Goncalo Mendes

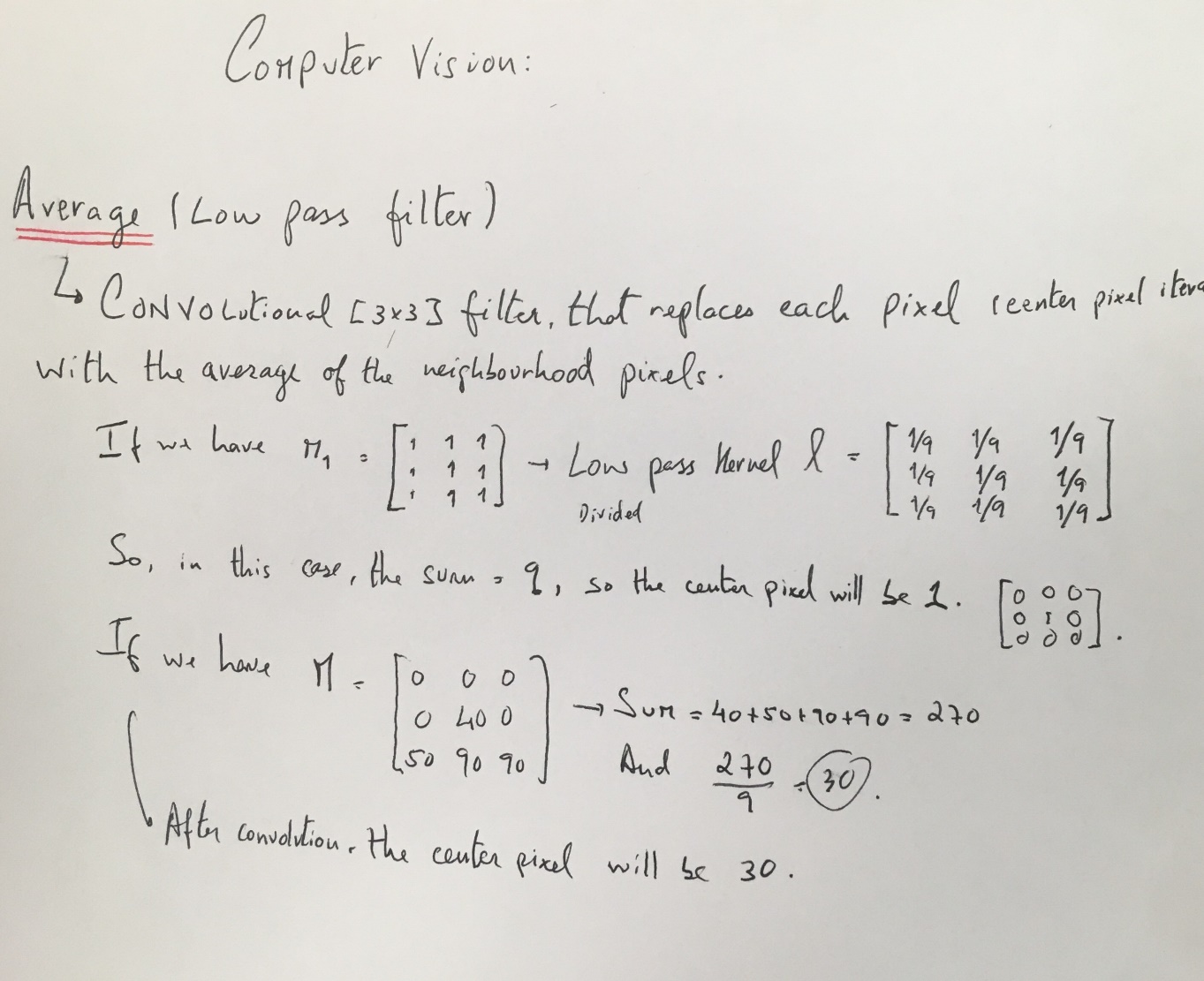
Computer Vision – CPEG 585

Assignment 2

For this assignment, the main goal was to go over the Mathematics behind each kernel as detailed as possible so we understand what we can use in the future for the course and to what kinds of applications each can be used as a tool to achieve certain things.

The first kernel was the Identity, which produces the exact same image, and in case of images is a square matrix with all zeros except the center value, that is set to 1.

**Average Kernel**

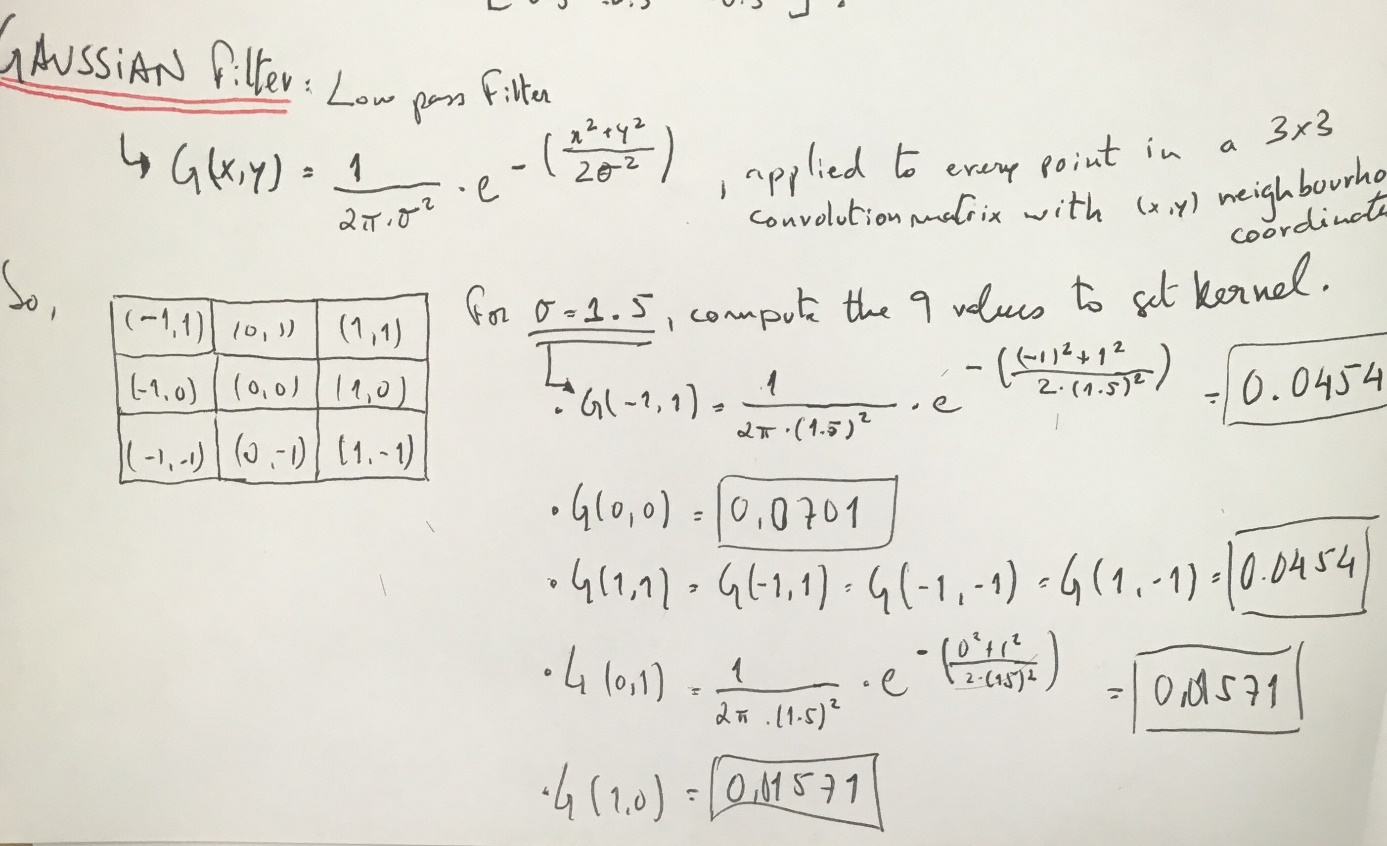
We took a look at the signals background behind these concepts and realized that we can put to use low pass filters to avoid big spikes and too high range of frequencies. The first example is the Average filter that will blur the image a little, due to be taking the average through convolution every 3x3 matrix. 

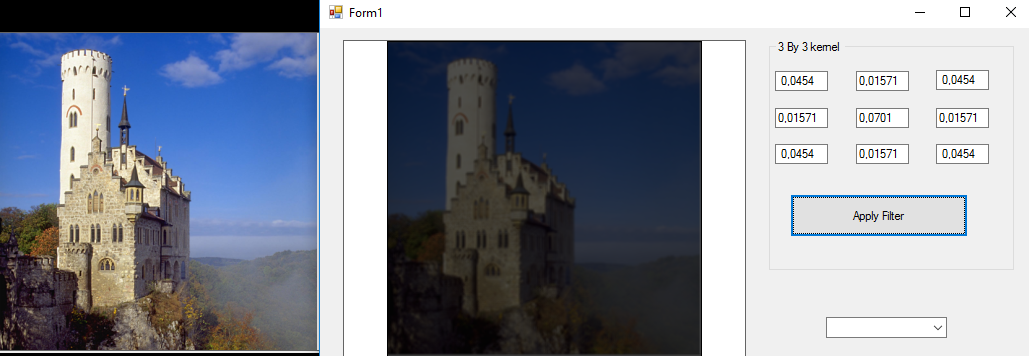
Below, after running the average kernel, we are able to notice some blur on the image.



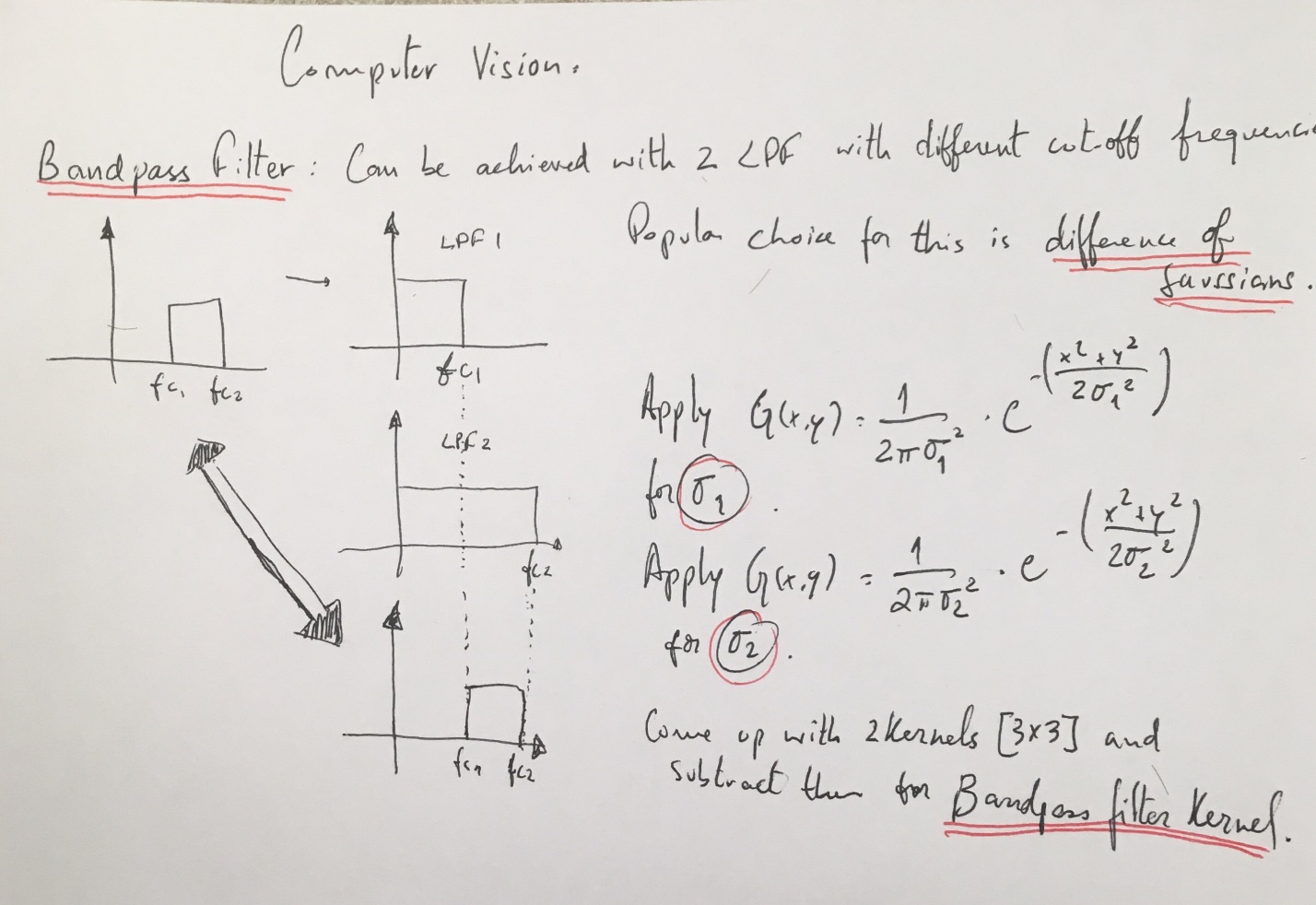
**2D – Gaussian Filter**

As shown in the math below, the Gaussian takes the formula of a 2D normal distribution and applies it to each center pixel’s neighborhood, acting as a low pass filter, since the formula cuts high values (frequencies).





**Bandpass Filter**

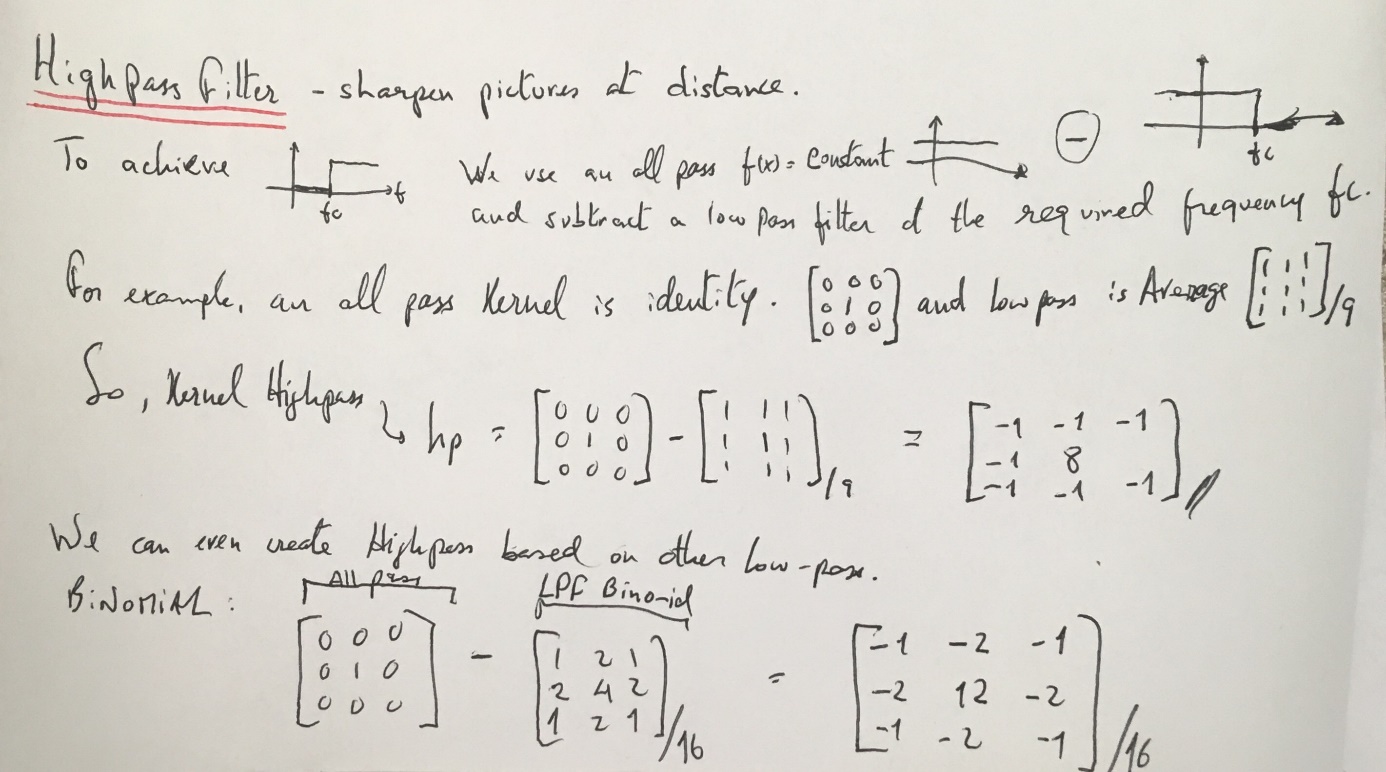
This filter is achieved by taking two low pass filter kernels that have two different cut-off frequencies. A way to achieve this is through difference of gaussians, since each is a Low Pass filter, and by taking different values of sigma, we can control the frequency change behavior and acquire a certain band between whose values the filter will pass the signal and after which it will block the signal. 

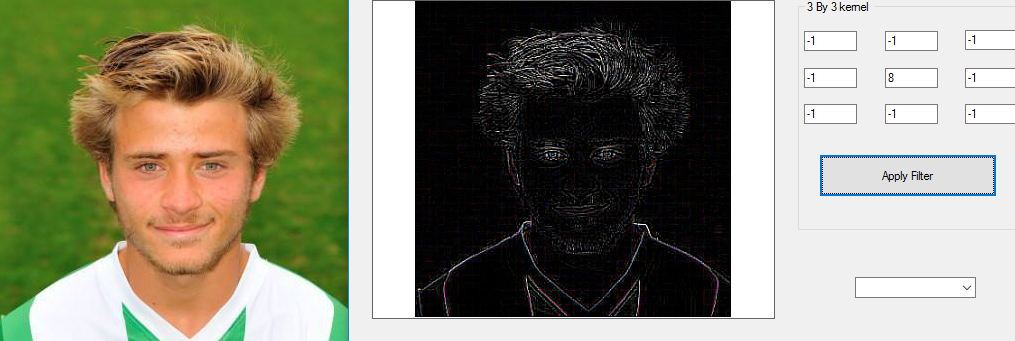
Using the numbers computed for the first Gaussian filter (sigma = 1.2 ) and a second computation (sigma = 1.5 ), we can get two matrixes as Gaussian kernels. If we subtract them, we obtain the difference of gaussians, which will act as a bandpass filter.

**High Pass Filter**

We can achieve a high pass filter by taking an all pass filter and subtracting a low pass filter.

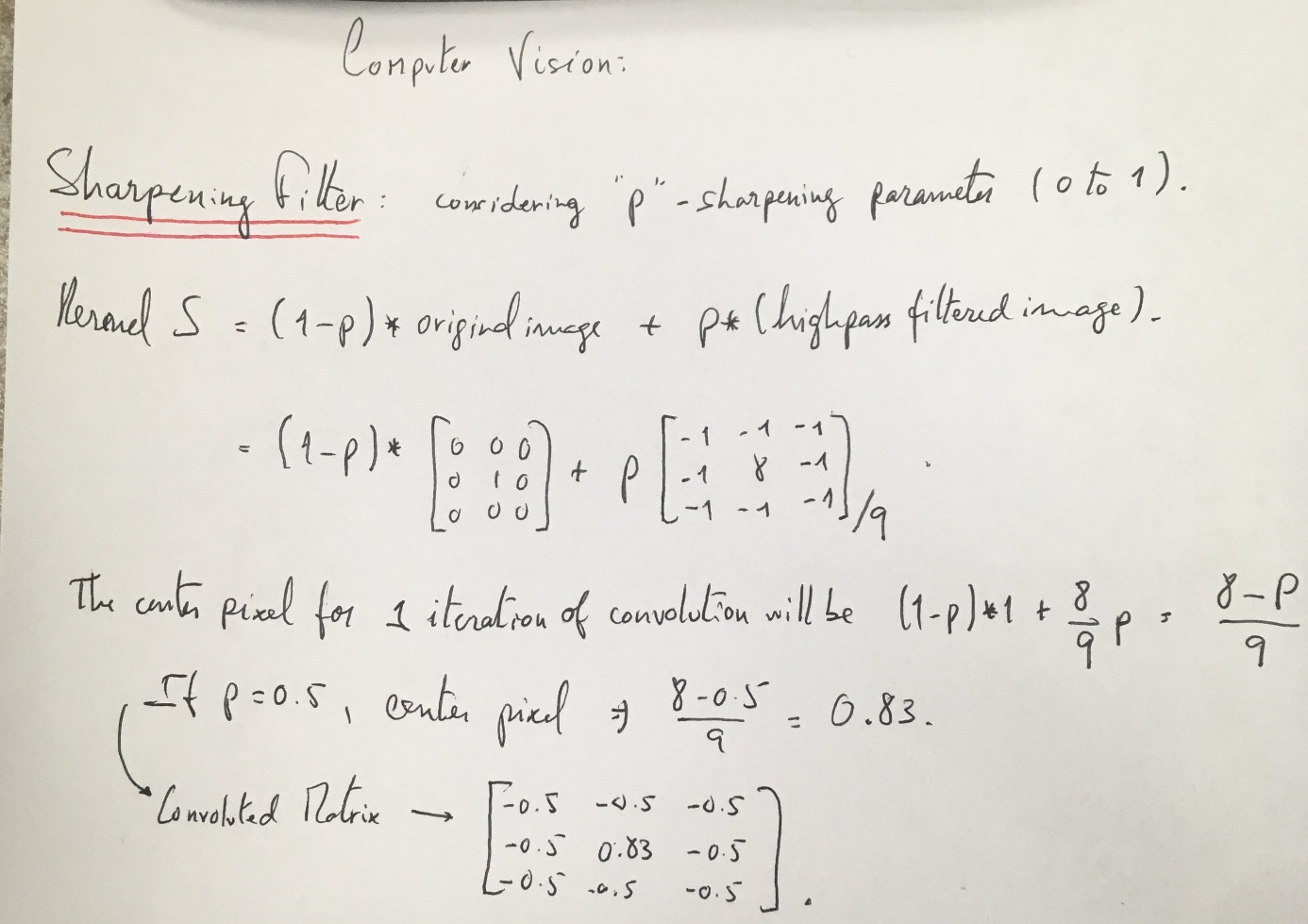
As we can see below, only high frequencies stay, giving us the brighter pixels in the image.



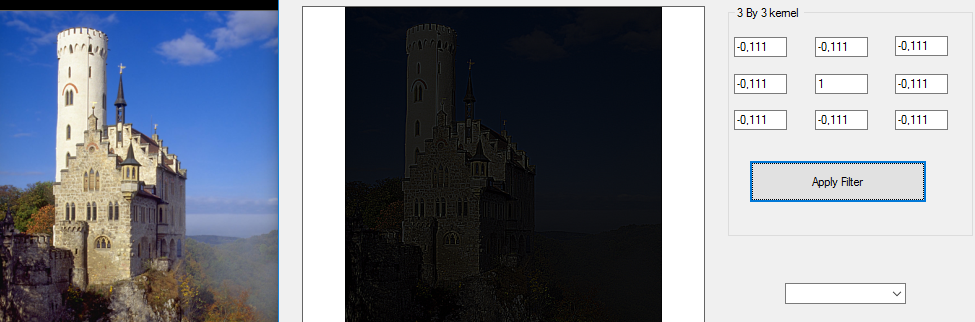


**Sharpening Filter**

This example, we can sharpen an image by taking the sharpening parameter p (0 to 1), that is then multiplied by the original image, and adding the same parameter multiplied by the image that results from a high pass filter.

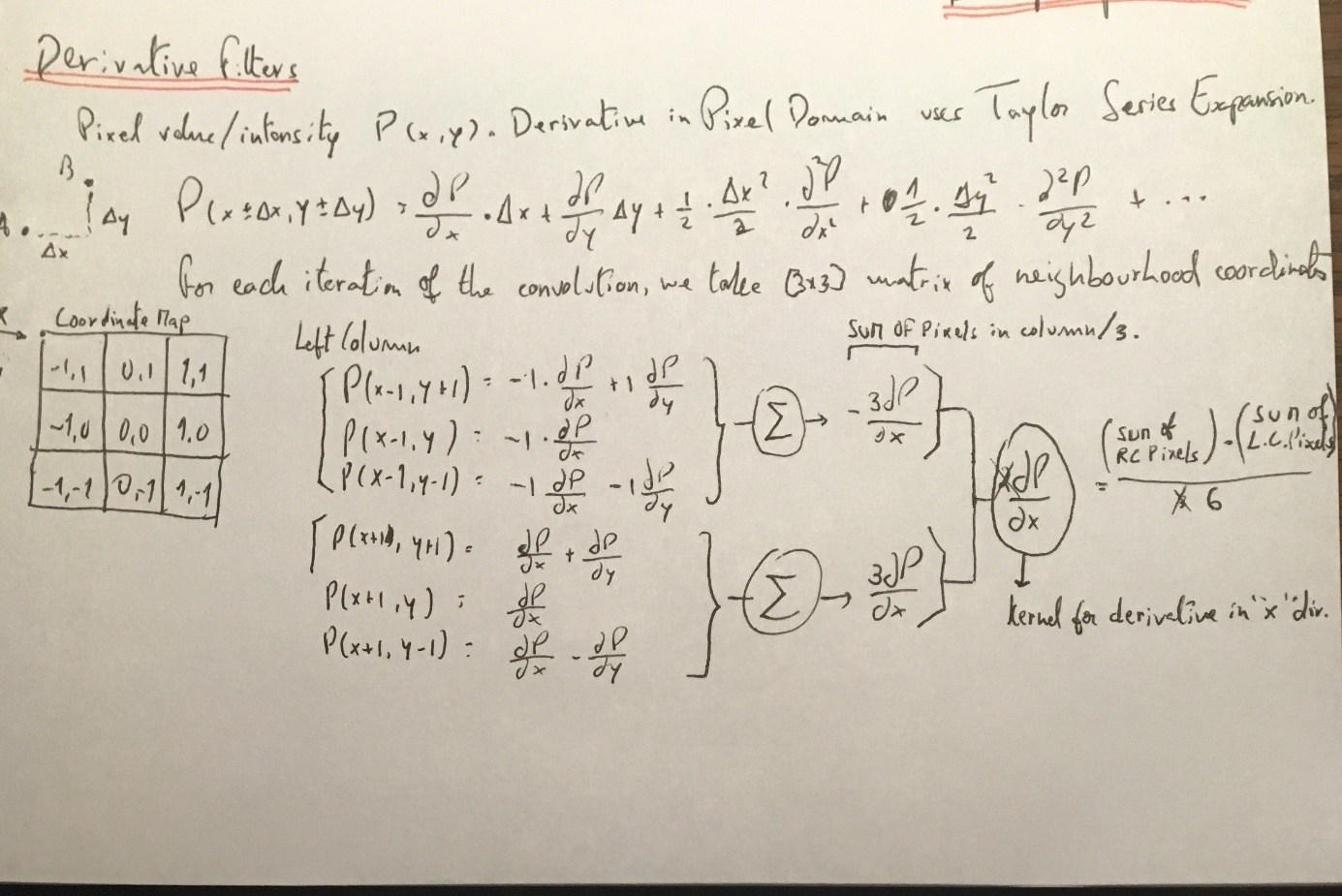


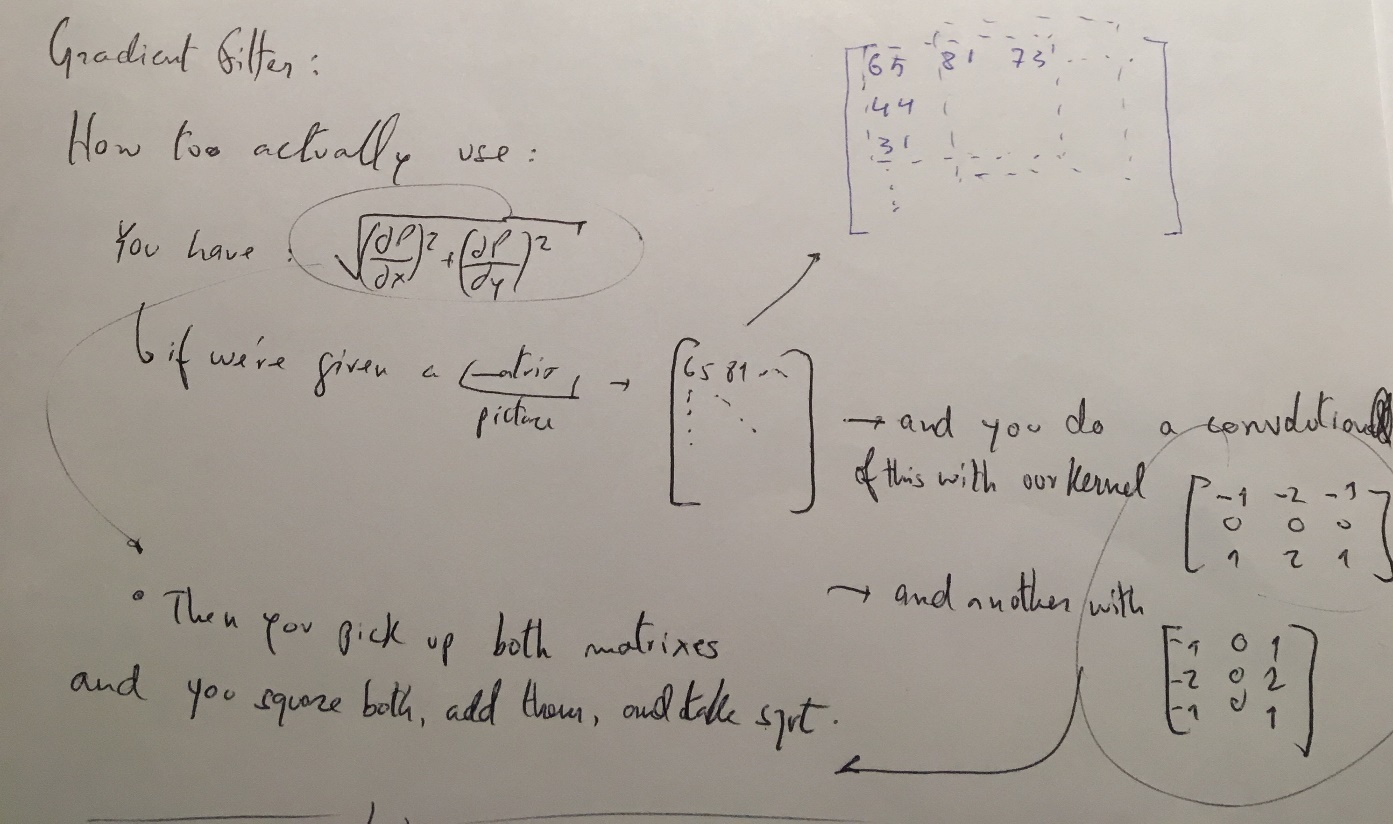
In this case, the image also lost a lot of brightness, which should not be the case, but we can notice how sharp the countours and differences between colours are.



**Derivative Filters** – First Derivative – Gradient Filter

As shown in the math below, we use the Taylor Series approximation to find derivatives in the pixel domain. Then, for each pixel, we are trying to find a kernel that portrays our derivative in respect to either x or y. Convolution is also done with at least a 3x3 matrix, so to get this derivative in function of either of the variables, we look at the changes in relation to the neighborhood of the center pixel.

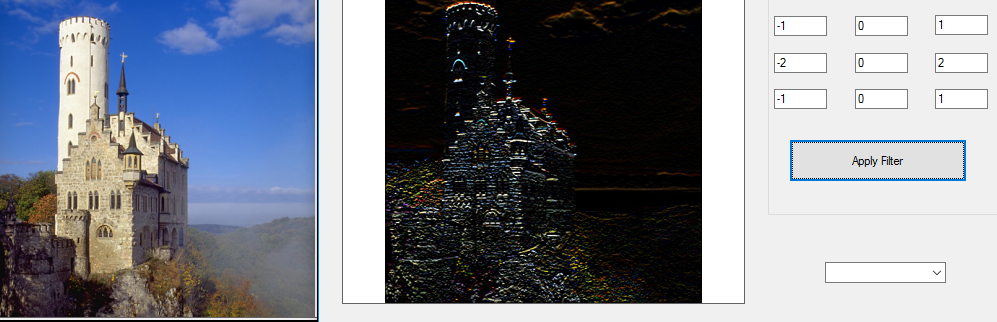


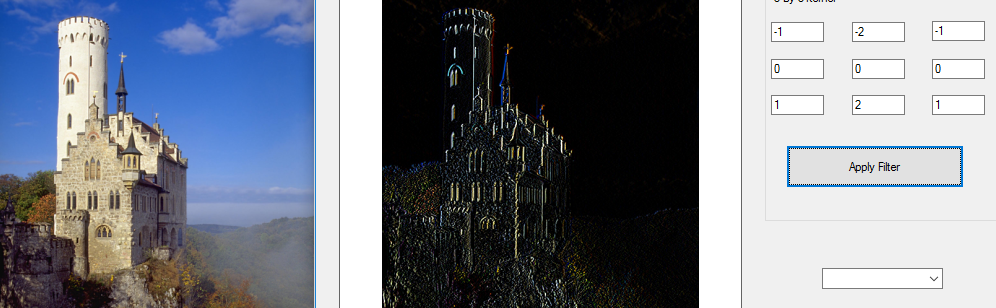


After running the example achieved, called the Sobel Filter, this is what we can see.

We can notice the geometric difference of applying this filter to the different axis, x and y.

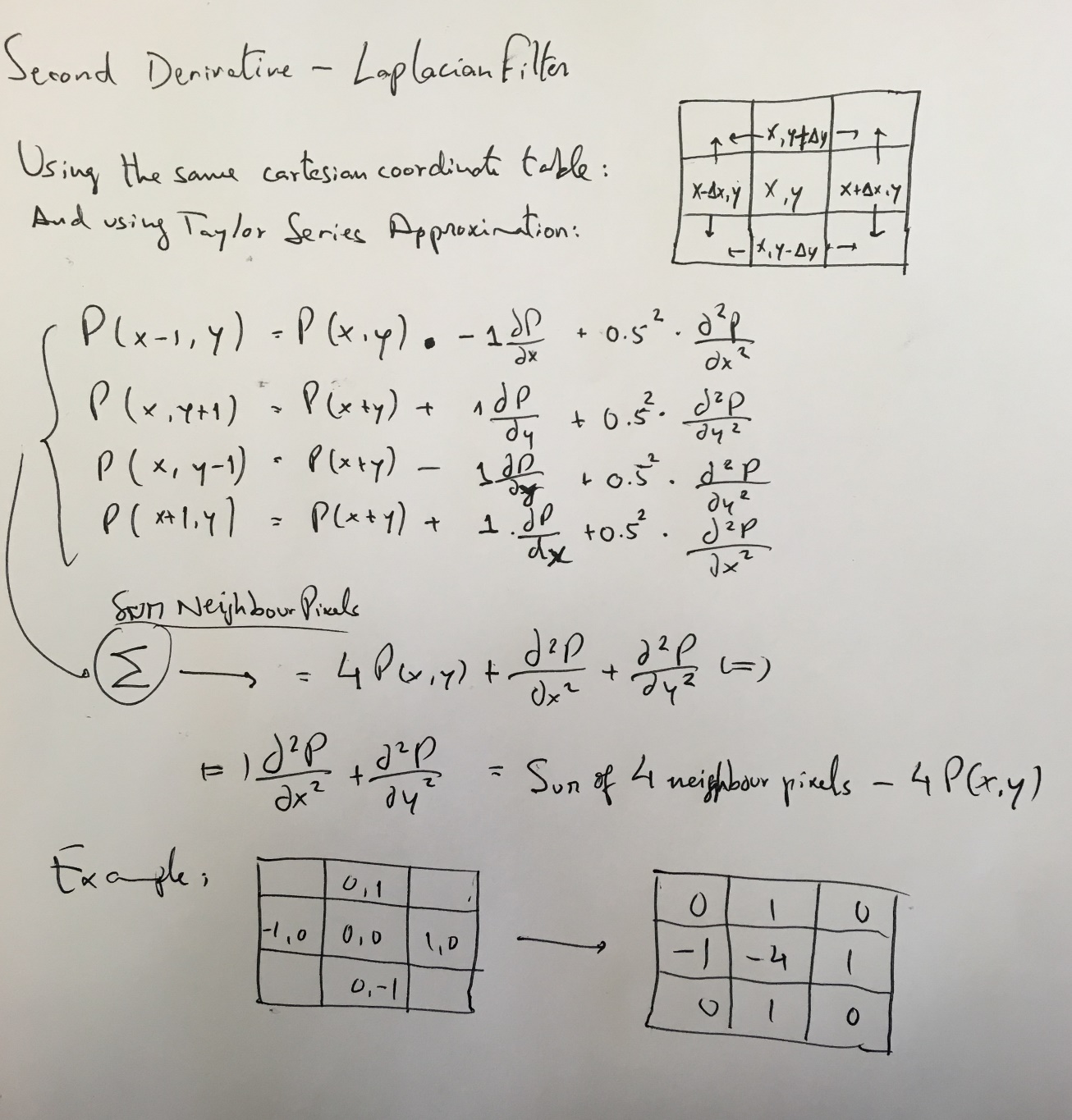
In the Y direction and then X direction, respectively.

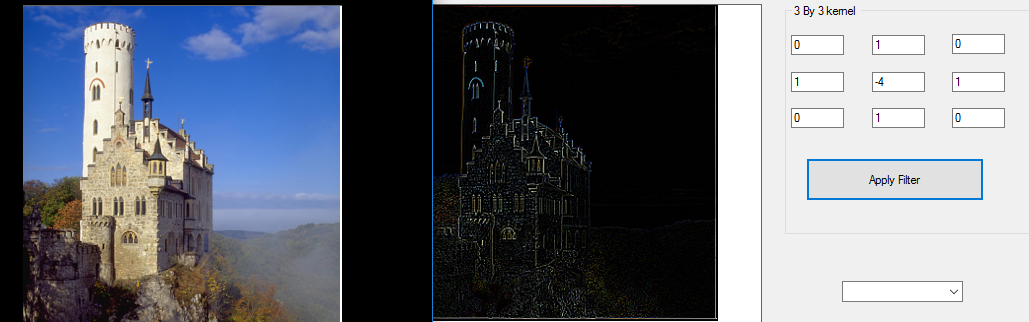




**Second Derivative – Laplacian Filter**

By using the same 3x3 map that resembles the coordinates in a 2D axis like shown in the map before we can then compute the second derivative through the Taylor Series approximation and come up with the Laplacian Kernel.





Source Code for the Convulotion Class developed:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace Image\_Convolution

{

class convolution

{

public Bitmap Convolution(Bitmap Original\_Image, double[,] Mask\_Matrix)

{

// get the values of pixels of the original image and store them in one D array

double[] pixelArray = ReadPixels(Original\_Image);

// declare a new 1 D array to handel the new values of pixels after manipulation:

double[] resultArray = new double[(Original\_Image.Width \* 3) \* Original\_Image.Height];

// loop through image pixels and do the convultion opration

for (int ImgY = 1; ImgY < Original\_Image.Height - 1; ImgY++)

{

for (int ImgX = 1; ImgX < Original\_Image.Width - 1; ImgX++)

{

// declare colors

double B = 0;

double G = 0;

double R = 0;

int imgCo = ImgY \* (Original\_Image.Width \* 3) + ImgX \* 3;

for (int Msk\_Y = -1; Msk\_Y <= 1; Msk\_Y++)

{

for (int Msk\_X = -1; Msk\_X <= 1; Msk\_X++)

{

int MskC = imgCo + (Msk\_X \* 3) + (Msk\_Y \* (Original\_Image.Width \* 3));

R += (double)(pixelArray[MskC]) \* Mask\_Matrix[Msk\_Y + 1, Msk\_X + 1];

G += (double)(pixelArray[MskC + 1]) \* Mask\_Matrix[Msk\_Y + 1, Msk\_X + 1];

B += (double)(pixelArray[MskC + 2]) \* Mask\_Matrix[Msk\_Y + 1, Msk\_X + 1];

}

}

// scale the value between 0 - 255

if (R > 255)

R = 255;

else if (R < 0)

R = 0;

if (B > 255)

B = 255;

else if (B < 0)

B = 0;

if (G > 255)

G = 255;

else if (G < 0)

G = 0;

resultArray[imgCo] = (byte)(R);

resultArray[imgCo + 1] = (byte)(G);

resultArray[imgCo + 2] = (byte)(B);

}

}

Bitmap resultImage = new Bitmap(Original\_Image.Width, Original\_Image.Height);

// create a new Bitmap & assign the new pixel values to it.

int Size = 0;

for (int y = 0; y < Original\_Image.Height; y++)

{

for (int x = 0; x < Original\_Image.Width; x++)

{

resultImage.SetPixel(x, y, Color.FromArgb(Convert.ToInt32(resultArray[Size]), Convert.ToInt32(resultArray[Size + 1]), Convert.ToInt32( resultArray[Size + 2]) ));

Size = Size + 3;

}

}

return resultImage;

}

double[] ReadPixels(Bitmap Original\_Image)

{

// retrieve the pixel values and store it in vector

Color c;

double[] xx = new double[Original\_Image.Width \* Original\_Image.Height \* 3];

int i = 0;

for (int y = 0; y < Original\_Image.Height; y++)

{

for (int x = 0; x < Original\_Image.Width; x++)

{

c = Original\_Image.GetPixel(x, y);

xx[i] = (double)(c.R);

xx[i + 1] = (double)(c.G);

xx[i + 2] = (double)(c.B);

i = i + 3;

}

}

return xx;

}

}

}

Source Code for the the form designed:

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;

using System.IO;

using System.Drawing.Imaging;

namespace Image\_Convolution

{

public partial class Form1 : Form

{

// declare classes and variables

Kernel Operator = new Kernel();

convolution convolution = new convolution();

public Bitmap origina\_Image = null;

public Bitmap resultImg = null;

public Form1()

{

InitializeComponent();

}

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

{

// load img

OpenFileDialog ofd = new OpenFileDialog();

ofd.Title = "Select an image file.";

ofd.Filter = "Png Images(\*.png)|\*.png|Jpeg Images(\*.jpg)|\*.jpg";

ofd.Filter += "|Bitmap Images(\*.bmp)|\*.bmp";

if (ofd.ShowDialog() == System.Windows.Forms.DialogResult.OK)

{

StreamReader streamReader = new StreamReader(ofd.FileName);

origina\_Image = (Bitmap)Bitmap.FromStream(streamReader.BaseStream);

streamReader.Close();

pictureBox1.Image = origina\_Image;

// enable the butttons on form

ApplyMaskBtn.Enabled = true;

textBox1.Text = String.Empty;

textBox2.Text = String.Empty;

textBox3.Text = String.Empty;

textBox4.Text = String.Empty;

textBox5.Text = String.Empty;

textBox6.Text = String.Empty;

textBox7.Text = String.Empty;

textBox8.Text = String.Empty;

textBox9.Text = String.Empty;

}

}

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

{

double[,] MyMask = new double[3, 3];

MyMask[0, 0] = Convert.ToDouble( textBox1.Text);

MyMask[0, 1] = Convert.ToDouble(textBox2.Text);

MyMask[0, 2] = Convert.ToDouble(textBox3.Text);

MyMask[1, 0] = Convert.ToDouble(textBox4.Text);

MyMask[1, 1] = Convert.ToDouble(textBox5.Text);

MyMask[1, 2] = Convert.ToDouble(textBox6.Text);

MyMask[2, 0] = Convert.ToDouble(textBox7.Text);

MyMask[2, 1] = Convert.ToDouble(textBox8.Text);

MyMask[2, 2] = Convert.ToDouble(textBox9.Text);

resultImg = convolution.Convolution(origina\_Image, MyMask);

pictureBox1.Image = resultImg;

}

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

{

SaveFileDialog dlg = new SaveFileDialog();

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

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

this.pictureBox1.Image.Save(dlg.FileName, ImageFormat.Jpeg);

}