R13945031 游心妤
程式執行的環境：Python==3.11.5 (VS Code)
使用的套件及版本：
- Pillow==8.2.0
- matplotlib==3.7.2
- numpy==1.24.3
- opencv-python== 4.9.0.80
- scikit-image== 0.20.0

EX1

1. code

```
7   def scale_image(image, scale_factor, interpolation_method):
8       scaled_image = cv2.resize(image, None, fx=scale_factor,
        fy=scale_factor, interpolation=interpolation_method)
9       return scaled_image
```

自定義的 function: scale_image(原始圖像,縮放倍率,計算內積的方法)

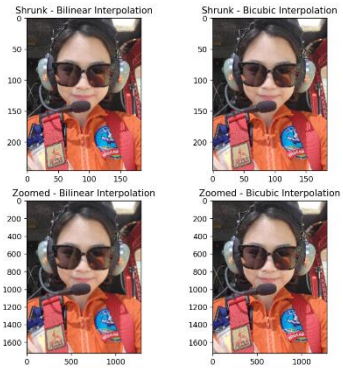
```
24  bilinear_zoomed = scale_image(bilinear_shrunk, zoom_factor, cv2.
    INTER_LINEAR)
25  bicubic_zoomed = scale_image(bicubic_shrunk, zoom_factor, cv2.
    INTER_CUBIC)
```

- Bilinear interpolation: 使用 OpenCV 的 cv2.INTER_LINEAR
- Bicubic interpolation: 使用 OpenCV 的 cv2. INTER_CUBIC

2. steps

i. 載入一張 512x512 以上尺寸的自拍照並命名為 origin.jpg。
ii. 在 bash 打 python ex1.py 執行(接著為執行內容說明)
iii. 使用 Bilinear interpolation 和 Bicubic interpolation 縮放圖片，先各自縮小至 12% (bilinear_shrunk 及 bicubic_shrunk)。
iv. 再將縮小的圖片放大 7 倍，回到接近原圖大小(bilinear_zoomed 及 bicubic_zoomed)。

v. 將結果圖片進行對比，並輸出處理後的圖片。



3. Compare the Quality

Bilinear interpolation:

- 在圖像縮小時，由於僅使用相鄰的四個像素進行插值，因此可能會出現些微模糊，導致細節損失。
- 在圖像放大時，可能會出現鋸齒狀邊緣，細節相對較少，顏色平滑過渡不如 bicubic interpolation 自然，整體效果可能顯得粗糙。

bicubic interpolation:

- 在縮小圖像時，由於考慮了周圍十六個像素，圖像通常更平滑，並且細節保留相較 Bilinear interpolation 更佳，特別是在邊緣和顏色漸變上，可從衣服的陰影處及衣服上的英文字母觀察到。
- 在放大圖像時，圖像更為平滑，細節相對豐富，邊緣更加柔和，色彩過渡自然，在左下角的英文字母 ICP 上可以明顯觀察出。


Bilinear interpolation


bicubic interpolation

4-1. Explanation of Bicubic Interpolation

- 周圍像素的考慮：考慮到目標像素周圍的 16 個鄰近像素，這些像素在水平和垂直方向上各有 4 個（形成 4x4 的網格）。在計算新的像素值時，這 16 個像素的強度（亮度或顏色值）會被用來計算目標像素。
- 使用立方函數：利用立方多項式來擬合這 16 個像素的強度值。計算一個平滑的曲線，這條曲線通過這些周圍像素，以此來確定新像素的值。使得插值過程不僅考慮最近的像素，還會依不同比例值考慮到較遠的像素，從而提高了圖像的平滑度和自然度，減少鋸齒狀邊緣的出現。

4-2. Comparing the computational complexity of bicubic and bilinear interpolation:

- Bilinear Interpolation 的計算複雜度較低 O(n^2)，因為它僅需要考慮四個相鄰像素 (2x2)，計算速度較快，適合在對速度要求較高的應用場景中使用。
- Bicubic Interpolation 的計算複雜度較高 O(n^3)，因為需要考慮十六個像素的值 (4x4)，這使得其運算時間相對較長，但能夠提供更優質的圖像效果，特別是在大幅度縮放時，效果更為明顯。

EX2

1. difference between barrel and pincushion distortion

- barrel distortion: 使得影像看起來從中心向外隆起，直線在遠離影像中心的地方會向外彎曲，類似於桶的形狀，隨著距離中心的增加，放大率增大，在 Brown's Conrady Model 中，barrel distortion 通常與負的徑向畸變係數相關，這意味著實際影像點比理想鏡頭預測的更接近光學中心，四周的扭曲較為明顯，如耳機、肩帶等物體在邊緣處顯得變寬且向外彎曲。
- Pincushion Distortion: 使得影像看起來向內凹陷，直線在遠離影像中心的地方會向內彎曲，類似於針墊的形狀，隨著距離中心的增加，放大率減小，在 Brown's Conrady Model 中，Pincushion Distortion 與正的徑向畸變係數相關，這意味著實際影像點比理想鏡頭預測的更遠離光學中心，結果圖片可知照片四周呈現向內拉扯凹陷狀，尤其是在邊緣處，直線向內彎曲，形成收縮的效果，人物面部相對於 barrel distortion 有所壓縮，耳機、肩帶在邊緣處顯得變窄且向內彎曲。

- 主要差異
    - 彎曲方向：barrel distortion 向外彎曲，而 Pincushion Distortion 向內彎曲。
    - 放大率：barrel distortion 的放大率在邊緣增加，而 Pincushion Distortion 的放大率在邊緣減少。
    - 徑向畸變係數：barrel distortion 有負的係數，而 Pincushion Distortion 有正的係數。

2. code

apply_radial_distortion(原始照片,K1,k2,k3)

k1（主要的係數）：

- 負值時會產生 barrel distortion（影像向外膨脹）；正值時，會產生 Pincushion Distortion（影像向內收縮）。
- 畸變的量隨著影像距離光軸的距離平方增長，因此越遠離光軸，畸變越明顯。

k2（輔助係數）：

- 對較遠的點進行更細微的畸變修正。當 k1 不能完全描述畸變時，加入 k2 以更準確地模擬複雜的變形。
- k2 對徑向畸變的影響是以距離光軸的四次方來增長的，這意味著它在離光軸更遠的地方才會有較大的影響。

k3（高階修正項）：

- 用於處理非常精細的畸變問題。它對影像點的徑向變形影響是距離光軸的六次方，影響範圍更為邊緣的區域。

fx 和 fy 代表焦距，cx 和 cy 代表主點（影像中心）

i. **初始化畸變係數**: dist_coeffs = np.array([k1, k2, k3, 0, 0], dtype=np.float32)
在這裡定義了畸變係數的數組，最後兩個參數（通常用於切變和高階畸變）設為 0，因為這段程式碼只關注徑向畸變。

ii. **生成新的矩陣**: new_camera_matrix, roi = cv2.getOptimalNewCameraMatrix(camera_matrix, dist_coeffs, (w, h), 0)
使用 OpenCV 的 getOptimalNewCameraMatrix 函數來計算基於畸變的最佳新相機矩陣。這會幫助在進行畸變校正時保留更多有效的影像區域。

iii. **應用畸變**: distorted_image = cv2.undistort(image, camera_matrix, dist_coeffs, None, new_camera_matrix)
使用 cv2.undistort 函數將畸變應用於影像，並返回畸變後的影像。這裡的 camera_matrix 和 dist_coeffs 用於描述原始影像的相機參數，而 new_camera_matrix 則用於獲得畸變影像的最佳效果。

iv. 返回畸變後的影像

```python
def apply_radial_distortion(image, k1, k2, k3):
    h, w = image.shape[:2]
    # Generate the camera matrix
    fx = w  # focal length in x
    fy = h  # focal length in y
    cx = w / 2  # principal point x
    cy = h / 2  # principal point y

    camera_matrix = np.array([[fx, 0, cx],
                              [0, fy, cy],
                              [0, 0, 1]], dtype=np.float32)

    # Initialize the distortion coefficients
    dist_coeffs = np.array([k1, k2, k3, 0, 0], dtype=np.float32)

    # Generate new camera matrix based on the distortion
    new_camera_matrix, roi = cv2.getOptimalNewCameraMatrix(camera_matrix, dist_coeffs, (w, h), 0)

    # Apply distortion
    distorted_image = cv2.undistort(image, camera_matrix, dist_coeffs, None, new_camera_matrix)

    return distorted_image
```

3. steps

i. 載入一張 512x512 以上尺寸的自拍照並命名為 origin.jpg。

ii. 在 bash 打 python ex2.py 執行(接著為執行內容說明)

iii. 使用 apply_radial_distortion，k1 分別設正(barrel_distorted_image)及負值(pincushion_distorted_image)。

iv. 將結果圖片進行對比，並輸出處理後的圖片 。