A Developer Diary

{about:"code learn and share"}

May 19, 2019 By Abhisek Jana — 4 Comments (Edit)

How to implement Sobel edge detection using Python from scratch

Sobel edge detection is one of the foundational building block of Computer Vision. Even when you start learning deep learning if you find the reference of Sobel filter. In this tutorial we will learn How to implement Sobel edge detection using Python from scratch.

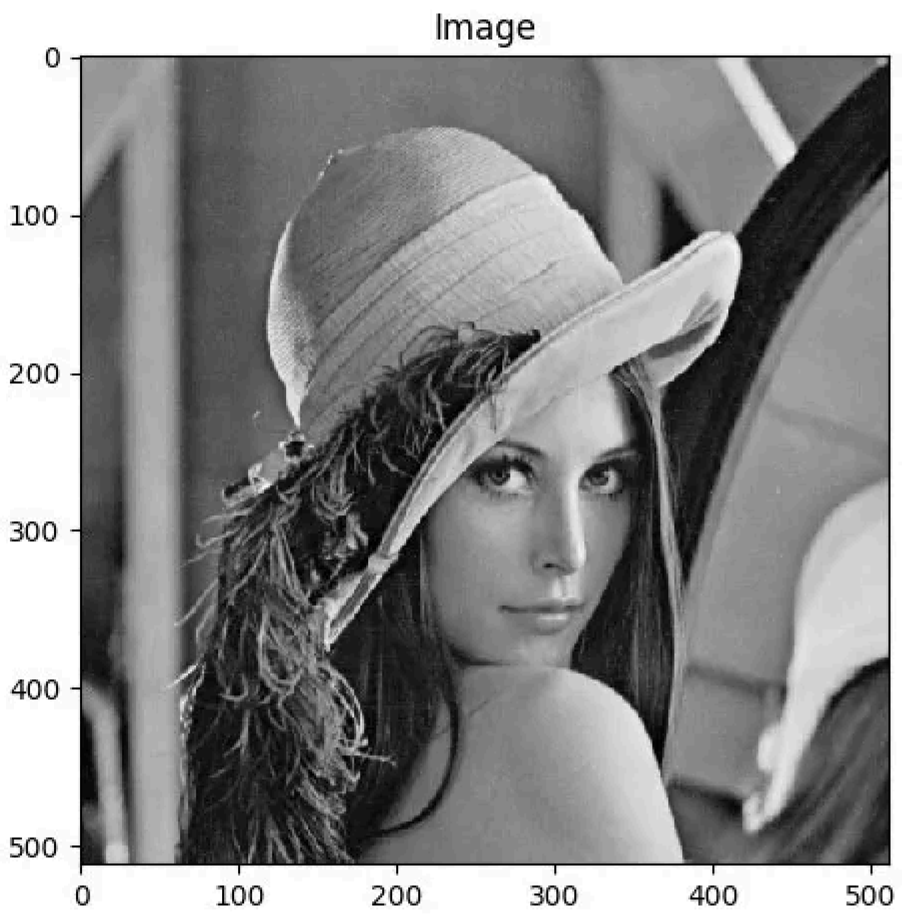
We will be referring the same code for the Convolution and Gaussian Smoothing function from the following blog.

Applying Gaussian Smoothing to an Image using Python from scratch

Using Gaussian filter/kernel to smooth/blur an image is a very important tool in Computer Vision. You will find many algorithms using it before actually processing the image. Today we will be Applying Gaussian Smoothing to an image using Python from scratch and not using library like OpenCV. High Level Steps: There are two steps to … Continue reading

A Developer Diary 6

The objective will be to find the edges in the below image:



What is an edge?

An edge is a place of rapid change in the image intensity function.

How to detect an edge?

In order to detect edge we need to detect the discontinuities in image and we know that we can use derivative to detect discontinuities.

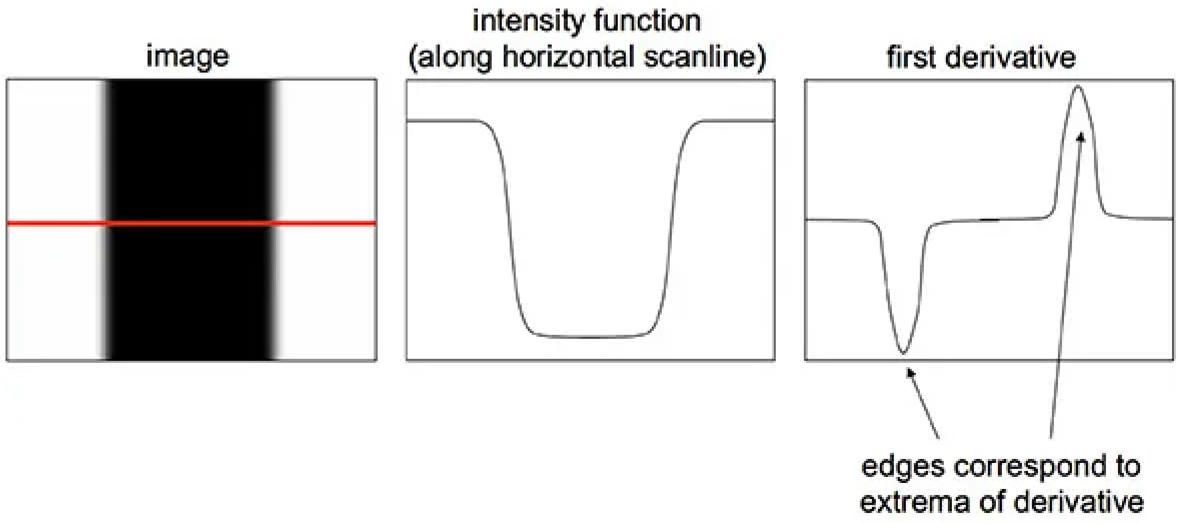


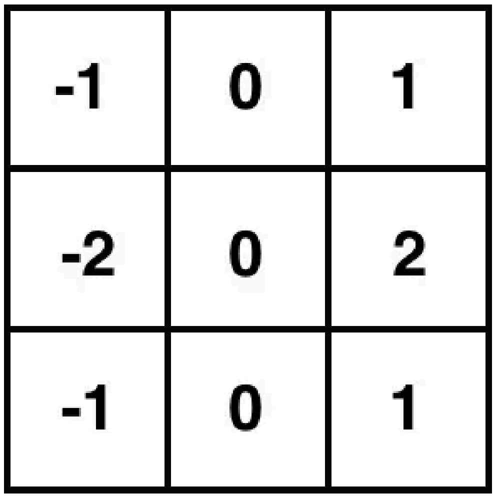
Image Credit: http://stanford.edu/

As you are seeing in the above picture, the edges corresponds to the derivatives. Since images are discrete in nature, we can easily take the derivate of an image using `2D derivative mask`.

However derivates are also effected by noise, hence it's advisable to smooth the image first before taking the derivative. Then we can use the convolution using the mask to detect the edges. Again, I am not going into the math part, we will focus only on the implementation details here.

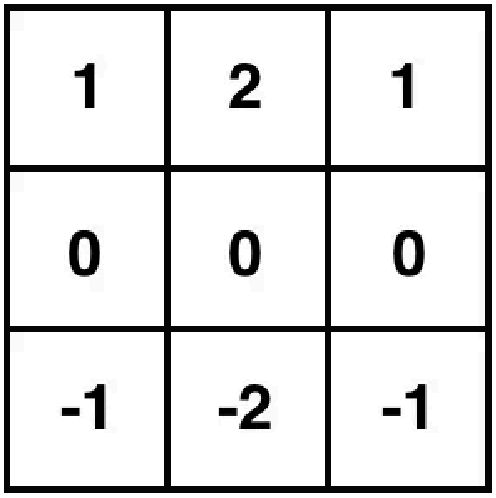
Sobel Operator:

Sobel Operator is a specific type of 2D derivative mask which is efficient in detecting the edges in an image. We will use following two masks:



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

Vertical



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

Horizontal

main:

Let's look at the implementation now.

```
if __name__ == '__main__':
    filter = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0,
1]])

    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required=True,
help="Path to the image")
    args = vars(ap.parse_args())

    image = cv2.imread(args["image"])
    image = gaussian_blur(image, 9, verbose=True)
    sobel_edge_detection(image, filter, verbose=True)
```

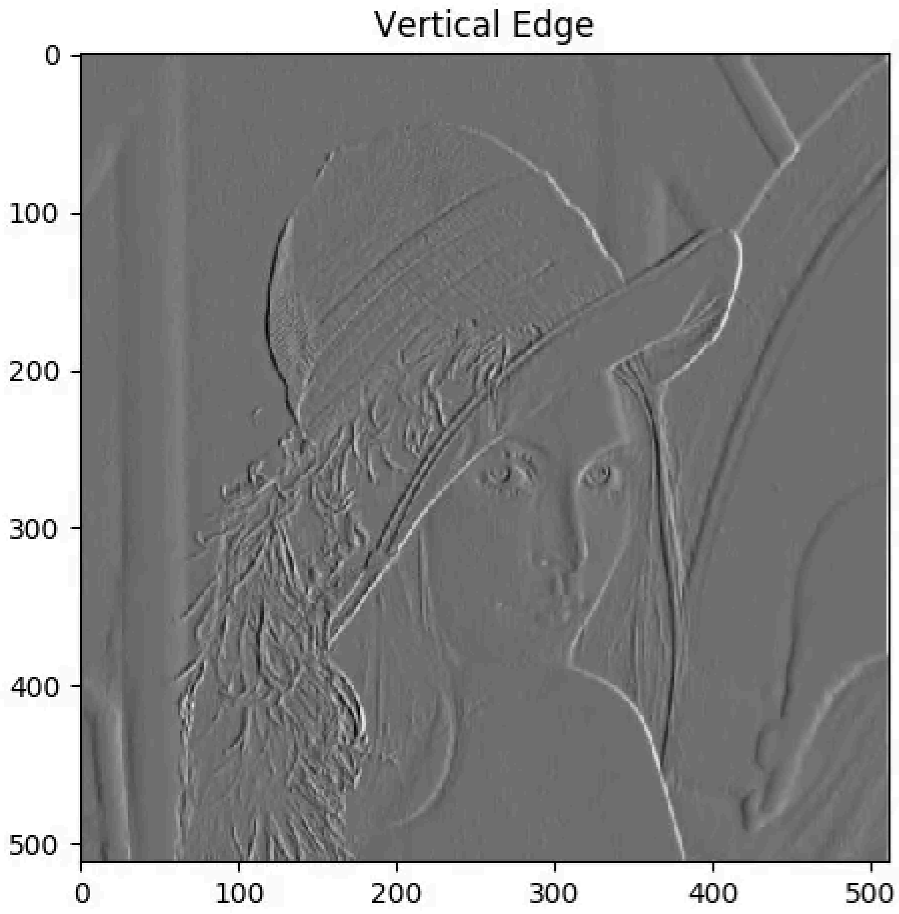
We will create the vertical mask using `numpy` array. The horizontal mask will be derived from vertical mask. We will pass the mask as the argument so that we can really utilize the `sobel_edge_detection()` function using any mask. Next apply smoothing using `gaussian_blur()` function. Please refer my tutorial on Gaussian Smoothing to find more details on this function.

Finally call the `sobel_edge_detection()` function by passing the image and the vertical filter.

sobel_edge_detection():

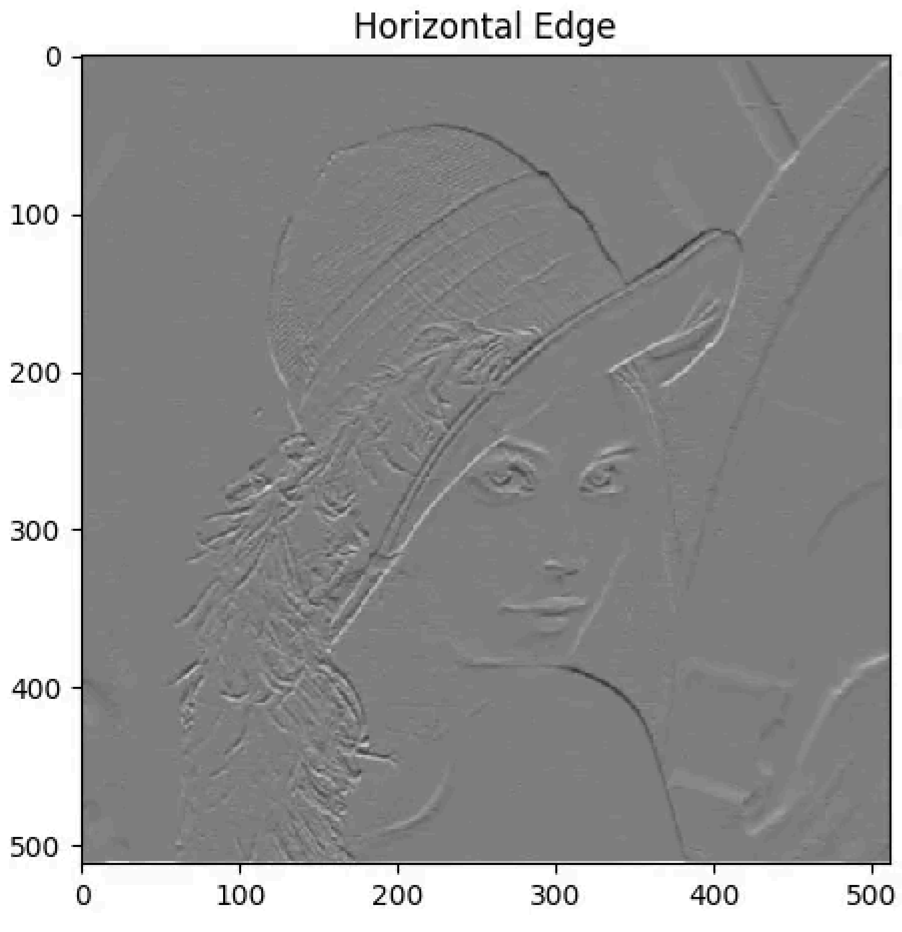
```
def sobel_edge_detection(image, filter, verbose=False):
    new_image_x = convolution(image, filter, verbose)

    if verbose:
        plt.imshow(new_image_x, cmap='gray')
        plt.title("Horizontal Edge")
        plt.show()
```

We will first call the `convolution()` function using the vertical mask. The output of the derivative looks like this:



```
new_image_y = convolution(image, np.flip(filter.T,
axis=0), verbose)

if verbose:
    plt.imshow(new_image_y, cmap='gray')
    plt.title("Vertical Edge")
    plt.show()
```

Then apply the convolution using the horizontal mask. We will simply take a transpose of the mask and flip it along horizontal axis. Here is the output:



In order to combine both the vertical and horizontal edges (derivatives) we can use the following equation:

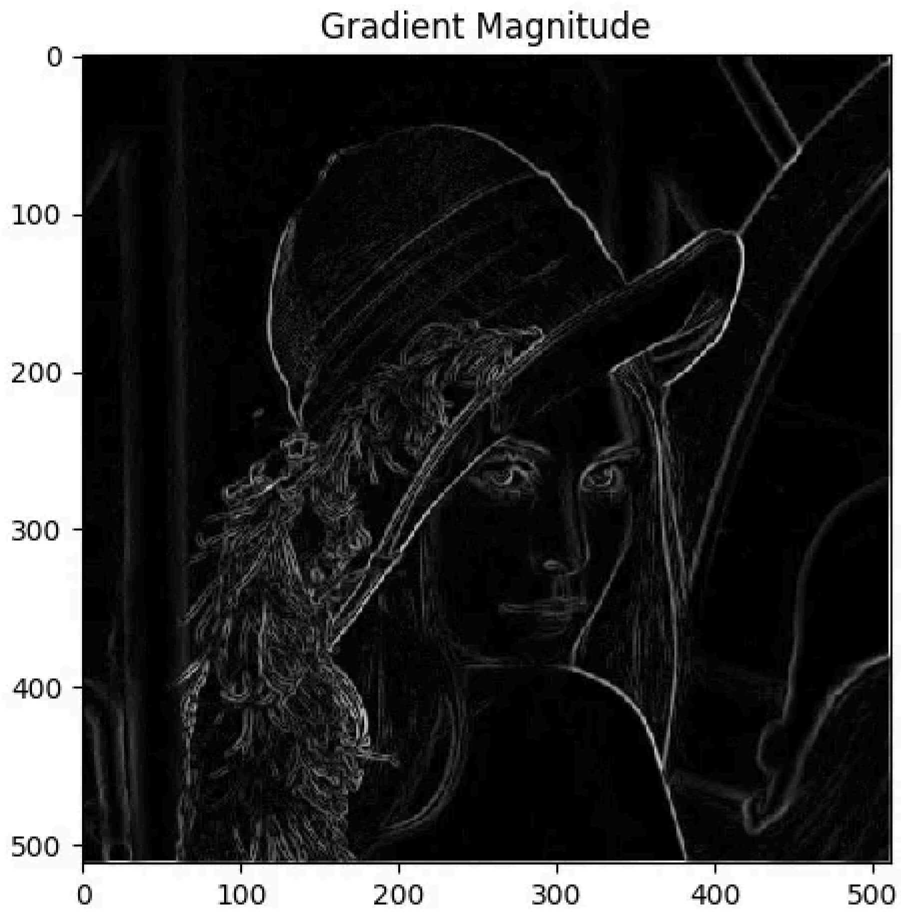
$$G = \sqrt{G_x^2 + G_y^2}$$

```
gradient_magnitude = np.sqrt(np.square(new_image_x) +
np.square(new_image_y))

gradient_magnitude *= 255.0 / gradient_magnitude.max()

if verbose:
    plt.imshow(gradient_magnitude, cmap='gray')
    plt.title("Gradient Magnitude")
    plt.show()
```

We will implement the same equation and then normalize the output to be between 0 and 255.

Output:

Here is the final output.



Limitation in Sobel Edge Detection Technique:

- Poor Localization, which means you will see many edges where we actually should have only edge.
- Can miss edges which are neither verticle or horizontal.

Canny Edge Detector:

Next we will implement Canny edge detector where we will overcome theses issues.

Full Code:

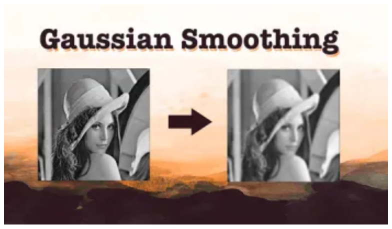
```
import numpy as np
import cv2
import argparse
import matplotlib.pyplot as plt
from Computer_Vision.Sobel_Edge_Detection.convolution
import convolution
from
Computer_Vision.Sobel_Edge_Detection.gaussian_smoothing
import gaussian_blur


def sobel_edge_detection(image, filter, verbose=False):
    new_image_x = convolution(image, filter, verbose)

    if verbose:
        plt.imshow(new_image_x, cmap='gray')
        plt.title("Horizontal Edge")
        plt.show()
    new_image_y = convolution(image, np.flip(filter.T,
axis=0), verbose)

    if verbose:
        plt.imshow(new_image_y, cmap='gray')
        plt.title("Vertical Edge")
        plt.show()

    gradient_magnitude = np.sqrt(np.square(new_image_x) +
np.square(new_image_y))

    gradient_magnitude *= 255.0 /
gradient_magnitude.max()

    if verbose:
        plt.imshow(gradient_magnitude, cmap='gray')
        plt.title("Gradient Magnitude")
        plt.show()

    return gradient_magnitude



if __name__ == '__main__':
    filter = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0,
1]])

    ap = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required=True,
help="Path to the image")
    args = vars(ap.parse_args())

    image = cv2.imread(args["image"])
    image = gaussian_blur(image, 9, verbose=True)
    sobel_edge_detection(image, filter, verbose=True)
```

Project in Github:

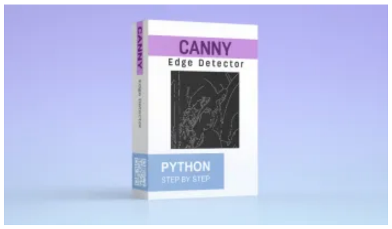
Please find the full project here:

GitHub

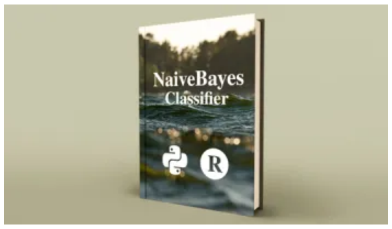
Related



Applying Gaussian Smoothing to an Image using Python from scratch
In "Computer Vision"



Implement Canny edge detector using Python from scratch
In "Computer Vision"



Introduction to Naive Bayes Classifier using R and Python
In "Data Science"

Filed Under: Computer Vision, Data Science | Tagged With: Computer Vision, Convolution, Edge Detection, Gaussian Smoothing, Image Derivative, Python, Smoothing, Sobel

Subscribe to stay in loop

* indicates required

Email Address *

Subscribe

Comments

Kai Chong says
May 3, 2020 at 9:19 am

(Edit)

Hi there! I went to look at the full project on your GitHub and was wondering the function of the padded image for the convolution function. I am new to computer vision, and thus am trying to fully understand your code and code out such functions!

code:

pad_height = int((kernel_row – 1) / 2)
pad_width = int((kernel_col – 1) / 2)

padded_image = np.zeros((image_row + (2 * pad_height), image_col + (2 * pad_width)))

padded_image[pad_height:padded_image.shape[0] – pad_height, pad_width:padded_image.shape[1] – pad_width] = image

Thanks!

Regards,

Kai

Reply

watsisname says

September 3, 2021 at 4:48 am

(Edit)

Kai chong (or anyone who may care).. Convolution involves multiplying two matrices and storing the output in the "center" pixel. So it results in an image smaller than one that was input. Simple fix is to enlarge the input image by padding it (It almost doesn't matter what it is padded with, as long as the result of convolution ends up with the right size).

Reply

watsisname says

September 3, 2021 at 4:52 am

(Edit)

BTW, if you don't want to use cv2.convertScaleAbs, the following code works. This is not what cv2 code description says (it says scale, abs, convert to int8, but abs, truncate and convert to int8 seems to be exactly equivalent).

```
result = abs(result)
result[result > 255] = 255
result = result.astype(np.uint8)
```

Reply

james says
September 13, 2021 at 3:00 pm

(Edit)

how is this from ‘scratch’? you are clearly using libraries…

Reply

Leave a Reply

Logged in as Abhisek Jana. Edit your profile. Log out? Required fields are marked *

Comment *

Post Comment