**CPEG 585 – COMPUTER VISION**

**HOMEWORK 1**

**Name:** Ivan Sangines Escrig

**ID#:** 968606

**Instructor:** Dr. Mahmood

**Date:** February 10, 2019

**TABLE OF CONTENT**

[**INTRODUCTION**](#_Toc446970371) 3

[**SCREENSHOTS:**](#_Toc446970373) 4

[**SOURCE CODE:**](#_Toc446970374) 7

**[CONCLUSION:](#_Toc446970375)**25

**INTRODUCTION**

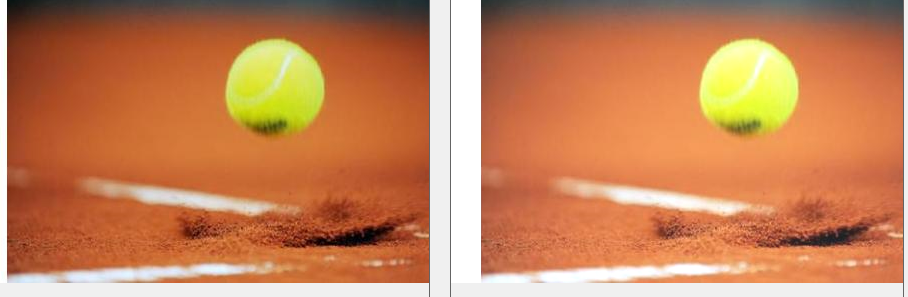
The porpuse of this assignment is to practice some image procesing techniques showed in the first handout. The first part of the assignment consists in going over the provided handout and try the examples such as convert to gray, resize and contrast.

The second part of the assignment is going to help us understand better how the Histogram Equalization works and how to implement it to an image with grey scale pixels.

**SCREEN SHOTS:**

**Problem 1:**

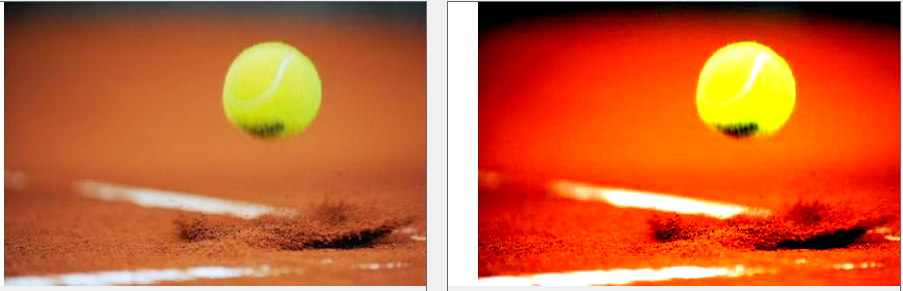
Brightening an image:

****

Converting an image to a gray scale:



Contrasting an image by a gradient of 50

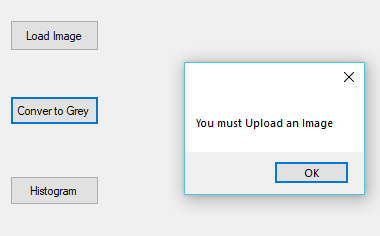
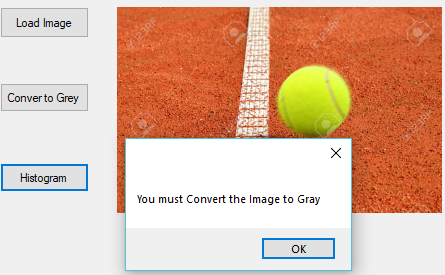
****

**Problem 2:**

On the screenshots we can see how the Histogram Equalitzation works and the results of applying it to an image. In order for it to work, we first need to convert the original picture uploaded to a Gray scale pixels, then we can apply the Histogram Equalitzation and see the eefects of it.

In order to create the Histogram array, I initialy created a 255x3 array where each index represents the pixel intensity. Once the image is converted to gray, I go over each pixel and increment the index with the pixels corresponding from the image. After doing the pixel count, the second column corresponds to the cdf value (skipping the 0 value pixels). The last column of the array corresponds to the h(v) column based on the cdf.

Trying to process image before uploading one or converting it to gray:

Applying Histogram Equalization to the original image:



**SOURCE CODE:**

**Problem 1:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Drawing.Imaging;

namespace ImageProcesing2010

{

public partial class Form1 : Form

{

private Point startPoint, endPoint, startRPoint;

bool modeRectangle = false;

public Form1()

{

InitializeComponent();

}

private void btnLoadImage\_Click(object sender, EventArgs e)

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "jpeg files (\*.jpg)|\*.jpg|(\*.gif)|gif||";

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

{

this.picOrigImage.Image = new Bitmap(dialog.FileName);

FileInfo finfo = new FileInfo(dialog.FileName);

lblImageUnderTest.Text = "Image Under Test\n " +

finfo.Name;

txtWidthOrig.Text = picOrigImage.Image.Width.ToString();

txtHeightOrig.Text = picOrigImage.Image.Height.ToString();

}

}

private void btnConvertToGray\_Click(object sender, EventArgs e)

{

try

{

Bitmap copy = new Bitmap((Bitmap)this.picOrigImage.Image);

MyImageProc.CovertToGray(copy);

picProcImage.Image = null;

picProcImage.Image = copy;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnRotateFillImage\_Click(object sender, EventArgs e)

{

try

{

FormRotate dlg = new FormRotate();

dlg.RotationAngle = 0;

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

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

{

if (MyImageProc.RotateFill(origimg, ref bmp, dlg.RotationAngle))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnCropAroundMouseClick\_Click(object sender, EventArgs e)

{

try

{

Image img = this.picOrigImage.Image;

Bitmap bmp = new Bitmap(img);

Point p1 = new Point(startPoint.X - int.Parse(txtCropAroundW.Text) / 2, startPoint.Y - int.Parse(txtCropAroundH.Text) / 2);

Rectangle rect = new Rectangle(p1, new Size(int.Parse(txtCropAroundW.Text),int.Parse(txtCropAroundH.Text)));

bmp = new Bitmap(img);

MyImageProc.CropImage(img, ref bmp, rect);

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void picOrigImage\_MouseDown(object sender, MouseEventArgs e)

{

modeRectangle = true;

startPoint = new Point(e.X, e.Y);

}

private void picOrigImage\_MouseUp(object sender, MouseEventArgs e)

{

endPoint = new Point(e.X, e.Y);

}

private void Form1\_Load(object sender, EventArgs e)

{

txtCropAroundW.Text = "14";

txtCropAroundH.Text = "14"; // 14 x14 window to crop around

}

private void btnBrightenImage\_Click(object sender, EventArgs e)

{

try

{

FormBrightness dlg = new FormBrightness();

dlg.Brightness = 30; // default brightness of 30

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

{

Bitmap copy = new Bitmap((Bitmap)this.picOrigImage.Image);

copy = MyImageProc.Brighten(copy, dlg.Brightness);

picProcImage.Image = null;

picProcImage.Image = copy;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnChangeContrast\_Click(object sender, EventArgs e)

{

try

{

FormContrast dlg = new FormContrast();

dlg.Contrast = 0;

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

{

Bitmap copy = new Bitmap((Bitmap)this.picOrigImage.Image);

copy = MyImageProc.Contrast(copy, (sbyte)dlg.Contrast);

picProcImage.Image = null;

picProcImage.Image = copy;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnSaveProcessedImage\_Click(object sender, EventArgs e)

{

SaveFileDialog dlg = new SaveFileDialog();

dlg.Filter = "jpeg files (\*.jpg)|\*.jpg";

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

this.picProcImage.Image.Save(dlg.FileName, ImageFormat.Jpeg);

}

private void btnResizeImage\_Click(object sender, EventArgs e)

{

if (modeRectangle == true)

{

modeRectangle = false;

Rectangle rect = new Rectangle(new Point(0, 0), new Size(endPoint.X, endPoint.Y));

Image img = this.picOrigImage.Image;

Bitmap bmp = new Bitmap(img);

MyImageProc.ResizeImage(img, ref bmp, rect);

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

else

MessageBox.Show("you must click mouse to select bottom of rectangle");

}

private void btnResizeProportional\_Click(object sender, EventArgs e)

{

try

{

modeRectangle = false;

int Width = int.Parse(txtNewSizeW.Text);

int Height = int.Parse(txtNewSizeH.Text);

Rectangle rect = new Rectangle(new Point(0, 0), new Size(Width, Height));

Image img = this.picOrigImage.Image;

Bitmap bmp = new Bitmap(img);

MyImageProc.ResizeImageProportional(img, ref bmp, rect);

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnCropAroundDragAndReleaseMouse\_Click(object sender, EventArgs e)

{

try

{

if (modeRectangle == true)

{

modeRectangle = false;

Rectangle rect = new Rectangle(startPoint, new Size(endPoint.X - startPoint.X, endPoint.Y - startPoint.Y));

Image img = this.picOrigImage.Image;

Bitmap bmp = new Bitmap(img);

MyImageProc.CropImage(img, ref bmp, rect);

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

else

MessageBox.Show("you must click mouse to select bottom of rectangle");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void btnDraRectangle\_Click(object sender, EventArgs e)

{

if (modeRectangle == true)

{

modeRectangle = false;

Rectangle rect = new Rectangle(startPoint, new Size(endPoint.X - startPoint.X, endPoint.Y - startPoint.Y));

Image img = this.picOrigImage.Image;

Bitmap bmp = new Bitmap(img);

MyImageProc.DrawRectangle(img, ref bmp, rect);

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

else

MessageBox.Show("you must drag mouse to select rectangle");

}

private void btnRotateClear\_Click(object sender, EventArgs e)

{

FormRotate frmRotate = new FormRotate();

frmRotate.RotationAngle = 0;

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

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

{

if (MyImageProc.RotateClear(origimg, ref bmp, frmRotate.RotationAngle))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

}

private void btnDrawX\_Click(object sender, EventArgs e)

{

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

if (MyImageProc.DrawX(origimg, ref bmp, startPoint))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

}

}

private void btnRotateByPoints\_Click(object sender, EventArgs e)

{

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

if (modeRectangle == true)

{

if (MyImageProc.RotateByPoints(origimg, ref bmp, startPoint, endPoint))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

else

MessageBox.Show("you must click mouse to select bottom of rectangle");

}

private void btnRotateFill\_Click(object sender, EventArgs e)

{

FormRotate dlg = new FormRotate();

dlg.RotationAngle = 0;

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

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

{

if (MyImageProc.RotateFill(origimg, ref bmp, dlg.RotationAngle))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

}

private void btnShiftHorizontal\_Click(object sender, EventArgs e)

{

FormShift dlg = new FormShift();

dlg.ShiftAmount = 0;

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

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

{

if (MyImageProc.ShiftImageHorizontal(origimg, ref bmp, dlg.ShiftAmount))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

}

private void btnShiftVertical\_Click(object sender, EventArgs e)

{

FormShift dlg = new FormShift();

dlg.ShiftAmount = 0;

Image origimg = picOrigImage.Image;

Bitmap bmp = new Bitmap(origimg);

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

{

if (MyImageProc.ShiftImageVertical(origimg, ref bmp, dlg.ShiftAmount))

{

picProcImage.Image = null;

picProcImage.Image = bmp;

txtWidthProc.Text = picProcImage.Image.Width.ToString();

txtHeightProc.Text = picProcImage.Image.Height.ToString();

}

}

}

}

}

IMAGE PROCESSING:

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;

using System.Windows.Forms;

namespace ImageProcesing2010

{

/// <summary>

/// Summary description for MyImageProc.

/// </summary>

public class MyImageProc

{

public MyImageProc()

{

//

// TODO: Add constructor logic here

//

}

public static bool CovertToGray(Bitmap b)

{

for (int i = 0; i < b.Width; i++)

for (int j = 0; j < b.Height; j++)

{

Color c1 = b.GetPixel(i,j);

int r1 = c1.R;

int g1 = c1.G;

int b1 = c1.B;

int gray = (byte)(.299 \* r1

+ .587 \* g1

+ .114 \* b1);

r1 = gray;

g1 = gray;

b1 = gray;

b.SetPixel(i,j,Color.FromArgb(r1,g1,b1));

}

return true;

}

public static Bitmap Brighten(Bitmap use, int nBrightness)

{

int Red,Green,Blue;

if (nBrightness < -255 || nBrightness > 255) //pixels go from 0 to 255

return use;

Bitmap Bright= new Bitmap(use.Width,use.Height);

for(int r=0;r<use.Height;r++)

{

for(int c=0;c<use.Width;c++)

{

Color cr = use.GetPixel(c,r);

Red=nBrightness+Convert.ToInt32(cr.R);

Green=nBrightness+Convert.ToInt32(cr.G);

Blue=nBrightness+Convert.ToInt32(cr.B);

//checking pixels are between 0 and 255

if (Red>255)

Red =255;

if(Red<0)

Red =0;

if(Green>255)

Green =255;

if(Green<0)

Green =0;

if(Blue>255)

Blue =255;

if(Blue<0)

Blue =0;

Bright.SetPixel(c,r,Color.FromArgb(Red,Green,Blue));

}

}

return Bright;

}

public static Bitmap Contrast(Bitmap b, sbyte nContrast)

{

Bitmap Contrast= new Bitmap(b.Width,b.Height);

if (nContrast < -100)

nContrast =-100;

if (nContrast > 100)

nContrast =100;

double pixel = 0, contrast = (100.0+nContrast)/100.0;

contrast \*= contrast;

int red, green, blue;

for(int y=0;y<b.Height;++y)

{

for(int x=0; x < b.Width; ++x )

{

Color cr = b.GetPixel(x,y);

red=Convert.ToInt32(cr.R);

green=Convert.ToInt32(cr.G);

blue=Convert.ToInt32(cr.B);

pixel = red/255.0;

pixel -= 0.5;

pixel \*= contrast;

pixel += 0.5;

pixel \*= 255;

if (pixel < 0) pixel = 0;

if (pixel > 255) pixel = 255;

red=(int)pixel;

pixel = green/255.0;

pixel -= 0.5;

pixel \*= contrast;

pixel += 0.5;

pixel \*= 255;

if (pixel < 0) pixel = 0;

if (pixel > 255) pixel = 255;

green=(int)pixel;

pixel = blue/255.0;

pixel -= 0.5;

pixel \*= contrast;

pixel += 0.5;

pixel \*= 255;

if (pixel < 0) pixel = 0;

if (pixel > 255) pixel = 255;

blue=(int)pixel;

Contrast.SetPixel(x,y,Color.FromArgb(red,green,blue));

}

}

return Contrast;

}

public static Bitmap RecoverBitmapFromArray(double [] BitmapArr, int w1, int h1)

{

Bitmap brecov = new Bitmap(w1,h1);

int p=0;

int q = 0;

for (int i = 0; i < BitmapArr.Length;i++)

{

int val = (int) (BitmapArr[i]/(1375669f\*900f));

if (val < 0)

val = 0;

Color c1 = Color.FromArgb(val,val,val);

brecov.SetPixel(p,q,c1);

p++;

if (p == w1)

{

p = 0;

q++;

}

}

return brecov;

}

public static bool RotateClear(Image img, ref Bitmap bm, double Rot)

{

if (Rot < -360 || Rot > 360)

return false;

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.Clear(Color.White);

dc.RotateTransform((float) Rot);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

return true;

}

public static bool RotateByPoints(Image img, ref Bitmap bm, Point p1, Point p2)

{

if ((p1.X == p2.X)) // no need to rotate

return true;

Point midPt = new Point();

midPt.X = (int)((p1.X+p2.X)/2.0);

midPt.Y = (int)((p1.Y+p2.Y)/2.0);

double RotRadians = Math.Atan2(-(p2.Y - p1.Y),(p2.X - p1.X));

Point newmidPt = new Point();

newmidPt.X = (int) (midPt.X \* Math.Cos(RotRadians) -

midPt.Y \* Math.Sin(RotRadians));

newmidPt.Y = (int) (midPt.Y \* Math.Cos(RotRadians) +

midPt.X \* Math.Sin(RotRadians));

double Rot = RotRadians \* 180/3.141516;

return MyImageProc.RotateFill2(img, ref bm,Rot,midPt,newmidPt);

}

public static bool RotateFill(Image img, ref Bitmap bm, double Rot)

{

if (Rot < -360 || Rot > 360)

return false;

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

dc.RotateTransform((float)Rot);

double rdegrees = Rot \* 3.141516/180;

int shift = (int)(bm.Height/2 \* Math.Tan(rdegrees));

if (rdegrees > 0)

//dc.DrawImage(img,new Rectangle((int)shift-(int)(1.2\*Rot),(int)-shift,img.Width,img.Height));

dc.DrawImage(img,new Rectangle((int)shift,(int)-shift,img.Width,img.Height));

else

dc.DrawImage(img,new Rectangle((int)shift,(int)-shift/2,img.Width,img.Height));

return true;

}

public static bool RotateFill2(Image img, ref Bitmap bm, double Rot, Point mpt, Point newmpt)

{

if (Rot < -360 || Rot > 360)

return false;

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

dc.RotateTransform((float)Rot);

double rdegrees = Rot \* 3.141516/180;

int shift = (int)(bm.Height/2 \* Math.Tan(rdegrees));

if (rdegrees > 0)

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

else

dc.DrawImage(img,new Rectangle(new Point(0,0),new Size(img.Width,img.Height)));

return true;

}

public static bool DrawRectangle(Image img, ref Bitmap bm, Rectangle rect)

{

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

dc.DrawRectangle(new Pen(Color.Red),rect);

return true;

}

public static bool DrawX(Image img, ref Bitmap bm, Point pt)

{

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

Brush br = new SolidBrush(Color.Red);

Point p1 = new Point(pt.X-3,pt.Y-3);

Point p2 = new Point(pt.X+3,pt.Y+3);

dc.DrawLine(new Pen(Color.Blue),p1,p2);

Point p3 = new Point(pt.X+3,pt.Y-3);

Point p4 = new Point(pt.X-3,pt.Y+3);

dc.DrawLine(new Pen(Color.Blue),p3,p4);

return true;

}

public static bool Draw2X(Image img, ref Bitmap bm, Point pt1, Point pt2)

{

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

Brush br = new SolidBrush(Color.Red);

Point p1a = new Point(pt1.X-3,pt1.Y-3);

Point p2a = new Point(pt1.X+3,pt1.Y+3);

dc.DrawLine(new Pen(Color.Red),p1a,p2a);

Point p3a = new Point(pt1.X+3,pt1.Y-3);

Point p4a = new Point(pt1.X-3,pt1.Y+3);

dc.DrawLine(new Pen(Color.Red),p3a,p4a);

Point p1b = new Point(pt2.X-3,pt2.Y-3);

Point p2b = new Point(pt2.X+3,pt2.Y+3);

dc.DrawLine(new Pen(Color.Red),p1b,p2b);

Point p3b = new Point(pt2.X+3,pt2.Y-3);

Point p4b = new Point(pt2.X-3,pt2.Y+3);

dc.DrawLine(new Pen(Color.Red),p3b,p4b);

return true;

}

public static bool Draw2Xbold(Image img, ref Bitmap bm, Point pt1, Point pt2)

{

bm = new Bitmap(img.Width,img.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,new Rectangle(0,0,img.Width,img.Height));

Brush br = new SolidBrush(Color.Red);

Point p1a = new Point(pt1.X-6,pt1.Y-6);

Point p2a = new Point(pt1.X+6,pt1.Y+6);

dc.DrawLine(new Pen(Color.Red,2),p1a,p2a);

Point p3a = new Point(pt1.X+6,pt1.Y-6);

Point p4a = new Point(pt1.X-6,pt1.Y+6);

dc.DrawLine(new Pen(Color.Red,2),p3a,p4a);

Point p1b = new Point(pt2.X-6,pt2.Y-6);

Point p2b = new Point(pt2.X+6,pt2.Y+6);

dc.DrawLine(new Pen(Color.Red,2),p1b,p2b);

Point p3b = new Point(pt2.X+6,pt2.Y-6);

Point p4b = new Point(pt2.X-6,pt2.Y+6);

dc.DrawLine(new Pen(Color.Red,2),p3b,p4b);

return true;

}

public static bool ResizeImage(Image img, ref Bitmap bm, Rectangle rect)

{

bm = new Bitmap(rect.Width,rect.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

dc.DrawImage(img,rect);

return true;

}

public static bool ResizeImageProportional(Image img, ref Bitmap bm, Rectangle rect)

{

Rectangle newR = new Rectangle(rect.X, rect.Y, rect.Width, img.Height\*rect.Width/img.Width);

bm = new Bitmap(newR.Width,newR.Height,PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bm);

//dc.InterpolationMode = InterpolationMode.High;

dc.DrawImage(img,newR);

return true;

}

public static bool CropImage(Image img, ref Bitmap bm, Rectangle rect)

{

Bitmap bmpCrop = new Bitmap (rect.Width, rect.Height,img.PixelFormat);

Graphics dc = Graphics.FromImage(bmpCrop);

Rectangle recDest = new Rectangle(0, 0, rect.Width,rect.Height);

dc.DrawImage(bm, recDest, rect.X, rect.Y, rect.Width, rect.Height, GraphicsUnit.Pixel);

bm = bmpCrop;

return true;

}

public static bool ShiftImageHorizontal(Image img, ref Bitmap bmp, int shiftAmt)

{

if (shiftAmt > img.Width)

return false;

bmp = new Bitmap(img.Width, img.Height, PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bmp);

dc.Clear(Color.Black);

dc.TranslateTransform(shiftAmt,0);

dc.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height));

return true;

}

public static bool ShiftImageVertical(Image img, ref Bitmap bmp, int shiftAmt)

{

if (shiftAmt > img.Width)

return false;

bmp = new Bitmap(img.Width, img.Height + Math.Abs(shiftAmt), PixelFormat.Format24bppRgb);

bmp = new Bitmap(img.Width, img.Height, PixelFormat.Format24bppRgb);

Graphics dc = Graphics.FromImage(bmp);

dc.Clear(Color.Black);

dc.TranslateTransform(0,shiftAmt);

dc.DrawImage(img, new Rectangle(0, 0, img.Width, img.Height));

return true;

}

}

}

**Problem 2:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace HistogramEqualization

{

public partial class Form1 : Form

{

int[,] Histogram;

bool isGray = false;

bool isImage = false;

Bitmap bm;

public Form1()

{

InitializeComponent();

}

//Loading Image

private void btnLoad\_Click(object sender, EventArgs e)

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "jpeg files (\*.jpg)|\*.jpg|(\*.gif)|gif||";

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

{

bm = new Bitmap(dialog.FileName);

while (bm.Height > picOriginal.Height || bm.Width > picOriginal.Width)

{

bm = new Bitmap(bm, new Size(bm.Width / 2, bm.Height / 2)); //if the image is greater than the picturebo make the picture smaller

}

picOriginal.Image = bm;

isImage = true;

isGray = false; //in case we are loading a new image

}

}

//Converting image to Gray

private void btnGrey\_Click(object sender, EventArgs e)

{

if (isImage)

{

try

{

bm = new Bitmap(picOriginal.Image);

isGray = MyImageProc.ConvertToGray(bm);

picOriginal.Image = bm;

isGray = true;

}

catch (Exception error)

{

MessageBox.Show(error.Message);

}

}

else

{

MessageBox.Show("You must Upload an Image");

}

}

private void btnHisto\_Click(object sender, EventArgs e)

{

if (isGray)

{

Bitmap modified = (Bitmap)bm.Clone();

MyImageProc.CreateHistogram(modified);

picModified.Image = modified;

}

else

{

MessageBox.Show("You must Convert the Image to Gray");

}

}

}

}

HISTOGRAM EQUALIZATION:

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace HistogramEqualization

{

class MyImageProc

{

public static int[,] Histogram = new int[256,3];

//Converting image to gray scale

public static bool ConvertToGray(Bitmap b)

{

for(int col = 0; col<b.Width; col++)

{

for(int row=0; row<b.Height; row++)

{

Color c = b.GetPixel(col, row);

int red = c.R;

int green = c.G;

int blue = c.B;

int gray = (byte)(.299 \* red + .557 \* green + .114 \* blue);

red = gray;

green = gray;

blue = gray;

b.SetPixel(col, row, Color.FromArgb(red, green, blue));

}

}

return true;

}

//

public static void CreateHistogram(Bitmap b)

{

//int[] allPixels = new int[b.Width \* b.Height];

//Setting array deffault values to 0

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

{

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

{

Histogram[row, col] = 0;

}

}

//Pixel Count

for (int i = 0; i < b.Height; i++)

{

for (int j = 0; j < b.Width; j++)

{

Color c = b.GetPixel(j, i);

int pValue = Convert.ToInt32(c.R); //Since it is in grey scale, we can get any pixel R,G or B

Histogram[pValue, 0] += 1; //counting how many pixels of each value

}

}

//computing the cdf (adding previous + current)

Histogram[0, 1] = Histogram[0, 0];

for (int i = 1; i<256; i++)

{

if (Histogram[i,0]!=0) //checking if the pixel exists in the bitmap

Histogram[i, 1] = Histogram[i - 1, 1] + Histogram[i, 0]; //using histogram at [i,0] because all the [i,1] at 0 at the begining

}

//Computing h (using given formula)

int cdfMin = 0;

int pixel = 0; //will be used to loop through the array to find first non-zero value

while (cdfMin == 0)

{

cdfMin = Histogram[pixel, 1];

pixel++;

}

int totalPixels = b.Width \* b.Height;

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

{

if(Histogram[i,1] !=0)

Histogram[i,2]= (int)(((Histogram[i, 1] - cdfMin) \* 255.0 / (totalPixels - 1)));

}

ApplyHistogram(b);

}

public static void ApplyHistogram(Bitmap b)

{

for (int col = 0; col < b.Width; col++)

{

for(int row=0; row<b.Height; row++)

{

Color c = b.GetPixel(col, row);

int pix = Convert.ToInt32(c.R);

b.SetPixel(col, row, Color.FromArgb(Histogram[pix, 2], Histogram[pix, 2], Histogram[pix, 2]));

}

}

}

}

}

**Conclusion:**

The first part of this assignment teach me different techniques to process an image which can be very useful such as resizing, contrast and brightening an image.

The second part of the assignment really helped me to understand better the concepts explained in class and how an image can be equalized applying the histogram equalization algorithm.