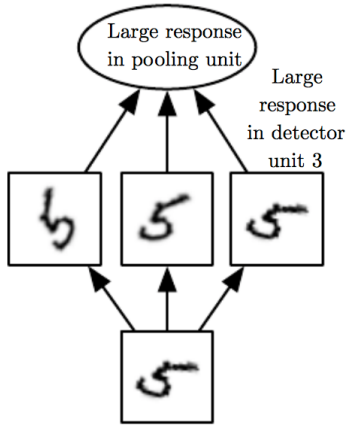
Convolved Feature

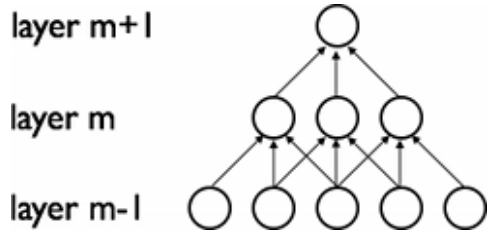


Convolved feature

Pooled feature







Scene Classification

Semantic context varies in the same scene category

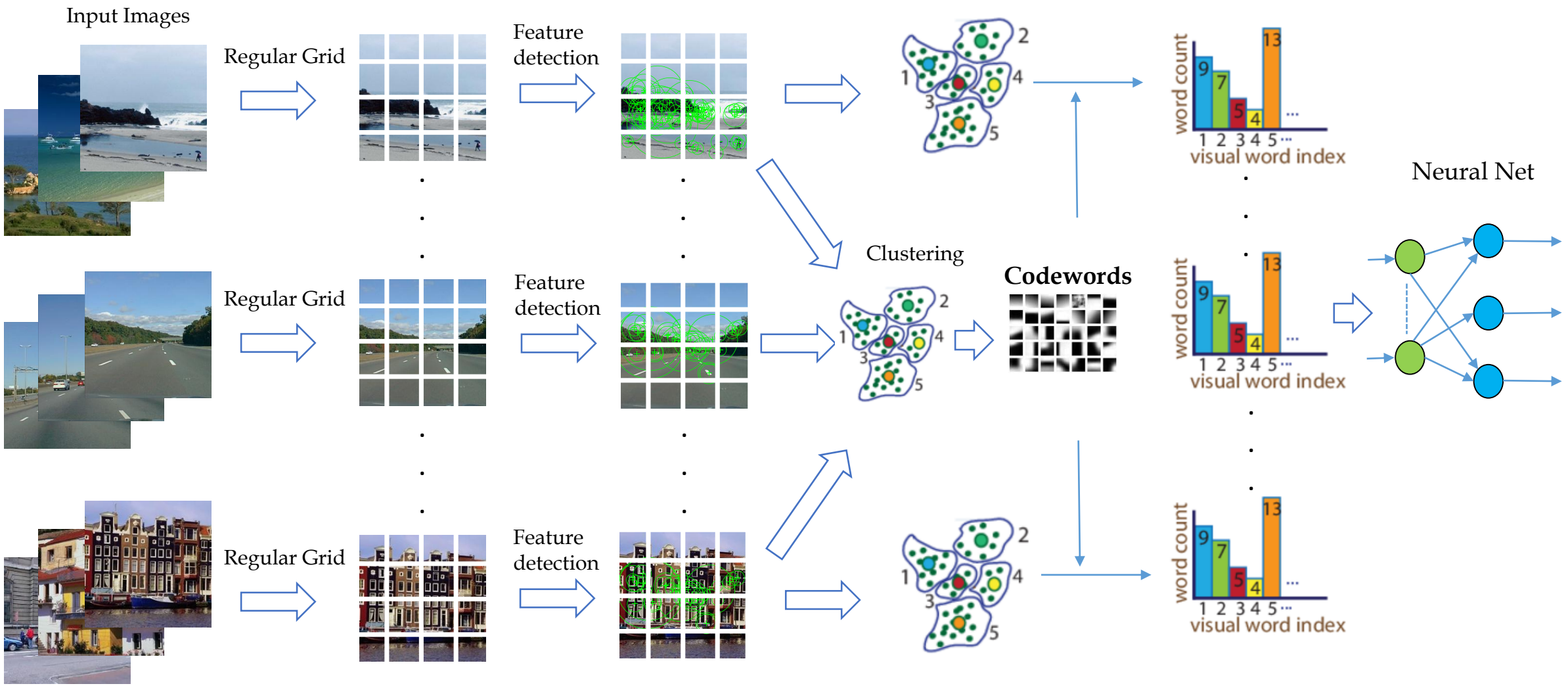
Coasts | Rivers/Lakes | Forests | Plains | Mountains | Sky/Clouds

Scene Classification

Similar contexts shared between different scene categories

Coasts | Rivers/Lakes | Forests | Plains | Mountains | Sky/Clouds

Training Stage



Testing: Dataset-I (8-scene category)

- Includes 8 outdoor scenes
- Total images : 2688 color images, each category ranging from 260 to 410


Coast


Forest


Highway


Inside city


Mountain


Open country


Street


Tall building

Average accuracy for 20 trials: 80.56%

Testing: Dataset-II (13-scene category)

- Includes 13 outdoor scenes (8-scene + 5)
  - Five additional scenes: offices, kitchen, bedroom, suburb, and living rooms.
- Total images : 3859 gray-scale


Office


Kitchen


Bedroom


Suburb


Living room

Average accuracy for 20 trials: 71.36%

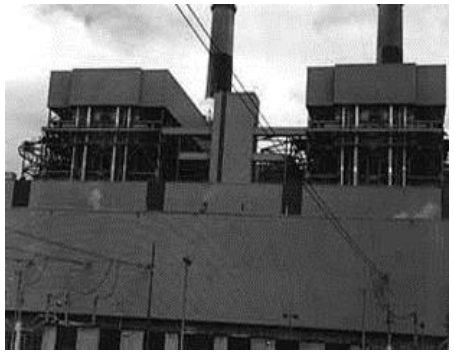
Testing: Dataset-III (15-scene category)

- Includes 15 outdoor scenes (13-scene + 2):
  - Two additional scenes: industrial and store.
- Total images : 4485 gray-scale

Average accuracy for 20 trials: 65.14%

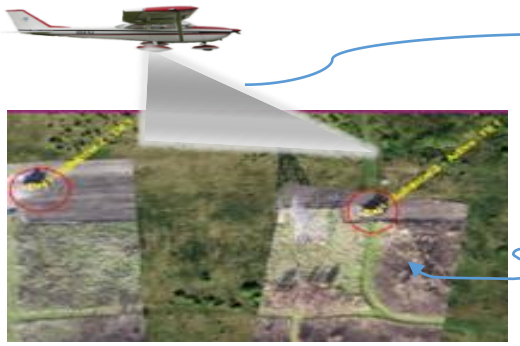



Store




Industrial

Automated Aerial Monitoring for Intrusion Detection



Aerial Imagery

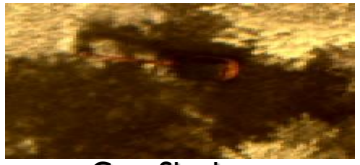
Image Analysis

Automated System

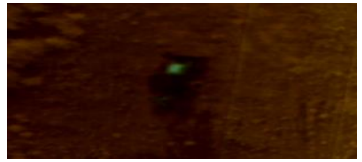
Object Classification

Threat Localization

KML File Mapping



Cast Shadows



Low illumination

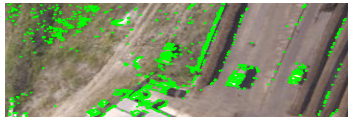
Orientation

Partial Occlusion

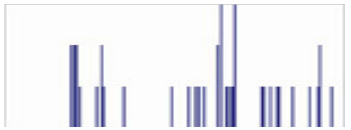
Small Scale

Color Transformation

Key Frame Selection



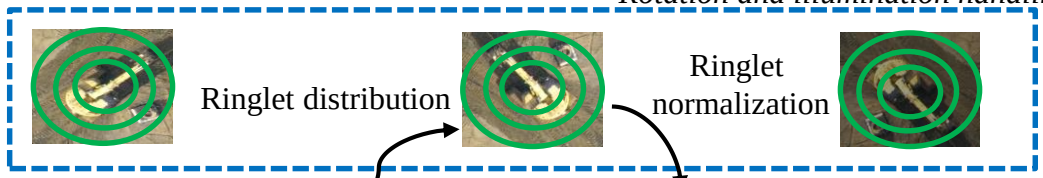
Key Region Selection



Local Feature Extraction

Object Detection

Rotation and illumination handling



Scaling and occlusion handling

Thanks

Sensing, Processing and Automatic Decision Making in Real Time

www.visionlab.udayton.edu