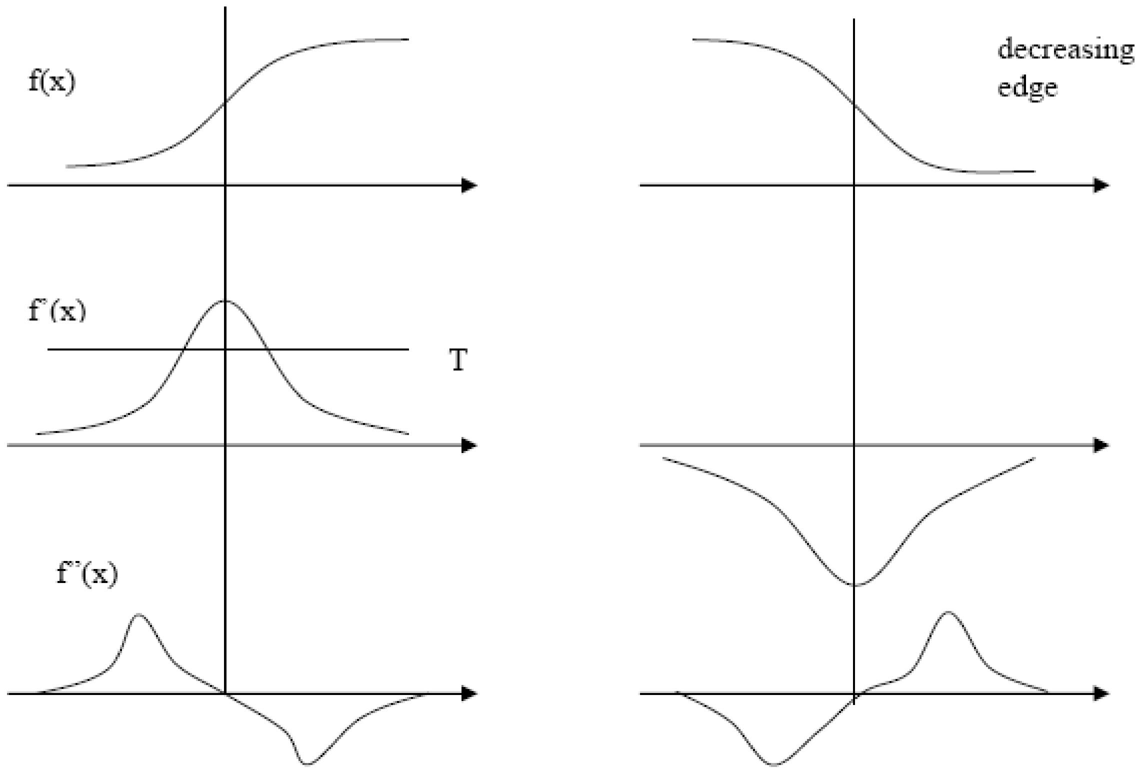
Chapter 10: Edge Detection

General Principles:

In a continuous 1-D signal *f(x),* an edge is defined as the point where the derivative of the signal *f'(x)* has an extremum, or equivalently where the second derivative *f"(x)* is zero, as shown below.



Task: Extend these concepts to:

i) two-dimensions, and
ii) discrete images.

The first can be accomplished by replacing the first and second derivatives with the gradient and Laplacian operators, respectively.

Approximating the first and second derivatives (partials) with the appropriate finite difference operators can do the latter.

Because the derivative (hence finite difference) operators are indeed high-pass filters, edge detection is very sensitive to noise. Two types of errors result due to noise:

1. **False positives:** Noise may generate many small peaks in the magnitude of the gradient resulting in false edges.
2. **False negatives:** Noise may result in shifts in true edge locations, resulting in missing actual edge pixels.

There are many different edge detection algorithms, which distinguish themselves on how they deal with these problems.

Gradient-Based Methods:

Gradient vector for a continuous image $f_c(x_1, x_2)$ is defined as:

$$\nabla f_c(x_1, x_2) = \begin{bmatrix} \frac{\partial f_c(x_1, x_2)}{\partial x_1} \\ \frac{\partial f_c(x_1, x_2)}{\partial x_2} \end{bmatrix} \text{ its magnitude: } |\nabla f_c(x_1, x_2)| = \sqrt{[\frac{\partial f_c(x_1, x_2)}{\partial x_1}]^2 + [\frac{\partial f_c(x_1, x_2)}{\partial x_2}]^2}$$

Techniques normally employ a thresholding mechanism:

$$|\nabla f_c(x_1, x_2)| \geq T$$

where T is a threshold, which is determined in many applications from the image database at hand.

Discrete Approximations:

Forward Difference: $\frac{\partial f_c(x_1, x_2)}{\partial x_1} \approx f(n_1+1, n_2) - f(n_1, n_2)$

Backward Difference: $\frac{\partial f_c(x_1, x_2)}{\partial x_1} \approx f(n_1, n_2) - f(n_1-1, n_2)$

Central Difference: $\frac{\partial f_c(x_1, x_2)}{\partial x_1} \approx \frac{1}{2}\{f(n_1+1, n_2) - f(n_1-1, n_2)$

Average Central Difference:

$$\begin{aligned} \frac{\partial f_c(x_1, x_2)}{\partial x_1} \approx \frac{1}{6}\{&[f(n_1+1, n_2) - f(n_1-1, n_2)] \\ &+[f(n_1+1, n_2-1) - f(n_1-1, n_2-1)] \\ &+[f(n_1+1, n_2+1) - f(n_1-1, n_2+1)]\} \end{aligned}$$

These computations can be interpreted as passing FIR filters, that is, 2-D convolution of an image with an impulse response. If $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$ denote the filter impulse responses that approximate the horizontal and vertical partials, respectively. Then, the gradient of a discrete image can be written as:

$$\nabla f_c(n_1, n_2) = \begin{bmatrix} f_{x_1}(n_1, n_2) \\ f_{x_2}(n_1, n_2) \end{bmatrix} = \begin{bmatrix} f(n_1, n_2) ** h_1(n_1, n_2) \\ f(n_1, n_2) ** h_2(n_1, n_2) \end{bmatrix}$$

The magnitude and the direction of the gradient vector at point n_1, n_2 are given by:

$$|\nabla f_c(n_1, n_2)| = \sqrt{[f_{x_1}(n_1, n_2)]^2 + [f_{x_2}(n_1, n_2)]^2}$$

$$\angle \nabla f(n_1, n_2) = \arctan\left(\frac{f_{x_2}(n_1, n_2)}{f_{x_1}(n_1, n_2)}\right)$$

With the following mask, we discuss some of the commonly used FIR filter kernels for $h_1(n_1,n_2)$ and $h_2(n_1,n_2)$. The **Prewitt kernel** represents the average central difference approximation. The kernels for the partials in x_1 and x_2 directions are given by:

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

These look like **Sobel kernel** (mask), where the latter applies twice the weight to the center row horizontally and vertically given by:

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Isotropic filters must not favor any particular edge direction. Prewitt and Sobel filters respond to diagonal edges differently than the horizontal and vertical edges because their coefficients do not take into account larger pixel distances in the diagonal directions. The Prewitt filter is less sensitive to diagonal edges than to horizontal and vertical ones, while the opposite is true for the Sobel filter.

Roberts Kernel:

0	1
-1	0

1	0
0	-1

Laplacian-Based Methods:

Laplacian of a continuous image is defined as the dot product of the gradient by itself:

$$\nabla^2 f_c(x_1,x_2) = \nabla \circ \nabla f_c(x_1,x_2) = \frac{\partial^2 f_c(x_1,x_2)}{\partial x_1^2} + \frac{\partial^2 f_c(x_1,x_2)}{\partial x_2^2}$$

The Laplacian is isotropic favoring no particular edge direction. In order to compute a discrete approximation to Laplacian, we can use the forward difference to approximate the first derivative,

$$\frac{\partial f_c(x_1,x_2)}{\partial x_1} \approx f(n_1+1,n_2) - f(n_1,n_2)$$

and then the backward difference to approximate the second derivative as the derivative of the first derivative:

$$\frac{\partial}{\partial x_1}\left[\frac{\partial f_c(x_1,x_2)}{\partial x_1}\right] \approx [f(n_1+1,n_2) - f(n_1,n_2)] - [f(n_1,n_2) - f(n_1-1,n_2)]$$
$$= f(n_1+1,n_2) - 2.f(n_1,n_2) + f(n_1-1,n_2)$$

When we place this and the corresponding one for the other variable x_2 to get an approximation to the Laplacian equation defined above:

$$\nabla^2 f_c(x_1,x_2) \approx f(n_1+1,n_2) + f(n_1-1,n_2) + f(n_1,n_2+1) + f(n_1,n_2-1) - 4.f(n_1,n_2)$$

This last equation for the Laplacian process is normally expressed as an FIR filter with an impulse response:

0	1	0
1	-4	1
0	1	0

However, depending upon the choice of derivative approximations other FIR filter kernels can also be obtained, such as:

1	1	1
1	-8	1
1	1	1

-1	2	-1
-2	-4	2
-1	2	-1

Laplacian of Gaussian (LoG) Filter: It is a Gaussian filter with the scale parameter σ:

$$h_c(x_1,x_2)=\frac{x_1^2+x_2^2-2\sigma^2}{\sigma^4}.\exp\{-\frac{x_1^2+x_2^2}{2\sigma^2}\}$$

Digital implementation of the **LoG** filter requires sampling $h_c(x_1,x_2)$ on a large enough support for a particular value of σ. It is important to note that the LoG filter is separable, hence can be efficiently implemented as a cascade of two 1-D filters.

Canny Edge Detection: Canny's edge detection procedure is among the most widely used employing elements of both gradient-based edge detection and Gaussian filtered scale space. It consists of the following steps:

1. Image smoothing with a Gaussian filter with a scale parameter σ.

$$h(n_1,n_2)=K.\exp\{-\frac{n_1^2+n_2^2}{2\sigma^2}\}$$

2. Find the image gradient, including magnitude and direction, at each pixel, using one of the operators, e.g., Roberts or Sobel.
3. Edge thinning by ***non-maximum suppression*:** Gradient direction is used to thin edges by suppressing any pixel response that is not higher than those of two neighboring pixels on either side of it along the direction of the gradient. The two 8-neighbors of a pixel that are to be compared are found by rounding off the computed gradient direction to one of 0, 45, 90 or 135 degrees.
4. Thresholding by gradient magnitude with ***hysterisis***. Two thresholds, an upper and a lower threshold, are defined for edge detection and edge following, respectively, where the upper threshold is two or three times the lower threshold. The edge detection is based on the upper threshold. However, once started a contour segment may be followed through pixels whose gradient magnitude exceeds the lower threshold.
5. A set of edge maps over a range of scales can be produced by varying σ. Fine-to-coarse feature synthesis is used to combine edges at different scales into a single edge map.

Color Edge Detection:

Let $f_Y(n_1,n_2),\ f_U(n_1,n_2),\ and\ f_V(n_1,n_2)$ denote the Y, U and V channels of a color image (a vector field), respectively. Color edge detection methods in the literature can be grouped as:

1. Detect edges in the luminance (Y) channel only;
2. Detect edges in each color channel independently and then merge the results heuristically;

3. Compute the gradient of each channel independently and then combine them as:
$$|\nabla f_Y(n_1,n_2)| + |\nabla f_U(n_1,n_2)| + |\nabla f_V(n_1,n_2)|$$
or
$$\sqrt{|\nabla f_Y(n_1,n_2)|^2 + |\nabla f_U(n_1,n_2)|^2 + |\nabla f_V(n_1,n_2)|^2}$$
4. Treat color image as a vector field, and compute the gradient of the vector field. The last approach has been shown to be less sensitive to noise than the sum of the magnitudes of the gradient of each channel computed independently, and take better advantage of the correlation between the color components [Lee and Cok][1].

The gradient of the discrete vector field can be approximated by:
$$D(n_1,n_2) = \begin{bmatrix} f_{x_1}^Y(n_1,n_2) & f_{x_2}^Y(n_1,n_2) \\ f_{x_1}^U(n_1,n_2) & f_{x_2}^U(n_1,n_2) \\ f_{x_1}^V(n_1,n_2) & f_{x_2}^V(n_1,n_2) \end{bmatrix}$$
where $f_{x_1}^Y(n_1,n_2)$, $f_{x_1}^U(n_1,n_2)$, and $f_{x_1}^V(n_1,n_2)$ denote the discrete approximations to partials of Y, U and V channels with respect to the horizontal variable x_1, respectively. In addition, a similar gradient is computed for the vertical variable x_2 as well. The square root of the largest eigenvalue of the symmetric, positive semi-definite matrix:
$$D^T.D(n_1,n_2) = \begin{bmatrix} p(n_1,n_2) & t(n_1,n_2) \\ t(n_1,n_2) & q(n_1,n_2) \end{bmatrix}$$
where individual terms are defined by:
$$p(n_1,n_2) = [f_{x_1}^Y(n_1,n_2)]^2 + [f_{x_1}^U(n_1,n_2)]^2 + [f_{x_1}^V(n_1,n_2)]^2$$
$$q(n_1,n_2) = [f_{x_2}^Y(n_1,n_2)]^2 + [f_{x_2}^U(n_1,n_2)]^2 + [f_{x_2}^V(n_1,n_2)]^2$$
$$t(n_1,n_2) = f_{x_1}^Y(n_1,n_2).f_{x_2}^Y(n_1,n_2) + f_{x_1}^U(n_1,n_2).f_{x_1}^U(n_1,n_2) + f_{x_1}^V(n_1,n_2).f_{x_2}^V(n_1,n_2)$$
and its corresponding eigenvector represent the magnitude and direction of the gradient, respectively. Note that all eigenvalues are real and nonnegative, $D^T.D$ is symmetric, positive semi-definite. From these, the magnitude of the gradient for the vector field can be computed as:
$$G(m,n) = \frac{1}{2}\left\{p(m,n) + q(m,n) + ([p(m,n)+q(m,n)]^2 - 4.[p(m,n).q(m,n) - t(m,n)^2])^{1/2}\right\}$$
and the resulting edge map can be obtained by thresholding $G(m,n)$.

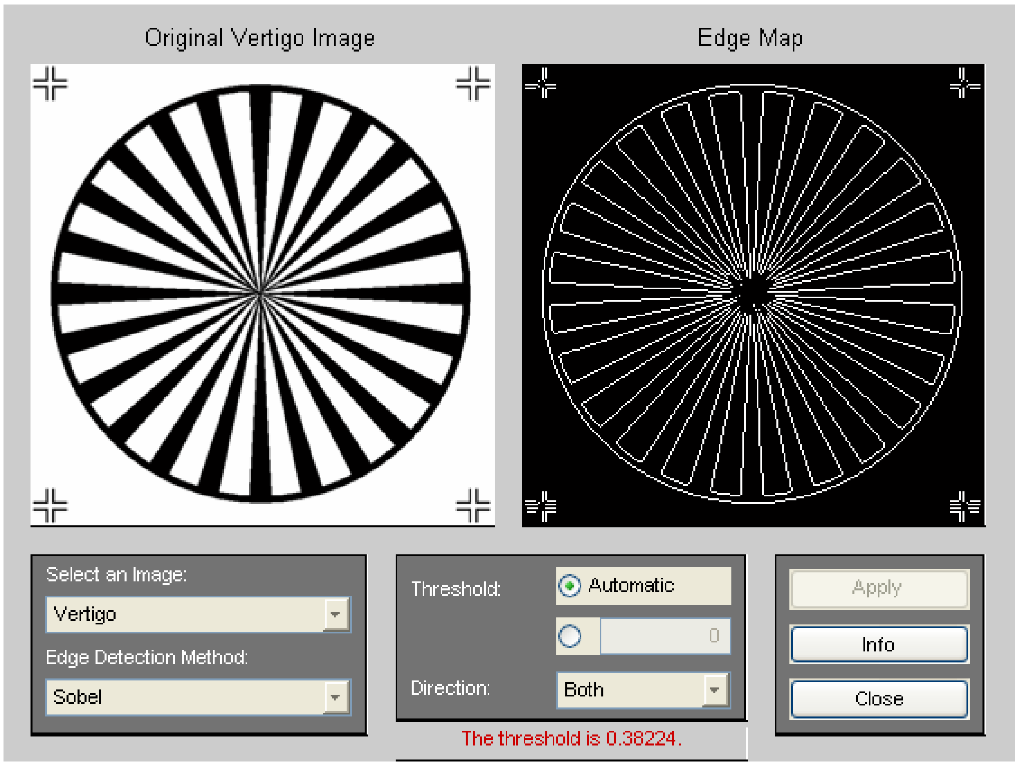
References

1. Handbook of Image and Video Processing, ed. Al Bovik, Academic Press, 2000.
2. L. Shapiro and G. Stockman, Computer Vision, Prentice Hall, 2001.
3. J. Canny, “A computational approach to edge detection,” IEEE Trans. on Patt. Anal. Mach. Intel., vol. 8, no. 6, pp. 679-698, 1986.

[1] H. Lee and D. Cok, “Detecting boundaries in vector field,” IEEE Trans. Signal Proc., vol. 39, no.5, pp. 1181-1194, May 1991.

Edge Detection Examples using demos in the Matlab Image Processing Toolbox.

Edge detection on Vertigo image using Sobel kernel with a threshold: 0.38224



Select an Image: Vertigo

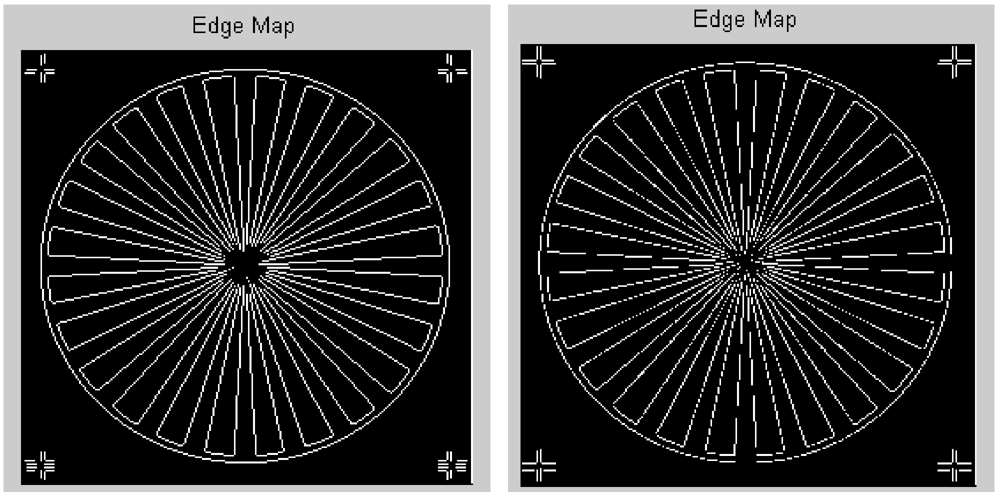
Edge Detection Method: Sobel

Threshold: [✓ Automatic] 0

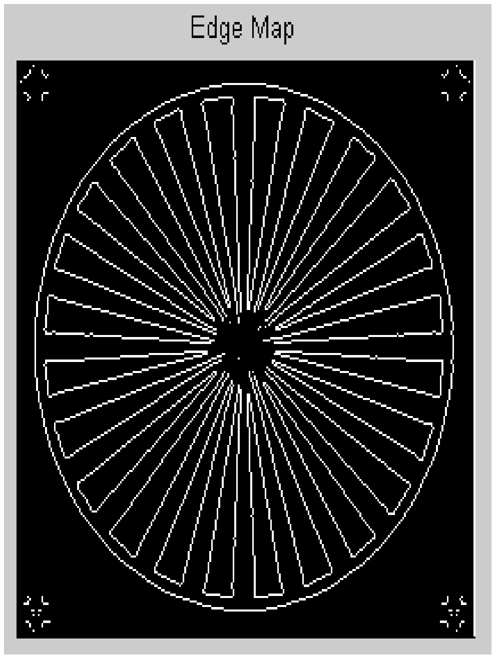
Direction: Both

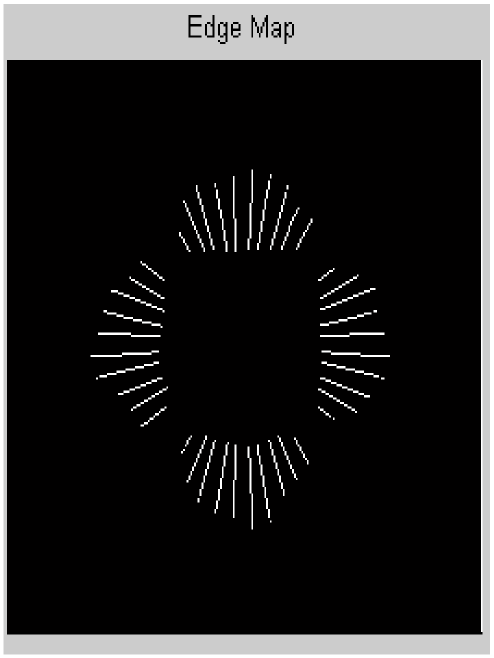
The threshold is 0.38224.

Edge detection on Vertigo image using Prewitt kernel with a threshold: 0.37413 and Roberts kernel with a threshold: 0.57818:

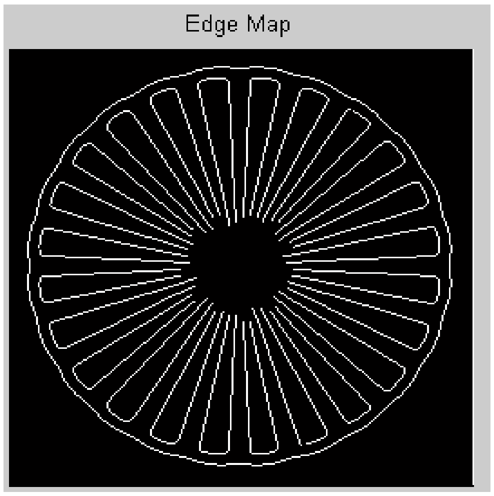


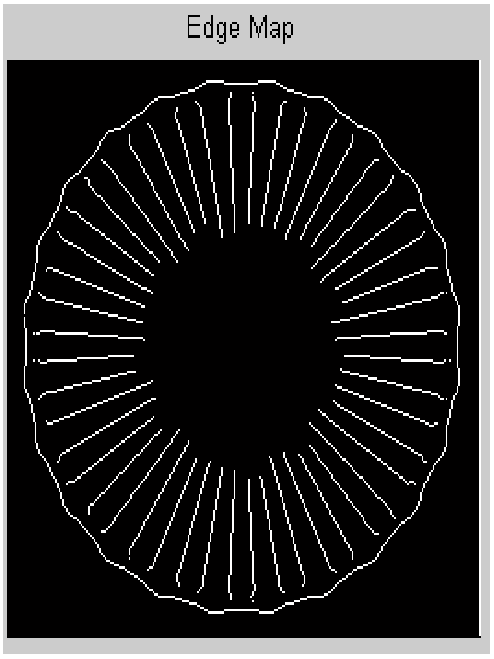
Edge detection on Vertigo image using LoG with sigma: 2 and a threshold: 0.02509, also with sigma: 4 and a threshold: 0.42188





Edge detection on Vertigo image using Canny with sigma: 2 and a threshold: 0.60938 and also with sigma: 4 and a threshold: 0.53125





Similar procedures will be tested with other images. Also Identifying Round Objects will be an integrated application of edge detection, Measuring Angle of Intersection.