Chapter 4
Enhancement

Fall 2020 Image processing

Outline

- Introduction
- 4.1 Why perform enhancement
- 4.2 Pixel neighborhoods
- 4.3 Filter kernels and the mechanics of linear filtering
- 4.3 Filtering for noise removal
- 4.4 Filtering for edge detection
- 4.5 Edge enhancement

Introduction

- Image filtering
  - pixel neighborhoods
  - noise removal filters
  - edge detection and edge
  - sharpening effects

4.1 Why perform enhancement?

- Image enhancement
  - Subjective
  - more suitable
  - Task specific

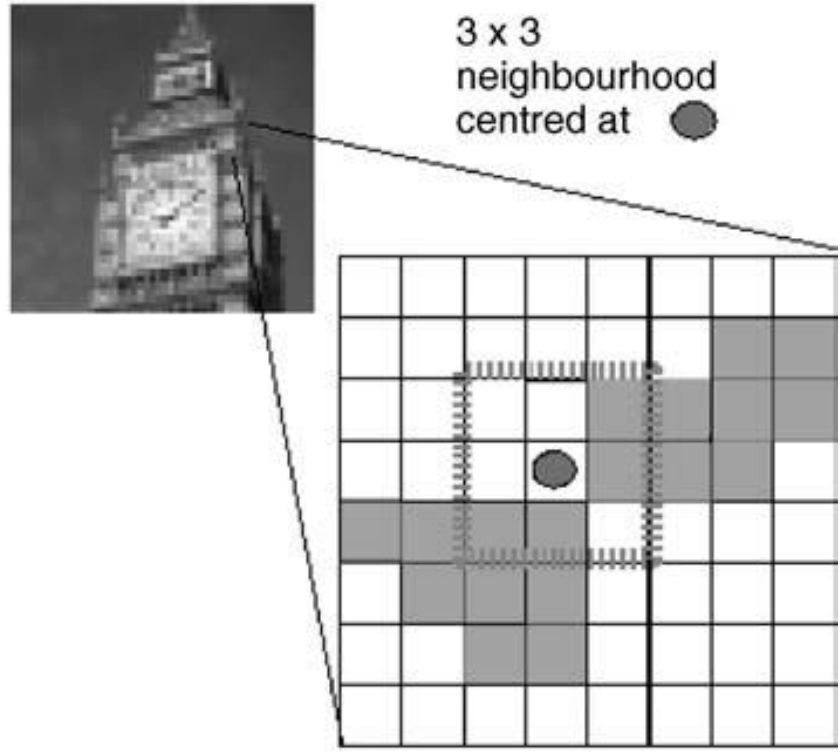
4.1 Why perform enhancement?

- spatial domain filtering
  - actual pixels of the image itself
  - the target pixel – is successively addressed
  - in a specified neighborhood around the target pixel

4.2 Pixel neighbourhoods

- Connectivity
  - 4-connectivity
    - N, W, E, S
  - 8-connectivity
    - N, NW, W, NE, SE, E, SW, S
    - default

NW	N	NE
W	(i,j)	E
SW	S	SE



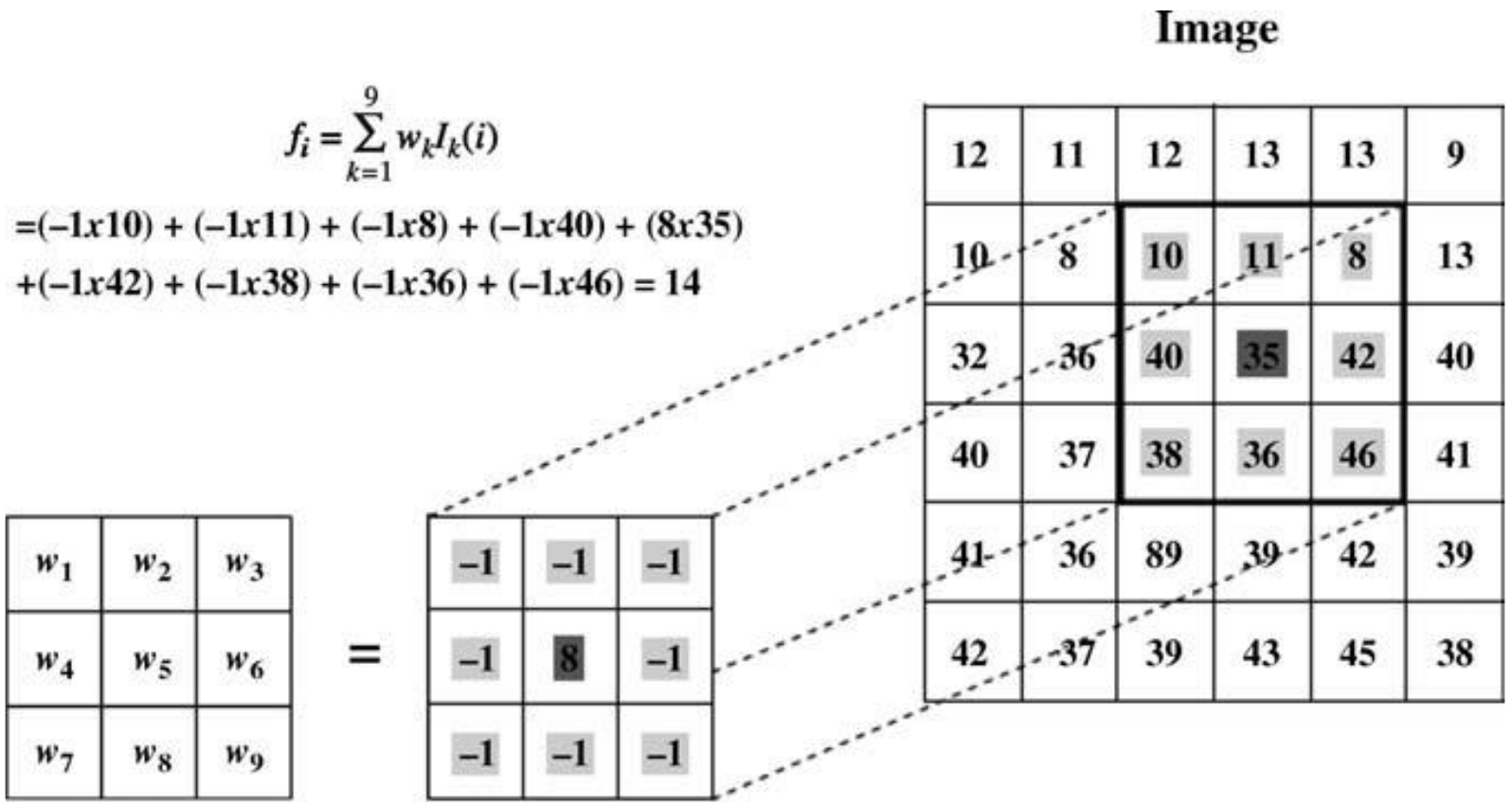
4.3 Filter kernels and the mechanics of linear filtering

- Filter kernel
  - array/subimage of exactly the same size as the neighborhood containing the weights that are to be assigned to each of the corresponding pixels in the neighborhood of the target pixel.

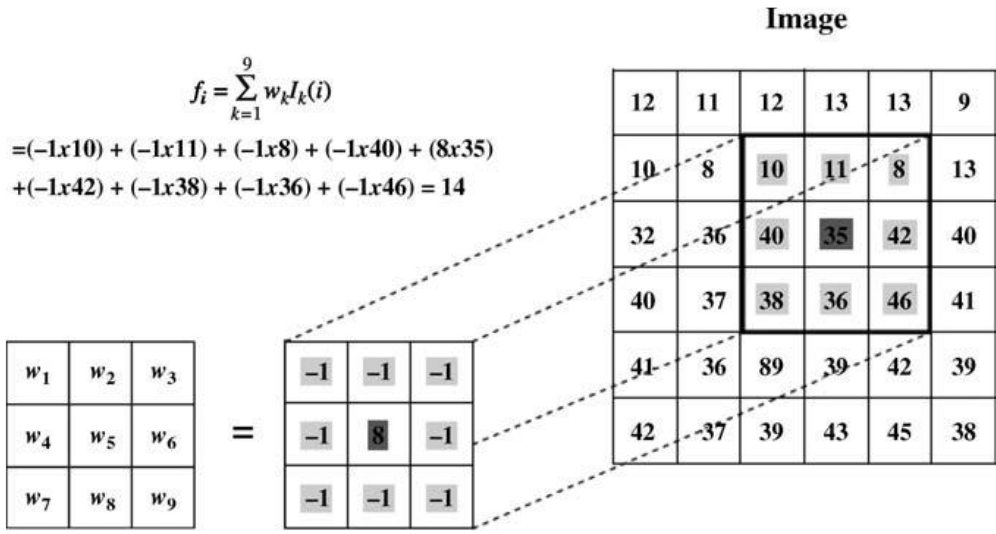
- Filtering
  - successively positioning the kernel so that the location of its center pixel coincides with the location of each target pixel, each time the filtered value being calculated by the chosen weighted combination of the neighborhood pixels.

  - in discrete forma process called convolution

4.3 Filter kernels and the mechanics of linear filtering



4.3 Filter kernels and the mechanics of linear filtering



- The steps in linear (convolution) filtering
  - (1) Define the filter kernel.
  - (2) Slide the kernel over the image so that its center pixel coincides with each (target) pixel in the image
  - (3) Multiply the pixels lying beneath the kernel by the corresponding values (weights) in the kernel above them and sum the total.
  - (4) Copy the resulting value to the same locations in a new (filtered) image.

4.3 Filter kernels and the mechanics of linear filtering

- when a target pixel lies close to the image boundary
  - (1) Simply leave unchanged those target pixels which are located within this boundary region.
  - (2) Perform filtering on only those pixels which lie within the boundary (and adjust the filter operation accordingly).
  - (3) ‘Fill in’ in the missing pixels within the filter operation by mirroring values over theboundary.

4.3 Filter kernels and the mechanics of linear filtering

- Nonlinear spatial filtering
  - nonlinear operation on the neighborhood pixels

$$f_i = \sum_{k=1}^{N} w_{k1} I_k^2(i) + w_{k2} I_k(i) + w_{k3}$$

  - order (or statistical) filters

4.4 Filtering for noise removal

- Noise
  - ‘salt and pepper’
  - Gaussian noise

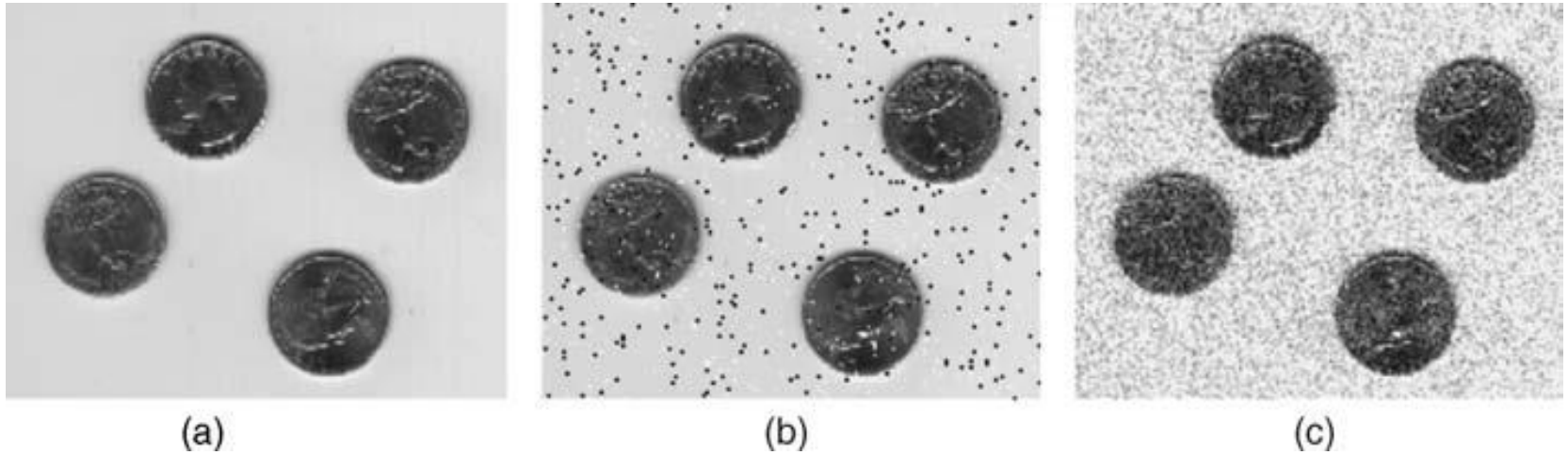


Figure 4.3 (a) Original image with (b) ‘salt and pepper’ noise and (c) Gaussian noise added

4.4 Filtering for noise removal

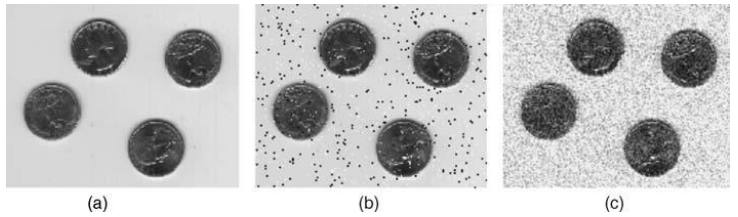


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- Mean filtering
  - giving equal weight to all pixels in the neighborhood.
  - suppress noise in an image
  - smooth the image

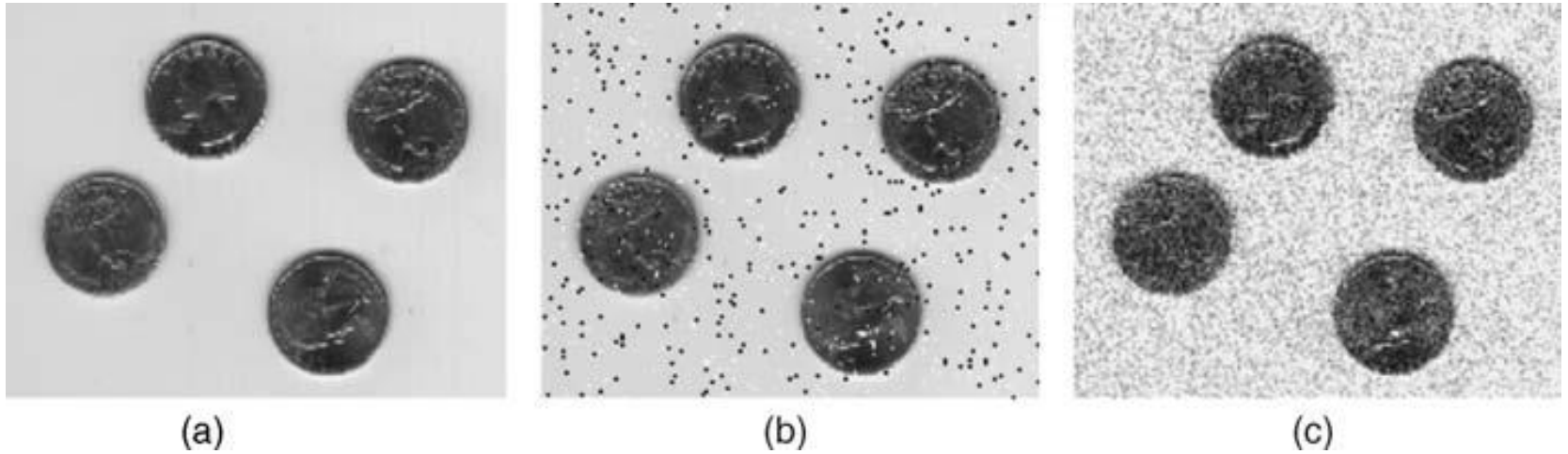


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

4.4 Filtering for noise removal

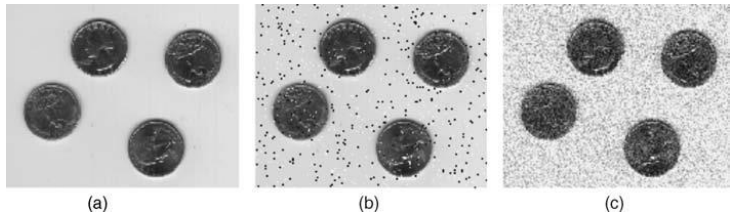


Figure 4.4 Mean filter (33) applied to the (a) original, (b) ‘salt and pepper’ noise and (c) Gaussian noise images of Figure 4.3

- Mean filtering
  - reasonably effective at removing the Gaussian
  - loss of high-frequency image data

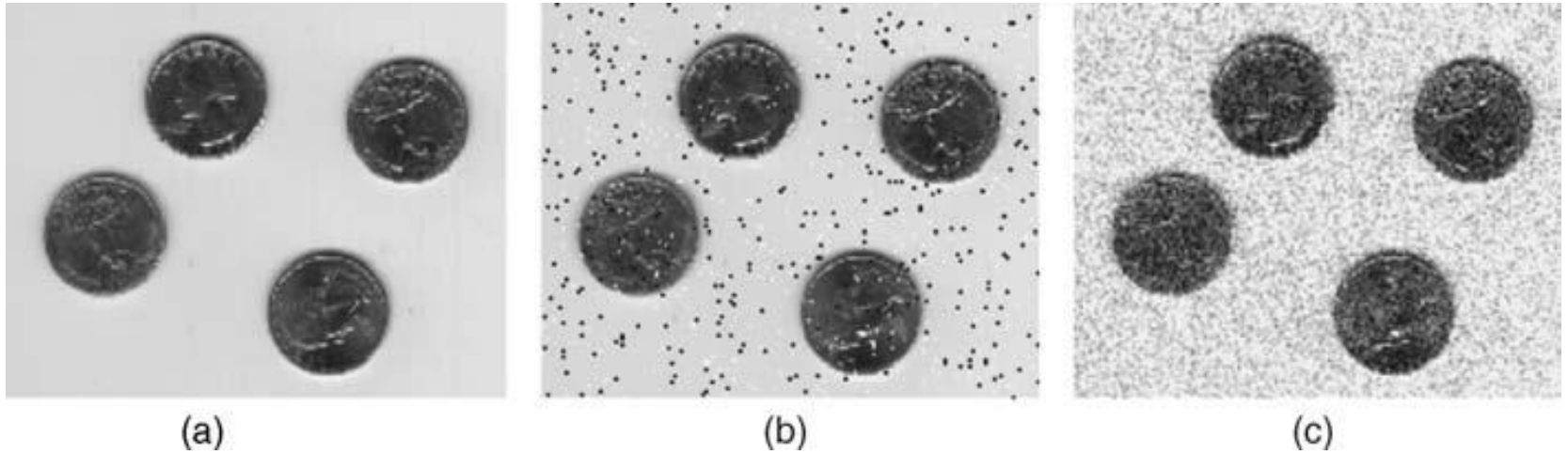


Figure 4.4 Mean filter (33) applied to the (a) original, (b) ‘salt and pepper’ noise and (c) Gaussian noise images of Figure 4.3

4.4 Filtering for noise removal

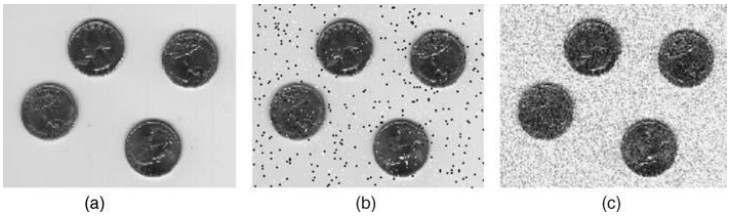


Figure 4.4 Mean filter (33) applied to the (a) original, (b) ‘salt and pepper’ noise and (c) Gaussian noise images of Figure 4.3

- Mean filtering
  - Larger kernel sizes will further suppress the Gaussian noise but will result in further degradation of image quality.
  - not effective for the removal of ‘salt and pepper’ noise

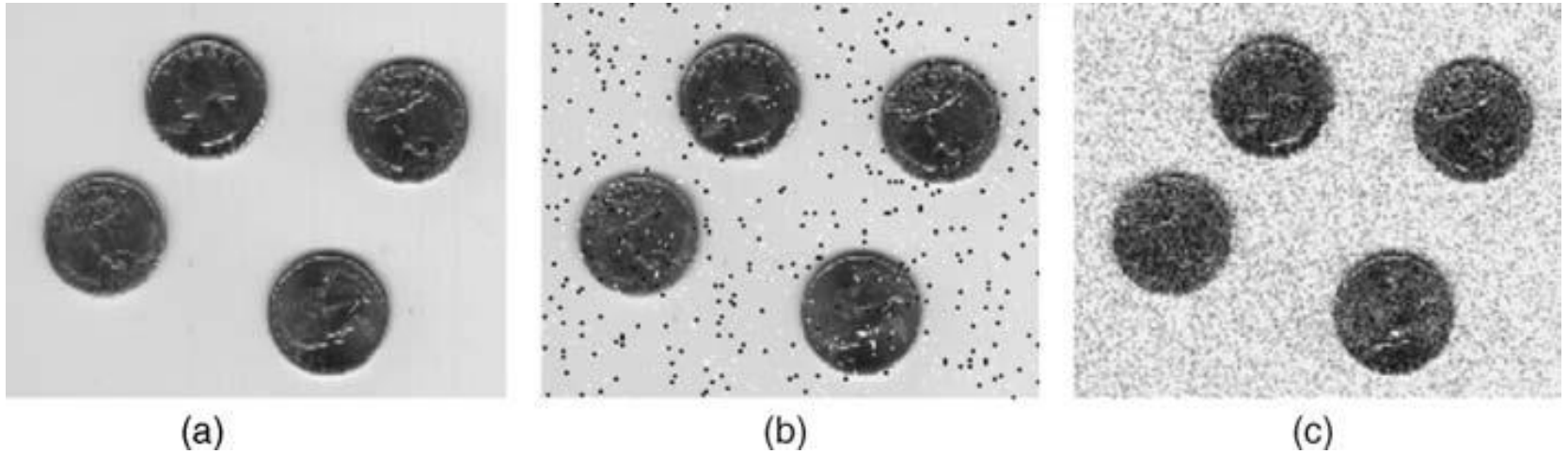


Figure 4.4 Mean filter (33) applied to the (a) original, (b) ‘salt and pepper’ noise and (c) Gaussian noise images of Figure 4.3

4.4 Filtering for noise removal

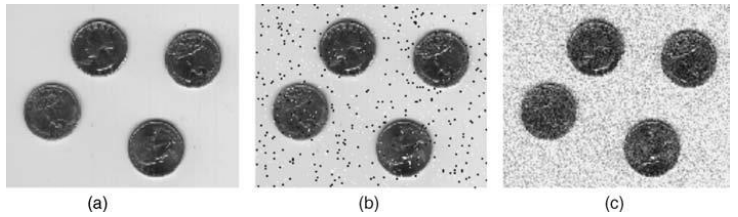


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- 4.4.2 Median filtering
  - overcomes the main limitations of the mean filter
  - greater computational cost
  - sharp high-frequency detail

Figure 4.5 Median filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

4.4 Filtering for noise removal

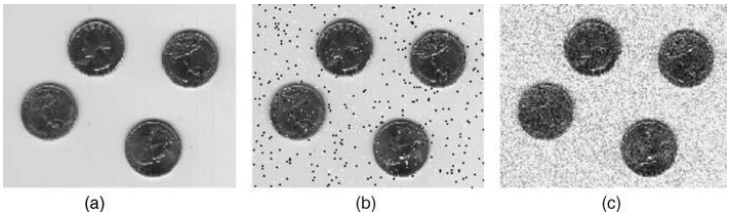


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- 4.4.2 Median filtering
  - removal of some Gaussian noise at the expense of a slight degradation in image quality
  - very good at removing 'salt and pepper

Figure 4.5 Median filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

4.4 Filtering for noise removal

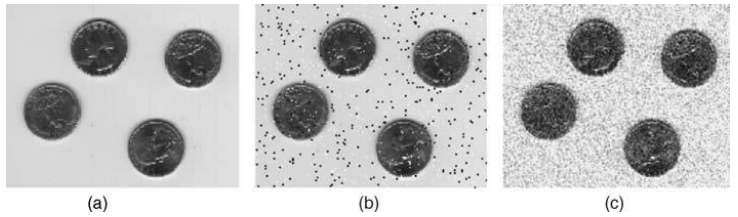


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- 4.4.3 Rank filtering
  - (1) Define the neighborhood of the target pixel (NxN).
  - (2) Rank them in ascending order (first is lowest value, (NxN)th is highest value).
  - (3) Choose the order of the filter (from 1 to N).
  - (4) Set the filtered value to be equal to the value of the chosen rank pixel.

4.4 Filtering for noise removal

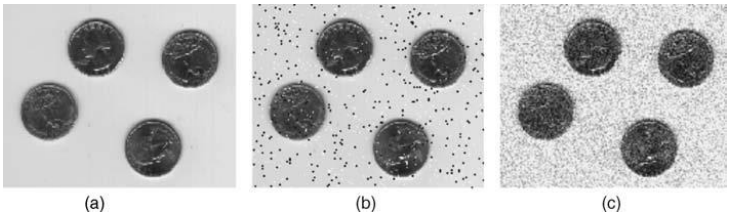


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- 4.4.3 Rank filtering
  - conservative smoothing

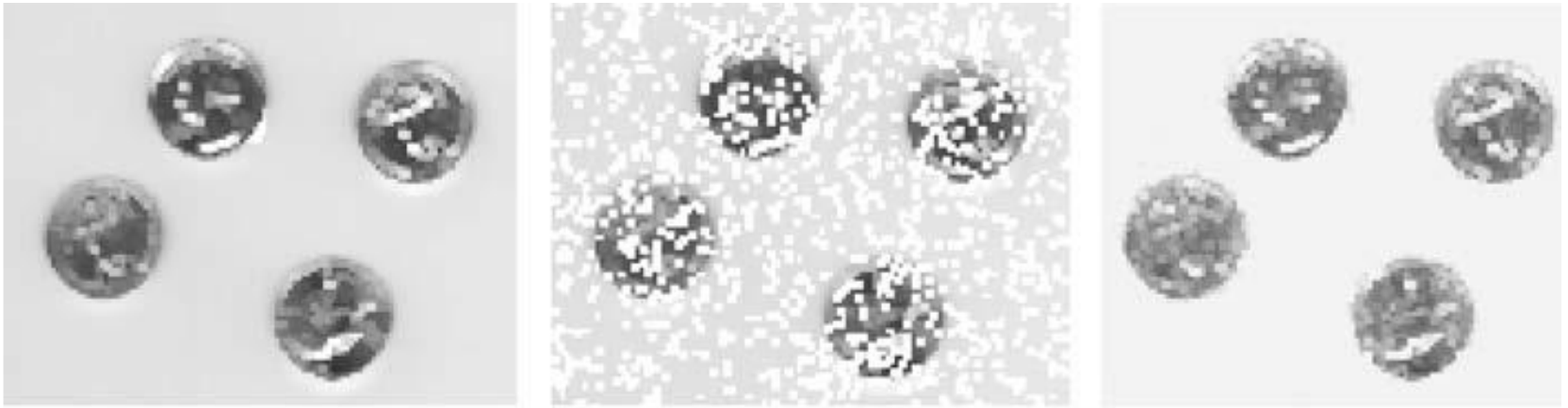


Figure 4.6 Order filtering (max, order ¼ 25, 55) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

4.4 Filtering for noise removal

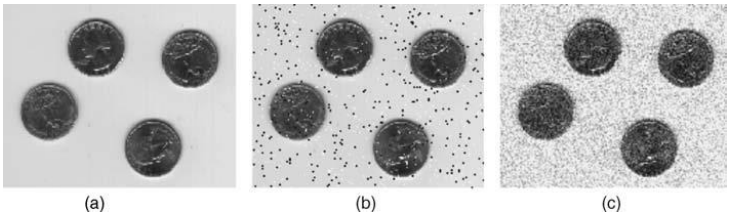


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- 4.4.4 Gaussian filtering
  - smoothing the image
  - two free parameters:
    - (1) the desired size of the kernel (as an NxN filter mask);
    - (2) the value of s, the standard deviation of the Gaussian function.
  - very convenient for the frequency-domain analysis of filters
  - With a large value of σ is an example of a so-called low-pass filter

4.4 Filtering for noise removal

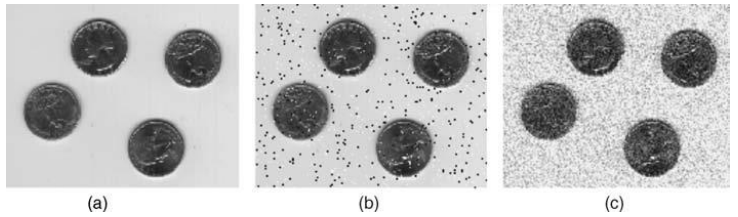


Figure 4.4 Mean filter (33) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

- 4.4.4 Gaussian filtering
  - degrades high frequency (edge) detail
  - removes to some degree the noise

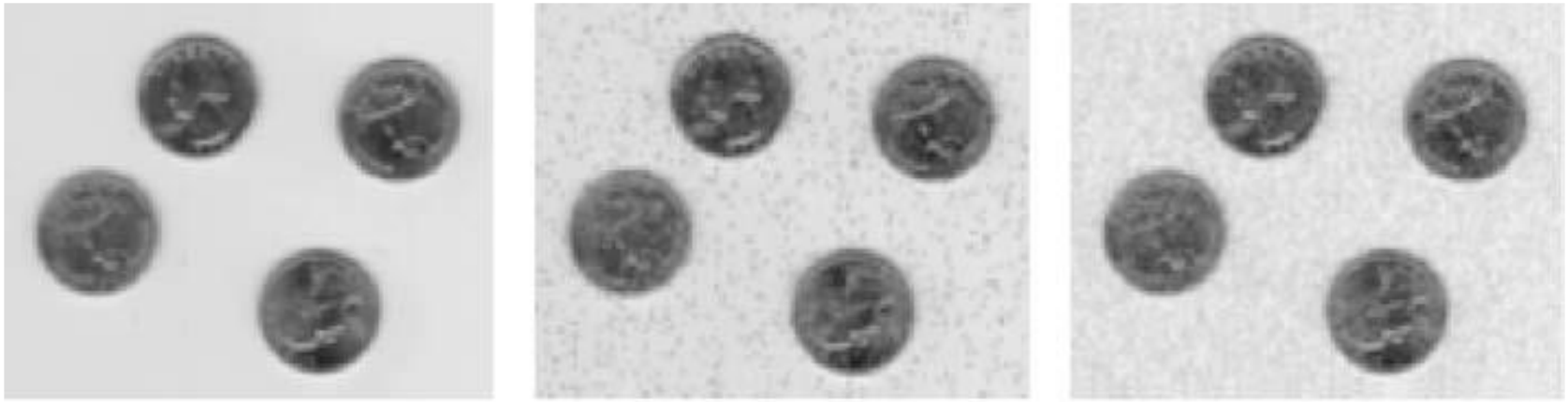


Figure 4.8 Gaussian filtering (5x5 with s¼2) applied to the (a) original, (b) 'salt and pepper' noise and (c) Gaussian noise images of Figure 4.3

4.5 Filtering for edge detection

- Main uses of image filtering
  - noise removal
  - Feature extraction
  - feature enhancement
- Detection of edges
  - discontinuity or gradient

4.5 Filtering for edge detection

- 4.5.1 Derivative filters for discontinuities noise removal
    - visual cortex contains a complex of feature detectors that are tuned to the edges and segments of various widths and orientations
    - Enhancing (or amplifying) the presence of these discontinuities in the image allows us to improve the perceived image quality under certain conditions
    - Sensitive to noise
    - Edge detection makes use of differential operators to detect changes in the gradients of the grey or color levels in the image
    - generally assisted by an initial stage of (most often Gaussian) smoothing to suppress noise

4.5 Filtering for edge detection

- 4.5.1 Derivative filters for discontinuities noise removal
  - first-order edge detection
  - second-order edge detection

Table 4.1 Derivative operators: their formal (continuous) definitions and corresponding discrete approximations

2 D derivative measure	Continuous case	Discrete case
$\frac{\partial f}{\partial x}$	$\lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$	$f(x+1,y) - f(x,y)$
$\frac{\partial f}{\partial y}$	$\lim_{\Delta y \to 0} \frac{f(x,y+\Delta y) - f(x,y)}{\Delta y}$	$f(x,y+1) - f(x,y)$
$\nabla f(x,y)$	$\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$	$[f(x+1,y) - f(x,y), f(x,y+1) - f(x,y)]$
$\frac{\partial^2 f}{\partial x^2}$	$\lim_{\Delta x \to 0} \frac{(\partial f/\partial x)(x+\Delta x, y) - (\partial f/\partial x) f(x,y)}{\Delta x}$	$f(x+1,y) - 2f(x,y) + f(x-1,y)$
$\frac{\partial^2 f}{\partial y^2}$	$\lim_{\Delta y \to 0} \frac{(\partial f/\partial x)(x,y+\Delta y) - (\partial f/\partial x)(x,y)}{\Delta y}$	$f(x,y+1) - 2f(x,y) + f(x,y-1)$
$\nabla^2 f(x,y)$	$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$	$f(x+1,y) + f(x-1,y) - 4f(x,y) + f(x,y+1) + f(x,y-1)$

4.5 Filtering for edge detection

- 4.5.2 First-order edge detection
  - Roberts operators

$$|G| = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) + \frac{1}{4}\pi$$

Roberts

0	–1
1	0

–1	0
0	1

Prewitt

X derivative

1	0	–1
1	0	–1
1	0	–1

Y derivative

1	1	1
0	0	0
–1	–1	–1

Sobel

1	0	–1
2	0	–2
1	0	–1

1	2	1
0	0	0
–1	–2	–1

4.5 Filtering for edge detection

- 4.5.2 First-order edge detection
  - Roberts operators
  - Prewitt/Sobel kernels
    - is not shifted by half a pixel
    - extension to larger sizes

Roberts

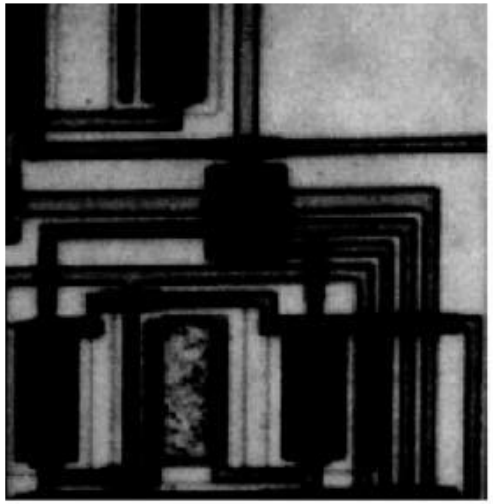
0	–1
1	0

–1	0
0	1

	Prewitt	Sobel
X derivative	1 0 –1 1 0 –1 1 0 –1	1 0 –1 2 0 –2 1 0 –1
Y derivative	1 1 1 0 0 0 –1 –1 –1	1 2 1 0 0 0 –1 –2 –1

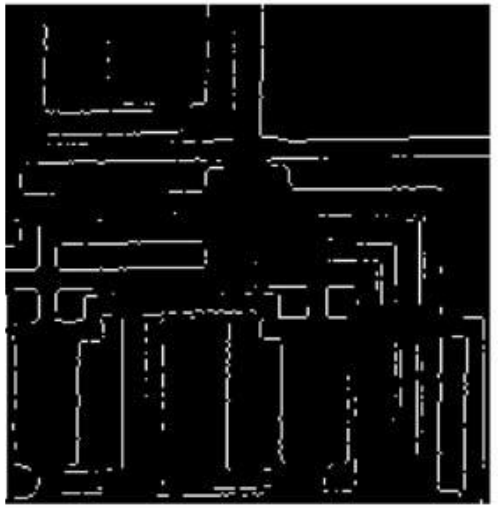
4.5 Filtering for edge detection

- 4.5.2 First-order edge detection
  - Roberts operators
    - notably susceptible to image noise
  - Prewitt/Sobel kernels
    - is not shifted by half a pixel
    - extension to larger sizes

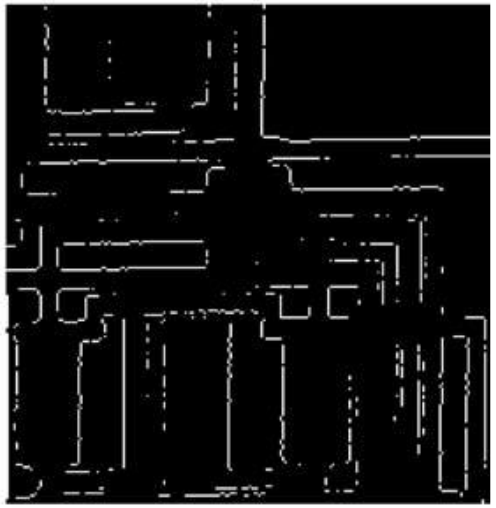


Original Image

Roberts Filter Edges



Prewitt Filter Edges



Sobel Filter Edges

4.5 Filtering for edge detection

- 4.5.2 First-order edge detection
  - Roberts operators
    - notably susceptible to image noise
  - Prewitt/Sobel kernels
    - is not shifted by half a pixel
    - extension to larger sizes
    - linearly separable filters
      - 2N operations as opposed to order N2 non-separable

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

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

4.5 Filtering for edge detection

- 4.5.3 Second-order edge detection
  - Laplacian edge detection

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

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

4.5 Filtering for edge detection

- 4.5.3 Second-order edge detection
  - Laplacian edge detection
    - points in the image where the local gradient changes most rapidly

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

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

4.5 Filtering for edge detection

- 4.5.3 Second-order edge detection
  - Laplacian edge detection
    - points in the image where the local gradient changes most rapidly
    - insensitivity to features lying in approximately diagonal directions

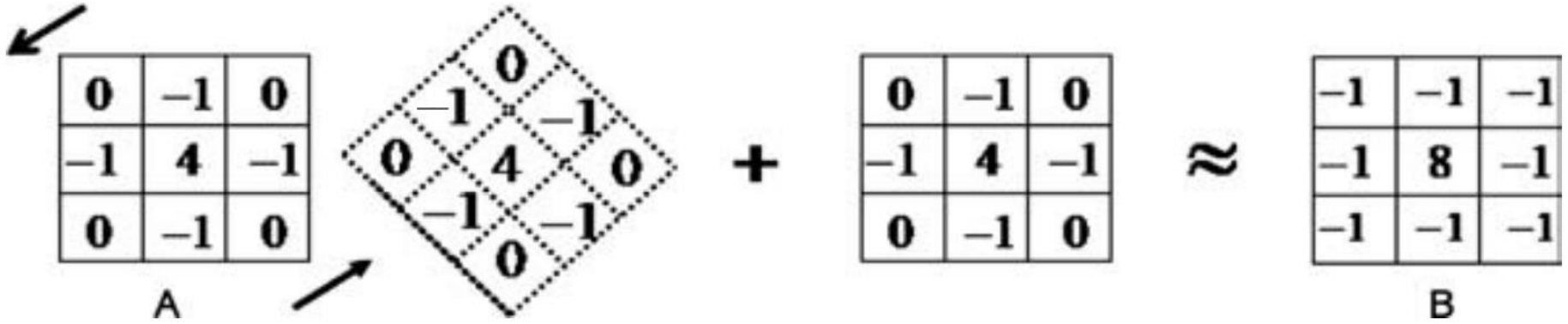


Figure 4.11 Construction of the Laplacian discrete kernel

4.5 Filtering for edge detection

- 4.5.3 Second-order edge detection
  - Laplacian edge detection
    - points in the image where the local gradient changes most rapidly
    - produce finer edges
    - approximate a second derivative
      - very sensitive to noise
        - Laplacian of Gaussian (LoG) filter

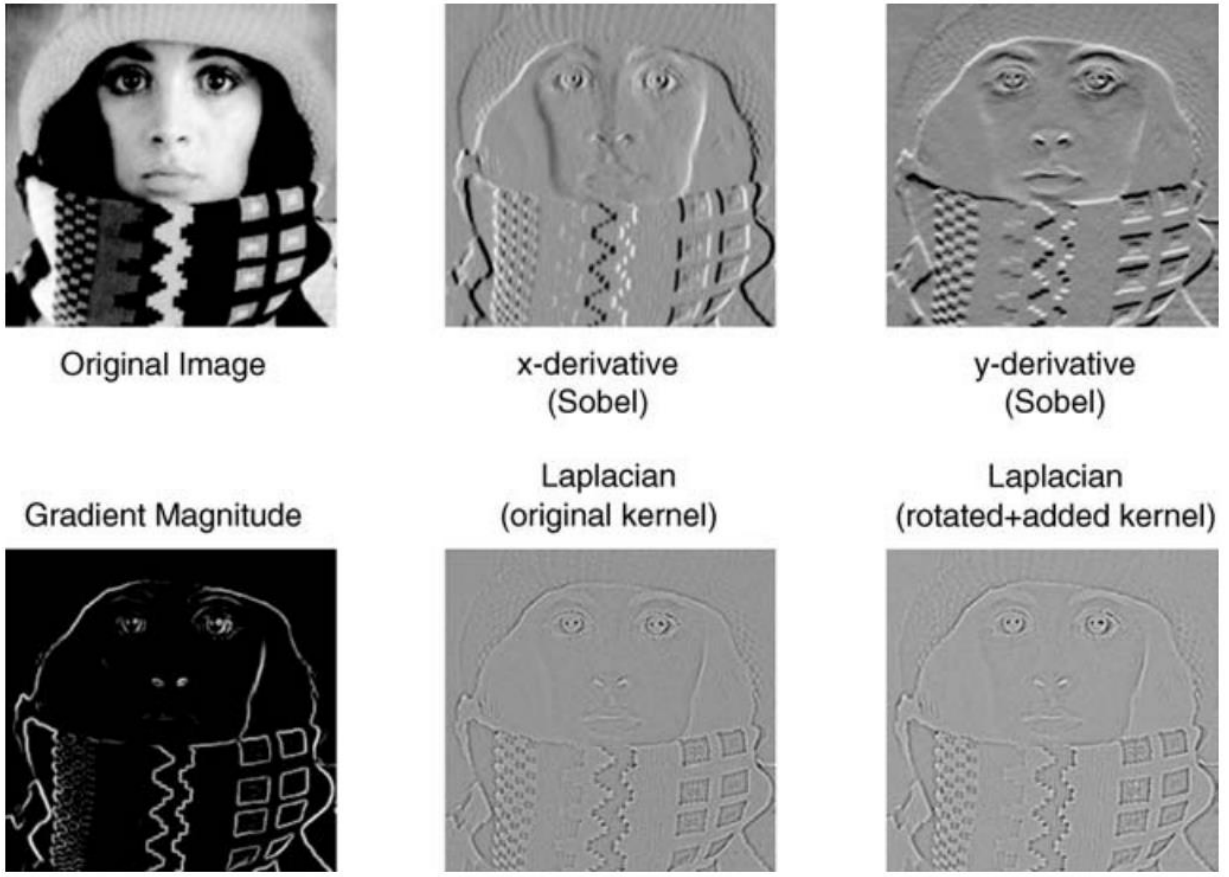


Figure 4.12 Comparison of first-order derivative (Sobel) and second-order (Laplacian) filters

4.5 Filtering for edge detection

- 4.5.3 Second-order edge detection
  - Zero-crossing detector
    - locate pixels at which the value of the Laplacian passes through zero
    - single-pixel thickness lines

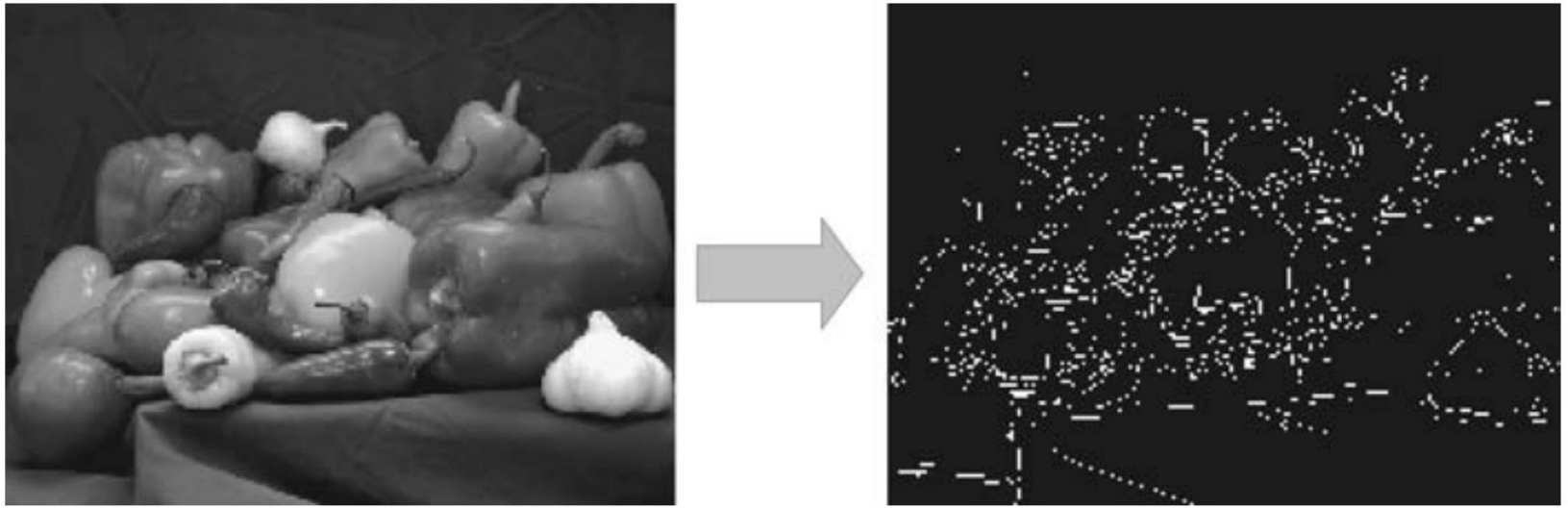


Figure 4.13 Edges detected as the zero crossings of the LOG operator

4.5 Filtering for edge detection

- 4.5.3 Second-order edge detection
  - Zero-crossing detector
    - in the form of closed curves
    - quite susceptible to noise

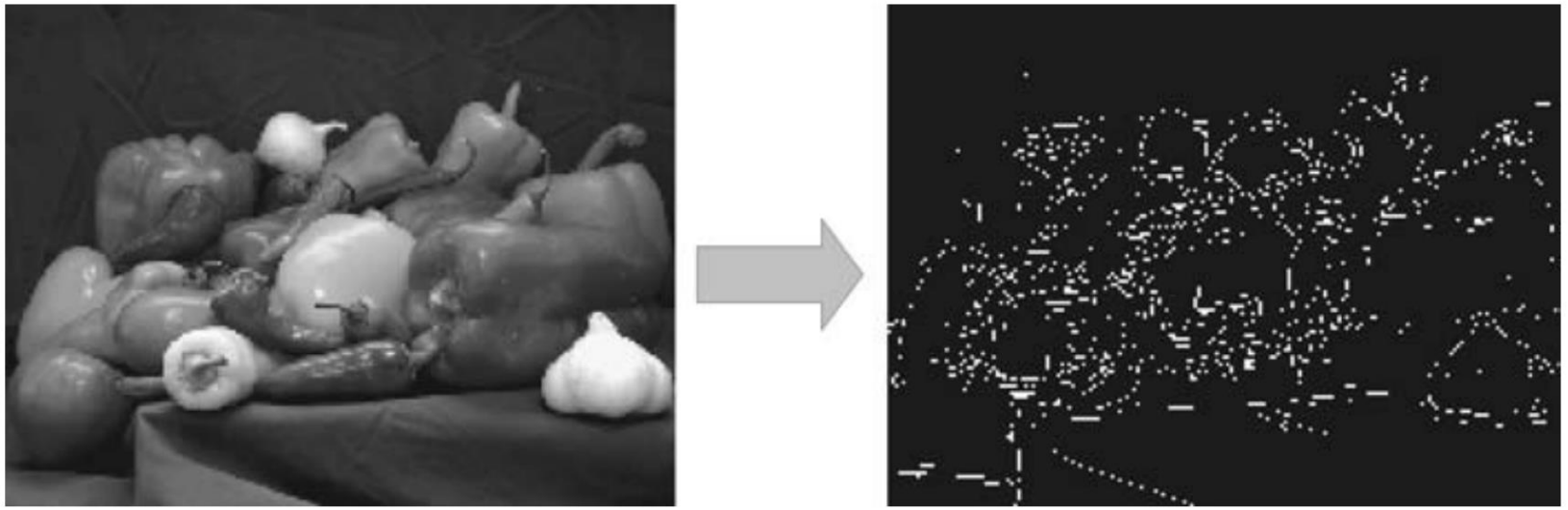


Figure 4.13 Edges detected as the zero crossings of the LOG operator

4.6 Edge enhancement

- 4.6.1 Laplacian edge sharpening
  - subtract Laplacian from the original

$$I_{\text{output}}(x, y) = I_{\text{in}}(x, y) - \nabla^2 I_{\text{in}}(x, y)$$

4.6 Edge enhancement

- 4.6.1 Laplacian edge sharpening
  - subtract Laplacian from the original
    - enhancement of edge contrast
    - an increase in image noise

Figure 4.14 Edge sharpening using the Laplacian operator

4.6 Edge enhancement

- 4.6.1 Laplacian edge sharpening
  - subtract Laplacian from the original
    - enhancement of edge contrast
    - an increase in image noise

Figure 4.14 Edge sharpening using the Laplacian operator

Figure 4.15 Edge sharpening using the LoG operator

4.6 Edge enhancement

- 4.6.2 The unsharp mask filter
  - also known as boost filtering

$$I_{\text{edges}}(c,r) = I_{\text{original}}(c,r) - I_{\text{smoothed}}(c,r)$$

$$I_{\text{enhanced}}(c,r) = I_{\text{original}}(c,r) + k(I_{\text{edges}}(c,r))$$

4.6 Edge enhancement

- 4.6.2 The unsharp mask filter
  - Also known as boost filtering
  - relatively unchanging regions of the original image will not be changed significantly
  - Image in which the intensity changes rapidly will be affected significantly.

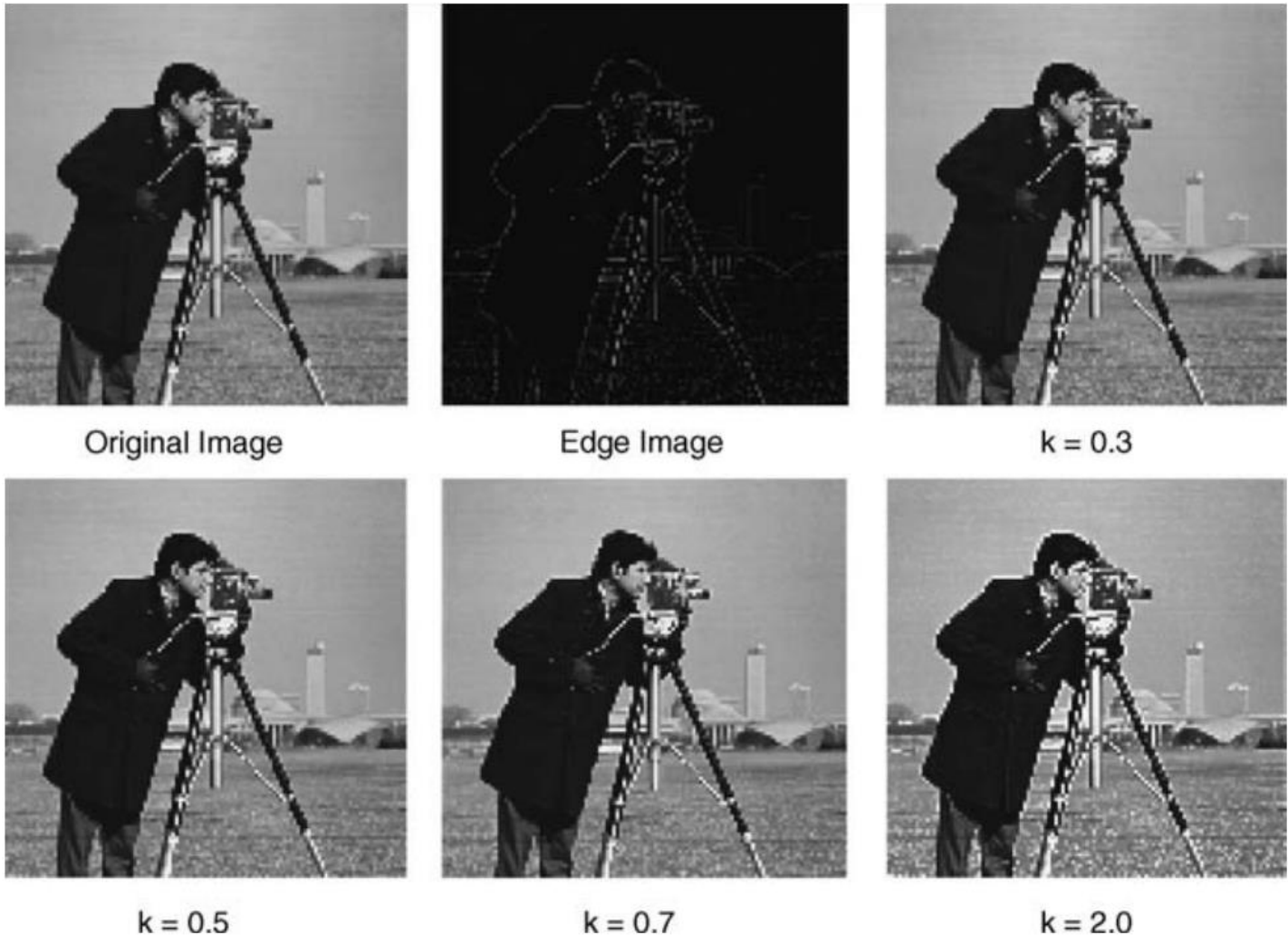


Figure 4.16 Edge sharpening using unsharp mask filter