ML19/20-3.13. Validate and improve Tests of Existing Algorithms: Gaussian & Mean Algorithm, Ring Projection Algorithm

Jayashree Regoti, Shruti Mathure
jayashree.regoti@stud.fra-uas.de, shruti.mathure@stud.fra-uas.de

***Abstract*— In image filtering, Gaussian and mean filter is used to modify the image by removing noise and to get original images at the output and ring algorithm is used to identify the image at any orientation.**

Gaussian filters are implemented on images which are used to remove the noise that are present in the image while capturing or injected into the image at the time of transmission.

Mean filter is mainly used to remove noise in the image, the main advantage of mean filter over Gaussian filter is that in mean filter the noise is removed while edges are kept relatively sharp.

In ring- projection, the image is identified when the original image is kept at different angles. This reduces the orientation problem and identifies the image. Ring Projection Algorithm is used in pattern recognition. In this Projection operation the pixels of the pattern are projected along a straight line such as horizontal or vertical line.

Keywords—Gaussian Filter, Mean filter, Ring Projection, Blur, Orientation

I. INTRODUCTION

The Gaussian filter is extensively utilized in image processing and computer vision for long years. For the most part it works with low pass filtering. These filters have been appeared to assume a significant job in edge detection in the human visual framework, and to be to a great extent valuable as detection for edge and line recognition. Conventional linear filters, for example, Gaussian mean filter and Gaussian filter smooth noise successfully however blur edges. The Gaussian smoothing operator is a 2-D convolution administrator that is utilized to 'blur' images and evacuate detail and noise. Right now, resembles the mean filter, yet it utilizes an alternate kernel that represents to the shape of a Gaussian ('bell-shaped') hump.
The Ring Projection Algorithm is used to solve the orientation issues. In this algorithm, the image can be recognized when placed in any angle. Even if the image is rotated to 180 degree or 270 degree, the algorithm matches with the original image and identifies it.

II. METHODOLOGY

In the concept of image processing, a Gaussian smoothing (also called as Gaussian blur) is used to blur the image by Gaussian filter. It is mostly used in graphics software, to eliminate noise in the image. The main ideology of Gaussian smoothing is to use the 2-dimensional distribution as a `point-spread' function, and this can be processed by convolution. where **σ** is the standard deviation for Gaussian distribution.

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

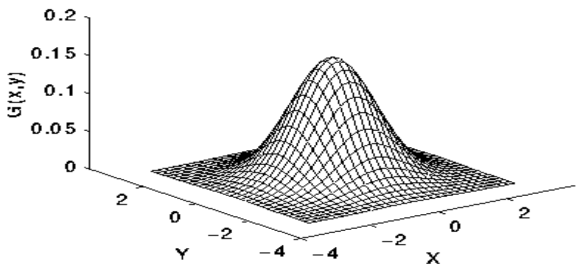


Fig 2.1: 2-D Gaussian distribution with mean (0,0) and σ =1

An integer valued 5/5 convolution kernel approximating a Gaussian with a σ of 1 is shown in fig.

$$\frac{1}{273}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline\end{array}$$

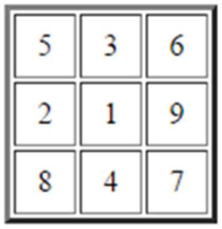
Fig 2.2: Gaussian Function

Gaussian filtering G is used in the equation. The Gaussian filter is also referred as non-uniform low pass filter. The kernel coefficients diminish with increasing distance from the kernel's center. Central pixels will have a high weighting than those on the periphery. Larger values of σ gives a wider peak (more blurring). Kernel size must increase with increasing σ to maintain the Gaussian value of the filter.
Gaussian kernel coefficients depend on the standard deviation value (σ).At the edge of the mask, coefficients must be near 0.The kernel is not directionally bias and is rotationally symmetric. Gaussian kernel is separable which allows fast computation 2.5. Gaussian kernel is distinct, which permits quick calculation. Gaussian filters might not preserve image brightness to blur images and remove noise and detail.

Mean Filter:

The Mean filter is also known as sliding-window spatial filter where the central element is replaced by the average of all the pixel values in the window. The window, or kernel, is generally a square but can be of any shape. The mean filtering is calculated below. Here a single 3x3 window of values are taken.

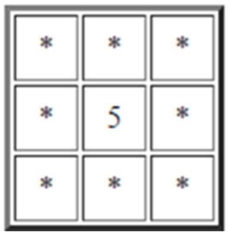
Unfiltered values
5 + 3 + 6 + 2 + 1 + 9 + 8 + 4 + 7 = 45
45 / 9 = 5



5	3	6
2	1	9
8	4	7

Fig 2.3

Mean Filter



*	*	*
*	5	*
*	*	*

Fig 2.4

The Mean filter is used to replace each pixel value in an image with the average value of its surrounding pixels, including itself. In the mean filter the effect of removing pixel values which are not represented of their neighboring pixels. Mean filtering is also called as a convolution filter.

Algorithm:

Gaussian Filter

Here we have taken an image as an input and the image is divided into pixels and each pixel is divided into RGB.

RGB (Red, Green and Blue) is the color space for digital images. We use RGB color mode to display anything on screen. A light source within a device creates any color you need by mixing red, green and blue and varying their intensity.

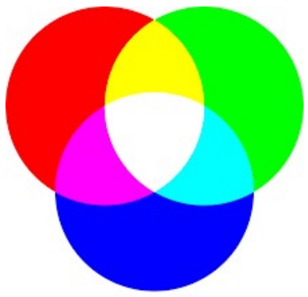


Fig 2.5 RGB

A pixel is equal to average = (red + green + blue)/3. We have taken 3*3 Matrix. For each pixel we have calculated the gaussian value using gaussian filter.

Once we find individual color pixel value, we then take the average of all 3 color pixels. We must take average of all the color pixels to get a new pixel value. These original pixels are replaced by these values.

Gaussian and Mean Filter:
In these firstly gaussian filter is applied then on gaussian filter image the mean filter is applied.

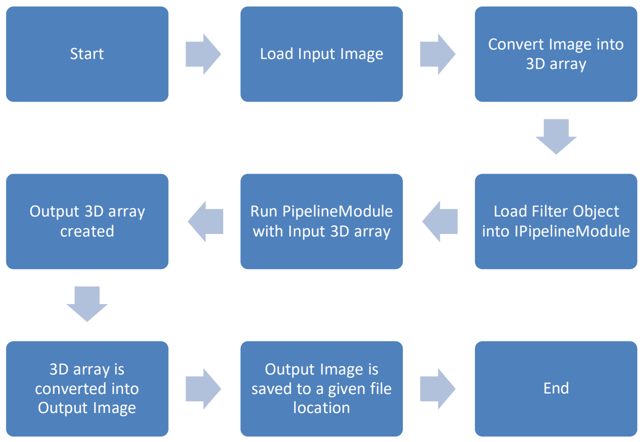

Start
Load Input Image
Convert Image into 3D array
Load Filter Object into IPipelineModule
Run PipelineModule with Input 3D array
Output 3D array created
3D array is converted into Output Image
Output Image is saved to a given file location
End

Fig 2.6: Architecture of Gaussian & Mean Filter

The algorithm was implemented in the .NET standard 2.0 framework. It is also based on the interface "IPipeLineModule" of the LearningApi, which has a 3-dimensional double array as input and output. After the conversion into the 3-dimensional array, the algorithm is executed in the main class "Gaussian & Mean Filter". To see the effect of the filter after executing the algorithm, the 3D double array has to be converted back into a bitmap and saved.

Ring Projection Transform

In ring projection suppose we have a set of M-patterns which are classified as M -classes and each class has only one pattern sample and if we have S-different sizes of each pattern and allow R different orientation. The first one is to treat them as single pattern sample classes. This means the uncertainty will increase considerably. For the above inputs we will have M x S x R classes. The second solution is that we only consider the categories of the pattern and disregard their sizes or orientation. This means the uncertainty will remain same as that of the set with M classes.

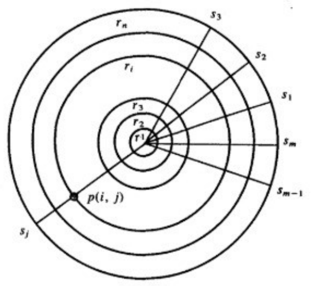


Fig 2.7: Ring-extraction panel

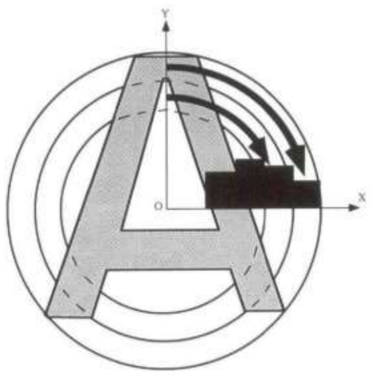


Fig 2.8: Ring projection

The ring projection Algorithm can be summarized as

Step 1: Find the center of gravity and translate it to the origin of the image plane.

Step 2: Find the largest distance d.

Step 3: Skill the input image by D/d.

Step 4: Find the ring-projection vector using ring projection operation.

$$p_{r_i} = \sum_{\theta=0}^{2\pi} f^*(i, \theta) , \qquad i = 1, 2, \ldots, n .$$

Step 5: Find the feature vector by accumulating the p $_{ri}$ `s

$$[P_1 \quad P_2 \quad \cdots \quad P_i \quad \cdots \quad P_n]^T ,$$

where,

$$P_1 = p_{r_1} ;$$

$$P_2 = p_{r_1} + p_{r_2} ;$$

$$\cdots\cdots$$

$$P_i = \sum_{k=1}^{i} p_{r_k} ;$$

$$\cdots\cdots$$

$$P_n = p_{r_1} + p_{r_2} + \cdots + p_{r_{n-1}} + p_{r_n} .$$

In Ring Projection Transform the row and column is divided by 2 taken from the center. The radius is calculated by taking the square root. The value of the center should be greater than the value of row and column.

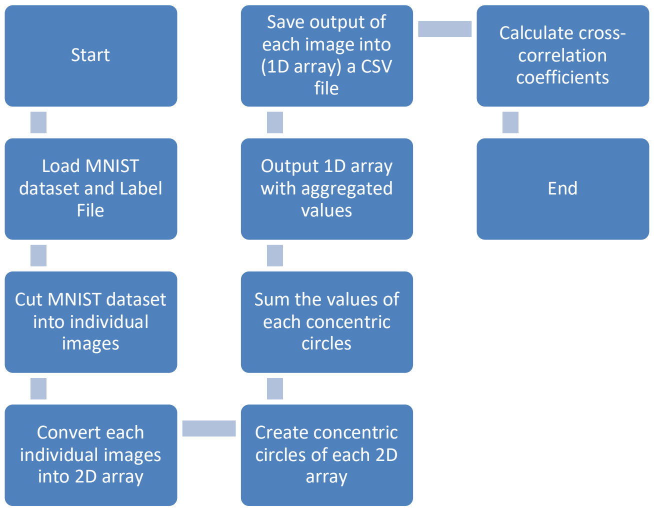

Start
Load MNIST dataset and Label File
Cut MNIST dataset into individual images
Convert each individual images into 2D array
Create concentric circles of each 2D array
Sum the values of each concentric circles
Output 1D array with aggregated values
Save output of each image into (1D array) a CSV file
Calculate cross-correlation coefficients
End

Fig 2.9: Architecture of Ring Projection algorithm

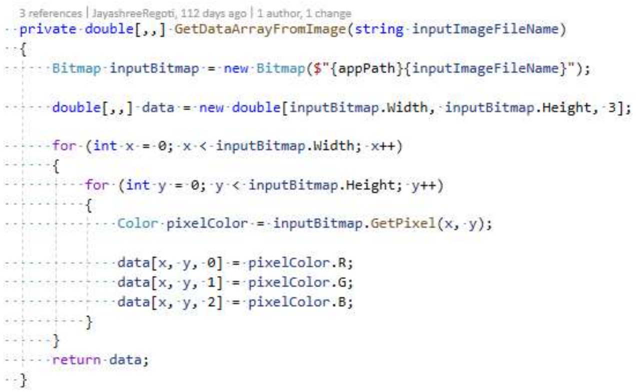
For Ring Projection Algorithm, which has a B/W image as an input. Since the input of projection algorithm is an image, it must be converted from a bitmap to 2D array. The loading and the conversion of the image takes place in a class called "LoadMNISTPipelineModule". After the parameters for the radius and the loops have been specified, an image can finally be loaded.

After the execution we get the result in 1D array for each image. Then we calculate the cross-correlation coefficients which can be used to train neural network to classified different numbers in any orientation.

III. RESULTS

Gaussian and Mean Algorithm

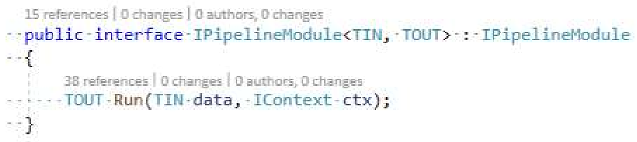
Load and Convert Image:
Since the default for the interface "IPipeLineModule" provides an input and output as object type, thus for calculation the image must be converted from the data type bitmap to double. This is done in the "GetDataArrayFromImage" method in the unit test class "GaussianAndMeanFilterTest".



```
3 references | JayashreeRegoti, 112 days ago | 1 author, 1 change
private double[,,] GetDataArrayFromImage(string inputImageFileName)
{
    Bitmap inputBitmap = new Bitmap($"{appPath}{inputImageFileName}");

    double[,,] data = new double[inputBitmap.Width, inputBitmap.Height, 3];

    for (int x = 0; x < inputBitmap.Width; x++)
    {
        for (int y = 0; y < inputBitmap.Height; y++)
        {
            Color pixelColor = inputBitmap.GetPixel(x, y);

            data[x, y, 0] = pixelColor.R;
            data[x, y, 1] = pixelColor.G;
            data[x, y, 2] = pixelColor.B;
        }
    }
    return data;
}
```

Fig 3.1 Conversion of Image to Data Array

Apply Filter:
These Algorithm was implemented with the interface IPipeLineModule of the LearningApi. The only method of this interface has is Run().

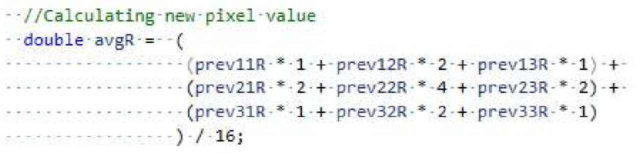


```
15 references | 0 changes | 0 authors, 0 changes
public interface IPipelineModule<TIN, TOUT> : IPipelineModule
{
    38 references | 0 changes | 0 authors, 0 changes
    TOUT Run(TIN data, IContext ctx);
}
```

Fig 3.2:

Each pixel is divided into 3 values (RGB), thus we have three 2D arrays and value of each pixel in each array is calculated from its neighboring pixels.

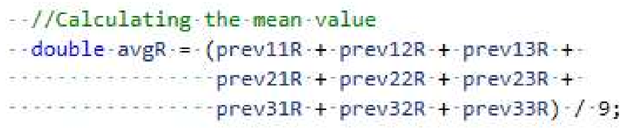
In class "GaussianandMeanFilter" we have calculated each pixel value by taking the addition of all neighboring pixels (multiplied with its corresponding weights) and dividing by 16. These can be illustrated by,



```
//Calculating new pixel value
double avgR = (
                (prev11R * 1 + prev12R * 2 + prev13R * 1) +
                (prev21R * 2 + prev22R * 4 + prev23R * 2) +
                (prev31R * 1 + prev32R * 2 + prev33R * 1)
              ) / 16;
```

Fig 3.3: Calculating the new pixel value of Gaussian Filter

Now,
In Mean filter the result is carried out by taking the addition of all neighboring pixels and dividing by 9.



```
//Calculating the mean value
double avgR = (prev11R + prev12R + prev13R +
               prev21R + prev22R + prev23R +
               prev31R + prev32R + prev33R) / 9;
```

Fig 3.4: Calculating the new pixel value of Mean Filter

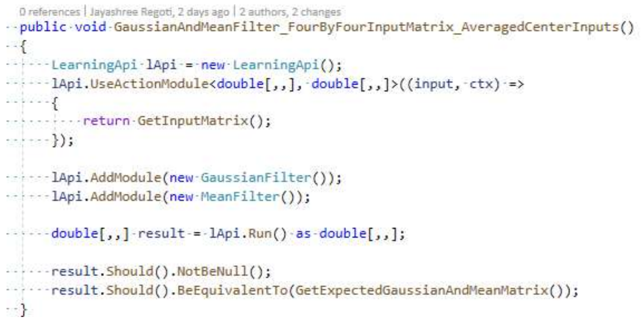
If we apply Gaussian Filter and Mean Filter on the same image then, firstly, the gaussian operation is performed and on the smoothened image mean operation is performed.

Unit Test

For Gaussian and Mean Filtering, we have given 2 different types of input.

1) 4*4 Matrix

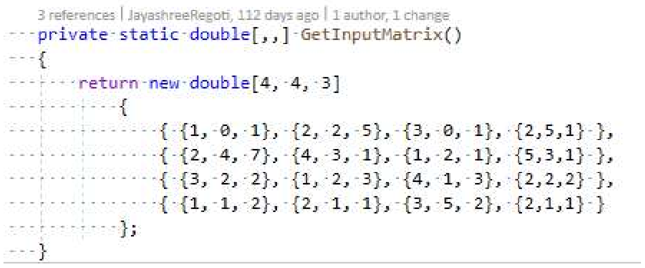
In learning API the Gaussian and Mean filter is implemented on input matrix and is compared with the expected output to check with the precision. In the class basic Gaussian And Mean Test.



```
0 references | Jayashree Regoti, 2 days ago | 2 authors, 2 changes
public void GaussianAndMeanFilter_FourByFourInputMatrix_AveragedCenterInputs()
{
    LearningApi lApi = new LearningApi();
    lApi.UseActionModule<double[,,], double[,,]>((input, ctx) =>
    {
        return GetInputMatrix();
    });

    lApi.AddModule(new GaussianFilter());
    lApi.AddModule(new MeanFilter());

    double[,,] result = lApi.Run() as double[,,];

    result.Should().NotBeNull();
    result.Should().BeEquivalentTo(GetExpectedGaussianAndMeanMatrix());
}
```

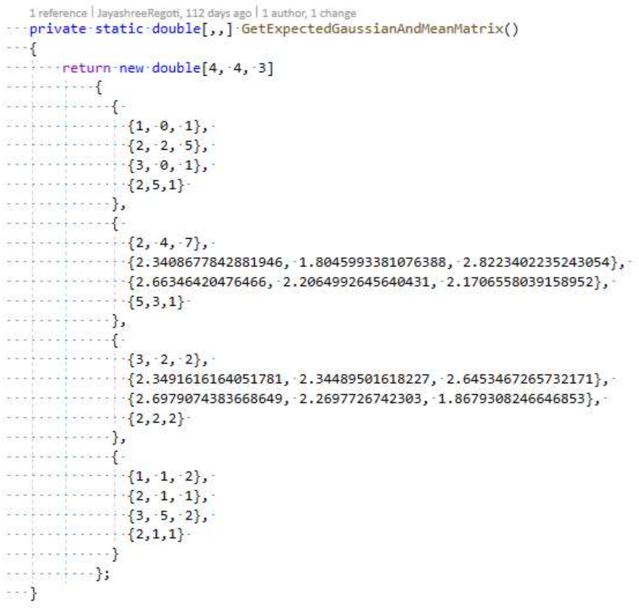
Fig 3.5

In this the input matrix is taken as,



```
3 references | JayashreeRegoti, 112 days ago | 1 author, 1 change
private static double[,,] GetInputMatrix()
{
    return new double[4, 4, 3]
        {
            { {1, 0, 1}, {2, 2, 5}, {3, 0, 1}, {2,5,1} },
            { {2, 4, 7}, {4, 3, 1}, {1, 2, 1}, {5,3,1} },
            { {3, 2, 2}, {1, 2, 3}, {4, 1, 3}, {2,2,2} },
            { {1, 1, 2}, {2, 1, 1}, {3, 5, 2}, {2,1,1} }
        };
}
```

Fig 3.6
After applying filter, the expected output would be



```
1 reference | JayashreeRegoti, 112 days ago | 1 author, 1 change
private static double[,,] GetExpectedGaussianAndMeanMatrix()
{
    return new double[4, 4, 3]
        {
            {
                {1, 0, 1},
                {2, 2, 5},
                {3, 0, 1},
                {2,5,1}
            },
            {
                {2, 4, 7},
                {2.3408677842881946, 1.8045993381076388, 2.8223402235243054},
                {2.66346420476466, 2.2064992645640431, 2.1706558039158952},
                {5,3,1}
            },
            {
                {3, 2, 2},
                {2.3491616164051781, 2.34489501618227, 2.6453467265732171},
                {2.6979074383668649, 2.2697726742303, 1.8679308246646853},
                {2,2,2}
            },
            {
                {1, 1, 2},
                {2, 1, 1},
                {3, 5, 2},
                {2,1,1}
            }
        };
}
```

Fig 3.7

2) 3D image

The 3D input image, the output after applying the image filtration is compared with expected output. As double[,,] it is not possible to view the filtered image. Therefore, it is necessary to convert the data of the 3D double array back to the data type bitmap.
Once the process is done the output image is saved.

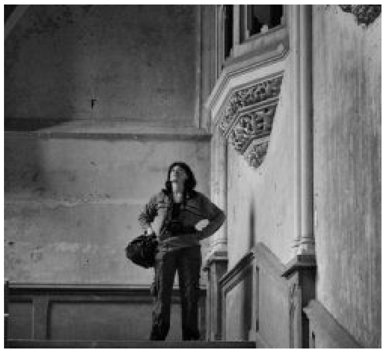


Fig 3.9(a): Input Image

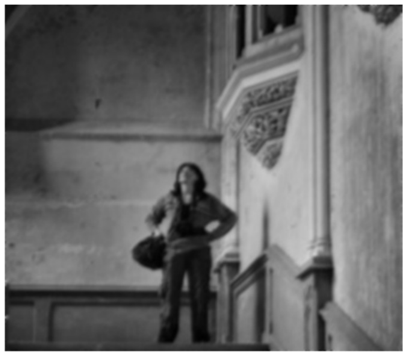


Fig 3.9(b): Mean Filter Output

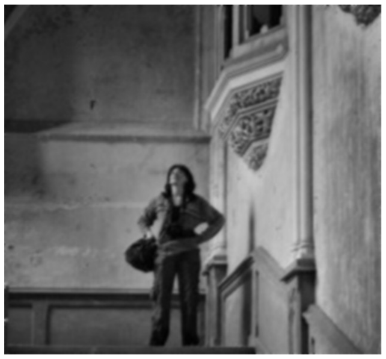


Fig 3.9(c): Gaussian Filter Output

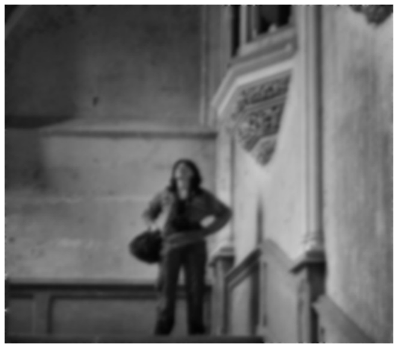


Fig 3.9(d): Gaussian and Mean Filter Output

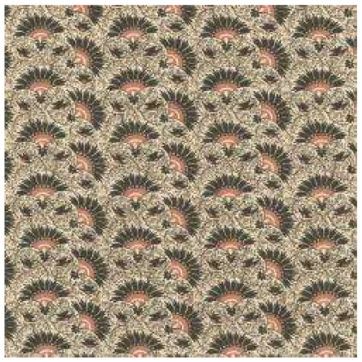


Fig 3.10(a): Input Image

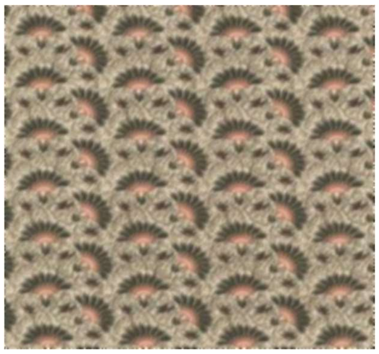


Fig 3.10(b): Mean Output

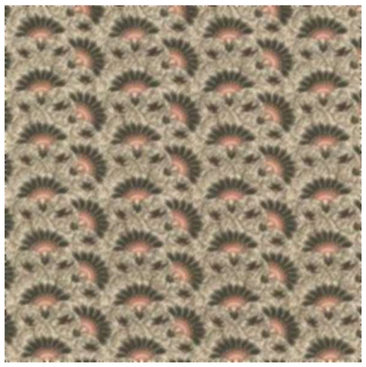


Fig 3.10(c): Gaussian Output

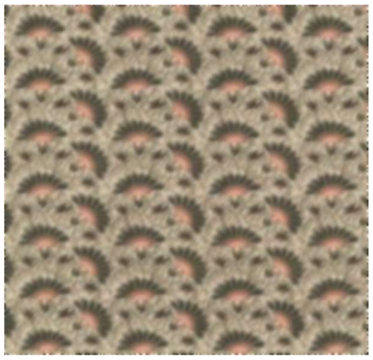


Fig 3.10(d): Gaussian and Mean Output

Ring Projection Algorithm

In Ring Projection Algorithm we have given four similar images with different orientation.
In result by algorithm calculations the radius and the corresponding value has been calculated, By comparing these values all the input images have different orientation the result comes out to be positive (i.e. original image (Fig 3.11(a))).

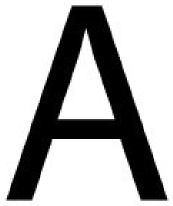


Fig 3.11(a): Letter A

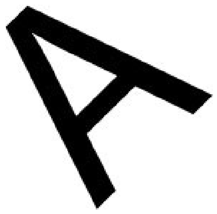


Fig 3.11(b): Letter A(-45degree)

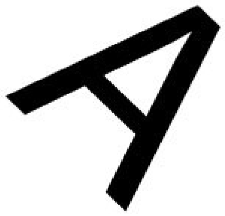


Fig 3.11(c): Letter A(45degree)

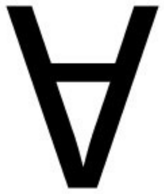


Fig 3.11(d): Letter A (180 degree)

Unit Test

In Ring Projection Algorithm we have taken 2 different types of input.

1) Input images at different orientation (as shown above):
    a. Where the csv output of each image is very less deviation
2) MNIST Images (dataset and labels)
    a. Which is used as a training data for neural network to recognize handwritten numbers

MNIST is determined by,

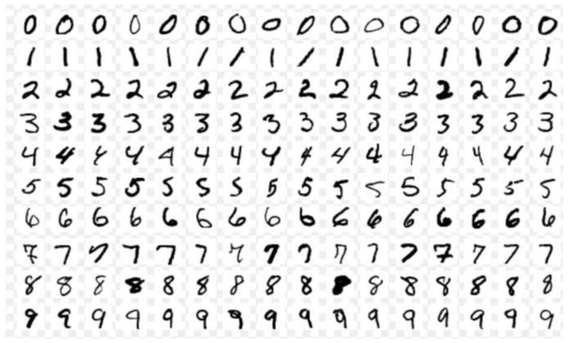
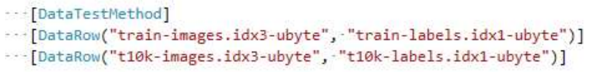


Fig 3.12

Image dimension reduction of MNIST database of handwritten digits (training data).



```
    [DataTestMethod]
    [DataRow("train-images.idx3-ubyte", "train-labels.idx1-ubyte")]
    [DataRow("t10k-images.idx3-ubyte", "t10k-labels.idx1-ubyte")]
```

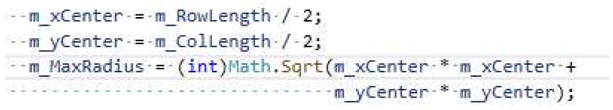
Fig 3.13

Load & Convert Image:

Here we use MNIST dataset and create images of numbers and save them.
This is done in the " LoadMNISTPipelineModule()" which creates 2D array of individual images which are saved using "DoubleArrrayToBitmapPipelineModule()" and " BitmapToImageFilePipelineModule()".

Perform Transform:

In a Ring Projection Algorithm,
The input image is characterized by column length and row length and using the following input we can determine xCenter, yCenter, MaxRadius by,



```
  m_xCenter = m_RowLength / 2;
  m_yCenter = m_ColLength / 2;
  m_MaxRadius = (int)Math.Sqrt(m_xCenter * m_xCenter +
                              m_yCenter * m_yCenter);
```

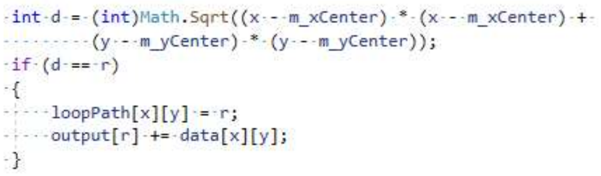
Fig 3.14

After determining all the parameters, the double output is calculated as,

output = new double [m_MaxRadius + 1];

Once the loop is created and the values are determined by each loop of different input images it is then compared with

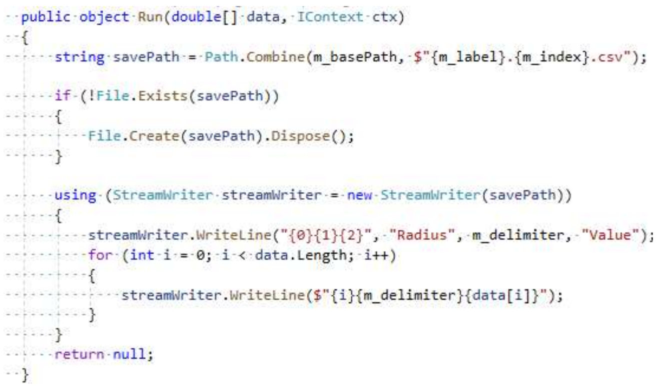
MaxRadius to get the data.



```
int d = (int)Math.Sqrt((x - m_xCenter) * (x - m_xCenter) +
          (y - m_yCenter) * (y - m_yCenter));
if (d == r)
{
    loopPath[x][y] = r;
    output[r] += data[x][y];
}
```

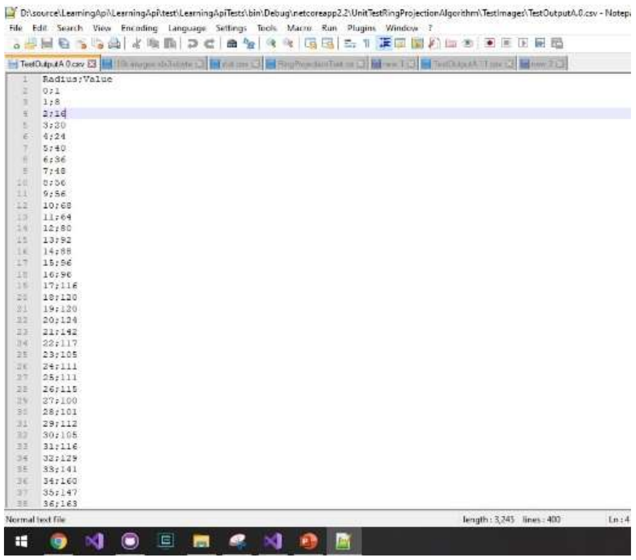
Fig 3.15

After determining all the loops in the image and calculationg the values of each loop of all input images. The array output is saved in a CSV file using "RingProjectionFunctionToCSVPipelineModule".



```
  public object Run(double[] data, IContext ctx)
  {
      string savePath = Path.Combine(m_basePath, $"{m_label}.{m_index}.csv");

      if (!File.Exists(savePath))
      {
          File.Create(savePath).Dispose();
      }

      using (StreamWriter streamWriter = new StreamWriter(savePath))
      {
          streamWriter.WriteLine("{0}{1}{2}", "Radius", m_delimiter, "Value");
          for (int i = 0; i < data.Length; i++)
          {
              streamWriter.WriteLine($"{i}{m_delimiter}{data[i]}");
          }
      }
      return null;
  }
```

Fig 3.16



D:\source\LearningApi\LearningApi\test\LearningApiTests\bin\Debug\netcoreapp2.2\UnitTestRingProjectionAlgorithm\TestImages\TestOutputA.0.csv - Notepa

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
TestOutputA.0.csv

```
Radius;Value
0;1
1;8
2;16
3;20
4;24
5;40
6;36
7;48
8;56
9;56
10;68
11;64
12;80
13;92
14;88
15;96
16;96
17;116
18;120
19;120
20;124
21;142
22;117
23;105
24;111
25;111
26;115
27;100
28;101
29;112
30;105
31;116
32;129
33;141
34;160
35;147
36;163
```

Normal text file
length : 3,245 lines : 400
Ln : 4

Fig 3.17

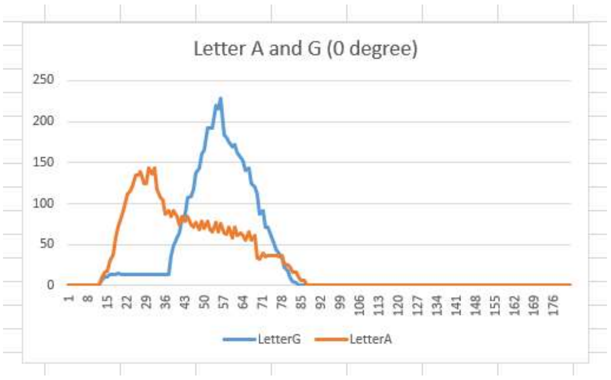
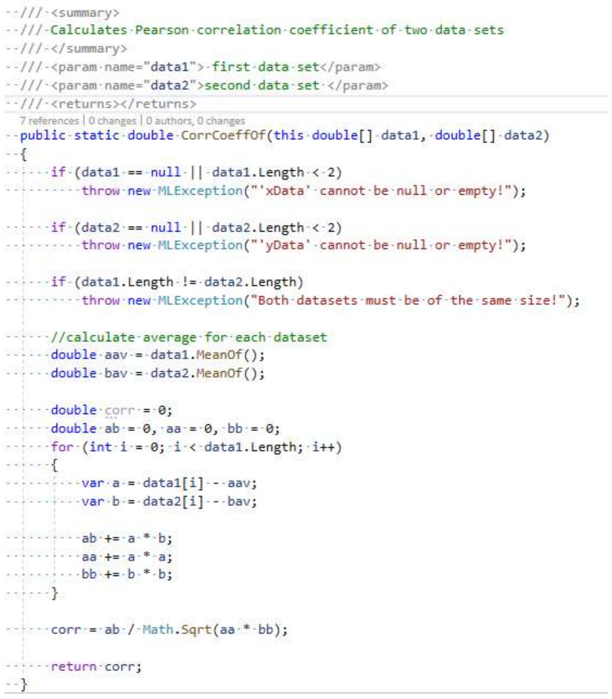


Fig 3.18 Graphical representation of Ring Projection Output

Once we determine all the values from the input sample, we need to calculate cross-correlation coefficient so that we can compare average, minimum, maximum, deviation of different input images.



```
  /// <summary>
  /// Calculates Pearson correlation coefficient of two data sets
  /// </summary>
  /// <param name="data1">first data set</param>
  /// <param name="data2">second data set </param>
  /// <returns></returns>
  7 references | 0 changes | 0 authors, 0 changes
  public static double CorrCoeffOf(this double[] data1, double[] data2)
  {
      if (data1 == null || data1.Length < 2)
          throw new MLException("'xData' cannot be null or empty!");

      if (data2 == null || data2.Length < 2)
          throw new MLException("'yData' cannot be null or empty!");

      if (data1.Length != data2.Length)
          throw new MLException("Both datasets must be of the same size!");

      //calculate average for each dataset
      double aav = data1.MeanOf();
      double bav = data2.MeanOf();

      double corr = 0;
      double ab = 0, aa = 0, bb = 0;
      for (int i = 0; i < data1.Length; i++)
      {
          var a = data1[i] - aav;
          var b = data2[i] - bav;

          ab += a * b;
          aa += a * a;
          bb += b * b;
      }

      corr = ab / Math.Sqrt(aa * bb);

      return corr;
  }
```

Fig 3.19

After calculating the cross-correlation, the deviation is verified and should be less than '1.5'.

IV. CONCLUSION

In Gaussian smoothing sigma and the window size is used and Gaussian filter blurs the image to decrease the noise from the image. Also, the Mean Filter does the same functionality and blurs the image and removes the noise. A Gaussian filter is a linear filter. It is used to blur the image or to reduce noise. If you include both of them and subtract, you can use them for "unsharp masking" (edge detection). The Gaussian filter can used to blur edges and reduce contrast. Mean filter is a non-linear filter that is mostly used as a simple way to reduce noise in an image. It claims to fame (over Gaussian for noise reduction) is that it removes noise while keeping edges relatively sharp.

The Ring Projection Algorithm reduces orientation problems and it is computationally efficient, insensitive to noise, High discriminating power and scale of characters. It helps a lot in image alignment, which is a fundamental technique that has many applications for machine vision and image processing, including image retrieval, object recognition, pose estimation, industrial inspection, and target tracking. This creates the possibility of establishing a fast image alignment algorithm that can be used for industrial inspection applications.

V. REFERENCES

[1] Koedkurang, K., & Rungsipanich, A. (2018, July). An Adaptive Gaussian filter for merging JICA and MOST DEM. In 2018 15th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) (pp. 736-739). IEEE.

[2] Devapriya, W., Nelson Kennedy Babu, C. and Srihari, T. (2016) Real Time Speed Bump Detection Using Gaussian Filtering and Connected Component Approach. Circuits and Systems, 7, 2168-2175.

[3] Bunke, H., & Wang, P. S. P. (1997). Handbook of character recognition and document image analysis. World scientific.

[4] Tang, Y. Y., Li, B. F., Ma, H., & Lin, J. (1998). Ring-projection-wavelet-fractal signatures: a novel approach to feature extraction. IEEE Transactions on circuits and systems II: Analog and digital signal processing, 45(8), 1130-1134.