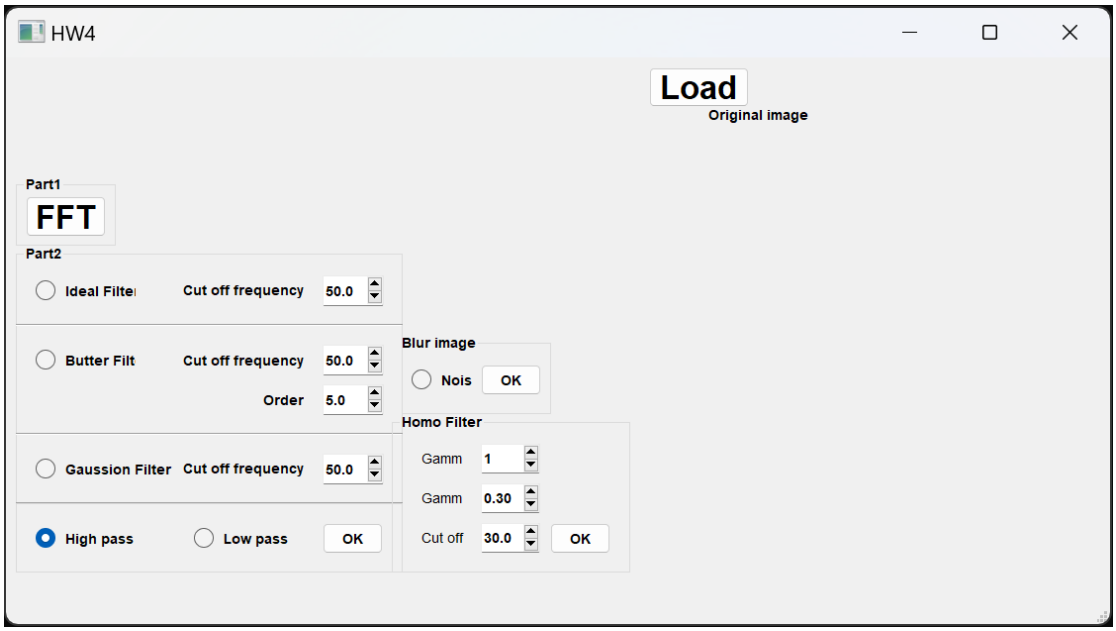
B09611007 陳柏霖

介面截圖
(請點擊 ”Load” 以選取欲測試照片)
圖片會顯示在右側



Part 1

FFT 按鈕的運算函數，生成三張輸出圖片

```
# Part 1
# FFT
def fft(self):
    if (fileName == ''):
        return

    result1 = np.dstack([r_show, g_show, b_show])
    self.do_showR1(result1)
    self.ui.lab_t1.setText(" Spectrum")

    r_p = phase_angle(dfred)
    g_p = phase_angle(dfgreen)
    b_p = phase_angle(dfblue)
    result2 = np.dstack([r_p,g_p,b_p])
    self.do_showR2(result2)
    self.ui.lab_t2.setText("Phase angle")

    r_i = ifft(dfred)
    g_i = ifft(dfgreen)
    b_i = ifft(dfblue)
    result3 = np.dstack([r_i,g_i,b_i])
    self.do_showR3(result3)
    self.ui.lab_t3.setText("Inverse img")
```

Part 2

```python
# Part 2
def secondpart(self):
    if (fileName == ''):
        return
    self.ui.lab_r1.clear(), self.ui.lab_r2.clear(), self.ui.lab_r3.clear()
    self.ui.lab_t1.clear(), self.ui.lab_t2.clear(), self.ui.lab_t3.clear()

    if  self.ui.rbtn_high.isChecked():
        passchoose = 1
    elif self.ui.rbtn_low.isChecked():
        passchoose = 0

    if self.ui.rbtn_ideal.isChecked():
        self.ideal(passchoose)
    elif self.ui.rbtn_gaussion.isChecked():
        self.gaussion(passchoose)
    elif self.ui.rbtn_butter.isChecked():
        self.butter(passchoose)
    else:
        pass
```

設計理想濾波器：

```python
# Ideal filter
def ideal(self,passchoose):
    if (fileName == ''):
        return

    r1 = ideal_filter(dfred,self.ui.ideal_cutoff.value(),passchoose)
    g1 = ideal_filter(dfgreen,self.ui.ideal_cutoff.value(),passchoose)
    b1 = ideal_filter(dfblue,self.ui.ideal_cutoff.value(),passchoose)
    r_i = ifft(r1)
    g_i = ifft(g1)
    b_i = ifft(b1)
    result = np.dstack([r_i,g_i,b_i])
    self.do_showR1(result)
    self.ui.lab_t1.setText("Ideal filter")
```

通過從中心點 center 出發，計算圖像中每個像素到中心的距離。
如果 lowpass 為 true，則檢查距離是否小於等於 cutoff，如果是，則在 ideal 中將對應像素位置設置為 1.0，這表示通過低通濾波器。
如果 lowpass 為 false，則檢查距離是否大於 cutoff，如果是，則在 ideal 中將對應像素位置設置為 1.0，這表示通過高通濾波器。
複數相乘：

逆傅立葉變換：

使用 ifft 函數做逆傅立葉變換，得到最終的處理後圖像。
通過 cv::normalize 函數將圖像的像素值正規化到範圍 [0, 255]。

設計巴特沃斯濾波器（Butterworth Filter）：

```
# Butterworth filter
def butter(self,passchoose):
    if (fileName == ''):
        return
    r1 = bufilter(dfred,self.ui.butter_cutoff.value(),self.ui.butter_order.value(),passchoose)
    g1 = bufilter(dfgreen,self.ui.butter_cutoff.value(),self.ui.butter_order.value(),passchoose)
    b1 = bufilter(dfblue,self.ui.butter_cutoff.value(),self.ui.butter_order.value(),passchoose)
    r_i = ifft(r1)
    g_i = ifft(g1)
    b_i = ifft(b1)
    result = np.dstack([r_i,g_i,b_i])
    self.do_showR1(result)
    self.ui.lab_t1.setText("Butter filter")
```

通過從中心點 center 出發，計算圖像中每個像素到中心的距離。
如果 lowpass 為 true（表示低通濾波器），則使用巴特沃斯低通濾波器公式設計濾波器：
butter.at<float>(i, j) = 1.0 / (1.0 + pow(distance / cutoff, 2 * order))
如果 lowpass 為 false（表示高通濾波器），則使用巴特沃斯高通濾波器公式設計濾波器：
butter.at<float>(i, j) = 1.0 - 1.0 / (1.0 + pow(distance / cutoff, 2 * order))
cutoff 控制過濾的截止頻率，order 控制濾波器的阶數，這些參數可以調整以改變濾波效果。

這段程式實現了一個巴特沃斯濾波器，可以通過設置 cutoff、order 和 lowpass 來控制過濾的頻率範圍和濾波效果，以處理輸入圖像，過濾特定頻率的成分。

設計高斯濾波器：

```
# Gaussion filter
def gaussion(self,passchoose):
    if (fileName == ''):
        return

    r1 = gauss_filter(dfred,self.ui.gaussion_cutoff.value(),passchoose)
    g1 = gauss_filter(dfgreen,self.ui.gaussion_cutoff.value(),passchoose)
    b1 = gauss_filter(dfblue,self.ui.gaussion_cutoff.value(),passchoose)
    r_i = ifft(r1)
    g_i = ifft(g1)
    b_i = ifft(b1)
    result = np.dstack([r_i,g_i,b_i])
    self.do_showR1(result)
    self.ui.lab_t1.setText("Gaussian filter")
```

這段程式是一個實現高斯濾波器（Gaussian Filter）的函數，用於圖像處理和頻域處理。高斯濾波器是一種常用的平滑濾波器，用於減少圖像中的噪聲和細節，它的過濾效果基於高斯分佈函數。這段程式的功能如下：

通過從中心點 `center` 出發，計算圖像中每個像素到中心的距離。如果 `lowpass` 為 true（表示低通濾波器），則使用高斯分佈函數設計濾波器：

- `gaussian.at<float>(i, j) = exp(-pow(distance, 2) / (2 * pow(cutoff, 2))`
- 如果 `lowpass` 為 false（表示高通濾波器），則設計反高斯濾波器：
    - `gaussian.at<float>(i, j) = 1.0 - exp(-pow(distance, 2) / (2 * pow(cutoff, 2))`
- `cutoff` 控制過濾的截止頻率，這個值越大，濾波器越寬，保留更多低頻信息。

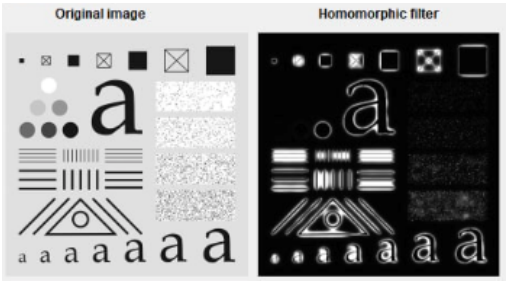
可以通過設置 `cutoff` 和 `lowpass` 來控制過濾的頻率範圍和濾波效果，以處理輸入圖像，平滑圖像並減少噪聲。

Part 3

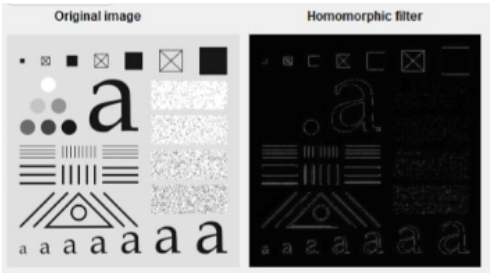
```python
# Homomorphic filter
def homo(self):
    if (fileName == ''):
        return

    self.ui.lab_r1.clear(), self.ui.lab_r2.clear(), self.ui.lab_r3.clear()
    self.ui.lab_t1.clear(), self.ui.lab_t2.clear(), self.ui.lab_t3.clear()

    r1 = homo_filter(img_arr[:, :, 0],self.ui.spin_rh.value(),self.ui.spin_rl.value(),self.ui.spin_d0.value())
    g1 = homo_filter(img_arr[:, :, 1],self.ui.spin_rh.value(),self.ui.spin_rl.value(),self.ui.spin_d0.value())
    b1 = homo_filter(img_arr[:, :, 2],self.ui.spin_rh.value(),self.ui.spin_rl.value(),self.ui.spin_d0.value())
    result = np.dstack([r1,g1,b1])
    self.do_showR1(result)
    self.ui.lab_t1.setText("Homomorphic filter")
```



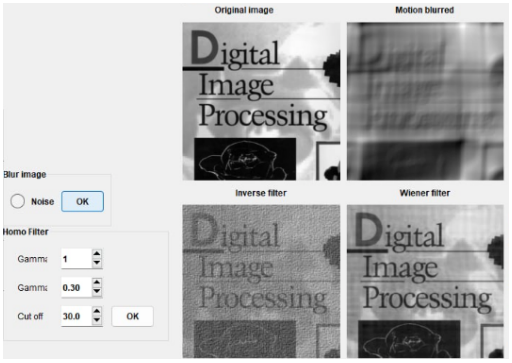
Gamma H=3
Gamma L=0.1
Cut-off frequency=20



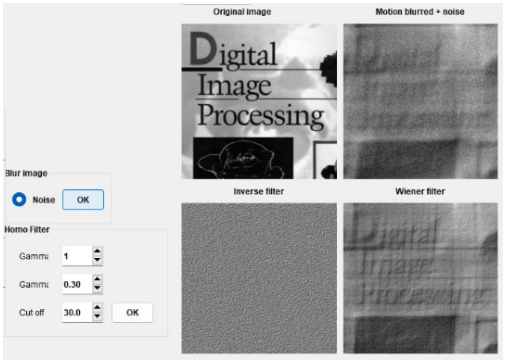
Gamma H=1
Gamma L=0.1
Cut-off frequency=20

當 gamma H 與 gamma L 的差值越大，輸出影像的邊緣亮度越大，細部的反射會更明顯。

Part 4



Motion Blurred



Motion Blurred + Gaussian Noise

在只有 Motion 造成的 image degradation，inverse filter 和 Weiner filter 都可以重建回到可以辨識字的程度；若在過程中加 Gaussian Noise，只做 inverse 基本上無法復原到能辨識的程度，但 Weiner filter 可以藉由調整 K 值調整到足夠的影像品質。