Goncalo Mendes

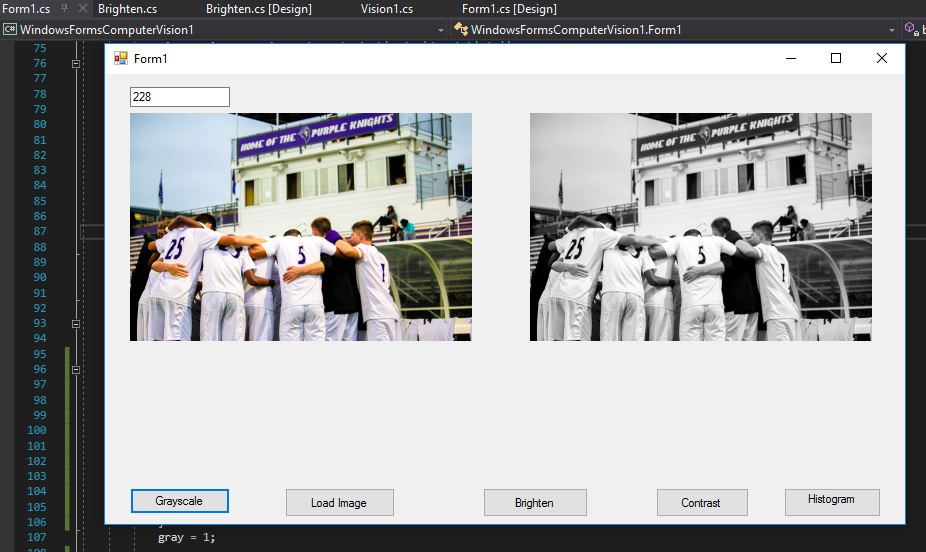
COMPUTER VISION

Assignment 1

In this assignment, ImageProcesing\_2010 C# project provided in handouts was recreated to understand the different possibilities of manipulation and processing of images that will later on be useful.

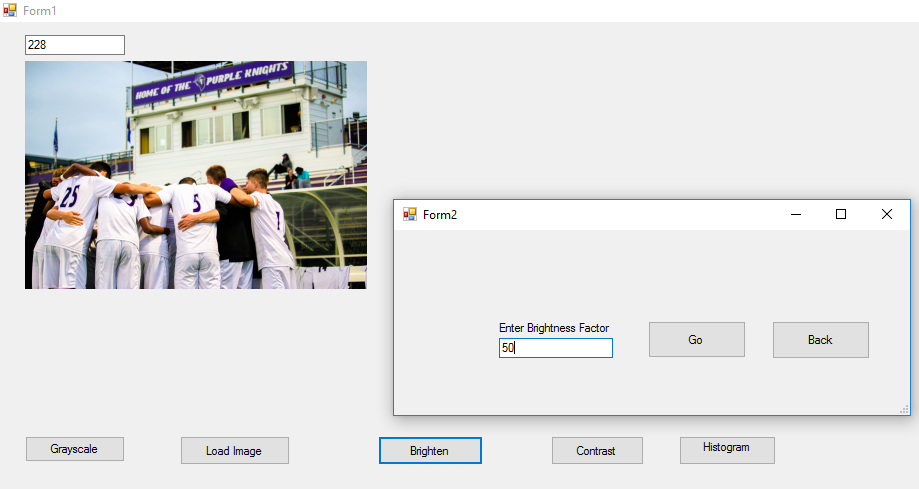
In the first part, I studied the notes taken from the lectures and paired them with the code provided to understand what is done.

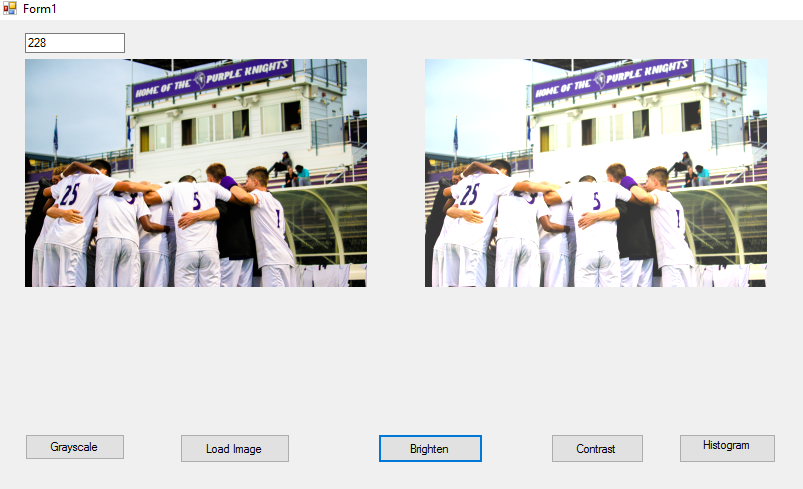
The first example was the function to convert the image into grayscale.



Through this, I learned the details of designing the forms and establishing the connections between the code from that design and our classes or main solution. This can even include the pop-ups with specifications like the brightness factor the user has to introduce, which is a form inside a form, and that has to be thought of in advance.

The second was the Brightening of the image and the contrast, which as mentioned before, demanded another small form to pop-up when these processes were called so that the user could specify the parameters.





The following source code is for the recreation of most of the essential functions of image processing the example given provided, like the ones above, as well as contrast and resizing.

Class : Vision1 – containing image processing function logic.

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Drawing.Imaging;

using System.Windows.Forms;

namespace WindowsFormsComputerVision1 {

public class Vision1

{

int gray=0;

int image=0;

public Vision1()

{

}

public static bool CovertToGray(Bitmap b, int gray)

{

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;

gray = (byte)(.299 \* r1

+ .587 \* g1

+ .114 \* b1);

r1 = gray;

g1 = gray;

b1 = gray;

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

}

}

gray = 1;

return true;

}

public static Bitmap Brighten(Bitmap b, int nBrightness)

{

int Red, Green, Blue;

if (nBrightness < -255 || nBrightness > 255) return b;

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

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

{

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

{

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

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

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

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

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;

}

FORM 1 – Design code for the main form.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsComputerVision1

{

public partial class Form1 : Form

{

int gray = 0;

int image = 0;

int equalized = 0;

public Form1()

{

InitializeComponent();

}

private void button5\_Click(object sender, EventArgs e)

{

if (image == 0) MessageBox.Show("No image has been loaded");

if (gray == 0) MessageBox.Show("Image has not yet been converted to Gray-scale");

try

{

Bitmap result = new Bitmap(this.OriginalPicBox.Image);

//Vision1.CovertToGray(result, gray);

Vision1.Histogram\_Equalization(result, gray, image, equalized);

this.ResultPicBox.Image = result;

MessageBox.Show("Ran thw ");

//ResultImageBox.Image = result;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

equalized = 1;

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void button2\_Click\_1(object sender, EventArgs e)

{

OpenFileDialog dialog = new OpenFileDialog();

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

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

{

Bitmap pic = new Bitmap(dialog.FileName);

while (pic.Width > OriginalPicBox.Width || pic.Height > OriginalPicBox.Height)

pic = new Bitmap(pic, new Size(pic.Width / 2, pic.Height / 2));

this.OriginalPicBox.Image = pic;

FileInfo finfo = new FileInfo(dialog.FileName);

SizeInfo.Text = OriginalPicBox.Image.Width.ToString();

SizeInfo.Text = OriginalPicBox.Image.Height.ToString();

}

image = 1;

}

private void button1\_Click\_1(object sender, EventArgs e)

{

try

{

Bitmap result = new Bitmap(this.OriginalPicBox.Image);

Vision1.CovertToGray(result, gray);

this.ResultPicBox.Image = result;

//ResultImageBox.Image = result;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

gray = 1;

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

Brighten dlg = new Brighten();

dlg.Brightness = 0;

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

{

Bitmap bright = new Bitmap(this.OriginalPicBox.Image);

bright = Vision1.Brighten(bright, dlg.Brightness);

ResultPicBox.Image = null;

ResultPicBox.Image = bright;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

FORM BRIGHTNESS (pop-up)

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 WindowsFormsComputerVision1

{

public partial class Brighten : Form

{

public int Brightness

{

get

{

return int.Parse(Input\_Bright.Text);

}

set { Input\_Bright.Text = value.ToString(); }

}

public Brighten()

{

InitializeComponent();

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

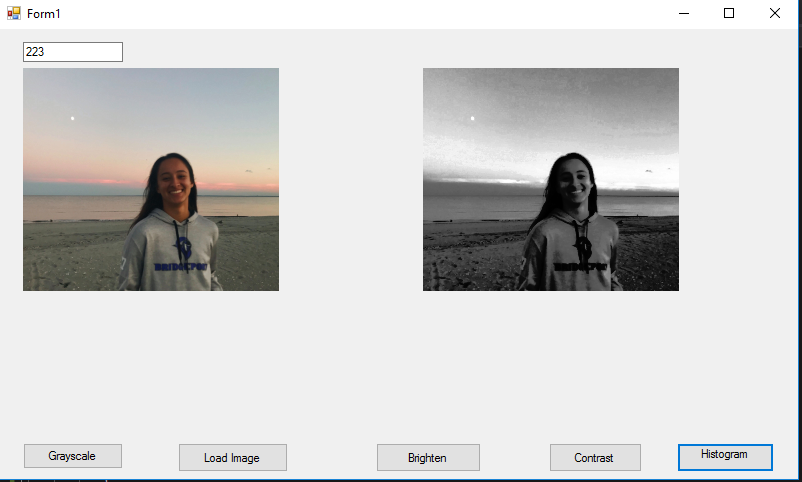
{

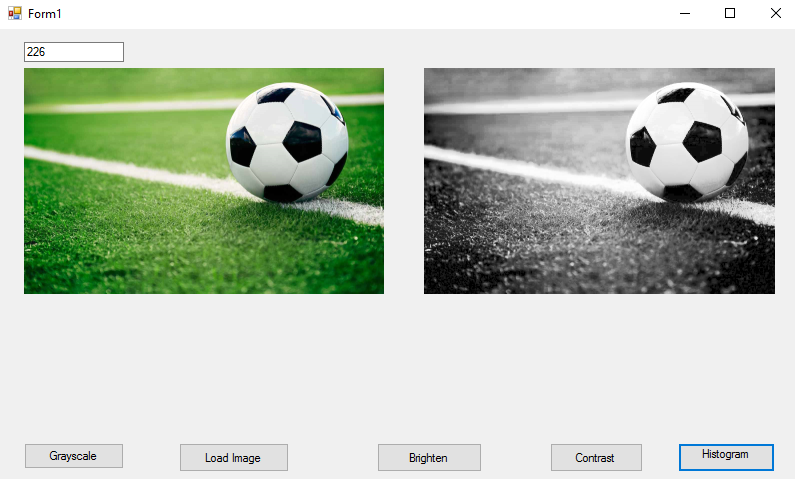
}

}

}

Finally, for the second part of the assignment, there is my own implementation of the Histogram Equalization. The details of the technique were given by Wikipedia as specified and the results seem to be good since we can easily spot the differences in highlight of shapes and details.





Below is specifically the source code for the histogram equalization function.

FORM :

private void button5\_Click(object sender, EventArgs e)

{

if (image == 0) MessageBox.Show("No image has been loaded");

if (gray == 0) MessageBox.Show("Image has not yet been converted to Gray-scale");

try

{

Bitmap result = new Bitmap(this.OriginalPicBox.Image);

//Vision1.CovertToGray(result, gray);

Vision1.Histogram\_Equalization(result, gray, image, equalized);

this.ResultPicBox.Image = result;

MessageBox.Show("Ran thw ");

//ResultImageBox.Image = result;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

equalized = 1;

}

FUNCTION – within class “Vision1”

public static void Histogram\_Equalization(Bitmap b, int gray, int image, int Equalized)

{

if (image != 1) MessageBox.Show("No image has been loaded");

if (gray != 1) MessageBox.Show("Image has to be converted to Grayscale first");

//Bitmap result = new Bitmap(b.Width, b.Height);

int[] intensity = new int[256];

int[] histogram = new int[256];

int min = 0;

int count = 0;

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

{

intensity[i] = 0;

histogram[i] = 0;

}

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

{

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

{

Color c1 = b.GetPixel(x, y);

int pixel = Convert.ToInt32( c1.R);

intensity[pixel] = intensity[pixel] + 1;

}

}

int pix = 0;

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

{

if(intensity[i] !=0)

{

intensity[i] = intensity[pix] + intensity[i];

pix=i;

}

}

while (min == 0)

{

min = intensity[count];

count++;

}

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

{

if (intensity[i] != 0)

{

histogram[i] = (int)( ( (intensity[i] - min) \*255 ) / ( (b.Height \* b.Width) - min));

}

}

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

{

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

{

int converted\_pixel = 0;

Color c1 = b.GetPixel(x, y);

converted\_pixel = histogram[c1.R];

b.SetPixel(x, y, Color.FromArgb(histogram[c1.R], histogram[c1.R], histogram[c1.R]));

}

}

Equalized += 1;

}