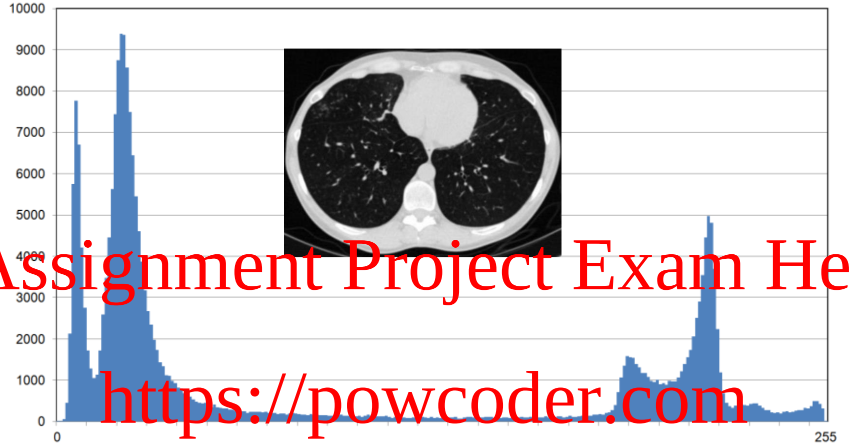


Computer Vision

Image Processing I

Image processing

- **Image processing** = image in > image out

- Aims to suppress distortions and enhance relevant information

- Prepares images for further analysis and interpretation

- **Image analysis** = image in > features out

- **Computer vision** = image in > interpretation out

Types of image processing

- Two main types of image processing operations:
  - Spatial domain operations (in image space)
  - Frequency domain operations (mainly in Fourier space)
- Two main types of spatial domain operations:
  - Point operations (intensity transformations on individual pixels)
  - Neighbourhood operations (spatial filtering on groups of pixels)

Types of image processing

- Two main types of image processing operations:
  - Spatial domain operations (in image space)
  - Frequency domain operations (mainly in Fourier space)

- Two main types of spatial domain operations:
  - Point operations (intensity transformations on individual pixels)
  - Neighbourhood operations (spatial filtering on groups of pixels)

Topics and learning goals

- Describe the workings of basic point operations

Contrast stretching, thresholding, inversion, log/power transformations

- Understanding and using the intensity histogram

Histogram specification, equalization, matching

- Defining arithmetic and logical operations

Summation, subtraction, AND, OR, et cetera

Spatial domain operations

- General form of spatial domain operations

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

where

$f(x, y)$ is the input image

$g(x, y)$ is the processed image

$T[\cdot]$ is the operator applied at (x, y)

Spatial domain operations

- Point operations: T operates on individual pixels

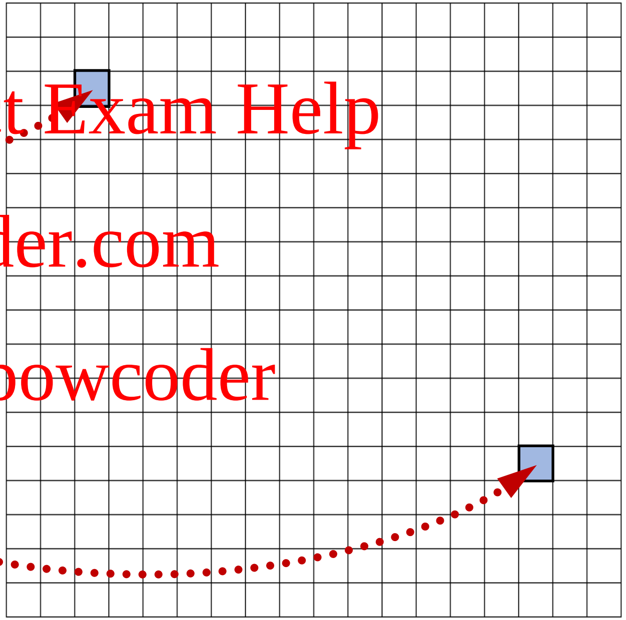
$$T: \mathbb{R} \longrightarrow \mathbb{R} \qquad g(x, y) = T\big(f(x, y)\big)$$

- Neighbourhood operations: T operates on multiple pixels

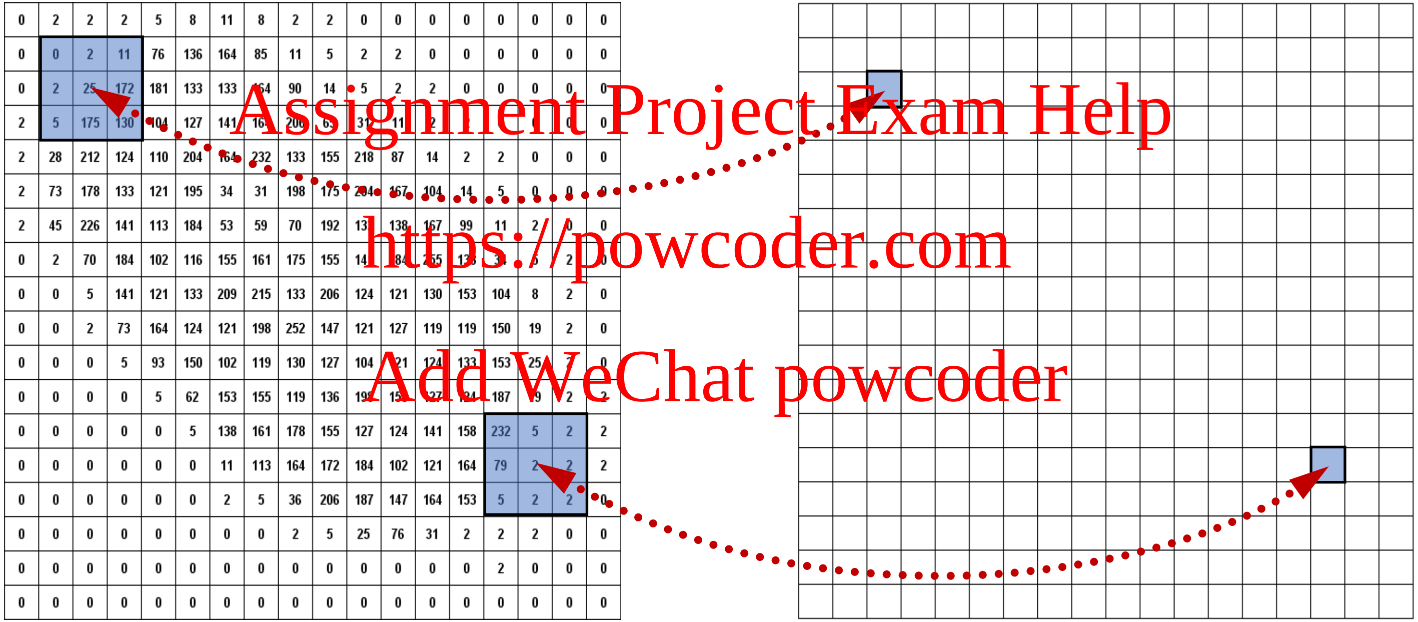
$$T: \mathbb{R}^2 \longrightarrow \mathbb{R} \qquad g(x, y) = T(f(x, y), f(x+1, y), f(x-1, y), \dots)$$

Point operations

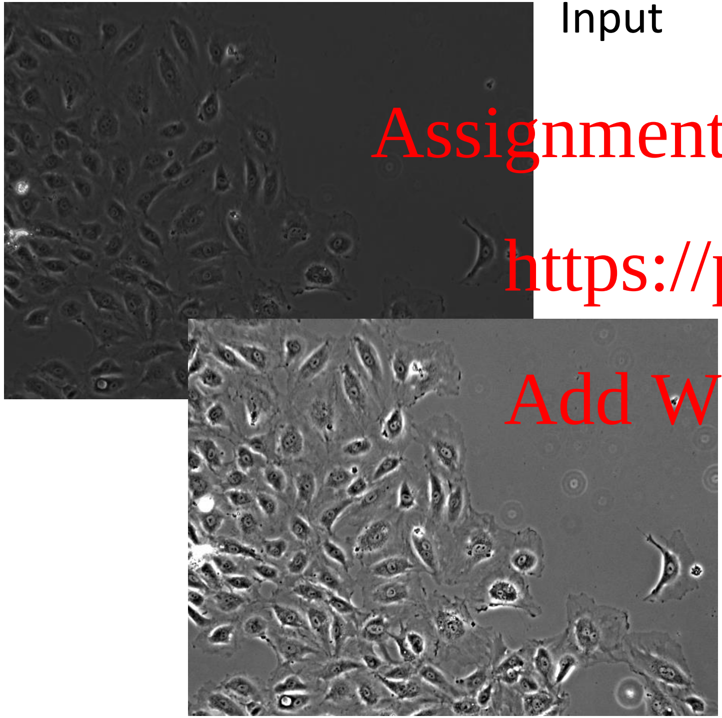
0	2	2	2	5	8	11	8	2	2	0	0	0	0	0	0	0	0
0	0	2	11	76	136	164	85	11	5	2	2	0	0	0	0	0	0
0	2	25	172	181	133	133	[illegible]	90	14	5	2	2	0	0	0	0	0
2	5	175	130	104	127	141	[illegible]	[illegible]	[illegible]	[illegible]	11	[illegible]	[illegible]	[illegible]	[illegible]	0	0
2	28	212	124	110	204	164	232	133	155	218	87	14	2	2	0	0	0
2	73	178	133	121	195	34	31	198	175	204	167	104	14	5	0	0	0
2	45	226	141	113	184	53	59	70	192	[illegible]	138	167	99	11	2	0	0
0	2	70	184	102	116	155	161	175	155	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	0
0	0	5	141	121	133	209	215	133	206	124	121	130	153	104	8	2	0
0	0	2	73	164	124	121	198	252	147	121	127	119	119	150	19	2	0
0	0	0	5	93	150	102	119	130	127	104	[illegible]	124	133	153	25	[illegible]	0
0	0	0	0	5	62	153	155	119	136	[illegible]	[illegible]	[illegible]	[illegible]	187	[illegible]	2	2
0	0	0	0	0	5	138	161	178	155	127	124	141	158	232	5	2	2
0	0	0	0	0	0	11	113	164	172	184	102	121	164	79	2	2	2
0	0	0	0	0	0	2	5	36	206	187	147	164	153	5	2	2	0
0	0	0	0	0	0	0	0	2	5	25	76	31	2	2	2	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Neighbourhood operations

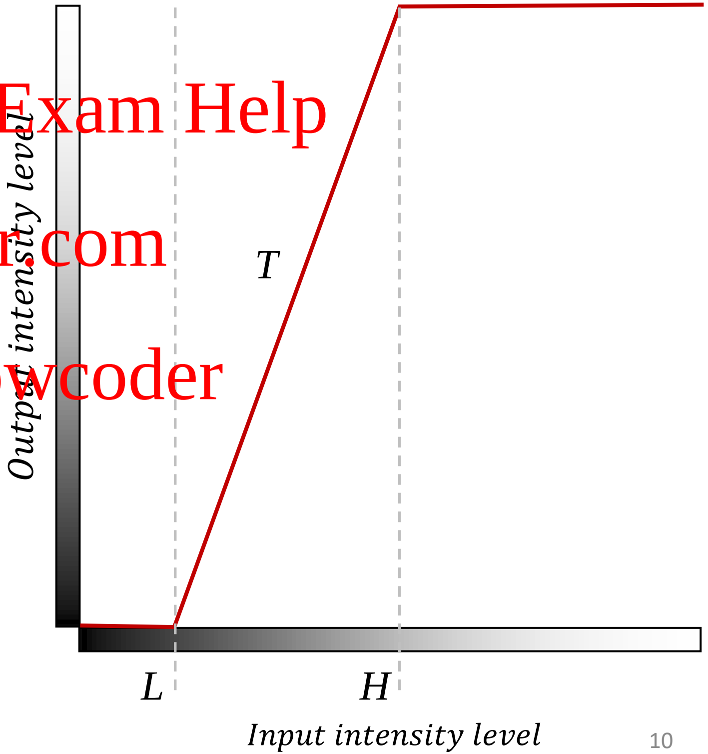


Contrast stretching



Input

Output



Contrast stretching

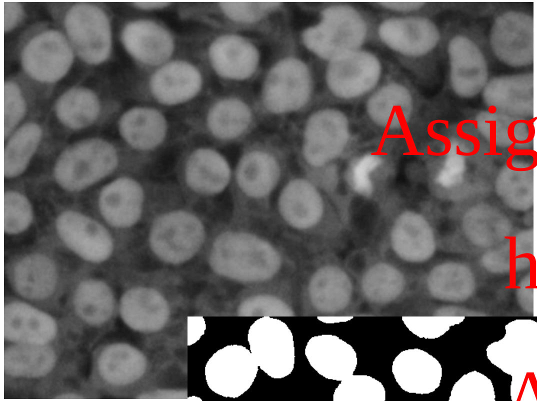
- Produces images of higher contrast

  - Puts values below L in the input to black in the output

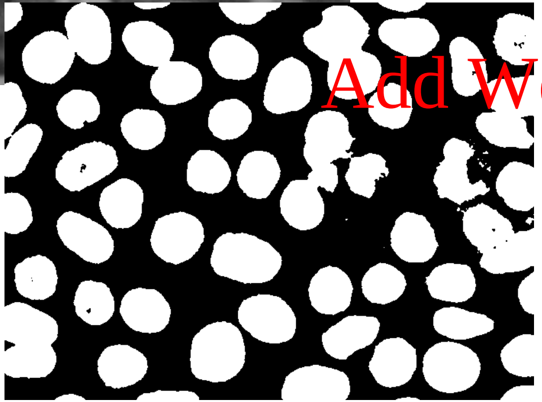
  - Puts values above H in the input to white in the output

- Linearly scales values between L and H in the input to the maximum range in the output

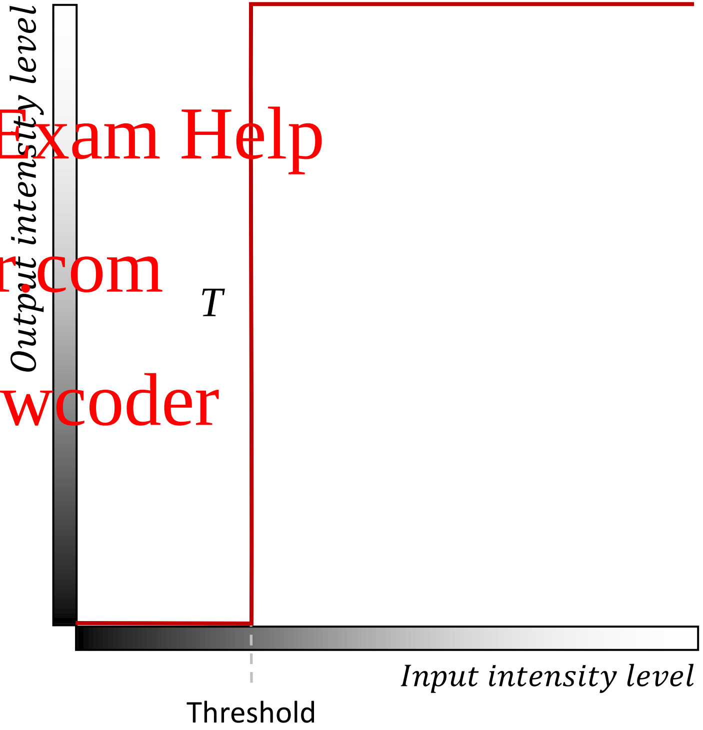
Intensity thresholding



Input



Output



Intensity thresholding

- Limiting case of contrast stretching
- Produces binary images of gray-scale images
  - Puts values below the threshold to black in the output
  - Puts values equal/above the threshold to white in the output
- Popular method for image segmentation (discussed later)
- Useful only if object and background intensities are very different

Automatic intensity thresholding

- Otsu's method for computing the threshold

Exhaustively searches for the threshold minimising the intra-class variance

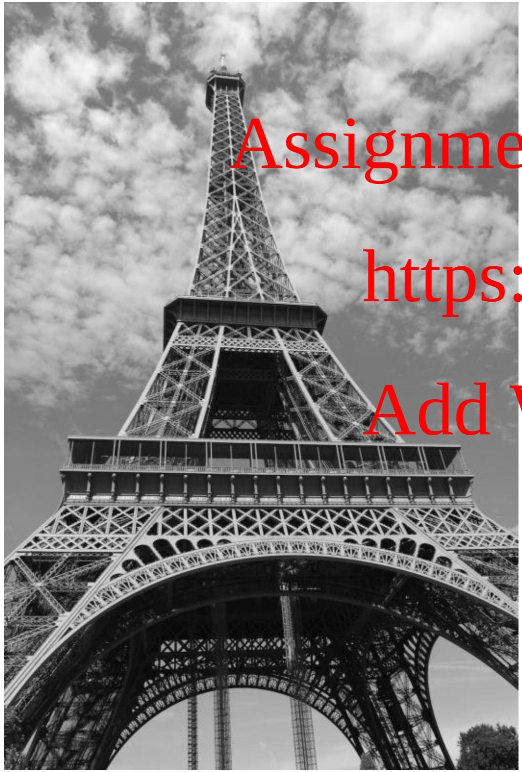
$$\sigma_W^2 = p_0 \sigma_0^2 + p_1 \sigma_1^2$$

Equivalent to maximising the inter-class variance (much faster to compute)

$$\sigma_B^2 = p_0 p_1 (\mu_0 - \mu_1)^2$$

Here, p_0 is the fraction of pixels below the threshold (class 0), p_1 is the fraction of pixels equal to or above the threshold (class 1), μ_0 and μ_1 are the mean intensities of pixels in class 0 and class 1, σ_0^2 and σ_1^2 are the intensity variances, and $p_0 + p_1 = 1$ and $\sigma_0^2 + \sigma_1^2 = \sigma^2$

Otsu thresholding



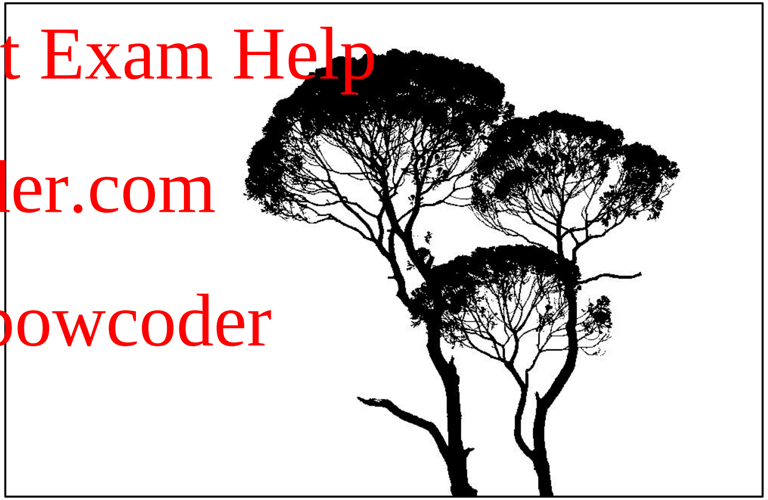
Automatic intensity thresholding

- Iso-data method for computing the threshold

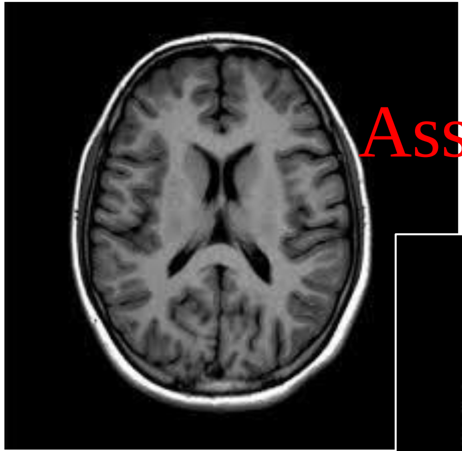
1. Select an arbitrary initial threshold t
2. Compute μ_0 and μ_1 with respect to the threshold
3. Update the threshold to the mean of the means: $t = (\mu_0 + \mu_1)/2$
4. If the threshold changed in Step 3, go to Step 2

Upon convergence, the threshold is midway between the two class means

Iso-data thresholding

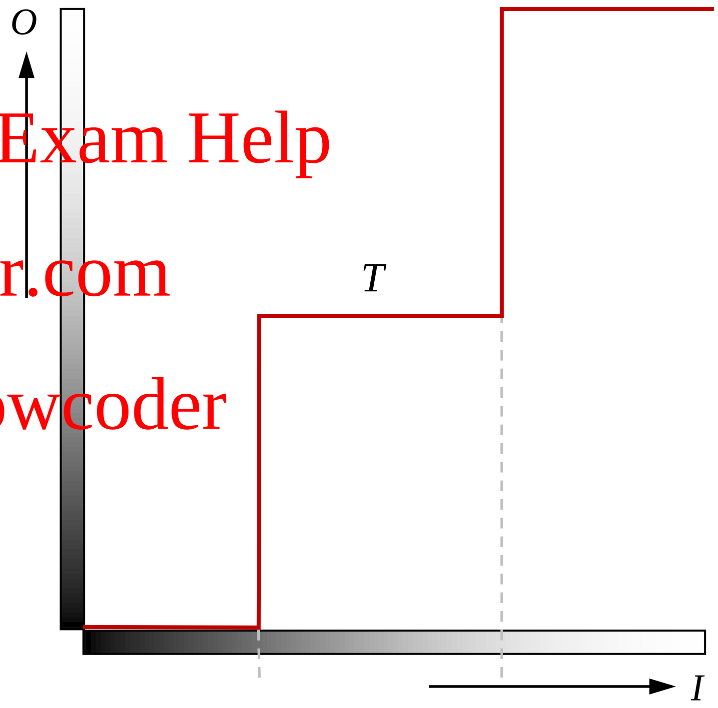


Multi-level thresholding



Input

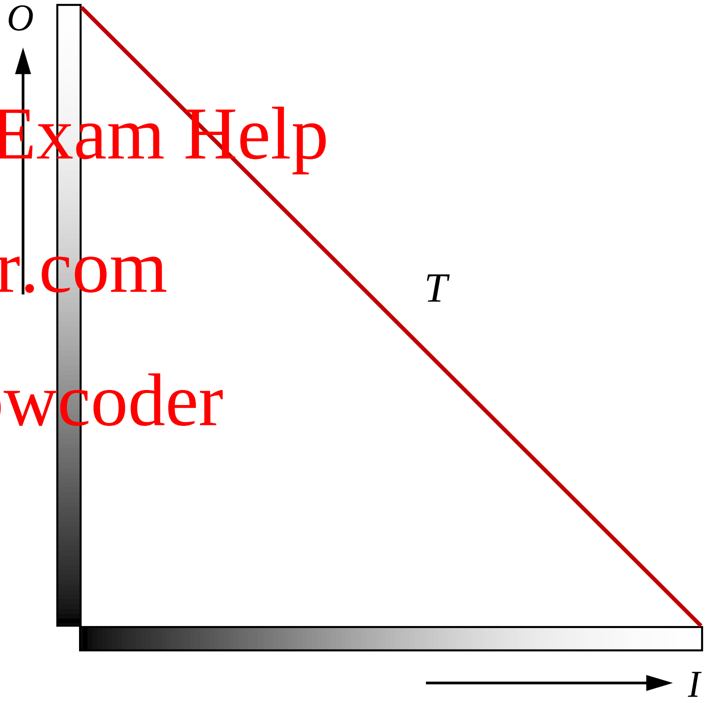
Output



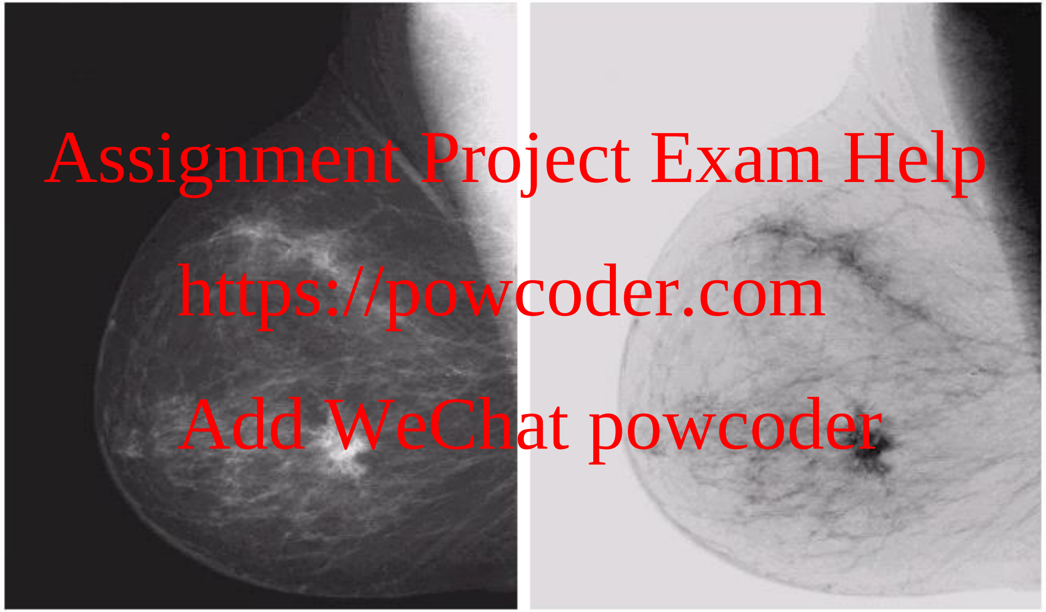
Intensity inversion

Input

Output



Intensity inversion



Useful for enhancing gray/white details in images within dominant black areas

Log transformation

- Definition of log transformation

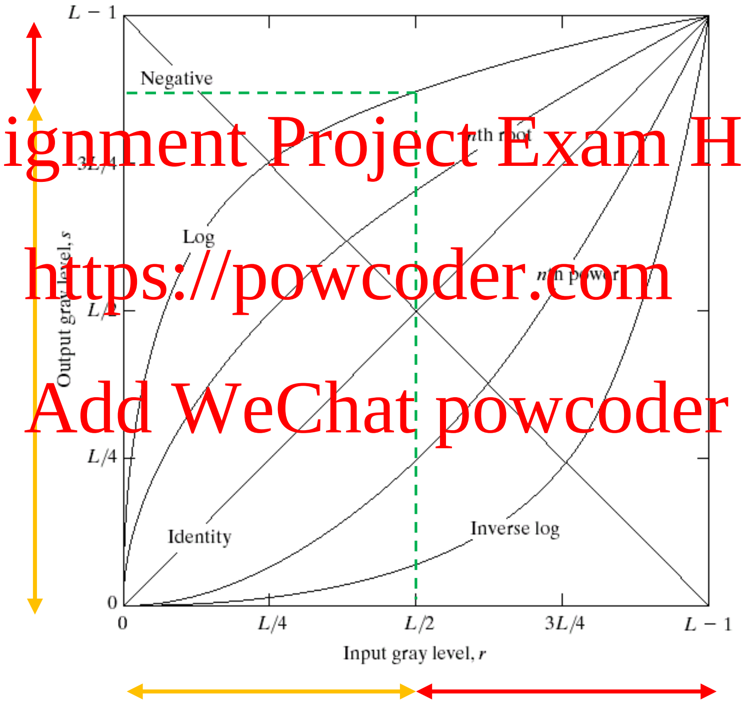
$$s = c\log(1 + r)$$

where r is the input intensity, s is the output intensity, and c is a constant

  - Maps narrow range of low gray-level values into wider range of output values, and opposite for higher gray-level values

  - Also compresses dynamic range of images with large variations in pixel values (such as Fourier spectra, discussed later)

Log transformation



Power transformation

- Definition of power transformation

$$s = c\,r^{\gamma}$$

where c and γ are constants

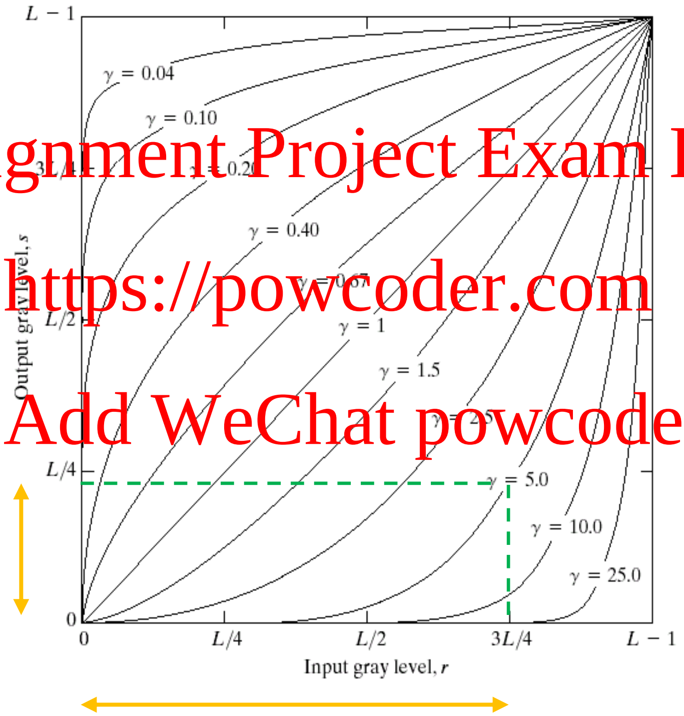
– Similar to log transformation

– Represents a whole family of transformations by varying γ

– Many devices respond according to a power law (gamma correction)

– Useful for general-purpose contrast manipulation

Power transformation



Power transformation

$c = 1$

Input

$\gamma = 3$

$\gamma = 4$

$\gamma = 5$

Piecewise linear transformations

- Complementary to other transformation methods
- Enable more fine-tuned design of transformations
- Can have very complex shapes
- Requires more user input

Piecewise contrast stretching

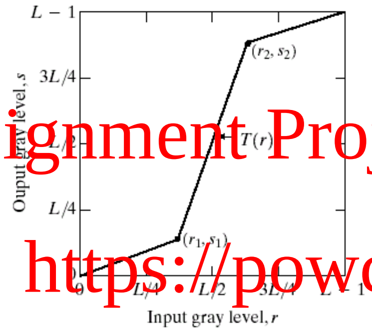
- One of the simplest piecewise linear transformations

- Increases the dynamic range of gray levels in images

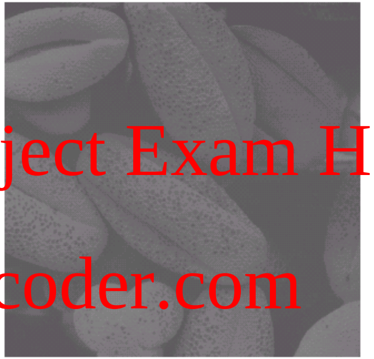
- Used in display devices or recording media to span full range

Piecewise contrast stretching

Transform



Input



Transformed

Binary Thresholding

Gray-level slicing

- Used to highlight specific range of gray levels

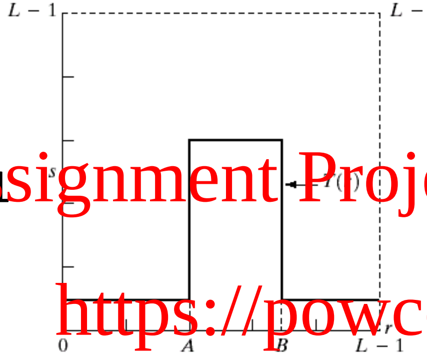
- Two different slicing approaches:

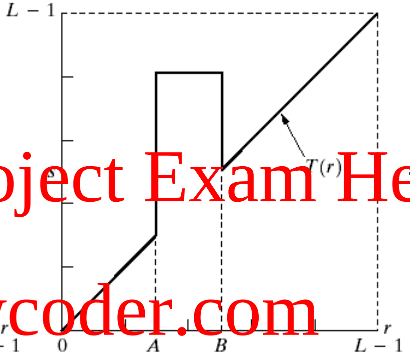
1) High value for all gray levels in a range of interest and low value for all others (produces a binary image)

2) Brighten a desired range of gray levels while preserving background and other gray-scale tones of the image

Gray-level slicing

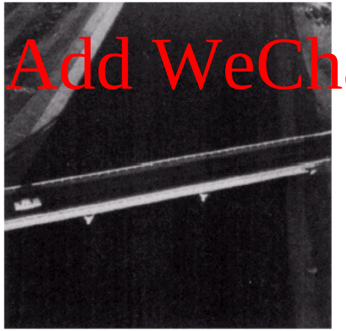
Transform 1

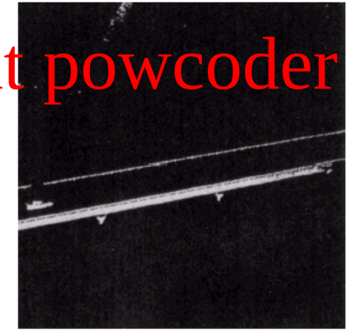




Transform 2

Input



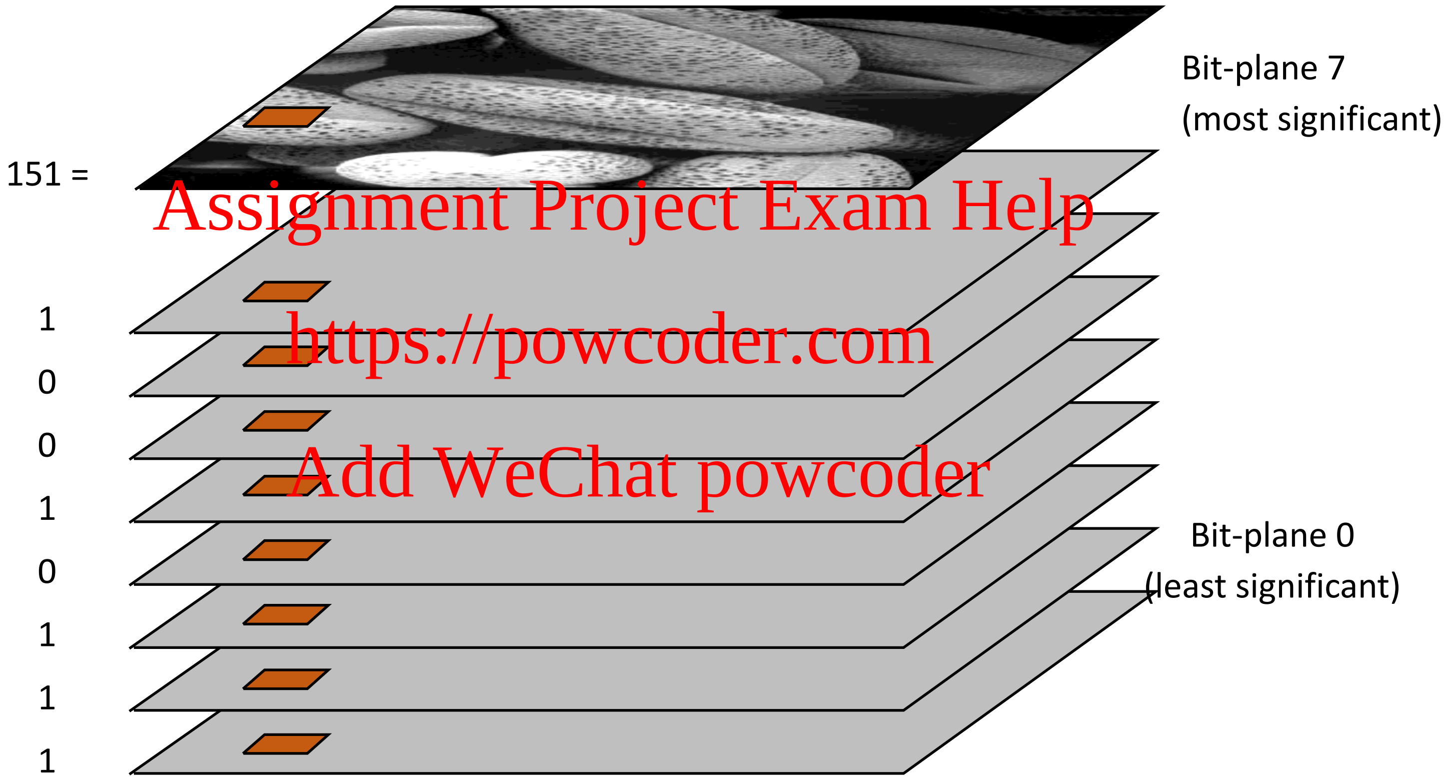


Result of Transform 1

Bit-plane slicing

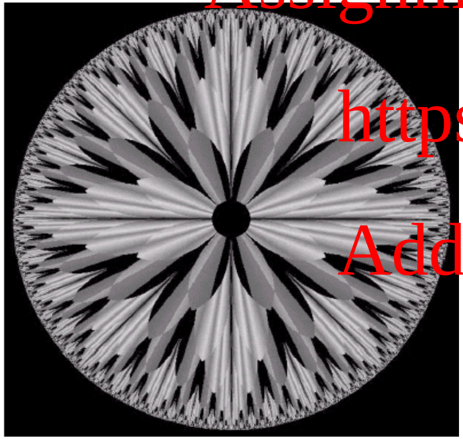
- Highlights contribution to total image by specific bits
- An image with *n*-bits/pixel has *n* bit-planes
- Slicing can be useful for image compression

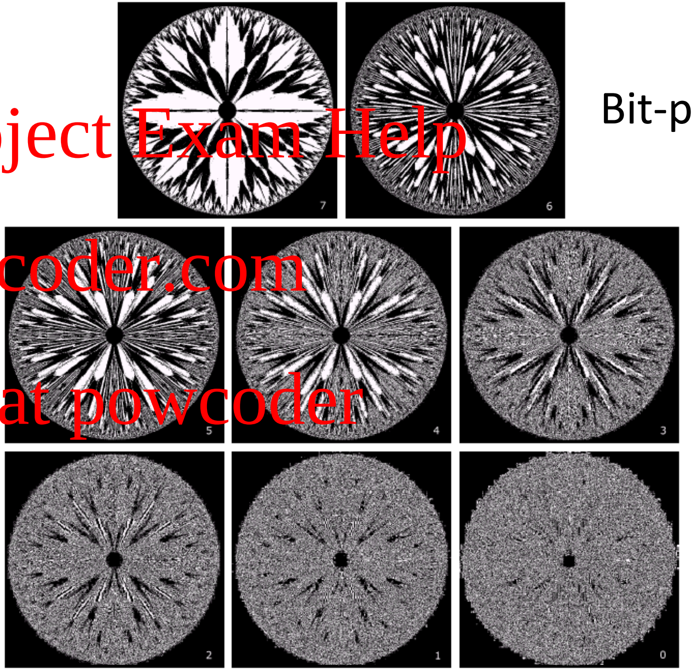
Bit-planes of an 8-bit image



Bit-planes of an 8-bit image

Input

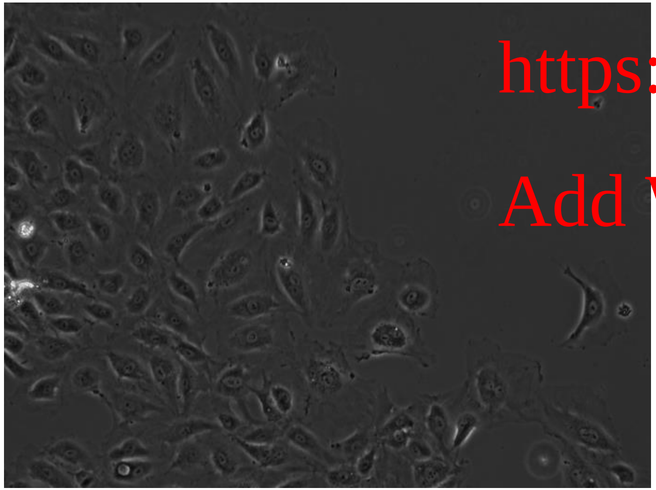




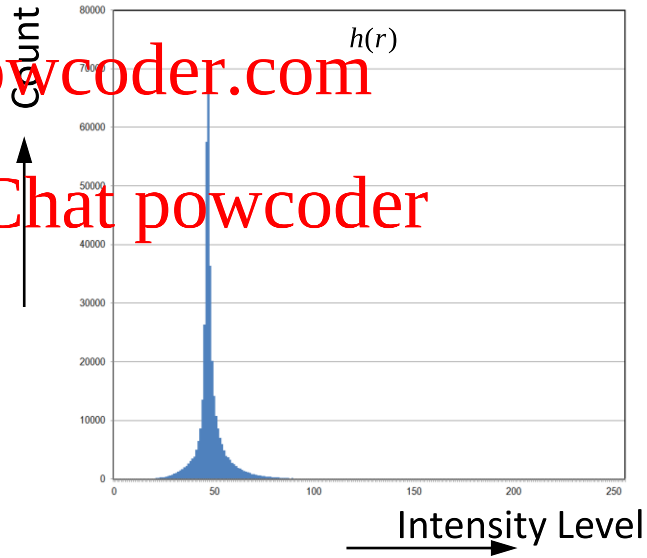
Bit-planes

Histogram of pixel intensities

- For every possible intensity level, count the number of pixels having that level, and plot the pixel counts as a function of the level



8-bit image



$L = 2^8 = 256$

N = #pixels

$$\sum_{r=0}^{L-1} h(r) = N$$

Normalized histogram = probability function

$$\frac{1}{N} h(r) = p(r)$$

Histogram processing

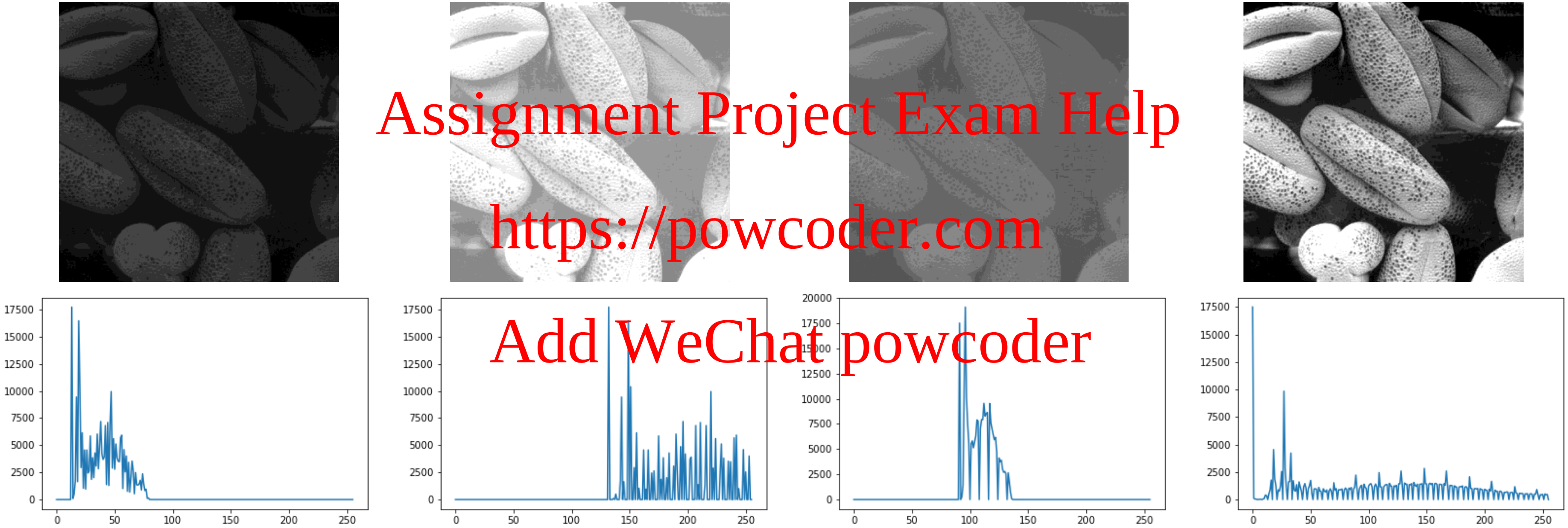
- **Histogram equalization**

  Aim: To get an image with equally distributed intensity levels over the full intensity range

- **Histogram specification** (also called **histogram matching**)

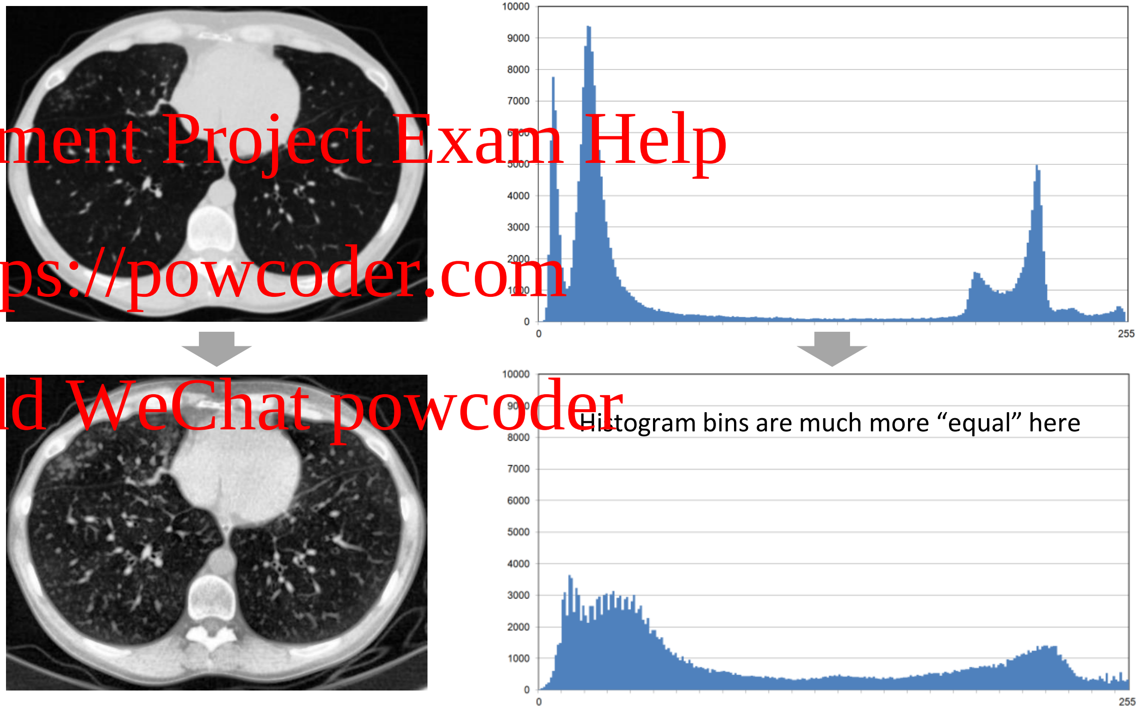
  Aim: To get an image with a specified intensity distribution, determined by the shape of the histogram

Histogram processing



Histogram equalization

Enhances contrast for intensity values near histogram maxima and decreases contrast near histogram minima



 37

Histogram equalization

- Let $r \in [0, L-1]$ represent gray levels of the image

  $r = 0$ represents black and $r = L - 1$ represents white

- Consider transformations $s = T(r)$, $0 \leq r \leq L - 1$, satisfying

  1) $T(r)$ is single-valued and monotonically increasing in $0 \leq r \leq L - 1$

     This guarantees that the inverse transformation exists

  2) $0 \leq T(r) \leq L - 1$ for $0 \leq r \leq L - 1$

     This guarantees that the input and output ranges will be the same

Histogram equalization (continuous case)

Consider r and s as continuous random variables over $[0, L-1]$ with PDFs $p_r(r)$ and $p_s(s)$

If $p_r(r)$ and $T(r)$ are known and $T^{-1}(s)$ satisfies monotonicity, then, from probability theory

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

Let us choose: $s = T(r) = (L-1)\int_0^r p_r(\xi)d\xi$

This is the CDF of r which satisfies conditions (1) and (2)

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

Therefore: $p_{s(s)} = p_{r(r)}\left|\frac{1}{(L-1)p_r(r)}\right| = \frac{1}{L-1}$ for $0 \leq s \leq L-1$

This is a uniform distribution!

Histogram equalization (discrete case)

For discrete values we get probabilities and summations instead of PDFs and integrals:

$$p_r(r_k) = n_k/MN \quad \text{for} \quad k = 0, 1, \dots, L-1$$

where MN is total number of pixels in image, n_k is the number of pixels with gray level r_k and L is the total number of gray levels in the image

Thus $s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$ for $k = 0, 1, \dots, L-1$

This transformation is called *histogram equalization*

However, in practice, getting a perfectly uniform distribution for discrete images is rare

Histogram matching (continuous case)

Assume that r and s are continuous intensities and $p_z(z)$ is the target distribution for the output image

From our previous analysis we know that the following transformation results in a uniform distribution:

$$s = T(r) = (L-1)\int_0^r p_r(\xi)d\xi$$

Now we can define a function $G(z)$ as:

$$G(z) = (L-1)\int_0^z p_z(\xi)d\xi = s$$

Therefore:

$$z = G^{-1}(s) = G^{-1}[T(r)]$$

Histogram matching (discrete case)

For discrete image values we can write:

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

$$k = 0, 1, \ldots, L-1$$

and

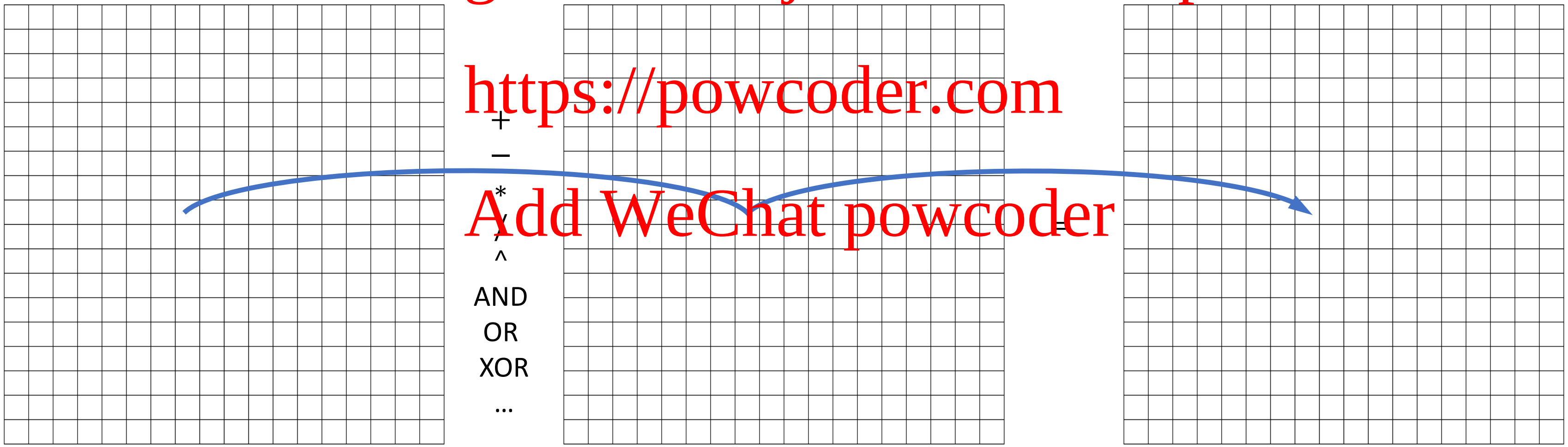
$$G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i)$$

therefore:

$$z_q = G^{-1}(s_k)$$

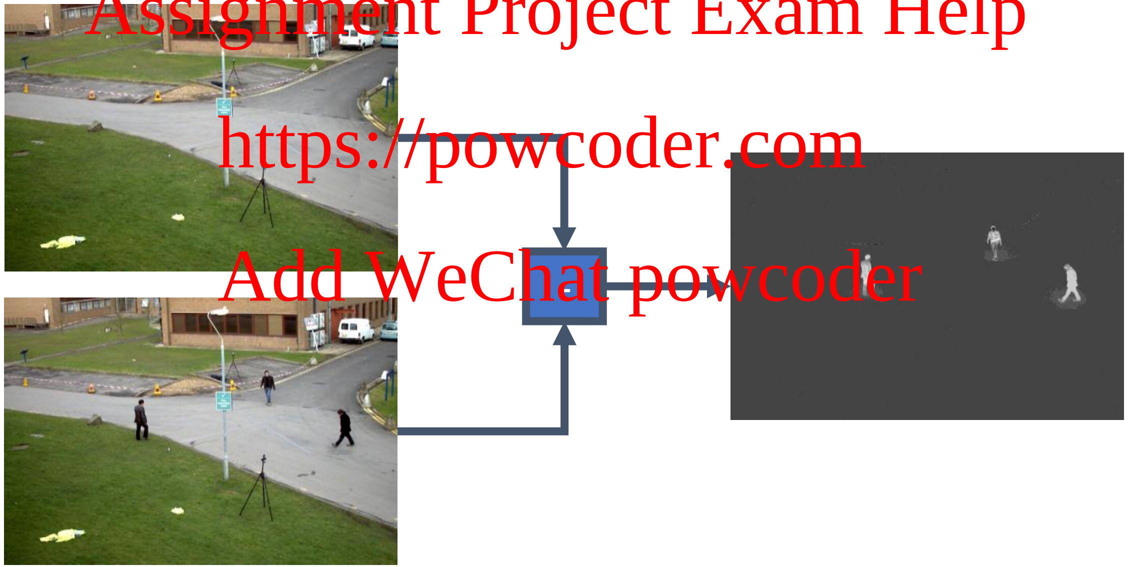
Arithmetic and logical operations

- Defined on a pixel-by-pixel basis between two images



Arithmetic and logical operations

- Useful arithmetic operations include addition and subtraction

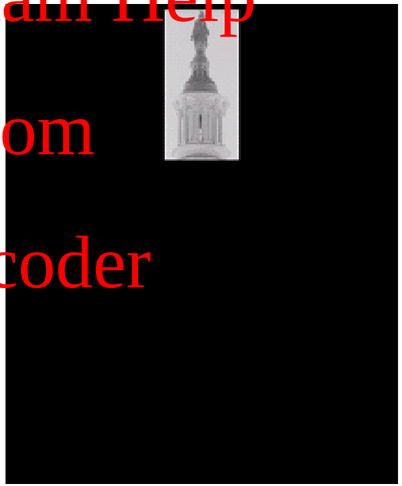


Arithmetic and logical operations

- Useful logical operations include AND and OR

Input

Mask



Input AND Mask

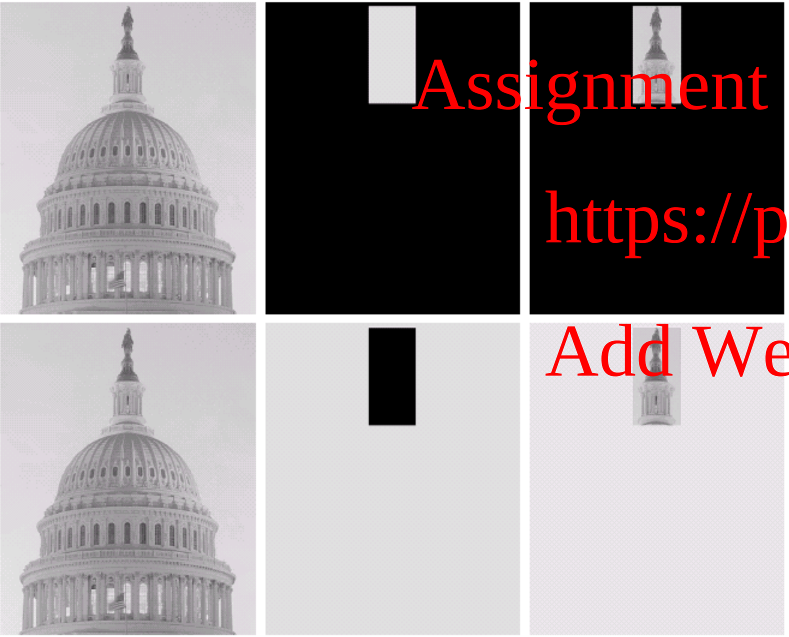
Arithmetic/Logic Operations

- on pixel-by-pixel basis between 2 or more images

- AND and OR operations are used for *masking*- selecting subimages as RoI

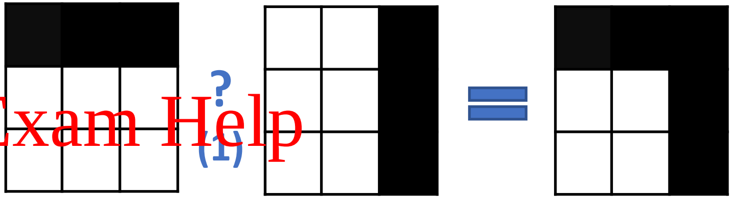
- subtraction and addition are the most useful arithmetic operations

Chapter 3
Image Enhancement in the Spatial Domain



a b c
d e f

FIGURE 3.27 (a) Original image. (b) AND image mask. (c) Result of the AND operation on images (a) and (b). (d) Original image. (e) OR image mask. (f) Result of operation OR on images (d) and (e).



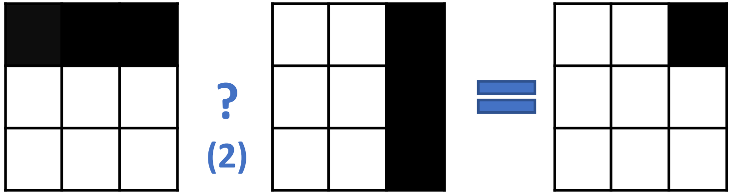


Image Averaging

- Noisy image *g(x, y)* formed by adding noise *n(x, y)* to uncorrupted image *f(x, y)*:

  $$g(x, y) = f(x, y) + n(x, y)$$

- Assume that at each *(x, y)*, the noise is uncorrelated and has zero average value.

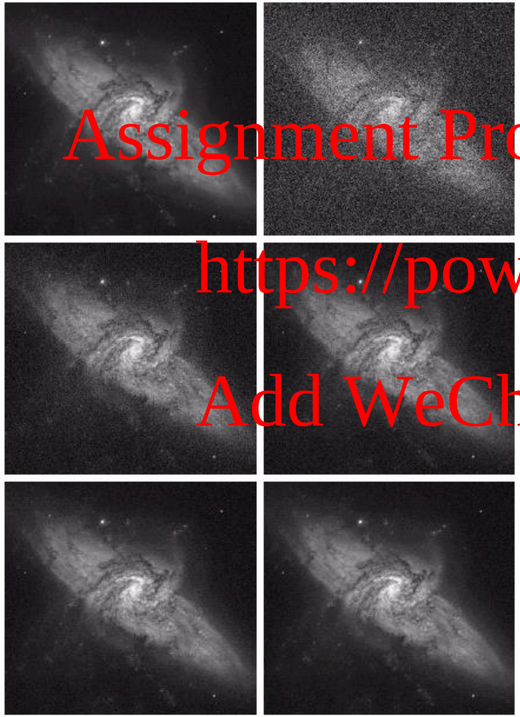
- **Aim:** To obtain smoothed result by averaging a set of noisy images

  $g_i(x, y), \; i = 1, 2, \ldots, K$

$$g(x, y) \approx \frac{1}{K} \sum_{i=1}^{K} g_i(x, y)$$

- As K increases, the variability of the pixel values decreases
- assumes that images are spatially registered

Chapter 3
Image Enhancement in the Spatial Domain



a b
c d
e f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64$, and 128 noisy images. (Original image courtesy of NASA.)

Spatial Filtering

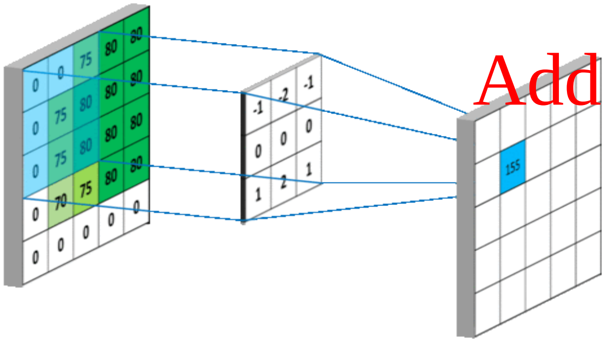
- These methods use a small *neighbourhood* of a pixel in the input image to produce a new brightness value for that pixel
- Also called *filtering* techniques
- Neighbourhood of (x, y) is usually a square or rectangular subimage centred at (x, y)- called *filter / mask / kernel /template / window*
- A *linear transformation* calculates a value in the output image *g(i, j)* as a linear combination of brightnesses in a local neighbourhood of the pixel in the input image *f(i, j)*, weighted by coefficients *h*.

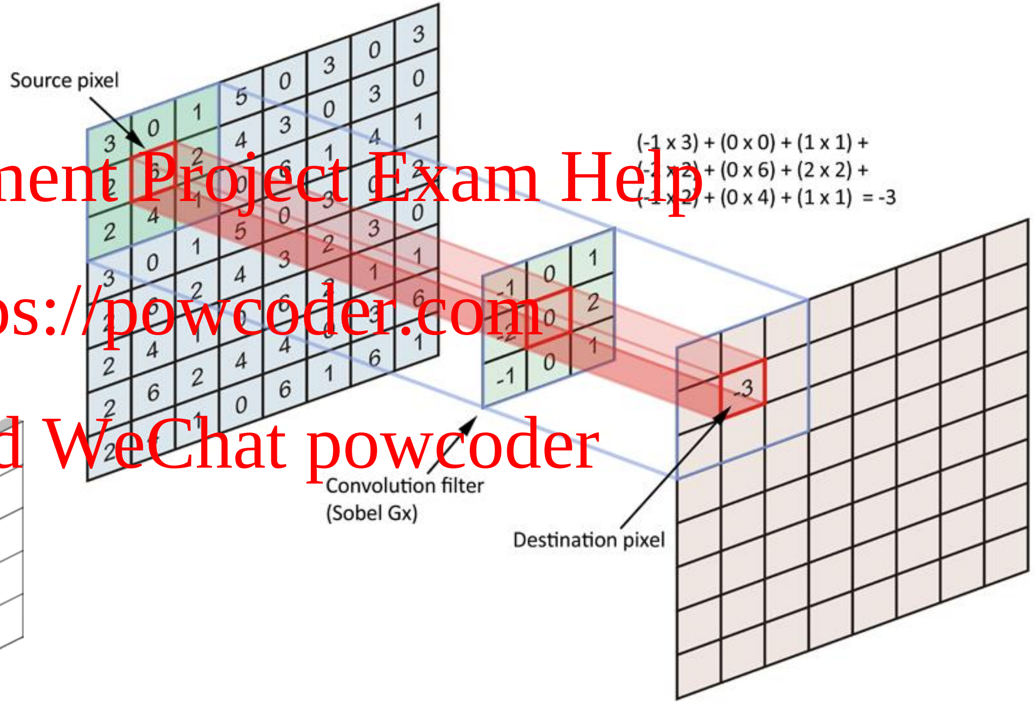
$$g(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} h(i, j) f(x - i, y - j)$$

- This is called a *discrete convolution* with a convolution mask h

Spatial Filtering

Convolution





Smoothing Spatial Filters

Used for blurring, noise reduction

Neighbourhood Averaging (Mean Filter)

$$g(x, y) = \frac{1}{P}\sum_{(n,m)\in S} f(n, m)$$

- Replace intensity at pixel (x, y) with the **average** of the intensities in a neighbourhood of (x, y).

- We can also use a **weighted average**, giving more importance to some pixels over others in the neighbourhood- reduces blurring
- Neighbourhood averaging blurs edges

Chapter 3
Image Enhancement in the Spatial Domain

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

a b

FIGURE 3.34 Two 3×3 smoothing (averaging) filter masks. The constant multiplier in front of each mask is equal to the sum of the values of its coefficients, as is required to compute an average.

Chapter 3
Image Enhancement in the Spatial Domain

a b c

FIGURE 3.36 (a) Image from the Hubble Space Telescope. (b) Image processed by a 15×15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Another example

Consider an image of constant intensity, with widely isolated pixels with different intensity from the background. We wish to detect these pixels.

Use the following mask:

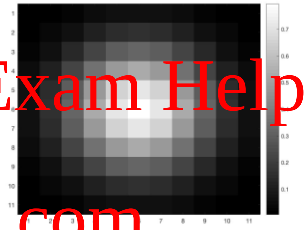
$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

Smoothing Spatial Filters

- **Aim:** To suppress noise, other small fluctuations in image- may be result of sampling, quantization, transmission, environment disturbances during acquisition
- Uses redundancy in the image data
- May blur sharp edges, so care is needed

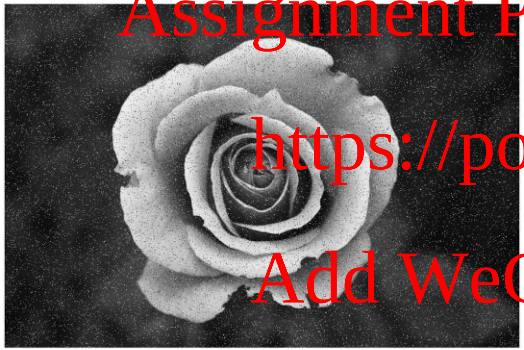
Gaussian Filter

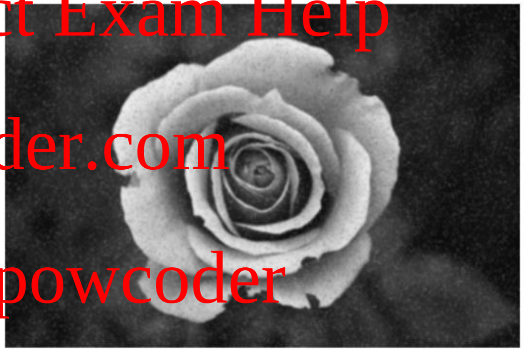
$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left[\frac{x^2+y^2}{2\sigma^2}\right]}$$



- Replace intensity at pixel (x, y) with the **weighted average** of the intensities in a neighbourhood of (x, y).
- It is a set of weights that approximate the profile of a Gaussian function.
- It is very effective in reducing noise and also reducing details (image blurring)

Gaussian Filter





Non-linear Spatial Filters

Also called order-statistics filters- response based on ordering the pixels in the neighbourhood, and replacing centre pixel with the ranking result.

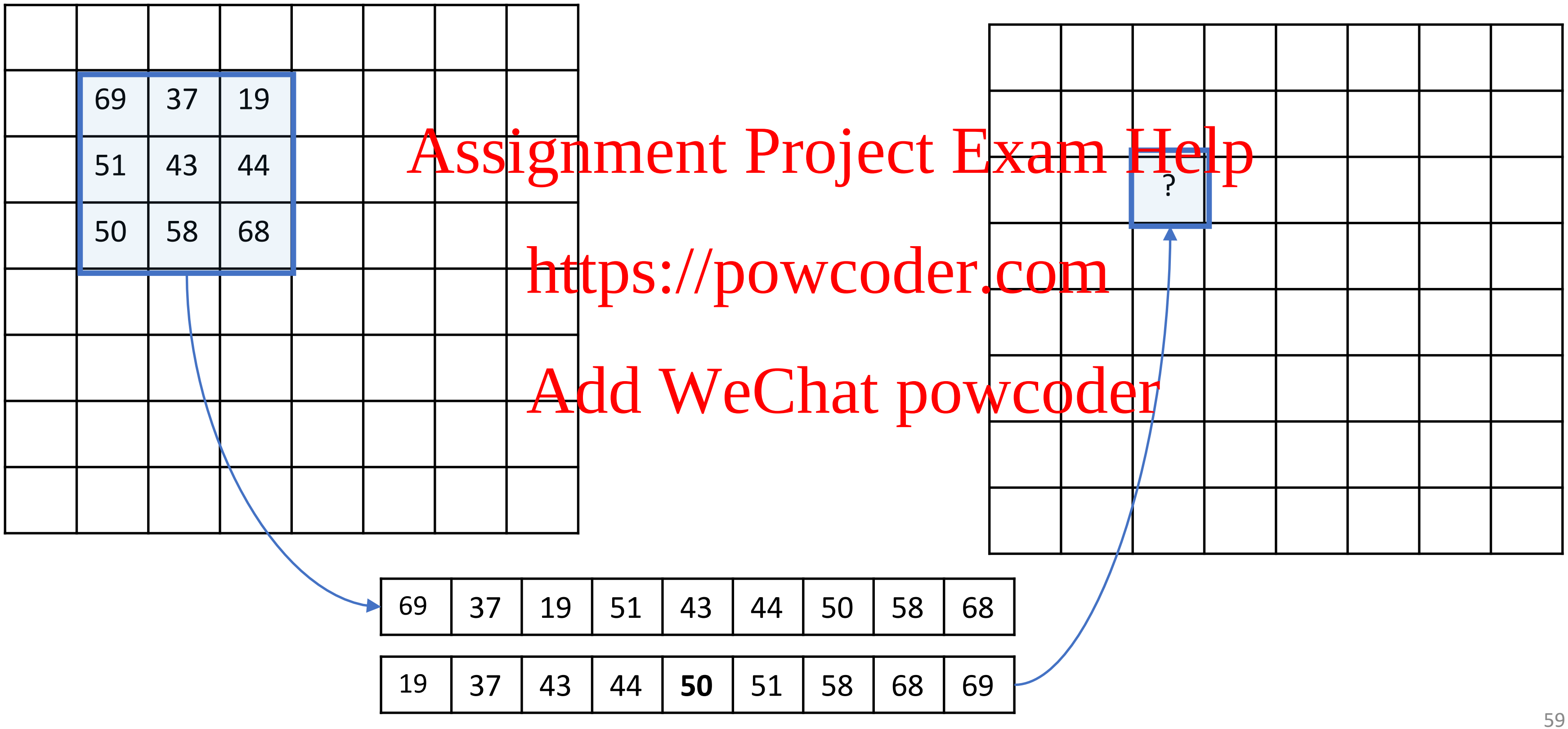
Median Filter

- intensity of each pixel is replaced by the *median* of the intensities in neighbourhood of that pixel

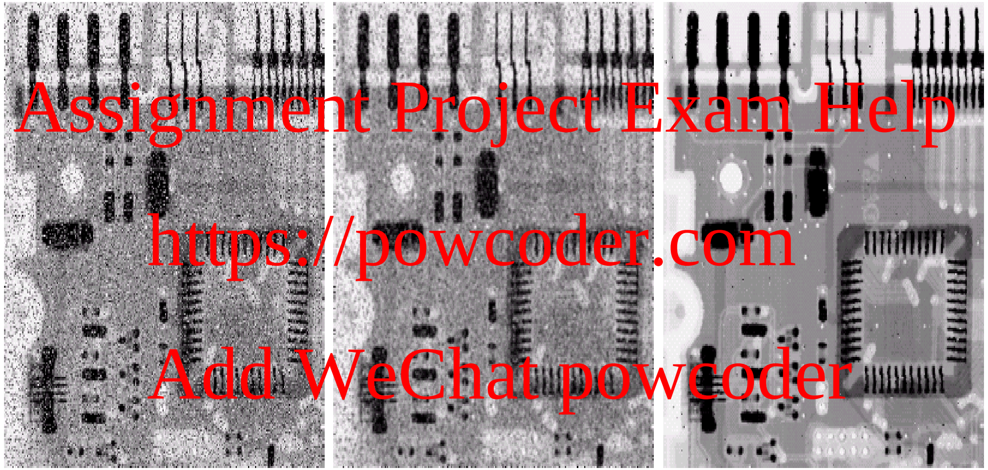
- Median M of a set of values is the middle value such that half the values in the set are less than M and the other half greater than M
- Median filtering forces points with distinct intensities to be more like their neighbours, thus eliminating isolated intensity spikes

- Also, isolated pixel clusters (light or dark), whose area is <= n^2/2, are eliminated by nxn median filter
- Good for impulse noise (salt-and-pepper noise)
- Other examples of order-statistics filters are max and min filters

Median Filter



Chapter 3
Image Enhancement in the
Spatial Domain



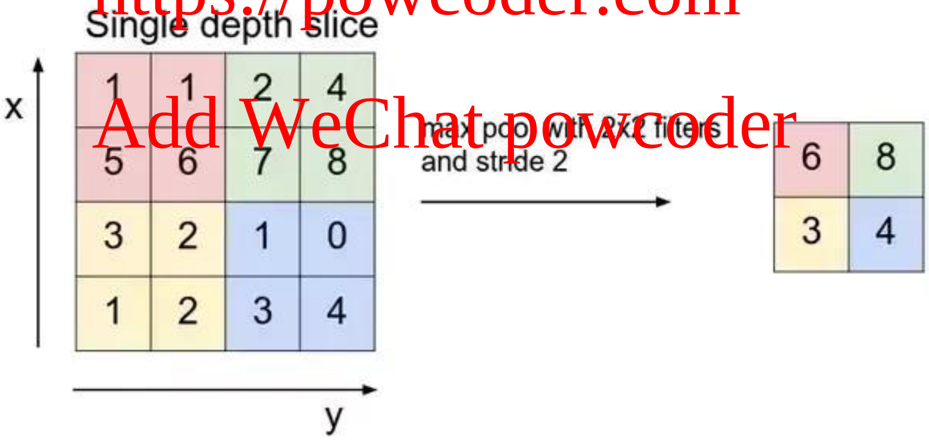
a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Pooling

Max / average/ median pooling

- Provides translation invariance
- Reduces computations
- Popular in deep convolutional neural networks (deep learning)



Sharpening Spatial Filters-Edge Detection

- Goal is to highlight fine details or enhance details that have been blurred

- Spatial differentiation is the tool-strength of response of derivative operator is proportional to degree of discontinuity of the image at the point where operator is applied

- Image differentiation enhances edges, and de-emphasizes slowly varying gray-level values.

Derivative definitions

- For 1-D function f(x), the first order derivative is approximated as:

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

- The second-order derivative is approximated as:

$$\frac{d^2 f}{dx^2} = f\,(x+1) + f\,(x-1) - 2\,f\,(x)$$

- These are partial derivatives, so that extension to 2D is easy.

a b
c

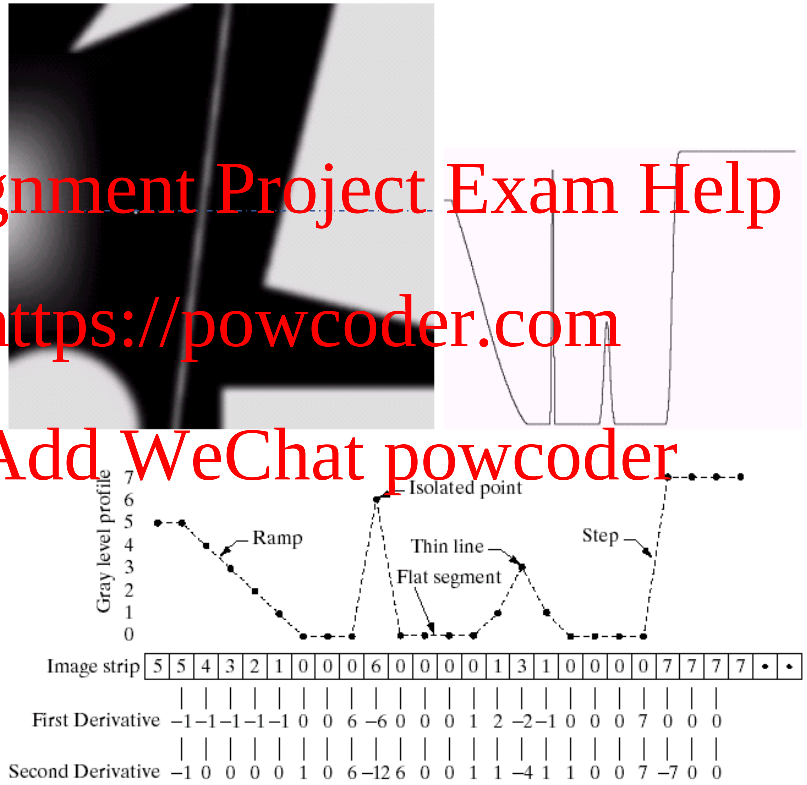


FIGURE 3.38 (a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).

Basic idea - Derivatives

- Horizontal scan of the image
- Edge modelled as a ramp- to represent blurring due to sampling
- First derivative is
  - Non-zero along ramp
  - zero in regions of constant intensity
  - constant during an intensity transition
- Second derivative is
  - Nonzero at onset and end of ramp
  - Stronger response at isolated noise point
  - zero everywhere except at onset and termination of intensity transition
- Thus, magnitude of first derivative can be used to detect the presence of an edge, and sign of second derivative to determine whether a pixel lies on dark or light side of an edge.

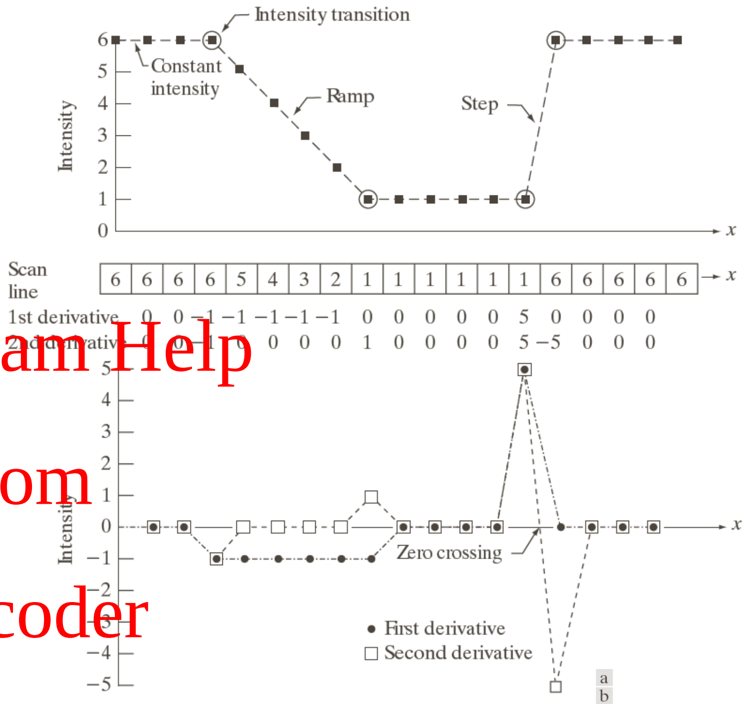


FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

Summary - Derivatives

- First-order derivatives produce thicker edges, have stronger response to gray-level step
- Second-order derivatives produce stronger response to fine detail (thin lines, isolated points), produce double response at step changes in gray level

Gradient Operator

First-order derivatives implemented using magnitude of the gradient

For function *f(x, y)*, the gradient of *f* at *(x, y)* is **G** with *x* and *y* components G_x , G_y

The magnitude of the gradient vector is

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2}$$

This is commonly approximated by: $G[f(x, y)] = |G_x| + |G_y|$

G_x and G_y are linear and may be obtained by using masks

We use numerical techniques to compute these- give rise to different masks, e.g. Roberts' 2x2 cross-gradient operators, Sobel's 3x3 masks

Chapter 3
Image Enhancement in the Spatial Domain

a
b c
d e

FIGURE 3.44 A 3 × 3 region of an image (the z's are gray-level values) and masks used to compute the gradient at point labeled z_5. All masks coefficients sum to zero, as expected of a derivative operator.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

The Laplacian

Second order derivatives based on the Laplacian.
For a function *f (x, y),* the Laplacian is defined by

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

This is a linear operator, as all derivative operators are.
In discrete form:

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

and similarly in *y* direction.

Summing them gives us

$$\Delta^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

Chapter 3
Image Enhancement in the Spatial Domain

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a	b
c	d

FIGURE 3.39 (a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

Laplacian ctd

- There are other forms of the Laplacian- can include diagonal directions, for example

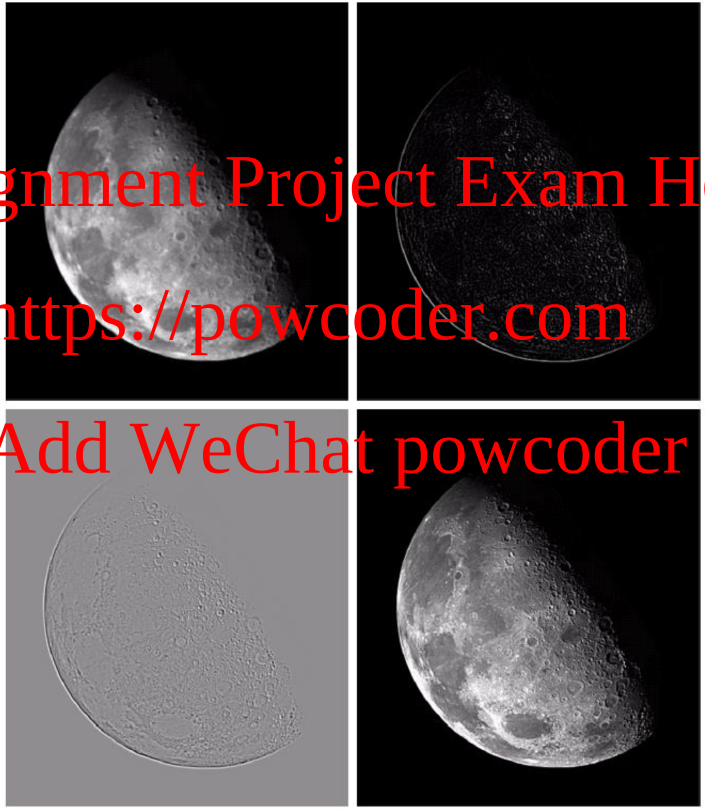
- Laplacian highlights grey-level discontinuities and produces dark featureless backgrounds

- The background can be recovered by adding or subtracting the Laplacian image to the original image

Chapter 3
Image Enhancement in the Spatial Domain

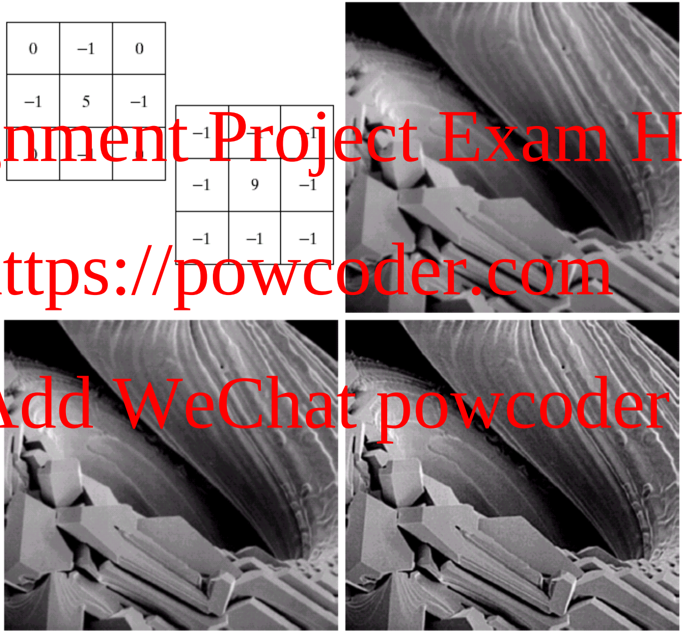
a b
c d

FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)



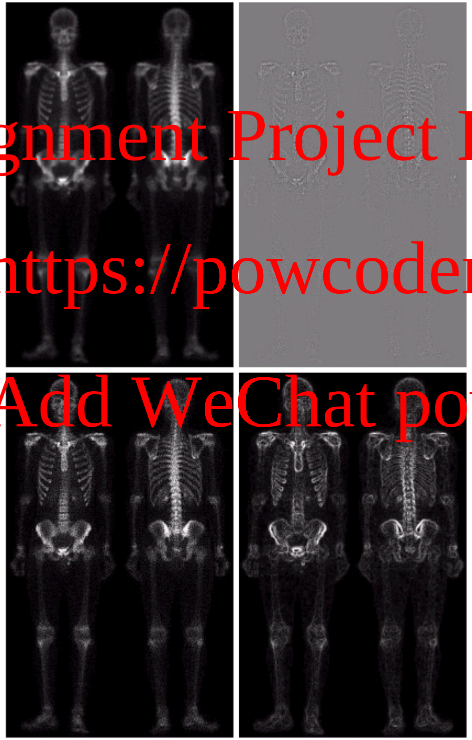
0	−1	0
−1	5	−1
0	−1	0

−1	−1	−1
−1	9	−1
−1	−1	−1



a b c
d e

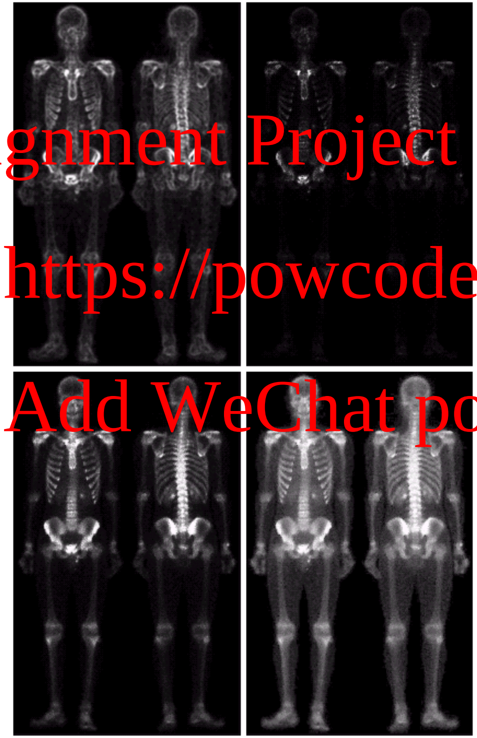
FIGURE 3.41 (a) Composite Laplacian mask. (b) A second composite mask. (c) Scanning electron microscope image. (d) and (e) Results of filtering with the masks in (a) and (b), respectively. Note how much sharper (e) is than (d). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)



a b
c d

FIGURE 3.46 (a) Image of whole body bone scan. (b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).

Chapter 3
Image Enhancement in the Spatial Domain



e f
g h

FIGURE 3.46 *(Continued)* (e) Sobel image smoothed with a 5 × 5 averaging filter. (f) Mask image formed by the product of (c) and (e). (g) Sharpened image obtained by the sum of (a) and (f). (h) Final result obtained by applying a power-law transformation to (g). Compare (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Padding

- When we use spatial filters for pixels on the boundary of an image, we do not have enough neighbours

- To get an image with the same size as input image

  - ***Zero***: set all pixels outside the source image to 0
  - ***Constant***: set all pixels outside the source image to a specified border value
  - ***Clamp***: repeat edge pixels indefinitely

  - ***Wrap***: copy pixels from opposite side of the image
  - ***Mirror***: reflect pixels across the image edge

Padding Example

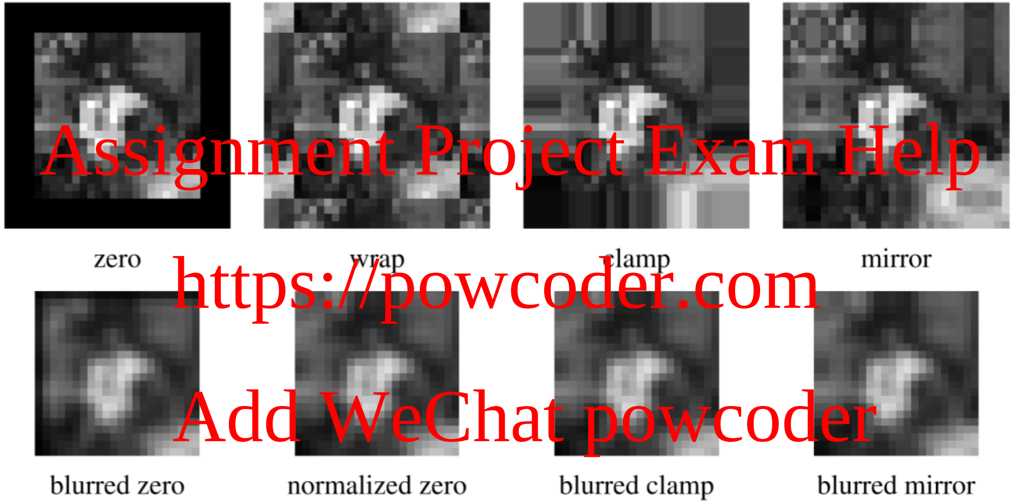


Figure 3.13 Border padding (top row) and the results of blurring the padded image (bottom row). The normalized zero image is the result of dividing (normalizing) the blurred zero-padded RGBA image by its corresponding soft alpha value.

Szeliski, “Computer Vision”, Chapter 3

References and acknowledgements

- Chapter 3 of Gonzalez and Woods 2002
- Sections 3.1-3.3 of Szeliski
- Some images drawn from above resources