

# Image Enhancement

## Intensity Transformation and Spatial Filtering

February 12, 2019

# Enhancement

Objective: To improve the interpretability of the information in the image. NOTE no new information can be created out of the existing image. Whatever is done is only an extrapolation useful for visual interpretation.

Two broad categories:

1. Spatial domain method- two principal categories
  - ▶ Intensity transformations - operate on single pixels of an image principally for the purpose of contrast manipulation and image thresholding. Elementary enhancement methods are based on the histogram.
  - ▶ Spatial filtering - deals with performing operations in a neighbourhood of every pixel in an image
2. Frequency domain method - we operate on the Fourier Transform of the image and then transform it back to the spatial domain.

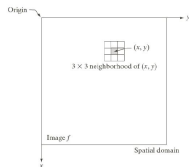
# Spatial Filtering

- ▶ Spatial domain refers to the Image Plane- two categories
  - ▶ Intensity Transformation- operates on single pixels of an image principally for contrast manipulation and image thresholding
  - ▶ Spatial Filtering - performs operations in a neighbourhood of every pixel in an image
- ▶ Contrast this to processing in the transform domain and reverting back to the spatial domain

Some image processing techniques are easier and more meaningful in spatial domain while others are suited for frequency domain processing. Generally spatial domain techniques are more efficient.

# Intensity Transformation

The figure illustrates the basic concept of spatial filtering. Take a window and apply an operator  $T : g(x, y) = T[f(x, y)]$  in the neighbourhood of the centre pixel. The value of the output of this transformation is assigned to the centre pixel. The window is now slid from left to right and then from top to bottom, so that the whole image is covered



# Intensity Transformation

- ▶ The point  $(x, y)$  is shown as an arbitrary location in the image, and the small region containing the point is a neighbourhood of  $(x, y)$ . Typically the neighbourhood is a rectangle (more often a square), centred on  $(x, y)$ , much smaller in size than the image.
- ▶ The smallest possible neighbourhood is of size  $1 \times 1$ . In this case, the mapping becomes an intensity transformation of the form  $s = T(r)$ , where  $r$  and  $s$  denote the gray levels in the input and output image respectively.

# Point Operation

In point operation, each pixel is modified by an equation that is not dependent on other pixel values. The point operation is given by the following equation:

$g(m, n) = T(f(m, n))$ , where  $m, n$  are pixel and scanlines covering the entire image. Since this is a point operation, the same gray count present anywhere will be mapped to the same output gray count as given in the previous slide.

The mapping  $T$  should be a one-one mapping if you want it to be invertible. Otherwise, it should be an into mapping from the set of input gray values to the set of output gray values. The mapping can apply to a single image or in some cases (for noise reduction) to a set of images.

# Types of Point operation

## 1. Brightness modification:

There are two types possible - a) Increasing/decreasing the brightness of an image, and b) Contrast enhancement.

(a) In this case, the image is originally of very low/high brightness. The overall brightness is adjusted by adding/subtracting a constant gray value from all the pixels.

$$g(m, n) = f(m, n) \pm k$$

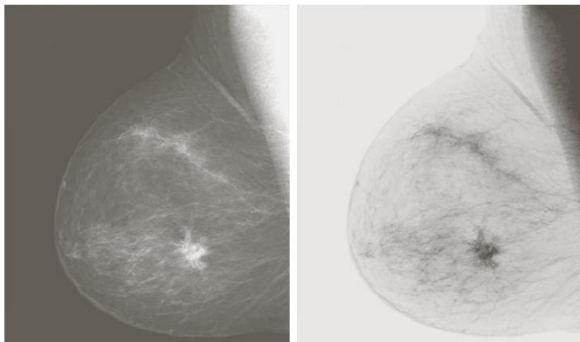
Care should be taken that the resultant gray value does not go beyond the possible range - it should not exceed 255 and not be negative (for eight bit image)

(b) In this case the gray values of the pixels are modified by scaling. Sometimes, you can also use the equation to a straight line and use piecewise linear functions. By doing this you get better control over the variation in gray counts over the different portions of the image.

$$g(m, n) = a + f(m, n) * k$$

# Some typical Point operations and their effects

- Negative image - In this case, the gray values are mapped to their complementary values. Particularly used in Medical Image Processing

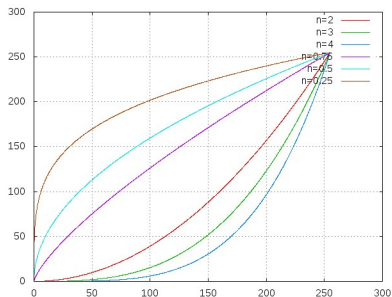


Mammogram image and its negative



# Typical Power Law curves, $y = cx^n$

Observe  
that different values of  
the exponent gives different  
stretching at different  
locatons. This can also be  
observed from the derivative  
of the power law function



# PowerLaw Effect on image



Original



$\gamma = 1.1$



$\gamma = 1.2$



$\gamma = 0.9$

# Power Law Enhancement



Original,  $\gamma = 1$ ,  $\gamma = 4$ ,  $\gamma = 5$

# Histogram Manipulation

For a good contrast image, the histogram should be well spread out covering as much of the total range as possible. If however for a given image, the histogram is too narrow and occupies a low gray level portion, then clearly the image will be 'dark' with poor contrast. In such situations, we can define a mapping  $\text{New}_{\text{Gray}} = f(\text{Old}_{\text{Gray}})$ , any way we like so that the  $\text{New}_{\text{Gray}}$  histogram is more towards a good histogram. The function we choose should be one-one. Generally linear fit is used for this kind of mapping.

$g_{\min} \rightarrow 0, g_{\max} \rightarrow 255$  gives a reasonable result in most of the cases.

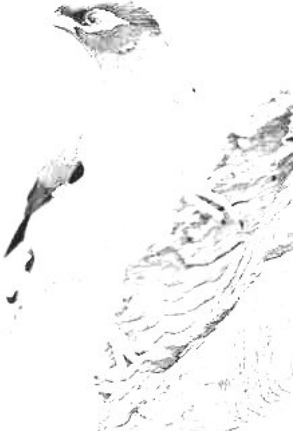

Sometimes depending upon applications, we can also define piecewise linear functions, so that the enhancement is done selectively over specified range.

**Assignment:** Take any image. Find its histogram. Modify the histogram so that it occupies the full dynamic range.

Transform the original image to correspond to this histogram.

Play around with the minimum and maximum of the output

# PiecewiseLinearStretchingExamples

	
$(30, 80) \rightarrow (20, 230)$	$(30, 200) \rightarrow (5, 240)$

# Image Negative

In this all the gray values are replaced by their complementary value - with respect to the maximum gray level possible for that image.



# Non-linear gray level transformations

In this case, since the mapping is non-linear, equal ranges of gray values in the input image are mapped to unequal intervals of gray values in the output image.

Some of the important non-linear transformations are:

1. Thresholding
2. Gray level slicing
3. Logarithmic transformation
4. Exponential transformation
5. Power Law transformation

# Thresholding

This method is used when we want to extract some portion of an image - the portion being defined in terms of Gray levels. In fact, thresholding is part of a more general Segmentation Problem. Thresholding can be of two types:

1. Hard thresholding - in this case, we obtain a binary image from a gray scale image. The transformation which is used to obtain this is
$$g(m, n) = 0, \text{ for } f(m, n) < T; g(m, n) = 255, \text{ otherwise}$$
2. In case we want to preserve the background while extracting information of our interest, we should do soft thresholding



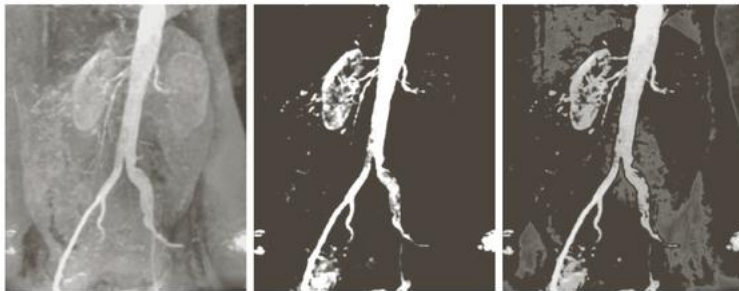
# Soft and Hard Thresholding



← Background lost

# Soft and Hard Thresholding

Also called Intensity slicing



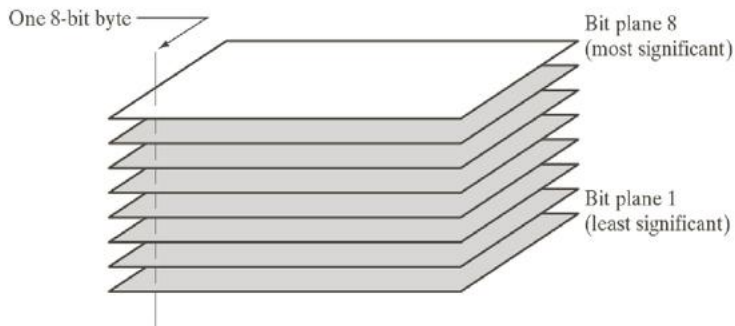
Aortic angiogram; Hard thresholding; Soft thresholding

# Logarithmic transformation



# BitPlane Slicing

Gray levels in images are digital numbers (8 bits etc). We might as well manipulate the individual bits to change the appearance of the image.



Schematics diagram


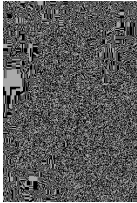
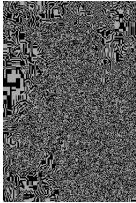
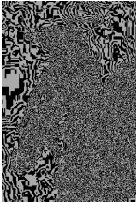

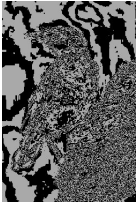
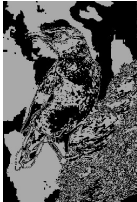

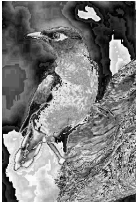
# BitPlane Slicing



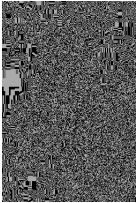
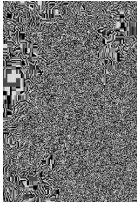
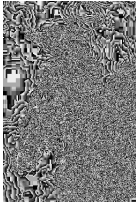
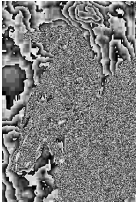
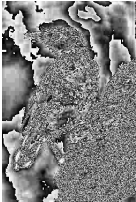
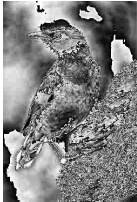
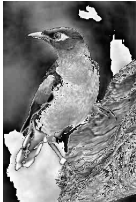


Top: An original 8 bit image and bit planes 1 through 8.

Bottom: Images formed by bit planes 8,7; bit planes 8,7,6; bit planes 8,7,6,5

# BitPlane Slicing-All images individually equalised for display

				
Original	Bit 1	Bit 2	Bit 3	Bit 4
				
Bit 5	Bit 6	Bit 7	Bit 8	

# BitPlane Slicing-All images individually equalised for display

				
1 Bit	2 Bits	3 Bits	4 Bits	
				
5 Bits	6 Bits	7 Bits	8 Bits	Original

# Histogram Manipulation

Since a linear map needs two constants, we need two conditions to determine them. These can be in terms of the mean and standard deviation of the output histogram. Thus, we can specify that the output image should have a mean level of  $\mu$  and a standard deviation of  $\sigma$ . This way we can modify the brightness and contrast of the image.



# Histogram Equalisation

We had said that the 'best' image is the one with a histogram well spread out and all the gray counts have more or less equal probability of occurrence. We can modify our given image to have this property.

Suppose the probability density function of the image is given by  $p(x)$ .

Let us transform the image to another one wherein we represent the gray counts by  $y$ . Let the transformation be  $y = f(x)$ .

$$\text{Prob}(y \leq Y \leq y + \Delta y) = \text{Prob}(x \leq X \leq x + \Delta x).$$

But since we want  $Y$  to have uniform histogram,  $\text{LHS} = \frac{\Delta y}{L-1}$ , while the  $\text{RHS} = p(X)\Delta x$

# Histogram Equalisation

Hence in the limit as  $\Delta x \rightarrow 0$ , we obtain

$$\frac{dy}{dx} = (L - 1)p(x) \Rightarrow y = (L - 1) \int p(x) dx.$$

Thus,  $y$  is the cumulative pdf of  $x$ .

However, this exact result is valid only for continuous variables. In reality, we are dealing with discrete values, and hence the resultant histogram will only approximately be flat

# Histogram EqualisationExample

Suppose the intensity values in an image have the PDF given by  $p_r(r) = \frac{2r}{(L-1)^2}, 0 \leq r \leq L-1$   
 $= 0$ , otherwise

$$s = T(r) = (L-1) \int_0^r p_r(w) dw = \frac{2}{L-1} \int_0^r w dw = \frac{r^2}{L-1}$$

This gives the transformation from input gray values to output gray values.

# Histogram Equalisation Discrete case

For discrete values, we deal with probabilities and summations instead of probability density functions and integrals. The probability of occurrence of intensity level  $f_k$  in a digital image is approximated by

$$p_r(r_k) = \frac{n_k}{MN}, k = 0, 1, \dots, L - 1$$

The discrete form of the transformation given above is

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j)$$

$$= \frac{(L-1)}{MN} \sum_{j=0}^k n_j$$

# Histogram Equalisation-Example

Suppose we have a 3 bit image ( $L = 3$ ) of size  $64 \times 64$ . It has the intensity distribution given below:

$r_k$	$n_k$	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 6$	56	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

# Histogram Equalisation-Example

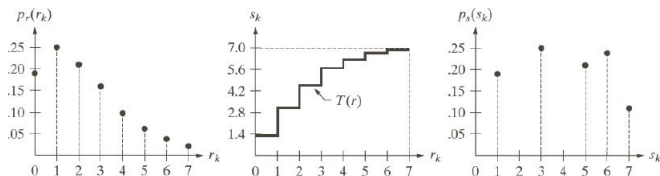
$$s_0 = T(r_0) = 7 \sum_{j=0}^0 p_r(r_j) = 7p_r(r_0) = 1.33$$

Similarly,

$$s_1 = T(r_1) = \sum_{j=0}^1 p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

Similarly

$$s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.6, s_7 = 7.0$$



Original Histogram, Transformation function and Equalized Histogram

# Histogram Equalisation-Example

Consider the image

$$\begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 4 & 5 & 4 & 3 \\ 3 & 5 & 5 & 5 & 3 \\ 3 & 4 & 5 & 4 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

Do a histogram equalisation of this image

The histogram of the input image is given below:

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0

# Histogram Equalisation Example

Form the cumulative pdf of the same

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Cumulative frequency	0	0	0	6	20	25	25	25

Divide the cumulative frequencies by Total number of pixels=25.

Multiply the result by the maximum gray count in the input image(=7).

Gray level	0	1	2	3	4	5	6	7
Number of pixels	0	0	0	6	14	5	0	0
Cumulative frequency	0	0	0	$\frac{6}{25}$	$\frac{20}{25}$	$\frac{25}{25}$	$\frac{25}{25}$	$\frac{25}{25}$
Result of multiplication	0	0	0	2	6	7	7	7



# Histogram Equalisation Example

The one-to-one mapping

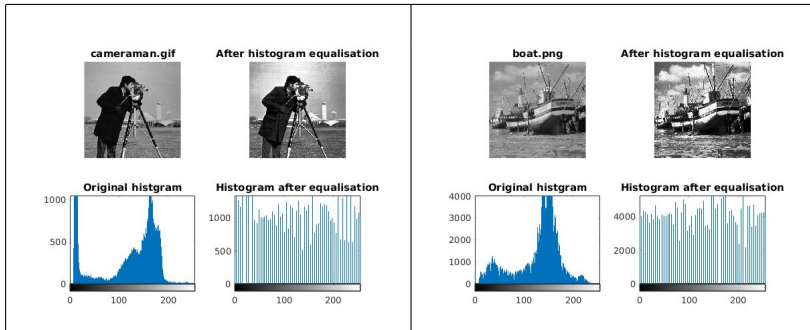
Original gray level	Histogram equalised gray level
0	0
1	0
2	0
3	2
4	6
5	7
6	7
7	7

# Histogram Equalisation Example

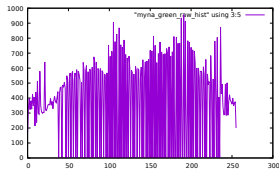
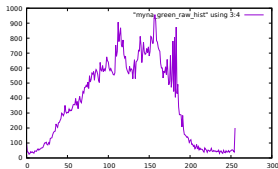
The original image and the histogram equalised image are given below:

$$\begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 4 & 5 & 4 & 3 \\ 3 & 5 & 5 & 5 & 3 \\ 3 & 4 & 5 & 4 & 3 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix} \text{ and } \begin{bmatrix} 6 & 6 & 6 & 6 & 6 \\ 2 & 6 & 7 & 6 & 2 \\ 2 & 7 & 7 & 7 & 2 \\ 2 & 6 & 7 & 6 & 2 \\ 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

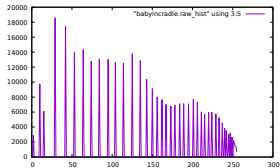
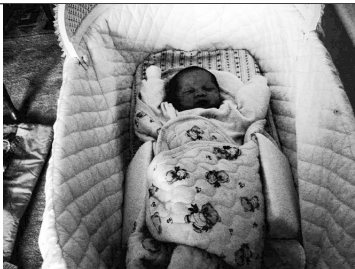
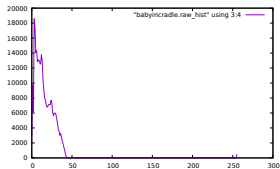
# Histogram Equalisation Example on image



# Histogram Equalisation Images



# Histogram Equalisation Images



# Histogram Matching

Histogram equalisation automatically produces a transformation which transforms a given image into another image which has a uniform histogram ( or nearly so). When blind enhancements are needed, this is fine, but there are situations, when we would like to have the histogram to have a specified shape.

The method of transforming an image so that the histogram matches with a specified histogram is called Histogram Matching Technique

# Histogram Matching

Consider the continuous intensities  $r$  and  $z$  (continuous random variables) with pdf  $p_r(r)$  and  $p_z(z)$ . Here  $p_z(z)$  is the specified histogram - histogram of  $r$  image has to be matched to this.

Let  $s$  be the random variable with the property

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

Suppose we define a random variable  $z$  with the property

$$G(z) = (L - 1) \int_0^z p_z(t) dt = s \Rightarrow z = G^{-1}(s)$$

We have converted both  $s$  and  $w$  to a uniform histogram.

# Histogram Matching

Thus, when the histogram of input image is known, the transformation  $s = T(r)$  can be evaluated. Similarly the transformation  $G(z) = s$  can be evaluated from the histogram specified  $p_z(z)$ . Perform the inverse mapping  $z = G^{-1}(s)$ . Thus from  $r$  we go to the  $s$  plane and then to  $z$  plane. This establishes the histogram matching procedure.



# Histogram Matching-Example

Assume continuous intensity values, suppose the PDF of a given image is

$$p_r(r) = \frac{2r}{(L-1)^2}, 0 \leq r \leq (L-1)$$

Find the transformation function that will produce an image whose intensity PDF is

$$p_z(z) = \frac{3z^2}{(L-1)^3}, 0 \leq z \leq (L-1)$$

First the histogram equalization transformation of the given input image is

$$s = T(r) = (L-1) \int_0^r p_r(w) dw = \frac{2}{(L-1)} \int_0^r w dw = \frac{r^2}{(L-1)}$$

# Histogram Matching-Example

The same transformation on the target image is

$$G(z) = (L - 1) \int_0^z p_z(w) dw = \frac{3}{(L-1)^2} \int_0^z w^2 dw = \frac{z^3}{(L-1)^2}$$

We require that

$$\begin{aligned} G(z) = s &\Rightarrow \frac{z^3}{(L-1)^2} = s \\ z &= [(L-1)^2 s]^{\frac{1}{3}} \end{aligned}$$

Substituting  $s$  in terms of  $r$ , we get the final transformation as

$$z = [(L-1)^2 s]^{\frac{1}{3}} = [(L-1)^2 \frac{r^2}{(L-1)}]^{\frac{1}{3}} = [(L-1)r^2]^{\frac{1}{3}}$$

# Histogram Matching - Discrete case

The transformation equation

$$\begin{aligned}s_k &= T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) \\ &= \frac{(L-1)}{MN} \sum_{j=0}^k k, k = 0, 1, \dots, L - 1\end{aligned}$$

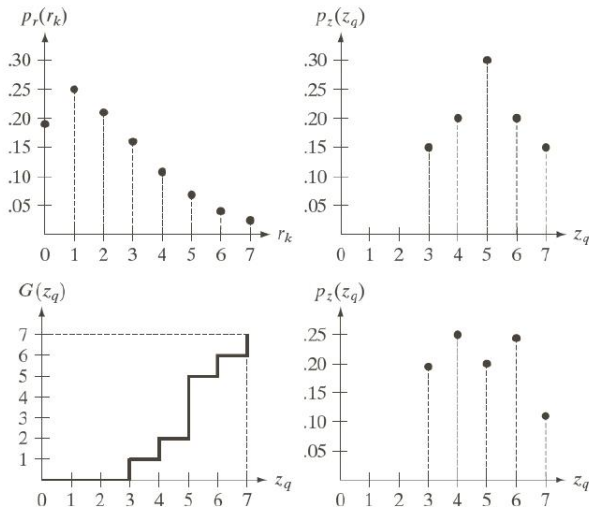
Similarly the discrete version of the transformation of the target image is

$$G(z_q) = (L - 1) \sum_{i=0}^q p_k(z_i)$$

Hence, the transformation is  $z_q = G^{-1}(s_k)$

Thus this gives the transformation for each value of  $s$  to the target image.-

# Histogram Matching - Discrete case



Top: Original PDF and Target PDF

Bottom: Transformation and Resultant PDF

# Histogram Matching-Example



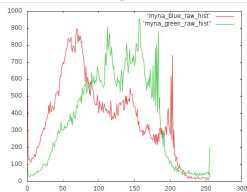
Original



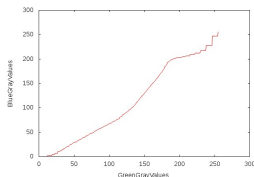
Target



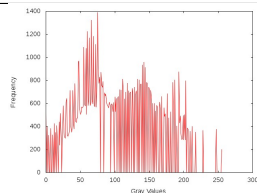
Matched



Original Histograms



MappingFn

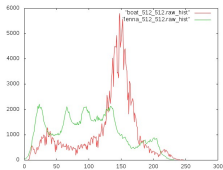


HistMatchedImage

# Histogram Matching-Example



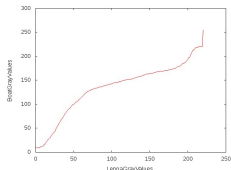
Original



Original Histograms



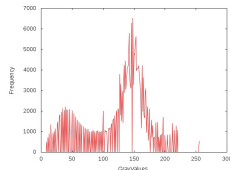
Target



MappingFn



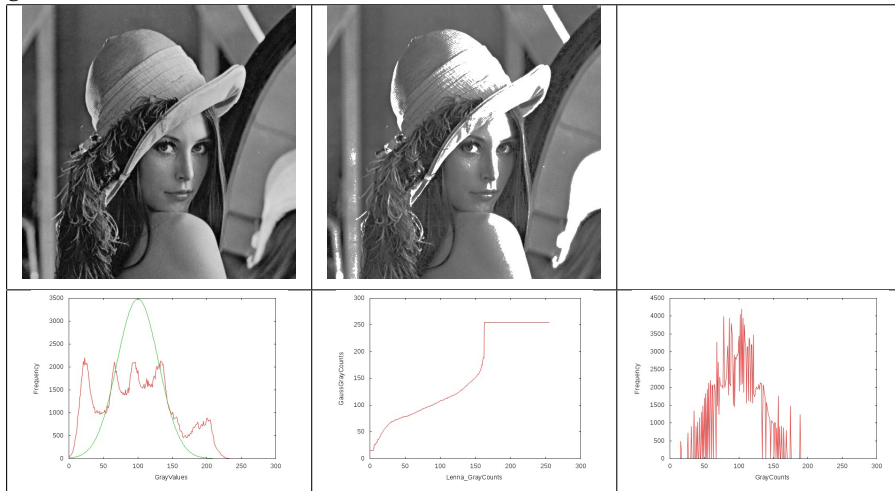
Matched



HistMatchedImage

# Histogram Specified Modification

Same as the previous case but the second histogram is already given.



# Local or Neighbourhood Operation

In neighbourhood operation, the pixels in an image are modified based on some function of the pixels in their neighbourhood. Linear spatial filtering is often referred to as convolving a mask with an image. The filter masks are called convolution masks or convolution kernels.

**Linear Filtering:** In this, each pixel in the input image is replaced by a linear combination of intensities of the neighbouring pixels. Thus, each pixel value in the output image is a weighted sum of the pixels in the neighbourhood of the corresponding pixel in the input image. Linear filtering can be used to smoothen an image as well as sharpen the image. A spatially invariant linear filter can be implemented using a convolution mask. If different weights are used for different parts of the image, then the linear filter is spatially variant.



# Mean or Average Filter

The mean filter replaces each pixel by the average of all values in the local neighbourhood. The size of the neighbourhood controls the amount of filtering. In a spatial averaging operation, each pixel is replaced by a weighted average of its neighbourhood pixels. The low pass filter preserves the smooth regions in the image and it removes the sharp variations leading to blurring effect. The  $3 \times 3$  spatial mask which can perform the averaging operation is given by

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

## Average Filter Example Images



## Average Filter Images Examples- $3 \times 3$ filter



## Average Filter Example Images $11 \times 11$ filter



# Average Filter Limitations

1. Averaging operation leads to the blurring of the image. It affects the sharpness of the image.
2. If the image is corrupted by impulse noise, then the impulse noise is attenuated but diffused to nearby pixels, but is not removed.
3. A single pixel with a very unrepresentative value can affect the mean value of all the pixels in its neighbourhood significantly.

## Average filter - physical significance

The mean filter is actually a low pass filter.

How?

Let us for simplicity a  $3 \times 3$  window for averaging.

Then the filter is given by  $x(n, n) =$

$$\frac{1}{9}(\delta(n-1, n-1) + \delta(n, n-1) + \delta(n+1, n-1) + \delta(n-1, n) + \delta(n, n) + \delta(n+1, n) + \delta(n-1, n+1) + \delta(n, n+1) + \delta(n+1, n+1))$$

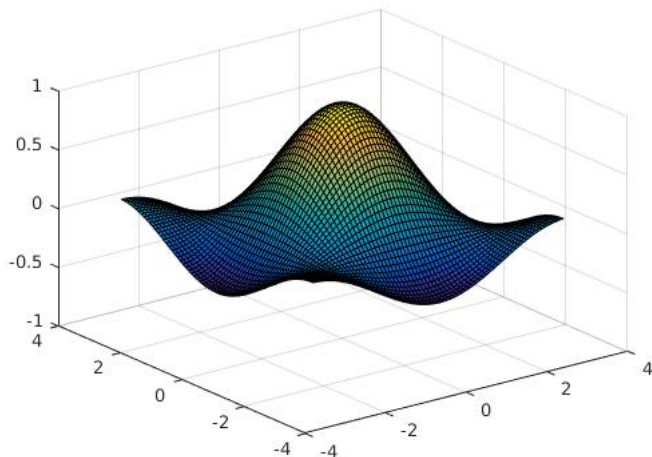
If we take Fourier transform of this filter it gives

$$\begin{aligned} H(\omega, \nu) &= \frac{1}{9}(e^{-i\omega}e^{-i\nu} + e^{-i\nu} + e^{i\omega}e^{-i\nu} + e^{-i\omega} + 1 + e^{i\omega} + e^{-i\omega}e^{i\nu} + e^{i\nu} + e^{i\omega}e^{i\nu}) \\ &= \frac{1}{9}(1 + 2\cos \omega + 2\cos \nu + 4\cos \omega \cos \nu) \end{aligned}$$

Clearly it can be seen that increasing the size of the window blurs the image.

# Average filter -physical significance

Clearly the system behaves like a low pass filter.



# Median Filter

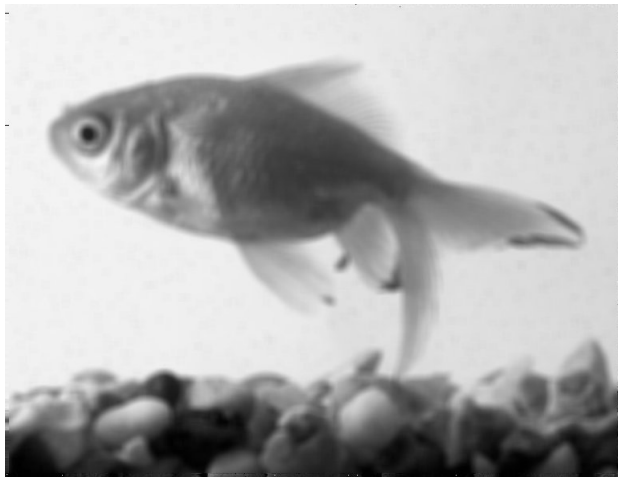
In this case, a window of size  $m \times m$  is moved as in Average Filter and at each location, the centre pixel is replaced by the median value of the grayvalues in that window. This filter is very useful in removing Salt and Pepper Noise. If we use the average filter to remove this, the noise will be diffused and will lead to a very blurred image, while median filter removes the noise without blurring the image.



# Salt and Pepper Noise removed



# Salt and Pepper Noise removed average filter size 11



# Salt and Pepper Noise removed median filter size 11



# Weighted Average Filter

The mask of a weighted average filter is given by  $\begin{bmatrix} 1 & 2 & 2 \\ 2 & 4 & 2 \\ 1 & 2 & 2 \end{bmatrix}$

It is obvious that the pixels earst to the centre are weighted more than the distant pixels, and hence the name. The pixel to be updated is replaced by a sum of the nearby piel value times the weights given in the matrix **and divided by the sum of the coefficients in the matrix**. If you do not do this, what will happen?

# Bartlett Filter

Obtained by convolving two box spatial filters in the spatial domain. The  $3 \times 3$  box filter is given by  $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ .

Bartlett window is

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} * \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{81} \begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

# Gaussian Filter

Gaussian filters are a class of linear smoothing filters with the weights chosen according to the shape of a Gaussian function. The Gaussian kernel is widely used for smoothing purpose. The Gaussian filter in the continuous space is given by

$$h(m, n) = \left[ \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{m^2}{2\sigma^2}} \right] \times \left[ \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{n^2}{2\sigma^2}} \right]$$

Thus the filter is separable. The Gaussian filter is very good for removing noise from a normal distribution.

# Advantages of Gaussian Filter

- ▶ Gaussian functions are rotationally symmetric in two dimensions. The amount of smoothing done is same in all directions. Thus, it will not bias subsequent edge detection operations
- ▶ Fourier Transform of a Gaussian function is itself a Gaussian function. Images are often corrupted by high frequency noise, and the desirable feature of the image will be distributed both in the low and the high frequency spectrum. The single lobe in the Fourier transform of a Gaussian means that the smoothed image will not be corrupted by contributions from unwanted high frequency signals, while most of the desirable signal properties will be retained
- ▶ The degree of smoothing is governed by variance  $\sigma^2$ . A larger value of variance implies a wider Gaussian and hence greater smoothing

# Gaussian Filtering

- ▶ Two dimensional Gaussian convolution can be performed by convolving the image with a one dimensional Gaussian and then convolving the result with the same one dimensional filter oriented orthogonal to the Gaussian used in the first stage
- ▶ Gaussian filter can be generated from a Pascal Triangle.



# Edge Detection

## Basic philosophy

Edge is found when two regions of different intensities meet. Points in an image where brightness changes abruptly are called edges or edge points. Edges are significant local changes of intensity in an image. Edges are the boundaries between segments. Edges are very important portion of the perceptual information content in an image. Edges are classified as:

Step edge - defines a perfect transition from one segment to another. In this case, the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the other side.

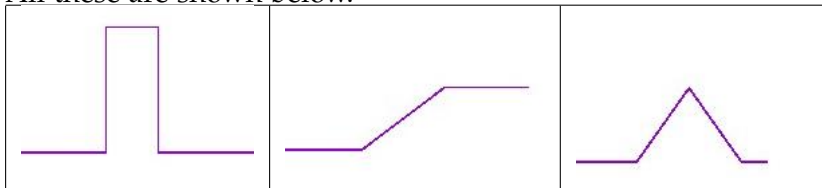


# Edge Detection

**Line Edge-** If segment of an image is very narrow, it necessarily has two edges in close proximity. This arrangement is called a line.

**Ramp Edge-** A ramp allows for a smoother transition between segments.

**Roof Edge-** Two nearby ramp edges result in a roof edge.  
All these are shown below.



# Edge Detection

What causes edges? They can be created by shadows, texture, geometry. Edges are basically discontinuities in the image intensity due to changes in the image structure. These discontinuities originate from different features in an image. Edge points are to be associated with the boundaries of objects and other kinds of changes. Edges within an image generally occur at various resolutions or scales and represent transitions of different degrees.

# Edge Detection

Edge detection is the process of finding meaningful transitions in an image. Points where sharp changes in the brightness occur typically form the border between different objects. These points can be detected by computing intensity differences in local image regions. That is, the edge detection algorithm should look for a neighbourhood with strong signs of change. Most of the edge detectors work on measuring the intensity gradient at a point in the image.

# Edge Detection and Sharpening

The principal objective of sharpening is to highlight transitions in intensity. The applications range from medical imaging to industrial inspection and robotics. We have seen that averaging using a spatial filter blurs the image. Now, averaging is analogous to integration, and hence we conclude that sharpening can be accomplished by spatial differentiation. Fundamentally, the strength of the response of a derivative operator is proportional to the degree of intensity discontinuity of the image at the point at which the operator is applied. The image differentiation enhances edges and other discontinuities (this includes noise as well) and deemphasises areas with slowly varying intensities.

# Fundamental of Edge Detection

Sharpening filters are based on first and second order derivatives

We shall consider one dimensional case first and focus attention on the behaviour of these derivatives in areas of constant intensity, at the onset and end of discontinuities (step and ramp discontinuities), and along intensity ramps.

Derivatives must have the following properties:

- ▶ Must be zero in areas of constant intensity;
- ▶ Must be nonzero at the onset of an intensity step or ramp
- ▶ Must be nonzero along ramps.

# Image Sharpening and Edge Detection

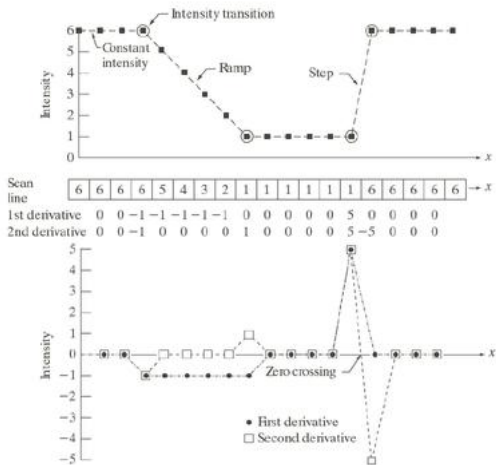
Basic definition of first derivative in  $x$  direction is

$\frac{\partial f}{\partial x} = f(x+1) - f(x)$  - the finite difference approximation.

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

We shall verify that these two definitions satisfy the conditions mentioned earlier.

# Sharpening and Edge Detection





# Sharpening and Edge Detection

The previous figure shows a section of a scan line (intensity profile). The values inside the small square are the gray values which are plotted as black dots above it. The scan contains - an intensity ramp, three sections of constant intensity, and an intensity step. The circles indicate the onset or end of intensity transitions. The first and second order derivatives computed using the two definitions are plotted.

# Derivatives Observations

- ▶ At the area of constant intensity, both derivatives are zero - so first condition is satisfied
- ▶ Next there is an intensity ramp, followed by a step- first derivative is nonzero at the onset of the ramp and the step; the second derivative also has the same property, and hence the second condition mentioned earlier is satisfied
- ▶ Along the ramp, both the first and second derivative are constant - first derivative non-zero, second derivative zero; and hence the third condition is also satisfied.
- ▶ The sign of the second derivative changes at the onset and end of a ramp or step. In a step transition, a line joining these two values crosses the horizontal axis midway between the two extremes. This *zero crossing* property is useful for locating edges.

# Edge Detection

Gradient Operator: A gradient is a two-dimensional vector that points to the direction in which the intensity grows fastest. The gradient operator  $\nabla$  is given by

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

If this operator  $\nabla$  is applied to the function  $f$ , then,

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

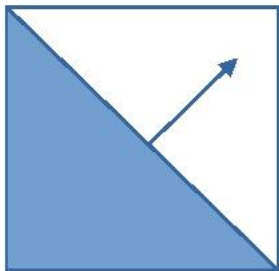
## Edge Detection

From this, we can find the magnitude of the gradient  $\|\nabla f\|$  and the gradient orientation  $\phi(\nabla f)$ .

$$\|\nabla f\| = \sqrt{f_x^2 + f_y^2}$$

The magnitude of the gradient gives the maximum rate of change of  $f(x, y)$  per unit distance and it is in the direction of the gradient.

The gradient orientation is  $\phi(\nabla f) = \tan^{-1}(\frac{f_y}{f_x})$



# Edge Detection using first order derivatives

The derivative of a digital pixel can be defined in terms of differences (finite differences in Numerical Methods). The first derivative of an image containing gray value pixels must fulfill the following conditions - it must be zero in flat segments, i.e., in area of constant gray level values; it must be non-zero at the beginning of a gray level step or ramp, and it must be non-zero along the ramp. The first order derivative of a one dimensional function  $f(x)$  can be obtained using  $\frac{\partial f}{\partial x} = f(x + 1) - f(x)$ .

An image is a function of two variables  $f(x, y)$ .

The equation above gives the partial derivative along the  $x$  - axis. Pixel discontinuity can be determined along eight possible directions such as up,down, left, right and along the four diagonals.

# Edge Detection using first order derivatives

In image processing, the partial derivatives are expressed in terms of finite differences. Thus, in the pixel direction, the derivative is given by  $f_x = f(i + 1, j) - f(i, j)$ . Similarly the derivative in the scan direction is  $f_y = f(i, j + 1) - f(i, j)$ . The approximations can be improved by using Central difference approximations.

# Roberts Kernel

Roberts kernel implements the finite difference approximations of the derivatives in the simplest way.

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

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

These are actually derivatives in diagonal directions.

The masks are given by  $G_x = \begin{vmatrix} -1 & 0 \\ 0 & 1 \end{vmatrix}$  and  $G_y = \begin{vmatrix} 0 & -1 \\ 1 & 0 \end{vmatrix}$

These filters have the shortest support, thus the position of the edges is more accurate, but the problem with short support of the filters is its vulnerability to noise.

# Prewitt Kernel

This kernel is based on Central difference approximation for the partial derivatives. The masks are given by

$$G_x = \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix} \text{ and } G_y = \begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix}$$

The Prewitt kernel is long. It differentiates in one direction and averages in other direction, so the edge detector is less vulnerable to noise.



# Edge Detection Prewitt



Prewitt:Vertical,Horizontal,Combined

# Sobel Operator

This is another first derivative based operator for edge detection. The masks are given by

$$G_x = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \text{ and } G_y = \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix}$$

# Edge Detection Sobel



Sobel:Vertical,Horizontal,Combined

## Second Derivative Methods of Detecting Edges in an Image

Theoretically, the edge point is one where the derivative is an extremum ( a maximum or a minimum). A **necessary condition - it is not sufficient** for this is that the second order derivative has to be zero. Thus, finding the optimal edges is equivalent to finding places where the second derivative is zero. The differential operators can be applied to images; the zeroes rarely fall exactly on a pixel. Typically, they fall between pixels. The zeros can be isolated by finding the **zero crossing**. Zero crossing is the place where one pixel is positive and a neighbouring pixel is negative. The problems with zero-crossing methods are the following:

1. Zero crossing methods produce two-pixel thick edges.
2. Zero crossing methods are extremely sensitive to noise.

The second derivative is obtained in images by the operator

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

## Second derivative based of Edge Detection

When applied to images this translates into

$$f_{xx} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$

$$f_{yy} = f(x, y+1) - 2f(x, y) + f(x, y-1)$$

Combining these two, we get an estimate of

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)]$$

This can be represented as a mask as

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

# Laplacian based Edge Detection

Sometimes, the corner pixels are also included. To do this, we write the second derivative in terms of those four pixels and add. Then combining the two results, we get the following

$$\text{mask. } L = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

The Laplacian operator subtracts the brightness values of each of the neighbouring pixels from the central pixel. When a discontinuity is present within the neighbourhood in the form of a point, line or edge, the result of the Laplacian is a non-zero value.

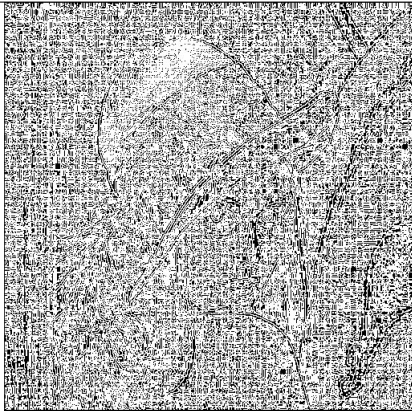
The Laplacian operator is rotationally invariant.

Drawback: Since Laplacian is an approximation of the Second derivative, it doubly enhances the noise in the image.

# Laplace Method

Besides, points where the gradient changes sign, but at the neighbouring pixels the gradients are small, will also be detected. If you mark all the zero crossing points and call them as edge points, you will get a ghostly image as shown below. To avoid this, we get the zero crossover points and only those points where the absolute value of Laplacian exceeds a threshold should be labelled as an actual point. However, there is a subjectivity on the choice of threshold. This can be overcome by first smoothing the image before the Laplacian is applied. This way spurious edges due to noise pixels can be avoided. Generally this smoothing is done by a Gaussian Mask. Hence the method is called Laplacian of Gaussian.

# Edge Detection using Laplacian Operator



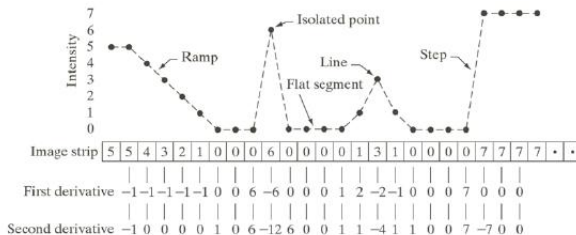
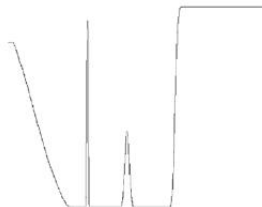
ZeroCross Image



Thresholded Laplacian Image



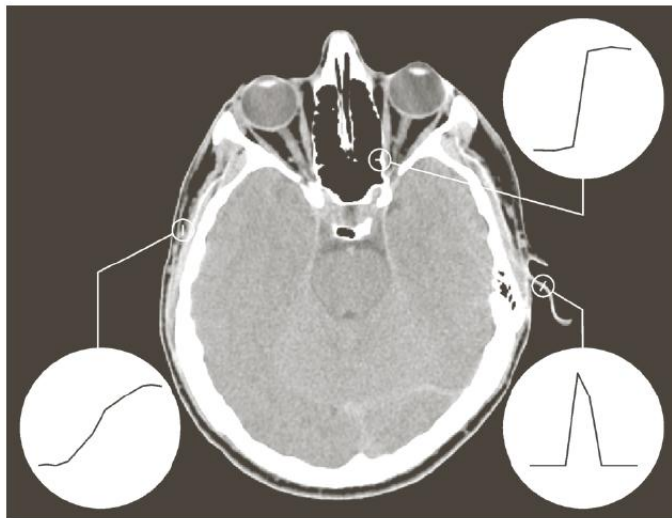
# Impact of Noise on Edge Detection-Edge Models



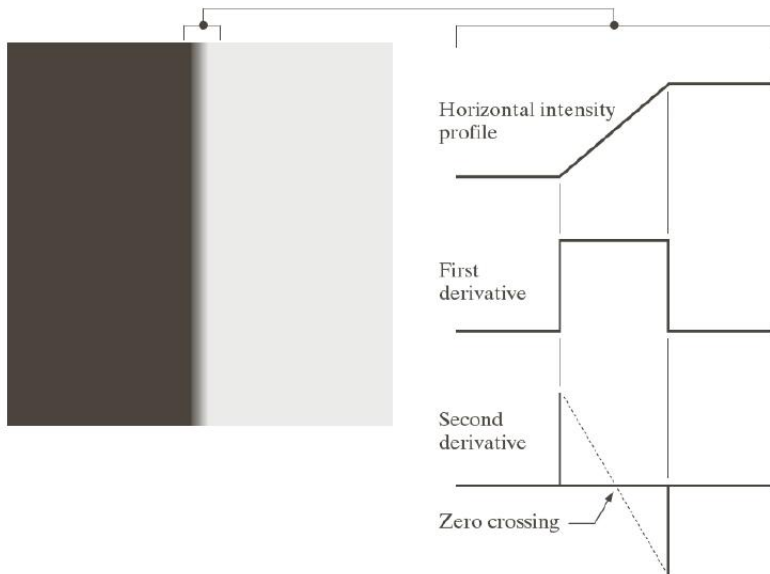
# Edge Models



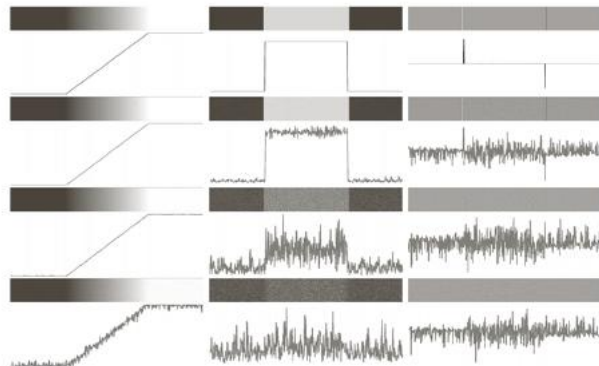
## Edge Models observed



# Impact of Noise on Edge Detection



# Impact of Noise on Edge Detection



First Column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviation 0,0.1,1.0,10 respectively. Second Column First derivative images and the intensity profiles; Third column: Second derivative images and the intensity profiles.

# Laplacian of Gaussian LOG

An important source of performance degradation of the Laplacian operator is the presence of noise in the input image. The effect of noise in edge determination can be minimised by smoothing the image prior to edge enhancement. The Laplacian of Gaussian operator (LOG) smooths the image through convolution with a Gaussian kernel followed by applying the Laplacian operator. The sequence of operations involved in an LOG operator is as follows:

# Laplacian of Gaussian LOG steps

1. Step 1 Smoothing of the input image  $f(m, n)$

The input image is smoothed by convolving it with the Gaussian mask  $h(m, n)$  to get the resultant smooth image  $g(m, n)$ .

$$g(m, n) = f(m, n) \otimes h(m, n)$$

2. Step 2 The Laplacian operator is applied to the result obtained in Step 1. This is represented by

$$g'(m, n) = \nabla^2(g(m, n)) = \nabla^2(f(m, n) \otimes h(m, n))$$

# Laplacian of Gaussian

The Gaussian mask is given by  $h(m, n) = e^{-\frac{r^2}{2\sigma^2}}$  where  $r^2 = m^2 + n^2$  and  $\sigma$  is the width of the Gaussian.

Since Convolution is a linear operator, hence, we can write

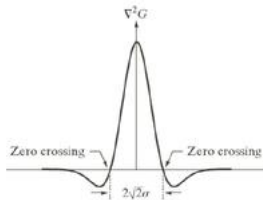
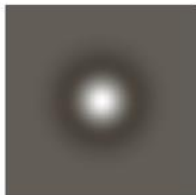
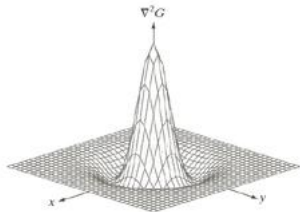
$$g'(m, n) = [\nabla^2(h(m, n))] \otimes f(m, n)$$

Substituting for  $h(m, n)$ , we get

$$\nabla^2[h(m, n)] = \frac{1}{\sigma^2} \left[ \frac{r^2}{\sigma^2} - 2 \right] e^{-\frac{r^2}{2\sigma^2}}$$



# Laplacian of Gaussian



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

# Unsharp Masking and Highboost Filtering

Another traditional approach for sharpening images consists of subtracting an unsharp (smoothed) version of an image from the original image. The process consists of the following steps:

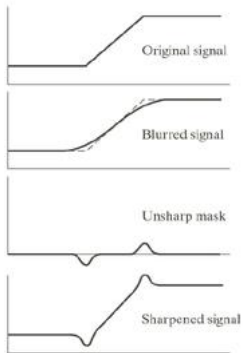
- ▶ Blur the original image
- ▶ Subtract the blurred image from the original image - result is mask
- ▶ Add a weighted portion of the mask to the original image

Mathematically, the entire operation can be written as

- ▶  $\bar{f}(x, y) = \text{Average of the image } f(x, y)$
- ▶ Get the mask as  $g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$
- ▶ Reconstructed image  $g(x, y) = f(x, y) + k g_{\text{mask}}(x, y)$

$k = 1$  means unsharp masking,  $k > 1$  high boost filtering, and  $k < 1$  deemphasizes the contribution of the unsharp mask.

# High Boost Filter



# Importance of Phase in Fourier Transform



Mag a Phase b



a



Mag b Phase a



b