Sobel Image Edge Detection

Abhishek Sharma
abhisharmaec0982@gmail.com

***Abstract*— This paper introduces sobel Image edge detection algorithm for the Learning API of Frankfurt University of applied science. Image edge detection is a process of locating the edge of an image which is important in finding the approximate absolute gradient magnitude at each point I of an input grayscale image. The problem of getting an appropriate absolute gradient magnitude for edges lies in the method used. The Sobel operator performs a 2-D spatial gradient measurement on images. Transferring a 2-Dpixel array into statistically uncorrelated data set enhances the removal of redundant data, as a result, reduction of the amount of data is required to represent a digital image. The Sobel edge detector uses a pair of 3 x 3 convolution masks, one estimating gradient in the x-direction and the other estimating gradient in y–direction. The Sobel detector is incredibly sensitive to noise in pictures, it effectively highlight them as edges. Hence, Sobel operator is recommended in massive data communication found in data transfer.**

Keywords: Image Processing, Edge Detection, Absolute gradient magnitude.

I. INTRODUCTION

This Project is a part of Software Engineering Module of Frankfurt University of Applied Science. The aim of this project is to implement Image Edge detection Algorithm using sobel edge detection technique for the Learning API(Machine Learning API).

Edge detection is a fundamental tool in image processing and computer vision, particularly in the area of feature detection and feature extraction, which aims at identifying points in a digital image at which the image brightness change sharply. Edge detection is also called high frequency spatial filtering which is used to highlight the boundaries of different object in an image and hides the homogeneous background. Sobel filter is used in image processing particularly within edge detection algorithms. The sobel operator is based on convolving the image with a small, separable and integer valued filter in horizontal and vertical direction[1]

The purpose of detecting sharp changes in image brightness is to capture important events. Applying an edge detector to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image. The image quality reflects significant information in the output edge and the size of the image is reduced. This in turn explains further that edge detection is one of the ways of solving the problem of high volume of space images occupy in the computer memory. The problems of storage, transmission over the Internet and bandwidth could be solved when edges are detected .Since edges often occur at image locations representing object boundaries; edge detection is extensively used in image segmentation. when images are divided into areas corresponding to different objects.

II. METHODOLOGY

The sobel filter is used for edge detection. It works by calculating the gradient of image intensity at each pixel within the image. It finds the direction of the largest increase from light to dark and the rate of change in that direction. The result shows how abruptly or smoothly the image changes at each pixel, and therefore how likely it is that that pixel represents an edge. It also shows how that edge is likely to be oriented. The result of applying the filter to a pixel in a region of constant intensity is a zero vector. The result of applying it to a pixel on an edge is a vector that points across the edge from darker to brighter values.

The sobel filter uses two 3 x 3 kernels. One for changes in the horizontal direction, and one for changes in the vertical direction. The two kernels are convolved with the original image to calculate the approximations of the derivatives. If we define Gx and Gy as two images that contain the horizontal and vertical derivative approximations respectively, the computations are:

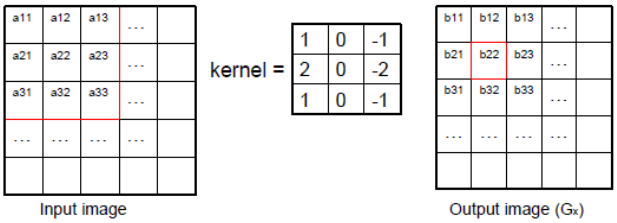
$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * A \quad \text{and} \quad G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * A$$

Where **A** is the original source image.

The x coordinate is defined as increasing in the right direction and the y coordinate is defined as increasing in the down-direction.

The x coordinate is defined as increasing in the right direction and the y coordinate is defined as increasing in the down-direction.

To compute Gx and Gy we move the appropriate kernel (window) over the input image, computing the value for one pixel and then shifting one pixel to the right. Once the end of the row is reached, we move down to the beginning of the next row.



$b_{22} = a_{13} - a_{11} + 2a_{23} - 2a_{21} + a_{33} - a_{31}$

The kernels contain positive and negative coefficients. This means the output image will contain positive and negative values.[2]

For display purposes we can map the gradient of zero onto a half-tone grey level.

– **This makes negative gradients appear darker, and positive gradients appear brighter.**

Use the absolute values of the gradient map (stretched between 0 and 255).

-This makes very negative and very positive gradients appear brighter.

The kernels are sensitive to horizontal and vertical transitions. The measure of an edge is its amplitude and angle. These are readily calculated from G_x and G_y.

At each pixel in the image, the gradient approximations given by Gx and Gy are combined to give the gradient magnitude, using:

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

The gradient's direction is calculated using:

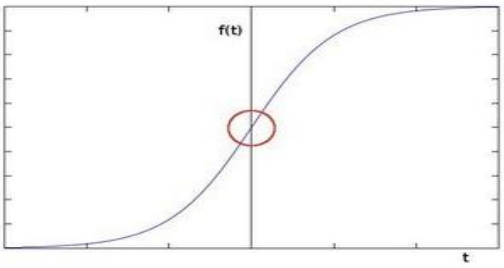
$$\Theta = \arctan\left(\frac{G_y}{G_x}\right)$$

A angle (gradient's direction) value of 0 would indicate a vertical edge that is darker on the left side.

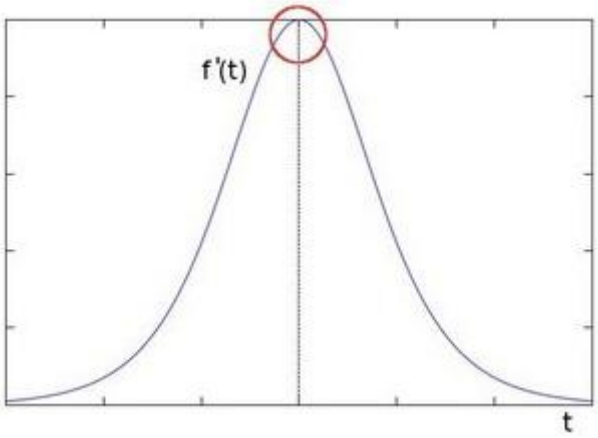
Pseudo-codes for Sobel edge detection method

Input: A Sample input Image
Output: Detected Edges (output Image)
Step 1: Accept the input image and convert in to 3-D double Array
Step 2: Apply mask Gx (Horizontal kernel), Gy (vertical kernel) to the input image.
Step 3: Apply Sobel edge detection algorithm, convert image into grayscale and the gradient i.e. convolution.
Step 4: Masks manipulation of Gx, Gy separately on the input image.
Step 5: Results combined to find the absolute magnitude of the gradient
Step 6: The absolute magnitude is the output edges

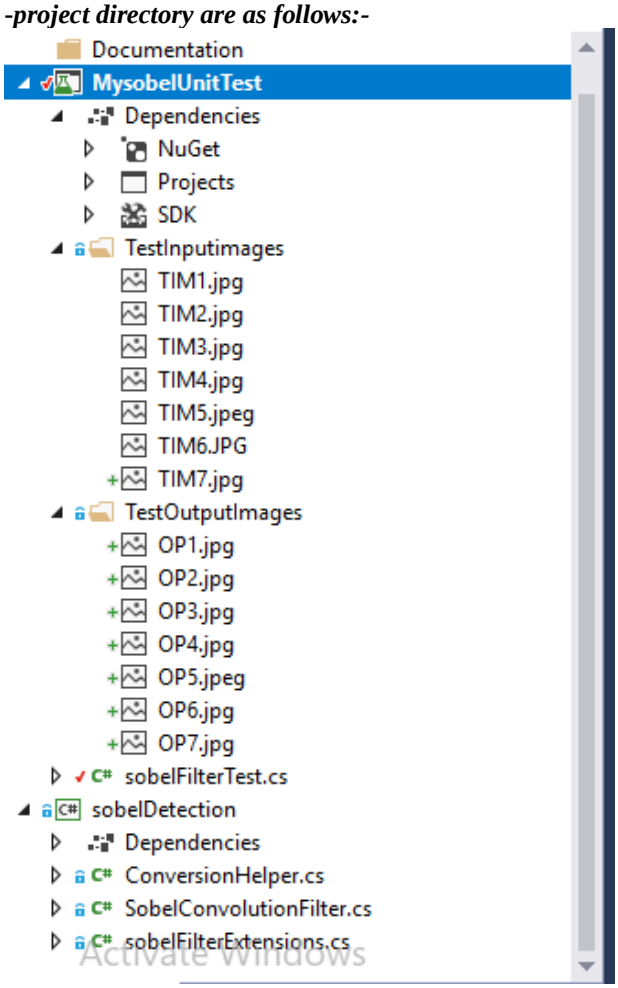
To be more graphical, let's assume we have a 1D-image. An edge is shown by the "jump" in intensity in the plot below:[3]



The edge "jump" can be seen more easily if we take the first derivative (actually, here appears as a maximum)



Lets Look at the program of sobel edge detection:-
-project directory are as follows:-



1. Creating Sobel kernel:-

Using convolution, mathematical process, we apply our kernels to each pixel in our image. Each kernel is a matrix the size of 3×3,that holds predetermined values. When a kernel is over a pixel it calculates values using neighboring pixels that are also under this kernel. Below are displayed both kernels or matrices.

```
public static double[,] XSobel
{
    get
    {
        return new double[,]
        {
            { -1, 0, 1 },
            { -2, 0, 2 },
            { -1, 0, 1 }
        };
    }
}

public static double[,] YSobel
{
    get
    {
        return new double[,]
        {
            {  1,  2,  1 },
            {  0,  0,  0 },
            { -1, -2, -1 }
        };
    }
}
```

2. Code for Convolution filter

First part of the function is to get image data and saving it to an array using Run method of IPipeline module interface.

```
public double[,,] Run(double[,,] data, IContext ctx)
{
    return convolutionFilter(data, XSobel, YSobel);

}
```

3. Convert image into grayscale

```
if (grayscale == true)
{
    float rgb = 0;
    for (int i = 0; i < pixelarray.Length; i += 4)
    {
        rgb = pixelarray[i] * .21f;
        rgb += pixelarray[i + 1] * .71f;
        rgb += pixelarray[i + 2] * .071f;
        pixelarray[i] = (byte)rgb;
        pixelarray[i + 1] = pixelarray[i];
        pixelarray[i + 2] = pixelarray[i];
        pixelarray[i + 3] = 255;
    }
}
```

4. Setting variables used in convolution process

```
double xr = 0.0;
double xg = 0.0;
double xb = 0.0;
double yr = 0.0;
double yg = 0.0;
double yb = 0.0;
double rt = 0.0;
double gt = 0.0;
double bt = 0.0;
```

5. Applying Kernel convolution to image pixel.

6. Processed image output code

```
//Create new bitmap which will hold the processed data
Bitmap resultImage = new Bitmap(width, height);

// Lock bits into system memory
BitmapData resultData = resultImage.LockBits(new Rectangle(0, 0, wi

// Copy from byte array that holds processed data to bitmap
Marshal.Copy(resultArray, 0, resultData.Scan0, resultArray.Length);

// Unlock bits from system memory
resultImage.UnlockBits(resultData);

// Return processed image

double[,,] resultImageDouble = new double[width, height, 3];
'/convert the data in to double using conversion helper class
ConversionHelper helper = new ConversionHelper();
resultImageDouble = helper.convertBitmapToDouble(resultImage);

return resultImageDouble:
```

Now, This code is implemented as per the architecture of learning api and Unit test is also written in the same manner as shown below :-

Unit test:-

```
LearningApi api = new LearningApi();

api.UseActionModule<double[,,], double[,,]>((input, ctx) =>
{

    string baseDirectory = AppDomain.CurrentDomain.BaseDirectory;
    string path = Path.Combine(baseDirectory, "TestInputimages\\TIM1.jpg");
    Bitmap bitmap = new Bitmap(path);
    double[,,] data = helper.convertBitmapToDouble(bitmap);// convert bitmap
    return data;

});
api.AddModule(new SobelConvolutionFilter());
double[,,] result = api.Run() as double[,,];

Bitmap bitresult=helper.convertDoubleToBitmap(result);// convert double to bi

string baseDirectory2 = AppDomain.CurrentDomain.BaseDirectory;
string outpath = baseDirectory2 + "\\TestOutputImages\\";
if (!Directory.Exists(outpath))
{
    Directory.CreateDirectory(outpath);
}
```

All the conversion from bitmap to double and vice versa is done through conversion helper class.

Input test images are loading from 'Testinputimages' folder created in the directory and output image is saved in 'TestoutputImages 'folder.

III. Result and Discussion

The Sobel operator performs a 2-D spatial gradient measurement on an image. Typically, it is used to find the approximate absolute gradient magnitude at each point I of an input grayscale image. The Sobel edge detector uses a pair of 3 x 3 convolution masks, one estimating gradient in the x-direction and the other estimating gradient in y- direction. A convolution is usually much smaller than the actual image. As a result, the mask is slide over the image manipulating a square of pixels at a time. The mask is slides over an area where the input image changes with that pixel's value and then shifts one pixel to the right and continues to the right until it reaches the end of the row which automatically starts again at the beginning of the next row. It is important to note that pixels in the first row and last row, as well as the first and last column cannot be manipulated by a 3 x 3 mask. This is because when placing the center of the mask over a pixel in the first row for example, the mask will be outside the image boundaries.[5] The Gx mask highlights the edges in the horizontal direction while the Gy mask highlights the edges in vertical direction. After taking the magnitude of both, the resulting output detects edges in both directions. This is done by:

(1) Applying noise smoothing to the original image

(2) Filtering the original image following two kernels gives the result in Table 1.

Table 1: Filtering Results of the two Kernel

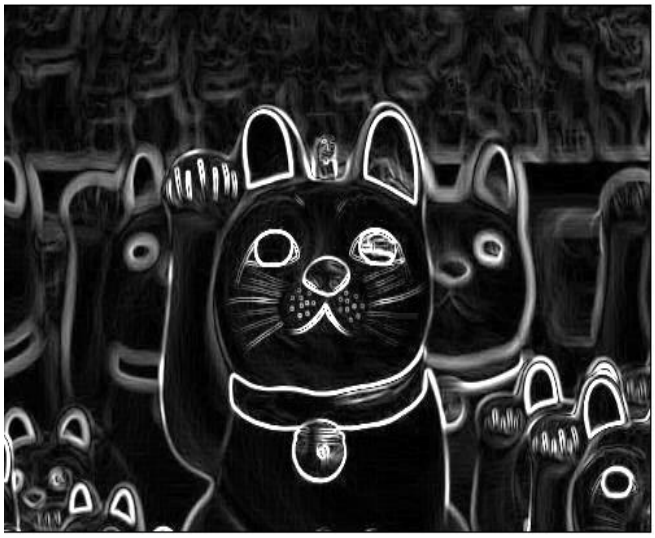
Kernel1= *Gx*			Kernel 2 = *Gy*		
-1	0	1	-1	-2	-1
-2	0	2	0	0	0
-1	0	1	1	2	1

Below are the Test input image and output images saved in bin folder of project directory.

Input image 1

Output Image 1

Input Image2



Output Image 2

IV. PRACTICAL IMPLICATION AND IMPORTANCE OF EDGE DETECTION

The following advantages of Sobel edge detector justify its superiority over other edge detection:-[6]

1. **Edge Orientation:** The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges.

2. **Noise Environment**: Edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges. Operators used on noisy images are typically larger in scope, so they can average enough data to discount localized noisy pixels. This results in less accurate localization of the detected edges.

3. **Edge Structure:** Not all edges involve a step change in intensity. Effects such as refraction or poor focus can result in objects with boundaries defined by a gradual change in intensity. The operator is chosen to be responsive to such a gradual change in those cases. Newer wavelet-based techniques actually characterize the nature of the transition for each edge in order to distinguish, for example, edges associated with hair from edges associated with a face.

Edges play quite an important role in many applications of image processing, in particular for machine vision systems that analyze scenes of man-made objects under controlled illumination conditions. Detecting edges of an image represents significantly reduction the amount of data and filters out useless information, while preserving the important structural properties in an image. Hence, edge detection is a form of knowledge management[5].

V. CONCLUSION

The Sobel operator performs a 2-D spatial gradient measurement on an image. Typically it is used to find the approximate absolute gradient magnitude at each point I of an input grayscale image. The Sobel edge detector uses a pair of 3 x 3 convolution masks, one estimating gradient in the x direction and the other estimating gradient in y–direction. It is easy to implement than the other operators. Transferring a 2-D pixel array into statistically uncorrelated data set enhances the removal of redundant data, as a result, reduction of the amount of data required to represent a digital image. Considering data communication especially the internet, massive data transfer causes serious problems for interactive network users. Edge detection helps in optimizing network bandwidth and it is needed to keep track of data flowing in and out of the network. It helps to extract useful features for pattern recognition. Although the Sobel operator is slower to compute, it's larger convolution kernel smooth's the input image to a greater extent and so makes the operator less sensitive to noise. The larger the width of the mask, the lower its sensitivity to noise and the operator also produces considerably higher output values for similar edges. Sobel operator effectively highlights noise found in real world pictures as edges though, the detected edges could be thick. The Canny edge detector and similar algorithm [3]solved these problems by first blurring the image slightly then applying an algorithm that effectively thins the edges to one-pixel. This may constitute a much slower process, hence, Sobel operator is Rehighly recommended in massive data communication found in image data transfer. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

VI. REFERENCES

[1] Agaian, S. S., Baran, T. A., & Panetta, K. A. (2003). Transform-based image compression by noise reduction and spatial modificat ion using Boolean minimization. IEEE Workshop on Statistical Signal Processing28 Sept.-1 Oct. pp. 226 – 229.

[2] Baker, S., & Nayar, S. K. (1996). Pattern rejection. Proceedings of IEEE Conference Computer Vision and Pattern Recognition, 544-549

[3] Canny, J. F. (1986). A computational approach to edge detection. IEEE Trans Pattern Analysis and Machine Intelligence, 8(6), 679-698

[4] Chang-Huang, C. (2002). *Edge detection based on class ratio*. 152, sec.3, Peishen Rd., Shenkeng, Taipei, 22202, Taiwan, R.O.C.

[5] Folorunso O., Vincent, O. R., & Dansu, B. M. (2007). Image edge detection: A knowledge management technique for visual scene analysis. *Information Management and Computer Security, 15*(1), 23-32.

[6] Gonzalez, R., & Woods, R. (2002). Digital image processing (2nd ed.). Prentice-Hall Inc. 567-612

[7] https://epochabuse.com/csharp-sobel/