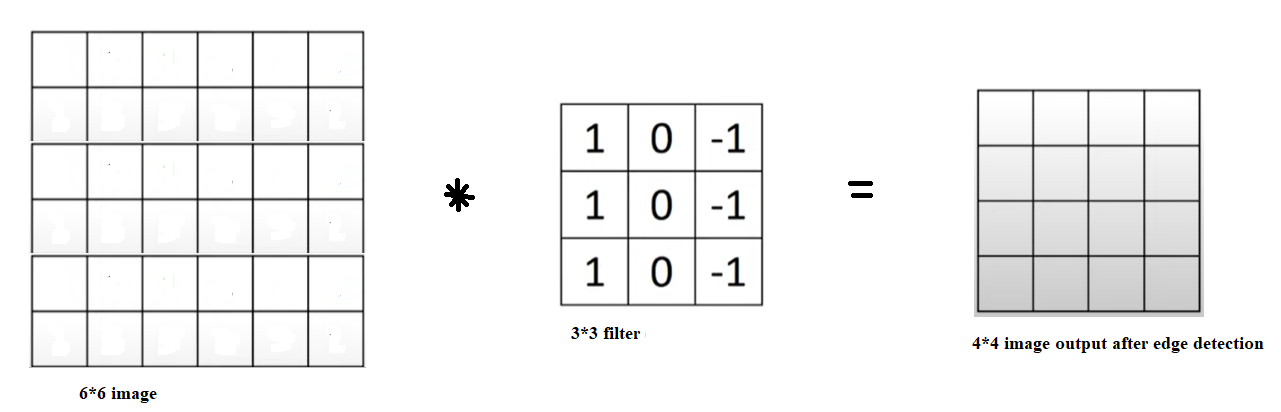
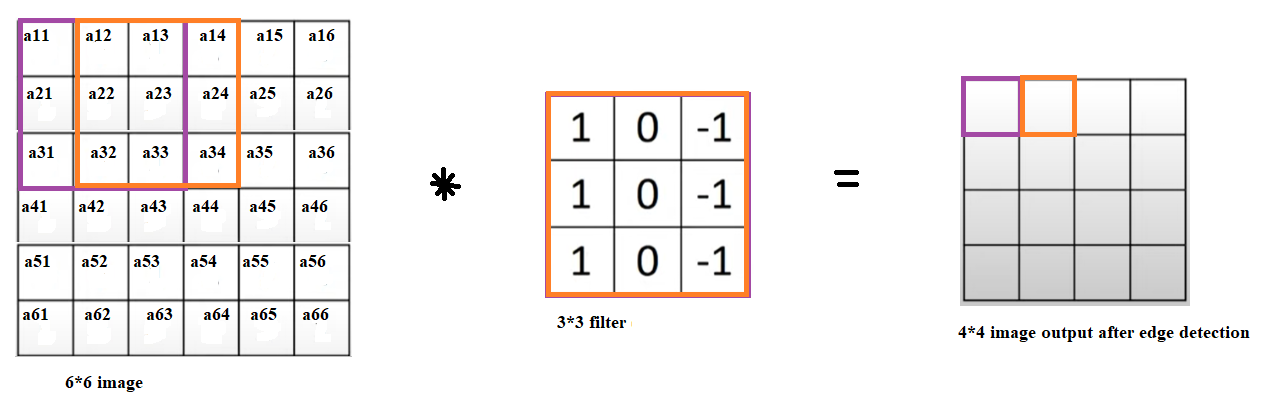
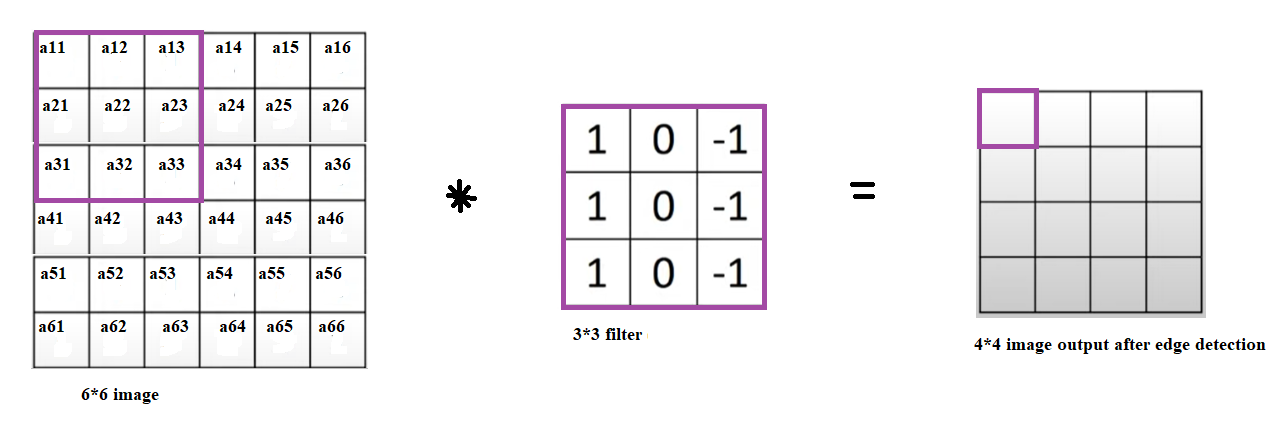
**What is Edge Detection?**

Edge detection is a technique of image processing used to identify points in a digital image with discontinuities, simply to say, sharp changes in the image brightness. These points where the image brightness varies sharply are called the edges (or boundaries) of the image.

It is one of the basic steps in image processing, pattern recognition in images and computer vision. When we process very high-resolution digital images, convolution techniques come to our rescue. Let us understand the convolution operation (represented in the below image using \*) using an example-



For this example, we are using 3\*3 Prewitt filter as shown in the above image. As shown below, when we apply the filter to perform detection on the given 6\*6 image (we have highlighted it in purple for our understanding) the output image will contain ((a11\*1)  + (a12\*0) + (a13\*(-1))+(a21\*1)+(a22\*0)+(a23\*(-1))+(a31\*1)+(a32\*0)+(a33\*(-1))) in the purple square.  We repeat the convolutions horizontally and then vertically to obtain the output image.



We would continue the above procedure to get the processed image after edge-detection. But, in the real world, we deal with very high-resolution images for Artificial Intelligence applications. Hence we opt for an algorithm to perform the convolutions, and even use Deep Learning to decide on the best values of the filter.

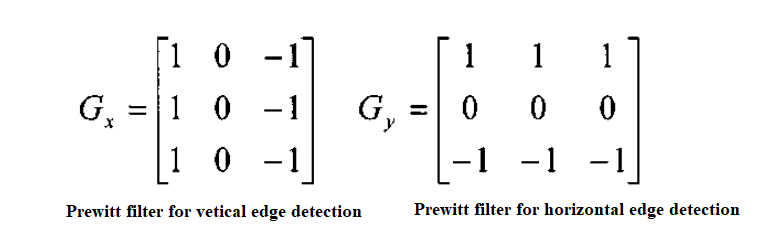
**Methods of Edge Detection**

There are various methods, and the following are some of the most commonly used methods-

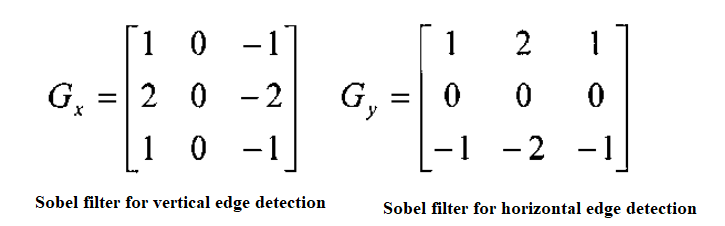
* Prewitt edge detection
* Sobel edge detection
* Laplacian edge detection
* Canny edge detection

**Prewitt Edge Detection**

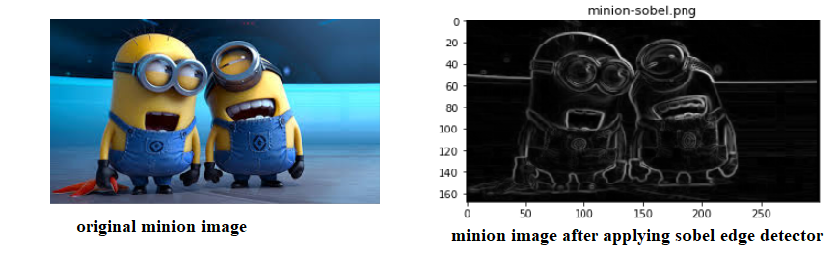
This method is a commonly used edge detector mostly to detect the horizontal and vertical edges in images. The following are the Prewitt edge detection filters-



**Sobel Edge Detection:**This uses a filter that gives more emphasis to the centre of the filter. It is one of the most commonly used edge detectors and helps reduce noise and provides differentiating, giving edge response simultaneously. The following are the filters used in this method-

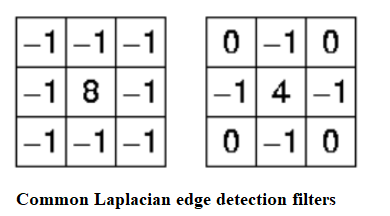


The following shows the before and after images of applying Sobel edge detection-

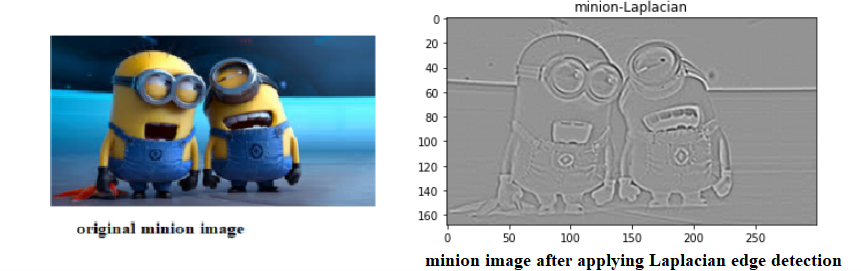


**Laplacian Edge Detection**

The Laplacian edge detectors vary from the previously discussed edge detectors. This method uses only one filter (also called a kernel). In a single pass, Laplacian detection performs second-order derivatives and hence are sensitive to noise. To avoid this sensitivity to noise, before applying this method, Gaussian smoothing is performed on the image.



The above are some of the commonly used Laplacian edge detector filters that are small in size. The following shows the original minion image and the final image after applying Gaussian smoothing (GaussianBlur() method of cv2) followed by Laplacian detection-

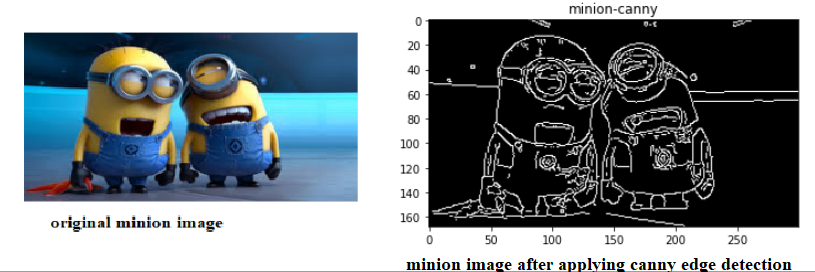


**Canny Edge Detection**

This is the most commonly used highly effective and complex compared to many other methods. It is a multi-stage algorithm used to detect/identify a wide range of edges.

* Convert the image to grayscale
* Reduce noise – as the edge detection that using derivatives is sensitive to noise, we reduce it.
* Calculate the gradient – helps identify the edge intensity and direction.
* Non-maximum suppression – to thin the edges of the image.
* Double threshold –  to identify the strong, weak and irrelevant pixels in the images.
* Hysteresis edge tracking – helps convert the weak pixels into strong ones only if they have a strong pixel around them.

The following are the original minion image and the image after applying this method.



**Drawbacks of applying edge computation**

* **Size of output will be shrunk.**

If you notice in the above example with an input of 6\*6 image after applying 3\*3 filter, the output image is only 4\*4. Usually, the formula is if the size of the input image is n\*n and the filter size is r\*r, the output image size will be (n-r+1)\*(n-r+1).

* **Loss of a lot of valuable information, especially from the edges of the input image.**

As the output image size is much reduced than the original image used as input (as discussed above), the information towards the edges of the input image is lost as we don’t iterate multiple times using the filter on the input images’ outer edges (unlike the middle of the input image).

**Techniques to overcome the drawbacks of edge computation**

To prevent the loss of such valuable information by image shrinkage, we usually use “padding” the input image before applying detection to avoid losing the valuable information in the input images.

What are edges

We can also say that sudden changes of discontinuities in an image are called as edges. Significant transitions in an image are called as edges.

Types of edges

Generally edges are of three types:

* Horizontal edges
* Vertical Edges
* Diagonal Edges

Why detect edges

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Here are some of the masks for edge detection that we will discuss in the upcoming tutorials.

* Prewitt Operator
* Sobel Operator
* Robinson Compass Masks
* Krisch Compass Masks
* Laplacian Operator.
* Above mentioned all the filters are Linear filters or smoothing filters.

### **Prewitt Operator**

* Prewitt operator is used for detecting edges horizontally and vertically.

### **Sobel Operator**

* The sobel operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection. It also calculates edges in both horizontal and vertical direction.

### **Robinson Compass Masks**

* This operator is also known as direction mask. In this operator we take one mask and rotate it in all the 8 compass major directions to calculate edges of each direction.

### **Kirsch Compass Masks**

* Kirsch Compass Mask is also a derivative mask which is used for finding edges. Kirsch mask is also used for calculating edges in all the directions.

### **Laplacian Operator**

* Laplacian Operator is also a derivative operator which is used to find edges in an image. Laplacian is a

second order derivative mask. It can be further divided into positive laplacian and negative laplacian.

All these masks find edges. Some find horizontally and vertically, some find in one direction only and some find in all the directions. The next concept that comes after this is sharpening which can be done once the edges are extracted from the image

## Sharpening

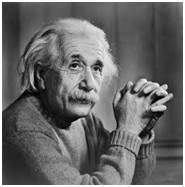
Sharpening is opposite to the blurring. In blurring, we reduce the edge content and in Sharpening, we increase the edge content. So in order to increase the edge content in an image, we have to find edges first.

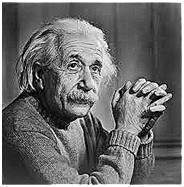
Edges can be find by one of the any method described above by using any operator. After finding edges, we will add those edges on an image and thus the image would have more edges, and it would look sharpen.

This is one way of sharpening an image.

The sharpen image is shown below.

### **Original Image**





## Prewitt operator

Prewitt operator is used for edge detection in an image. It detects two types of edges

* Horizontal edges
* Vertical Edges

Edges are calculated by using difference between corresponding pixel intensities of an image. All the masks that are used for edge detection are also known as derivative masks. Because as we have stated many times before in this series of tutorials that image is also a signal so changes in a signal can only be calculated using differentiation. So that’s why these operators are also called as derivative operators or derivative masks.

All the derivative masks should have the following properties:

* Opposite sign should be present in the mask.
* Sum of mask should be equal to zero.
* More weight means more edge detection.

Prewitt operator provides us two masks one for detecting edges in horizontal direction and another for detecting edges in an vertical direction.

## Vertical direction

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

Above mask will find the edges in vertical direction and it is because the zeros column in the vertical direction. When you will convolve this mask on an image, it will give you the vertical edges in an image.

### **How it works**

When we apply this mask on the image it prominent vertical edges. It simply works like as first order derivate and calculates the difference of pixel intensities in a edge region. As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. This increase the edge intensity and it become enhanced comparatively to the original image.

Horizontal Direction

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Above mask will find edges in horizontal direction and it is because that zeros column is in horizontal direction. When you will convolve this mask onto an image it would prominent horizontal edges in the image.

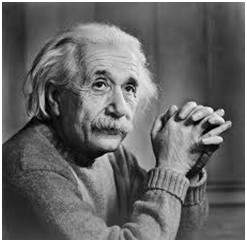
How it works

This mask will prominent the horizontal edges in an image. It also works on the principle of above mask and calculates difference among the pixel intensities of a particular edge. As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge. Thus increasing the sudden change of intensities and making the edge more visible. Both the above masks follow the principle of derivate mask. Both masks have opposite sign in them and both masks sum equals to zero. The third condition will not be applicable in this operator as both the above masks are standardize and we can’t change the value in them.

Now it’s time to see these masks in action:

Sample Image

Following is a sample picture on which we will apply above two masks one at time.



After applying Vertical Mask

After applying vertical mask on the above sample image, following image will be obtained. This image contains vertical edges. You can judge it more correctly by comparing with horizontal edges picture.



### **After applying Horizontal Mask**

After applying horizontal mask on the above sample image, following image will be obtained.



### **Comparison**

As you can see that in the first picture on which we apply vertical mask, all the vertical edges are more visible than the original image. Similarly in the second picture we have applied the horizontal mask and in result all the horizontal edges are visible. So in this way you can see that we can detect both horizontal and vertical edges from an image.

## sobel operator

The sobel operator is very similar to Prewitt operator. It is also a derivate mask and is used for edge detection. Like Prewitt operator sobel operator is also used to detect two kinds of edges in an image:

Vertical direction

Horizontal direction

Difference with Prewitt Operator

The major difference is that in sobel operator the coefficients of masks are not fixed and they can be adjusted according to our requirement unless they do not violate any property of derivative masks.

Following is the vertical Mask of Sobel Operator:

-1 0 1

-2 0 2

-1 0 1

This mask works exactly same as the Prewitt operator vertical mask. There is only one difference that is it has “2” and “-2” values in center of first and third column. When applied on an image this mask will highlight the vertical edges.

### **How it works**

When we apply this mask on the image it prominent vertical edges. It simply works like as first order derivate and calculates the difference of pixel intensities in a edge region.

As the center column is of zero so it does not include the original values of an image but rather it calculates the difference of right and left pixel values around that edge. Also the center values of both the first and third column is 2 and -2 respectively.

This give more weight age to the pixel values around the edge region. This increase the edge intensity and it become enhanced comparatively to the original image.

### **Following is the horizontal Mask of Sobel Operator**

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

Above mask will find edges in horizontal direction and it is because that zeros column is in horizontal direction. When you will convolve this mask onto an image it would prominent horizontal edges in the image. The only difference between it is that it have 2 and -2 as a center element of first and third row.

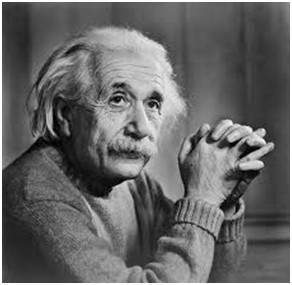
### **How it works**

This mask will prominent the horizontal edges in an image. It also works on the principle of above mask and calculates difference among the pixel intensities of a particular edge. As the center row of mask is consist of zeros so it does not include the original values of edge in the image but rather it calculate the difference of above and below pixel intensities of the particular edge. Thus increasing the sudden change of intensities and making the edge more visible.

Now it’s time to see these masks in action:

### **Sample Image**

Following is a sample picture on which we will apply above two masks one at time.



### **After applying Vertical Mask**

After applying vertical mask on the above sample image, following image will be obtained.



### **After applying Horizontal Mask**

After applying horizontal mask on the above sample image, following image will be obtained



### **Comparison**

As you can see that in the first picture on which we apply vertical mask, all the vertical edges are more visible than the original image. Similarly in the second picture we have applied the horizontal mask and in result all the horizontal edges are visible.

So in this way you can see that we can detect both horizontal and vertical edges from an image. Also if you compare the result of sobel operator with Prewitt operator, you will find that sobel operator finds more edges or make edges more visible as compared to Prewitt Operator.

This is because in sobel operator we have allotted more weight to the pixel intensities around the edges.

### **Applying more weight to mask**

Now we can also see that if we apply more weight to the mask, the more edges it will get for us. Also as mentioned in the start of the tutorial that there is no fixed coefficients in sobel operator, so here is another weighted operator

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -5 | 0 | 5 |
| -1 | 0 | 1 |

If you can compare the result of this mask with of the Prewitt vertical mask, it is clear that this mask will give out more edges as compared to Prewitt one just because we have allotted more weight in the mask.

## Laplacian Operator

Laplacian Operator is also a derivative operator which is used to find edges in an image. The major difference between Laplacian and other operators like Prewitt, Sobel, Robinson and Kirsch is that these all are first order derivative masks but Laplacian is a second order derivative mask. In this mask we have two further classifications one is Positive Laplacian Operator and other is Negative Laplacian Operator.

Another difference between Laplacian and other operators is that unlike other operators Laplacian didn’t take out edges in any particular direction but it take out edges in following classification.

* Inward Edges
* Outward Edges

Let’s see that how Laplacian operator works.

## Positive Laplacian Operator

In Positive Laplacian we have standard mask in which center element of the mask should be negative and corner elements of mask should be zero.

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

Positive Laplacian Operator is use to take out outward edges in an image.

## Negative Laplacian Operator

In negative Laplacian operator we also have a standard mask, in which center element should be positive. All the elements in the corner should be zero and rest of all the elements in the mask should be -1.

|  |  |  |
| --- | --- | --- |
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

Negative Laplacian operator is use to take out inward edges in an image

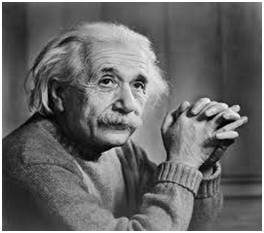
### **How it works**

Laplacian is a derivative operator; its uses highlight gray level discontinuities in an image and try to deemphasize regions with slowly varying gray levels. This operation in result produces such images which have grayish edge lines and other discontinuities on a dark background. This produces inward and outward edges in an image

The important thing is how to apply these filters onto image. Remember we can’t apply both the positive and negative Laplacian operator on the same image. we have to apply just one but the thing to remember is that if we apply positive Laplacian operator on the image then we subtract the resultant image from the original image to get the sharpened image. Similarly if we apply negative Laplacian operator then we have to add the resultant image onto original image to get the sharpened image.

Let’s apply these filters onto an image and see how it will get us inward and outward edges from an image. Suppose we have a following sample image.

### **Sample Image**



### **After applying Positive Laplacian Operator**

After applying positive Laplacian operator we will get the following image.



### **After applying Negative Laplacian Operator**

After applying negative Laplacian operator we will get the following image.

