MV HW1 Report

資工三 110590004 林奕廷

Requirements

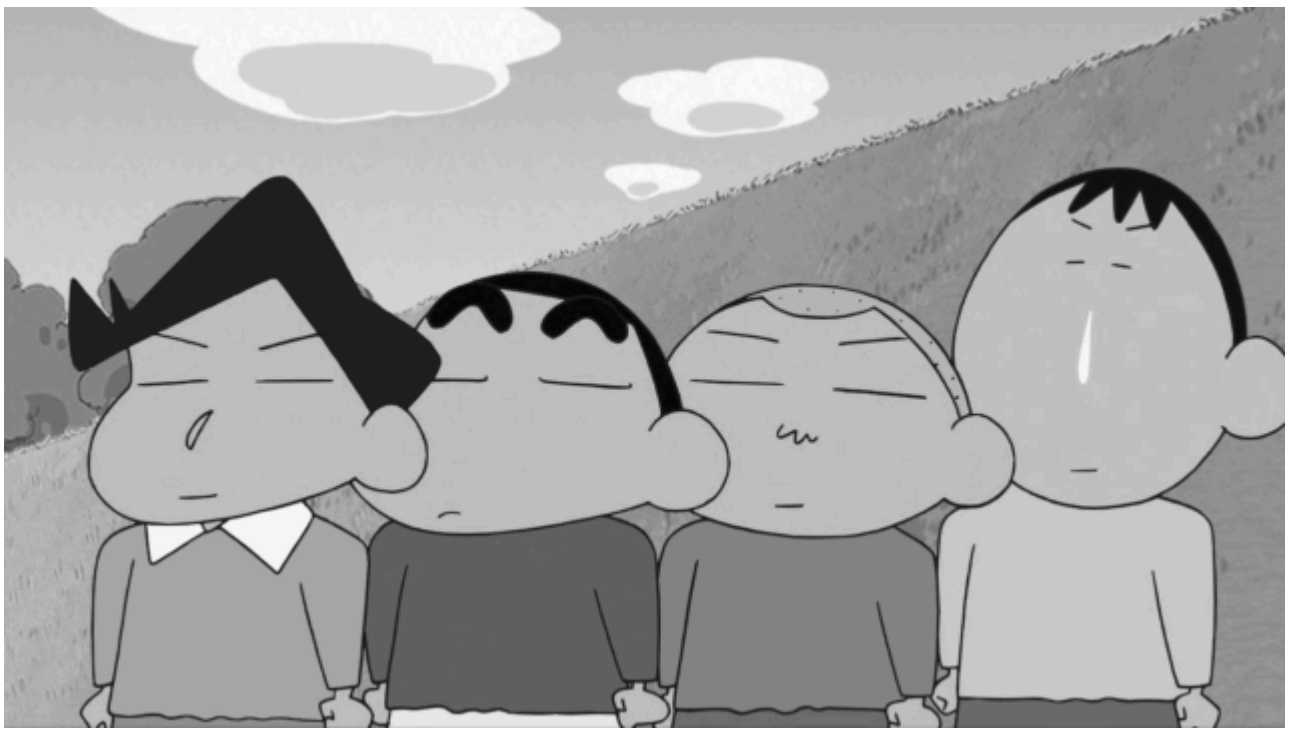
```
python = ">=3.8,<4"
opencv-python = "^4.9.0.80"
alive-progress = "^3.1.5"
```

Usage

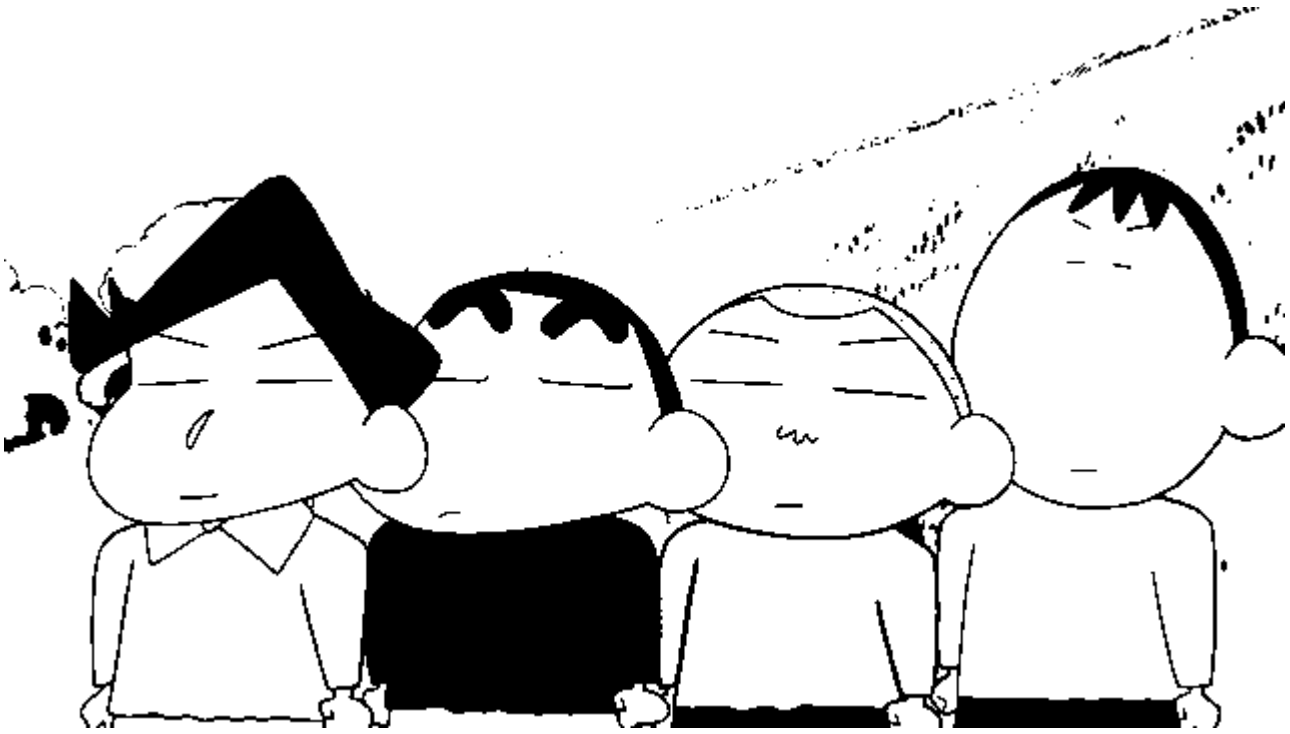
```
python main.py
```

Q1

- P1: 遍歷所有 pixel 並依照給定公式 $(0.3 \times R) + (0.59 \times G) + (0.11 \times B)$ 直接轉換，須注意 OpenCV 使用 BGR 而非 RGB



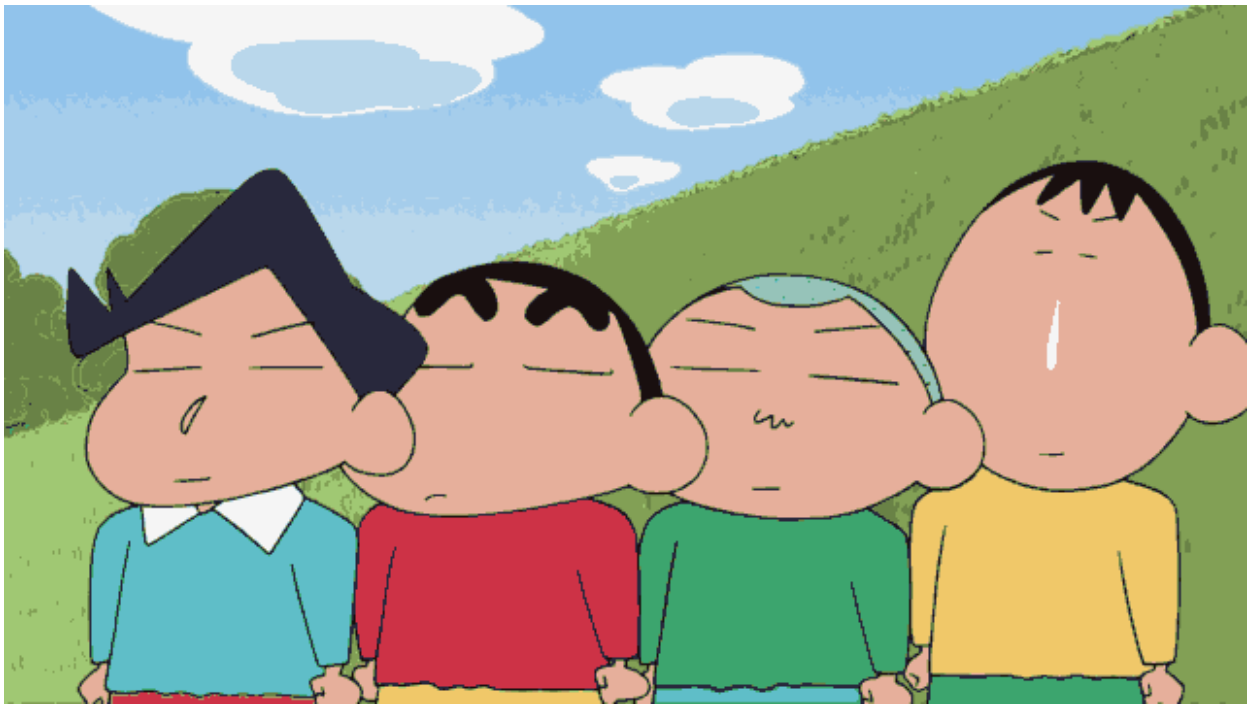
- P2: 使用 P1 產生的圖片並隨意地使用 128 作為閾值，遍歷所有 pixel 進行二值化



- P3:
- Method 1 (abandoned), can bd found in the function `P3A` in `main.py`
    - 以 `DIVS*DIVS` 的尺寸對整張圖片進行採樣，以降低複雜度，須對 pixel 進行轉型否則預設的 `uint8` 在運算過程中會溢位
    - 在採樣的過程中維護一個 `List[ColorPool]`，每個 `ColorPool` 維護一個與 `ColorPool.MainColor` 差距在 `THRESH` 以內的顏色，並記錄各自的出現次數
    - 當某個顏色不屬於 `List[ColorPool]` 中的任一一個 `ColorPool` 創建屬於他自己的 `ColorPool`
    - 遍歷結束後把 `List[ColorPool]` 依照 `Pool.Size` 進行排序，並把 `ColorPool` 的顏色出現次數排序
    - 重複遍歷 `List[ColorPool]` 並從裡面取出出現次數最高的顏色直到拿出 16 個顏色至調色盤
    - 遍歷所有 pixel 並替換成與調色盤最近的顏色
    - 每張圖片各自的 ColorMap
        - img1: #f9dc96 #dbaf56 #200401 #846031 #9c1501 #1e597a #f8e20c #ab0601 #65523e #a69277 #f74f0e #1e2e3d #e0970a #278a06 #607a75 #ede4c4\
        - img2: #dfcdbf #645335 #998678 #e2d2c3 #1d0902 #b57062 #d9c7b9 #635234 #9c897b #e1d1c2 #1a0903 #625133 #938072 #e0d0c1 #2e1b02 #503c21\
        - img3:#729d3d #82a45b #65c3ca #e8b098 #1c1d45 #f7f9f9 #3fad6f #d03149 #f2c66a #6f5548 #755b51 #aa746b #eaf378 #358555 #090909 #752833\
- Method 2
    - 選擇一個好的隨機種子
    - 將圖片中所有 pixel 進行隨機打亂，並遍歷所有的pixel
    - 若所有的調色盤的代表顏色距離給定顏色的距離都大於給定的閾值且調色盤數量不足 16 ，則將該顏色加入調色盤，否則將此顏色加進與其最近的調色盤中，並更新該調色盤的代表色。
    - 遍歷所有原圖的 pixel 並替換成與調色盤最近的顏色
    - 每張圖片各自的 color map
    - img1: #f8dd93 #ddb25a #2e1005 #583117 #740b03 #164370 #ee4209 #e6d9bb #cfa148 #873b13 #1c7e0f #ecc975 #b81c06 #b8a787 #94703a #e8cc05

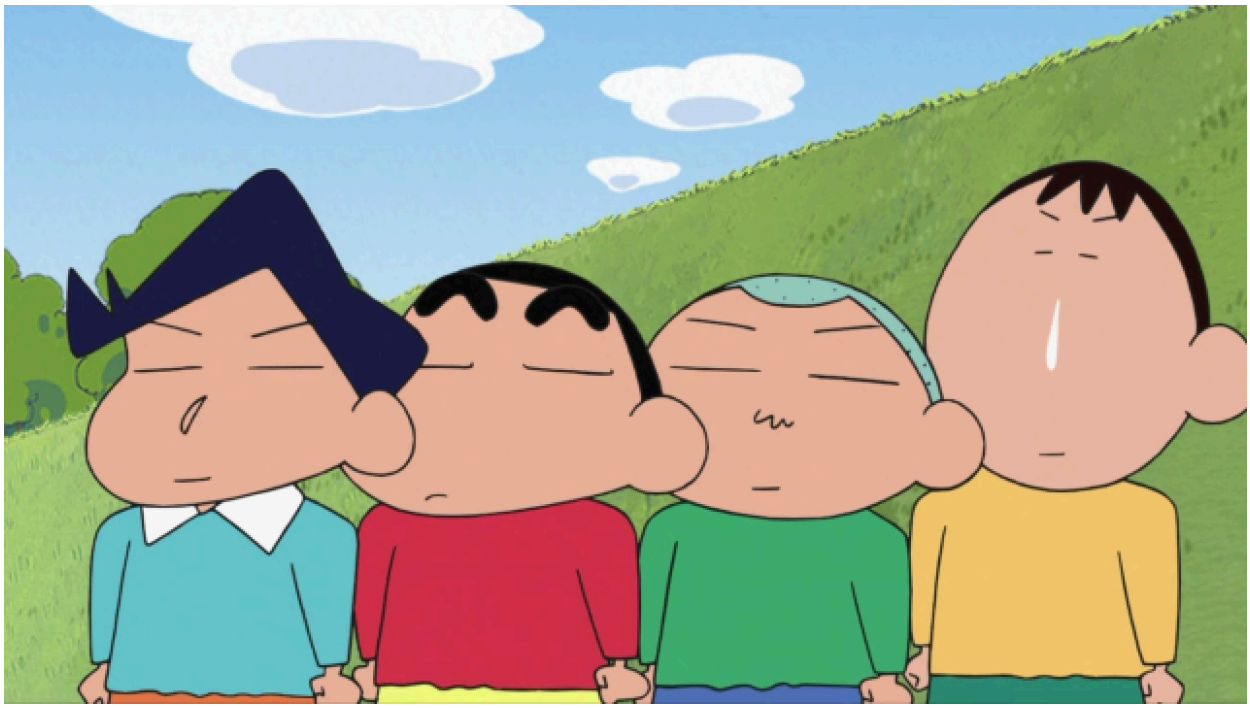
    - img2: #8e7662 #ead8c6 #c1ab94 #816954 #503c20 #6d593e #dec8b5 #a38f7d #5e4b2f #1e0d05 #443112 #d2b89d #b79e85 #c9b8a9 #9a816b #322008

- img3:#bbd9ef #2a2b3e #e8b29b #a59b68 #19130f #63bfc9 #a3c874 #a5d1ee #f0c96c #f6f8f9 #83a456 #99c4ba #6b8646 #92c7ec #3fa96e #cf3249

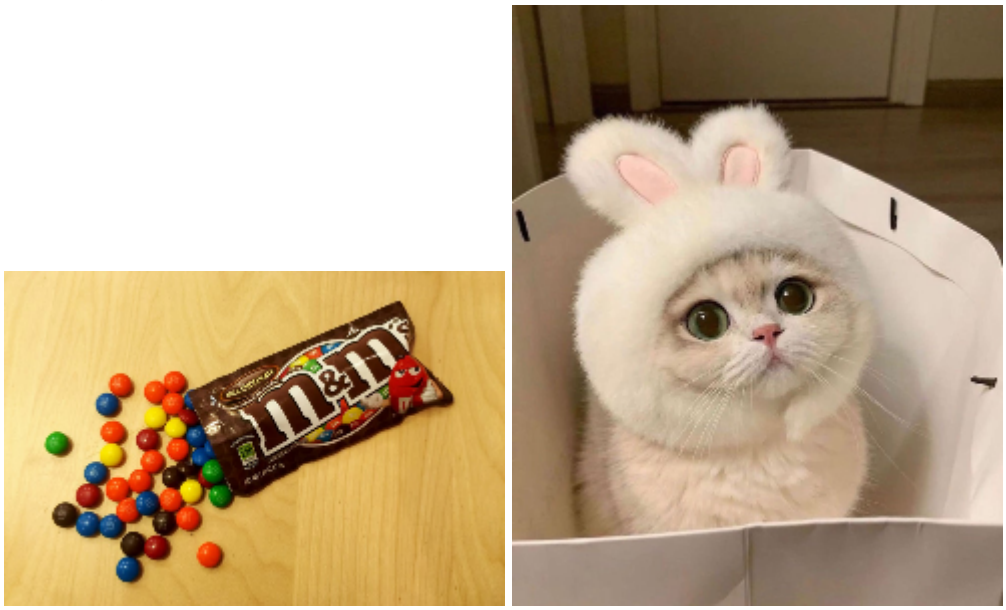


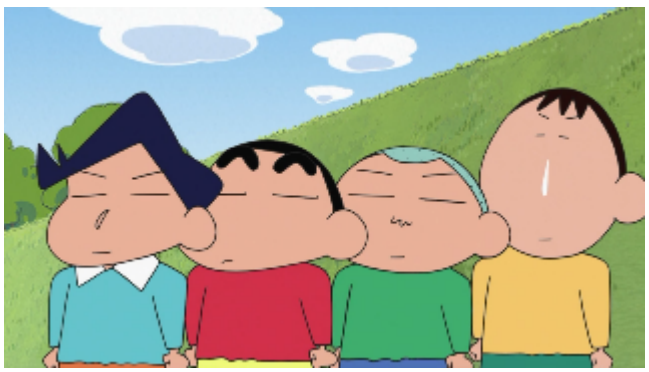
Q2

- P1:
  - 創建一張空的兩倍的圖片並將在原始圖片的每個像素（在 `imgs[i]`中的 `row, col`）複製到 `result` 中的四個相鄰像素（在 `row*2, col*2`、`row*2, col*2+1`、`row*2+1, col*2`、和 `row*2+1, col*2+1`）



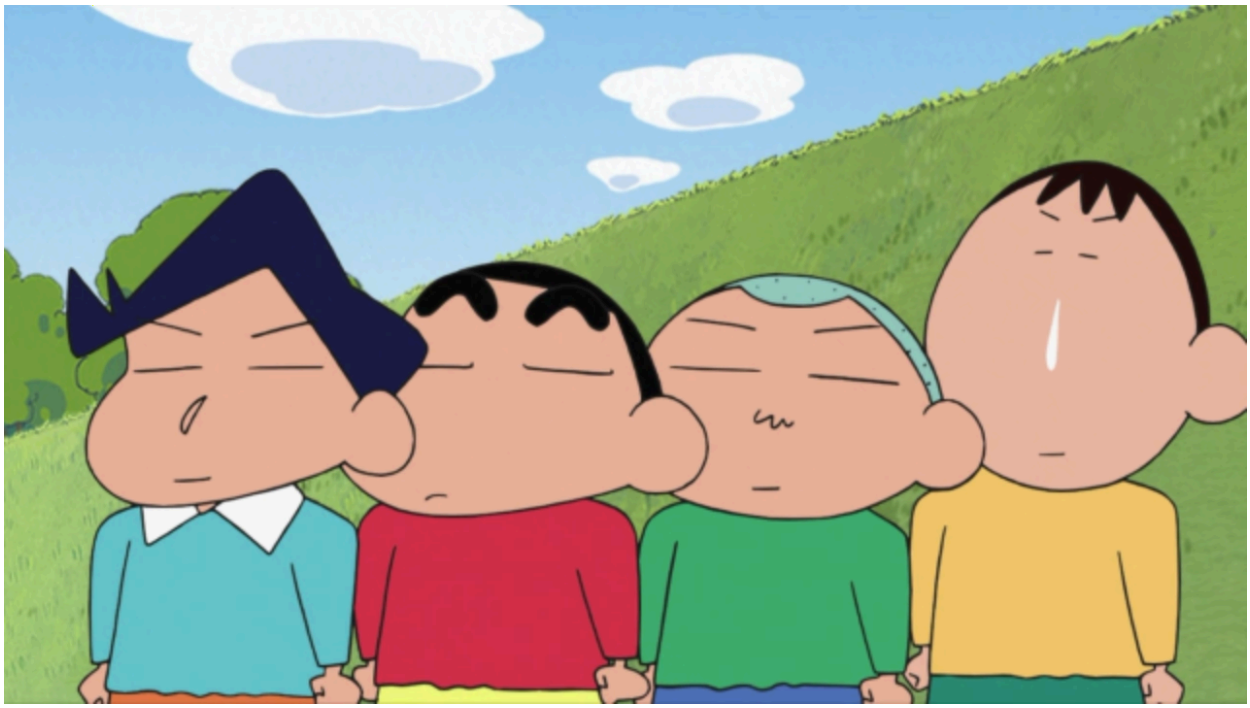
- 創建一張空的二分之一被大小的圖片並將每個像素 使用 `result[row, col] = (imgs[i][row*2, col*2])` 進行替換





- P2:
  - 先計算對應 pixel 在原始圖片的相對座標，並進一步對應到在圖中的 pixel 位置，並對邊界座標處理
  - 利用前一步驟得到的座標依照線性插值公式計算出新的 pixel 值

- 放大



- 縮小