

# Canny Edge Detection

## Goal

In this chapter, we will learn about

- Concept of Canny edge detection
- OpenCV functions for that : `cv.Canny()`

## Theory

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in

1. It is a multi-stage algorithm and we will go through each stages.
2. **Noise Reduction**

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

3. **Finding Intensity Gradient of the Image**

Smoothed image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction ( $G_x$ ) and vertical direction ( $G_y$ ). From these two images, we can find edge gradient and direction for each pixel as follows:

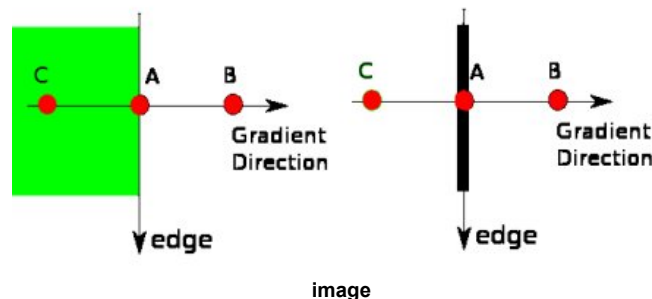
$$Edge\_Gradient (G) = \sqrt{G_x^2 + G_y^2}$$

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

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

4. **Non-maximum Suppression**

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

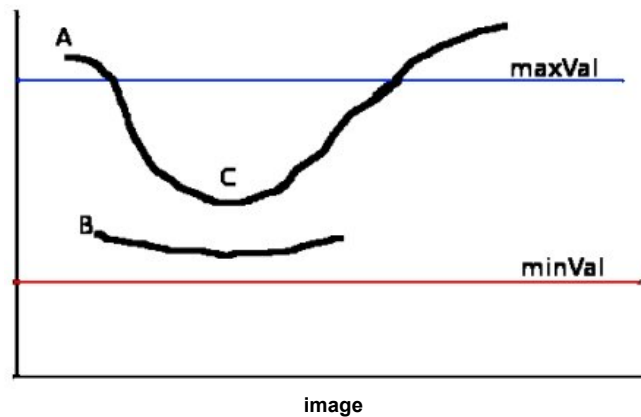


Point A is on the edge ( in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed ( put to zero).

In short, the result you get is a binary image with "thin edges".

5. **Hysteresis Thresholding**

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:



The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So what we finally get is strong edges in the image.

## Canny Edge Detection in OpenCV

OpenCV puts all the above in single function, `cv.Canny()`. We will see how to use it. First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Fourth argument is aperture\_size. It is the size of Sobel kernel used for find image gradients. By default it is 3. Last argument is L2gradient which specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate, otherwise it uses this function:  $Edge\_Gradient(G) = |G_x| + |G_y|$ . By default, it is False.

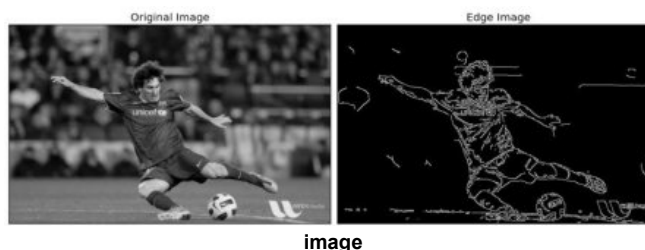
```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('messi5.jpg',0)
edges = cv.Canny(img,100,200)

plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([], plt.yticks([]))
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([], plt.yticks([]))

plt.show()
```

See the result below:



## Additional Resources

1. Canny edge detector at [Wikipedia](#)
2. [Canny Edge Detection Tutorial](#) by Bill Green, 2002.

## Exercises

1. Write a small application to find the Canny edge detection whose threshold values can be varied using two trackbars. This way, you can understand the effect of threshold values.