離散傅立葉轉換 DFT 練習

作業說明:撰寫傅利葉轉換程式(Forward Fourier Transform and Inverse Fourier Transform)將一張圖 像轉換至頻域後，將頻譜大小與相位角度各以灰階 256 色圖像方式呈現出，再呈現還 原後圖像。

語言:c++
開發環境:Windows 10 + Visual Studio 2019 + OpenCV 3.4.6

a.圖像轉換至頻域

擴展圖像矩陣，為 2，3，5 的倍數時運算速度快

```
int m = getOptimalDFTSize(img.rows);
int n = getOptimalDFTSize(img.cols);
copyMakeBorder(img, img, 0, m - img.rows, 0, n - img.cols, BORDER_CONSTANT, Scalar::all(0));
```

創建一個雙通道矩陣 planes，用來儲存複數的實部與虛部

```
Mat planes[2] = { Mat_<float>(img), Mat::zeros(img.size(), CV_32F) };
```

從多個單通道數組中創建一個多通道數組:transform_image。函數 Merge 將幾個數組合併為一個多通道陣列，即輸出數組的每個元素將是輸入數組元素的級聯

```
Mat transform_image;
merge(planes, 2, transform_image);
```

進行傅立葉變換

```
dft(transform_image, transform_image);
```

b.將頻譜大小與相位角度各以灰階 256 色圖像方式呈現出

計算複數的幅值與相位

```
split(transform_image, planes);// 將雙通道分為兩個單通道，一個表示實部，一個表示虛部
Mat magnitude_image, phase_img;
magnitude(planes[0], planes[1], magnitude_image);//計算複數的幅值
phase(planes[0], planes[1], phase_img)
```

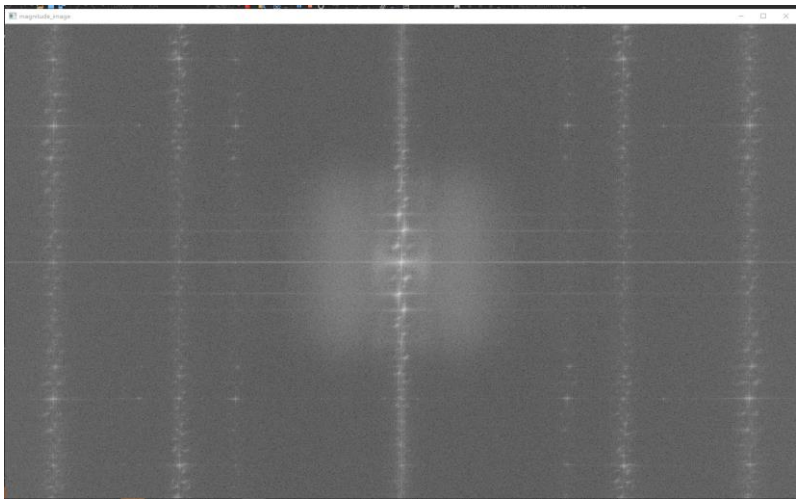
得到的頻譜圖與相位圖數級過大，不好顯示，因此轉換

```
magnitude_image += Scalar(1);
log(magnitude_image, magnitude_image);
normalize(magnitude_image, magnitude_image, 0, 1, NORM_MINMAX);
normalize(phase_img, phase_img, 0, 1, NORM_MINMAX);
```

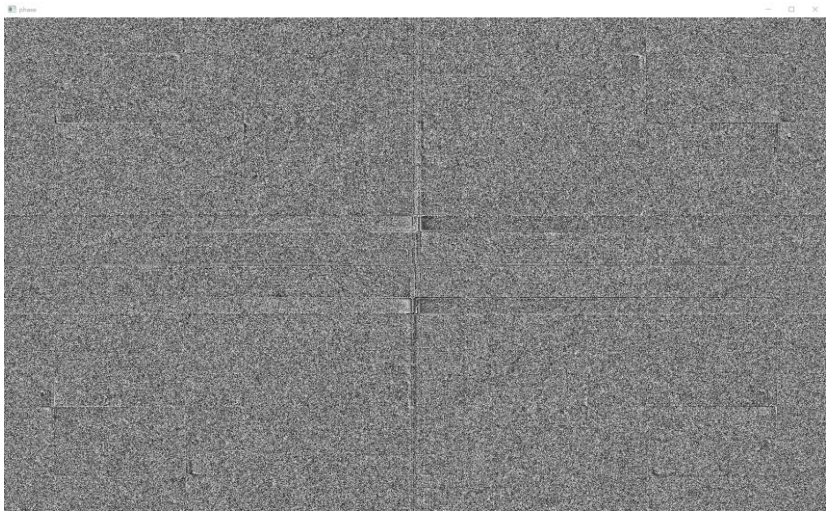
剪切和重分佈幅度圖像限，使原點位於圖像中心

成果:

頻譜圖



相位圖



c.呈現還原後圖像

對頻域圖像執行逆傅里葉變換，將圖像轉換回空域圖像。
使用 magnitude() 函數計算逆變換後的複數矩陣的幅值，以獲取還原後圖像的強度信息。

成果: