影像處理實習課

02_仿射轉換、雙線性內插

助教:莊媞涵

仿射轉換 灰階影像平移

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

如果在邊界內就填入

```
if 0 <= new_x < width and 0 <= new_y < height:
    translated_image[new_y, new_x] = image[y, x]
```

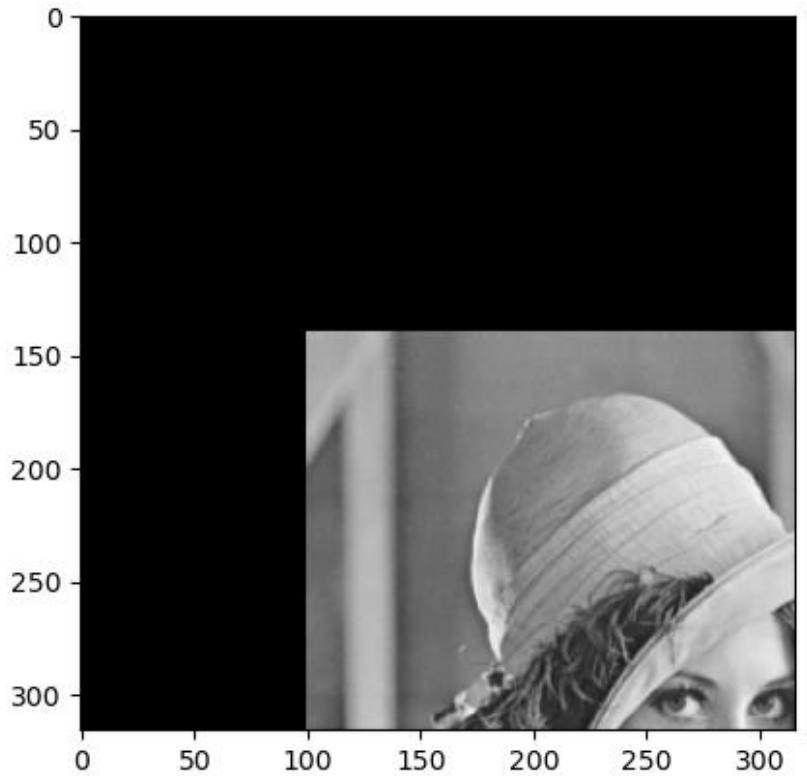
開全黑畫布

```
# 創建一個與原圖大小相同的空陣列，用來存放平移後的圖像
translated_image = np.zeros_like(image)
print(translated_image.shape)  # 輸出平移後圖像的尺寸
```

計算新的x與y的值

```
new_x = x+100
new_y = y+140
```

$$x' = x + t_x$$
$$y' = y + t_y$$



仿射轉換 彩色影像平移

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

如果在邊界內就填入

```
if 0 <= new_x < width and 0 <= new_y < height:
    translated_image[new_y, new_x] = image[y, x]
```

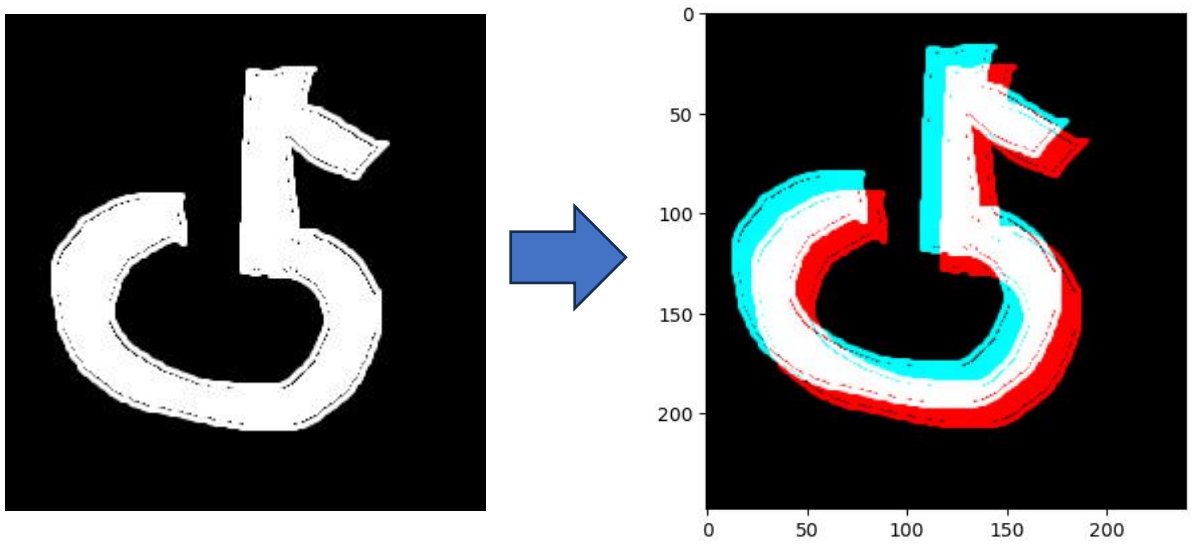
分R、G、B三通道處理

```
for c in range(channl):  # 遍歷每個通道（R、G、B）
    for y in range(height):  # 遍歷圖像的每一行
        for x in range(width):  # 遍歷圖像的每一列
```

只將G、B兩通到往左上移，R留原地

```
if c == 1:  # 處理G通道
    new_x = x-10
    new_y = y-10
```

```
if c == 2:  # 處理B通道
    new_x = x-10
    new_y = y-10
```



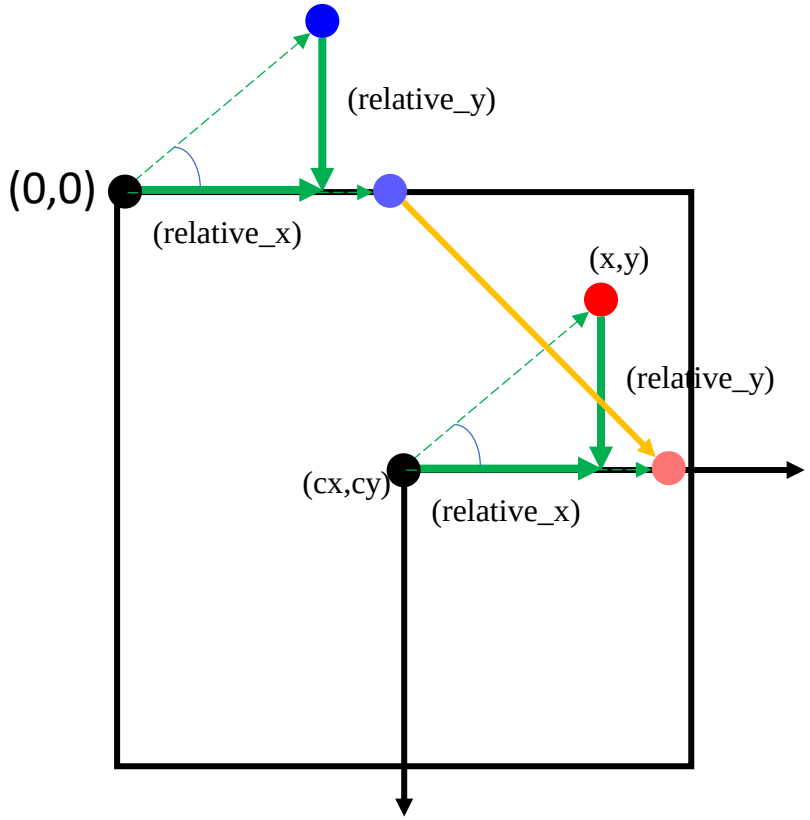
仿射轉換 旋轉矩陣 不填洞

把旋轉中心定在影像中心，找新的x,y

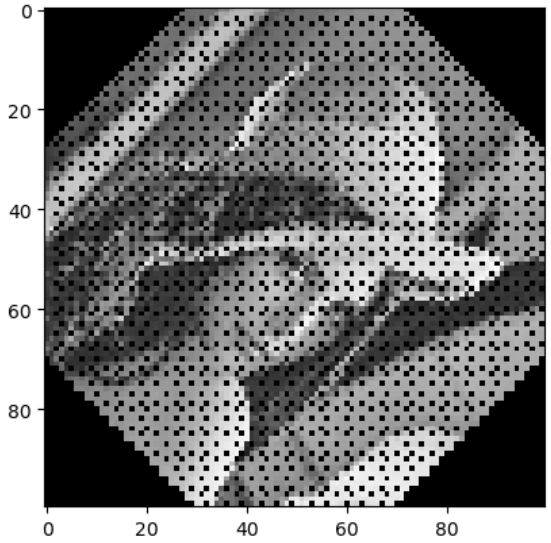
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -sin\,\theta & 0 \\ sin\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

```python
# 計算以中心點為基準的相對坐標
relative_x = x - center_x
relative_y = y - center_y
```

```python
# 旋轉變換公式
new_x = int(cos_theta * relative_x - sin_theta * relative_y + center_x)
new_y = int(sin_theta * relative_x + cos_theta * relative_y + center_y)
```



這邊做完只會是基於(0,0)那個點做完的x',y'，
因為我們要的是基於(cx,cy)去轉
所以要加上(cx,cy)的位移量



仿射轉換 旋轉矩陣 最鄰近插值

使用旋轉的反矩陣，並且向下取整找座標

```
# 計算相對於旋轉中心的新坐標
relative_x = new_x - center_x
relative_y = new_y - center_y

# 反向映射公式：計算該點在原圖中的位置
original_x = int(cos_theta * relative_x + sin_theta * relative_y + center_x)
original_y = int(-sin_theta * relative_x + cos_theta * relative_y + center_y)
```

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -sin\,\theta & 0 \\ sin\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & sin\,\theta & 0 \\ -sin\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$ 反矩陣

仿射轉換 旋轉矩陣 雙線性內插

使用旋轉的反矩陣，找位於原圖之浮點數座標

```
# 反向映射公式：計算該點在原圖中的浮點位置
original_x = cos_theta * relative_x + sin_theta * relative_y + center_x
original_y = -sin_theta * relative_x + cos_theta * relative_y + center_y
```

鄰近四點

```
x0, y0 = int(original_x), int(original_y)
x1, y1 = x0 + 1, y0 + 1
```

雙線性內插

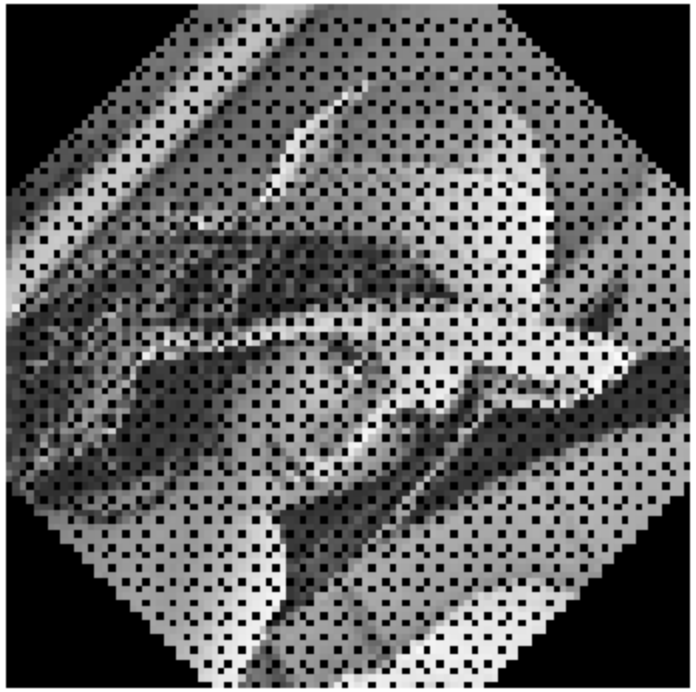
```
interpolated_value = (f00 * (1 - dx) * (1 - dy) +
                      f10 * dx * (1 - dy) +
                      f01 * (1 - dx) * dy +
                      f11 * dx * dy)

Bilinear_rotated_image[new_y, new_x] = int(interpolated_value)
```

Rotated Image (45 degrees with Bilinear Interpolation)

仿射轉換 旋轉矩陣 比較

rotated_image



Nearest Neighbor

Bilinear Interpolation

HW_練習 旋轉之後平移

自己整合程式 搞出 旋轉45度後 x,y 各往右下平移25