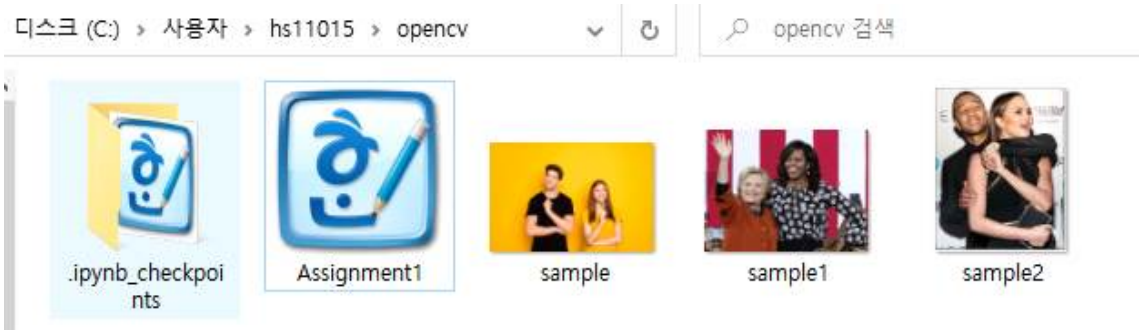
[3193] 디지털영상처리 영상로딩 프로그램 실습 및 컬러변환 과제 202012468 김현서

우선 C드라이브>사용자>hs11015 에 opencv라는 하위 폴더를 하나 만들어주고, 결과를 볼 수 있는 사진을 3개정도 저장을 하였다.



아나콘다를 다운받고, anaconda prompt를 이용해 jupyter에 접속해줬다.

접속 후 Assignment1이라는 python3 파일을 하나 만들어주고, 코드를 작성하기 시작했다.

<코드리뷰>

```
In [238]: import cv2
          import numpy as np
          from math import acos, pi, sqrt
```

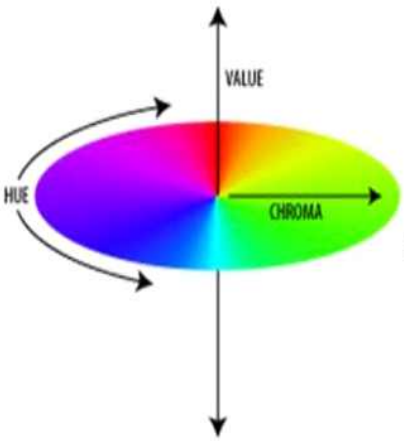
1번째 cell에서 cv2, numpy, math 라이브러리를 import 해준다.
cv2로 불러올 수 있는 opencv 라이브러리는 이미지, 영상을 읽고 화면에 출력하거나 새로 저장하는 기능을 한다.
numpy의 경우 다차원 배열을 다룰 수 있게 도와주는 라이브러리이고, 배열을 이용해 이미지나 영상을 픽셀단위로 읽고, 나타낼 수 있도록 한다.
math 라이브러리 안에서 RGB를 HSI로 변환할 때 사용할 acos, pi, sqrt 함수만 불러온다.

```
In [239]: # B,G,R
          def compute_Hue(B, G, R):
              angle = 0
              if B != G != R:
                  angle = 0.5*((R - G) + (R - B)) / sqrt((R-G)*(R-G) + (R-B)*(G-B))
              return acos(angle) if B <= G else (2*pi - acos(angle))
```

2번째 cell은 Hue(색)을 계산하는 함수를 만든 것이다. 강의 시간에 배운 공식을 적용했다.

HSI/HSV Color space

☐ RGB → HSI 스톡맨의 공식을 사용



$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

$$s = 1 - 3 \cdot \min(r, g, b); \quad s \in [0,1]$$

$$i = (R+G+B)/(3 \cdot 255); \quad i \in [0,1]$$

$$h = \cos^{-1}\left\{\frac{0.5 \cdot [(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{1/2}}\right\} \quad h \in [0,\pi] \text{ for } b \le g$$

$$h = 2\pi - \cos^{-1}\left\{\frac{0.5 \cdot [(r-g)+(r-b)]}{[(r-g)^2+(r-b)(g-b)]^{1/2}}\right\} \quad h \in [\pi,2\pi] \text{ for } b > g$$

In [246]:
```
src = cv2.imread('sample1.jpg', cv2.IMREAD_COLOR)
height, width = src.shape[0], src.shape[1]

I = np.zeros((height, width))
S = np.zeros((height, width))
H = np.zeros((height, width))

for i in range(height):
    for j in range(width):
        B, G, R = src[i][j][0]/255., src[i][j][1]/255., src[i][j][2]/255.

        I[i][j] = (B+G+R)/3.
        if B+G+R != 0:
            S[i][j] = 1 - 3*np.min([B,G,R])/(B+G+R)
        H[i][j] = compute_Hue(B, G, R)

dst = np.zeros((height, width, 3), dtype=np.uint8)

for i in range(height):
    for j in range(width):
        #if H[i][j] >= 0.1 and H[i][j] <= 0.6 and S[i][j] > 0.08 and S[i][j] <= 0.5 and I[i][j]>=0.27 and I[i][j]<=0.95:
        #    dst[i][j] = src[i][j]
        if H[i][j] <= 0.7 and S[i][j] > 0.02 and S[i][j] <= 0.37 and I[i][j]>=0.25:
            if H[i][j] > 0.5 and S[i][j] >= 0.125 and I[i][j]<=0.9:
                continue
            else:
                dst[i][j] = src[i][j]

cv2.imshow("dst", dst)
cv2.imshow("src", src)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3번째 Cell은 opencv 라이브러리 안의 imread 매서드를 사용해 사진을 읽어온다.
사진을 읽은 후, 사진의 높이와 너비를 사진의 shape 정보에 저장해준다(리스트 형태)

그 후, I, S, H에 각각 (높이)x(너비) 행렬 형태로 0이 들어간 배열을 생성한다.

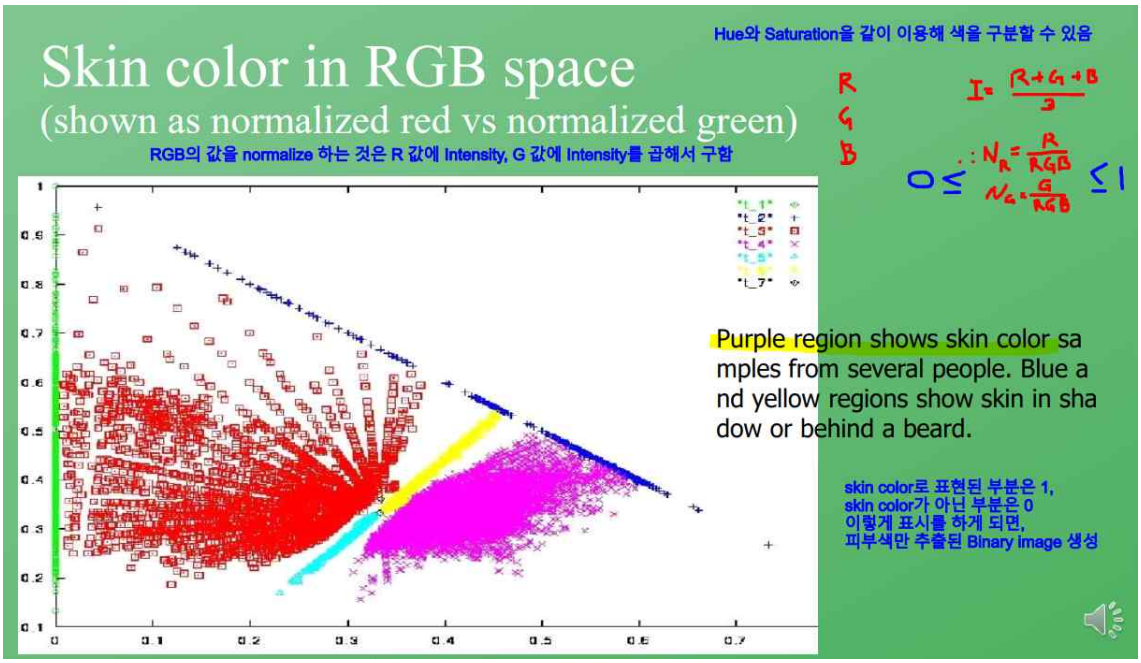
다음으로 for 반복문을 이용해 사진(src)의 픽셀을 하나씩 살핀다.
해당 픽셀의 RGB 값을 구한 후 변수에 할당하고, RGB 값들을 이용해 HSI 값을 구한다.
여기서 HSI 값들은 위에 사진에 있듯 강의에서 배운 공식을 이용해 구한다.
그 후 dst라는 변수에 (높이)x(너비) 행렬에 3가지 값(RGB)씩 들어가는 배열을 생성하고,
데이터타입은 부호가 없는 8비트의 정수형태이다.(Unsigned int - 8bit)

다음으로 for 반복문에서 H, S, I의 범위를 조건으로 이용해 사진을 출력한다.
나는 처음에는 조교님께서 코드를 짜주신 대로

```
for i in range(height):
    for j in range(width):
        if H[i][j] >= 0.25 and H[i][j] <= 0.6:
            dst[i][j] = src[i][j]
```

이라는 코드를 사용했었는데, 이렇게 하면 피부색이 제대로 추출되지 않았다.

따라서, Saturation과 Indensity의 범위도 조건에 사용해봐야겠다는 생각을 하게 되었다.
마침 2주차에 Skin color in RGB space라고 해서 피부색이 나타나는 범위를 배웠었다.



따라서 나는 이를 이용해 코드를 짜봐야겠다고 생각했다.
사진을 다운 받아 picsact라는 어플로 피부색을 스포이드로 검출해 HBI 수치를 측정했는데, 어떤 사진을 이용하든 대부분 Saturation은 0.5 이내가 나왔다.
그리고 Indensity는 너무 어두워서 검정색으로 보이지 않고 너무 밝아서 하양색으로 보이지 않는 범위를 지정해 코드를 짜봤다.

```
for i in range(height):
    for j in range(width):
        if H[i][j] >= 0.1 and H[i][j] <= 0.6 and S[i][j] > 0.03 and S[i][j] <= 0.5 and I[i][j]>=0.27 and I[i][j]<=0.95:
            dst[i][j] = src[i][j]
```

그래서 이런 코드를 짰다. Saturation도 indensity와 마찬가지로 0에 가까워질 수록 하양색이 되기 때문에 0.03 초과라는 범위를 잡았다.

그 결과로 이렇게 피부색을 추출해낼 수 있었다.

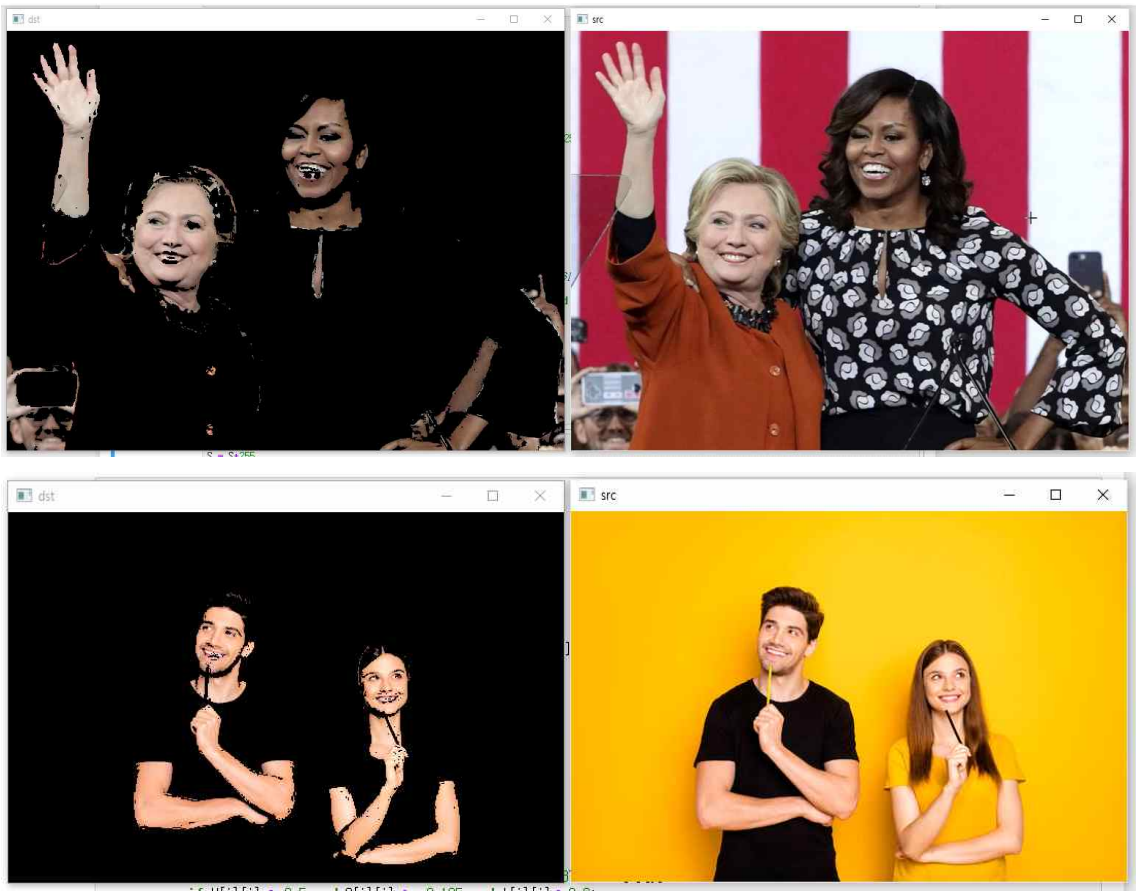
이대로 제출을 해도 괜찮을 것 같았지만, 왼쪽 여성분의 팔 피부색이 제대로 검출되지 않았고, 빨강색 단추가 너무나 선명히 잘 보여서 이를 수정하기 위해 고민을 해봤다.

고민을 하면서 Saturation과 Indensity 사이의 관계를 떠올려봤는데, Saturation은 값이 커질수록 색이 진해지고, Indensity는 색이 커질수록 하양색 즉 색이 없어진다는 것을 깨달았다. 따라서 Saturation과 Indensity는 반비례한다는 가설을 세우고 코드를 짜봤다.

```
for i in range(height):
    for j in range(width):
        if H[i][j] <= 0.7 and S[i][j] > 0.02 and S[i][j] <= 0.37 and I[i][j]>=0.25:
            if H[i][j] > 0.5 and S[i][j] >= 0.125 and I[i][j]<=0.9:
                continue
            else:
                dst[i][j] = src[i][j]
```

이런 코드였고, 빨강색 글씨로 서술한 if문의 범위에 속하는 픽셀 중 파랑색 글씨로 서술한 if문에 해당하는 픽셀은 나타내지 않고, 나머지만 나타내는 코드를 짰다.

빨강색 if문의 코드만 돌리면 이런 식으로 왼쪽 여성의 머리카락이 너무 많이 검출된다. 따라서 파랑색 if문을 이용해 Hue가 0.5를 초과하고, Saturation이 피부색이라고 하기엔 꽤 진하고, Indensity가 0.9 이하라 명도도 어느 정도 진한 색은 검출하지 않는 코드를 만들어 머리카락을 최대한 검출하지 않게 만들었다.



최종적으로 검출된 사진들이다! 아직 피부색이 아닌 값이 조금씩 검출되거나 피부색이 조금씩 덜 검출되기 때문에 코드를 더 발전시킬 필요가 있다.

사진은 cv2.imshow("사진 제목", 변수명)을 이용해 출력한다.

```
In [174]: I = I*255
          S = S*255
          H = H*255/(2*pi)
          I = np.asarray(I, dtype=np.uint8)
          S = np.asarray(S, dtype=np.uint8)
          H = np.asarray(H, dtype=np.uint8)

          cv2.imshow("src", src)
          cv2.imshow("H", H)
          cv2.imshow("S", S)
          cv2.imshow("I", I)
          cv2.waitKey(0)
          cv2.destroyAllWindows()
```

다음 셀에는 HSI 값에 다시 255를 곱하고 각각 값을 담은 8바이트 리스트로 만들어준다. 그 후 사진과 H값만 흑백으로 나타낸 사진, S값만 흑백으로 나타낸 사진, I값만 흑백으로 나타낸 사진을 각각 출력한다.