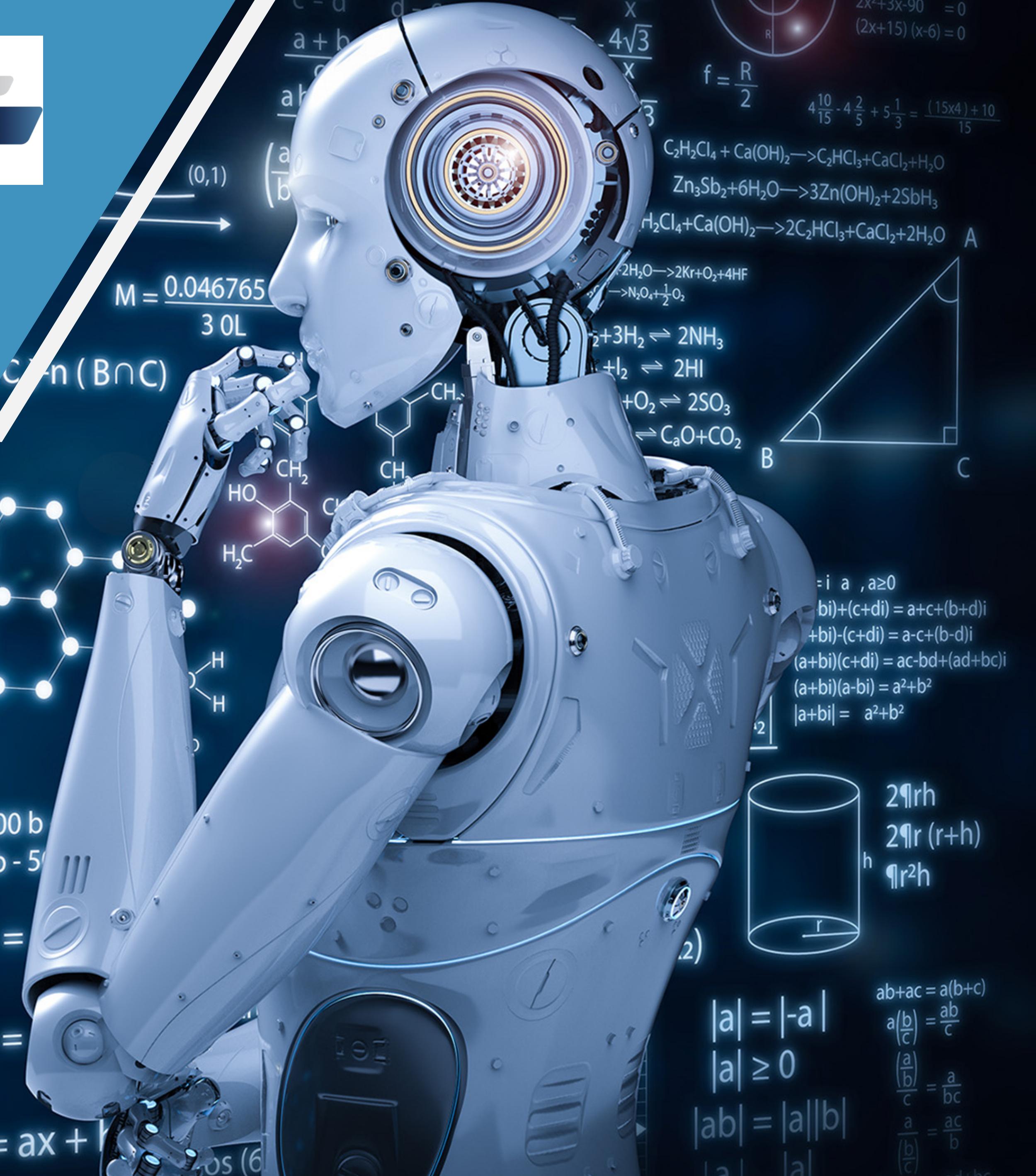




Day 04

深度學習與電腦視覺 學習馬拉松

Cupay 陪跑專家：楊鎮銘



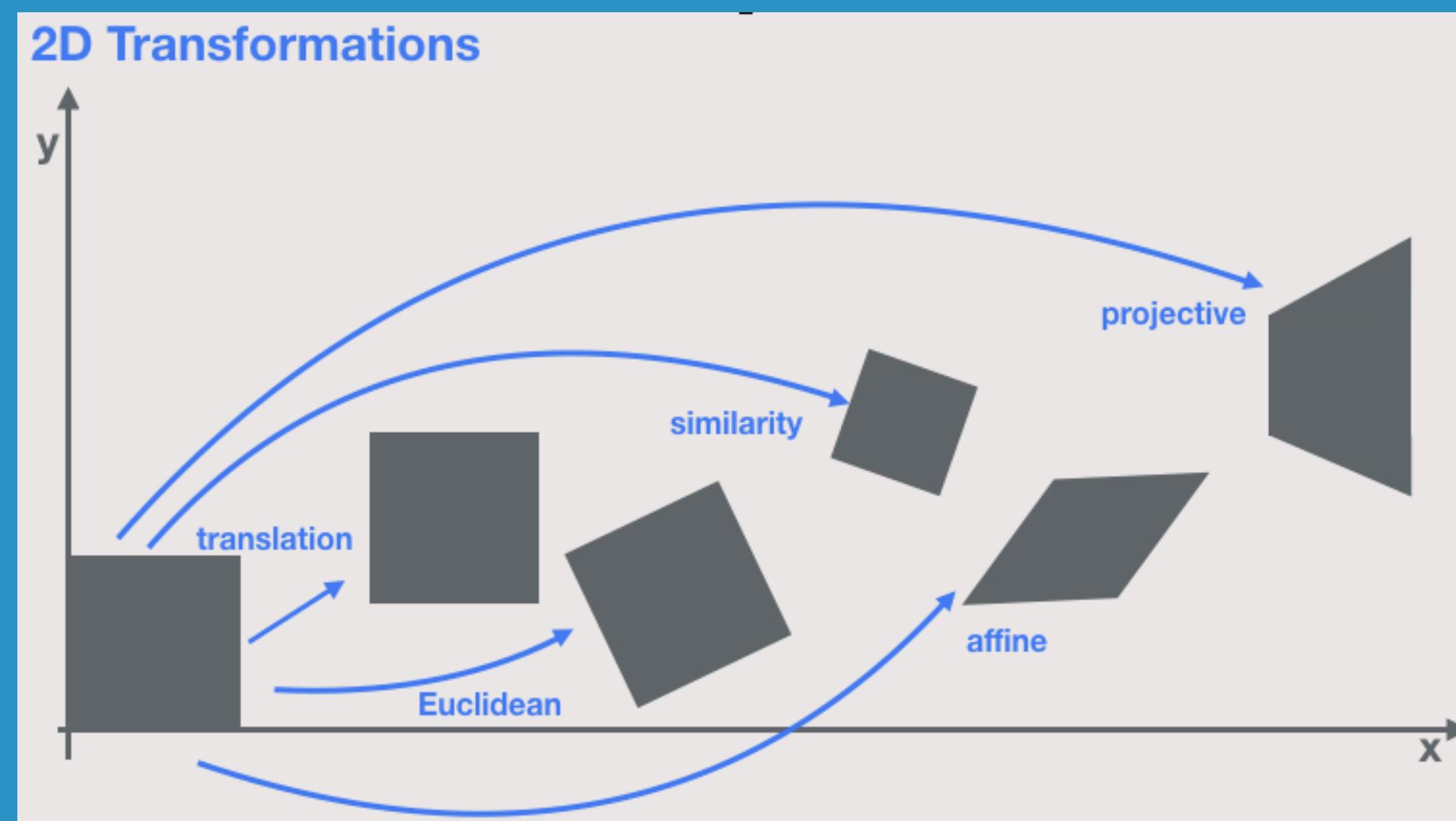


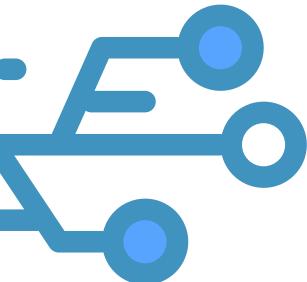
基礎影像處理

翻轉、縮放與平移幾何轉換

重要知識點

- 了解翻轉與縮放的操作
- 了解 Transformation 的概念與平移操作





幾何轉換對於電腦視覺的意義

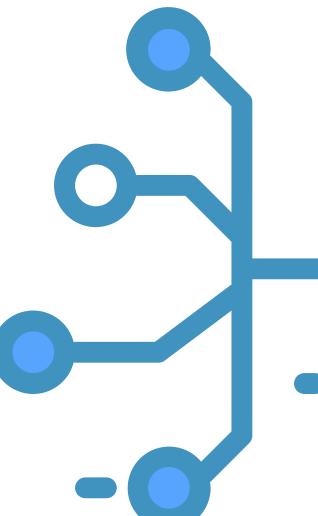


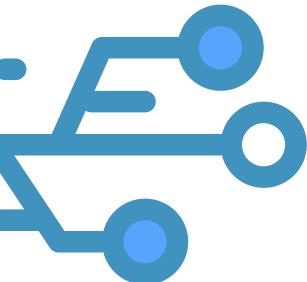
除了增加資料量的 Image Augmentation 以外
幾何轉換的概念也經常應用在架構上的設計或是前處理與後處理

- 透過縮放做 upsample / downsample
- 透過 affine transform 做匹配處理
- 透過 perspective transform 做對齊處理



這邊現階段不清楚沒關係，未來如果遇到電腦視覺相關問題
至少會記得還有這些選項跟操作可以試試看。





水平與垂直翻轉 (Flip)

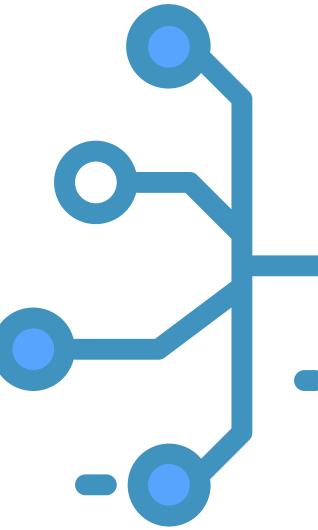


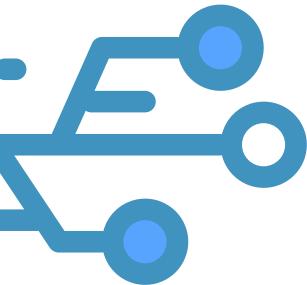
水平與垂直翻轉其實只是 xy 軸順序顛倒

這邊介紹 Python 一個特別的操作

```
list[<start>:<end>:<step>]
```

概念可以參考 Python 的 range
代表從 start 到 end 每次走 step 步





水平與垂直翻轉 (Flip)

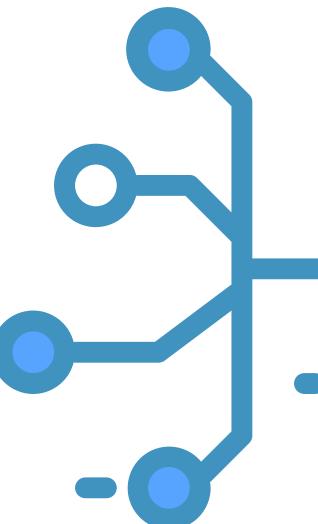


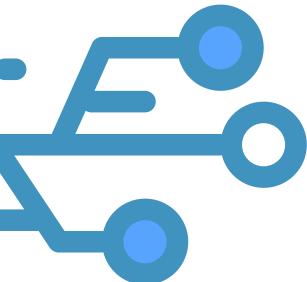
垂直翻轉代表 x 軸的值順序顛倒
用前面提到的表示方式可以變為

```
    h   w   c  
img[::-1, :, :] # 垂直翻轉  
      y   x
```

::-1代表從 end 走到 start (倒序)

同樣的方式對 x 軸處理就會變成水平翻轉





縮放操作 (Scale) - OpenCV



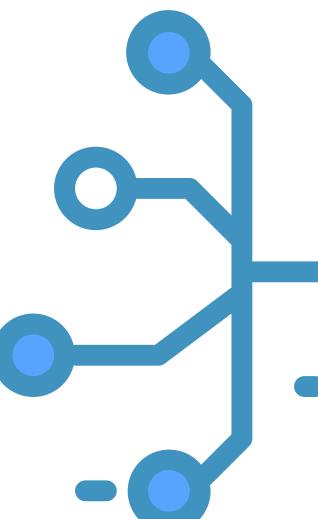
放大與縮小都可以透過 OpenCV 的 function 操作

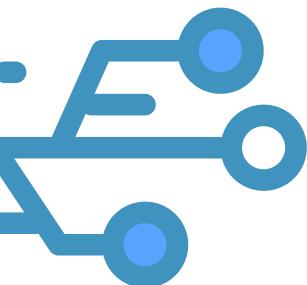
```
cv2.resize(img, new_img_shape, fx, fy, interpolation)
```

輸出的圖片大小 ←—————— fx, fy →—————— 處理方式

↓

選擇其中一種方式表示 ←——————



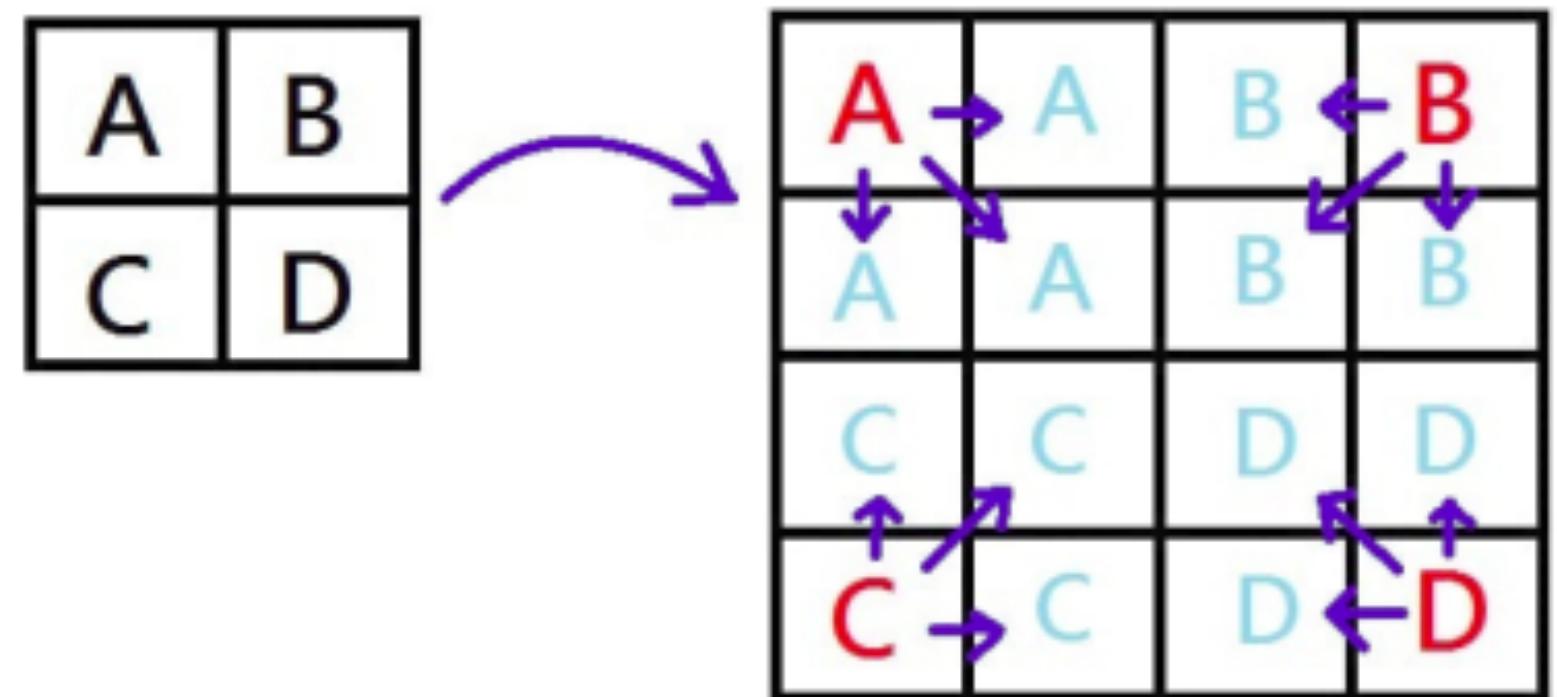


縮放操作 (Scale) - OpenCV

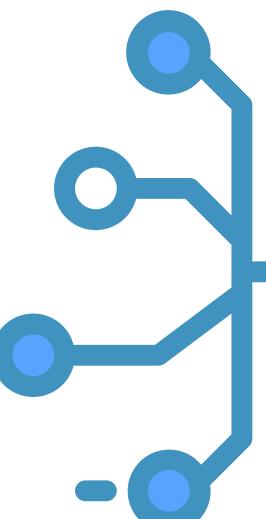


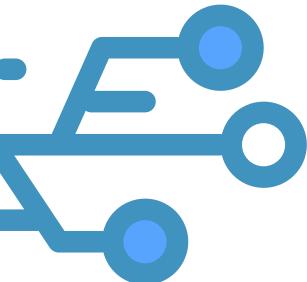
其中縮放操作的方式 (interpolation) 會影響處理的速度與圖片品質：

- cv2.INTER_NEAREST: 最鄰近插點法。
- cv2.INTER_LINEAR: 雙線性插補(預設)。
- cv2.INTER_AREA: 臨域像素再取樣插補。
- cv2.INTER_CUBIC: 雙立方插補， 4×4 大小的補點。
- cv2.INTER_LANCZOS4: Lanczos插補， 8×8 大小的補點



因為縮小跟放大要參考周圍的 pixel 值
經過統計與運算去減少 / 生成新的 pixel 值





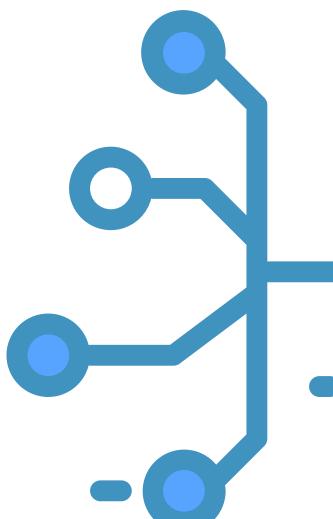
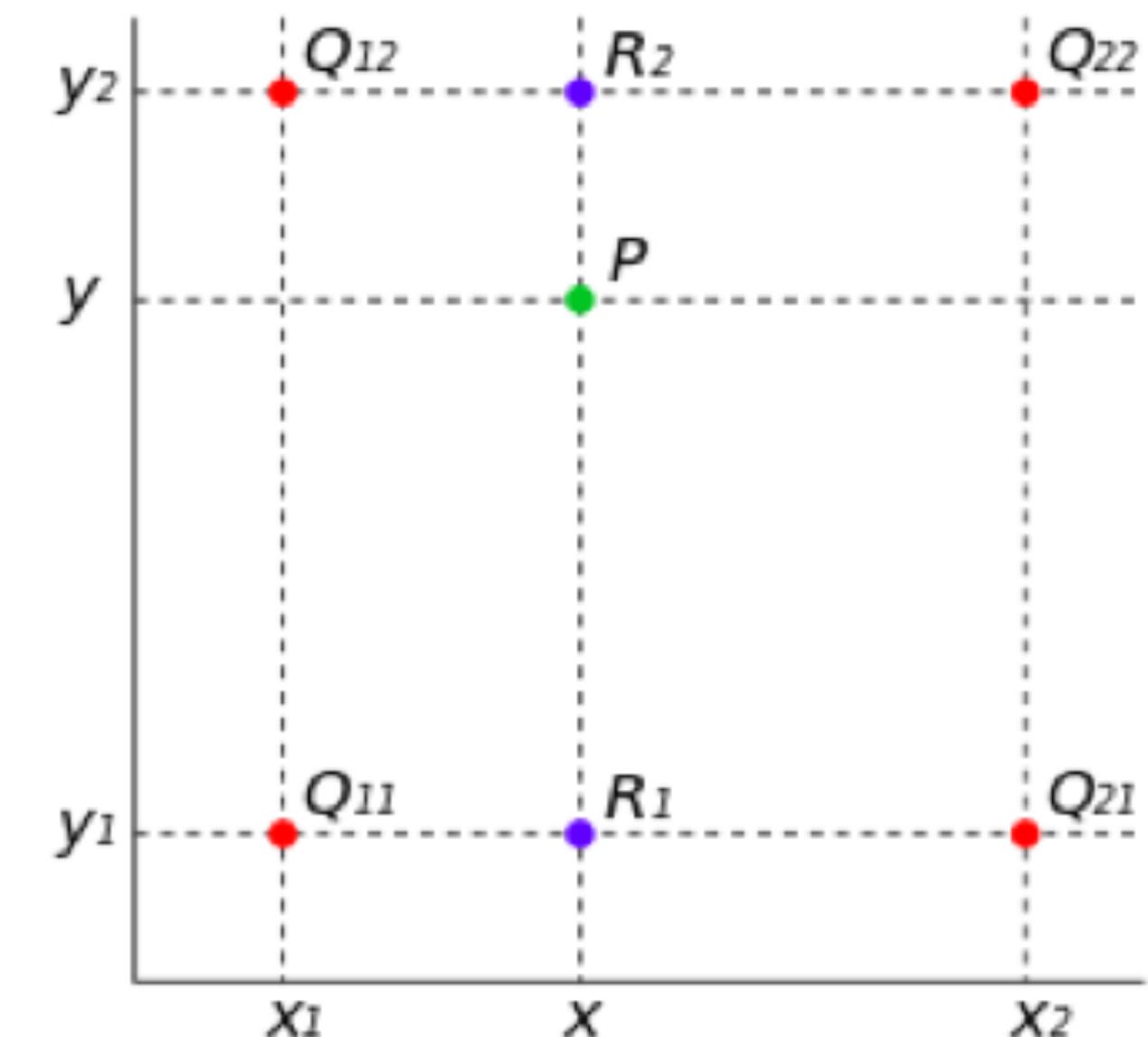
縮放操作 (Scale) - 雙線性插補

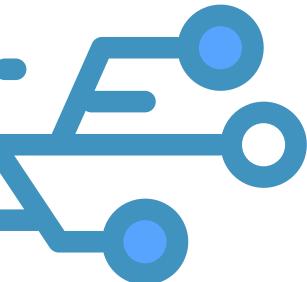


其中預設操作的雙線性插補 (Bilinear Interpolation) 在電腦視覺領域是特別常用的方法，這邊進一步詳細介紹

像素是離散值，但是雙線性會參考周遭 4 個點根據目標位置到 4 個點的距離當作權重這樣處理的圖片比較不會產生鋸齒狀邊緣

可以參考 [Reference](#) 詳細了解運算過程





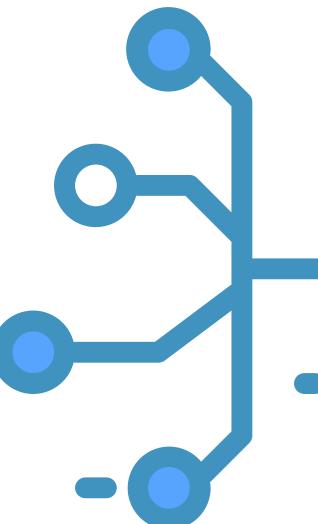
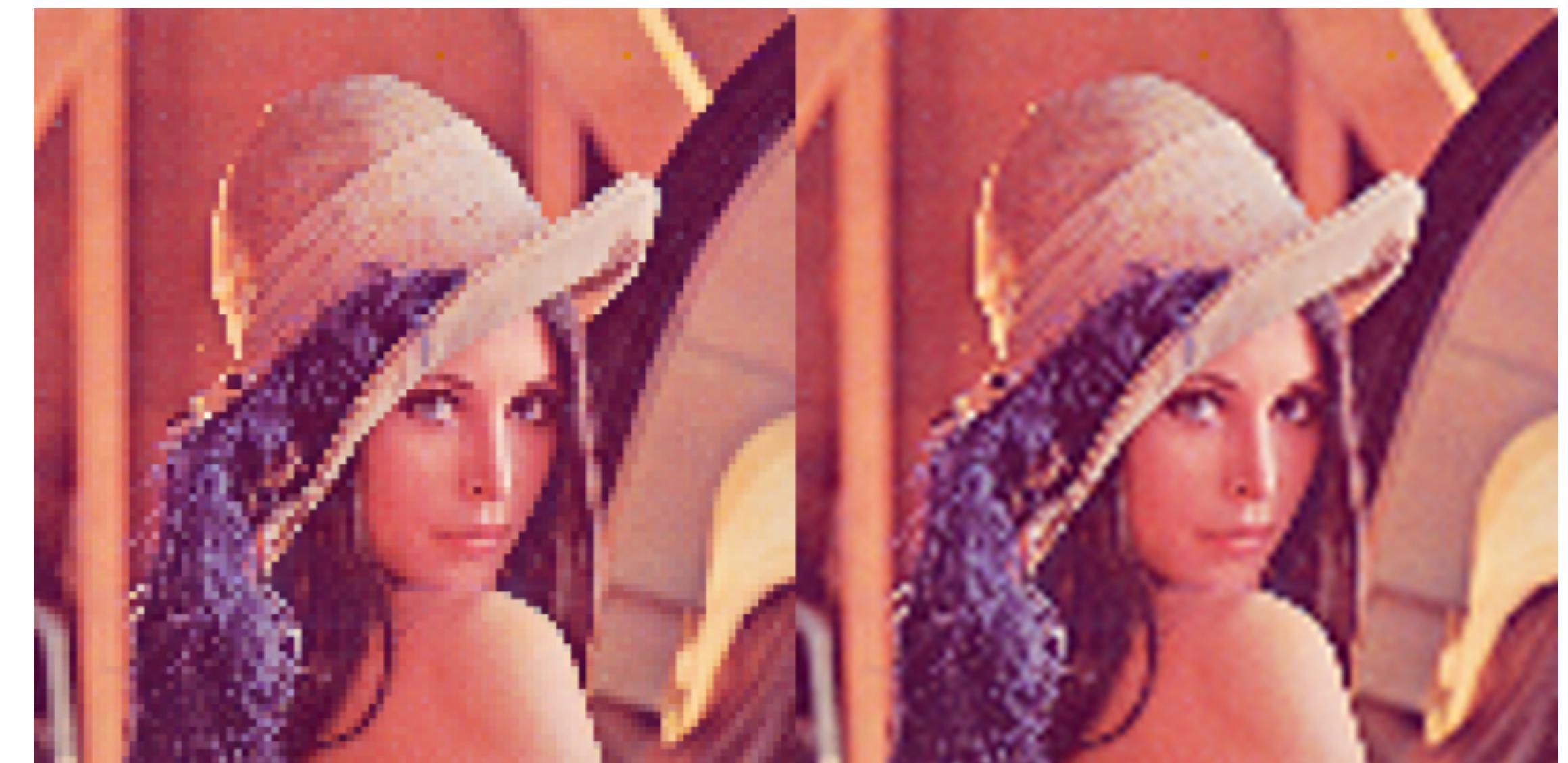
縮放操作 (Scale)

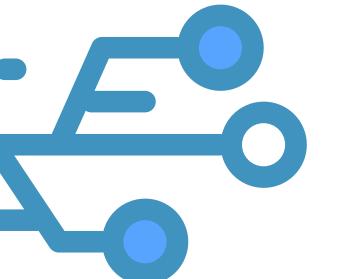


- 預設縮放皆 INTER_LINEAR
- 建議縮小用 INTER_AREA
- 建議放大用 INTER_CUBIC (slow) 或是 INTER_LINEAR

INTER_AREA vs. INTER_CUBIC
放大操作的圖片品質變化

可以明顯看到 INTER_AREA
較模糊且有鋸齒邊緣 (左)



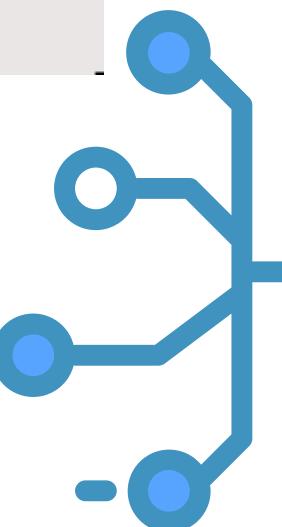
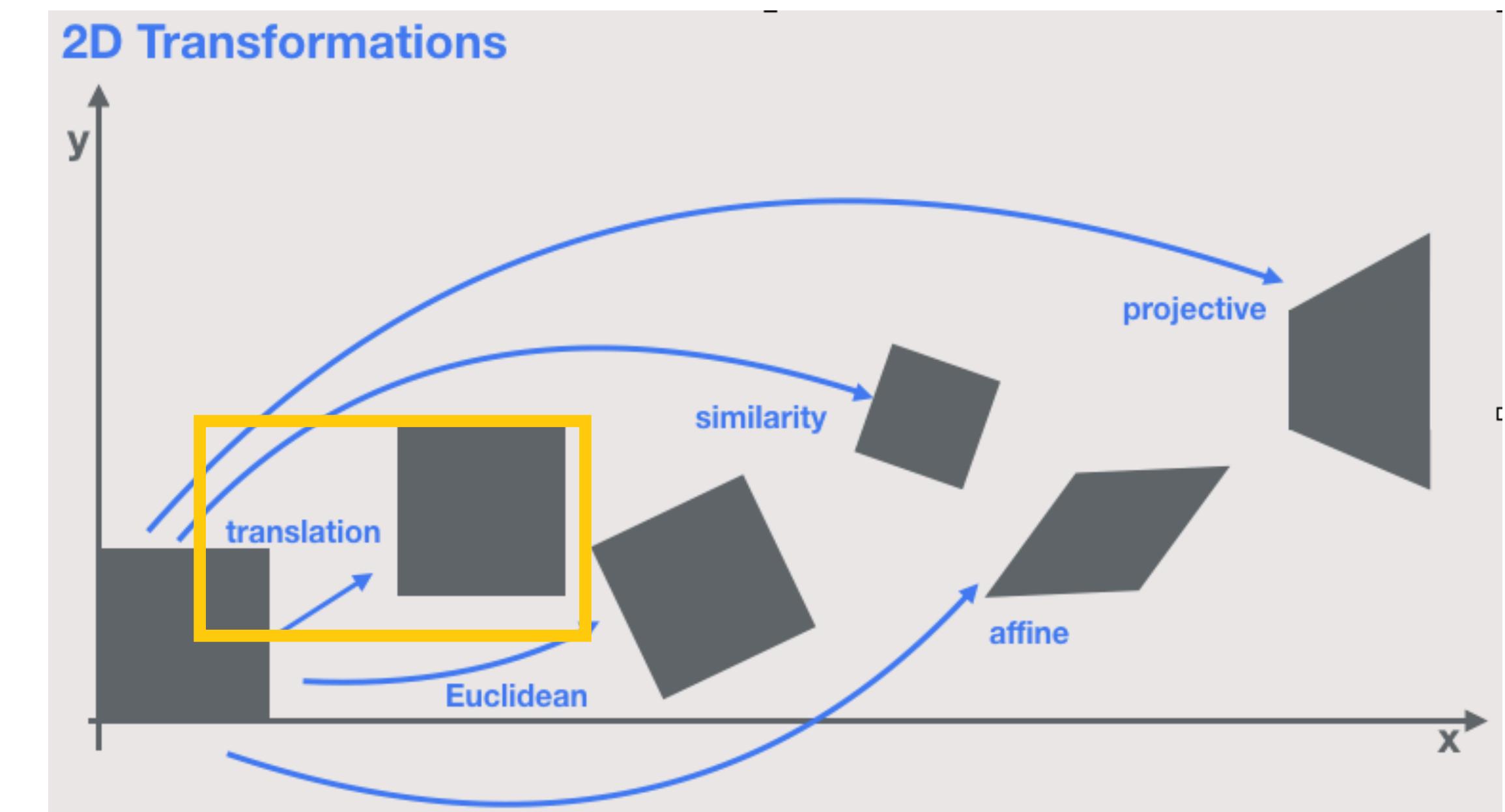


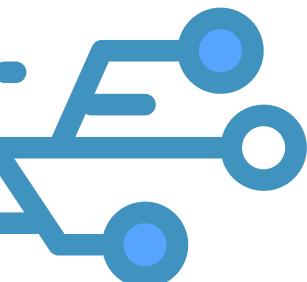
平移操作 (Translation Transformation)



所謂的平移操作的意義為物體往某個向量方向移動
但是其形狀，結構與視角都不發生改變

- 方法一：手動做 xy 軸的四則運算
取得移動後的位置 (慢)
- 方法二：以矩陣運算方式操作 (快)
 - 圖片與 Transformation Matrix 相乘
 - 一次性操作就可以得到平移後的值





Transformation 概念

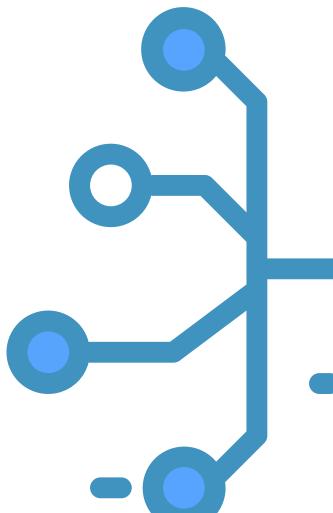


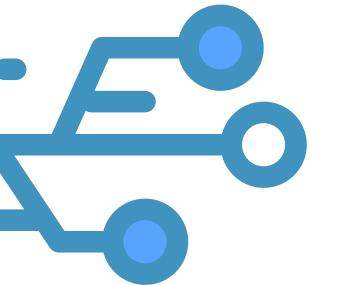
圖片是矩陣，而 Transformation 是圖片經過一連串操作產生新圖，也是產生一個新的矩陣，所以通常設計成矩陣相乘運算來實現
除了原圖外會另外設計 Transformation Matrix 包含了一連串的操作

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ i \end{bmatrix} \rightarrow \text{新的Pixel位置}$$

Transformation Matrix 通式
設計 a~f 不同的參數代表不同的幾何轉換

以齊次座標的方式表示 (x, y) 點，
這邊可以先了解成 z 軸 = 1，後面章節會再介紹





平移操作 (Translation Transformation)

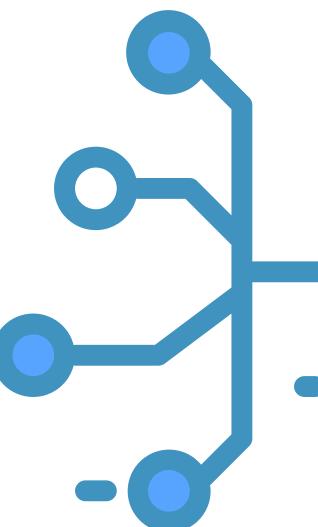


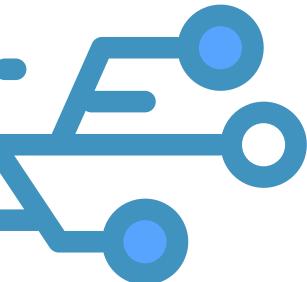
平移操作我們知道會沿著一個向量移動

$$x' = ax+cy+e; \quad y' = bx+dy+f$$

- x, y 軸的移動不會考慮 y, x 軸的值 : $c = b = 0$
- x, y 軸的移動不會做 scale : $a = d = 1$
- $e = x$ 軸移動多少 pixel
- $f = y$ 軸移動多少 pixel

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ i \end{bmatrix}$$





平移操作 (Translation Transformation)

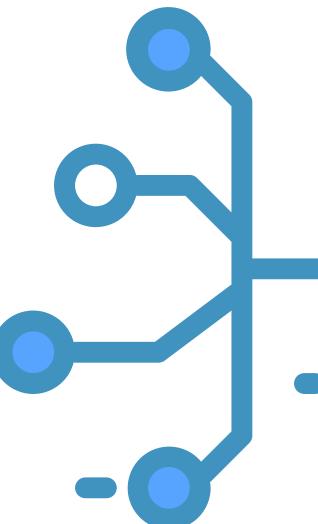


假設我們把 x, y 的移動設為 t_x, t_y

Translation Transformation 的矩陣操作會變為下面的式子

$$\begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ i \end{bmatrix} \rightarrow \begin{array}{l} x \text{ 軸移動 } t_x \text{ 個位置} \\ y \text{ 軸移動 } t_y \text{ 個位置} \end{array}$$

Translation Transformation Matrix





平移操作 (Translation Transformation)

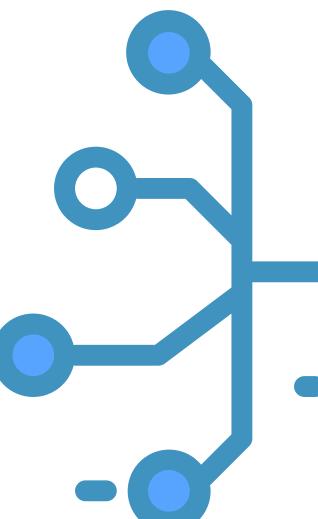


OpenCV 實作上有提供一個廣泛的變換函數

```
cv2.warpAffine(img, Matrix, (column, row))
```

當我們給定不一樣的 Matrix
就可以做不一樣的 Transformation

- 透過 np.array 產生平移矩陣
- 透過 warpAffine 產生平移後圖片

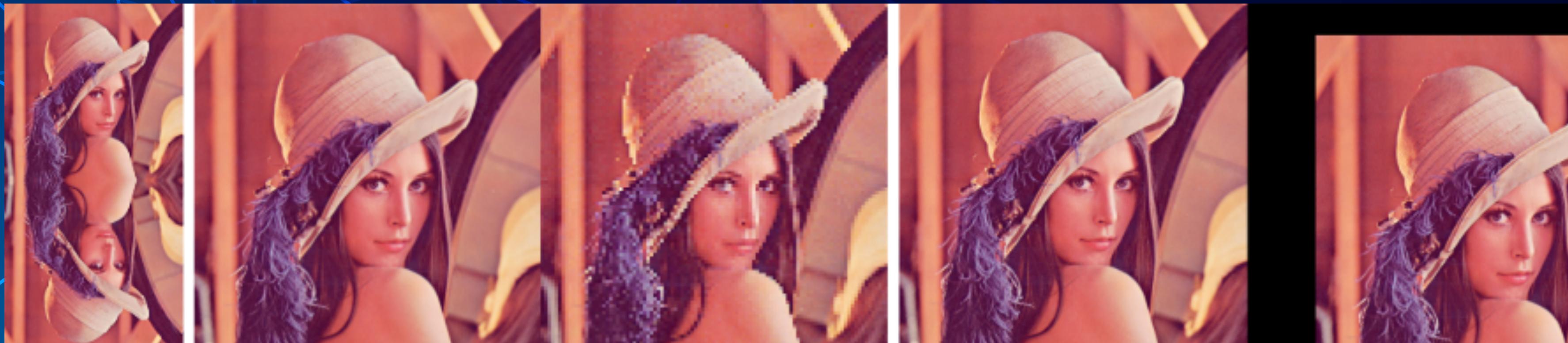


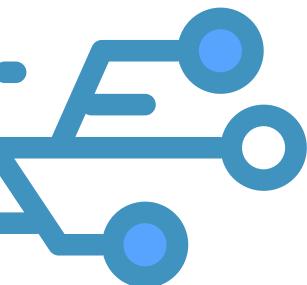
知識點回顧

- 了解並加深 x, y 軸的操作來實作翻轉
- 至少了解縮放中常用的雙線性插值的定義
 - * 比較不會產生鋸齒狀的邊緣
 - * 圖片品質還可以，速度較快，為預設的 interpolation
- 了解 Transformation 的概念與 Transformation Matrix 的設計
 - * 進一步理解以矩陣操作的平移

範例

- 實作本篇提到的三大概念：
 - 翻轉：實作上下翻轉
 - 縮放：實作鄰近差值
 - 平移：建立 Translation Transformation Matrix 來做平移
 - $x + 100, y + 50$





推薦延伸閱讀



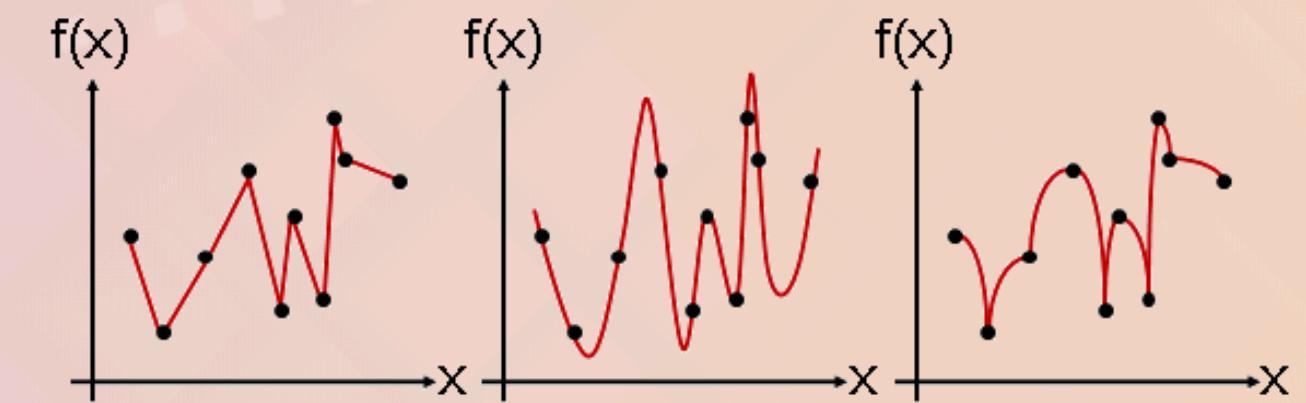
插值演算法 (interpolation) [連結](#)

參考網頁中的 Piecewise Polynomial Interpolation 可以了解在縮放過程中其他插值的詳細數學方式以及比較，但相對的比較不好理解 (optional)

Interpolation

Interpolation

「內插」就是找一個函數，完全符合手邊的一堆函數點。此函數稱作「內插函數」。

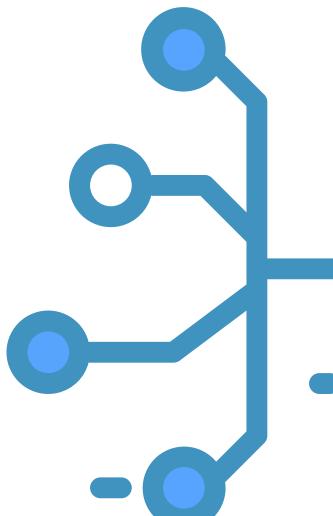
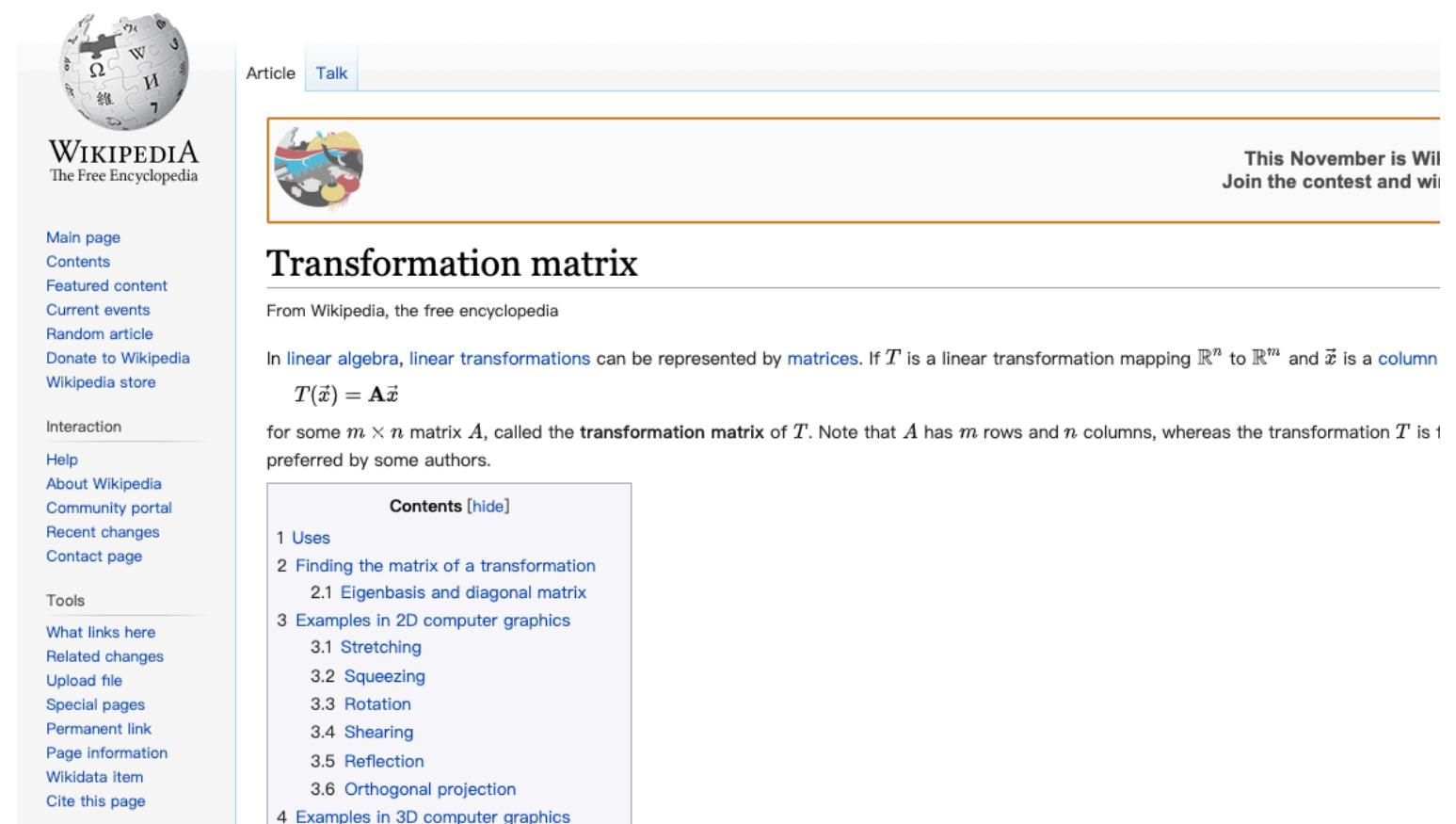


換句話說，找到一個函數，穿過所有給定的函數點。外觀就像是在相鄰的函數點之間，插滿函數點，因而得名「內插」。

方便起見，以下用座標表示函數點。

Transformation Matrix 數學 (wiki) [連結](#)

對於數學有興趣的同學可以再進一步看文件學習，
wiki 上面列了更多種在齊次座標 (Homogenous coordinate) 中使用的 Transformation Matrix



解題時間 Let's Crack It



請跳出 PDF 至官網 Sample Code & 作業開始解題