Lecture (2): Edge detectors

* **An edge detection problem** is considered one of the most important and difficult operations in image processing. In many cases, image segmentation processes need edge detection.

* **Image segmentation** is the process of partitioning image into constituent objects. * **Edge** is the boundary between object(s) and background, and edges are the most common features for object detection. Edges are the image positions where the local image intensity changes distinctly along a particular orientation. There are three types of edges: **horizontal**, **vertical**, and **diagonal**.

An image and the corresponding edge image.

First-order edge detection operators

* The edge operators approximate the **local gradient** of an image. The gradient component can give both **magnitude** and **phase** information, i.e.; strength of edge points and local direction of the edge points. The most popular edge operates are **Prewitt** and **Sobel**.

(1) Prewitt operates: The Prewitt operators use the following filters for determining both horizontal and vertical gradients.

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(2) Sobel operators: The Sobel filter provides more smoothing than the Prewitt filter or mask. The Sobel filter computes horizontal and vertical gradients as follows:

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

* The **edge strength** and **edge orientation** can be estimated by either Prewitt or Sobel operator as follows:

$$G_X(x,y) = H_x * f(x,y) \text{ and } G_Y(x,y) = H_y * f(x,y)$$

where ***f(x,y)*** is an input image and ***Hx*** and ***Hy*** are the horizontal and vertical edge operators.

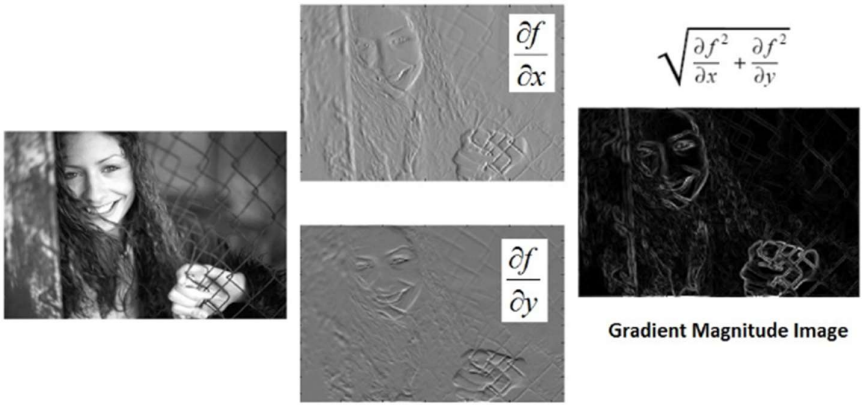
* The Sobel **edge strength** ***M(x, y)*** is the gradient magnitude, i.e.;

$$M(x,y) = \sqrt{(G_X(x,y))^2 + (G_Y(x,y))^2}$$

* The local **edge orientation angle** (direction of the normal to the edge pixel), ***ϕ(x, y)*** is given by:

$$\phi(x,y) = \tan^{-1}\left(\frac{G_Y(x,y)}{G_X(x,y)}\right)$$

* To calculate the final edge boundaries, The edge strength ***M(x, y)*** is compared with a predefined threshold to decide whether the candidate pixel is an edge pixel or not.



Extraction of a gradient magnitude image.

(3) Compass operators: These operators are used to detect lines which are oriented at **0, 45, 90, and 135** directions. The Kirsch or compass operator uses the following eight orientations which are spaced at 45°.

$$H_0^K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad H_4^K = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$H_1^K = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad H_5^K = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

$$H_2^K = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad H_6^K = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$H_3^K = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \quad H_7^K = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$

* Out of all these eight masks, only four masks are considered for convolution with the image. The four others are identical except for the reversed sign.

* The edge strength ***M*(*x*, *y*)** at position **(*x*, *y*)** is defined as the maximum of the above-mentioned eight filter outputs as $M(x,y) = \max(|H_0(x,y)|, |H_1(x,y)|, |H_2(x,y)|, |H_3(x,y)|)$

* The local edge orientation **0°, 45°, 90°, 135°** can be estimated from the strongest-responding filter using $\emptyset(x,y) = \frac{\pi}{4} j, \quad with \quad j = \underset{0 \le i \le 7}{\operatorname{argmax}} H_i(x,y)$

* The **main advantage of compass operator** is that it is not required to compute square roots for detecting edges, while gradient magnitude needs to determine for Sobel and Prewitt operations.

Second-order derivative edge detection method:

* A change in intensity (**an edge**) corresponds to an **extreme value in the first derivative**. Also, a change in intensity corresponds to a **zero crossing in the second-order derivative**.

* In ***first-order method***, the objective is to find a point where the derivative is maximum or minimum. Based on this principle, edge pixels are determined.

* In the second-order derivative, finding the location of edge pixels is equivalent to finding the place where the second-order derivative is zero, i.e., zero crossing gives the location of the edge pixels.

* A ***zero-crossing*** is a point where the sign of a mathematical function changes (e.g. from positive to negative), represented by a crossing of the axis (zero value) in the graph of the

function. The ***zero-crossing algorithm*** would move through the image and see where the sign changed from negative to positive or vice versa

* However, the ***zero-crossing method*** produces two-pixel thick edges (double edges) and they are very sensitive to noise due to double differentiation. The second-order derivative in horizontal and vertical directions can be computed by *Laplace operator*.

(1) Laplacian operator:

The second derivative of an image can be computed by a set of filters as given below:

$$\frac{\partial^2 f}{\partial x^2} \equiv H_x = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \text{ and } \frac{\partial^2 f}{\partial y^2} \equiv H_y = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

These two filters can estimate the second-order derivatives along the x and y directions. These two 1D filters can be combined to make a **2D Laplacian filter**.

$$H = H_x + H_y = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

* Other common variants of 3 × 3 Laplacian masks are:

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ and } H = \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

* Hence, the **sum of the coefficients is zero** in Laplacian operator filter, such that the mask response is zero in an area of constant image intensity.

* The **main advantage of Laplacian** based edge detection is that ***no thresholding is required*** for taking a decision. Also, ***Laplacian operator is symmetric***.

*** Edge sharpening with the Laplacian filter:** The Laplacian filter is first applied to an image $f(x, y)$ and then a fraction is subtracted (determined by the weight w) of the result from the original image as follows:

$$\underset{\text{Edge sharpening image}}{\hat{f}(x,y)} \leftarrow \underset{\text{Original image}}{f(x,y)} - \underset{\text{Laplacian filtering}}{w(H * f(x,y))}$$

The result of edge sharpening by Laplacian filter below

FIGU .10: (a) Original image, (b) edge sharpening by Laplacian filter, and (c) edge detection by Sobel mask.

Unsharp masking: The first step for unsharp masking is to subtract a smoothed-version of an image from the original image. This step enhances the edges. This result is called the "**mask M**". Subsequently, the unsharp version of the image ***f(x, y)*** is obtained by again adding a fraction (determined by the weight ***w***) of the result/**mask M** to the original image. So, the edges in the image are sharpened. The steps are mathematically put as follows:

$$
\begin{aligned}
M &\leftarrow f(x,y) - (f(x,y) * H) = f(x,y) - \hat{f}(x,y) \\
\check{f}(x,y) &\leftarrow f(x,y) + wM = f(x,y) + w(f(x,y) - \hat{f}(x,y)) \\
\therefore \check{f}(x,y) &\leftarrow f(x,y) + wM = (1+w)f(x,y) - w\hat{f}(x,y)
\end{aligned}
$$

(2) Laplacian of Gaussian (LoG):

Convolving an image with Gaussian and Laplacian operator is equivalent to convolution with Laplacian of Gaussian (**LoG**) operator. As a first step, the image is blurred using a **Gaussian operator** and then the **Laplacian operator** is applied. The Gaussian smoothing reduces the noise and hence the Laplacian minimizes detection of false edges. So, the **LoG** kernel is now obtained as:

$$\mathrm{LoG} \triangleq \frac{\partial^2}{\partial x^2} G_\sigma(x,y) + \frac{\partial^2}{\partial y^2} G_\sigma(x,y)$$

$$\therefore \mathrm{LoG} \triangleq \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

The parameter σ controls the extent of blurring of the input image. As σ increases, wider convolution masks are required for better performance of the edge operator.

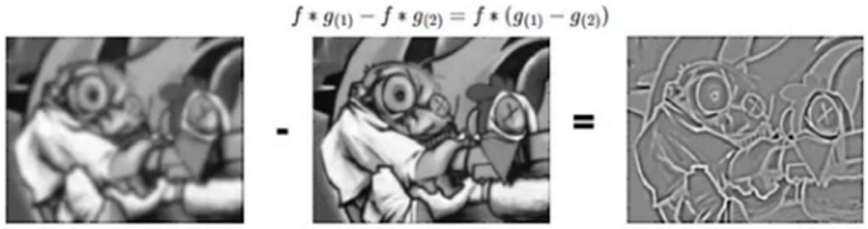
(3) Difference of Gaussian operator (DoG):

The LoG operator can be approximated by taking the differently sized Gaussians. The difference of Gaussian (DoG) filter is obtained as follows:

$$\mathrm{DoG} = G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y)$$

$$\mathrm{DoG} = \frac{1}{\sqrt{2\pi}}\left[\frac{1}{\sigma_1}e^{\left(-\frac{x^2+y^2}{2\sigma_1^2}\right)} - \frac{1}{\sigma_2}e^{\left(-\frac{x^2+y^2}{2\sigma_2^2}\right)}\right]$$

The procedure of edge detection by DoG operator is the same as that of LoG operator. The image is first convolved with DoG operator. Subsequently, the zero crossings are detected and threshold is applied to suppress the weak zero crossings. The main benefit of using DoG is that no actual derivative computation is needed and it is commonly used in many computer vision applications when images are processed using Gaussian filters with different scales.



Difference of Gaussian result without thresholding

(4) Canny edge detector:

Canny's method considers the following issues: **(1)** Minimization of errors of edge detection, **(2)** Localization of edges, i.e., edge should be detected where it is present in the input image, and **(3)** Single response corresponding to one edge pixel.

* ***CANNY EDGE DETECTION ALGORITHM***

• **Smoothing:** Smooth the input image with a Gaussian filter.

- If we need more detail of the edge, then the variance of the filter should be made large. If less image detail is required, then the variance of the filter should be small.

- Convolution of the image with Gaussian removed noises or blurred the image.

• **Gradient Operation:** Determine gradient magnitudes $M(x, y)$ and direction of edge normals $\phi(x, y)$.

• **Non-maximal Suppression:** For this, consider the pixels in the neighbourhood of the current image pixel (x, y). If the gradient magnitude in either of the neighbourhood pixels is greater than the current pixel, then mark the current pixel as a non-edge pixel.

- The range $(0 - 360^{\circ})$ of direction of edge normal is divided into eight equal portions. Two equal portions are considered as one sector, and so four sectors are defined. The gradient

direction $\phi(x, y)$ of an edge point is first approximated to one of these sectors. After getting the appropriate sector, the gradient magnitude of a point $M(x, y)$ is compared to gradient magnitudes $M1(x, y)$ and $M2(x, y)$ of two neighbouring pixels that fall on the same gradient direction. If $M(x, y)$ is less than either $M1(x, y)$ or $M2(x, y)$, then the value is suppressed as $M(x, y) = 0$; otherwise the value is retained.

• **Thresholding with hysteresis:** The idea behind hysteresis thresholding is that only large changes of gradient magnitudes are considered:

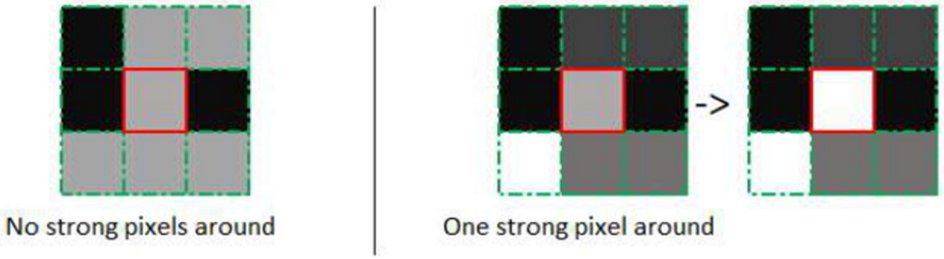
- mark all pixels with gradient magnitude $M(x, y) > T_H$ as the **strong edges**.

- mark all pixels with gradient magnitude $M(x, y) < T_L$ as **nonedges.**

- a pixel with $T_L < M(x, y) < T_H$ is marked as a weak edge.

• **Edge linking using hystersis:** After labeling the edge pixels, we have to link the similar edges to get the object boundary.

Based on the thresholding results, the hysteresis consists of transforming weak edges into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below:



Grayscale image

Gaussian blur

Sobel edge detection

Non-maximum suppression

Double threshold

Hysteresis result

Steps of calculating Canny edge detector

* To quantitatively ***evaluate the performance of various edge detectors***, we should formulate a criterion that may help in judging the relative performance under controlled conditions. We observe that in the response of an edge detector, there can be three types of errors:

(a) Missing valid edges

(b) Errors in localizing edges

(c) Classification of noise as edges

A figure of merit for an edge detector should consider these three errors.

One such figure of merit, called Pratt's figure of merit is:

$$FM = \frac{1}{\max(I_A, I_I)} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha d_i^2}$$

where I_A, I_i, d, and a are detected edges, the ideal edges, the distance between the actual and ideal edges, and a design constant used to penalize displaced edges, respectively.

Assignments: Design an edge detector based on Fuzzy set theory, and deep learning