Information Technology Course
Module Software Engineering
by
Pro:**Damir Dobric / Andreas Pech**

Image *Edge Detection:Laplacian of Gaussian*

Name:BODIGE MADHU

Metriculation:1274288

bodigemadhushashi@gmail.com

***Abstract*—Laplacian of Gaussian filters is convolution filter which is detect the edges of a real images. The gaussian blur will be applied for the image first then Laplacian filters and then searches for the zero-crossings when the values change from the positive to negative. The main purpose of the filter is to brighten the edges of the resulting image. Laplacian filters are too sensitive consequently it can also add the noise to the output image, by smoothening the image noise will be removed. LoG operator applied to the second derivative of the single grayscale image and produces another grayscale image resultantly. Where there is no rapid change in the pixels of the image, the result will be zero. Wherever conversion occurs, the it will show a positive response on the darker side of the image and shows a negative response on the l front side of the image.**

Log attempts to eliminate image noise by applying image smoothing by using Gaussian blur. To get effective performance, we can calculate a kernel matrix representing a Gaussian blur and Laplacian matrix.

The moto of the project is to initiate a model that detects the edges of an image and converts it into the grayscale image and applies Laplacian of Gaussian (Log) filter and build the Microsoft. Net core platform for Learning API.

Keywords—Laplacian filter, Gaussian filter, Laplacian of Gaussian, edges, image, Gaussian blur, kernel matrix

I.INTRODUCTION

Edge is the result of changes in light, color, shade, body texture and above changes will be used to Resolve the depth, orientation and surface of a image.Digital analysis of the image helps in filter unnecessary information to select the edge points. The detection of refined changes are added up by noise and it depends upon the pixel of change that defines an threshold edge. Detection of edges is time-consuming and very complex. especially when an image is mixed by noise. Edge detection is a intial and effective tool in the main areas of image processing and computer vision applications such as feature detection and feature extraction.

Edge detection is used to detect objects, locate boundaries, and extract features. Edge detection can identifying faster, local changes in the depth values of the pixels in an image. It is a series of actions used to identify the points in an image where clear and defined changes occur in the intensity. This kind of action is mandatory to extract the image related information e.g. image sharpening, enhancement of the image.

Edge detection algorithm:

- load the 3D color image.
- Refining: Refining is used to remove the noise without a damage to the image.
- 3 Threshold: Edge threshold is used to not allow the noisy edge pixels.
- 5. Localization: Some applications to approximates the position of an edge and spacing between pixels, sub pixels resolution also required.
- 6. After edge exposures Take the image.

II . LAPLACIAN OF GAUSSIAN(LOG)

A.WORKFLOW:

Log is useful to detect the edges and also useful to find the blobs of the image.it attempts to reduce the noise by implementing the image smoothening by gaussian blur.in order to optimize the performance for the best possible results we can use the Laplacian and gaussian matrix.

Fig 1: basic block diagram of Log

B. LoG Algorithm:

- 1.convoltion of image with 3-dimentional gaussian function
- 2.comptuting the Laplacian of the convoluted image
- 3.findout the edge pixels of as those which there is a zero-crossings available

C.Laplacian Filter:

Laplacian filters are derivative filter which we use to remove the noise and detect the edges

- First smooth (Gaussian filter),
- Then, find zero-crossings to find edges
- Since Laplacian filter is sensitive to noise, we introduce gaussian to remove the noise.

Laplacian Filter is a filter used for edge-detection, in which uses Laplace operator. The Laplace operator is as follows

A. *First order derivatives:*

$$\nabla^2 \times f = \frac{\partial^2 f}{\partial^2 y} \times \frac{\partial^2 f}{\partial^2 x}$$

The above equation defines the 1st order derivative of the laplacian filter which we use to detect the edges of the image where the partial 1st order derivative in the x direction is defined as follows:

$$(\partial \hat{} 2\ f)/(\partial \hat{} 2\ x) = (f(x+1,y), f(x-1,y) - 2f(x,y)$$

And in the y direction is

$$(\partial \hat{} 2\ f)/(\partial \hat{} 2\ x) = (f(x,y+1), f(x,y-1) - 2f(x,y)$$

Then to find the zero crossings of the image we have to apply the 2nd order derivatives in which it helps to detect the edges.as we know now we have to convolute the image pixels with the Laplacian filter where the intensity levels are(0,255) accordingly.

D.Gaussian Filter

The Gaussian smoothing operator are sensitive to noise and it is used to blur the image and remove the noise.It is kind of similar to the Mean filter, but it uses a ideal kernel filter that represents the structure of a Gaussian (`bell-shaped').

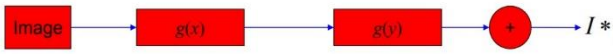

Fig 2: Block diagram of gaussian filter

This kernel filter properties has detailed below.

$$G(x,y) = \frac{1}{2\Pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

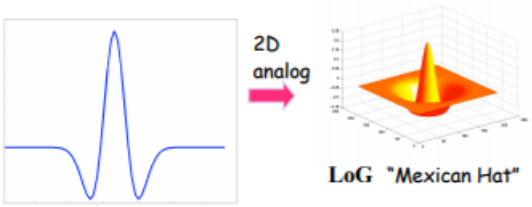

Fig 3: Second derivative of gaussian

- This function has a lowest at its origin, but it is regular to invert the filter.
- The value of σ is determines the filter width and controls the amount of smoothing which is produced bz the gaussian component.

Gaussian smoothing uses this 2-D distribution as a `point-spread' function, and this is complimented by convolution. we need to produce a discrete approximation to the Gaussian function before we can perform the convolution Since the image is stored as a collection of discrete pixels. theorotically Gaussian distribution is non-zero everywhere, which requires an large number of convolution kernel, but in practicaly it is zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point.The outputs a `weighted average' of each pixel's gaussian neighborhood, with the calculated average weighted more towards the mean of the middle pixels.mean filter's uniform weightage averag Is in contrast to this., a Gaussian provides sensitive smoothing and detects edges better than a similarly sized mean filter. The kernel is used for convolution is

1	2	1
2	4	2
1	2	1

Table 1: gaussian 3x3 kernel matrix

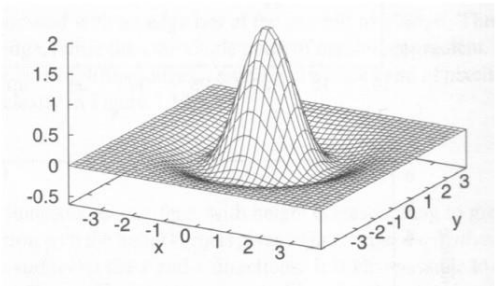

Fig 4: classic Mexican hat shape distribution

The below picture is an example of gaussian filter where you can see the blured image as an output image after applying the gaussian filter

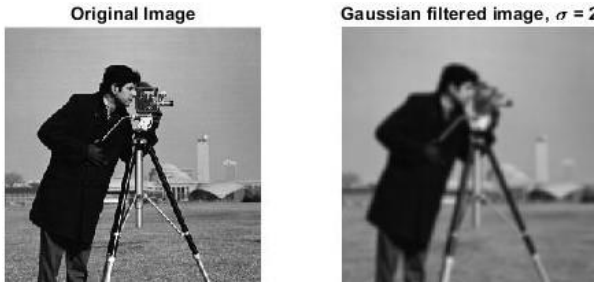

Figure 5: Gaussian filter Output

In the above image we see blurring produced by size of the image pixel by gaussian blurring theory.

E.Laplacian of gaussian *(LoG) Filter*

Laplacian of Gaussian Filter is a combination of Gaussian and Laplacian Filter where the image smoothening process done by Laplacian of gaussian filter. As we know this filter is to detect the edges and also it smoothens the image by removing noise. Since it is similar to the Laplacian filter it also performs the second derivative for image pixel.

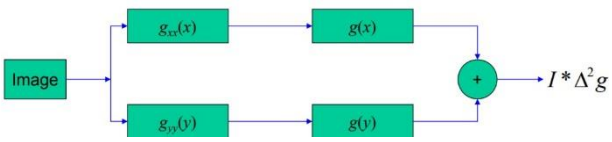

Figure 6: Block diagram of LoG

Second derivative is usually doing to find the zero crossings.

$(\partial \hat{} 2\, f)/(\partial \hat{} 2\, x) = (f(x+1,y), f(x-1,y) - 2f(x,y)$

These are 1st order partial derivatives in x-direction and y-direction respectively. Now, we after doing second derivative we get the equation as follows:

$$\nabla \hat{} 2 \times f = (f(x+1,y), f(x-1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y)$$

The 2-D Log function centered on zero and with Gaussian standard deviation σ has the form:

The Log operator calculates the second spatial derivative of an image. This means that in areas where the image has a constant intensity (*i.e.* where the intensity gradient is zero), the LoG response will be zero. In the vicinity of a change in intensity, however, the LoG response will be positive on the darker side, and negative on the lighter side. This means that at a reasonably sharp edge between two regions of uniform but different intensities, the LoG response will be:

- At long distance from the edge should be zero
- positive just to one dark side of edge
- negative just to another lighter side of edge
- zero at some point in between, on the edge itself

The response of Laplacian of Gaussian LoG for Step Edge is as follows:

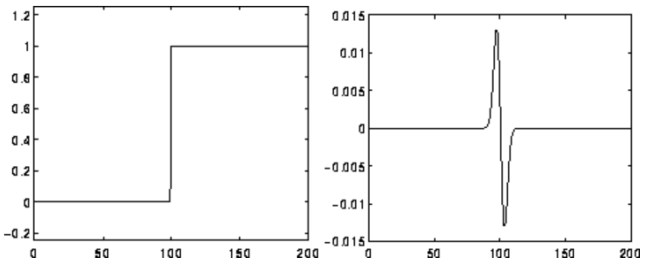

Figure 7: LoG Filter to Step Edge

F. Applications of LoG filter

- The gaussian filter is set to remove the unwanted noise or detail
- Zero crossing edge detection is the parameter and highly governed by the LOG filter to remove the noise.
- When we set the higher value of this the more smoothed values will exist hence minimum number of zero crossings will be produced.
- The smoothening value depends upon varying value of the standard deviation

G. Zero-crossings

To form an edge map from the Laplacian of gaussian output we have to mark and locate the zero-crossings. If trying to detect zeros in the LoG will fail, follow below procedure.

An alternative approx. to finding edges as peaks in first derive is to find zero-crossings in second deriv. In 1D, convolve with [1 -2 1] and look for pixels where response is (nearly) zero? Problem: when first derivative is zero, so is second. I.e. the filter [1 -2 1] also produces zero when convolved with regions of constant intensity. So, in 1D, convolve with [1 -2 1] and look for pixels where response is nearly zero AND magnitude

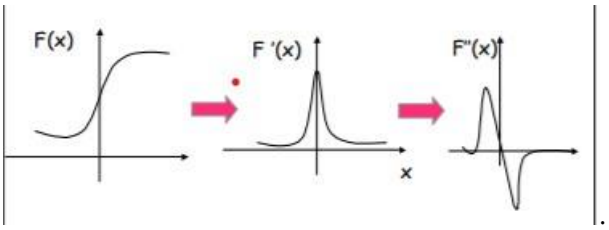

Figure 8: Zero-Crossing for second derivate filters

H.Laplacian of Gaussian Code steps:

- First load the 3-dimensional input image then convert that into a bitmap image
- Now we have to apply Laplacian of gaussian filter to the input image and convert it into a gray scale image
- Now Apply the convolution filter to load image which is gray scaled with the Laplacian of Gaussian kernel 3x3 matrix.
- Now, again change the bitmap image to the 3-dimentional image.
- Then test the input image and load the output image
- Now the final step detects edges of the image.

I. Laplacian of Gaussian project description

The Laplacian of Gaussian is the main class of the program and it classified into two sub classes i.e., Laplacian of gaussian class and another is LOG Extension class. The Laplacian of gaussian class defines the convolution filter and where the conversion images process takes places as LoG Extension is for compatibility for LearningAPI Foundation. This program also contains the LOGTest with the added references and in which it is useful to run the program and also it starts the program.LOG test has combined with the reference of Laplacian of gaussian project to fetch the data directly and provide the results classified two folders to take the Input images and to store the Output images. This program also contains the LOG Test with the added references and in which it is useful to run the program and also it starts the program Input images consist the load images and we can see the LOG filter applied results in output images.

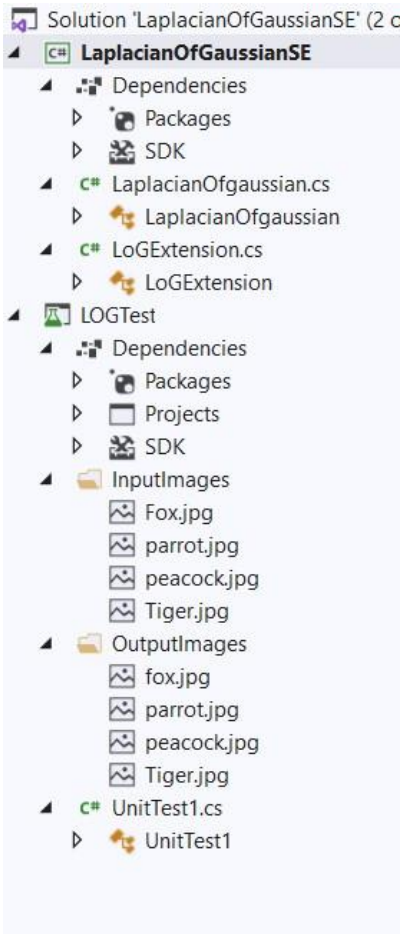

Fig 9:Program of Image Edge Detection of Laplacian of Gaussian

III .RESULTS

J.Load Input Image

Provide a the 3-dimensional array Input image which need to be converted in the code to Bitmap image and convert it to the gray scale image on the next step, later with the pixel values so that we can apply the convolution filter with the image pixels and with the Laplacian of Gaussian kernel 3x3 to get the final output. This final output later converted back to the 3-dimensional array.

Figure 10: Input image

K.Conversion of 3D array to Bitmap image

```
public static Bitmap changeFrom3dArrayToBitmap(double[,,]
filterimageArray)
      {
 Bitmap bitmapreslut = new
Bitmap(filterimageArray.GetLength(0),
filterimageArray.GetLength(1));

         for (int i = 0; i < filterimageArray.GetLength(0); i++)
         {
            for (int j = 0; j < filterimageArray.GetLength(1); j++)
            {
               int red = (int)filterimageArray[i, j, 0];
               int green = (int)filterimageArray[i, j, 1];
               int blue = (int)filterimageArray[i, j, 2];
               bitmapreslut.SetPixel(i, j, Color.FromArgb(255, red,
green, blue));
                                                }
```

L.Image to be Converted into Gray scale

Here the source code implements the convolution method and extension method to target bitmap classes. The motive behind the using of convolution filter is to perform convolution of defined kernel3x3 which I have predefined in the code and convert the image into a gray scale image. The implementation follows here

```
// convertion of input image to gray scale

 if (grayscale == true)
         {
            float redgreenblue = 0;

            for (int k = 0; k < pixelBuffer.Length; k += 4)
            {
               redgreenblue = pixelBuffer[k] * 0.11f;
               redgreenblue += pixelBuffer[k + 1] * 0.59f;
               redgreenblue += pixelBuffer[k + 2] * 0.3f;


               pixelBuffer[k] = (byte)redgreenblue;
               pixelBuffer[k + 1] = pixelBuffer[k];
               pixelBuffer[k + 2] = pixelBuffer[k];
               pixelBuffer[k + 3] = 255;
            }
                  }
```

M.Run Method for Interface IPipeline module

```
/// Interface IPipeline run method

      public double[,,] Run(double[,,] data, IContext ctx)
      {

         return Convolutionfilter(data, LaplacianOfGaussian);
   }
```

Each pipeline elements will read and write the common data from the input and output. And all the pipeline elements are passive and have input and output.

N.Convert image from Bitmap to 3D Array

```
   public static double[,,] BitmapTo3dArray(Bitmap bitmapimage)
```

```
    {
        int resultingimgWidth = bitmapimage.Width;
        int resultingimgHeight = bitmapimage.Height;

        //Required values for RGB 0 -> Red, 1 -> Green, 2 ->
Blue
        double[,,] imageArray = new
double[resultingimgWidth, resultingimgHeight, 3];

        for (int i = 0; i < resultingimgWidth; i++)
        {
            for (int j = 0; j < resultingimgHeight; j++)
            {
                Color variablecolor = bitmapimage.GetPixel(i, j);
                imageArray[i, j, 0] = variablecolor.R;
                imageArray[i, j, 1] = variablecolor.G;
                imageArray[i, j, 2] = variablecolor.B;
            }
                                        }
```

O.Final Output Image

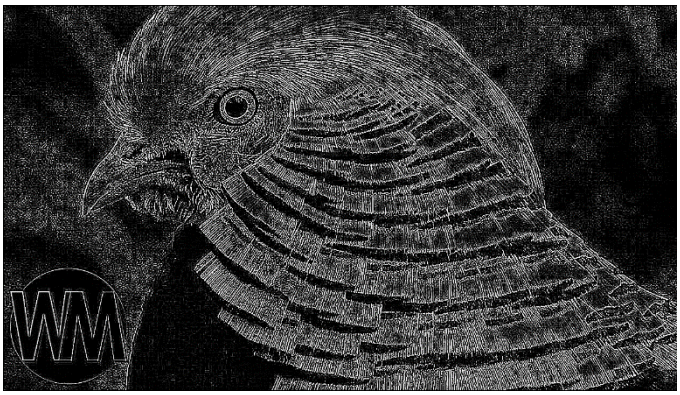

Figure 11: Output Image after LoG Filter parrot

```
string LocalDirectory =
AppDomain.CurrentDomain.BaseDirectory;
            string imagepath = Path.Combine(LocalDirectory,
"InputImages\\parrot.jpg");
            double[,,] imagedata = UploadImage(imagepath);
            return imagedata;
        });

            api.UseLaplacianOfgaussian();

            double[,,] imageoutput = api.Run() as double[,,];

            Assert.IsNotNull(imageoutput);
string Directorybase = AppDomain.CurrentDomain.BaseDirectory;
            string outputPath = Path.Combine(Directorybase,
"OutputImages\\parrot.jpg");
            SaveImage(imageoutput, outputPath);
        }
```

Fig.12 shows identified and much more thinner borders than the gaussian method.

IV Discussion and Conclusion

The main idea of the project is to implement the Laplacian of Gaussian filter for Learning API. The project is successfully implemented for Learning API by using Run method. We have detected the edges of the image with help of the Laplacian of gaussian. Compare to Laplacian this filter has many advantages. The output image smoothened and noise less and edges are detected we used more than two deviations to include all scales and applying the gaussian blur filter before the convolution will reduce the generated artifacts noise. So, for detecting the edges of an image is better to use the Laplacian of Gaussian filter instead of using Laplacian filter because it is very sensitive to noise in which noise is removed in Laplacian of Gaussian filter and gaussian filter should be applied to remove the noise by using gaussian blur.

V. References

[1]. https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm

[2]. https://automaticaddison.com/how-the-laplacian-of-gaussian-filter-works/

[3]. https://academic.mu.edu/phys/matthysd/web226/Lab02.htm

[4]. http://www.cse.psu.edu/~rtc12/CSE486/lecture11_6pp.pdf

[5]. http://www.roborealm.com/help/LOG.php

[6]. https://theailearner.com/2019/05/25/laplacian-of-gaussian-log/