Edge Detection

- Edge detection methods for finding object boundaries in images
- Edge detection is an image processing technique for finding the boundaries of objects within images.
- It works by detecting discontinuities in brightness.
- Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

Algorithms

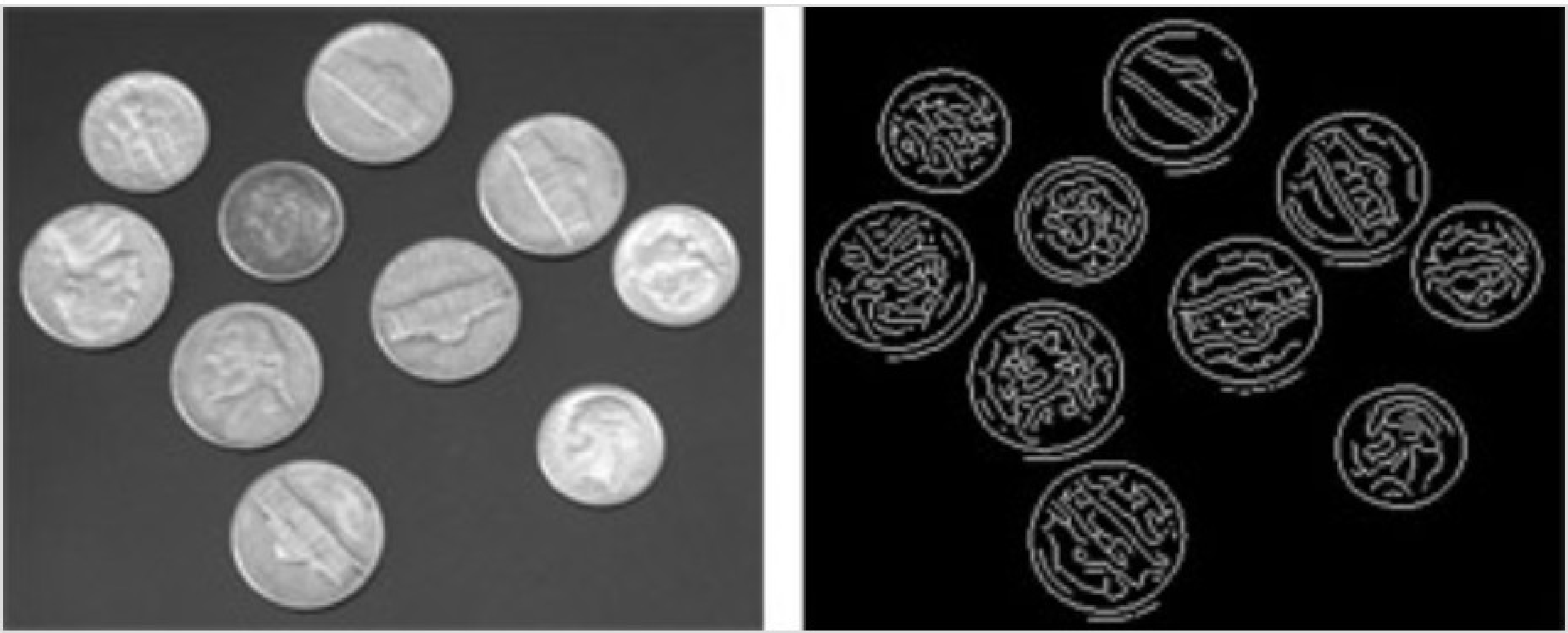
Common edge detection algorithms include:

- Sobel,
- Canny,
- Prewitt,
- Roberts, and
- fuzzy logic methods.

Image segmentation using the Sobel method.

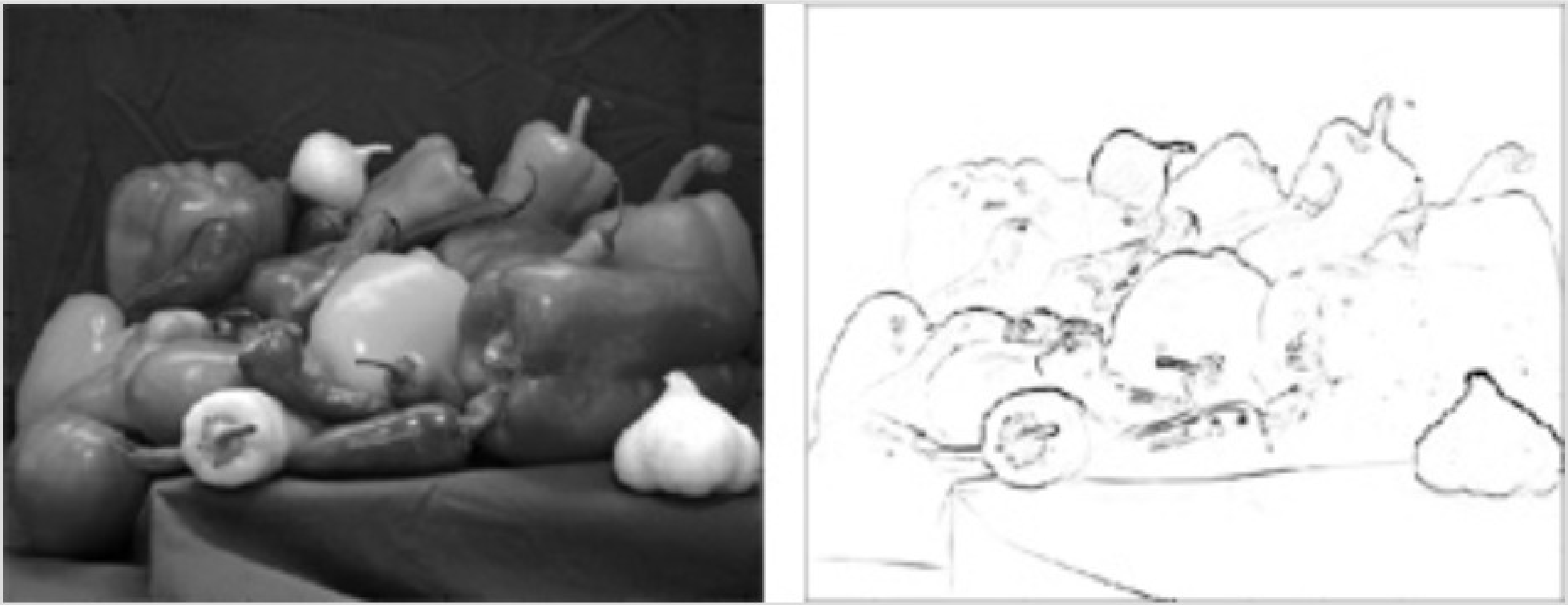
Matlab - Mathworks

Image segmentation using the Canny method.



Matlab - Mathworks

Image segmentation using a Fuzzy Logic method



Matlab - Mathworks

Edge Detection

- In an image, an edge is a curve that follows a path of rapid change in image intensity.
- Edges are often associated with the boundaries of objects in a scene.
- Edge detection is used to identify the edges in an image.

edge

To find edges, you can use the **edge** function. This function looks for places in the image where the intensity changes rapidly, using one of these two criteria:

- Places where the first derivative of the intensity is larger in magnitude than some threshold
- Places where the second derivative of the intensity has a zero crossing

edge

edge provides a number of derivative estimators, each of which implements one of the definitions above.

For some of these estimators, you can specify whether the operation should be sensitive to horizontal edges, vertical edges, or both. edge returns a binary image containing 1's where edges are found and 0's elsewhere.

Canny

The most powerful edge-detection method that edge provides is the Canny method.

The Canny method differs from the other edge-detection methods in that it uses two different thresholds (to detect strong and weak edges), and includes the weak edges in the output only if they are connected to strong edges.

This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.

Matlab Example

Read image and display it.

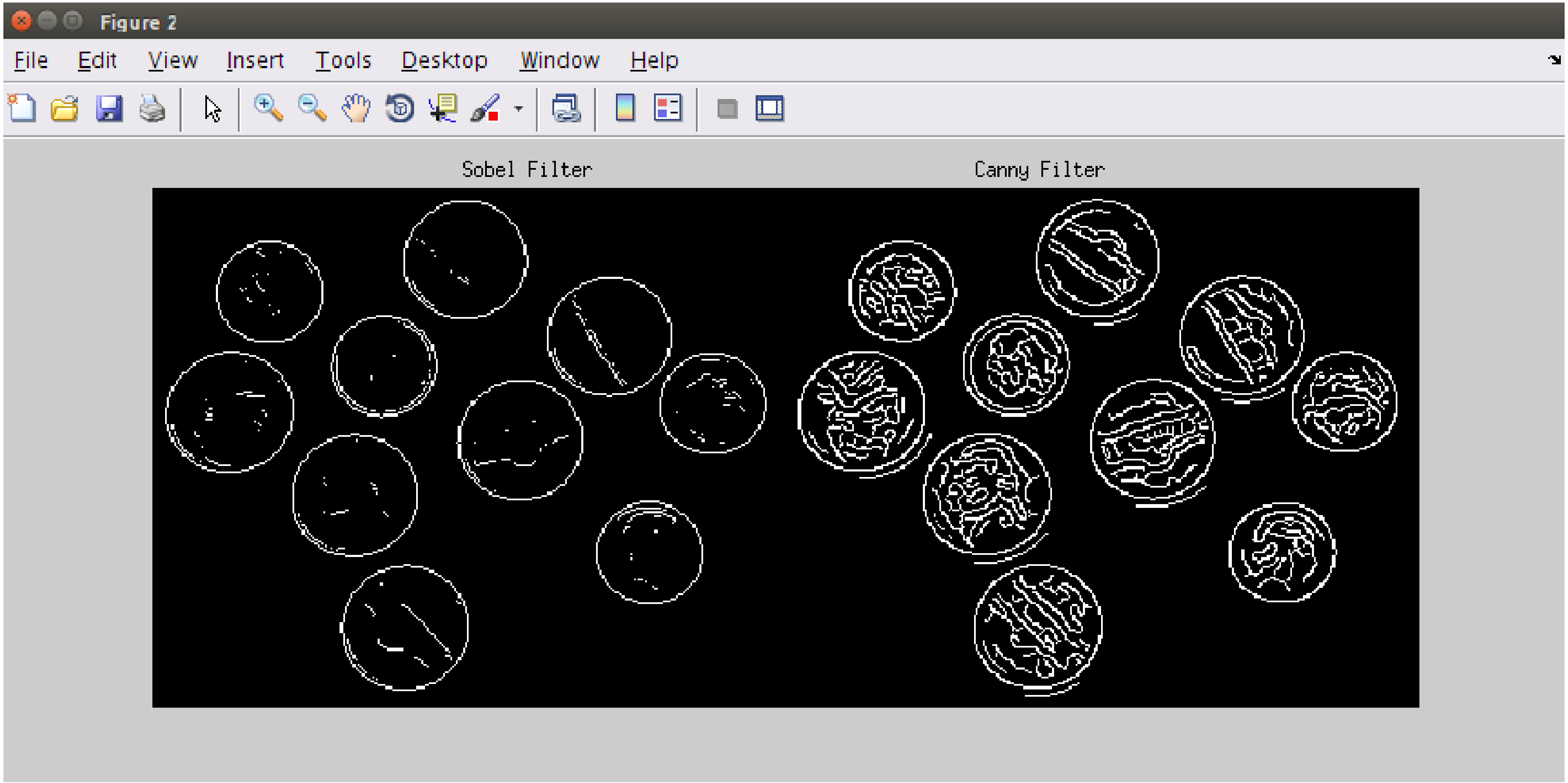
```
I = imread('coins.png');
imshow(I);
```

Compare Sobel & Canny

Apply both the Sobel and Canny edge detectors to the image and display them for comparison.

```
BW1 = edge(I,'sobel');
BW2 = edge(I,'canny');
figure;
imshowpair(BW1,BW2,'montage')
title('Sobel Filter                    Canny Filter');
```

Sobel vs. Canny



Matlab - Mathworks

Compare 4 methods

```
I = imread('cameraman.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'roberts');
BW3 = edge(I,'sobel');
BW4 = edge(I,'canny');
figure, imshow(BW1);
figure, imshow(BW2);
figure, imshow(BW3);
figure, imshow(BW4);
```

Kirsch Operator

The Kirsch operator or Kirsch compass kernel is a non-linear edge detector that finds the maximum edge strength in a few predetermined directions. It is named after the computer scientist Russell A. Kirsch.

Kirsch

The operator takes a single kernel mask and rotates it in 45 degree increments through all 8 compass directions: N, NW, W, SW, S, SE, E, and NE. The edge magnitude of the Kirsch operator is calculated as the maximum magnitude across all directions:

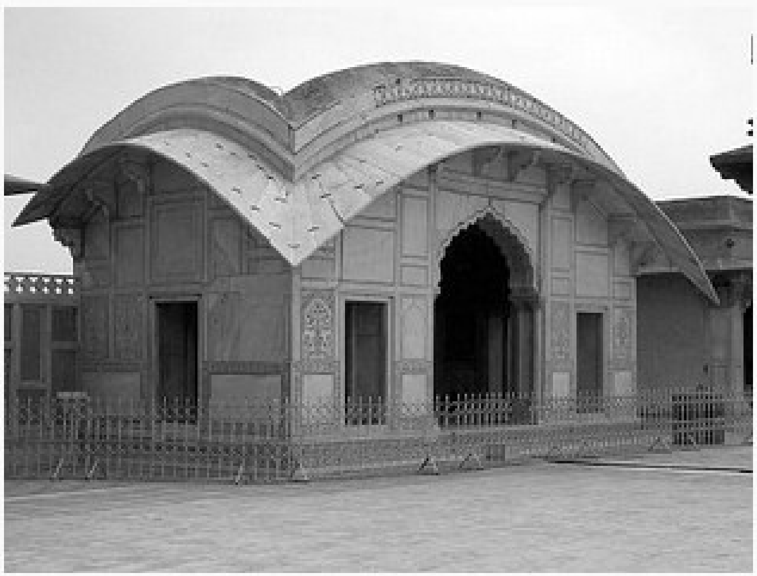
$$h_{n,m} = \max_{z=1,\ldots,8} \sum_{i=-1}^{1} \sum_{j=-1}^{1} g_{ij}^{(z)} \cdot f_{n+i,m+j}$$

z enumerates the compass direction kernels:

$$\mathbf{g^{(1)}} = \begin{bmatrix} +5 & +5 & +5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \mathbf{g^{(2)}} = \begin{bmatrix} +5 & +5 & -3 \\ +5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \mathbf{g^{(3)}} = \begin{bmatrix} +5 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & -3 & -3 \end{bmatrix}, \mathbf{g^{(4)}} = \begin{bmatrix} -3 & -3 & -3 \\ +5 & 0 & -3 \\ +5 & +5 & -3 \end{bmatrix}$$

The edge direction is defined by the mask that produces the maximum edge magnitude.

Kirsch Edges



Original

Maximum gradient in the 8 directions

Matlab - Mathworks

Some masks (Kirsch)

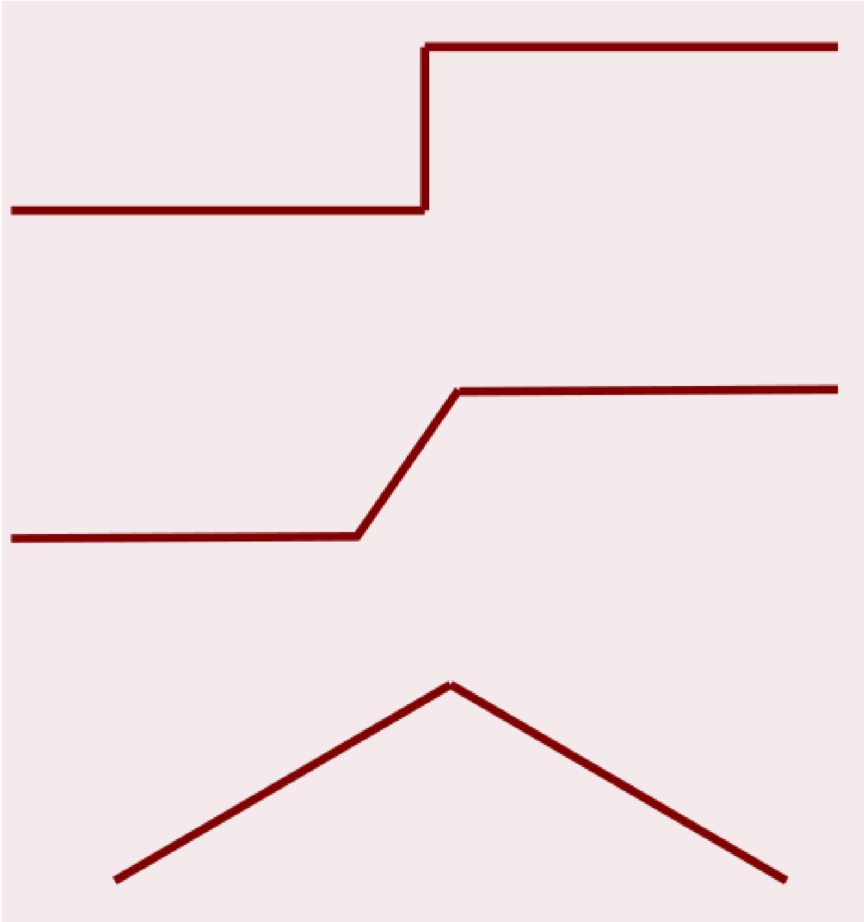
Image filtered with g1

Image filtered with g2

Image filtered with g3

Matlab - Mathworks

Edge Types



Ideal step Edge

Ramp Edge

Roof Edge

Matlab - Mathworks

Gradient

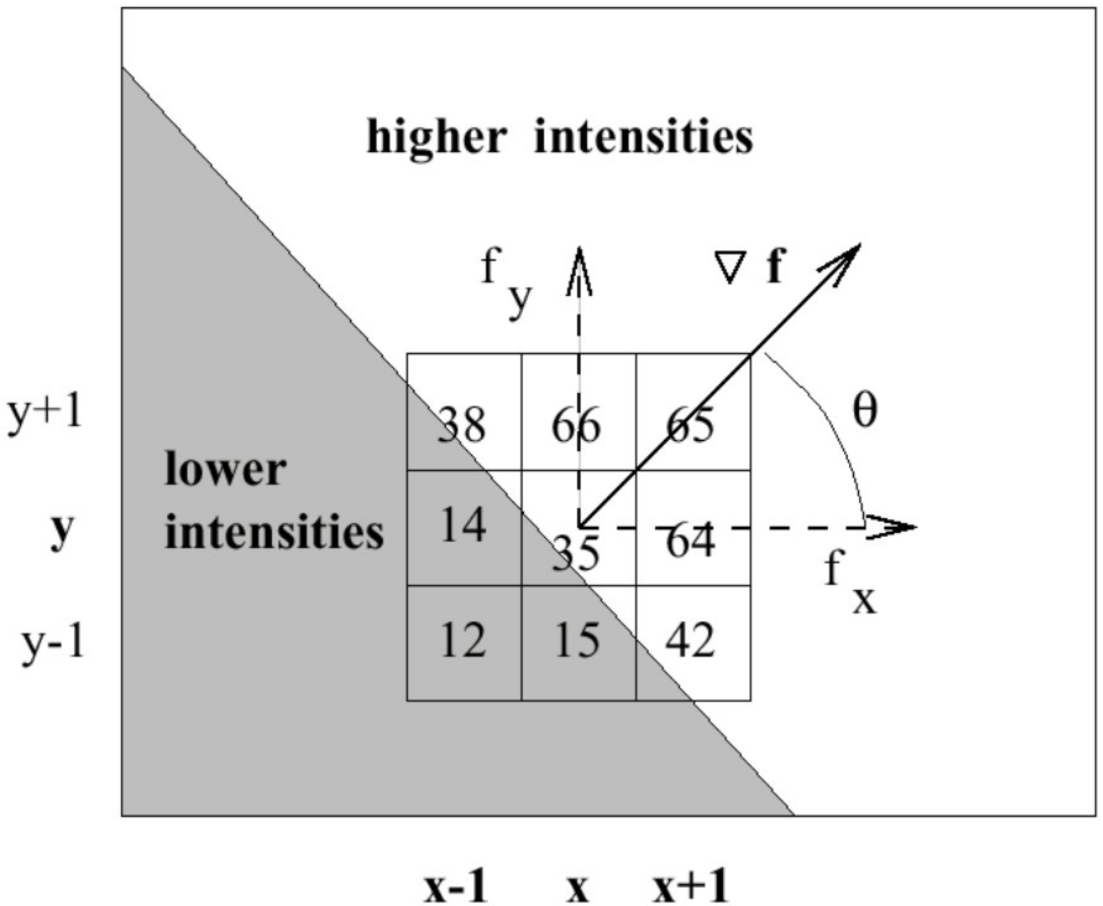
Intensity derivatives f(x,y) horizontally and vertically:

$$\nabla f(x,y) = G\{f(x,y)\} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

Size G and angle θ:

$$G_R = \left[G_x^2 + G_y^2\right]^{1/2} \qquad \boldsymbol{\theta = \tan^{-1} \frac{G_x}{G_y}}$$

Gradient



$$f_y = ((38-12)/2 + (66-15)/2 + (65-42)/2) / 3 = (13 + 25 + 11) / 3 = 16$$

$$f_x = ((65-38)/2 + (64-14)/2 + (42-12)/2) / 3 = (13 + 25 + 15) / 3 = 18$$

$$\theta = \tan^{-1} (16 / 18) = 0.727 \text{ rad} = 42 \text{ degrees}$$

$$|\nabla f| = (16^2 + 18^2)^{1/2} = 24$$

Matlab - Mathworks