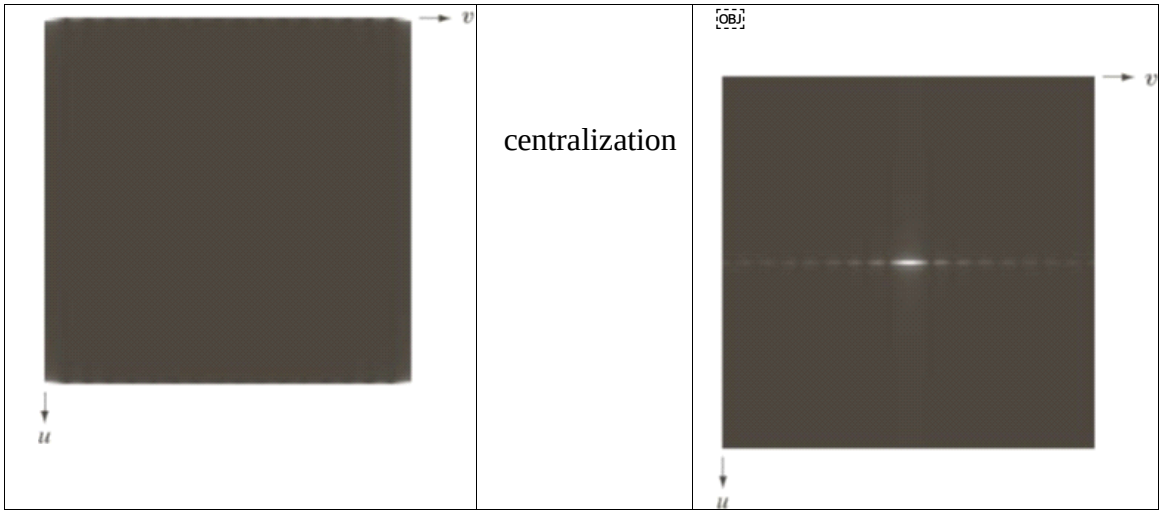
NTUT, Gratuate Institute of Automation Technology

Digital Image Processing Instructor：Chen Chin-Sheng (陳金聖)

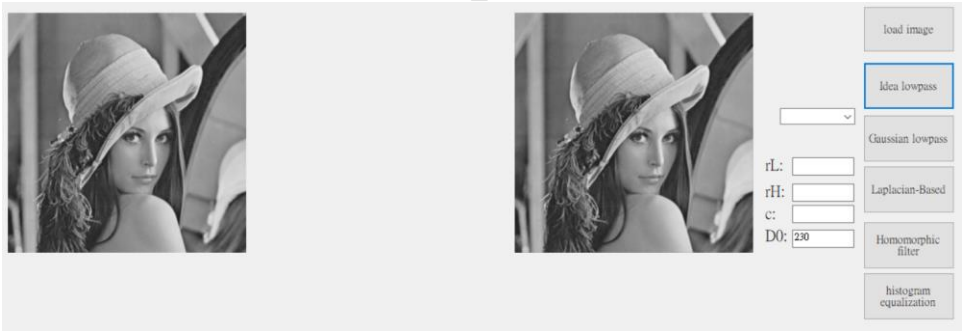
Homework Assignment #3 Due date: 2022/12/06

1. (20%) Spectrum shift property
   (1) Prove that $f(x,y)e_{j2\pi(u^{0x/M+u0y/N})} \Leftrightarrow F(u-u,v-v_0)$.
   (2) If you want to center the spectrum, show in the following figures, what processyou need to do. Please prove it from the result of (a).

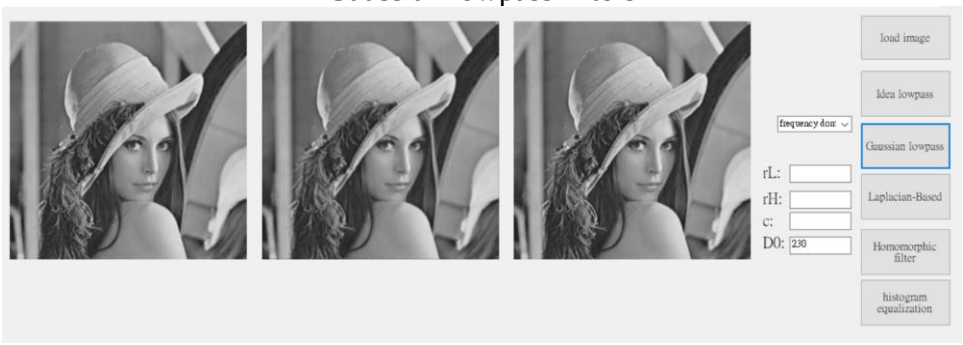


2. (40%)Programming smoothing and sharping enhancement in frequency domain:
   (1) Take the ideal and Gaussian lowpass filters for Fig. 4.40 (4th global edition) and your own image by frequency-domain approach. Comparing the results with the spatial mask filtering.

ILPF_Filter



Gaussian lowpass filters



左圖第一張為原始圖，中間為 time domin，右邊為 frequency domin。

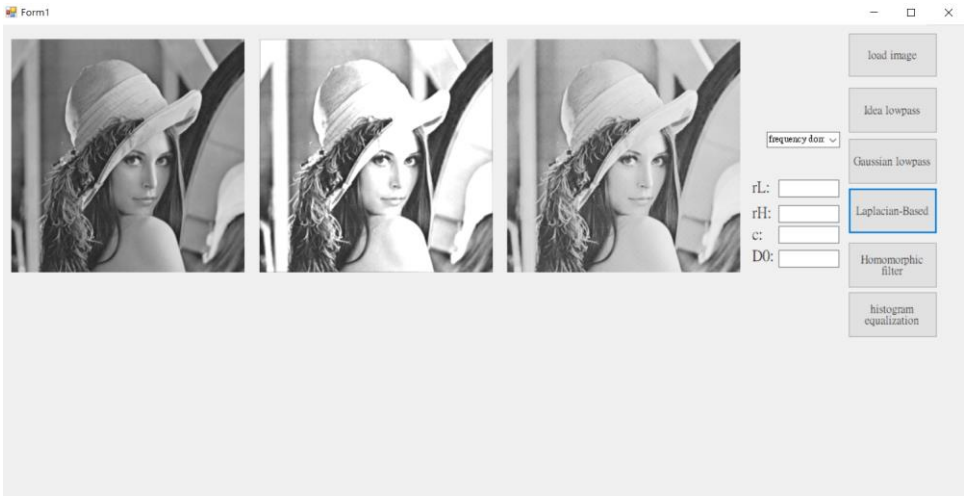
```
public void GLPF_Filter(IntPtr src, IntPtr src2, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);
    int Height = iImage.GetHeight(src);
    int pad_Width = Width * 2;
    int pad_Height = Height * 2;
    byte[,] Imagematrix = new byte[Height, Width];
    double[,] pad_matrix = new double[pad_Height, pad_Width];
    err = iImage.iPointerFromiImage(Image, ref Imagematrix[0, 0], Width, Height);
    print_error(err);
    for (int i = 0; i < Height; i++)
    {
        for (int j = 0; j < Width; j++)
        {
            pad_matrix[i, j] = Imagematrix[i, j];
        }
    }
    Fftshift(pad_matrix, pad_Height, pad_Width);
    Complex[,] F = new Complex[pad_Height, pad_Width];

    for (int i = 0; i < pad_Height; i++)
    {
        for (int j = 0; j < pad_Width; j++)
        {
            F[i, j] = (Complex)pad_matrix[i, j];
        }
    }
    FourierTransform.FFT2(F, FourierTransform.Direction.Forward);
```

```
double d0 = Convert.ToDouble(tb_d0.Text);
double n = pad_Height / 2;
double m = pad_Width / 2;

Complex[,] H = new Complex[pad_Height, pad_Width];
Complex[,] G = new Complex[pad_Height, pad_Width];

for (int i = 0; i < pad_Height; i++)
{
    for (int j = 0; j < pad_Width; j++)
    {
        double d = Math.Sqrt(Math.Pow((i - n), 2) + Math.Pow((j - m), 2));
        H[i, j] = (Complex)Math.Exp((-1 * d) / (2 * d0 * d0));
        G[i, j] = F[i, j] * H[i, j];
    }
}
FourierTransform.FFT2(G, FourierTransform.Direction.Backward);

for (int i = 0; i < pad_Height; i++)
{
    for (int j = 0; j < pad_Width; j++)
    {
        pad_matrix[i, j] = Convert.ToDouble(G[i, j].Re);
    }
}
Fftshift(pad_matrix, pad_Height, pad_Width);
for (int i = 0; i < Height; i++)
{
    for (int j = 0; j < Width; j++)
    {
        Imagematrix[i, j] = Convert.ToByte(pad_matrix[i, j]);
    }
}
```

```
void Fftshift(double[,] data, int h, int w)
{
    int Height = h, Width = w;
    double[,] tmp = new double[Height, Width];
    for (int i = 0; i < Height; i++)
    {
        for (int j = 0; j < Width; j++)
        {
            tmp[i, j] = Convert.ToDouble(data[i, j] * Math.Pow(-1, i + j));
        }
    }
    for (int i = 0; i < Height; i++)
    {
        for (int j = 0; j < Width; j++)
        {
            data[i, j] = tmp[i, j];
        }
    }
}
public void Gaussian_3_3_Filter(IntPtr src, IntPtr src2, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);
    int Height = iImage.GetHeight(src);
    byte[,] Imagematrix = new byte[Height, Width];
    byte[,] Imagematrix_af = new byte[Height, Width];
    int mask_size = 3;
    double[,] Gaussian_filter = new double[mask_size, mask_size];
    err = iImage.iPointerFromiImage(Image, ref Imagematrix[0, 0], Width, Height);
    print_error(err);
    double sigma = 7;
    double mask_total = 0;
    for (int i = 0; i < mask_size; i++)
    {
        for (int j = 0; j < mask_size; j++)
        {
            Gaussian_filter[i, j] = (1 / (2 * Math.PI * sigma * sigma)) *
                Math.Exp(-1 * (Math.Pow(i - (mask_size / 2), 2) + Math.Pow(j - (mask_size / 2), 2)) / (2 * sigma * sigma));
            mask_total += Gaussian_filter[i, j];
        }
    }
double number = 0;
for (int i = 0; i < Height; i++)
{
    for (int j = 0; j < Width; j++)
    {
        number = 0;
        for (int s = -1 * (mask_size / 2); s < (mask_size / 2) + 1; s++)
        {
            for (int t = -1 * (mask_size / 2); t < (mask_size / 2) + 1; t++)
            {
                if ((i + s) >= 0 && (i + s) < Height && (j + t) >= 0 && (j + t) < Width)
                {
                    number += Imagematrix[i + s, j + t] * Gaussian_filter[s + (mask_size / 2), t + (mask_size / 2)];
                }
                else
                {
                    number += 0 * Gaussian_filter[s + (mask_size / 2), t + (mask_size / 2)];
                }
            }
        }
        number = number / mask_total;
        if (number > 255)
        {
            number = 255;
        }
```

```
            if (number < 0)
            {
                number = 0;
            }
            Imagematrix_af[i, j] = (Byte)number;
        }
    }
```

(2) Take Laplacian-Based Enhancement for both “Lena image” and your own image by frequency-domain approach. Comparing the results with spatial enhancement.



左圖第一張為原始圖，中間為 time domin，右邊為 frequency domin。

```
public void Lapl_Filter(IntPtr src, IntPtr src2, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);
    int Height = iImage.GetHeight(src);
    int sWidth = Width, sHeight = Height;
    byte[,] Graymatrix = new byte[Height, Width];
    err = iImage.iPointerFromiImage(src, ref Graymatrix[0, 0], Width, Height);

    int paddingWidth = Convert.ToInt32(Math.Pow(2, (int)(Math.Ceiling(Math.Log(Convert.ToDouble(Width), 2))) + 1));     //轉換成2^n次方
    int paddingHeight = Convert.ToInt32(Math.Pow(2, (int)(Math.Ceiling(Math.Log(Convert.ToDouble(Height), 2))) + 1));   //假設510=>2^9 * 在加一 * 2^10

    double[,] paddingmatrix = new double[paddingHeight, paddingWidth];
    AForge.Math.Complex[,] bigmatrix = new AForge.Math.Complex[paddingHeight * 2, paddingWidth * 2];

    for (int i = 0; i < Height; i++)
        for (int j = 0; j < Width; j++)
            paddingmatrix[i, j] = Graymatrix[i, j];            //把影像複製過去不包括邊緣


    for (int i = 0; i < paddingHeight * 2; i++)
        for (int j = 0; j < paddingWidth * 2; j++)
            if (i < paddingHeight && j < paddingWidth)
                bigmatrix[i, j].Re = Math.Pow(-1, i + j) * paddingmatrix[i, j] / 255;   //(-1)^(x+y)
            else
                bigmatrix[i, j].Re = 0;


    AForge.Math.FourierTransform.FFT2(bigmatrix, AForge.Math.FourierTransform.Direction.Forward);//fft轉換
```

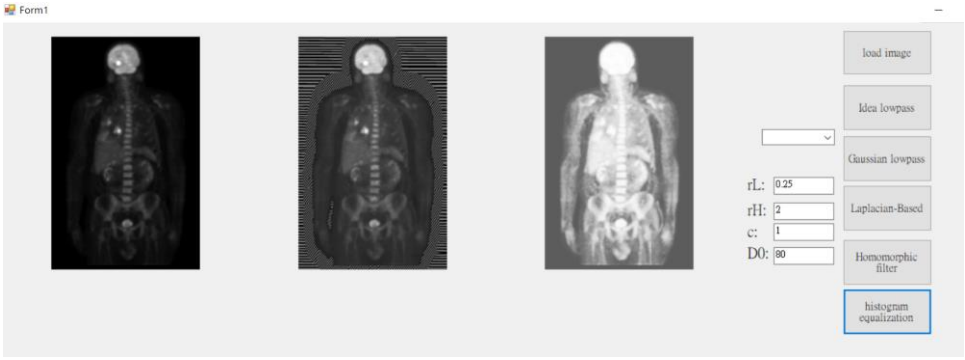
```
for (int i = 0; i < paddingHeight * 2; i++)
    for (int j = 0; j < paddingWidth * 2; j++)
        bigmatrix[i, j] = bigmatrix[i, j] * (4 * Math.Pow(Math.PI, 2) * (Math.Pow(i - sHeight / 2, 2) + Math.Pow((j - sWidth / 2), 2)));//Laplac

AForge.Math.FourierTransform.FFT2(bigmatrix, AForge.Math.FourierTransform.Direction.Backward);//ifft轉換
double min = 100;
double max = 0;

for (int i = 0; i < 2 * paddingHeight; i++)
    for (int j = 0; j < 2 * paddingWidth; j++)
    {
        if (bigmatrix[i, j].Re > 0)
            bigmatrix[i, j].Re = (Int64)(bigmatrix[i, j].Re + 0.5); //進位
        else
            bigmatrix[i, j].Re = 0;
        if (bigmatrix[i, j].Re > max)                           //紀錄大小(歸一化用)
            max = bigmatrix[i, j].Re;
        else if (bigmatrix[i, j].Re < min)                      //紀錄大小(歸一化用)
            min = bigmatrix[i, j].Re;
    }

byte[,] fouriermatrix = new byte[Height, Width];
for (int i = 0; i < Height; i++)
    for (int j = 0; j < Width; j++)
    {
        fouriermatrix[i, j] = (byte)(((bigmatrix[i, j].Re - min) * (255 / (max - min))) + Graymatrix[i, j]);    //歸一化最大值255追加原本的圖
        if (((bigmatrix[i, j].Re - min) * (255 / (max - min))) + Graymatrix[i, j] > 255)
            fouriermatrix[i, j] = 255;                                                                          //過大給予數直
    }
```

```
int Width = iImage.GetWidth(src);
int Height = iImage.GetHeight(src);

byte[,] Imagematrix = new byte[Height, Width];
byte[,] Imagematrix_af = new byte[Height, Width];
double[,] temp = new double[Height, Width];
err = iImage.iPointerFromiImage(Image, ref Imagematrix[0, 0], Width, Height);
print_error(err);
double number = 0;
int c = 1;
for (int i = 0; i < Height; i++)
{
    for (int j = 0; j < Width; j++)
    {
        number = 0;

        if (i - 1 >= 0 && i + 1 < Height && j - 1 >= 0 && j + 1 < Width)
        {
            number = Imagematrix[i + 1, j] + Imagematrix[i, j + 1] + Imagematrix[i - 1, j] + Imagematrix[i, j - 1];
            temp[i, j] = Imagematrix[i, j] + c * Math.Pow(number, 2);
        }
    }
}

double min = 100;
double max = 0;
for (int i = 0; i < Height; i++)
    for (int j = 0; j < Width; j++)
    {
        if (temp[i, j] > 0)
            temp[i, j] = temp[i, j] + 0.5; //進位
        else
            temp[i, j] = 0;
        if (temp[i, j] > max)                           //紀錄大小(歸一化用)
            max = temp[i, j];
        else if (temp[i, j] < min)                      //紀錄大小(歸一化用)
            min = temp[i, j];
    }

for (int i = 0; i < Height; i++)
    for (int j = 0; j < Width; j++)
    {
        Imagematrix_af[i, j] = (byte)(((temp[i, j] - min) * (255 / (max - min))) + Imagematrix[i, j]);    //歸一化最大值255追加原本的圖
        if (((temp[i, j] - min) * (255 / (max - min))) + Imagematrix[i, j] > 255)
            Imagematrix_af[i, j] = 255;
    }
```

3. (40%) Homomorphic filtering:

   (1) Design the Homomorphic filter, then apply to Fig. 4.60 (4th global edition) and your own image having illumination variations. Show your best output image and the corresponding tunning parameters.

(2) Enhance the original image by histogram equalization and compare the result with (1). Which one do you feel better? Explain why?



左圖第一張為原始圖，中間為 Homomorphic filter，右邊為 histogram equalization 。

```
for (int i = 0; i < padding_height * 2; i++)
{
    for (int j = 0; j < padding_width * 2; j++)
    {
        if (big_matrix[i, j].Re > 0)
        {
            big_matrix[i, j].Re = Math.Log(big_matrix[i, j].Re + 1, Math.E);
        }
    }
}

double min_value = 255; //歸一化
double max_value = 0;
for (int i = 0; i < Height; i++)              // i index for cols ( 0~Hight-1)(Because that matrix index is start from 0)
{
    for (int j = 0; j < Width; j++)           // j index for rows ( 0~Width-1)
    {
        if (padding_matrix[i, j] > max_value)
        {
            max_value = padding_matrix[i, j];
        }
        if (padding_matrix[i, j] < min_value)
        {
            min_value = padding_matrix[i, j];
        }
    }
}

for (int i = 0; i < padding_height * 2; i++)
{
    for (int j = 0; j < padding_width * 2; j++)
    {
        if (i < padding_height && j < padding_width)
        {
            padding_matrix[i, j] = (padding_matrix[i, j] - min_value) / (max_value - min_value);
            big_matrix[i, j].Re = Math.Pow(-1, i + j) * padding_matrix[i, j];   //(-1)^(x+y) 轉置
        }
        else
        {
            big_matrix[i, j].Re = 0;
        }
    }
}

AForge.Math.FourierTransform.FFT2(big_matrix, AForge.Math.FourierTransform.Direction.Forward); //fft傅立誠轉換
for (int i = 0; i < padding_height * 2; i++)
{
    for (int j = 0; j < padding_width * 2; j++)
    {
        big_matrix[i, j].Re = big_matrix[i, j].Re * ((rh_value - rl_value) *
            (1 - Math.Exp(-1 * c_value * (Math.Pow(i - Height / 2, 2) + Math.Pow(j - Width / 2, 2)) / (2 * d0_value * d0_value))) + rl_value);
        big_matrix[i, j].Im = big_matrix[i, j].Im * ((rh_value - rl_value) *
            (1 - Math.Exp(-1 * c_value * (Math.Pow(i - Height / 2, 2) + Math.Pow(j - Width / 2, 2)) / (2 * d0_value * d0_value))) + rl_value);
    }
}
AForge.Math.FourierTransform.FFT2(big_matrix, AForge.Math.FourierTransform.Direction.Backward);//ifft轉換
```

```
for (int i = 0; i < padding_height * 2; i++)
{
    for (int j = 0; j < padding_width * 2; j++)
    {
        if (big_matrix[i, j].Re > 0)
        {
            big_matrix[i, j].Re = Math.Exp(big_matrix[i, j].Re);
        }
    }
}

min_value = 255; //歸一化
max_value = 0;
for (int i = 0; i < padding_height * 2; i++)
{
    for (int j = 0; j < padding_width * 2; j++)
    {
        if (big_matrix[i, j].Re > 0)
        {
            big_matrix[i, j].Re = (int)Math.Ceiling(big_matrix[i, j].Re);
        }
        else
        {
            big_matrix[i, j].Re = 0;
        }
        if (big_matrix[i, j].Re > max_value)
        {
            max_value = big_matrix[i, j].Re;
        }
        if (big_matrix[i, j].Re < min_value)
        {
            min_value = big_matrix[i, j].Re;
        }
    }
}

byte[,] Graymatrix_af = new byte[Height, Width];
AForge.Math.Complex[,] big_matrix_af = new AForge.Math.Complex[padding_height * 2, padding_width * 2];
for (int i = 0; i < padding_height * 2; i++)
{
    for (int j = 0; j < padding_width * 2; j++)
    {
        big_matrix_af[i, j].Re = big_matrix[i, j].Re * Math.Pow(-1, i + j);
    }
}

for (int i = 0; i < Height; i++)
{
    for (int j = 0; j < Width; j++)
    {
        Graymatrix_af[i, j] = (byte)((big_matrix_af[i, j].Re - min_value) / (max_value - min_value) * 255 + Graymatrix[i, j]);

        if ((big_matrix_af[i, j].Re - min_value) / (max_value - min_value) * 255 + Graymatrix[i, j] > 255)
        {
            Graymatrix_af[i, j] = 255;
        }
```

```
public void His(IntPtr src, IntPtr src2, PictureBox pictureBox)
{
    int Width = iImage.GetWidth(src);        // Get image width
    int Height = iImage.GetHeight(src);      // Get image height
    byte[,] Graymatrix = new byte[Height, Width];
    byte[,] Graymatrix_his = new byte[Height, Width];
    int[] gray_level = new int[256];
    for (int i = 0; i < gray_level.Length; i++)
    {
        gray_level[i] = i;
    }
    int[] gray_org_count = new int[256];
    double[] gray_org_pr = new double[256];
    double[] gray_af_pr = new double[256];

    err = iImage.iPointerFromiImage(src, ref Graymatrix[0, 0], Width, Height);
    print_error(err);
    for (int i = 0; i < Height; i++)//統計每個灰階值出現次數
    {
        for (int j = 0; j < Width; j++)
        {
            gray_org_count[Graymatrix[i, j]] = gray_org_count[Graymatrix[i, j]] + 1;
        }
    }
    for (int i = 0; i < gray_level.Length; i++)//算出各個灰階值的 PDF
    {
        gray_org_pr[i] = gray_org_count[i] / (double)(Width * Height);
    }
for (int i = 0; i < gray_level.Length; i++)//算出各個灰階值的 PDF
{
    gray_org_pr[i] = gray_org_count[i] / (double)(Width * Height);
}
for (int i = 0; i < 256; i++) //將PDF 做累加求出 CDF
{
    for (int j = 0; j <= i; j++)
    {
        gray_af_pr[i] = gray_af_pr[i] + gray_org_pr[j];
    }
}
for (int i = 0; i < Height; i++)
{
    for (int j = 0; j < Width; j++)
    {
        Graymatrix_his[i, j] = (Byte)(gray_af_pr[Graymatrix[i, j]] * 255);
    }
}
```

Homomorphic filter 的結果比較好，因為其對比度比較明顯。