International Journal of Engineering Research and Advanced Technology (IJERAT)

E-ISSN : 2454-6135

DOI: 10.31695/IJERAT.2021.3736

Volume.7, No. 8
August -2021

Edge Detection for Face Image Using Multiple Filters

Haitham Salman Chyad [1], Raniah Ali Mustafa [2], & Zainab Yasser Mohamed [3]

[1-3]Mustansiriyah University

Department of Computer Science

College of Education,

Iraq

ABSTRACT

Edge detection is a technique for detecting the presence and location of sharp changes in an image's intensity. Edges help in segmentation and object recognition by defining the boundaries between sections in an image. Edge detection minimizes the quantity of data in an image and filters out unnecessary information while keeping the image's crucial structural qualities. The general method of edge detection is to study the changes of a single image pixel in an area, use the variation of the edge neighboring first-order or second-order to detect the edge. This paper is an overview of different edge detection techniques like differential operator method such as Sobel operator, canny technique and we used Visual Basic Language version 6.0 for this paper.

Key Words: *Edge Detection, Roberts Edge Detection, Sobel Edge Detection, Prewitt Edge Detection, Canny Edge Detection.*

1. INTRODUCTION

In image processing, edge detection approaches based on difference operations are commonly utilized. It can detect changes in gray levels, but it is susceptible to noise. In image processing, edge detection is a crucial task. Pattern recognition, image segmentation, and scene analysis all use it. An edge detector is just a high pass filter that can be used to retrieve the image's edge points. An image's edge is a contour where the brightness of the image rapidly shifts. An edge is frequently regarded as one class of singularities in image processing. In a function, singularities can be characterized easily as discontinuities where the gradient approaches infinity. However, image data is discrete, so edges in an image often are defined as the local maxima of the gradient [1, 2]. Edge widely exists between objects and backgrounds, objects and objects, primitives and primitives. The edge of an object is reflected in the discontinuity of the gray. Edge extraction is an important technique in graphics processing and feature extraction. The basic idea of edge detection is as follows: First, use edge enhancement operator to highlight the local edge of the image. Then, define the pixel "edge strength" and set the threshold to extract the edge point set.

Face is one of the most appropriate biometrics to monitor applications [3]. In social life, faces are our fundamental focus and they play an important role in expressing emotions and identity [4]. Face has gained an increasing amount of interest in computer vision and machine learning during the past few years, due to the fact that face recognition technology can be applied in a wide range of areas, such as, access control, identity authentication and others [5].

Dataset contains many faces; the process of matching a face to any of faces in the dataset is called Face Recognition. Although the existing systems are as yet far away from the ability of the individual perception system, numerous techniques have been suggested and much progress has been made to recognize faces beneath small variation in facial expressions, orientations, and illumination [6].

2. EDGE DETECTION TECHNIQUES

Edge detection is a fundamental tool for image segmentation. Edge detection methods transform original images into edge images benefits from the changes of grey tones in the image. In image processing especially in computer vision, the edge detection treats the localization of important variations of a gray level image and the detection of the physical and geometrical properties of objects of the scene. It is a fundamental process detects and outlines of an object and boundaries among objects and the background in the image. Edge detection is the most familiar approach for detecting significant discontinuities in intensity values .

Edges are local changes in the image intensity. Edges typically occur on the boundary between two regions. The main features can be extracted from the edges of an image. Edge detection has major feature for image analysis [7]. These features are used by advanced computer vision algorithms. Edge detection is used for object detection which serves various applications like medical image processing, biometrics etc [8].

2.1 Roberts Edge Detection

The Roberts edge detection is introduced by Lawrence Roberts (1965). It performs a simple, quick to compute, 2-D spatial gradient measurement on an image. This method emphasizes regions of high spatial frequency which often correspond to edges. The input to the operator is a grayscale image the same as to the output is the most common usage for this technique. Pixel values in every point in the output represent the estimated complete magnitude of the spatial gradient of the input image at that point.

-1	0
0	+1

G_x

0	-1
+1	0

G_y

These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy| \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

which is much faster to compute [9][10].

2.2 Sobel Edge Detection

The Sobel edge detection method is introduced by Sobel in 1970. The Sobel method of edge detection for image segmentation finds edges using the Sobel approximation to the derivative. It precedes the edges at those points where the gradient is highest. The Sobel technique performs a 2-D spatial gradient quantity on an image and so highlights regions of high spatial frequency that correspond to edges. In general it is used to find the estimated absolute gradient magnitude at each point in n input grayscale image. In conjecture at least the operator consists of a pair of 3x3 complication kernels as given away in under table [11]. One kernel is simply the other rotated by 90o. This is very alike to the Roberts Cross operator. The Sobel operators have the advantage of providing both a differencing and a smoothing effect. Because derivatives enhance noise, the smoothing effect is particularly attractive feature of the Sobel operators [9].

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

G_x

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

G_y

2.3 Prewitt Edge Detection

The Prewitt edge detection is proposed by Prewitt in 1970 [12]. To estimate the magnitude and orientation of an edge Prewitt is a correct way. Even though different gradient edge detection wants a quiet time consuming calculation to estimate the direction from the magnitudes in the x and y-directions, the compass edge detection obtains the direction directly from the kernel with the highest response.The Prewitt operator uses the same equations as the Sobel operator, except that the constant c = 1. Therefore:

$$s_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline\end{array} \qquad s_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$$

Note that, unlike the Sobel operator, this operator does not place any emphasis on pixels that are closer to the center of the masks. At each point in the image, the result of the Prewitt operator is either the corresponding gradient vector or the normal of this vector. The Prewitt 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.

2.4 Canny Edge Detection

The Canny edge detector is the first derivative of a Gaussian and closely approximates the operator that optimizes the product of signal-to-noise ratio and localization. The Canny edge detection algorithm is summarized by the following notation. Let I[i, j] denote the image [13]. The result from convolving the image with a Gaussian-smoothing filter using separable filtering is an array of smoothed data.

$$S[i,j] = G[i,j;\sigma] \star I[i,j], \qquad \text{…………………(3)}$$

The canny edge detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image [14].The performance of the Canny algorithm depends heavily on the adjustable parameters, σ, which is the standard deviation for the Gaussian filter, and the threshold values, th and tl. The gradient of the smoothed array S [i, j] can be computed using the 2x2 first-difference approximations to produce two arrays P [i,j] and Q [i,j] for the x and y partial derivatives:

$$\begin{aligned} P[i,j] &\approx (S[i,j+1] - S[i,j] \\ &\qquad + S[i+1,j+1] - S[i+1,j])/2 \\ Q[i,j] &\approx (S[i,j] - S[i+1,j] \\ &\qquad + S[i,j+1] - S[i+1,j+1])/2. \end{aligned}$$

The finite differences are averaged over the 2 x 2 square so that the x and y partial derivatives are computed at the same point in the image. The magnitude and orientation of the gradient can be computed from the standard formulas for rectangular-to-polar conversion:

$$\begin{aligned} M[i,j] &= \sqrt{P[i,j]^2 + Q[i,j]^2} \\ \theta[i,j] &= \arctan(Q[i,j], P[i,j]), \end{aligned}$$

Where the arctan function takes two arguments and generates an angle over the entire circle of possible directions. These functions must be computed efficiently, preferably without using floating-point arithmetic. It is possible to compute the gradient magnitude and orientation from the partial derivatives by table lookup. The arctangent can be computed using mostly fixed-point arithmetic2 with a few essential floating-point calculations performed in software using integer and fixed-point arithmetic.

2.4.1 The Canny Edge Detection Algorithm

The algorithm runs in 5 separate steps:

1. Smoothing: Blurring of the image to remove noise.
2. Finding gradients: The edges should be marked where the gradients of the image has large magnitudes.
3. Non-maximum suppression: Only local maxima should be marked as edges.
4. Double thresholding: Potential edges are determined by thresholding.
5. Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge [15].

2.4.2 The main aims of the Canny Edge Detector are as follows

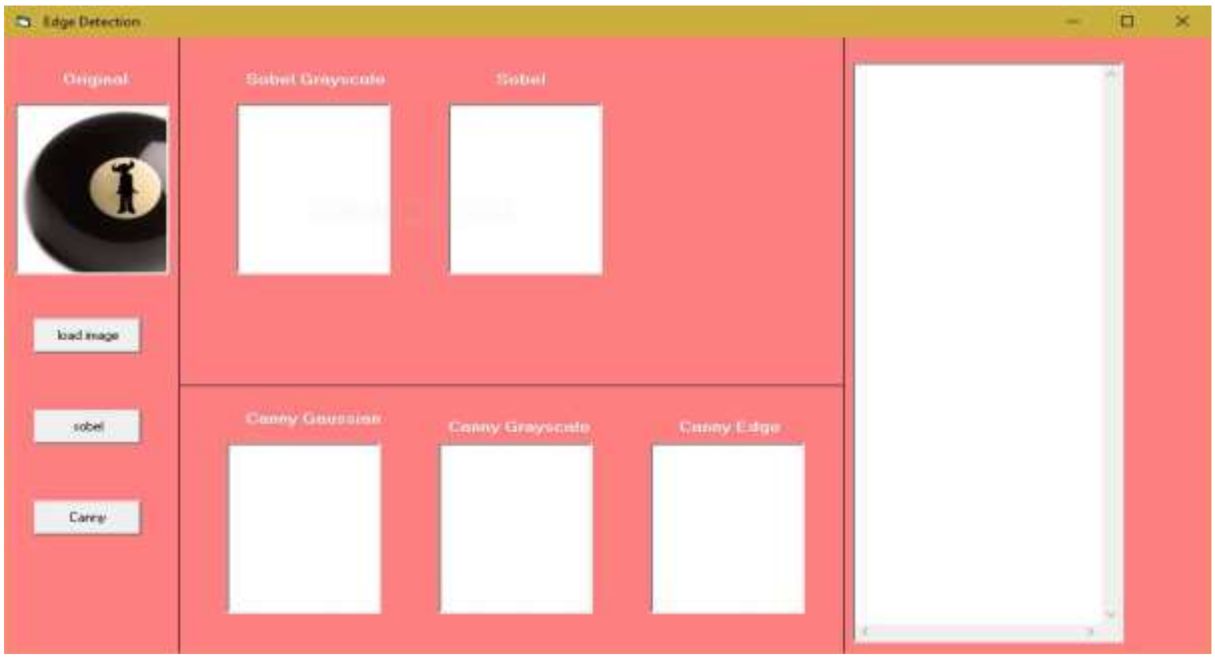
(a) Good detection - There should be a low probability of failing to mark real edge points, and low probability of falsely marking non edge points. Since both these probabilities are monotonically decreasing functions of the output signal-to-noise ratio, this criterion corresponds to maximizing signal-to-noise ratio. So basically, we need to mark as many real edges as possible.

(b) Good localization - The points marked out as edge points by the operator should be as close as possible to the center of the true edge [16]. In essence, the marked out edges should be as close to the edges in the real edges as possible.

(c) Minimal response - Only one response to a certain edge. This is implicitly captured in the first criterion since when there are two responses to the same edge, one of them must be considered false. Therefore, the idea is that an edge should be marked only once, and image noise should not create false edges.

3. PROPOSED SYSTEM

In this proposed system, we will implement techniques of edge detection on image we used to edge detection technique (Sobel & Canny).



This paper is consisting of three Command and six picture box and one text box. The Commands is for (Load image, Sobel edge detection & Canny edge detection).

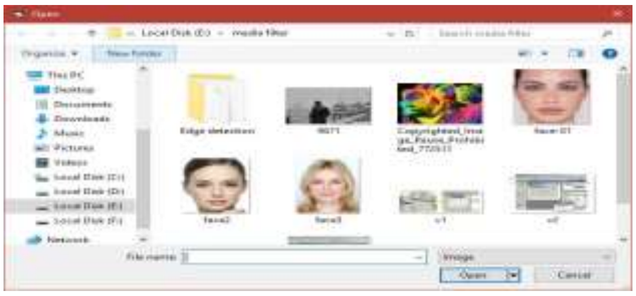
3.1 Load Image

To load image from computer to picturebox we write code in command2 the code is as follow :

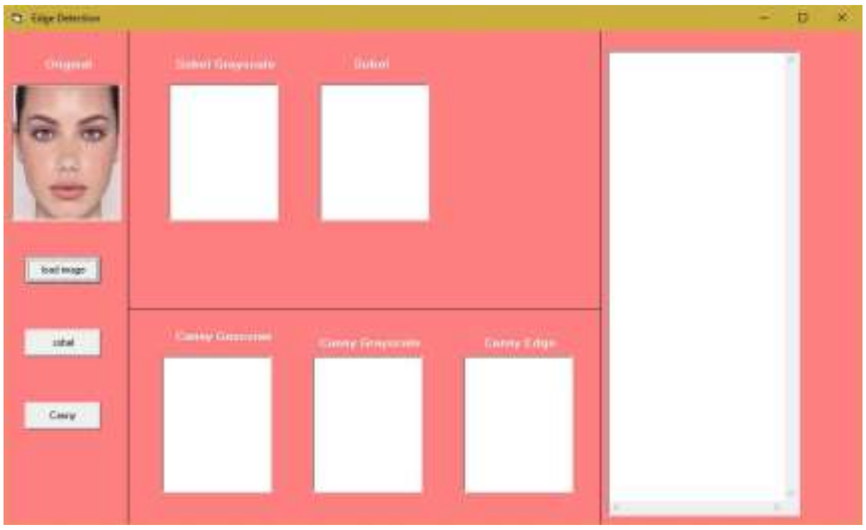
```
Private Sub Command2_Click()
Dim Token As Long
CM.Filter = "Image|*.bmp;*.jpg"
CM.ShowOpen
If CM.FileName <> "" Then
Token = InitGDIPlus
Pic1.Picture = LoadPictureGDIPlus(CM.FileName, Pic1.Width, Pic1.Height, , False)
FreeGDIPlus Token
End If
Text1.Text"" =
End Sub
```

When we Click on Command2 (Load Image) the we show the browser that ask to choose image in format (bmp & jpg) from the computer as shown in the follow figure:

DOI : 10.31695/IJERAT.2021.3736



When we choose an image the image is loaded into picturebox (pic1) as sown in the follow figure:



3.2 Sobel Edge Detection

To implement the sobel edge detection on image that loaded in (pic1) we must convert image to grayscale in the first like in the follow code:

```
Private Sub Command1_Click()
Const MaxData = 255
Const DataGranularity = 1
Const Delta = DataGranularity / (2 * MaxData)
Dim Op_X(-1 To 1, -1 To 1) As Integer, Op_Y(-1 To 1, -1 To 1) As Integer
Op_X(-1, -1) = -1: Op_X(0, -1) = -2: Op_X(1, -1) = -1
Op_X(-1, 0) = 0: Op_X(0, 0) = 0: Op_X(1, 0) = 0
Op_X(-1, 1) = 1: Op_X(0, 1) = 2: Op_X(1, 1) = 1
Op_Y(-1, -1) = -1: Op_Y(0, -1) = 0: Op_Y(1, -1) = 1
Op_Y(-1, 0) = -2: Op_Y(0, 0) = 0: Op_Y(1, 0) = 2
Op_Y(-1, 1) = -1: Op_Y(0, 1) = 0: Op_Y(1, 1) = 1
Dim X As Integer, Y As Integer
Dim fBias As Integer, fScaleFactor As Single
Dim fRadius As Single, Sum As Single, RR2 As Single
Dim MaxGaussianSize As Integer, GaussianSize As Integer
Dim GaussianKernel() As Double, Kernel() As Single
Dim KernelSize As Long
Dim fKernel As Long, fWidth As Long, fHeight As Long, fCount As Long
Dim SF As Single, Rad As Single, W As Single, C As Single
Dim KWH As Long, KWL As Long, KHH As Long, KHL As Long
Pic11.Cls: Pic5.Cls
Dim Greycolor As Integer, PixNum As Integer
PixNum = InputBox("enter the value for gray between [0 - 255] :", "Input Grayscale")
PixNum = 90
```

```
If PixNum > 255 Then PixNum = 255
Text1.Text = Text1.Text & " Grayscale color value = " & PixNum & vbCrLf
Text1.SelStart = Len(Text1.Text)
DoEvents
For Y = 0 To Pic1.Height
   For X = 0 To Pic1.Width
      PixelValue = GetPixel(Pic1.hdc, X, Y)
      DecTORGB PixelValue, R, G, B
      Greycolor = Greyscale(PixelValue, PixNum)
      SetPixel Pic11.hdc, X, Y, RGB(Greycolor, Greycolor, Greycolor)
      Pic11.Refresh
   Next X
   Pic11.Refresh
Next Y
Pic11.Refresh
```

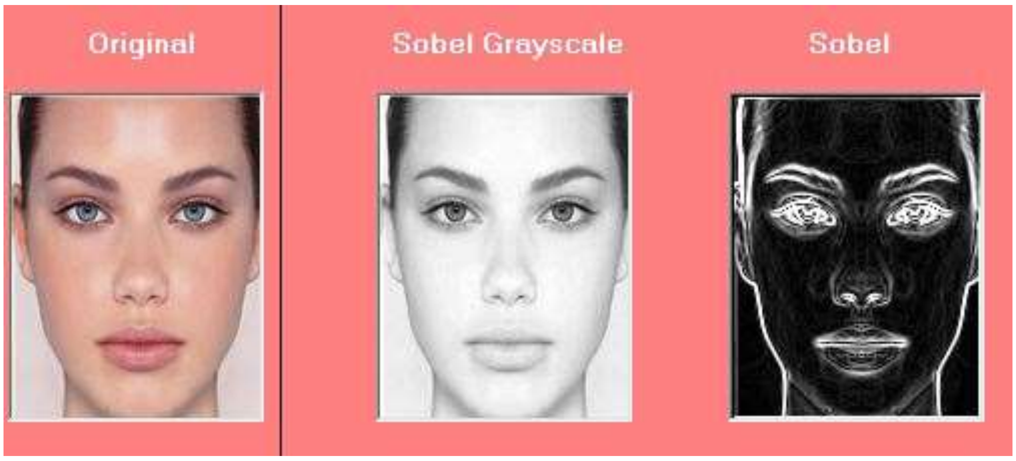
And when convert image to gray we show the result as shown in the follow figure:

And after convert image to gray scale we implement Sobel edge detection on image as the follow code:

```
Pic5.Cls
Grad = 0
Op_X(-1, -1) = -1: Op_X(0, -1) = -2: Op_X(1, -1) = -1
Op_X(-1, 0) = 0: Op_X(0, 0) = 0: Op_X(1, 0) = 0
Op_X(-1, 1) = 1: Op_X(0, 1) = 2: Op_X(1, 1) = 1
Op_Y(-1, -1) = -1: Op_Y(0, -1) = 0: Op_Y(1, -1) = 1
Op_Y(-1, 0) = -2: Op_Y(0, 0) = 0: Op_Y(1, 0) = 2
Op_Y(-1, 1) = -1: Op_Y(0, 1) = 0: Op_Y(1, 1) = 1
tmps = "using Kernel for Sobel" & vbCrLf & "Horisontal >>" & vbCrLf
For X = -1 To 1
   For Y = -1 To 1
      tmps = tmps & Format(Op_X(X, Y), "0")" " &
   Next Y
   tmps = tmps & vbCrLf
Next X
tmps = tmps & "Vertical >>" & vbCrLf
For X = -1 To 1
   For Y = -1 To 1
      tmps = tmps & Format(Op_Y(X, Y), "0")" " &
   Next Y
   tmps = tmps & vbCrLf
```

```
Next X
Text1.Text = Text1.Text & tmps & vbCrLf
DoEvents
For Y = 0 To Pic11.Height - 1
   For X = 0 To Pic11.Width - 1
      GradX = 0: GradY = 0: Grad = 0
      If X = 0 Or Y = 0 Or X = Pic11.Width - 1 Or Y = Pic11.Height - 1 Then
         Grad = 0
      Else
         For I = -1 To 1
            For J = -1 To 1
            PixelValue = GetPixel(Pic1.hdc, X + I, Y + J) '  x + i dan y + j
            DecTORGB PixelValue, R, G, B
            Itensity = (R + G + B) / 3 ' / B & W
            GradX = GradX + (Itensity * Op_X(I, J))
            GradY = GradY + (Itensity * Op_Y(I, J))
            Next J
         Next I
         Grad = Round(Sqr(Abs(GradX * GradX) + Abs(GradY * GradY)))
      End If
      If Grad <= 0 Then Grad = 0: If Grad >= 255 Then Grad = 255
      SetPixel Pic5.hdc, X, Y, RGB(Grad, Grad, Grad)
      Pic5.Refresh
   Next X
   Pic5.Refresh
Next Y
```

After implement of Sobel edge detection we show the result of sobel as shown in the follow figure:



3.3 Canny Edge Detection

To implement the Canny edge detection on image that loaded in (pic1) we must implement Gaussian filter in the first like in the follow code:

```
On Error GoTo ErrHandle
Const MaxData = 255
Const DataGranularity = 1
Const Delta = DataGranularity / (2 * MaxData)
Dim Op_X(-1 To 1, -1 To 1) As Integer, Op_Y(-1 To 1, -1 To 1) As Integer
Op_X(-1, -1) = -1: Op_X(0, -1) = -2: Op_X(1, -1) = -1
Op_X(-1, 0) = 0: Op_X(0, 0) = 0: Op_X(1, 0) = 0
Op_X(-1, 1) = 1: Op_X(0, 1) = 2: Op_X(1, 1) = 1
Op_Y(-1, -1) = -1: Op_Y(0, -1) = 0: Op_Y(1, -1) = 1
```

DOI : 10.31695/IJERAT.2021.3736

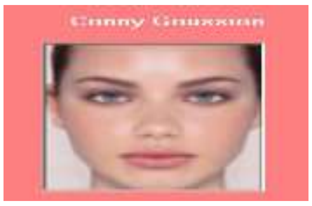
```
Op_Y(-1, 0) = -2: Op_Y(0, 0) = 0: Op_Y(1, 0) = 2
Op_Y(-1, 1) = -1: Op_Y(0, 1) = 0: Op_Y(1, 1) = 1

Dim X As Integer, Y As Integer
Dim fBias As Integer, fScaleFactor As Single
Dim fRadius As Single, Sum As Single, RR2 As Single
Dim MaxGaussianSize As Integer, GaussianSize As Integer
Dim GaussianKernel() As Double, Kernel() As Single
Dim KernelSize As Long
Dim fKernel As Long, fWidth As Long, fHeight As Long, fCount As Long
Dim SF As Single, Rad As Single, W As Single, C As Single
Dim KWH As Long, KWL As Long, KHH As Long, KHL As Long
Pic8.Cls: Pic9.Cls: Pic10.Cls
MaxGaussianSize = 50
fBias = 0: fScaleFactor = 1: fCount = 1
fRadius = InputBox("enter the number between [0.5 - 2] : ", "Gaussian Blur Radius")
If fRadius < 0 Then fRadius = 0
If fRadius > 10 Then MsgBox "enter the number between [0.5 - 2]", vbInformation, "excess numbers": Exit Sub
ReDim GaussianKernel(-MaxGaussianSize To MaxGaussianSize, -MaxGaussianSize To MaxGaussianSize)


Sum = 0
RR2 = -2 * fRadius * fRadius

For Y = -MaxGaussianSize To MaxGaussianSize
   For X = -MaxGaussianSize To MaxGaussianSize
      GaussianKernel(Y, X) = Exp((X * X + Y * Y) / RR2)
      Sum = Sum + GaussianKernel(Y, X)
   Next X
Next Y
For Y = -MaxGaussianSize To MaxGaussianSize
   For X = -MaxGaussianSize To MaxGaussianSize
      GaussianKernel(Y, X) = GaussianKernel(Y, X) / Sum
   Next X
Next Y
Sum = 0
GaussianSize = MaxGaussianSize
Do While (GaussianSize > 1) And (Sum < Delta)
   Sum = Sum + 4 * GaussianKernel(0, GaussianSize)
   GaussianSize = GaussianSize - 1
Loop
For Y = -GaussianSize To GaussianSize
   For X = -GaussianSize To GaussianSize
      Sum = Sum + GaussianKernel(Y, X)
   Next X
Next Y
For Y = -GaussianSize To GaussianSize
   For X = -GaussianSize To GaussianSize
      GaussianKernel(Y, X) = GaussianKernel(Y, X) / Sum
   Next X
Next Y
KernelSize = (2 * GaussianSize) + 1
SF = 0: Rad = 1
Dim RKernel(99999) As Single, cnt As Integer
cnt = 0
```

```
For Y = -GaussianSize To GaussianSize
    C = 0
    For X = -GaussianSize To GaussianSize
        W = Round((1 / GaussianKernel(GaussianSize, GaussianSize)) * GaussianKernel(Y, X))
        RKernel(cnt) = W
        SF = SF + W
        C = C + 1
        cnt = cnt + 1
    Next X
    Rad = Rad + 1
Next Y
fScaleFactor = SF

Me.Cls
ReDim Kernel(1, Rad, C)
cnt = 0
Dim tmps As String
Text1.Text = Text1.Text & vbCrLf
Text1.Text = Text1.Text & "looking for methods Canny kernel" & vbCrLf
Text1.Text = Text1.Text & "step 1 image conversion to gaussian blur ..." & vbCrLf
Text1.Text = Text1.Text + "the calculation result of gaussian blur kernel >>" & vbCrLf
Text1.SelStart = Len(Text1.Text)
ReDim Kernel(0 To 1, -GaussianSize To GaussianSize, -GaussianSize To GaussianSize)
For I = -GaussianSize To GaussianSize
    For J = -GaussianSize To GaussianSize
        Kernel(1, I, J) = RKernel(cnt)
        tmps = tmps & Format(Kernel(1, I, J), "000")" " &
        cnt = cnt + 1
    Next J
    Text1.Text = Text1.Text + tmps & vbCrLf
    tmps"" =
Next I
Text1.Text = Text1.Text + "use Theta = " & fRadius & " and size Kernel = " & KernelSize & " x " & KernelSize & " Pixel" &
vbCrLf
Text1.Text = Text1.Text + "using 2D drawing functions"
Text1.SelStart = Len(Text1.Text)
Dim tmpIntR As Double, tmpIntG As Double, tmpIntB As Double
Dim CTotal As Single
Dim CDataR As Single, CDataG As Single, CDataB As Single
Dim Intensity As Long, Grad As Single
CTotal = (((GaussianSize * GaussianSize) + GaussianSize) * 4) + 1
Dim tmpC As Long
Dim CountClr As Long
For Y = 0 To Pic1.Height - 1
    For X = 0 To Pic1.Width
        For J = -GaussianSize To GaussianSize
            For I = -GaussianSize To GaussianSize
                PixelValue = GetPixel(Pic1.hdc, X + I, Y + J)
                DecTORGB PixelValue, R, G, B
                CDataR = CDataR + (Kernel(1, J, I) * R)
                CDataG = CDataG + (Kernel(1, J, I) * G)
                CDataB = CDataB + (Kernel(1, J, I) * B)
                If mX < C Then mX = mX + 1
            Next I
        Next J
```

DOI : 10.31695/IJERAT.2021.3736

```
        CDataR = CDataR / (fScaleFactor + fBias)
        CDataG = CDataG / (fScaleFactor + fBias)
        CDataB = CDataB / (fScaleFactor + fBias)

        SetPixel Pic8.hdc, X, Y, RGB(CDataR, CDataG, CDataB)
        Pic8.Refresh
        DoEvents
    Next X
Next Y
Pic8.Refresh
Text1.Text = Text1.Text & vbCrLf & vbCrLf & "Langkah 2. Konversi gambar ke Grayscale..." & vbCrLf
Text1.SelStart = Len(Text1.Text)
```

After we implement of Gaussian filter on image we shown the result as sown in the follow figure:



In the step two we convert image to GrayScale as the follow code:

```
 Dim Greycolor As Integer, PixNum As Integer

'PixNum = InputBox("enter the value for gray between [0 - 255] :", "Input Grayscale")

PixNum = 90

If PixNum > 255 Then PixNum = 255

Text1.Text = Text1.Text & " Grayscale color value = " & PixNum & vbCrLf

Text1.SelStart = Len(Text1.Text)

DoEvents

For Y = 0 To Pic8.Height

    For X = 0 To Pic8.Width

        PixelValue = GetPixel(Pic8.hdc, X, Y)

        DecTORGB PixelValue, R, G, B

        Greycolor = Greyscale(PixelValue, PixNum)

        SetPixel Pic9.hdc, X, Y, RGB(Greycolor, Greycolor, Greycolor)

        Pic9.Refresh

    Next X

    Pic9.Refresh

Next Y
```

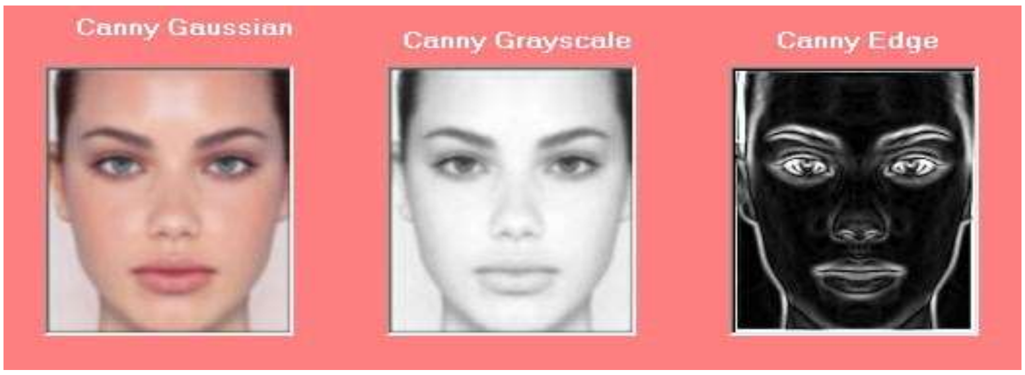
The result of implement gray scale image we shown in the follow figure:

In the step3: we implement Canny filter On Image in this Filtering image using 2 Thresholds.Min Threshold = 0 & Max Threshold = 1 as shown in the follow code:

```
Dim Itensity As Long, GradX As Long

Dim ThresholdMin As Integer, ThresholdMax As Integer
Dim sR As Long, sG As Long, sB As Long
Text1.Text = Text1.Text & vbCrLf & "step 3. Filtering image using 2 Thresholds..." & vbCrLf
Text1.SelStart = Len(Text1.Text)
ThresholdMin = InputBox("enter value Min Threshold :", " Min Threshold")
If ThresholdMin > 255 Then ThresholdMin = 255
Text1.Text = Text1.Text & "Min Threshold = " & ThresholdMin & vbCrLf
Text1.SelStart = Len(Text1.Text)
ThresholdMax = InputBox("enter value Max Threshold :", " Max Threshold")
If ThresholdMax > 255 Then ThresholdMax = 255
Text1.Text = Text1.Text & "Max Threshold = " & ThresholdMax & vbCrLf
Text1.SelStart = Len(Text1.Text)
'fScaleFactor = GaussianSize * GaussianSize
DoEvents
'ThresholdMin = 0
'ThresholdMax = 0
For Y = 0 To Pic1.Height - 1
   For X = 0 To Pic1.Width - 1
      GradX = 0: GradY = 0: Grad = 0
      If X = 0 Or Y = 0 Or X = Pic1.Width - 1 Or Y = Pic1.Height - 1 Then
         Grad = 0
      Else
         For I = -1 To 1
            For J = -1 To 1
            PixelValue = GetPixel(Pic8.hdc, X + I, Y + J) ' dapatkan pixel dari posisi x + i dan y + j
            DecTORGB PixelValue, R, G, B 'fungsi proses mendapatkan nilai RGB
            Itensity = (R + G + B) / 3 'Itensitas / B & W
            GradX = GradX + (Itensity * Op_X(I, J))
            GradY = GradY + (Itensity * Op_Y(I, J))
            Next J
         Next I
         Grad = Round(Sqr(Abs(GradX * GradX) + Abs(GradY * GradY)))
      End If
      If Grad >= ThresholdMin And Grad <= ThresholdMax Then
         Grad = Grad - Abs(((ThresholdMax + ThresholdMin) / KernelSize))
         If Grad <= ThresholdMax Then Grad = 0: If Grad >= ThresholdMax Then Grad = 255
         SetPixel Pic10.hdc, X, Y, RGB(Grad, Grad, Grad)
      Else
         If Grad <= 0 Then Grad = 0: If Grad >= 255 Then Grad = 255
         SetPixel Pic10.hdc, X, Y, RGB(Grad, Grad, Grad)
```

```
        End If
        Pic10.Refresh
    Next X
    Pic10.Refresh
Next Y
Exit Sub
ErrHandle:
```

The result of Canny filter was shown as the shown in the follow figure:



4. DATASET

The datasets that are used in proposed system are collected from 30 volunteers of different samples; 15 samples for each person, the images were taken under different lighting conditions, varied in expression, orientations, illumination, skin color, background, varied in ages, and face shapes (the mouth and eyes are open or closed, with or without glasses, male and female... etc.), the face is in frontal position.

The face images are 24 bit RGB, 300*400 pixels resolution of (BMP) format, the distance between the person and the camera is 125 cm.

DOI : 10.31695/IJERAT.2021.3736

Figure (1): Some Persons' Images with their Samples Chosen from the Dataset

5. CONCLUSION

In the discipline of computer vision, image processing is a quickly moving field. Its growth has been fueled by technological advances in digital imaging, computer processors and mass storage devices. In this paper an attempt is made to review the edge detection techniques which are based on discontinuity intensity levels. The relative performance of various edge detection techniques (sobel and canny) is carried out with an image by using visual Basic 6.0 software. Canny result is superior one when compared to all for a selected image since different edge detections work better under different conditions. In this paper we conclude that the results of canny are better than Sobel.

ACKNOWLEDGMENT

The authors would like to thank Mustansiriyah University (https://uomustansiriyah.edu.iq, Baghdad- Iraq) for its support in this work.

REFERENCES

[1] J. J. Benedetto and M.W. Frazier, "Wavelets Mathematics and Applications", CRC Press, Inc, 1994.

[2] K. A. Stevens, "Surface perception from local analysis of texture and contour",Artificial Intell. Lab., Mass. Instr. Technol., Cambridge, Tech. Rep. AI-TR512,1980.

[3]D. Yi, Z. Lei, and S. Z. Li, "Towards Pose Robust Face Recognition", provided by the Computer Vision Foundation the authoritative version of this paper is available in IEEE Xplore, 2013.

[4]K. Davakhar, S. B. Mule and A. Deshmukh, " Person Authentication Using Color Face Recognition", Journal of Engineering Research and Applications, Vol. 3, Issue 5, pp.178-182, Sep-Oct 2013.

[5]L. Xianwei and Z. Haiyang, "A Survey of Face Recognition Methods", Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), 2013.

[6]E. Al Daoud, "Enhancement of the Face Recognition Using a Modified Fourier-Gabor Filter", Computer Science Department Zarka Private University, Jordan, Vol. 1, No. 2, November 2009.

[7]International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 6, Dec 2011

[8]International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 6, Dec 201,PP 261.

[9]S. Rangarajan, "Algorithms For Edge Detection".

[10] P.Sharma,G. Singh and Amandeep Kaur , "Algorithms For Edge Canny Detection" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 3, Issue 3, pp.458-461, May-Jun 2013.

[11] R1. Muthukrishnan and M.Radha,"EDGE DETECTION TECHNIQUES FOR IMAGE SEGMENTATION " ,P.262, 2010.

[12] R.Gonzalez & R.Wood, "Digital Image Processing", 3rd ed, Englewood Cliffs, NJ: Prentice Hall, 2007.

[13] http://www.cse.usf.edu

[14] S. Bhardwaj and A. Mittal "A Survey on Various Edge Detector Techniques", Procedia Technology 4, 220 – 226, 2012.

[15] J. Arati Vyavahare, " Edge Detector Techniques" International Journal of Computer Applications (0975 – 8887) Volume 102– No.7, September 2014.

[16] D. Ray, "Edge Detection in Digital Image Processing ", Thursday, June 06, 2013.

C. Author: rania83computer@uomustansiriyah.edu.iq