

學號：D12922014

姓名：莊謹譽

程式執行環境與使用套件

- 執行環境:
 - Windows 10 專業版
 - Python - 3.11.5
 - Visual Studio Code - 1.82.2
 - Git for Windows - 2.42.0
- 使用套件:
 - opencv-python - 4.8.1.78

For requirements 1 & 2, you need to show

- which function you use or implement

程式使用 opencv-python 所提供的相關函數與參數。

- 函數

cv2.imread	讀取圖檔
cv2.imwrite	寫入圖檔
cv2.resize	圖檔縮放

- 參數

cv2.INTER_LINEAR	對應 Bilinear
cv2.INTER_CUBIC	對應 Bicubic

- how does your program work

程式從 `__name__ == "__main__"` 開始，分別用 `cv2.INTER_LINEAR` 及 `cv2.INTER_CUBIC` 來呼叫自定義的 `scale_by_method()` 函式，之後 `scale_by_method()` 再以此參數呼叫 `cv2.resize()`，達成 0.2x, 5x, 32x 的縮放，最後再用 `cv2.imwrite()` 將結果存起來。

```
# 程式碼進入點
if __name__ == "__main__":
    # 讀取原始圖片
    image = cv2.imread(os.path.abspath(ORIGIN_JPG_PATH))
    # 用 bilinear 的方式，將圖片縮放 0.2x、5x、32x 倍
    scale_by_method(image, cv2.INTER_LINEAR)
    # 用 bicubic 的方式，將圖片縮放 0.2x、5x、32x 倍
    scale_by_method(image, cv2.INTER_CUBIC)
```

```
# 依所選擇的方法 (bilinear or bicubis)，將圖片縮放 0.2x、5x、32x 倍
def scale_by_method(image, method):
    RESULTS_DIR_PATH.mkdir(exist_ok=True)
    prefix = "bilinear" if method == cv2.INTER_LINEAR else "bicubic"
    # 0.2x
    image_shrunked = cv2.resize(image, None, fx=0.2, fy=0.2, interpolation=method)
    path_shrunked = RESULTS_DIR_PATH.joinpath(f"{prefix}_0.2x.png")
    cv2.imwrite(os.path.abspath(path_shrunked), image_shrunked)
    # 5x
    image_5_times = cv2.resize(image, None, fx=5, fy=5, interpolation=method)
    path_5_times = RESULTS_DIR_PATH.joinpath(f"{prefix}_5x.png")
    cv2.imwrite(os.path.abspath(path_5_times), image_5_times)
    # 32x
    image_32_times = cv2.resize(image, None, fx=32, fy=32, interpolation=method)
    path_32_times = RESULTS_DIR_PATH.joinpath(f"{prefix}_32x.png")
    cv2.imwrite(os.path.abspath(path_32_times), image_32_times)
```

- how to use your program

先開啟 **Git Bash**，再切換到 **d12922014** 資料夾，然後依序執行以下指令

```
$ python -m venv .venv
$ source .venv/Scripts/activate
$ pip install opencv-python
$ python code/hw1.py
```

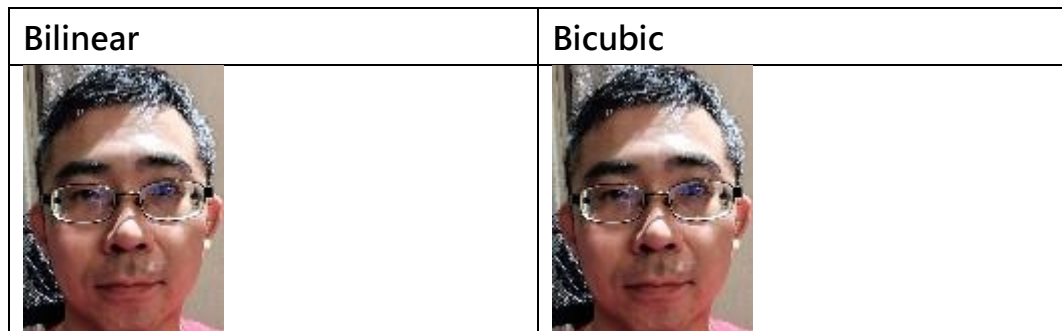
實際執行情況可參考下圖

```
/usr/bin/bash --login -i
cyyc1 ~/Documents/code/ntu-csie-digital-image-processing/hw1/d12922014 hw1*
$ python -m venv .venv
cyyc1 ~/Documents/code/ntu-csie-digital-image-processing/hw1/d12922014 hw1*
$ source .venv/Scripts/activate
cyyc1 ~/Documents/code/ntu-csie-digital-image-processing/hw1/d12922014 [.venv] hw1*
$ pip install opencv-python
Collecting opencv-python
  Obtaining dependency information for opencv-python from https://files.pythonhosted.org/packages/38/d2/3e8c13ffc37ca5ebc6f382b242b44acb43eb489042e1728407ac3904e72f/opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl.metadata
  Using cached opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl.metadata (20 kB)
Collecting numpy>=1.21.2 (from opencv-python)
  Obtaining dependency information for numpy>=1.21.2 from https://files.pythonhosted.org/packages/93/fd/3f826c6d15d3bdcf65b8031e4835c52b7d9c45add25efa2314b53850e1a2/numpy-1.26.0-cp311-cp311-win_amd64.whl.metadata
  Using cached numpy-1.26.0-cp311-cp311-win_amd64.whl.metadata (61 kB)
Using cached opencv_python-4.8.1.78-cp37-abi3-win_amd64.whl (38.1 MB)
Using cached numpy-1.26.0-cp311-cp311-win_amd64.whl (15.8 MB)
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.26.0 opencv-python-4.8.1.78
cyyc1 ~/Documents/code/ntu-csie-digital-image-processing/hw1/d12922014 [.venv] hw1*
$ python code/hw1.py
cyyc1 ~/Documents/code/ntu-csie-digital-image-processing/hw1/d12922014 [.venv] hw1*
$
```

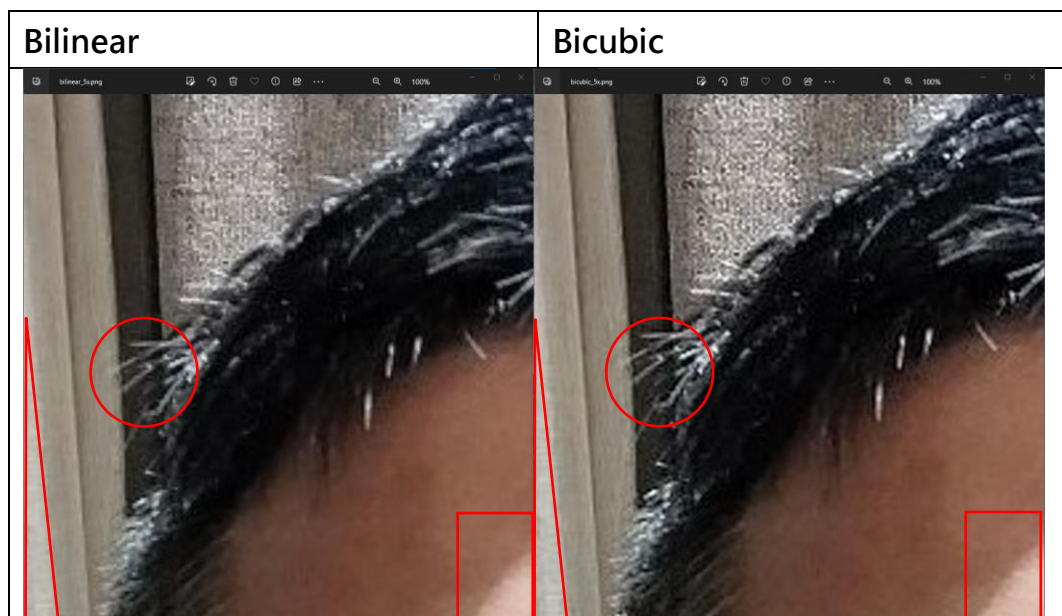
For requirements 3 & 4, you need to provide

- Resulted images for comparison

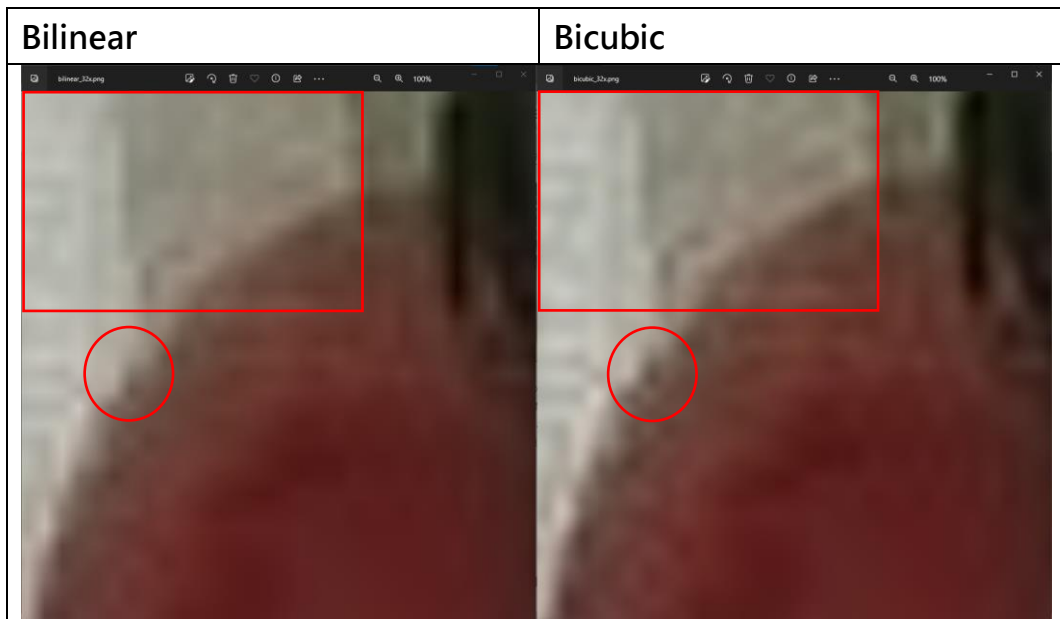
0.2x 的情況下，Bilinear 跟 Bicubic 看不出明顯差異。



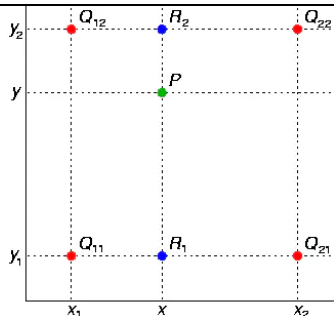
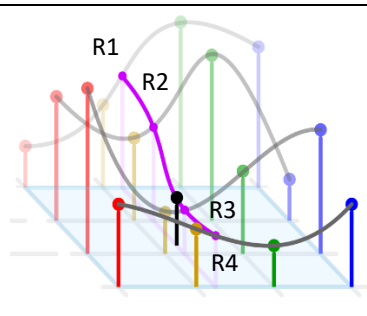
5x 的情況下，觀察左下角背景窗簾光線與右下角額頭反光，會看到 Bicubic 比 Bilinear 來的亮，而且面積比較多。此外，在黑色背景中，Bicubic 的毛髮反射比較明顯，Bilinear 則有點融入在背景中。總結來說，比起 Bilinear，Bicubic 明暗對比較強烈，線條較銳利。



32x 的情況下，相較 Bilinear，Bicubic 在邊界輪廓較明顯，而且非邊界處的色塊，也是 Bicubic 較大較清晰，此外，Bicubic 在背景也是白的地方較白、黑的地方較黑，比較沒有像 Bilinear 糊成一片，而是有比較好的對比。



● Explanation

Bilinear	Bicubic
 <p>先找到包含(x, y)的 2×2 網格，再從網格的 4 個頂點對 x 方向做線性內插(上下各做一次)，得到 $R1$、$R2$，之後再如法炮製，用 $R1$、$R2$ 在 y 方向做線性內插，得到最後結果。</p> <p>由於 Bilinear 只用 2×2 網格上的 4 個頂點，因此，如果圖片放大，要填補的細節過多，會顯得有點模糊，反之縮小就比較沒什麼問題。</p> <p>至於複雜度，由於每個點都需要鄰近 4 個點做 3 次內插，因此 N 個點時間複雜度為 $O(3N) = O(N)$，比 Bicubic 快。</p>	 <p>先找到包含(x, y)的 4×4 網格，再從網格的 16 個頂點，由上而下，分別對 x 方向做三次多項式內插，得到 $R1$、$R2$、$R3$、$R4$，之後再如法炮製，用 $R1$、$R2$、$R3$、$R4$ 在 y 方向做線性內插，得到最後結果。</p> <p>由於 Bicubic 用到 4×4 網格上的 16 個頂點，因此，無論放大縮小，都能比 Bilinear 保留更多細節，讓線條較為銳利，對比較為明顯。</p> <p>至於複雜度，由於每個點都需要鄰近 16 個點做 5 次內插，因此 N 個點時間複雜度為 $O(5N) = O(N)$，比 Bilinear 慢。</p>