
Prob1

Chroma Key Editor



기타 제출일 : 2020.11.26

작성자 소속 : 산업경영시스템공학과

학번 : 2018316501

성명 : 박민지

■ 서론

이번 과제는 크로마키 에디터 과제입니다. 프로그램이 수행되면 창을 열고 단색 배경에 색상이 있는 객체를 A창으로 출력합니다. 마우스 왼쪽 버튼을 눌러 배경 화소 샘플을 채취한 후, 원하는 화소를 추출할 때까지 트랙바를 조정해 설정합니다. 원하는 화소가 추출되면 마우스 왼쪽으로 드래그앤 드롭을 사용하여 원하는 부분을 오려냅니다. 오려낸 사진을 새로운 창 B에 출력되는 배경사진에 붙여넣기 합니다.

■ 본문

1. 특정 색상의 화소 샘플 채취

```
ch_key = cv.imread('data/KakaoTalk_20201118_223449005.jpg')
back_gr = cv.imread('data/3956278e5c1e79e.jpg')
# back_gr = cv.imread(FullName2)
assert ch_key is not None, 'No image file....!'
assert back_gr is not None, 'No image file....!'
```

➔ 영상을 불러오는 변수를 ch_key로 지정합니다. 배경영상을 불러오는 변수는 back_gr로 지정합니다.

```
def mouse_callback(event, x, y, flags, param):
    global lower_blue1, upper_blue1, lower_blue2, upper_blue2, lower_blue3, upper_blue3, hsv,
           h_val, s_val, v_val
    if event == cv.EVENT_LBUTTONDOWN:
        print("x,y : ", x, y) # 마우스 클릭한 부분의 x,y좌표 출력
        color = ch_key[y, x]

        one_pixel = np.uint8([color]) # 한픽셀로 구성된 이미지로 변환
        hsv = cv.cvtColor(one_pixel, cv.COLOR_BGR2HSV) # HSV색 공간으로 변환
        hsv = hsv[0][0] # HSV 가져오기 위해 픽셀값을 가져옴
```

```

print("bgr : ", color)
print("hsv : ", hsv)

# h_val = cv.getTrackbarPos('H', 'HSV')
# s_val = cv.getTrackbarPos('S', 'HSV')
# v_val = cv.getTrackbarPos('V', 'HSV')

print("h_val : ", h_val)
print("s_val : ", s_val)
print("v_val : ", v_val)

# 마우스 클릭한 픽셀값과 유사한 픽셀값 범위 지정
if hsv[0] < 10:
    lower_blue1 = np.array([hsv[0] - 10 + 180, s_val, v_val])
    upper_blue1 = np.array([180, 255, 255])
    lower_blue2 = np.array([0, s_val, v_val])
    upper_blue2 = np.array([hsv[0], 255, 255])
    lower_blue3 = np.array([hsv[0], s_val, v_val])
    upper_blue3 = np.array([hsv[0] + 10, 255, 255])

elif hsv[0] > 170:
    lower_blue1 = np.array([hsv[0], s_val, v_val])
    upper_blue1 = np.array([180, 255, 255])
    lower_blue2 = np.array([0, s_val, v_val])
    upper_blue2 = np.array([hsv[0] + 10 - 180, 255, 255])
    lower_blue3 = np.array([hsv[0] - 10, s_val, v_val])
    upper_blue3 = np.array([hsv[0], 255, 255])

else:
    lower_blue1 = np.array([hsv[0], s_val, v_val])
    upper_blue1 = np.array([hsv[0] + 10, 255, 255])
    lower_blue2 = np.array([hsv[0] - 10, s_val, v_val])
    upper_blue2 = np.array([hsv[0], 255, 255])
    lower_blue3 = np.array([hsv[0] - 10, s_val, v_val])

```

```
upper_blue3 = np.array([hsv[0], 255, 255])
```

- ➔ 마우스 왼쪽을 클릭하여 클릭한 부분의 특정 색상의 화소 샘플을 채취합니다. 클릭한 좌표의 x와 y좌표를 출력합니다. RGB로 되어있는 영상을 HSV로 변환한 뒤 h,s,v로 나눠 트랙바의 시작을 알려줍니다.

2. 트랙바 움직임

```
def nothing(x):
```

```
    pass
```

- ➔ 트랙바를 실행하기 위해 nothing 함수를 작성합니다.

```
cv2.namedWindow('HSV')
```

```
cv2.createTrackbar('H', 'HSV', h_val, 360, nothing)
```

```
cv2.createTrackbar('S', 'HSV', s_val, 255, nothing)
```

```
cv2.createTrackbar('V', 'HSV', v_val, 255, nothing)
```

- ➔ 트랙바가 생성되는 창 이름을 HSV로 지정합니다. 트랙바를 H, S, V로 생성하고 360, 255, 255까지 움직일 수 있도록 설정합니다.

3. 영상 드래그 및 파일 저장

```
def mouse_callback2(event, x, y, flags, param):
```

```
    global image_to_show, s_x, s_y, e_x, e_y, mouse_pressed, drawing_needed
```

```
    if event == cv2.EVENT_LBUTTONDOWN: # 마우스 왼쪽버튼 클릭
```

```
        mouse_pressed = True
```

```
        s_x, s_y = x, y # 선택 시작 좌표 기록
```

```
    elif event == cv2.EVENT_LBUTTONUP: # 마우스 왼쪽버튼 떼기
```

```
        mouse_pressed = False
```

```
        e_x, e_y = x, y # 선택 종료 좌표 기록
```

```
drawing_needed = True
```

→ mouse_callback2함수를 생성하여 마우스 왼쪽 버튼을 누르는 순간의 좌표를 저장하고, 때는 순간의 좌표를 기록해서 원하는 부분의 영상을 오려냅니다.

```
# 이미지 크롭하기
```

```
if drawing_needed == True:
```

```
    if s_y > e_y: # y축상의 시작점과 끝점이 바뀌었으면 두 좌표 바꿈
```

```
        s_y, e_y = e_y, s_y
```

```
    if s_x > e_x: # x축상의 시작점과 끝점이 바뀌었으면 두 좌표 바꿈
```

```
        s_x, e_x = e_x, s_x
```

```
img_crop = result[s_y:e_y, s_x:e_x]
```

```
cv.imshow('crop', img_crop)
```

```
cv.imwrite('crop.jpg', img_crop)
```

```
drawing_needed = False
```

→ 만약 시작점과 끝점의 x와 y좌표가 바뀌었으면 두 좌표를 바꿔주고, 그 결과를 img_crop변수에 저장해 줍니다. crop.jpg명으로 오려낸 이미지를 저장합니다.

4. 새로운 창 B

```
# b버튼을 누르면 배경영상을 출력
```

```
if k == ord('b'):
```

```
    cv.imshow('B', back_gr)
```

```
    bitOperation(10, 10)
```

→ 키보드의 b 버튼을 누르면 B창이 새로 로드 되면서 배경영상을 출력합니다.

```
def bitOperation(hpos, vpos):
```

```
    global img1
```

```
    img2 = cv.imread('crop.jpg')
```

```
    if img1.shape < img2.shape:
```

```
        img1 = cv.resize(img1, (1400, 750), interpolation=cv.INTER_AREA)
```

```

# 크롭 이미지 크기 구하기
rows, cols, channels = img2.shape

# 이미지 위치 영역 잡기
roi = img1[vpos:rows + vpos, hpos:cols + hpos]

# 크롭이미지 흑백 변환 후 마스크 생성
img2gray = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
ret, mask = cv.threshold(img2gray, 10, 255, cv.THRESH_BINARY)
mask_inv = cv.bitwise_not(mask)

img1_bg = cv.bitwise_and(roi, roi, mask=mask_inv)
img2_fg = cv.bitwise_and(img2, img2, mask=mask)

# 검정색 픽셀값은 0이니 검은색은 없어지고 검정색이 아닌 색 출력
dst = cv.add(img1_bg, img2_fg)
# 최종 이미지 출력
img1[vpos:rows + vpos, hpos:cols + hpos] = dst

cv.imshow('result', img1)

```

- ➔ 잘라내기 한 이미지의 크기를 구한다음에 배경영상에 hpos, vpos를 입력한 위치의 x와 y 좌표에 영상을 집어넣습니다. 넣기 전에 잘라내기한 영상을 흑백으로 변환 후 마스크를 생성합니다. 그러면 검정색의 픽셀값은 0이기 때문에 검정색이 없어진 후 검정색이 아닌 색이 출력됩니다.

5. 최종영상 저장

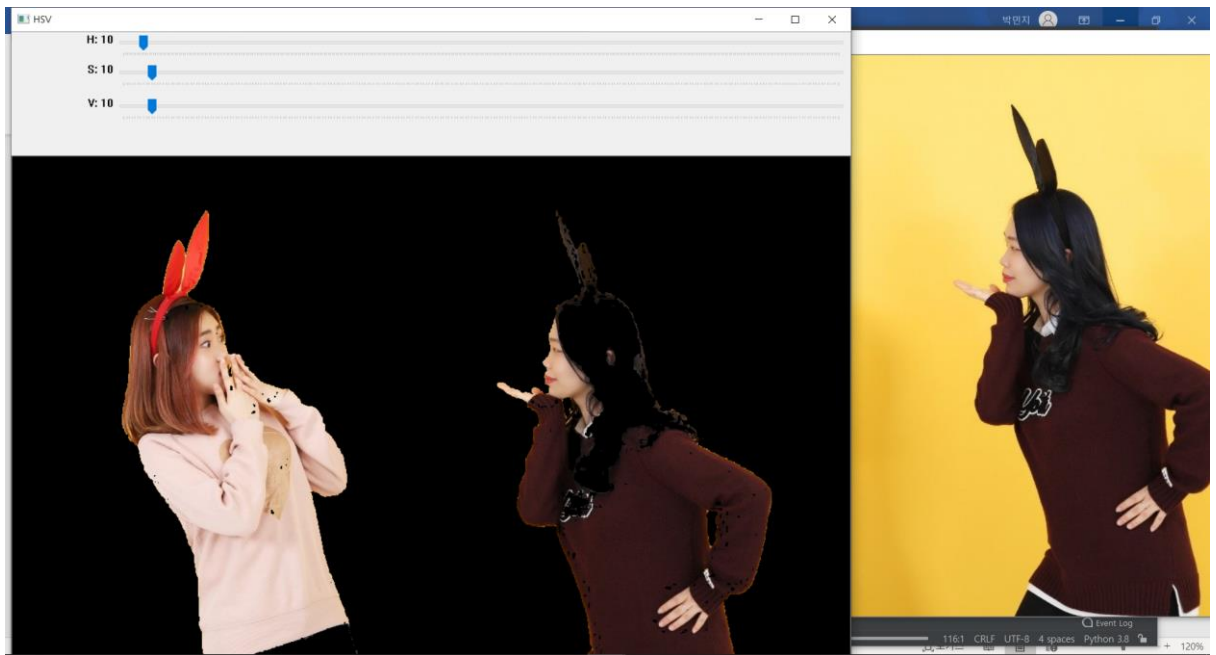
```

if k == ord('s'):
    cv.imwrite('image.jpg',img1)

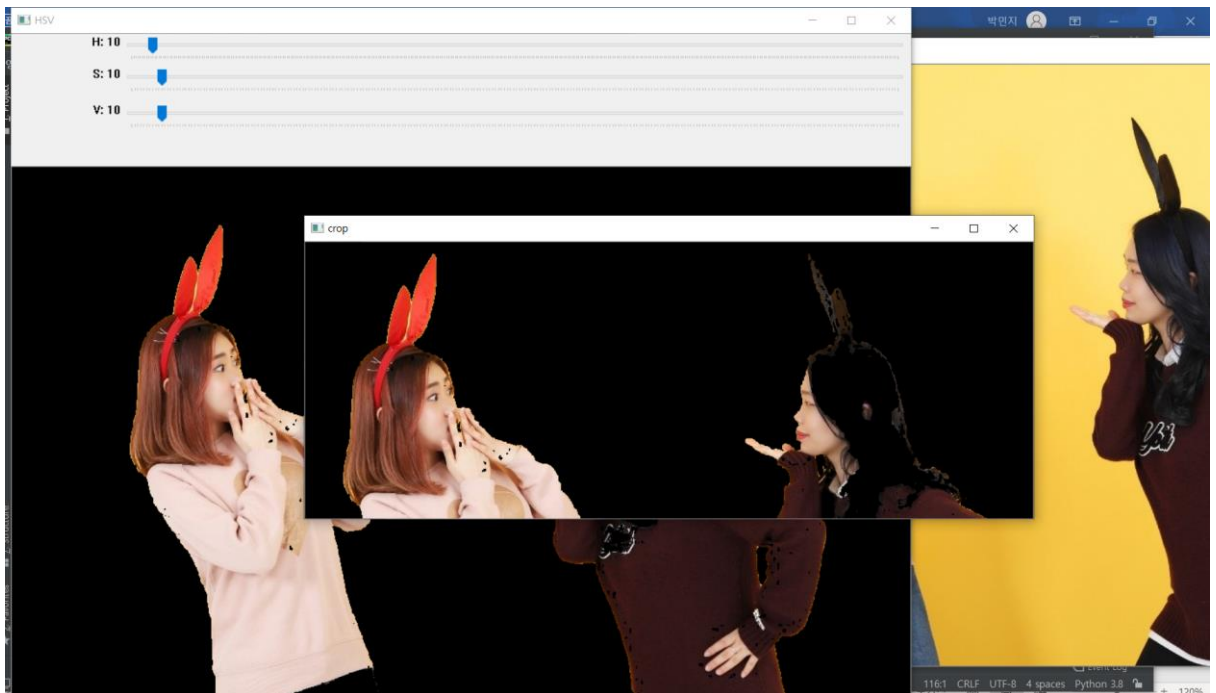
```

- ➔ 키보드의 s 버튼을 누르면 최종적으로 배경영상에 크로마키 된 영상이 image.jpg로 저장됩니다.

6. 프로그램 실행

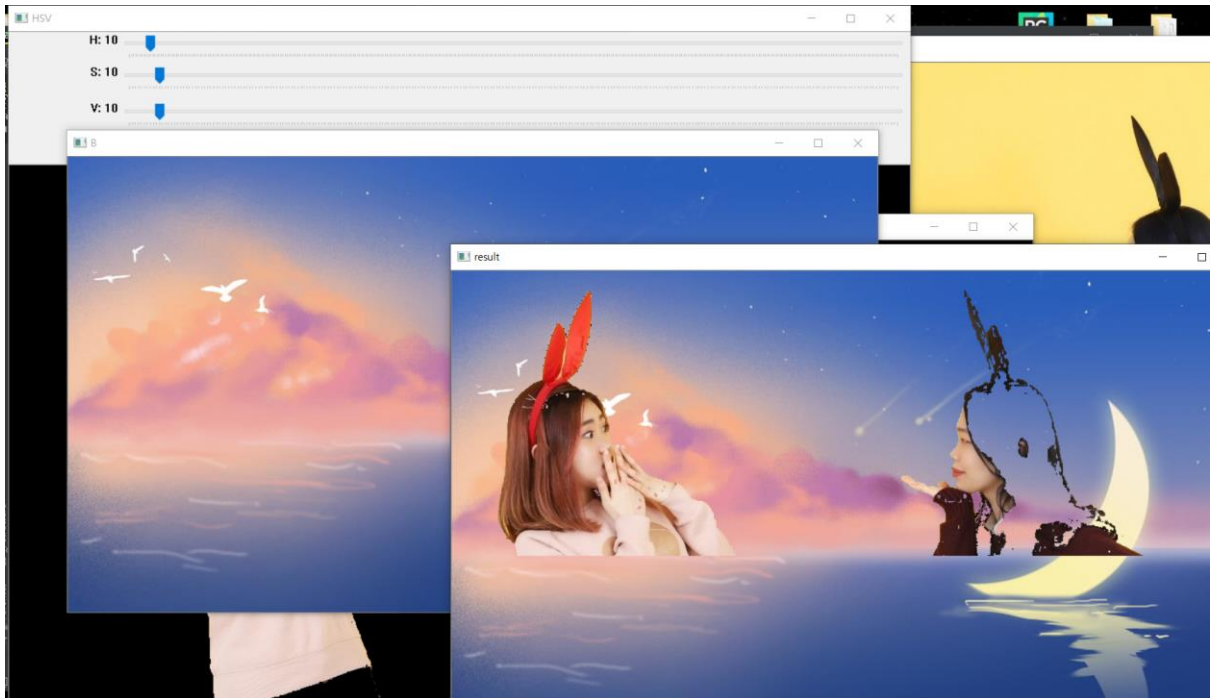


프로그램을 처음 실행 했을 시 HSV와 창A가 뜨게 됩니다. 창 A에서 필요한 부분을 클릭 하면 HSV에 그 부분을 제외하고 배경이 검은색으로 변하게 됩니다. HSV를 움직이면서 좀 더 원하는 화소를 채취합니다.



HSV 창에서 드래그를 하게되면 crop이란 이름으로 드래그 한 부분이 생성됩니다. Crop

창이 뜨면서 crop.jpg로 위 영상이 저장됩니다.



B버튼을 누르게 되면 창 B가 출력됩니다. 그리고 result 창에 배경영상과 crop영상이 함께 출력 됩니다. 이 화면에서 s를 누르게 되면 result 영상이 image.jpg로 저장됩니다.

■ 결론

우선 지난 1차 과제때의 트랙바를 생성했을 때와는 다르게 바로바로 영상이 H, S, V에 따라서 영상이 처리되지 않았습니다. 트랙바를 움직이고 다시 영상의 좌표를 클릭해야 새로 바뀐 H, S, V에 따라 영상이 처리되었습니다.

B를 입력하면 새로운 창 B가 열리는는 하나 창 B에서 다시 드래그 해서 지정한 위치에 crop한 영상의 크기가 재지정 되어 overwrite가 되지 않고, bitOperation에 넣은 x와 y좌표에 넣어 집니다. 이 문제를 해결하려고 하였으나, mouse_callback2가 b창에서는 제대로 작동하지 않았습니다. 그래서 C를 입력하였을 때 A창과 B창을 왔다갔다 하며 크로마키를 새로 만들 수 없습니다.

또한, 영상을 crop 하고 b를 눌러 새 창을 열 때 만약 crop한 이미지가 back_gr이미지보다 크기가 크다면 크로마키가 되지 않고 오류가 나면서 종료가 됩니다. 이 부분을 고치기 위해선 b 창에서 드래그가 되어야 하는데 되지 않아 이미지 사이즈를 조절하지 못해 생긴 오류라고 봅니다. 그래서 임시 방편으로 crop이미지가 back_gr보다 크다면 back_gr의 가로와 세로 길이를 늘려 출력하도록 하였습니다.