Objectives

In this reading assignment, you will be able to,

- Describe basics of Canny Edge Detection
- Understand brief history and motivation
- Explain each step of Canny algorithm
- Implement Canny edge detection algorithm

Prerequisites

To be able to understand this chapter properly, you are assumed to have following prerequisties.

- Understanding of basic calculus
- Basics of edge detection

Introduction

Canny edge detection algorithm is one of the most widely used edge detection algorithm in various image processing applications. It is also considered as the optimal edge detection algorithm that uses multi-stage approaches to detect variety of edges in the image.

Previously, we discussed about the criteria for optimality of edge detector algorithm. Canny, the inventor of this algorithm, followed the same criteria to improved edge detection algorithm and create this elegant algorithm - Canny edge detector.

History and Motivation

Canny edge detector was developed by John F. Canny in 1986. He published a paper A Computatioanal Approach to Edge Detection (https://www.scribd.com/document/345205255/A-Computational-Approach-To-Edge-Detection-j-Canny-pdf) explaining how and why this technique works.

In the previous chapter, we understand the limitation of classical edge detectors i.e. they are sesitive to noise. Since classical operators follow derivative and smoothing properties separately, they won't produce good results under noisy conditions. In contrast, Canny edge detector combines both properties in suitable way to produce good results. In Canny edge detection algorithm, there are three adjustable parameters. They are,

- Width of Gaussian
- Low threshold
- High threshold

By adjusting these parameters, the algorithm follows the following criteria step by step as mentioned above.

- Good detection or Low error rate
- Good localization
- Single response

Canny Edge Detection algorithm

The algorithm consists of 5 steps. It only works once the image is converted to grayscale one. So, grayscale conversion is not included in this algorithm. In the figure below, input image refers to the grayscale image.

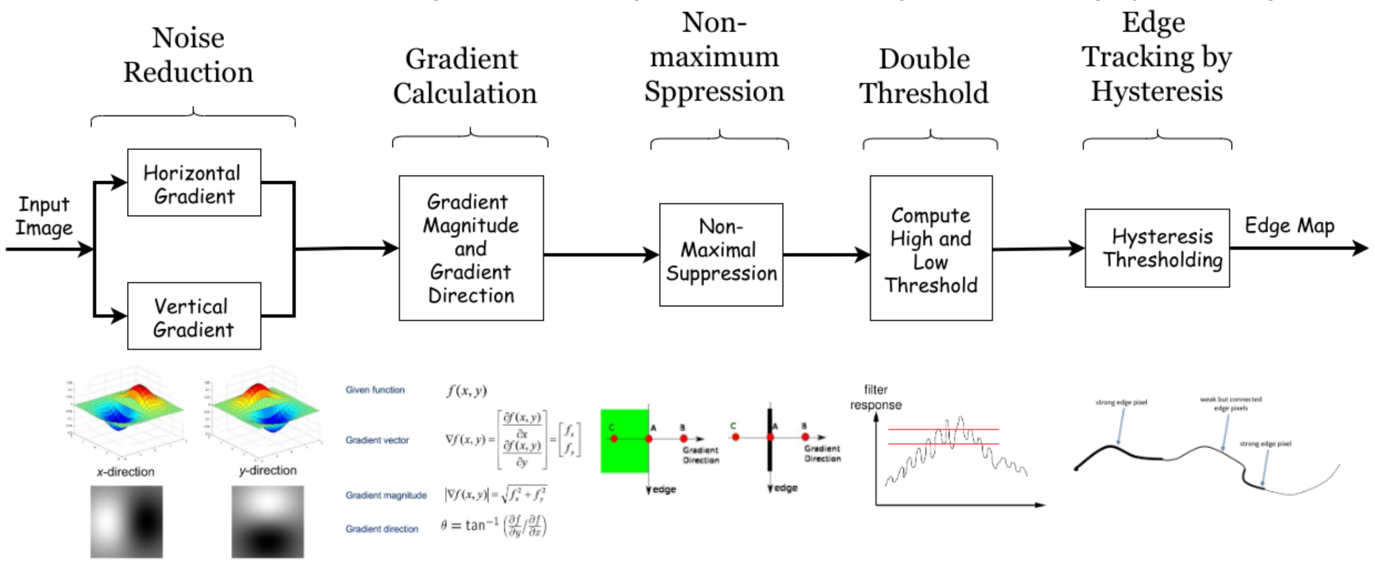


Fig. 1: Diagram of Canny Edge Detector

The above figure shows 5 steps of the algorithm. We will describe each step in detail in further sections. These steps can be given as,

Step 1: Noise Reduction

Step 2: Gradient Calculation

Step 3: Non-maximum Suppression

Step 4: Double Threshold

Step 5: Edge Tracking by Hysteresis.

Step 1: Noise Reduction

The main purpose of noise reduction step is to reduce noise as much as possible. One way to supress these noises is to use gaussian for smoothing. Please notice that the gaussian blur is not used separately here. This means derivatives of this gaussian blur is used for smoothing.

What is the difference then? May be this question have arouse in your mind, right? In classical operator, at first, the gaussian blur is used then, the derivative is taken. However, this method fails when there is much noise in the image so the classical operators fails in noisy images.

In order to overcome this problem, Canny used derivatives of gaussian blur to supress the noise of the image. The derivative of gaussian in x and y direction can be shown in the following figures.

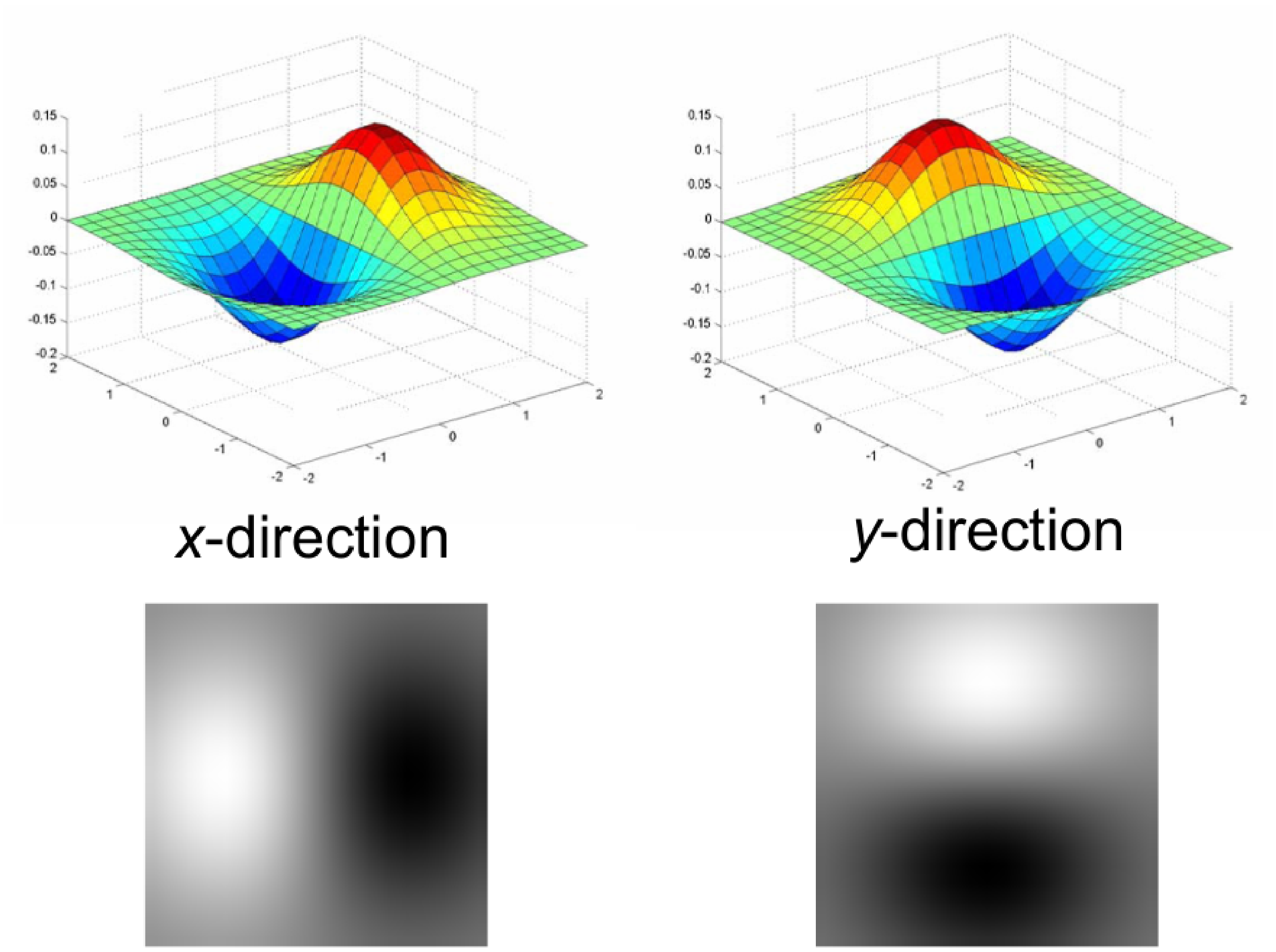


Fig. 2: Diagram of DoG in X and Y direction

The image is convolutes with these derivative of gaussian blurs to suppress noises.

Step 2: Gradient Calculation

As we know, edges are the areas where pixels' intensity changes abruptly. One way of detecting these edges is to track these intensity changes.

In this gradient calculation step, the same is done. This calculation can be given as following steps,

1. Suppose, $f(x, y)$ is the image function. Firstly, it's gradient in x and y direction is calculated.
2. After that, magnitude and direction of the image function are calcuated.

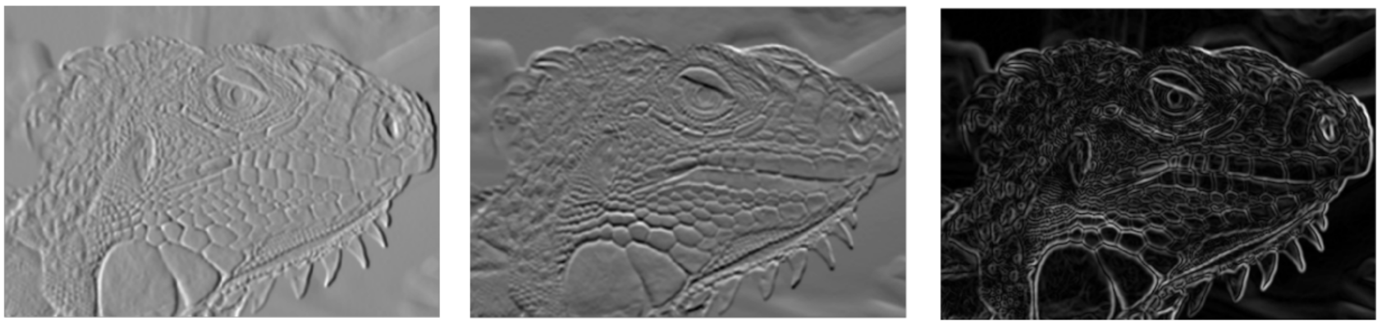
Mathematically, the above two steps can be shown as following, Given function $$f(x, y)$$

Gradient vector $$\nabla f(x, y) = \begin{bmatrix} \dfrac{\partial f(x, y)}{\partial x} \\ \dfrac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude $$\left|\nabla f(x, y)\right| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Fig. 3: Gradient calculation



X-Derivative of Gaussian Y-Derivative of Gaussian Gradient Magnitude

Fig. 4: Image in the different stages of DoG

In the above figures, you can see the differences among the images. In the first figure, only x-direction features are captured and similarly, in second figure, only y-direction features. However, the gradient magnitude step tries to capture all features of x and y and produces better results the previous images.

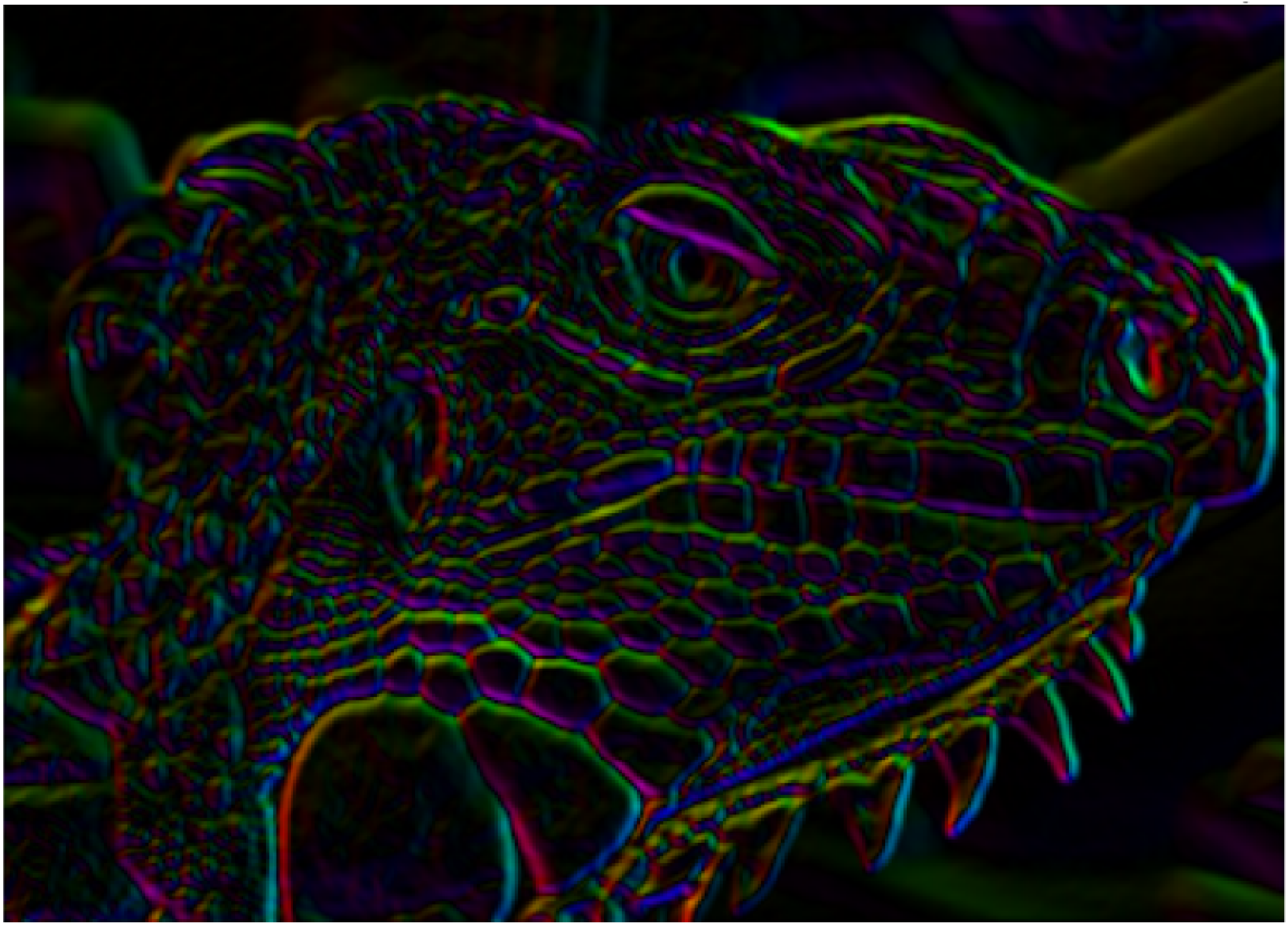


Fig. 5: Image after applying gradient direction step

The figure 4 represents the image after applying gradient orientation step. Here, different colors, such as green, red, voilet, represent the direction of each pixels' gradient.

Step 3: Non-Maximum Suppression

Edges represent the boundary of the objects or surfaces by continuous lines or curves. The thickness of these curves or lines doesn't give much information about the edges. So, it is good idea to make edge as thin as possible.

Here, non-maximum suppression step does the same. In this step, the pixel with the maximum intensity in an edge is identified. The figure below illustrates how non-maximum suppresion works.

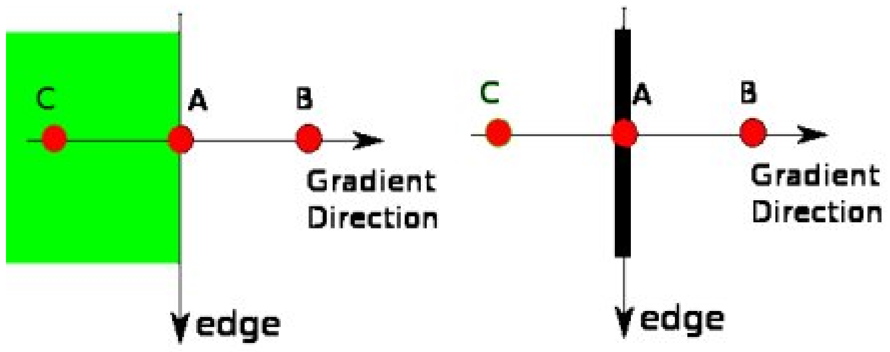


Fig. 6: Non-maximum suppression

In more detail, let's consider the figure below. Point q represents the pixel on the edge whose intensity is greater than intensity of both p and r pixels. In this condition, the point q is kept otherwise, the pixel value is set to 0 (i.e., make black pixel).

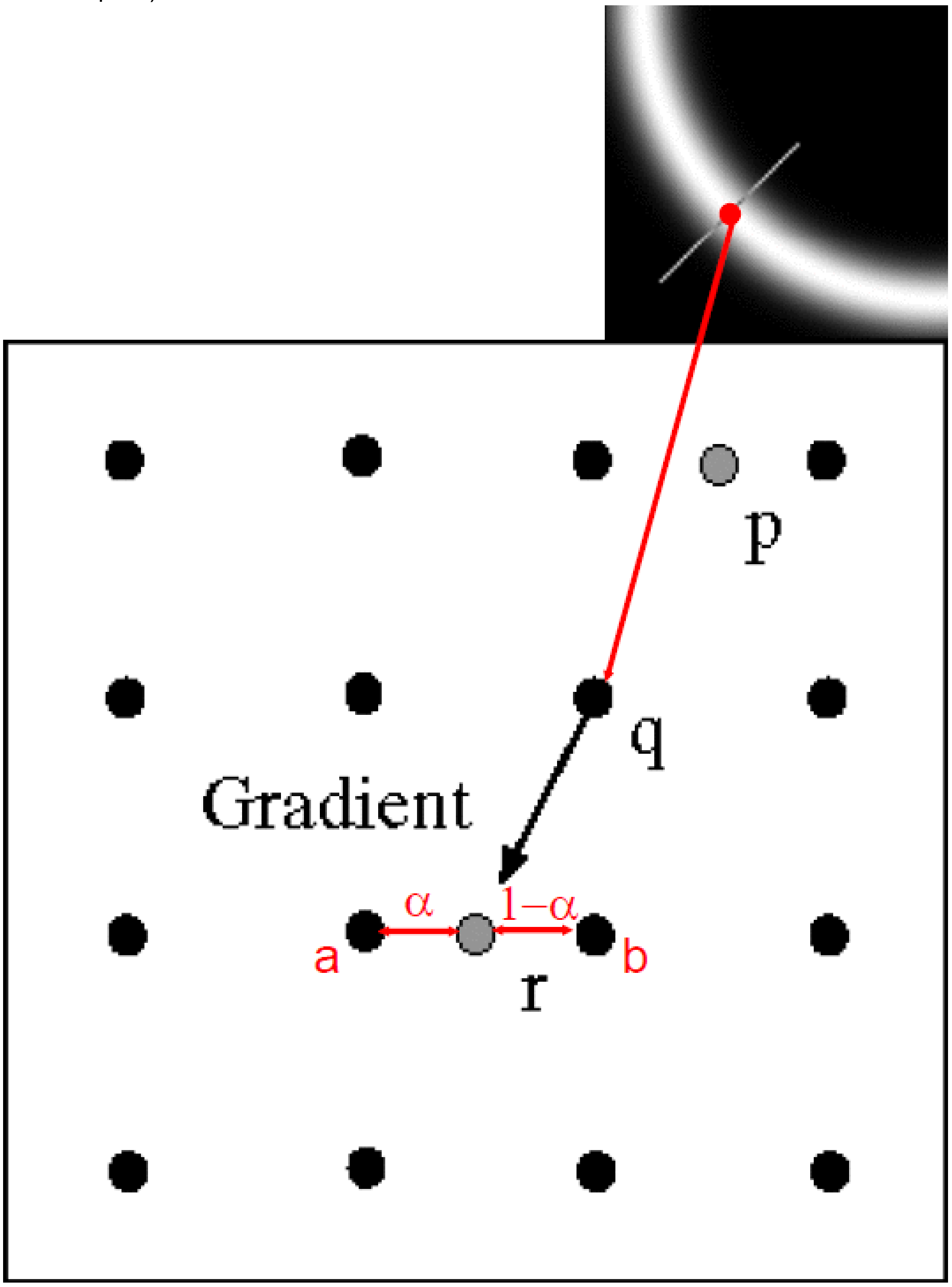


Fig. 7: Non-maximum suppression

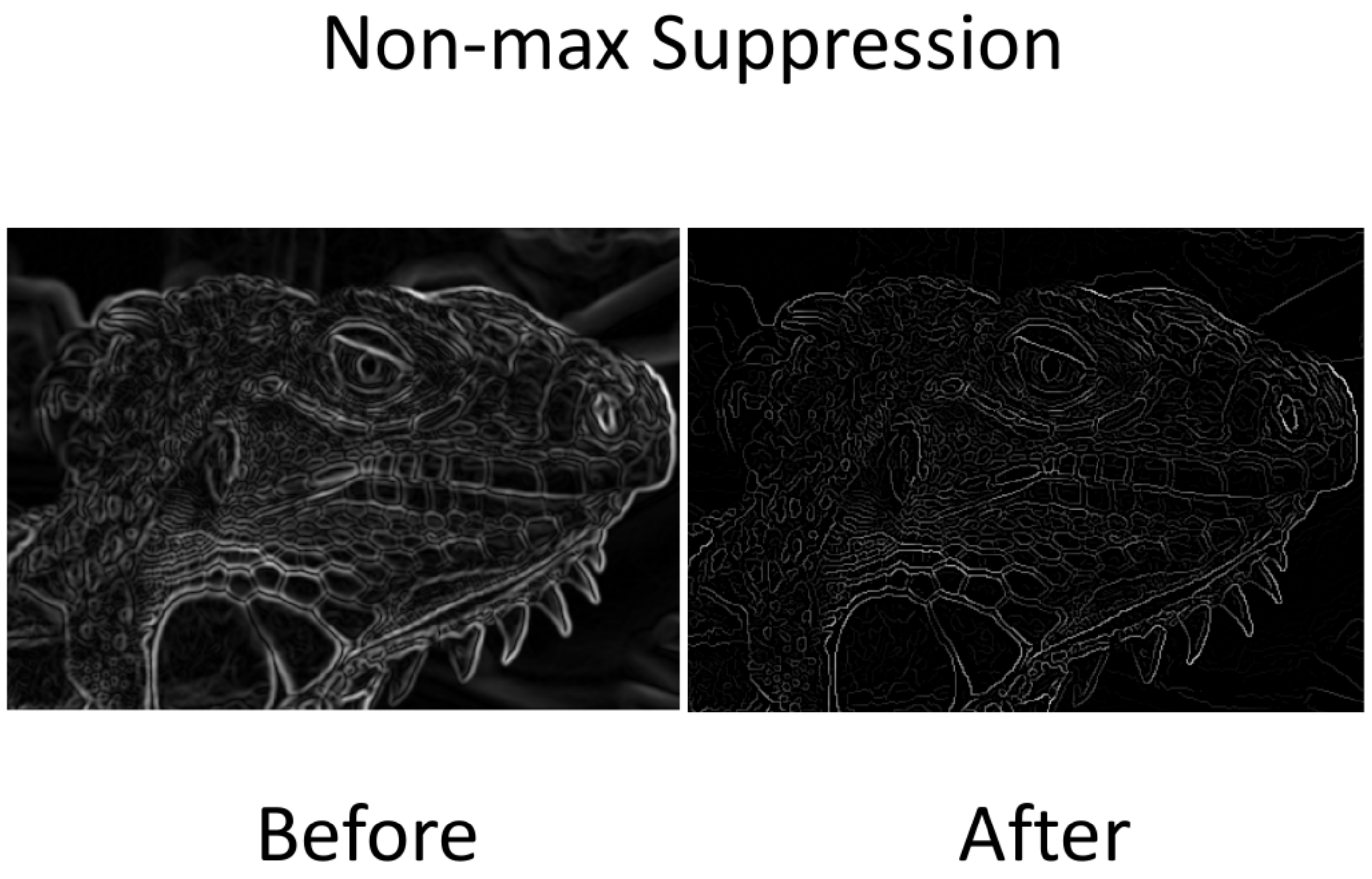


Fig. 8: Image after applying non-maximum suppression

In above figure shows, how an image looks like once we apply non-maximum suppression step. You can clearly see the edges are thinner than the previous image and better representation of the image than previous one.

Step 4: Double Threshold

Still we can't find all the edges of the actual image. There are some missing edges and some redundant edges too due to noises. Step 4 and and step 5 are represents some most amazing concepts in the canny edge algorithm to find strong and weak edges.

In double threshold step, two threshold set i.e., high and low. Suppose, we set high threshold 0.7 and low threshold 0.3. Then, all the pixels with a value larger than 0.7 are strong edge and value less than 0.3 are not considered as an edge and we set these pixels value as 0. The values in between 0.7 and 0.3 could be weak edges. However, we don't know these are true edges or not at all. The step 5 will describe how to determine weak edges as true edges or not.

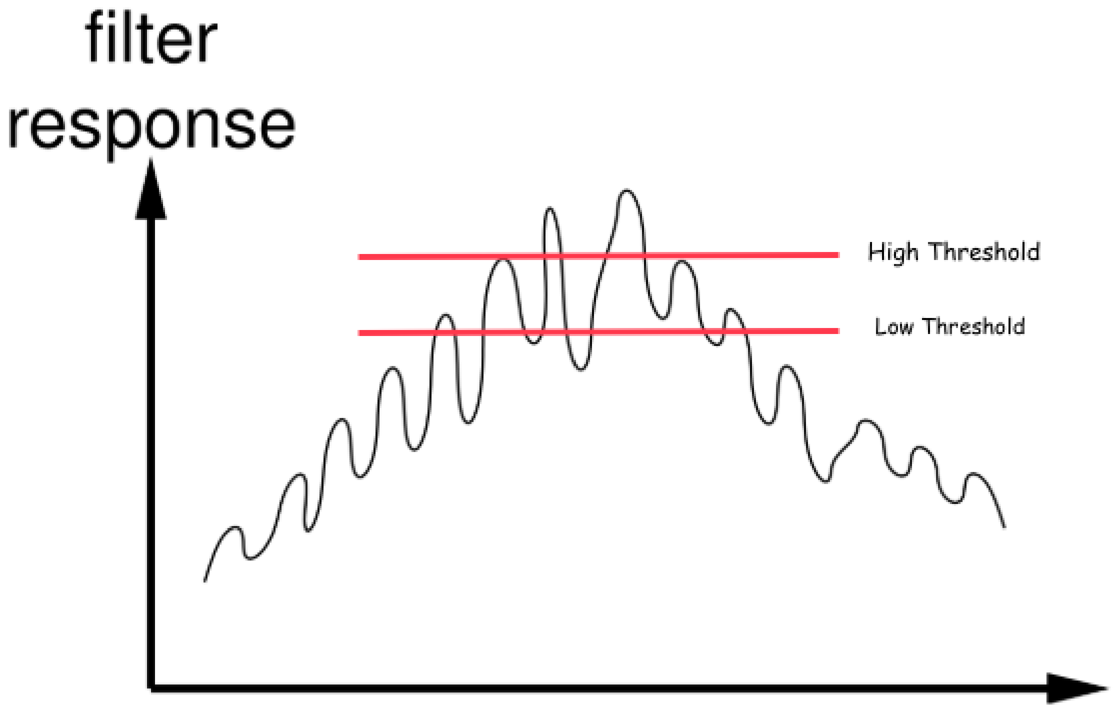


Fig. 9: Computing high and low threshold

In the figure above, the two red lines represent high and low threshold respectively. Practically, it was found that setting threshold ratio rather than particular threshold value will work better.

Step 5: Edge Tracking by Hysteresis

So far, we have a concept of strong edges and weak edges. Now, we will figure it out which weak edges are actually true edges. For this, Canny invented very elegant step which is known as edge tracking by hysteresis step.

The basic idea behind this step is that the weak edges connected to strong edges are true edges. In contrast, the weak edges which are not connected to strong edges are not considered as true edges and are removed. The figure below illustrates the weak edge between two strong edges and is true edge.

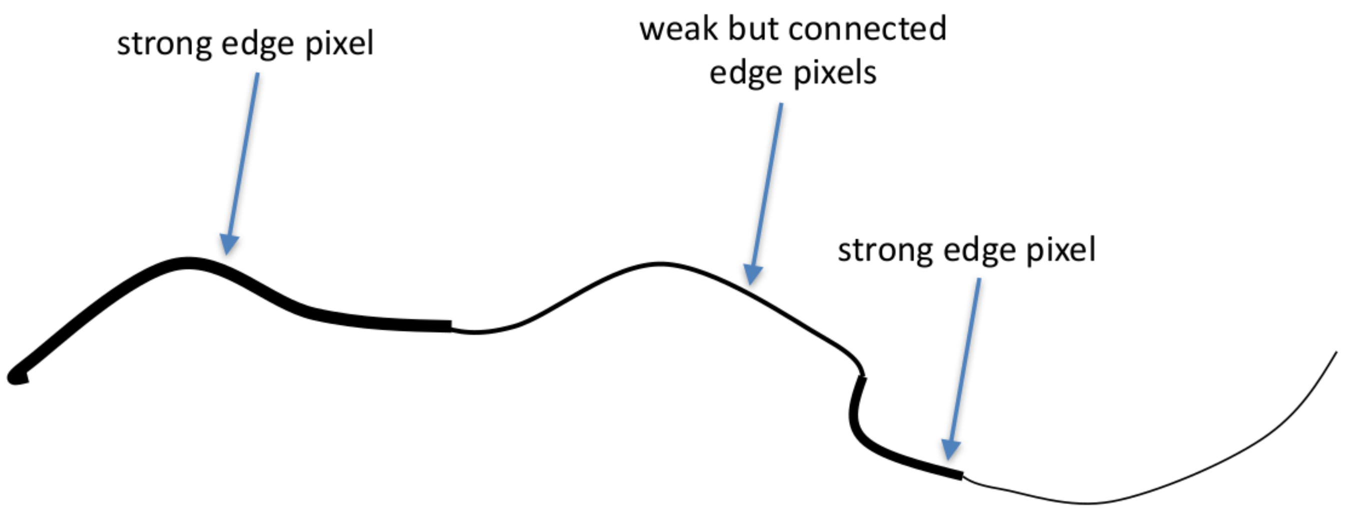


Fig. 10: Edge tracking by hysteresis

Final Canny Edges

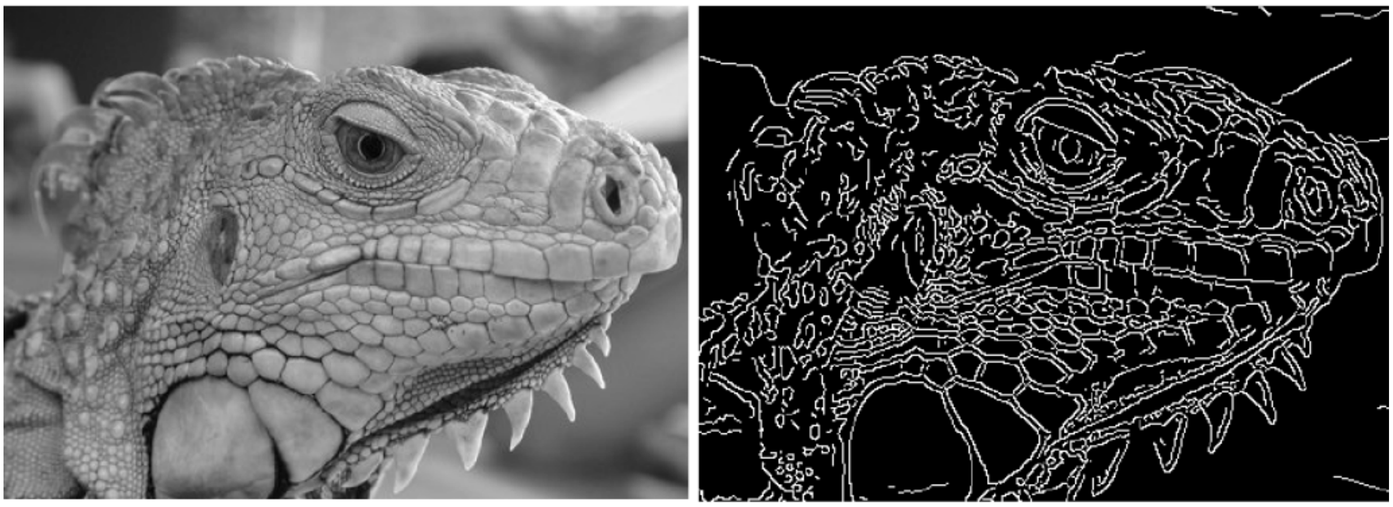


Fig. 11: Edge tracking by hysteresis

That's all. You can see how Canny edge detection algorithm works. In the above figure, left image is the actual lizard image and right image is image after applying canny edge detection algorithm. You can clearly see how well the algorithm detects the edges.

Comparison

The figure below show output images after applying canny edge detector with different values of σ. The lesser the σ more finer details in the output which may subjected to the noise. The higher the value of σ, more smoother will be the output image and true edges may be missing.

Effect of σ (Gaussian kernel spread/size)

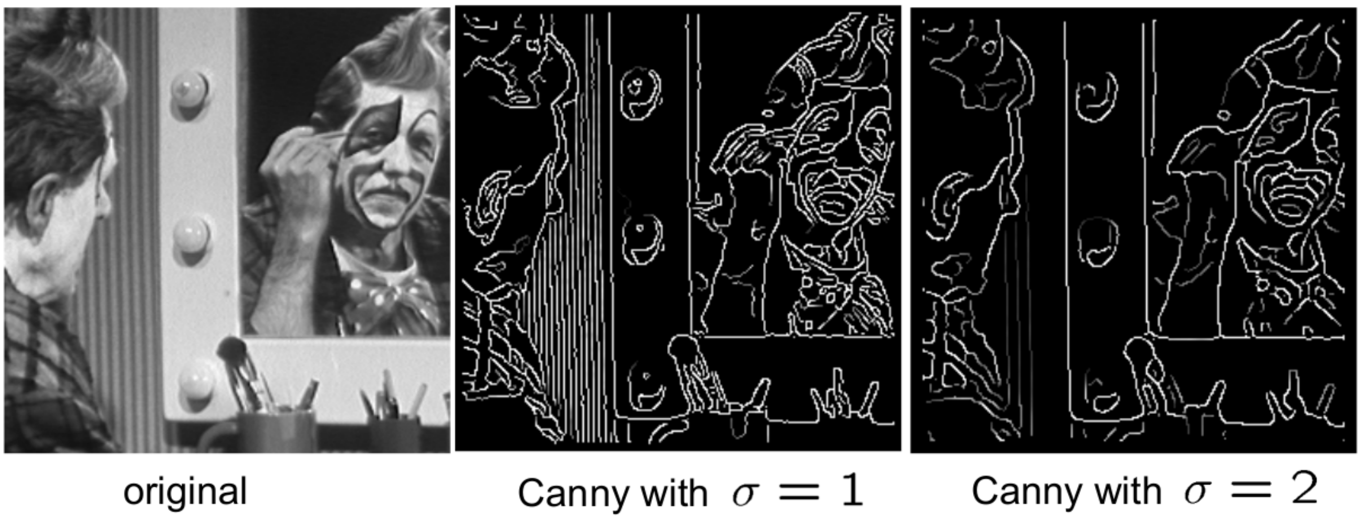


Fig. 12: Edge tracking by hysteresis

The figure below shows the performance comparison of sobel operator, Laplacian of Gaussian (LoG) operator and canny edge detector on the bird image.

You can see, sobel operator misses many true edges. LoG operator performs better compared to sobel although it misses few edges too. Canny edge detector detected all the true edges and also subjected to few noises you can see. However, the performance of canny edge detector can be improved by varying the parameters such as width of guassian, high and low thresholds.

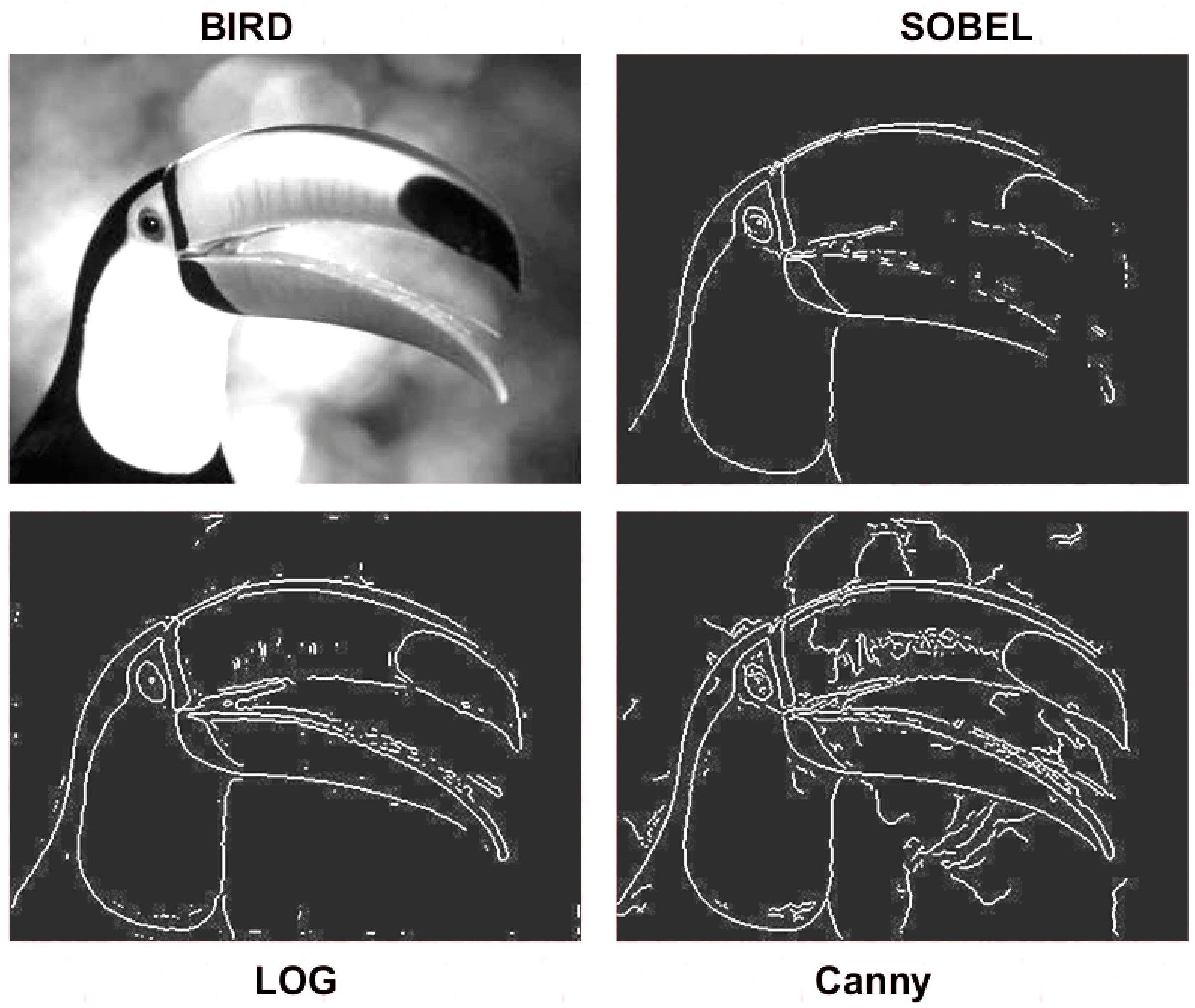


Fig. 13: Edge tracking by hysteresis

Implementation

We understand what is happening in each step in the canny edge detection algorithm. Now, let's implement this algorithm to see how it detects edges from the image.

We implement this algorithm using `opencv` library as follows,

In [ ]:

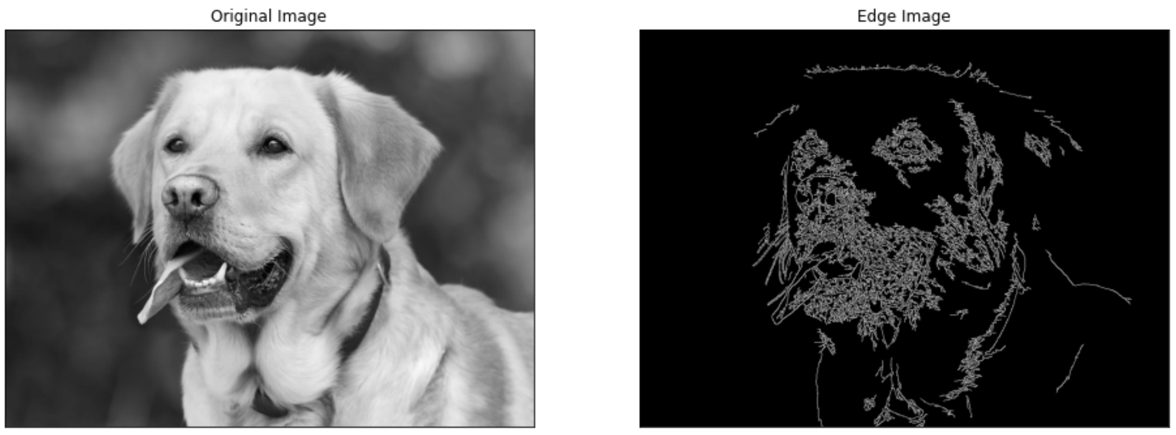
```
# Importing libraries
import cv2
import numpy as np
from matplotlib import pyplot as plt

# Load image
img = cv2.imread('dog.jpg', 0)

# Implement Canny edge detector algorithm in the image
edges = cv2.Canny(img, 30, 250)

plt.figure(figsize=(16, 10))
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()
```



As you can see how well the algorithm detects edges. That's all for this chapter.

Key takeaways

- Why canny edge detector?
  - To detect edges by applying combination of derivative and gaussian blur and removes much noises
- Canny flows edge detection criteria,
  - Low error rate
  - Good localization
  - Single response
- What are canny's parameters?

    - Width of Gaussian
    - High threshold
    - Low threshold
- What are the steps in canny algorithm?
    - Noise suppression: Suppress noises using DoG blurring
    - Gradient calculation: Calculate magniture and direction of each pixel gradient
    - Non-maximum suppression: Make edges thinner by keeping maximum intensity pixels only
    - Double threshold: Set high and low threshold to classify strong and weak edges
    - Edge tracking by hyteresis: Identify weak edges and discard the all the other pixels which are not weak edges.

References

- Papers
    - John C. (1986), A Computational Approach to Edge Detection (https://ieeexplore.ieee.org/document/4767851)
        - Refer this paper to understand Canny edge detection mathematically.
- Books
    - Richard S. (2010), Computer Vision: Algorithms and Applications (http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf), 2010 Draft, Springer
        - Check pages 215 - 218 to explore more about edge linking in more detail.
- University Lectures
    - Juan C. & Ranjay K., Edge detection (http://vision.stanford.edu/teaching/cs131_fall1718/files/05_edges.pdf)
        - Check page no. 114 to see the 45 years of boundary progression.