影像處理原理及應用 HW6

R11543063　葉乃綸

此次作業目標是設計一介面，供使用者對輸入的影像進行圖像扭轉、影像融合與線條檢測等處理。

一、程式介紹

本次作業共有三個小題。第一小題為對輸入影像進行梯形、波型、圓形等影像扭轉。其中，梯形與圓形扭轉可以透過輸入影像與扭轉影像間的幾何關係，對輸入影像的每一列進行 scaling 求得（圖一）；波行扭轉則透過幾何關係改變輸入影像的 pixel 位置。第二小題是對多個輸入影像進行一次小波轉換，對其低通子通道（Lowpass subband）的簡略係數取平均值，細節子通道（Detail subbands）的細節係數取最大值（圖二），最後將結果進行反轉換以得融合影像。第三小題則是透過 opencv 函式庫所提供的函式對輸入影像進行 Hough Transform，描繪出輸入影像的線條並計算圖形的面積與周長。

```
QImage image(file);
propimage = image.scaled(512, 512, Qt::KeepAspectRatio);

QImage imgbar;
QImage imgbar2;
QImage output(512,512,QImage::Format_RGB32);
int w;

for(int k = 0; k < 512; k++)
{
    w = round(2*sqrt(pow(256,2) - pow((256-k),2)));
    imgbar = propimage.copy(0, k, 512, 1);
    imgbar2 = imgbar.scaled(w, 1, Qt::IgnoreAspectRatio);

    for(int i = 0; i < 512; i++)
    {
        if(i < round(256 - w/2))
            output.setPixel(i, k , qRgb(0, 0, 0));
        else if(i >= round(256 - w/2) && i < round(256 + w/2))
        {
            output.setPixel(i, k , qRgb(qGray(imgbar2.pixel(i - round(256 - w/2), 0)),
                                        qGray(imgbar2.pixel(i - round(256 - w/2), 0)), qGray(imgbar2.pixel(i - round(256 - w/2), 0))));
        }
        else
        {
            output.setPixel(i, k , qRgb(0, 0, 0));
        }
    }

}
```

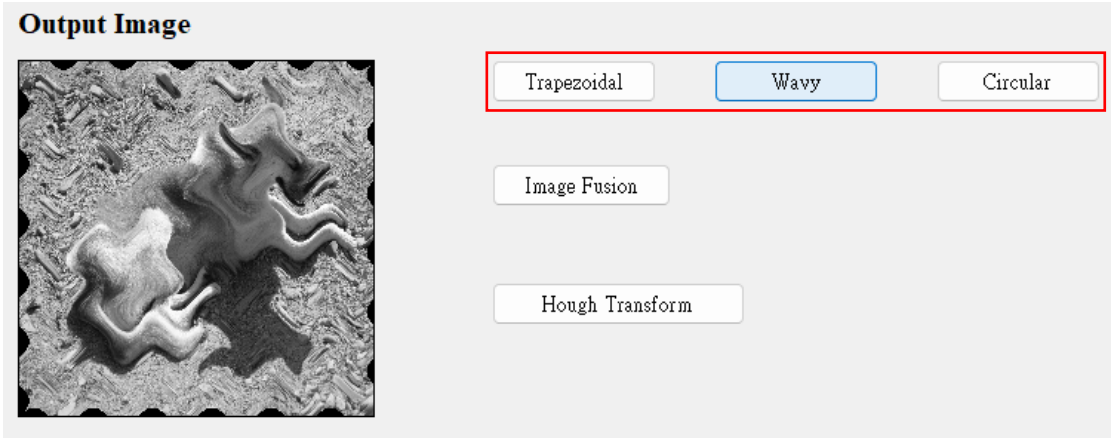
圖一、圓形扭轉影像演算法

```
c_1=(src1.at<float>(2*y,2*x)+src1.at<float>(2*y,2*x+1)+src1.at<float>(2*y+1,2*x)+src1.at<float>(2*y+1,2*x+1))*0.5;
c_2=(src2.at<float>(2*y,2*x)+src2.at<float>(2*y,2*x+1)+src2.at<float>(2*y+1,2*x)+src2.at<float>(2*y+1,2*x+1))*0.5;

dh_1=(src1.at<float>(2*y,2*x)+src1.at<float>(2*y+1,2*x)-src1.at<float>(2*y,2*x+1)-src1.at<float>(2*y+1,2*x+1))*0.5;
dh_2=(src2.at<float>(2*y,2*x)+src2.at<float>(2*y+1,2*x)-src2.at<float>(2*y,2*x+1)-src2.at<float>(2*y+1,2*x+1))*0.5;

dv_1=(src1.at<float>(2*y,2*x)+src1.at<float>(2*y,2*x+1)-src1.at<float>(2*y+1,2*x)-src1.at<float>(2*y+1,2*x+1))*0.5;
dv_2=(src2.at<float>(2*y,2*x)+src2.at<float>(2*y,2*x+1)-src2.at<float>(2*y+1,2*x)-src2.at<float>(2*y+1,2*x+1))*0.5;

dd_1=(src1.at<float>(2*y,2*x)-src1.at<float>(2*y,2*x+1)-src1.at<float>(2*y+1,2*x)+src1.at<float>(2*y+1,2*x+1))*0.5;
dd_2=(src2.at<float>(2*y,2*x)-src2.at<float>(2*y,2*x+1)-src2.at<float>(2*y+1,2*x)+src2.at<float>(2*y+1,2*x+1))*0.5;

if(src3.empty())
{
    c_out = c_1*0.5 + c_2*0.5;
    dh_out = max({dh_1, dh_2});
    dv_out = max({dv_1, dv_2});
    dd_out = max({dd_1, dd_2});
}


else
{
    c_3=(src3.at<float>(2*y,2*x)+src3.at<float>(2*y,2*x+1)+src3.at<float>(2*y+1,2*x)+src3.at<float>(2*y+1,2*x+1))*0.5;
    dh_3=(src3.at<float>(2*y,2*x)+src3.at<float>(2*y+1,2*x)-src3.at<float>(2*y,2*x+1)-src3.at<float>(2*y+1,2*x+1))*0.5;
    dv_3=(src3.at<float>(2*y,2*x)+src3.at<float>(2*y,2*x+1)-src3.at<float>(2*y+1,2*x)-src3.at<float>(2*y+1,2*x+1))*0.5;
    dd_3=(src3.at<float>(2*y,2*x)-src3.at<float>(2*y,2*x+1)-src3.at<float>(2*y+1,2*x)+src3.at<float>(2*y+1,2*x+1))*0.5;


    c_out = c_1*0.5 + c_2*0.5 + c_3*0.5;
    dh_out = max({dh_1, dh_2, dh_3});
    dv_out = max({dv_1, dv_2, dv_3});
    dd_out = max({dd_1, dd_2, dd_3});
}
```

圖二、小波轉換後的係數

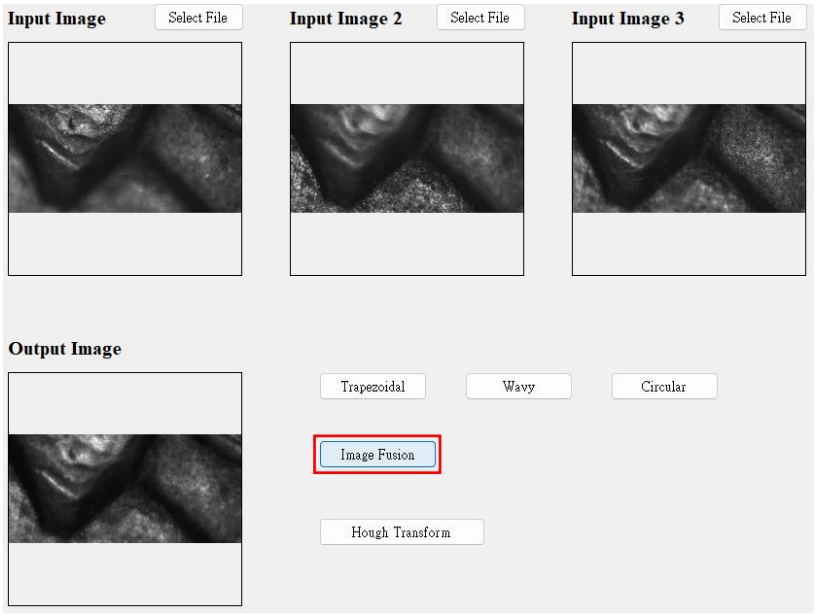
二、UI 介紹

1. 選定輸入影像後，點選介面上的“Trapezoidal”、 “Wavy”、“Circular”等按鈕可分別對輸入影像進行梯形、波型、圓形扭轉，並輸出影像顯示於介面左下角的視窗中（圖三）。



圖三

2. 選定多個輸入影像，點選介面上的“Image Fusion”按鈕可將多個輸入影像進行融合，並輸出於介面左下角的視窗中（圖四）。

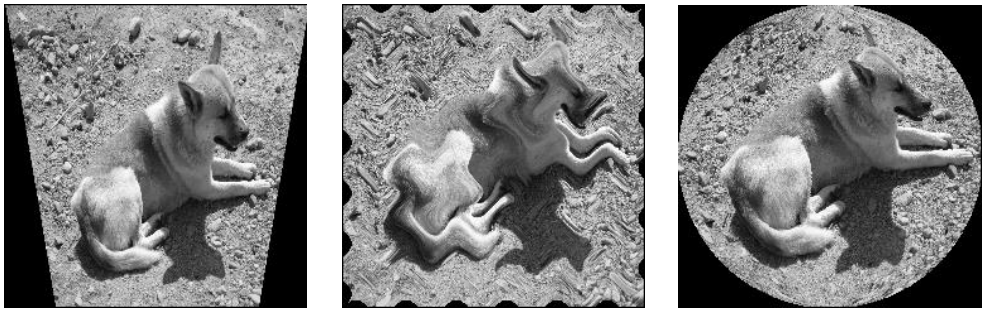


圖四

3. 點選介面上的“Hough Transform”按鈕則可對輸入影像進行線條檢測，並輸出於介面左下角的視窗中。

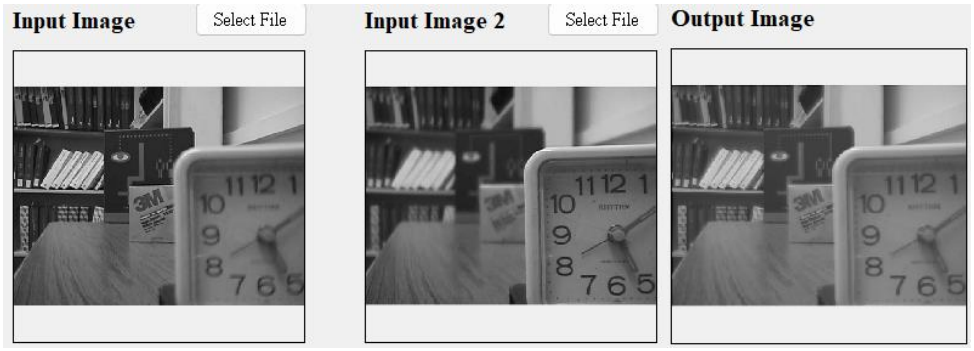
三、結果與討論

1. 給定扭轉影像四個角的座標位置，透過梯形扭轉能把影像轉成任意四邊形；波型扭轉影像能透夠調整參數改變其波型震幅以及頻率；圓形扭轉影像則能夠呈現魚眼鏡頭效果（圖五）。



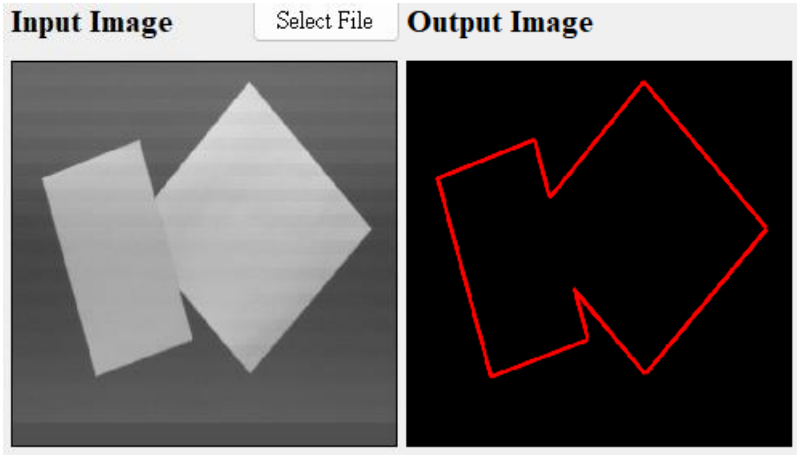
圖五

2. 透過波型轉換取得輸入影像轉換後的各係數，再分別對簡略係數以及細節係數進行取平均值與最大值等運算，能夠融合兩影像的細部於輸出影像中（圖六）。



圖六

3. 透過 Hough Transform 能夠偵測輸入影像的線條，並以紅線框選，顯示於輸出影像中（圖七）。



圖七