

Lab File

Digital Image Processing

Semester VI

**BY: Sumit Shrivastava and Amit Kumar**

**Submitted to:** **Saurabh Shanu**

**Program:** B. Tech CS GG VI

# Lab Exercise 1

## Name

Image Acquisition

## Description:

Image acquisition is method of retrieving any image from a particular source that can be any hardware source for image processing. After this step, other processes can be applied on it. This image is initially unprocessed.

## Algorithm:

1. START
2. Create dialog box’s object.
3. If Dialog is open then
4. Create Picture box’s object.
5. If open.ShowDialog() == DialogResult.OK
6. picturebox.image = new bitmap(open. Filename)
7. Initialize ragb array.
8. EXIT

## Code Snippets:

open.Filter = "Image Files(\*.jpg; \*.jpeg; \*.gif; \*.bmp)|\*.jpg; \*.jpeg;

if (open.ShowDialog() == DialogResult.OK )

{

pictureBox1.Image = new Bitmap(open.FileName);

for (int i = 0; i < 3; i++)

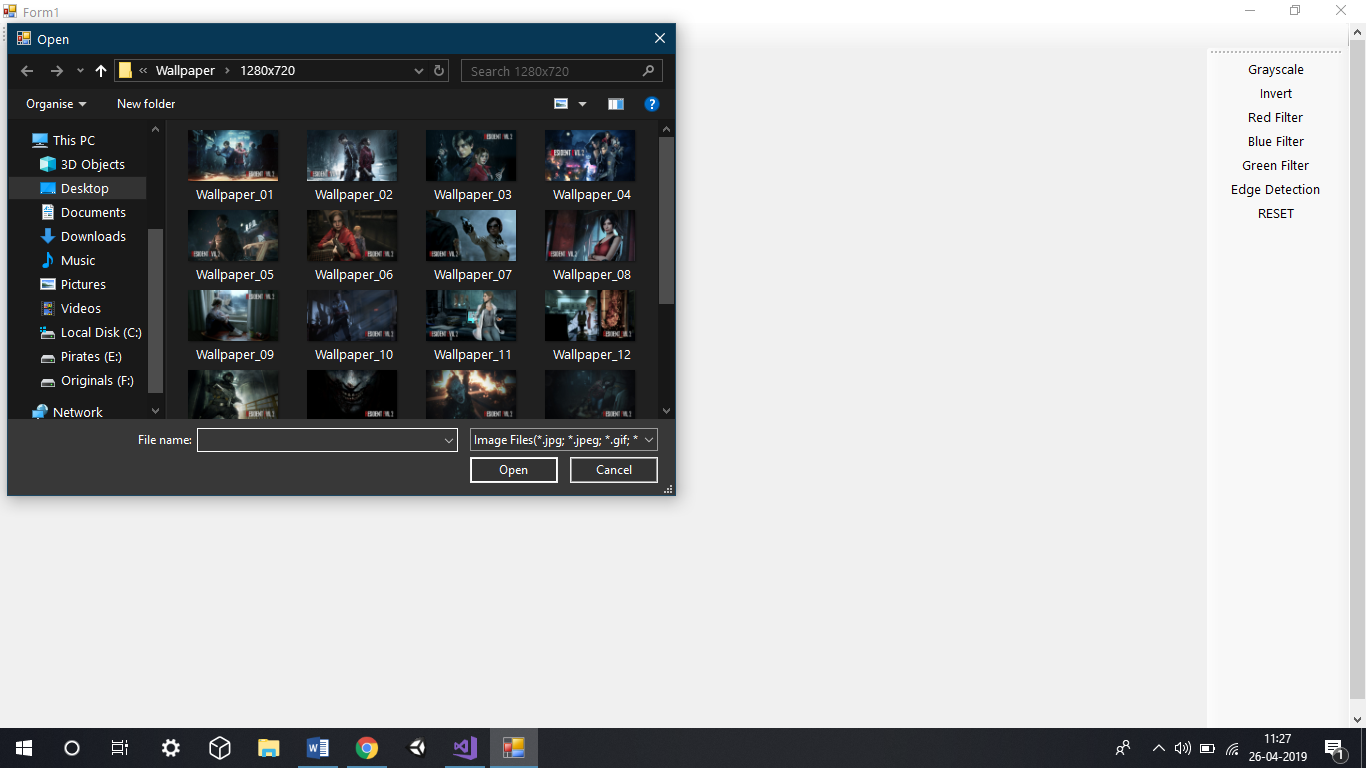
{

for (int j = 0; j < 256; j++)

{ rgb[i, j] = 0;

}

## Outputs:



# Lab Exercise 2

## Name

Edge Detection

## Description:

Sudden changes of discontinuities in an image are called as edges. In this technique different type of edges are detected like horizontal edges, vertical edges and diagonal edges.

Edges are detected using filters and then increase sharpness which makes the image cleaner.

Mask used is horizontal and vertical detection mask.

## Algorithm:

1. START
2. Initialize bitmap image height and width.
3. Horizontal and Vertical Convolution is applied.
4. For x<width
5. For y<height
6. Apply mask on image.
7. Initialize Result image with average of intensities.
8. For x<height-2
9. For x<width-2
10. Initialize convoluted matrix to the image matrix.
11. For x< width-2
12. For x< height -2
13. If convoluted matrix <0
14. Then apply matrix to bit map matrix 2
15. Else if convoluted matrix >255
16. Else
17. Then apply matrix to bit map matrix 2
18. For y<width-2
19. Initialize Result image with average of intensities.
20. EXIT

## Code Snippets:

int[,] convoluted = new int[height - 2, width - 2];//convoluted matrix

int[,] matrixlx = new int[,]

{

{1,1,1},

{1,-8,1},

{1,1,1}

};

int[,] image = new int[height, width];

Color p;

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

p = bmp2.GetPixel(x, y);

int a = p.A;

int b = p.B;

int r = p.R;

int g = p.G;

int avg = (r + g + b) / 3;

image[y,x] = avg;

}

}

for (int y = 0; y < height - 2; y++)

{

for (int x = 0; x < width - 2; x++)

{

convoluted[y, x] = (matrixlx[0, 0] \* image[y, x] + matrixlx[0, 1] \* image[y, x + 1] + matrixlx[0, 2] \* image[y, x + 2]) +

(matrixlx[1, 0] \* image[y + 1, x] + matrixlx[1, 1] \* image[y + 1, x + 1] + matrixlx[1, 2] \* image[y + 1, x + 2]) +

(matrixlx[2, 0] \* image[y + 2, x] + matrixlx[2, 1] \* image[y + 2, x + 1] + matrixlx[2, 2] \* image[y + 2, x + 2]);

}

}

for (int y = 0; y < height - 2; y++)

{

for (int x = 0; x < width - 2; x++)

{

int avg;

if (convoluted[y, x] < 0)

{

avg = 0;

bmp2.SetPixel(x, y, Color.FromArgb(avg, avg, avg));

}

else if (convoluted[y, x] > 255)

{

avg = 255;

bmp2.SetPixel(x, y, Color.FromArgb(avg, avg, avg));

}

else

{

avg = convoluted[y, x];

bmp2.SetPixel(x, y, Color.FromArgb(avg, avg, avg));

}

}

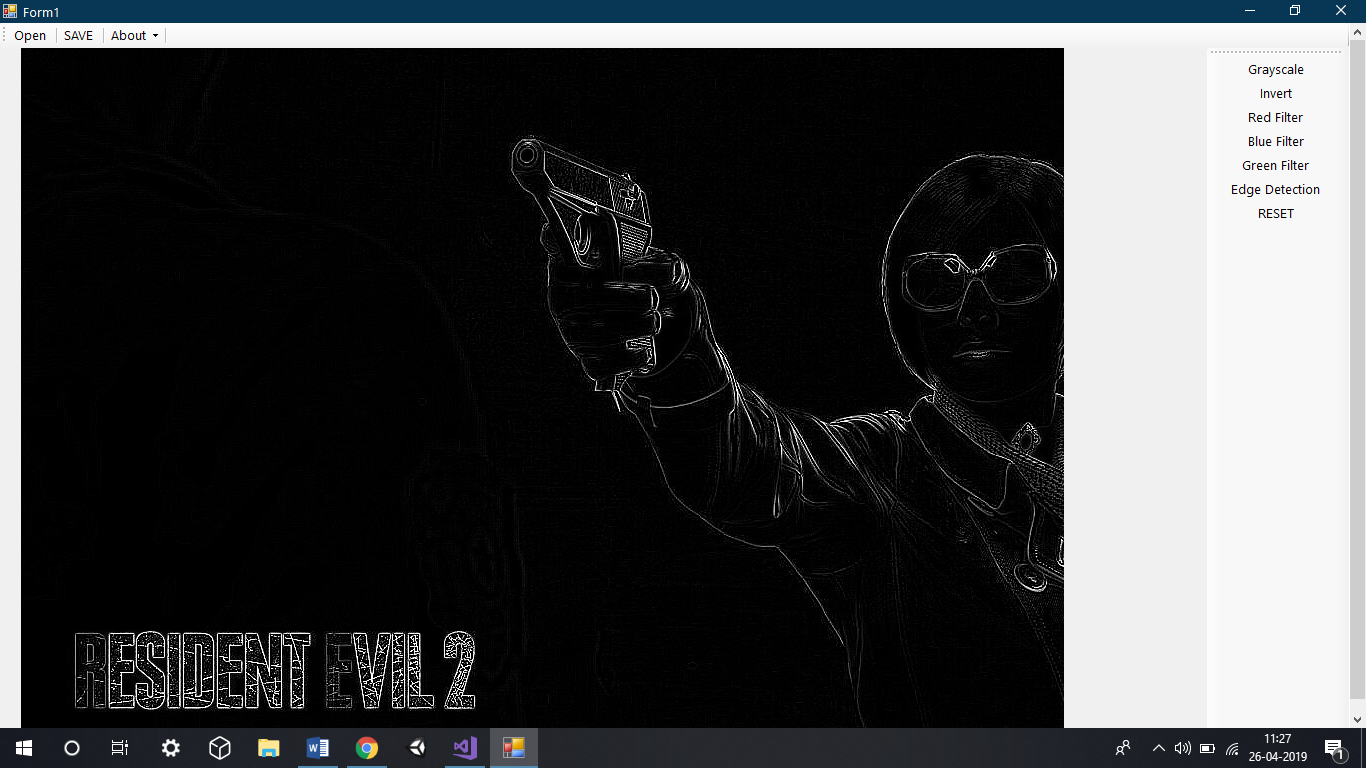
}

//load EdgeDected Image in picturebox2

pictureBox2.Image = bmp2;

}

## Outputs:



# Lab Exercise 3

## Name

Image Smoothening

## Description:

Sudden often used to reduce noise within an image or to produce a less pixelated image. Low pass filters are used to accomplish this task.

## Algorithm:

1. START
2. Initialize bitmap objects
3. Filter design: Butterworth / Bessel / Chebyshev
4. Filter type: Lowpass / Highpass / Bandpass / Bandstop
5. Filter order
6. Corner frequency/frequencie
7. EXIT

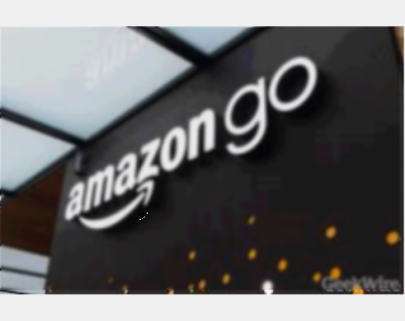
## Code Snippets:

## Output:

## Input Image



## Output Image



# Lab Exercise 4

## Name

Image Filter: RGB filter

## Description:

Sudden often used to reduce noise within an image or to produce a less pixelated image.Low pass filters are used to accomplish this task.

## Algorithm:

1. START
2. Get RGB Value at each pixel.
3. For y<height
4. For x<width
5. Initialize alpha.
6. Set bitmapimage for output image.
7. EXIT

## Code Snippets:

## private void redifyToolStripMenuItem\_Click(object sender, EventArgs e)

## {

## Bitmap bmp1 = new Bitmap(pictureBox1.Image);

## Bitmap bmp2 = new Bitmap(pictureBox1.Image);

## int width = bmp1.Width;

## int height = bmp1.Height;

## for (int y = 0; y < height; y++)

## {

## for (int x = 0; x < width; x++)

## {

## //extract pixel component ARGB

## Color p = bmp1.GetPixel(x, y);

## int a = p.A;

## int r = p.R;

## bmp2.SetPixel(x, y, Color.FromArgb(a, r, 0, 0));

## pictureBox2.Image = bmp2;

## }

## }

## }

## private void greenifyToolStripMenuItem\_Click(object sender, EventArgs e)

## {

## Bitmap bmp1 = new Bitmap(pictureBox1.Image);

## Bitmap bmp2 = new Bitmap(pictureBox1.Image);

## int width = bmp1.Width;

## int height = bmp1.Height;

## for (int y = 0; y < height; y++)

## {

## for (int x = 0; x < width; x++)

## {

## //extract pixel component ARGB

## Color p = bmp1.GetPixel(x, y);

## int a = p.A;

## int g = p.G;

## bmp2.SetPixel(x, y, Color.FromArgb(a, 0, g, 0));

## pictureBox2.Image = bmp2;

## }

## }

## }

## private void blueifyToolStripMenuItem\_Click(object sender, EventArgs e)

## {

## Bitmap bmp1 = new Bitmap(pictureBox1.Image);

## Bitmap bmp2 = new Bitmap(pictureBox1.Image);

## int width = bmp1.Width;

## int height = bmp1.Height;

## for (int y = 0; y < height; y++)

## {

## for (int x = 0; x < width; x++)

## {

## //extract pixel component ARGB

## Color p = bmp1.GetPixel(x, y);

## int a = p.A;

## int b = p.B;

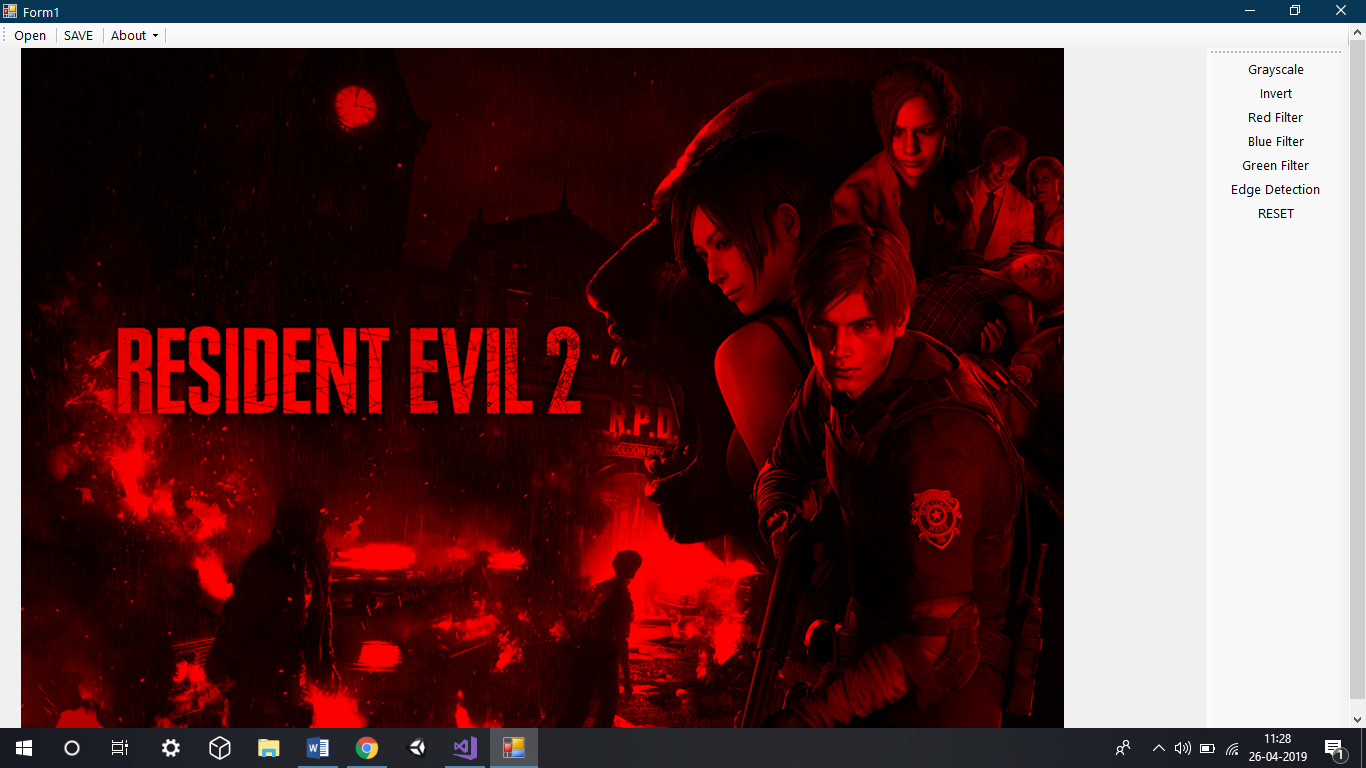
## bmp2.SetPixel(x, y, Color.FromArgb(a, 0, 0, b));

## pictureBox2.Image = bmp2;

## }

## }

## }Output:



# Lab Exercise 5

## Name

Image Compression

## Description:

Image compression is the process of encoding or converting an image file in such a way that it consumes less space than the original file. It is a type of compression technique that reduces the size of an image file without affecting or degrading its quality to a greater extent.

## Algorithm:

1. START
2. EXIT

## Code Snippets:

int counter = height \* width;

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

p = bmp1.GetPixel(x, y);

int a = p.A;

int b = p.B;

int r = p.R;

int g = p.G;

int avg = (r + g + b) / 3;

gray[avg] += 1;

}

}

for (int i = 0; i < 256; i++)

{

pmf[i] = gray[i] / counter;

}

for (int i = 0; i < 256; i++)

{

if (i == 0)

{

cdf[i] = pmf[i];

}

else

cdf[i] = pmf[i] + cdf[i - 1];

}

for (int i = 0; i < 256; i++)

{

cdf[i] = cdf[i] \* 255;

cdf[i] = (float)Math.Round(cdf[i]);

}

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

//extract pixel component ARGB

p = bmp1.GetPixel(x, y);

int a = p.A;

int b = (int)(cdf[p.B]);

int r = (int)(cdf[p.R]);

int g = (int)(cdf[p.G]);

//int avg = (r + g + b) / 3;

//avg = (int)(cdf[avg]);

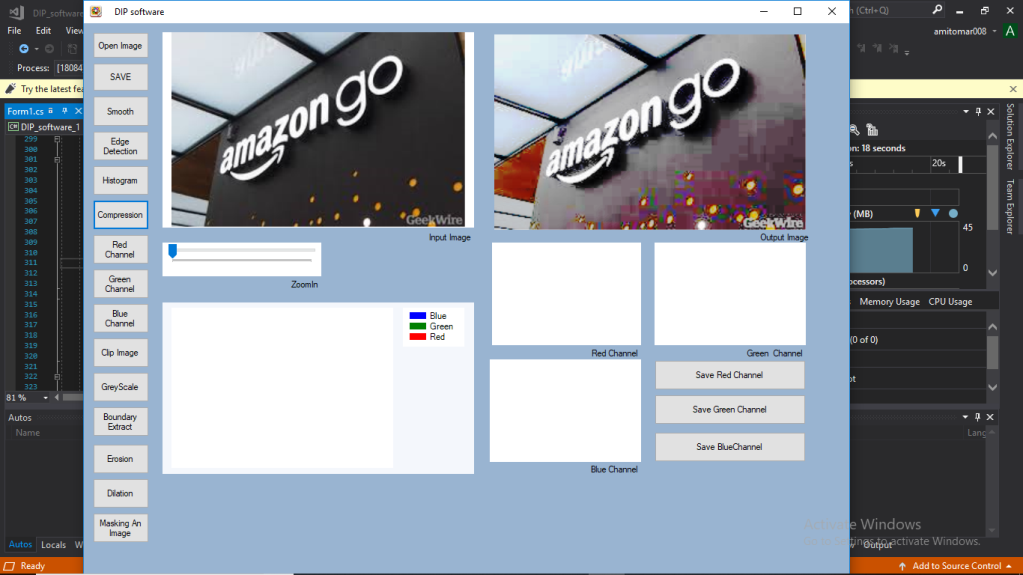
bmp2.SetPixel(x, y, Color.FromArgb(a, r, g, b));

//grayHist[avg] += 1;

}

}

## Output:



(Screenshot from Amit Kumar) (Build Crash on Mine)

# Lab Exercise 6

## Name

Clipping the image

## Description:

Clipping is a result of capturing or processing an image where the intensity in a certain area falls outside the minimum and maximum intensity which can be represented. It is an instance of signal clipping in the image domain.

## Algorithm:

1. START
2. Draw graphical shape on the image.
3. Transform origin of the image.
4. Combine image with shape.
5. Delete the image andreplace with white pixels.
6. EXIT

## Code Snippets:

g.TranslateTransform(bmp2.Width / 2, bmp2.Height / 2);

GraphicsPath path = new GraphicsPath();

path.AddEllipse(1000 - radius,1000- radius, radius/2, radius/2);

Region region = new Region(path);

g.SetClip(region, CombineMode.Replace);

g.DrawImage(bmp1, new Rectangle(5-radius,5-radius, 2\*radius,2\*radius),new Rectangle(x-radius,y-radius,2\*radius,2\*radius),GraphicsUnit.Pixel)

Output:

Error. Unable to clip properly

# Lab Exercise 7

## Name

Grayscale Image

## Description:

Grayscale is a range of monochromatic shades from black to white. Therefore, a grayscale image contains only shades of gray and no color.

## Algorithm:

1. START
2. For i<bmp.width
3. For j<bmp.height
4. Initialize array rgb[]
5. EXIT

## Code Snippets:

Bitmap bmp1 = new Bitmap(pictureBox1.Image);

Bitmap bmp2 = new Bitmap(pictureBox1.Image);

int width = bmp1.Width;

int height = bmp1.Height;

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

//extract pixel component ARGB

Color p = bmp1.GetPixel(x, y);

int a = p.A;

int b = p.B;

int r = p.R;

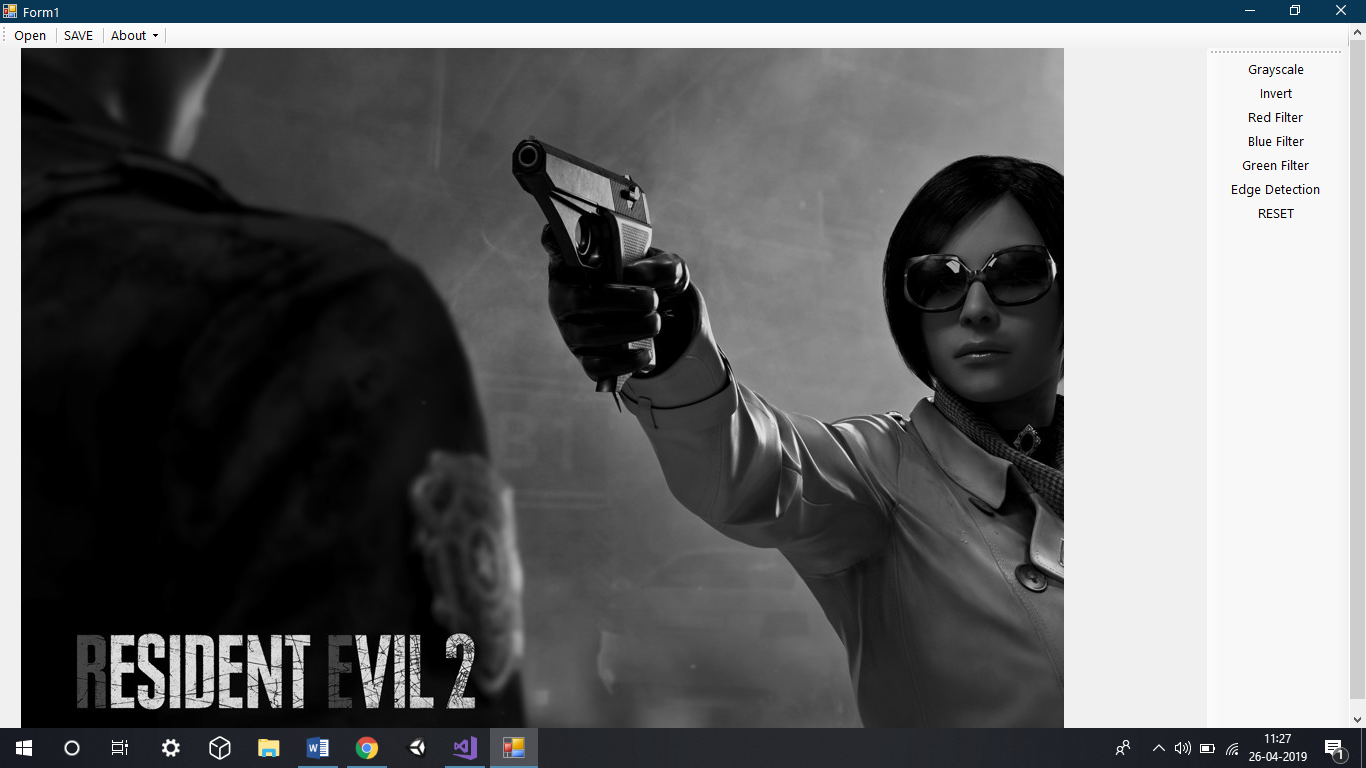
int g = p.G;

int avg = (r + g + b) / 3;

bmp2.SetPixel(x, y, Color.FromArgb(a, avg, avg, avg));

pictureBox2.Image = bmp2;

## Output:

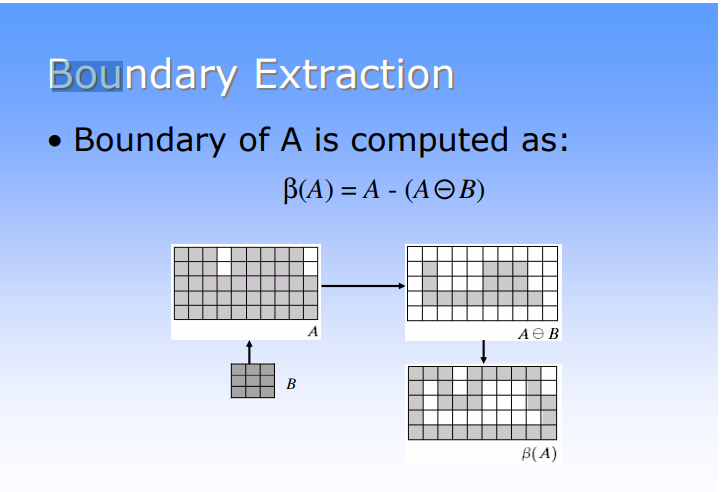


# Lab Exercise 8

## Name

Boundary Extraction

## Description:



## Algorithm:

1. START
2. Suppose that *X* is the set of Euclidean coordinates corresponding to the input binary image, and that *K* is the set of coordinates for the structuring element.
3. Let *Kx* denote the translation of *K* so that its origin is at *x*.
4. Then the dilation of *X* by *K* is simply the set of all points *x* such that the intersection of *Kx* with *X* is non-empty.
5. The mathematical definition of grayscale dilation is identical except for the way in which the set of coordinates associated with the input image is derived. In addition, these coordinates are 3-D rather than 2-D.
6. EXIT

## Code Snippets:

int RValue = 0;

int GValue = 0;

int BValue = 0;

for (int x2 = ApetureMin; x2 < ApetureMax; ++x2)

{

int TempX = x + x2;

if (TempX >= 0 && TempX < width)

{

for (int y2 = ApetureMin; y2 < ApetureMax; ++y2)

{

int TempY = y + y2;

if (TempY >= 0 && TempY < height)

{

Color TempColor = bmp1.GetPixel(TempX, TempY);

if (TempColor.R > RValue)

RValue = TempColor.R;

if (TempColor.G > GValue)

GValue = TempColor.G;

if (TempColor.B > BValue)

BValue = TempColor.B;

}

}

}

}

Color TempPixel = Color.FromArgb(RValue, GValue, BValue);

bmp2.SetPixel(x, y, TempPixel);

}

}

for (int x = 0; x < width; x++)

{

for (int y = 0; y < height; y++)

{

Color X = bmp1.GetPixel(x, y);

Color Y = bmp2.GetPixel(x, y);

int r = Y.R - X.R;

int g = Y.G - X.G;

int b = Y.B - X.B;

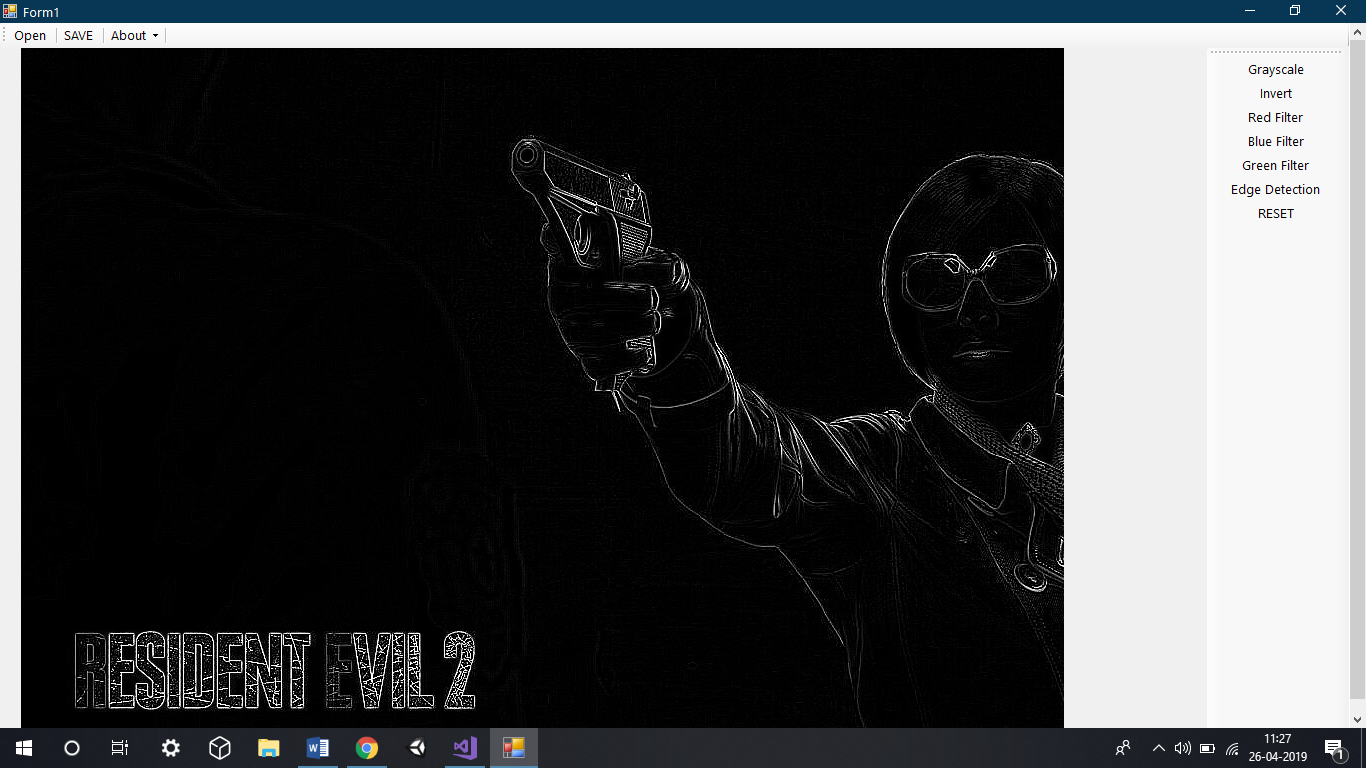
Color Pixel = Color.FromArgb(r, g, b);

bmp3.SetPixel(x, y, Pixel);

}

}

## Output:



# Lab Exercise 9

## Name

Erosion on digital image

## Description:

Erosion (usually represented by ⊖) is one of two fundamental operations (the other being dilation) in morphological image processing from which all other morphological operations are based.

## Algorithm:

1. START
2. For i<bmp.width
3. For j<bmp.height
4. Suppose that *X* is the set of Euclidean coordinates corresponding to the input binary image, and that *K* is the set of coordinates for the structuring element.
5. Let *Kx* denote the translation of *K* so that its origin is at *x*.
6. Then the dilation of *X* by *K* is simply the set of all points *x* such that the intersection of *Kx* with *X* is non-empty.
7. The mathematical definition of grayscale dilation is identical except for the way in which the set of coordinates associated with the input image is derived. In addition, these coordinates are 3-D rather than 2-D.
8. Initialize array rgb[]
9. EXIT

## Code Snippets:

int ApetureMax = (Size / 2);

for (int x = 0; x < width; ++x)

{

for (int y = 0; y < height; ++y)

{

int RValue = 255;

int GValue = 255;

int BValue = 255;

for (int x2 = ApetureMin; x2 < ApetureMax; ++x2)

{

int TempX = x + x2;

if (TempX >= 0 && TempX < width)

{

for (int y2 = ApetureMin; y2 < ApetureMax; ++y2)

{

int TempY = y + y2;

if (TempY >= 0 && TempY < height)

{

Color TempColor = bmp1.GetPixel(TempX, TempY);

if (TempColor.R < RValue)

RValue = TempColor.R;

if (TempColor.G < GValue)

GValue = TempColor.G;

if (TempColor.B < BValue)

BValue = TempColor.B;

}

# Lab Exercise 10

Inversion

Image inversion is the process of changing the colors to the opposite color, on the rgb scale.

Algorithm:

1. START
2. Get RGB Value at each pixel.
3. For y<height
4. For x<width
5. Initialize alpha.
6. Set bitmapimage for output image.
7. EXIT

Code Snippets:

Bitmap bmp1 = new Bitmap(pictureBox1.Image);

Bitmap bmp2 = new Bitmap(pictureBox1.Image);

int width = bmp1.Width;

int height = bmp1.Height;

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

Color p = bmp1.GetPixel(x, y);

int a = p.A;

int b = 255 - p.B;

int r = 255 - p.R;

int g = 255 - p.G;

int avg = (r + g + b) / 3;

bmp2.SetPixel(x, y, Color.FromArgb(a, r, g, b));

pictureBox1.Image = bmp2;

}

}

}

Output:

