EE4208

Engineering Design Control Option:

Design Of Edge Detectors

Dr Chua Chin Seng
Location: S1-B1c-84
Telephone: 6790-5412
Email: ecschua@ntu.edu.sg

Schedule

Lectures in Tutorial Room

- Week 1 -- Wang Han
- Week 2 -- Wang Han
- Week 3 -- Wang Han
- Week 4 -- C.S. Chua
- Week 5 -- C.S. Chua
- Week 6 -- C.S. Chua

Project Work

- Week 7 -- Wang Han
- Week 8 -- Wang Han
- Week 9 -- Wang Han
- Week 10 -- C.S. Chua
- Week 11 -- C.S. Chua
- Week 12 -- C.S. Chua

Assessment

- Week 13 -- Wang Han / C.S. Chua

What is Edge Detection

- Definition:

  Edge point, or edges, are image pixels which exhibit a sharp variation of intensity (or colour RGB) values in the neighbourhood of that pixel.

- Example:

  The figure (next page) shows an intensity image and the intensity profile along a particular scanline.

- Note:
  - Edge contours are the objective of edge detectors. Each pixel on the edge contour is found by extracting the pixel of **maximum change** from the scanline profile (more specifically, along the direction orthogonal to the edge direction).
  - Image noise causes significant intensity variations, giving rise to potential spurious edges.

Figure 1a. Intensity Image

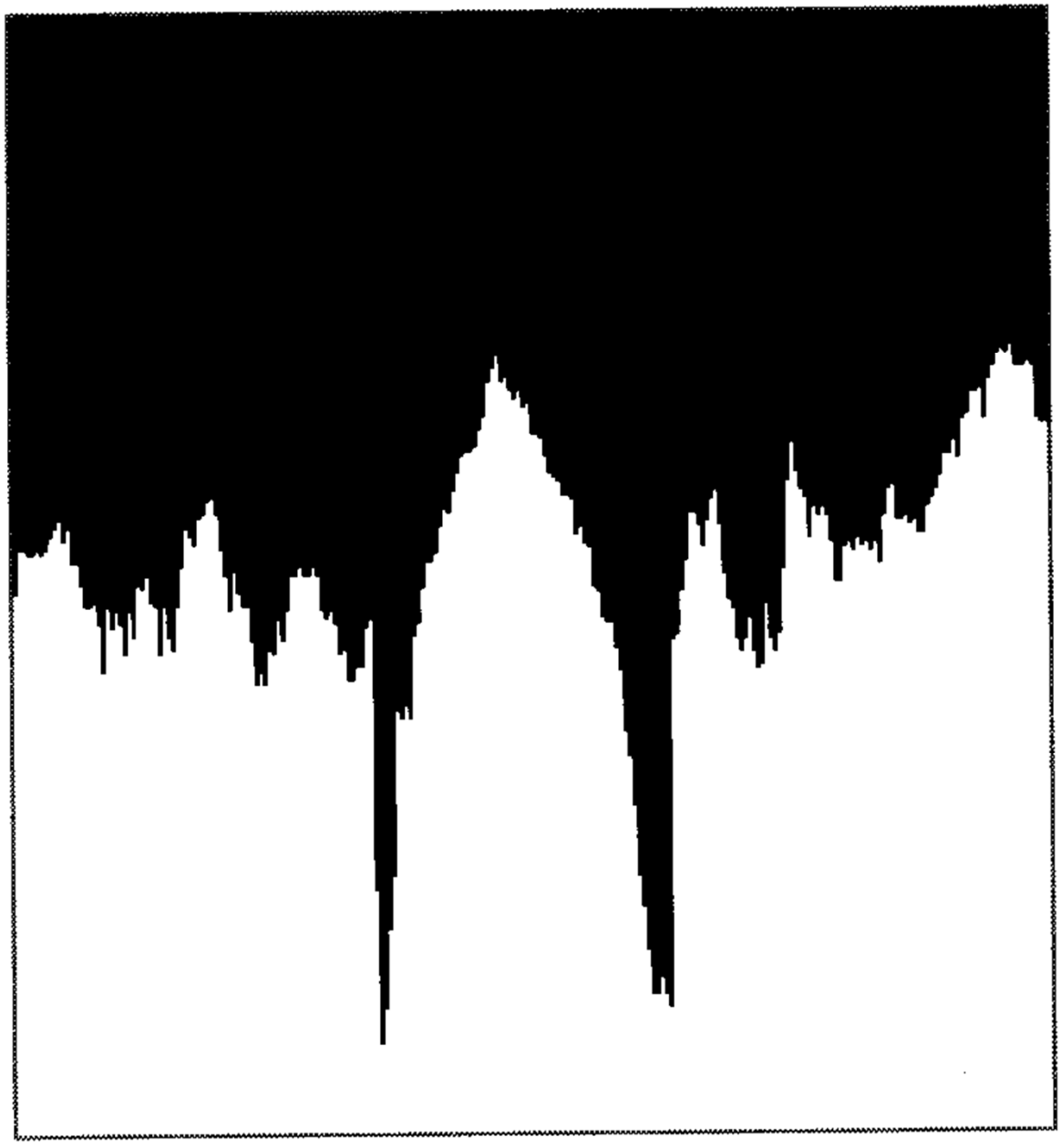


Figure 1b. Intensity values along scanline of Figure 1a.

Three Steps Of Edge Detection

- **Noise Smoothing**

  Suppress as much of the image noise as possible, without destroying the true edges. In the absence of specific information, assume that the noise is Gaussian and white.

- **Edge Enhancement**

  Design a filter responding to edges; that is, the filter's output is large at edge pixels and low elsewhere, so that edges can be located as the local maxima in the filter's output.

- **Edge Localization**

  Decide which local maxima in the filter's output are edges and which are just caused by noise. This involves:
  - thinning wide edges to 1-pixel width (non-maximum suppression)
  - establishing the minimum value to declare a local maxima an edge (thresholding)

Types of Edges To Detect

- Edges can be modelled according to their intensity profiles. For most practical purposes, a few models are sufficient to cover all interesting edges. (See Figure 2.)
- You should think of the 1-D signals in Figure 2 as cross-sections of 2-D images along a particular orientation (not necessarily a row or column). This orientation is defined as orthogonal to the edge direction.

- **Step Edges:**

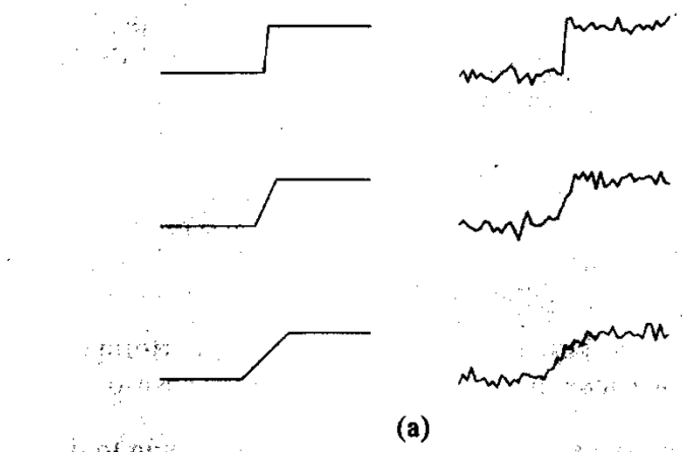
  The most common type in intensity images. They occur typically at contours of image regions of different intensityies. In most cases, the transition occurs over several pixels, not just one, which gives rise to **ramp edges**. Sharp variation of intensity values.

- **Ridge Edges:**

  Generated by thin lines. Wide ridge edges (generated by thick lines) can be modelled by two step edges. Sharp variation of intensity values.

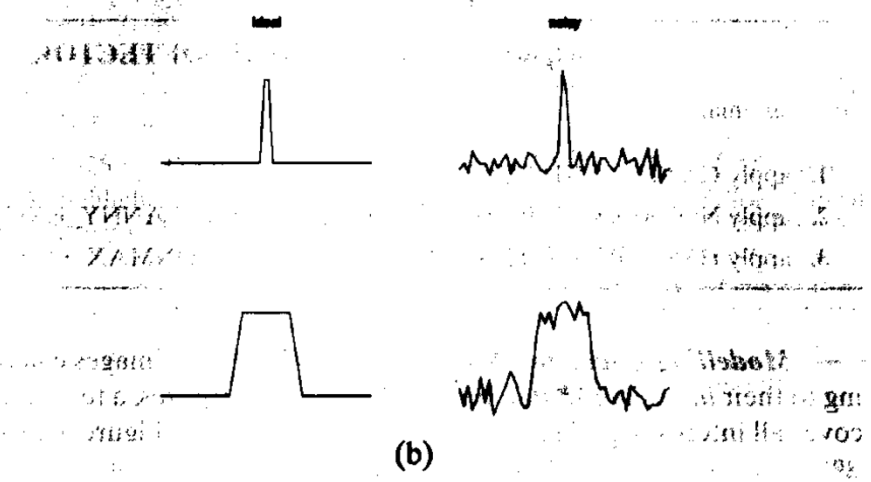
- **Roof Edges:**

  Relatively rare, but may appear along the intersection of surfaces. Sharp variation to derivatives of intensity values.



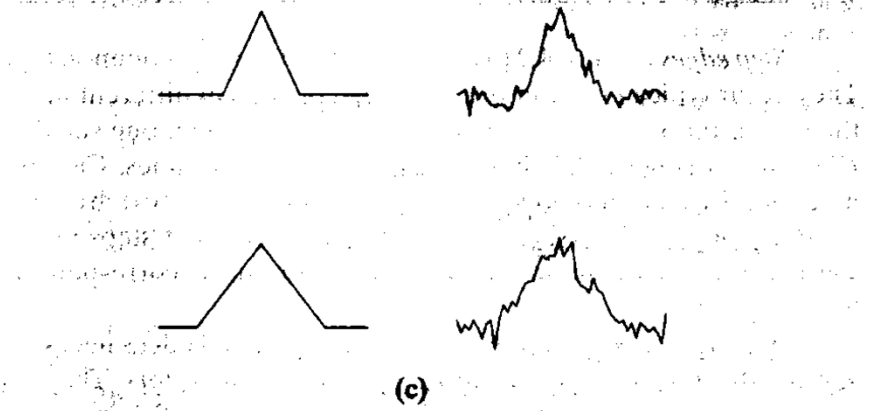
(a)

Left: ideal step (top, transition occurs over one pixel) and ramp edges. Right: corresponding noisy version, obtained by adding Gaussian noise (standard deviation 5% of the step height).



(b)

Left: ideal ridge edges. Right: corresponding noisy version, obtained as for step edges.



(c)

Left: ideal roof edges. Right: corresponding noisy version, obtained as for step edges.

Figure 2. Three Types of 1-D Edge Profiles

Definition Of Edge Terms

- **Edge Normal (or Gradient Direction):**

  The direction (unit vector) of the maximum intensity variation at the edge point. This identifies the direction perpendicular to the edge.

- **Edge Direction:**

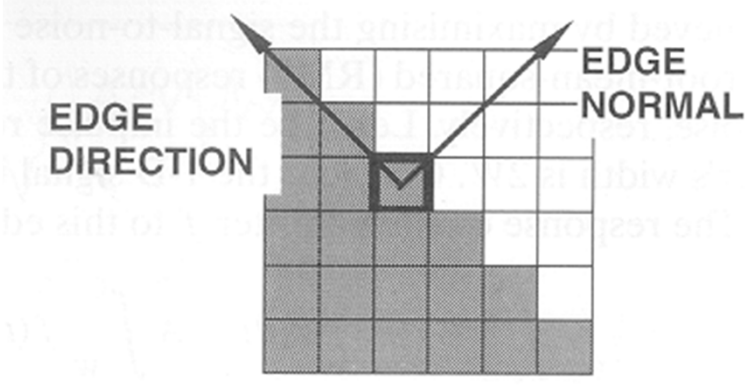
  Direction tangent to the contour of the edge.

- **Edge Position or Center:**

  Image pixel position at which the edge is located. This is usually saved in a binary image (1 for edge point, 0 for non-edge point).

- **Edge Strength:**

  Measure of the local image contrast across the edge.



Fundamental Concept: Convolution Masks

- A neighbourhood-based operation in which the output image is obtained by convolving the input with a mask, $w(u, v)$ (also called a kernel/operator/window) having a window size of $m \times n$. Usually $m = n$.
- The mask is centered at each pixel under consideration and shifted from the top left pixel to the bottom right pixel of the image
- Mathematically, the convolution for each pixel is given by

$$g(x, y) = \sum_{u=-m}^{m} \sum_{v=-n}^{n} I\left(x+u, y+v\right) w(u, v)$$

- Basic approach is to sum products between mask coefficients and associated pixel intensities.
- For example:

Mask Values

$$\begin{matrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{matrix}$$

Image Values $f(x, y)$

$$\begin{matrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} & z_{16} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} & z_{26} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} & z_{36} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} & z_{46} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} & z_{56} \\ z_{61} & z_{62} & z_{63} & z_{64} & z_{65} & z_{66} \end{matrix}$$

- With center of mask at position z_{22}, the resultant value
  $G(x, y) = y_{22} = w_1z_{11} + w_2z_{12} + w_3z_{13} + w_4z_{21} + w_5z_{22} + w_6z_{23} + w_7z_{31} + w_8z_{32} + w_9z_{33}$

- Move the mask to next pixel position z_{23} and repeat.

Different Masks

- We can choose different mask coefficients $w(u, v)$ to achieve different objectives in the output image.
  E.g.

1 1 1	–1 0 1
1 1 1	–1 0 1
1 1 1	–1 0 1
for image smoothing	for edge detection

Convolution Algorithm

```
void image_convolution()
/* to convolve I(x,y) with a mask w(u,v) of size
     (2m+1) × (2m+1) */
{
     int i,j,u,v,sum;

     for (i=m; i<N-m; i++) for (j=m; j<M-m; j++) {
          sum=0;
          for (u=-m; u<=m; u++) for (v=-m; v<=m; v++)
               sum = sum + I[i+u][j+v] * w[u][v];
          g[i][j] = sum;
     }
}
```

Fundamental Concept: Computation Of Derivatives

- **Edge Detection ≡ Finding Derivatives (Gradient)**
  Stronger Edge ⇔ Stronger Gradient
- For 1D continuous signal $f(x)$ given in functional form, its gradient can be computed exactly (df / dx).
- For discrete case, the derivative at x_i can be approximated by one of the following:

1. Backward formula

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

   Assuming $x_i - x_{i\text{-}1} = 1$, then

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = f(x_i) - f(x_{i-1})$$

2. Forward formula

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = f(x_{i+1}) - f(x_i)$$

3. Central formula

$$\left.\frac{df(x)}{dx}\right|_{x=x_i} = \frac{f(x_{i+1}) - f(x_{i-1})}{2}$$

1D Edge Detection Algorithm

```
unsigned char f[N];          // 1D signal of length N
unsigned char edge[N];       // result edge map
int N;                       // length N
int Threshold;               // Threshold for edge

void detect_edge_1d()
{
    int i;

    for (i=1; i<N; i++) {
        delta = fabs( f[i] - f[i-1] );      // 1st difference
        if (delta>T) edge[i]=1;             // mark as edge
        else         edge[i]=0;             // non-edge
    }
}
```

Mask Representation: 1D Derivative

Forward/backward differencing

-1	1

Central differencing

-1	0	1

2D Derivatives

For 2D image $f(x, y)$

- Calculate the gradient vector

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

whose direction is

$$\theta(x,y) = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

where the angle is w.r.t the x axis.

- Mark those (x, y) points whose gradient magnitude is greater than a threshold T.
- In discrete case, the gradient is approximated by a pixel difference operator
- The gradient magnitude is defined as

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

or approximated by

$$\|\nabla f(x,y)\| \approx \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

Gradient Operators

- Based on Backward differences

$$\left.\frac{\partial f}{\partial x}\right|_{x_i, y_j} = f\left(x_i, y_j\right) - f\left(x_{i-1}, y_j\right)$$

$$\left.\frac{\partial f}{\partial y}\right|_{x_i, y_j} = f\left(x_i, y_j\right) - f\left(x_i, y_{j-1}\right)$$

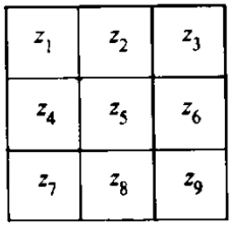
- Sobel Gradient Operator

$$\left.\frac{\partial f}{\partial x}\right|_{x_i, y_j} = \left[f\left(x, y_{j-1}\right) + 2 f\left(x, y_j\right) + f\left(x, y_{j+1}\right)\right]_{x=x_{i+1}} -$$

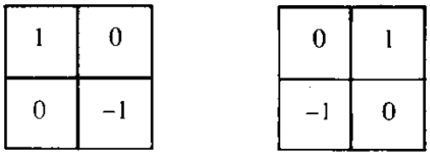
$$\left[f\left(x, y_{j-1}\right) + 2 f\left(x, y_j\right) + f\left(x, y_{j+1}\right)\right]_{x=x_{i-1}}$$

$$\left.\frac{\partial f}{\partial y}\right|_{x_i, y_j} = \left[f\left(x_{i-1}, y\right) + 2 f\left(x_i, y\right) + f\left(x_{i+1}, y\right)\right]_{y=y_{j+1}} -$$

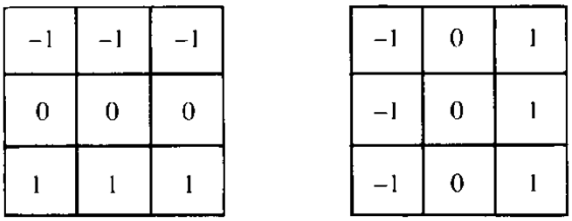
$$\left[f\left(x_{i-1}, y\right) + 2 f\left(x_i, y\right) + f\left(x_{i+1}, y\right)\right]_{y=y_{j-1}}$$



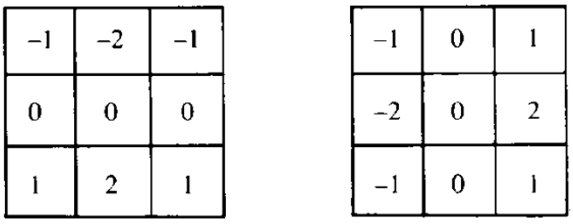
(a)



(b) Roberts



(c) Prewitt



(d) Sobel

Figure 4.28 *A 3 × 3 region of an image (the z's are gray-level values) and various masks used to compute the derivative at point labeled z_5. Note that all mask coefficients sum to 0, indicating a response of 0 in constant areas, as expected of a derivative operator.*

2D Edge Detection Algorithm (Roberts Operator)

```
unsigned char f[N][M];          // 2D image
unsigned char edge[N][M];       // Resulting edge map
int Threshold;                  // Threshold for edge

void detect_edge_2d()
{
    float Roberts(int i, int j);
    int i;

    for (i=1; i<N; i++) for (j=m; j<M; j++) {
            delta = grad_mag_Roberts(i,j);// 1st difference
            if (delta>T) edge[i][j]=1;        // mark as edge
            else         edge[i][j]=0;        // non-edge
    }
}
float grad_mag_Roberts(int i, int j)
{
    float dx, dy, mag;
    dx = f[i][j]-f[i-1][[j-1];
    dy = f[i][j-1]-f[i-1][j];
    mag = fabs(dx) + fabs(dy);
    return(mag);
}
```

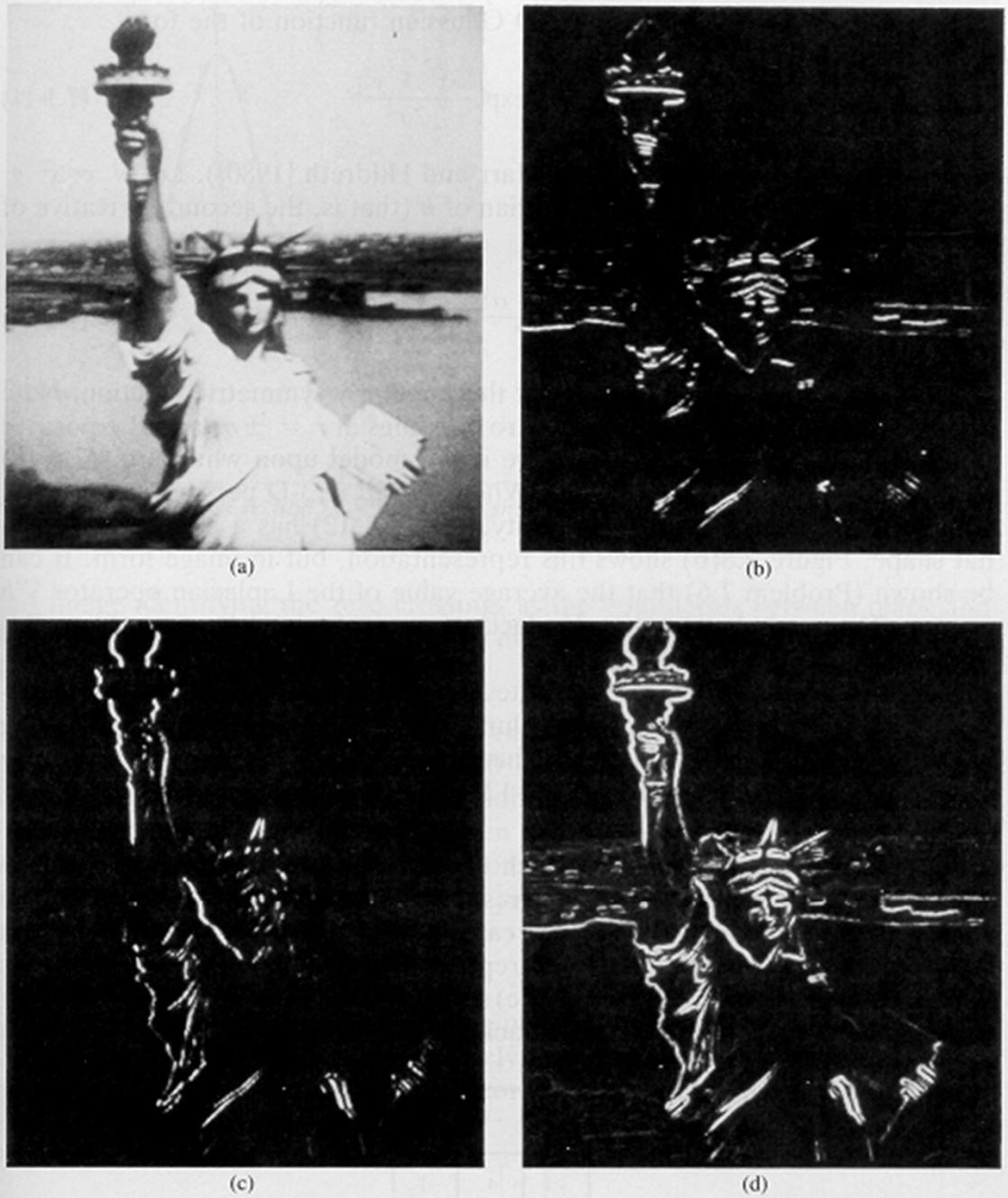


Figure 7.6 *(a) Original image; (b) result of applying the mask in Fig. 7.5(b) to obtain G_x; (c) result of using the mask in Fig. 7.5(c) to obtain G_y; (d) complete gradient image obtained by using Eq. (7.1-5).*

Design of Mask Weights

- Let the image values in a 3×3 neighbourhood be described by a linear relationship

$$f(x, y) = \alpha x + \beta y + \gamma$$

- The 3×3 image pattern for pixel is then given by

$-\alpha - \beta + \gamma$	$-\alpha + \gamma$	$-\alpha + \beta + \gamma$
$-\beta + \gamma$	γ	$\beta + \gamma$
$\alpha - \beta + \gamma$	$\alpha + \gamma$	$\alpha + \beta + \gamma$

- Taking the gradient,

$$\nabla f = \alpha \mathbf{i} + \beta \mathbf{j}$$

- Gradient magnitude, $|\nabla f|$

$$|\nabla f| = \sqrt{\alpha^2 + \beta^2}$$

- Gradient direction, $\tan\theta$

$$\tan\theta = \beta / \alpha$$

- Let the gradient masks be represented by

– a	– b	– a
a	b	a

g_x

– a		a
– b		b
– a		a

g_y

- Convolving these masks with the image pixels result in

$$f_x = 2\alpha(2a+b)$$
$$f_y = 2\beta(2a+b)$$
$$|\nabla f| = \sqrt{f_x^2 + f_y^2}$$
$$= 2(2a+b)\sqrt{\alpha^2+\beta^2}$$

which implies that $2(2a + b) = 1$

- Considering the effects of noise

$$f(x,y) = \alpha x + \beta y + \gamma + \delta(x,y)$$

where $\delta(x,y)$ is independent, with mean 0 and variance $\sigma^2(x,y)$, then

$$f_x = 2\alpha(2a+b) + a[\delta(-1,-1) + \delta(-1,1) + \delta(1,-1) + \delta(1,1)] + b[\delta(-1,0) + \delta(1,0)]$$

- Expectation (average) and Variance:

$$E[f_x] = 2\alpha(2a+b)$$

$$Var[f_x] = a^2\left[\sigma^2(-1,-1)+\sigma^2(-1,1)+\sigma^2(1,-1)+\sigma^2(1,1)\right] + b^2\left[\sigma^2(-1,0)+\sigma^2(1,0)\right]$$

- If the variances are of values below

σ_a^2	σ_b^2	σ_a^2
σ_a^2	σ_b^2	σ_a^2

$$Var[f_x] = 4a^2\sigma_a^2 + 2b^2\sigma_b^2$$

$$= 4a^2\sigma_a^2 + \frac{1}{2}\left(1-8a+16a^2\right)\sigma_b^2 \quad [\because 2(2a+b)=1]$$

- Minimizing the variance with respect to a and equating to zero

$$\frac{\partial}{\partial a}Var[f_x] = 8a\sigma_a^2 + \frac{1}{2}(-8+32a)\sigma_b^2 = 0$$

$$2a\left(\sigma_a^2+2\sigma_b^2\right) = \sigma_b^2$$

$$a = \frac{\sigma_b^2}{2\left(\sigma_a^2+2\sigma_b^2\right)}$$

- The value of a that minimizes the noise variance

$$a = \frac{\sigma_b^2}{2\left(\sigma_a^2 + 2\sigma_b^2\right)}$$

- If $\sigma_a^2 = \sigma_b^2$ then a =1/6 and b = 1/6 (why?). The result is

$1/6 \times$

– 1	– 1	– 1
1	1	1

which is the Prewitt operator.

- If $\sigma_a^2 = 2\sigma_b^2$ then a =1/8 and b = ¼. The result is

$1/8 \times$

– 1	– 2	– 1
1	2	1

which is the Sobel operator.

Types of Edge Detectors

- Roberts Edge Detector
- Prewitts Edge Detector
- Sobel Edge Detector
- Canny Edge Detector
- LoG (Laplacian of Gaussian) Edge Detector
- Others?

Criteria For Optimal Edge Detection

- **Good Detection**

  The optimal detector must :
  - minimise the probability of false positives (detecting spurious edges caused by noise), and
  - minimise the probability of missing real edges.

- **Good Localization**

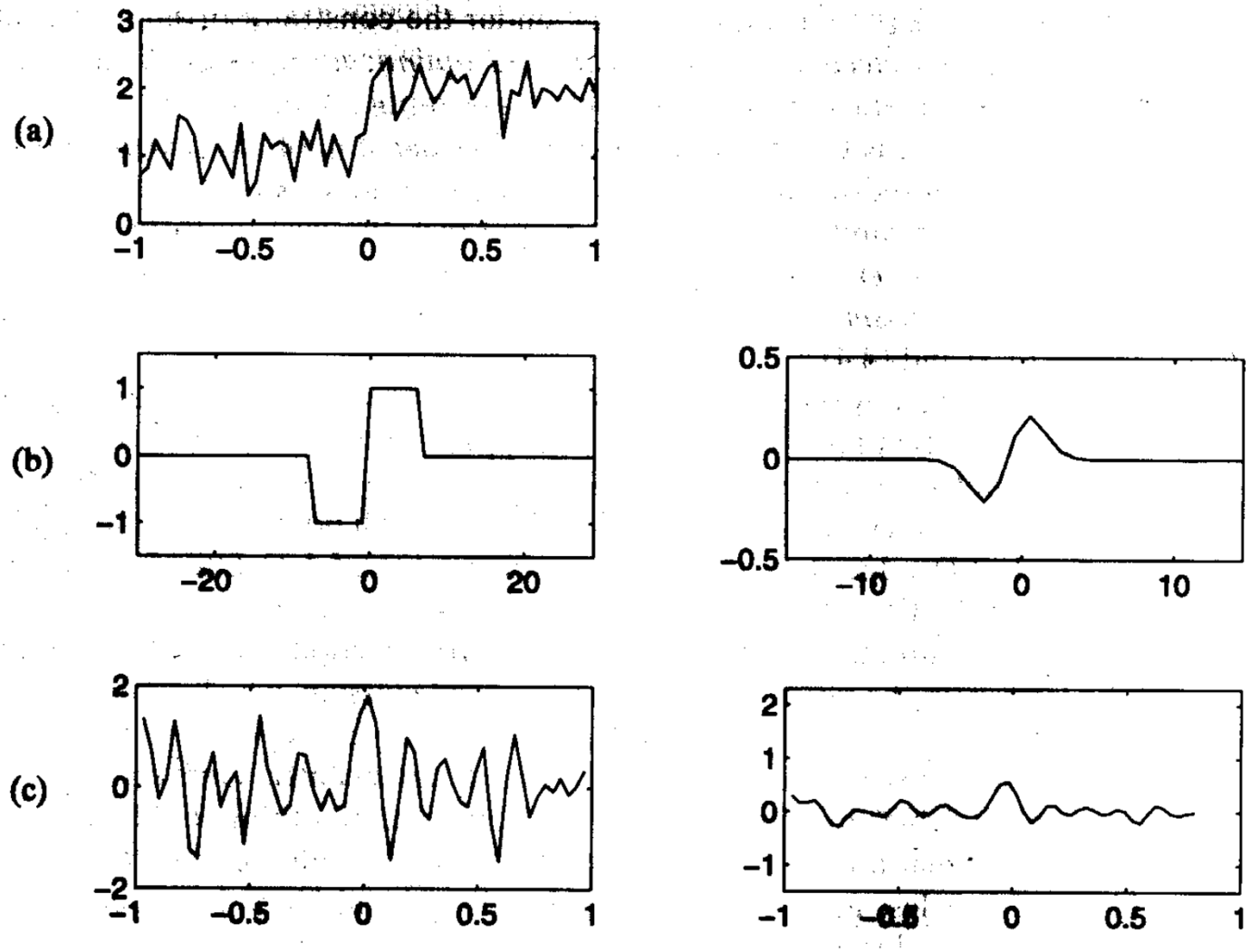
  The edges detected must be as close as possible to the true edges.

  One can prove that the optimal 1D step edge detector is a simple difference operator, or box filter, as shown in Figure 3b. But this filter suffers from multiple responses. Hence a new constraint is required.

- **Single Response Constraint**

  The detector must return one point only for each true edge point; that is, minimize the number of local maxima around the trued edge created by noise.

  The Canny edge operator shows that the first derivative of a Gaussian is a very good approximation to the ideal step edge detector satisfying all three constraints above.



(a) Noisy, step edge, corrupted by additive Gaussian noise (standard deviation is 15% of the uncorrupted step's height).

(b) The box filter (left) and the first derivative of a Gaussian (right).

(c) Response to the noisy edge of the box filter (left) and the first derivative of a Gaussian. The latter contains fewer local maxima due to the smoothing effect of the Gaussian.

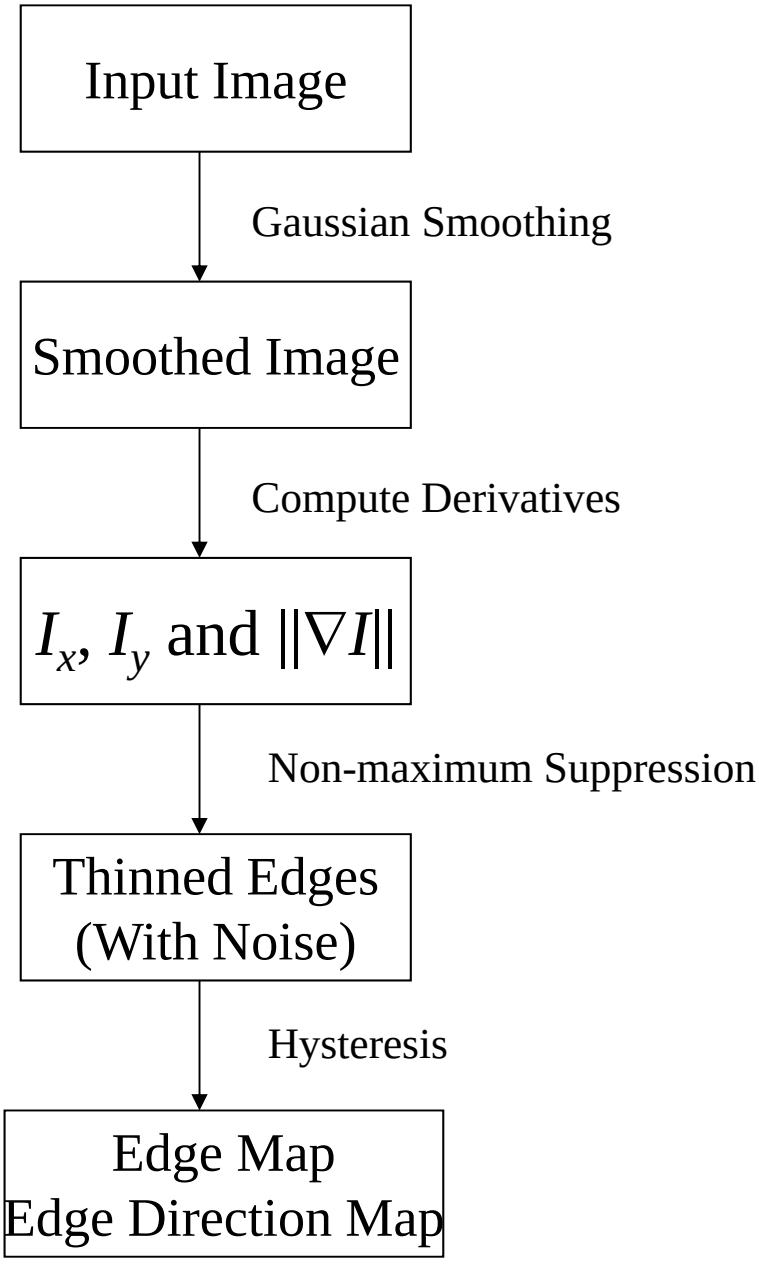
Note:

- The edge operator discussed so far assumes a 1D profile of image values. When applied to a 2D image, one must assume that the edge orientation is known beforehand and that the convolution of the Gaussian derivative is applied in the direction orthogonal to the edge.
- Without prior information about the edge orientation, one have to assume the edge direction in a few discrete directions (possibly 8 directions at 45 degrees interval) and convolve the Gaussian derivative in each of the discrete direction (ie 8 times in each of the 45 degree directions). This is extremely inefficient.
- A better implementation uses the following fact that the convolution of Gaussian derivative with the image is the same as finding the derivative of the Gaussian smoothed image.

$$\nabla G \otimes I = \nabla\left(G \otimes I\right)$$

- With this, the image is first smoothed using the Gaussian mask and the derivatives of the smoothed image are extracted. This is accomplished in 1 pass.

Canny Edge Operator



Smoothing

- Concept of Noise
  - Gaussian Noise (White Noise)
  - Impulse Noise

- Linear Filtering/Smoothing
  - Smoothing By Temporal Averaging
  - Smoothing By Spatial Averaging
  - Gaussian Smoothing

- Non-linear Smoothing
  - Median Filtering

Concept Of Noise

Assume that image noise is **additive** and **random**.

Noise $n\ (i, j)$ is added to true pixel values $I\ (i, j)$ resulting in corrupted pixel values $I_n(i, j)$

$$I_n(i, j) = I\ (i, j) + \ n\ (i, j)$$

- Additive noise is an adequate assumption. In some other system, multiplicative noise ($I_n(i, j) = n\ I\ (i, j)$) may be necessary. Examples include image degradation in television lines and grainy photographs.

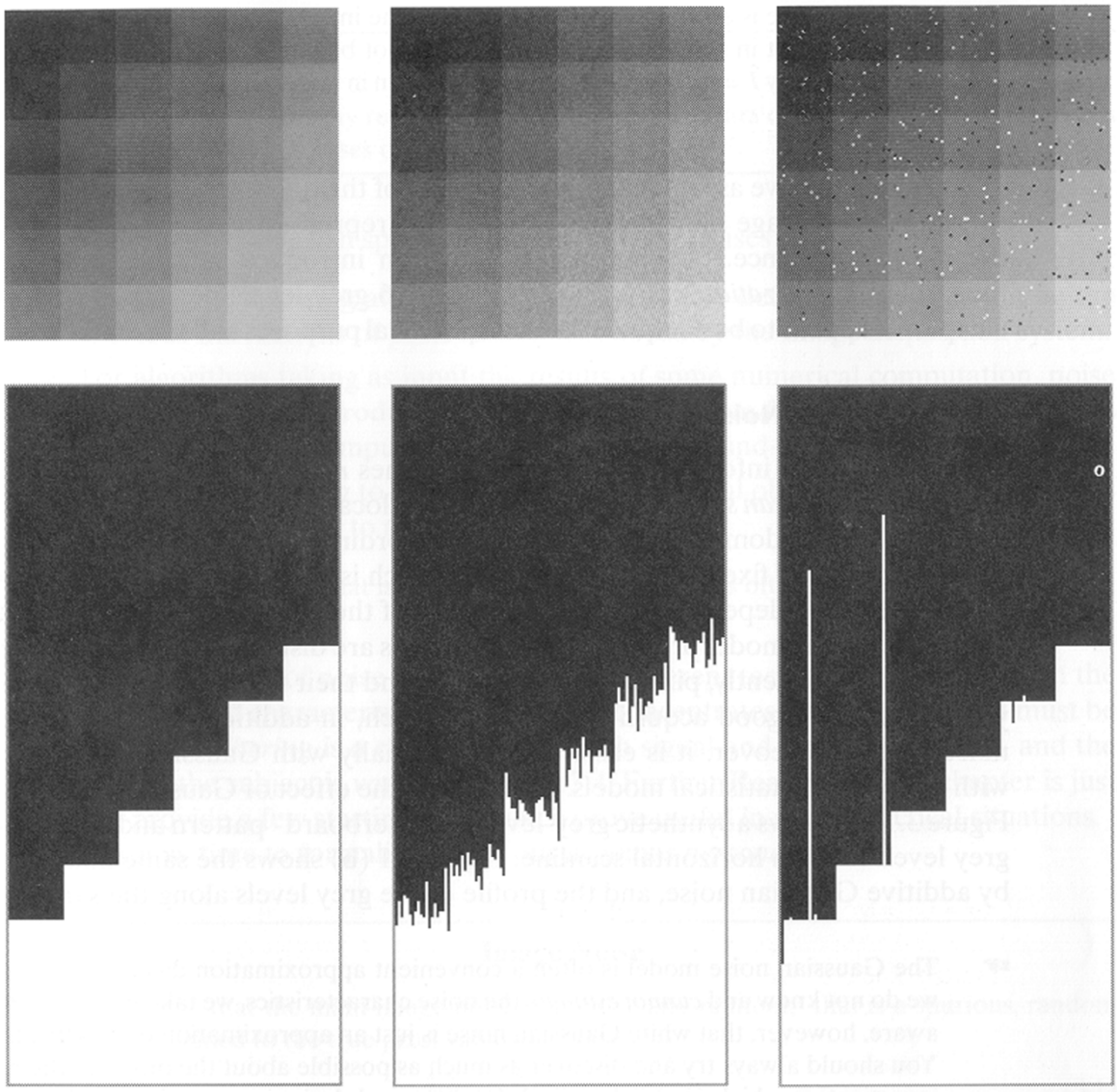
- Sources of noise include:
  - Spurious fluctuations of pixel values in image sensor
  - Round-off, computational errors in different stages of the algorithm
  - Interpretation errors such as noisy edges belonging to other objects of interest.

Types Of Noise

- Gaussian Noise

  In the absence of information, one often assumes that $n(i, j)$ is modelled as
  - random variable
  - distributed according to a zero-mean, Gaussian distribution of a certain standard deviation
  - stochastic - independent of each other in both time and space

- Impulsive Noise (Spot, peak, salt & pepper noise)
  - Random sprinkling of dark and bright spots
  - Usually occurs during image acquisition



(a) (b) (c)

Figure 3.1 (a) Synthetic image of a 120×120 grey-level "checkerboard" and grey-level profile along a row. (b) After adding zero-mean Gaussian noise ($\sigma = 5$). (c) After adding salt and pepper noise (see text for parameters).

Linear Filtering / Smoothing

- **Objective**

  Given an image *I*, corrupted by noise *n*, attenuate *n* as much as possible (ideally, eliminate it altogether) without altering *I* significantly.

- **Importance**

  Many subsequent image processing algorithms (e.g. computing image derivatives in edge detection) can be adversely affected by noise.

- **Types of Smoothing**
  - Temporal Smoothing (Smoothing using several frames of an image sequence)
  - Spatial Smoothing Using Simple Averaging (Smoothing using the neighbourhood of a pixel)
  - Spatial Gaussian Smoothing (Smoothing using the Gaussian distribution weights)

Image Averaging

- A form of noise suppression
- Consider a noisy image $g(x, y)$ formed by the addition of noise $\eta(x, y)$ to an original image $f(x, y)$

$$g(x,y) = f(x,y) + \eta(x,y)$$

- If the noise has zero mean ($\mu = 0$) and uncorrelated, then the noise may be reduced by ***image averaging***.

Algorithm

- Take K images $g_1(x, y), \ldots, g_K(x, y)$
- Average the K images

$$h(x,y) = \frac{1}{K}\sum_{k=1}^{K} g_k(x,y) = f(x,y) + \frac{1}{K}\sum_{k=1}^{K} \eta_k(x,y)$$

- The expected value of h is $E\{h(x, y)\} = f(x, y)$.
- The variance of h is $1/K$ of variance of η_k which vanishes when $K \to \infty$.

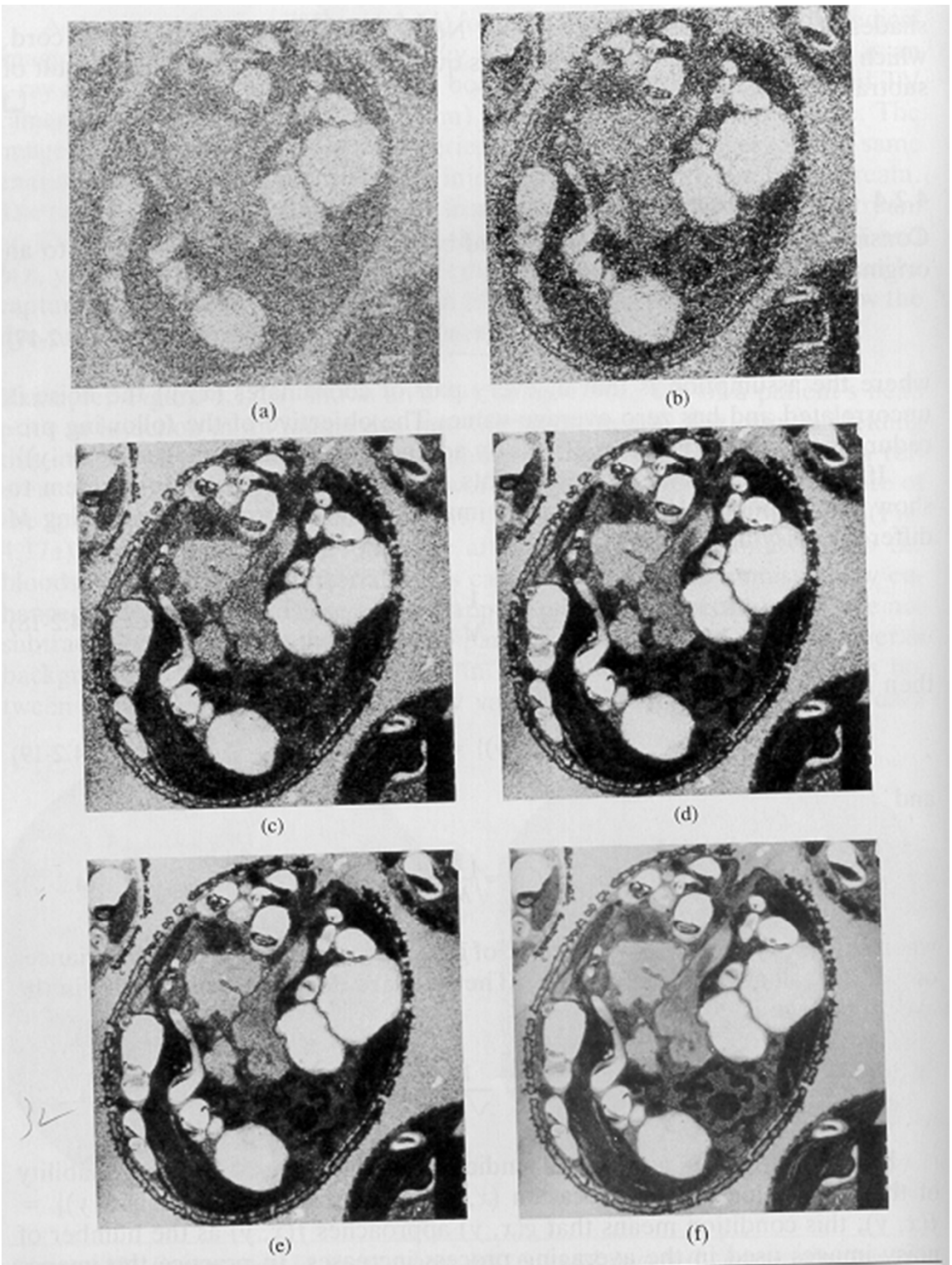


Figure 4.18 *Example of noise reduction by averaging: (a) a typical noisy image; (b)–(f) results of averaging 2, 8, 16, 32, and 128 noisy images.*

Spatial Smoothing Using Simple Averaging (Convolution)

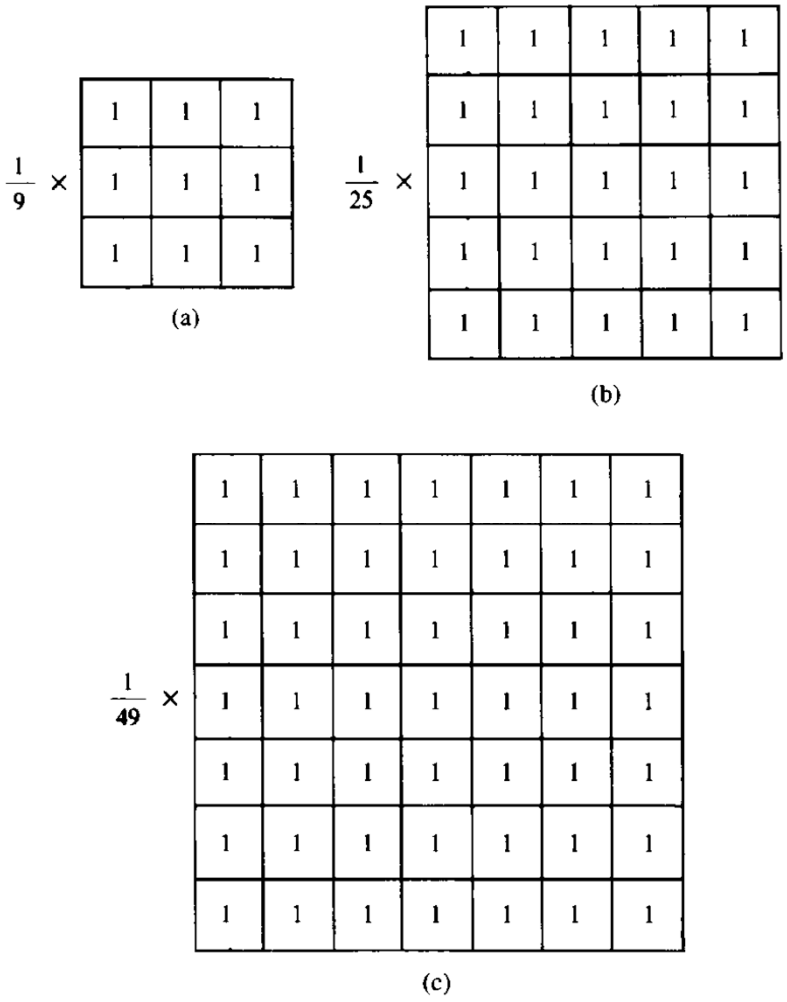


Figure 4.21 *Spatial lowpass filters of various sizes.*

- Limitations of Simple Averaging

  Sharp signal variations are lost due to the filtering effect. As a result, the image is blurred. See Figure.

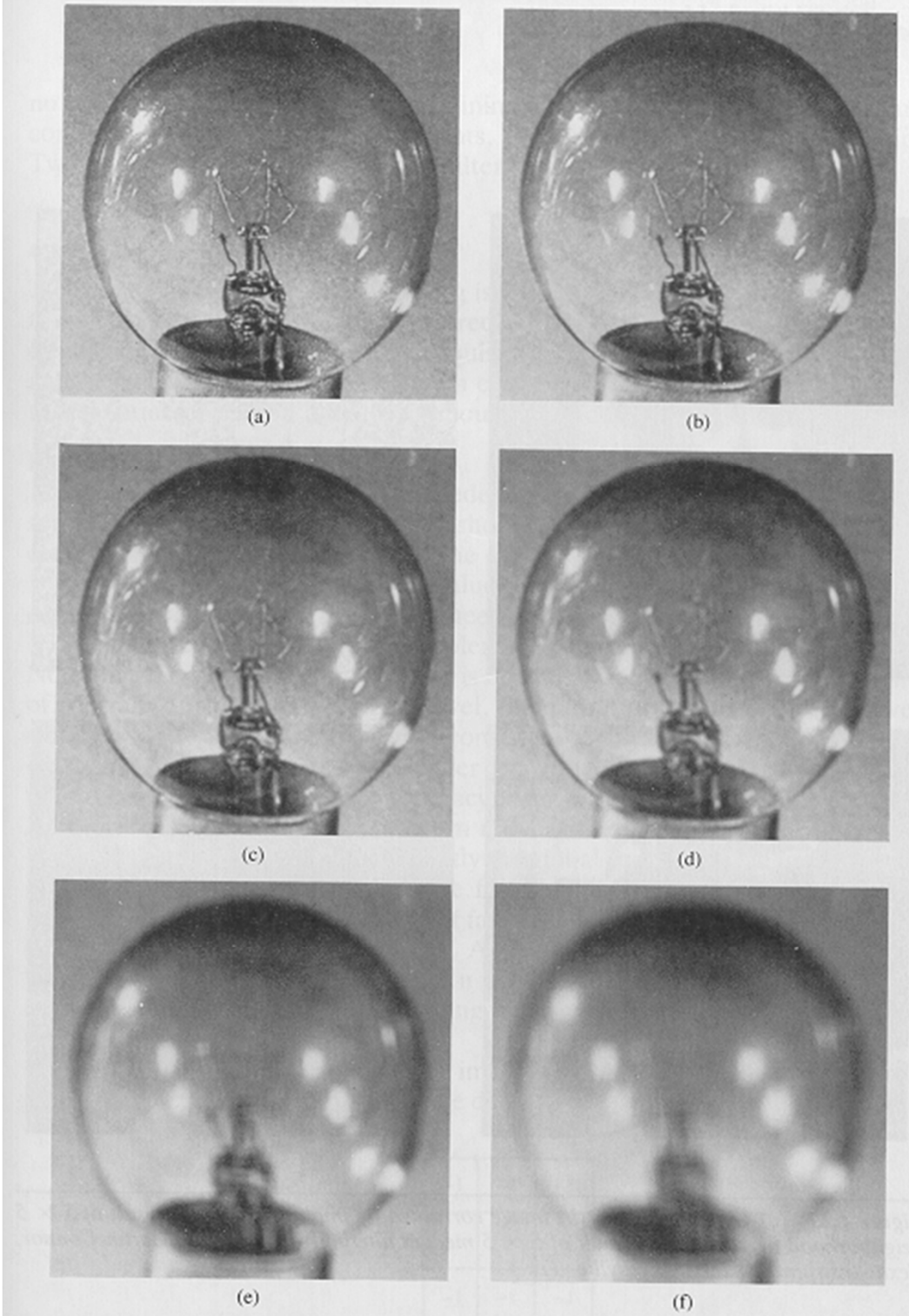


Figure 4.22 *(a) Original image; (b)–(f) results of spatial lowpass filtering with a mask of size $n \times n$, $n = 3, 5, 7, 15, 25$.*

Gaussian Smoothing

- Gaussian smoothing filter uses different weights for emphasis

$$G(x, y) = A \exp\left(-\frac{x^2 + y^2}{2\sigma^2} \right) \qquad (1)$$

where σ is the standard deviation used to control the level of smoothing.

- Gaussian mask must be discretized (or digially sampled). Say, $\sigma = 1$, the discrete mask for a 3 X 3 window is

G(-1, -1)	G(-1, 0)	G(-1, 1)
G(0, -1)	G(0, 0)	G(0, 1)
G(1, -1)	G(1, 0)	G(1, 1)

0.3679	0.6065	0.3679
0.6065	1	0.6065
0.3679	0.6065	0.3679

- To enclose 98.76% of the area (probability) under the Gaussian distribution, a mask must be truncated with a minimum width, *w*, of at least $w = 5\sigma$.
  - Mask size must be an odd integer for mask symmetry.

  For example:

  If $\sigma = 1$ pixel is required, $w = 5\sigma = 5$ pixels is required.

  If $\sigma = 1.5$ pixels, $w = 5\sigma = 7.5$ pixels is required. For odd integer, a mask size of 9 X 9 is chosen.

- Integer versus Floating Point Mask Values

  Sampling a continuous Gaussian yields mask entries of floating point numbers. These values will be convolved with image intensity values. Because of the greater computational burden of convolving floating point numbers, the process and be made more efficient by approximating the mask values with integers. Integer convolution will be more efficient computationally.

  To build an integer mask
  - normalize the mask by making the smallest entry 1
  - round off the results to the nearest integer.

- Example of Converting Floating Point Mask To Integer Mask

0.3679	0.6065	0.3679
0.6065	1	0.6065
0.3679	0.6065	0.3679

$\times \frac{1}{0.3679}$

1	1.6485	1
1.6485	2.7181	1.6485
1	1.6485	1

round-off

1	2	1
2	3	2
1	2	1

Scale Factor

- The weights in the mask tend to increase the overall intensity values of the smoothed image.
- Scale factor is introduced to "normalize" the weights since the area under the Gaussian PDF should sum to 1.

$$\int_{-\infty}^{\infty}\int G(x,y)dxdy = 1$$

- Using Eq. (1)

$$A = \frac{1}{\sum G(x,y)}$$

$$= \frac{1}{G(-1,-1)+G(-1,0)+G(-1,1)+\ldots+G(1,1)}$$

- Scale factor should only be introduced **after** convolving the integer mask with the image, and **not before**. Why?

Median Filtering

- Smoothing without blurring edges and other sharp details and removes impulse noise.
- Assumptions:
  - impulse noise has large values
  - does not occur frequently
- When added to an image, such noise produce a few abruptly high (or low) pixel values compared to normal pixel values in a neighbourhood
- ***Median filtering*** replaces every pixel value by the median value of the neighbouring pixels.
- What is median?
  E.g. Within a 3×3 window of pixel values:

20	180	10
25	30	25
15	20	35

  - Sort them in order (10,15,20,20,25,25,30,35,180)
  - Median is the value (25) in the middle position (5th)
  - Replace the centering pixel with the median of 25.
  - If averaging is used, the centering pixel would be replaced by average of 360/9 = 40

Figure 4.23 *(a) Original image; (b) image corrupted by impulse noise; (c) result of 5 × 5 neighborhood averaging; (d) result of 5 × 5 median filtering. (Courtesy of Martin Connor, Texas Instruments, Inc., Lewisville, Tex.)*

Zero-Crossing Edge Detectors

- Second derivatives filters which result in zero-crossings. At the step edge,
  position where 1st derivative is maximum ≡
  position where 2nd derivative has zero-crossing
- Isotropic generalization of the 2nd derivative in 2 dimensions is the Laplacian

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

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

- Two of the common 3 × 3 masks are

	1	
1	-4	1
	1	

1/3 ×

1	1	1
1	-8	1
1	1	1

Design of Mask Weights

- Let the image values in a 3×3 neighbourhood be described by a bi-quadratic relationship

$$f(x, y) = k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2$$

- The 3×3 image pattern for pixel is then given by

$k_1 - k_2 - k_3 + k_4 - k_5 - k_6$	$k_1 - k_2 + k_4$	$k_1 - k_2 + k_3 + k_4 - k_5 + k_6$
$k_1 - k_3 + k_6$	k_1	$k_1 + k_3 + k_6$
$k_1 + k_2 - k_3 + k_4 - k_5 + k_6$	$k_1 + k_2 + k_4$	$k_1 + k_2 + k_3 + k_4 + k_5 + k_6$

- Taking the gradient,

$$\frac{\partial f}{\partial x} = k_2 + 2k_4 x + k_5 y \qquad \frac{\partial^2 f}{\partial x^2} = 2k_4$$

$$\frac{\partial f}{\partial y} = k_3 + 2k_6 y + k_5 x \qquad \frac{\partial^2 f}{\partial y^2} = 2k_6$$

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 2k_4 + 2k_6$$

- Let the zero-crossing mask be represented by

a	b	a
b	e	b
a	b	a

g_x

- Convolving the mask with the image pixels result in

$$\nabla^2 f = 4a(k_1 + k_4 + k_6) + 2b(2k_1 + k_4 + k_6) + ek_1$$

Since

$$\nabla^2 f = 2k_4 + 2k_6$$

by equating coefficients of k_1, k_4 and k_6

$$2a + b = 1$$
$$e = -4a - 4b$$

- Considering the effects of noise

$$f(x, y) = k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2 + \delta(x, y)$$

where $\delta(x,y)$ is independent, with mean 0 and variance $\sigma^2(x,y)$.

$$\nabla^2 f = 2k_4 + 2k_6 + a[\delta(-1,-1) + \delta(-1,1) + \delta(1,-1) + \delta(1,1)] + b[\delta(-1,0) + \delta(1,0)] + e[\delta(0,0)]$$

- If the noise variance is constant for all pixels, then

$$Var\left[\nabla^2 f\right] = 4a^2\sigma^2 + 4b^2\sigma^2 + e^2\sigma^2$$
$$= \sigma^2\left[4a^2 + 4b^2 + (4a + 4b)^2\right]$$

- To minimize the variance subject to the constraint of $2a + b = 1$, we make use of the Lagrangian multiplier technique. Let

$$\varepsilon^2 = \sigma^2\left[4a^2 + 4b^2 + (4a + 4b)^2\right] + \lambda(2a + b - 1)$$

- Minimizing the variance would be equivalent to minimizing ε^2 by differentiating ε^2 with respect to a, b and ε.

$$\frac{\partial\varepsilon^2}{\partial a} = \sigma^2\left[8a + 2(4a + 4b)4\right] + 2\lambda$$

$$\frac{\partial\varepsilon^2}{\partial b} = \sigma^2\left[8b + 2(4a + 4b)4\right] + \lambda$$

$$\frac{\partial\varepsilon^2}{\partial\lambda} = 2a + b - 1$$

- Setting the partial derivatives to zero and solving for a and b results in a=2/3, b=–1/3 and e= –(4a+4b)= –4/3. The resulting mask is

$1/3 \times$

2	–1	2
–1	–4	–1
2	–1	2