Assignment 5

/*叶俊利 3130100672*/

1. Mean Filter

First, Load the gray scale image into a 2-dimension matrix.

```
int h = imgOriginal.GetHeight();
int w = imgOriginal.GetWidth();
byte** grayScale = (byte**)malloc(h*sizeof(byte *));
for (int i = 0; i < h; i++){
        grayScale[i] = (byte*)malloc(w*sizeof(byte));
}
for (int i = 0; i < h; i++)
for (int j = 0; j < w; j++){
        byte r, g, b, avg;
        pixel = imgOriginal.GetPixel(j, i);
        r = GetRValue(pixel);
        g = GetGValue(pixel);
        b = GetBValue(pixel);
        avg = (r + g + b) / 3;
        grayScale[i][j] = avg;
}
```

Then, according to the equation

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

So we can process the image according to the equation.

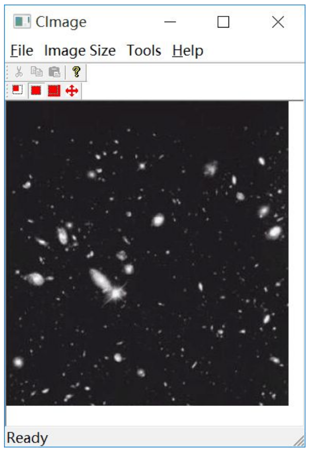
I make the weigh w=1, and the size of mask is a * b.

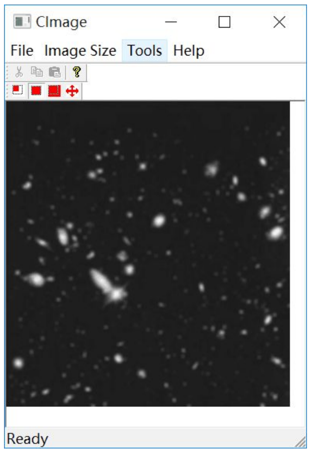
The code is shown below.

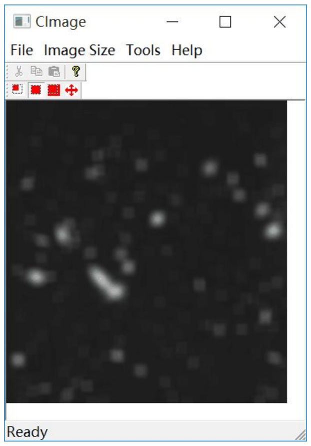
```
int a = 7;
int b = 7;
```

```
for (int i = 0; i < h; i++)
for (int j = 0; j < w; j++){
        double count = 0;
        double sum = 0;
        for (int ii = (i - a >= 0 ? i - a : 0); ii <= (i + a < h ? i + a : h - 1); ii++)
        for (int jj = (j - b >= 0 ? j - b : 0); jj <= (j + b < w ? j + b : w - 1); jj++){
                count++;
                sum += grayScale[ii][jj];
        }
        imgOriginal.SetPixelRGB(j, i, sum / count, sum / count, sum / count);
}
```

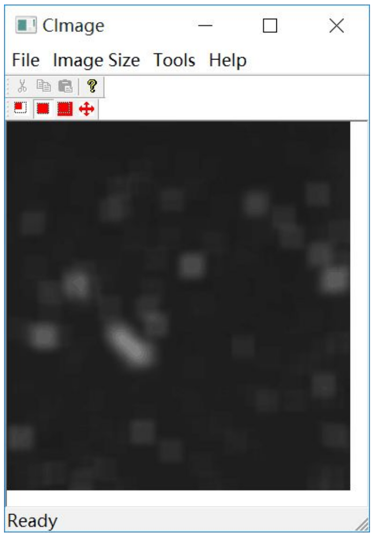
According to the code, we can get the result processed by mask of different size:



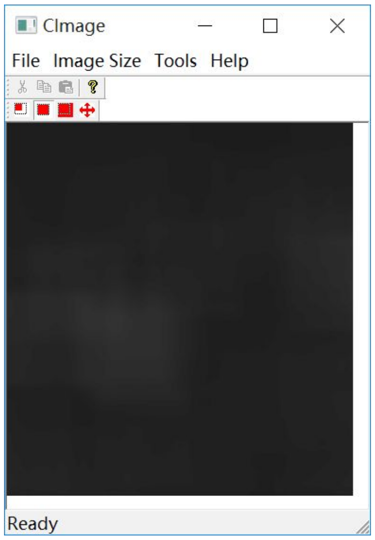




Origin　　　5*5 mask　　　15*15 mask



25*25 mask



101*101 mask

2. Laplacian Enhancement

First, Load the gray scale image into a 2-dimension matrix.

```
int h = imgOriginal.GetHeight();

int w = imgOriginal.GetWidth();

byte** grayScale = (byte**)malloc(h*sizeof(byte *));

for (int i = 0; i < h; i++){

        grayScale[i] = (byte*)malloc(w*sizeof(byte));

}

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

for (int j = 0; j < w; j++){

        byte r, g, b, avg;

        pixel = imgOriginal.GetPixel(j, i);

        r = GetRValue(pixel);

        g = GetGValue(pixel);

        b = GetBValue(pixel);

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

        grayScale[i][j] = avg;
        }
```

Then, according to the mask

0	1	0
1	-4	1
0	1	0

```
            int mask[3][3] = { { 0, 1, 0 }, { 1, -4, 1 }, { 0, 1, 0 } };
```

We can get the Laplacian Results, and record the max and min value:

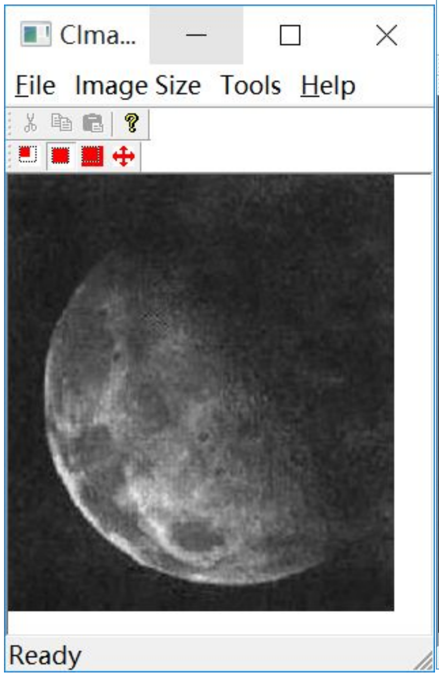
```
        for (int i = 1; i < h - 1; i++)
        for (int j = 1; j < w - 1; j++){
            double sum = 0;
            for (int x = -1; x<2; x++)
            for (int y = -1; y<2; y++)
            {
                sum += mask[x + 1][y + 1] * grayScale[i + x][j + y];
            }
            lap_res[i][j] = sum;
            if (sum  > max1)
                max1 = sum;
            if (sum <min1)
                min1 = sum;
            lap_res[i][j] = sum;
        }
```

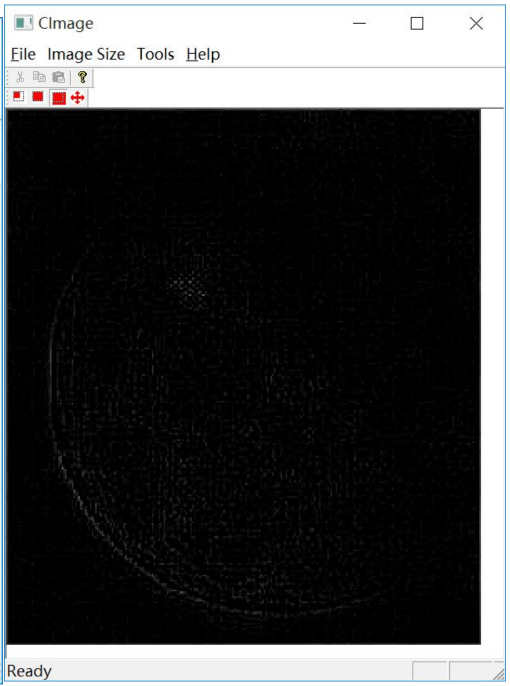
Then we need to rearrange the laplacian results, and fuse it with the origin image:

```
    for (int i = 1; i < h - 1; i++)
    for (int j = 1; j < w - 1; j++){
        final[i][j] = -(lap_res[i][j] - min1) * 50 / (max1 - min1) + grayScale[i][j];
        if (final[i][j]>255)
            final[i][j] = 255;
        if (final[i][j]<0)
```

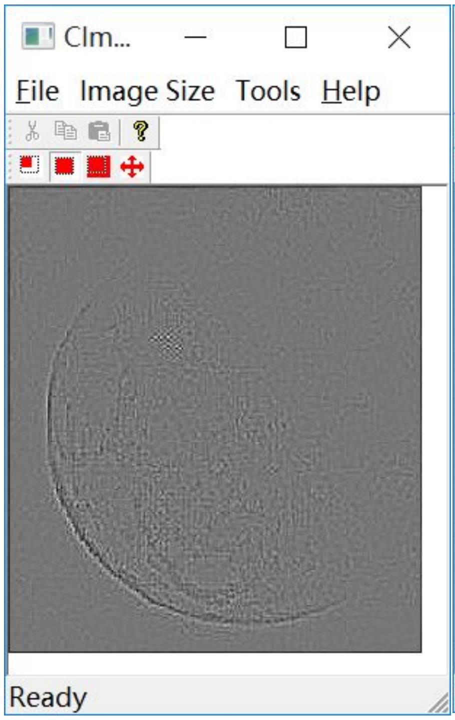
```
                final[i][j] = 0;
        }
```

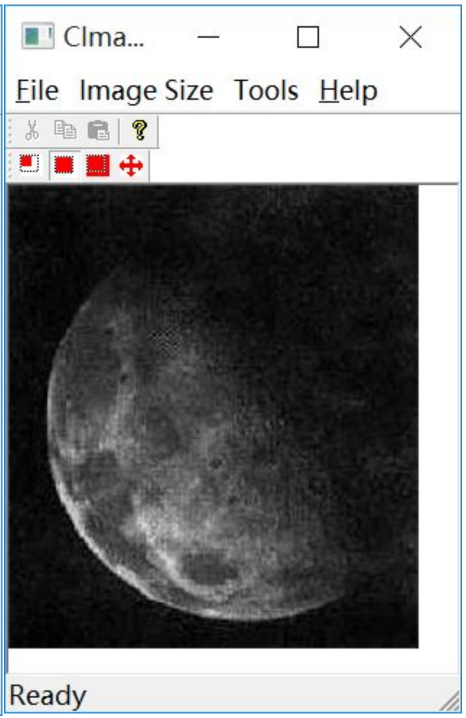
And here is the Results:





Original Laplacian Result





Rearranged Laplacian result Fusion