DIGITAL IMAGE PROCESSING (COM-3371)
Week 3 – January 28, 2002

Topics:

• Frame processing
  • Addition/subtraction
  • Logical operations
  • Averaging
• Area processes - spatial filtering
  • Introduction
  • Convolution in 1D
  • Convolution in 2D
  • Lowpass and highpass filtering
  • Smoothing
  • Median filtering
  • Sharpening
  • Unsharp masking
  • Line and edge detection
    - gradient operators
    - Laplacian
• Noise

Readings

• Chapters 3.4 - 3.8 and 10.1 of text
• Short article about noise (on our web site)
Frame processes generate a pixel value based on an operation involving two or more different images. The pixelwise operations in this section will generate an output image based on an operation on pixels from two separate images. Each output pixel will be located at the same position in the input image.

Addition

- Addition can be used to generate a new image C by using a fraction $\alpha$:
  \[ C(x,y) = \alpha A(x,y) + (1-\alpha) B(x,y) \]
  where $\alpha$ is included to prevent overflow

- Parameter $\alpha$ can be used to specify how one image can dominate the other by a certain amount;
  --> some graphics systems have extra information stored with each pixel - it is called the alpha channel and specifies how two images can be blended, switched or combined in some way
Subtraction

- Used to determine **differences** between two images

- Applications:
  1. In machine vision systems, image subtraction is used to inspect products coming off of an assembly line. An image is acquired of the finished product and compared against a master copy of how the product should look.
  2. Background subtraction - when a pattern of superimposed noise is known, it can be subtracted out of an image.
  3. Motion detection

Example: Image (c) was generated by subtracting pattern (b) from original image (a).

Multiplication and division

- Multiplication can be used to mask portions of an image (mask image is unity in areas to be left intact and zero in areas to be masked).

- Division may be used to generate ratio images - useful in multispectral analysis, fluorescence microscopy, etc.
Logical operations

- Logic operations are basic tools in binary image processing; they are used for masking, feature detection or shape analysis.

- Some examples:

  
  \[
  \begin{array}{c|c|c}
  A & B & A \text{ AND } B \\
  \hline
  1 & 1 & 1 \\
  0 & 0 & 0 \\
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  \end{array}
  \]

  
  \[
  \begin{array}{c|c|c}
  A & B & A \text{ OR } B \\
  \hline
  1 & 1 & 1 \\
  0 & 0 & 0 \\
  1 & 0 & 1 \\
  0 & 1 & 1 \\
  \end{array}
  \]

  
  \[
  \begin{array}{c|c|c}
  A & B & A \text{ XOR } B \\
  \hline
  1 & 1 & 0 \\
  0 & 0 & 0 \\
  1 & 0 & 1 \\
  0 & 1 & 1 \\
  \end{array}
  \]

  
  \[
  \begin{array}{c|c|c}
  A & B & \text{NOT}(A) \text{ AND } (B) \\
  \hline
  1 & 1 & 0 \\
  0 & 0 & 0 \\
  1 & 0 & 0 \\
  0 & 1 & 1 \\
  \end{array}
  \]

p.49, Fig.2.14, Gonzalez & Woods
Averaging

- Average function sums two pixel values and divides the sum by two
- It is used to filter out noise from image transmissions
- Averaging can help if the noise in each image is uncorrelated and has zero average value
- Images must be registered before averaging

see section 3.4.2 in Gonzalez & Woods (G&W)
This slide is from the previous lecture:

**image enhancement**

- spatial domain
  - point processes
  - area processes
- frequency domain
  - frame processes

TODAY WE WILL TALK ABOUT AREA PROCESSES
• Area processing is formulated in the context of so-called mask operations (the terms template, kernel, window, filter, and convolution mask are also used to denote a mask); mask is an array of numbers (typically 3x3 or 5x5, etc.); mask is usually very small with respect to the processed image; mask values are called coefficients; values of coefficients determine the nature of the process;

• The idea behind a mask operations is to let the value assigned to a pixel be a function of its gray level and the gray level of its neighbors; for example, consider a sub-image area:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( z_1 )</td>
<td>( z_2 )</td>
<td>( z_3 )</td>
</tr>
<tr>
<td>( z_4 )</td>
<td>( z_5 )</td>
<td>( z_6 )</td>
</tr>
<tr>
<td>( z_7 )</td>
<td>( z_8 )</td>
<td>( z_9 )</td>
</tr>
</tbody>
</table>

Suppose that we want to replace the value of \( z_5 \) with the average value of pixels in a 3x3 region centered at the pixel value \( z_5 \); to do so we perform the following operation:

\[
z = \frac{1}{9} (z_1 + z_2 + \ldots + z_9) = \frac{1}{9} \sum_{i=1}^{9} z_i
\]

and assign value of \( z \) to \( z_5 \).

• With reference to the mask

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1 )</td>
<td>( w_2 )</td>
<td>( w_3 )</td>
</tr>
<tr>
<td>( w_4 )</td>
<td>( w_5 )</td>
<td>( w_6 )</td>
</tr>
<tr>
<td>( w_7 )</td>
<td>( w_8 )</td>
<td>( w_9 )</td>
</tr>
</tbody>
</table>

the same operation can be written in more general form:
\[ z = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \]  

(2)

where \( w_i \) are mask coefficients

- Other image processing operations are possible by proper selection of the coefficients and application of the mask at each pixel position in an image; the masks can be designed for noise reduction, edge detection, etc.

- Applying a mask at each pixel in an image is a computationally expensive task; for example, a 3x3 mask applied to a 512 x 512 image requires 9 multiplications and 8 additions at each pixel, for a total of 2,359,296 multiplications and 2,097,152 additions.
Convolution in 1D

Consider two sets of numbers: \( A = \{0,1,2,3,2,1,0\} \) and \( B = \{1,3,1\} \).
The following sequence of seven tables shows the process of calculating a discrete convolution \( A*B \):

1. Center value of \( B \) set aligned to first value of \( A \) set. Convolution result \( A*B \) is 1, placed at position corresponding to center of \( B \) set, missing values assumed to be zero.

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]

2. Set \( B \) shifted by one position. Convolution result \( A*B \) is \((0 \times 1) + (1 \times 3) + (2 \times 1) = 5\).

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]

3. Set \( B \) shifted by one position. Convolution result \( A*B \) is \((1 \times 1) + (2 \times 3) + (3 \times 1) = 10\).

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]

4. Set \( B \) shifted by one position. Convolution result \( A*B \) is \((1 \times 2) + (3 \times 3) + (2 \times 1) = 13\).

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]

5. Set \( B \) shifted by one position. Convolution result \( A*B \) is \((3 \times 1) + (2 \times 3) + (1 \times 1) = 10\).

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]

6. Set \( B \) shifted by one position. Convolution result \( A*B \) is \((2 \times 1) + (1 \times 3) + (0 \times 1) = 5\).

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]

7. Set \( B \) shifted by one position. Convolution result \( A*B \) is \((1 \times 1) + (0 \times 3) + (0 \times 1) = 1\) (Missing values assumed to be zero).

   \[
   \begin{array}{c}
   A \\
   B \\
   A^*B
   \end{array}
   \begin{array}{cccccc}
   0 & 1 & 2 & 3 & 2 & 1 & 0 \\
   1 & 3 & 1 \\
   1 & 5 & 10 & 13 & 10 & 5 & 1
   \end{array}
   \]
The plot of the original data A and convolution with B is shown in the following plot:

The peak at the center of the data become steeper.

If the mask had been \{-1,3,-1\}, the result will be:
Convolution in 2D

- A 2D convolution is basically a weighted sum of pixels in the neighborhood of the input pixel

![Convolution Diagram]

- The dimensions of the convolution masks are usually odd, so that the center can be determined; the location of the center corresponds to the location of the output pixel

- Convolution can be performed for the masks with even dimensions by selecting a pixel position in the mask to serve as the compute point

- The sum of mask coefficients affects the overall intensity of the resulting image

- Many convolution masks have coefficients that sum to 1 ---\(\rightarrow\) then the convolved image will have the same average intensity as the original image

- Some convolution masks have negative coefficients and sum to 0 ---\(\rightarrow\) negative pixel values may be generated

- Please check Fig. 3.32 and Equation 3.5.1 (p. 117, G&W)
How to treat the image borders?

- Treat the empty cells in the convolution windows as zeros; this is known as **zero-padding**; it is not acceptable if the edges of the resulting image are as important as the rest of the image.

- Start convolving at the first position where the window doesn't overlap the image (if your convolution mask is 3x3, start convolving with the pixel at 1,1 instead of pixel at 0,0); in the output image, the convolved edges are copied to create an image with the same resolution as the output image - in case of a 3x3 mask, the output pixel from convolving the input pixel at (1,1) would be placed at (1,1) in the output image and also at (1,0); when that line is duplicated, that pixel is copied to (0,0) and (0,1); the same concept is true for the right side of the line, and the bottom of the image.

- Duplicate the edges - copy the left column, the right column, the top row and the bottom row of the image.

- "Wrap" the image - in case of a 3x3 mask operated on a 512 x512 image, the first convolution window would operate on the pixels at locations:
  
  $\begin{bmatrix}
  (511, 511) & (0, 511) & (1, 511) \\
  (511, 0) & (0, 0) & (1, 0) \\
  (511, 1) & (0, 1) & (1, 1)
  \end{bmatrix}$
Separable masks?

- Many convolution masks are separable. This means that the convolution can be performed by executing two convolutions with 1-dimensional masks. Separable functions reduce the number of computations required when using large masks. This is possible due to the linear nature of convolution. For example, a convolution using the following mask

\[
\begin{array}{ccc}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

can be performed faster by doing two convolutions using

\[
\begin{array}{c}
1 \\
0 \\
-1 \\
\end{array}
\quad \text{and} \quad
\begin{array}{ccc}
1 & 2 & 1 \\
\end{array}
\]

since the first matrix is the product of the two vectors.

- The savings in this example are not great (6 multiply accumulates versus 9), but do increase as masks sizes grow.
Lowpass and highpass filtering

The term spatial filtering implies the separation of frequency components within an image. The frequency components are spatial frequencies which relate to the rapidity of changing gray levels over a certain spatial distance. Where rapid intensity transitions are prevalent, we have high spatial frequency. Slow transitions represent low frequency.

- **Lowpass filters** attenuate or eliminate high-frequency components, while leaving low frequencies untouched (that is, the filter "passes" low frequencies); since high-frequency components characterize edges and other sharp details in image, the effect of lowpass filtering is blurring.

- **Highpass filters** attenuate or eliminate low-frequency components, while leaving high frequencies untouched (that is, the filter "passes" high frequencies); since low-frequency components characterize overall contrast and average intensity, the effect of highpass filtering is sharpening of edges and other sharp details.
Smoothing Filters

- Smoothing filters are used for blurring (removal of small details from an image prior to large object extraction) and for noise reduction

Lowpass filtering

- Lowpass filter always has **positive coefficients**
- **Sum** of the coefficients is equal to 1
- **It blurs edges**: if the objective is to achieve noise reduction rather than blurring, use median filter
- The bigger the mask, the greater blurring effect and the greater the computation time required
- Simple **average filters**:

\[
\frac{1}{9} \times \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\frac{1}{25} \times \begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\frac{1}{49} \times \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

- Averaging is an effective way of reducing Gaussian noise in an image
- Another commonly used convolution mask for blurring is:

\[
\begin{bmatrix}
\frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\
\frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\
\frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\
\end{bmatrix}
\]

It is a **weighted average** – compare it with Fig. 3.34b in G&W

- The effects of smoothing as a function of filter size are illustrated in Fig. 3.35 in G&W
Median filtering

- The gray level of each pixel is replaced by the median of the gray levels in the neighborhood of that pixel.

- The size of median filter mask must be an odd number - why? ......................

- **It is nonlinear filter** - operates directly on the values of the pixels in the neighborhood, but does not explicitly perform convolution operation, as previously described.

- Median filtering is an effective method for removing impulse noise (salt and pepper).

- The principal function of median filtering is to force pixels with very distinct intensities to be more like their neighbors, thus eliminating intensity spikes.

**Exercise:**

*Find the median of the following values: {10,20,20,20,15,20,20,25,100}*

![Diagram showing median filtering process]
Also, please check Fig. 3.37 in G&W

**Examples of median filter masks:**

- **Horizontal**
- **Vertical**
- **Cross**
- **Block**
- **X**
- **Diamond**
Sharpening

- Sharpening filters are used for highlighting fine detail in an image or to enhance detail that has been blurred.

**Highpass filtering**

- Highpass mask should have positive coefficients near its center, and negative coefficients in the outer periphery.
- Typical highpass spatial filter:

\[
\frac{1}{9} \times \begin{bmatrix}
-1 & -1 & -2 \\
-1 & 8 & -1 \\
-1 & -1 & -1
\end{bmatrix}
\]

- **Sum** of the coefficients is 0.
- Output may have negative values ---> perform scaling.
- Since highpass enhances fine detail in an image, it tends to amplify the noise.

*Figure 4.25* (a) Image of a human retina; (b) highpass filtered result using the mask in Fig. 4.24.
High-boost filtering (unsharp masking)

- It is computed by increasing the intensity of the original image and subtracting a lowpass image:

\[
\text{High-boost} = \alpha \text{Original} - \text{Lowpass}
\]

when \( \alpha = 1 \), the result is a standard highpass filter;
when \( \alpha > 1 \), part of the original is added back to the highpass result, which restores partially the low-frequency components lost in the highpass filtering operation

\( \rightarrow \) the high-boosted image looks more like the original image

- This is one of the basic tools for image processing applications in the printing and publishing industry
- Implementation:

\[
\frac{1}{9} \times \begin{pmatrix}
-1 & -1 & -1 \\
-1 & 9 \alpha & -1 \\
-1 & -1 & -1
\end{pmatrix}
\]

- Please check Section on Unsharp Masking and High Boost Filtering (pp. 132-3), Equations 3.7-7 through 3.7-11.
Effect of the unsharp masking in 1D case:

<table>
<thead>
<tr>
<th>original edge</th>
<th>low-pass</th>
<th>after unsharp mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>1.5 A - B</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>2.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>2.33</td>
<td>0.67</td>
</tr>
<tr>
<td>3</td>
<td>3.00</td>
<td>1.50</td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td>2.00</td>
</tr>
<tr>
<td>5</td>
<td>5.00</td>
<td>2.50</td>
</tr>
<tr>
<td>6</td>
<td>6.00</td>
<td>3.00</td>
</tr>
<tr>
<td>7</td>
<td>6.67</td>
<td>3.83</td>
</tr>
<tr>
<td>7</td>
<td>7.00</td>
<td>3.50</td>
</tr>
<tr>
<td>7</td>
<td>7.00</td>
<td>3.50</td>
</tr>
<tr>
<td>7</td>
<td>7.00</td>
<td>3.50</td>
</tr>
<tr>
<td>7</td>
<td>7.00</td>
<td>3.50</td>
</tr>
</tbody>
</table>

![Graph showing the effect of unsharp masking](image)
Line detection and edge detection

Enhancement of edges in an image is an operation used in segmentation and feature extraction. The operation basically reduces an image to display only its edge information. This information is then used for feature or object recognition by other algorithms.

- the simplest edge enhancement is the shift and difference method; by shifting an image to the left and then subtracting it from the original, vertical edges become apparent

Line detection

- Consider the following convolution masks:

```
-1 -1 -1  
2 2 2   
-1 -1 -1
```

Horizontal

```
-1 -1 2   
-1 2 -1  
2 -1 -1
```

+45°

```
-1 -1 -1  
-1 2 -1  
-1 2 -1
```

Vertical

```
2 -1 -1   
-1 2 -1  
-1 -1 2
```

-45°

- Convolving an image with the horizontal mask will enhance horizontal lines; the 45 degree mask responds best to lines oriented at 45 degrees; the vertical mask - to vertical lines; the -45 degrees mask - to lines in the -45 degrees direction

Note: the preferred direction of each mask is weighted with a larger coefficient than other possible directions
**Edge detection**

- **Edge** - a boundary between two regions with relatively distinct gray level properties

![Edge Types](image)

- There is an infinite number of edge orientations, widths and shapes ---> some edge detectors may work well in one application and perform poorly in others
- Basically, the idea underlying most edge detection techniques is the computation of a **local derivative operator**
- **The first derivative** at any point in an image is obtained by using the magnitude of the gradient at that point; **the second derivative** - by using the Laplacian

![Derivative Operators](image)
Gradient operators
An edge on an image is defined by the local pixel intensity gradient. A gradient is an approximation of the first-order derivative of the image function.

• For a given image $f(x,y)$, the gradient of $f$ at coordinates $(x,y)$ is defined as the vector

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

(4)

The $G_x$ and $G_y$ are gradients in directions $x$ and $y$, respectively. It is well known from vector analysis that the gradient vector points in the direction of maximum rate of change of $f$ at $(x,y)$. The magnitude of this vector

$$\nabla f = \text{mag}(\nabla f) = \sqrt{G_x^2 + G_y^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

(5)

is the basis for various approaches to image differentiation.

• Sometimes for computational simplicity, the magnitude is computed as:

$$\nabla f = |G_x| + |G_y|$$

(6)

• The direction of the edge gradient is computed using the formula:

$$D = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

where the angle is measured with respect to the x axis.

• Since the discrete nature of digital images does not allow the direct application of continuous differentiation, calculation of gradient is done by differencing. Consider the image region:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

An approximation of Eq. 6 in the 3x3 neighborhood of pixel $z_5$ is:

$$\nabla f \approx |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)|$$

(7)
The difference between the third and first row of the 3x3 region approximates the derivative in the x direction, and the difference between third and first column approximates the derivative in the y direction. This can be implemented by taking the absolute value of the response of the following two masks and summing the results:

\[
\begin{bmatrix}
-1 & -1 & 1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{bmatrix} \quad \begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]

**Prewitt masks**

- Another example:

\[
\begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{bmatrix} \quad \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]

**Sobel masks**

- Both magnitude and direction of the gradient can be displayed as images.

- The results of edge detection depend on the gradient mask. Some other edge operators are: Roberts, Prewitt, Robinson, Kirsch, Frei-Chen, etc.
Example of Sobel edge detection:

Examples of Sobel, Prewitt, and Frei-Chen edge detection:
**Compass gradient operators**
- compass gradient operators find edges in 8 different orientations
- the process consists of convolving an image with 8 convolution masks; the output is the maximum of all eight convolutions

Some examples:

<table>
<thead>
<tr>
<th></th>
<th>Prewitt</th>
<th>Kirsch</th>
<th>Robinson 3-level</th>
<th>Robinson 5-level</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>East</strong></td>
<td><img src="image" alt="East mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>Northeast</strong></td>
<td><img src="image" alt="Northeast mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>North</strong></td>
<td><img src="image" alt="North mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>Northwest</strong></td>
<td><img src="image" alt="Northwest mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>West</strong></td>
<td><img src="image" alt="West mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>Southwest</strong></td>
<td><img src="image" alt="Southwest mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>South</strong></td>
<td><img src="image" alt="South mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
<tr>
<td><strong>Southeast</strong></td>
<td><img src="image" alt="Southeast mask" /></td>
<td><img src="image" alt="Kirsch mask" /></td>
<td><img src="image" alt="Robinson 3-level mask" /></td>
<td><img src="image" alt="Robinson 5-level mask" /></td>
</tr>
</tbody>
</table>
**Laplacian**

- Since the peaks in first-order derivative correspond to zeros in the second-order derivative, the Laplacian operator (which approximates second-order derivative) can also be used to detect edges:

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

  \[ (8) \]

- Laplacian is approximated in digital images by an N by N convolution mask

- Laplacian can be implemented in many ways; for a 3x3 region, the form most frequently used is:

  \[ \nabla^2 f = 4z5 - (z2 + z4 + z6 + z8) \]  

  \[ (9) \]

Here are three examples of 3x3 Laplacian masks that represent different approximations of the Laplacian operator:

<table>
<thead>
<tr>
<th>0 -1 0</th>
<th>-1 -1 -1</th>
<th>1 -2 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1 4 -1</td>
<td>-1 8 -1</td>
<td>-2 4 -2</td>
</tr>
<tr>
<td>0 -1 0</td>
<td>-1 -1 -1</td>
<td>1 -2 1</td>
</tr>
</tbody>
</table>

- The mask coefficients always add to 0

- The image edges can be found by locating pixels where Laplacian makes a transition through zero (zero crossings).

- Laplacian generates sharper peaks at edges than does the gradient operation; the enhancement of edges in omnidirectional

- Constant or linearly increasing or decreasing intensity regions become black in the output image

- Laplacian produces double edges

- Other Laplacian masks:
• All edge detection methods which are based on a gradient or Laplacian are very sensitive to noise. In some applications, noise effects can be reduced by smoothing the image before applying an edge operation. Marr and Hildreth proposed to smooth the image with Gaussian filter before application of Laplacian (this operation is called Laplacian of Gaussian, LoG).
Other filters

Embossing

- Embossing an image makes it appear as though it has been etched in a sheet of nickel
- Embossing mask:

\[
\begin{bmatrix}
-1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & -1
\end{bmatrix}
\]

Max and Min

- Similar to median filtering
- The center pixel is replaced with minimum or maximum intensity value within the mask
- Useful in removing extreme impulse noise - the minimum filter removes white spikes; the maximum filter removes dark spikes
- Both min and max filters fail in the removal of mixed impulsive noise ("salt and pepper" noise)