URColor (너만의 컬러)

20185103 고성지 영상처리 프로그래밍 기말프로젝트

1) 프로그램 개요

감각적인 그림을 흥미롭게 보는 편인데 그 중 좋아하는 사진은 **고해상도 흑백 사진에 노란색 부분만 추출**한 사진이었다. 굉장히 단순한 사진이면서도 세련된 포인트를 잘 잡았다고 느꼈다. 이번 영상처리 프로그래밍수업에서 RGB에 대해서는 잘 알고 있었지만 HSV의 개념을 처음 알고 공부하였다. 이 개념에서 착안한 아이디어가 위에서 말한 일부 색상만을 추출하는 프로그램을 만들면 실제로도 편하게 쓸 수 있을 것 같았다. 따라서 **너만의 컬러라는 주제로 흑백 사진에서 원하는 색상만을 추출하여 저장할 수 있는 프로그램을 제공한다.**

2) 핵심 코드구현

A. UI의 구현

openCV를 사용하여 표현할 때 어떻게 인터페이스로 제공하면 좋을지 고민을 많이하였는데, 수업 교재인 OpenCV-Python으로 배우는 영상처리 및 응용 교재 속 그림판 실습예제를 참고하여 진행하기로 하였다. 색상을 고르고 버튼에 따라 기능을 달리하는 점에서 매우 좋은 소스코드임을 판단하였다.

B. 제공 기능

전반적인 아이디어: HSV로 변환 한 후 Hue값을 통해 색상을 추출한다. 즉, 비교는 Hue값으로 하고 흑백사진 속 Hue의 조건이 만족되면 컬러사진 속의 같은 픽셀의 RGB값을 흑백사진에 대입하여 컬러화 시키는 방식이다. 여기서 흑백사진 속에 RGB를 대입하려면 3차원이 되어야 하므로, GRAY로 변환 후 다시 COLOR로 변환함으로써 이를 테면 128 채널값에서 [128, 128, 128]의 3차원 채널값으로 바꾼다.

i. 파일 열기

파이썬 내장 라이브러리인 tkinter을 사용하여 파일 탐색기를 통해 사진을 선택할 수 있게 하며 경로를 저장해 cv2.imread를 통해 사진을 불러온다. 이 사진을 통해 color이미지, gray이미지 두 개로 저장하고 cv2cvtColor()를 통해 HSV로 변환한 이미지 총 3가지의 이미지를 저장한다. 또한 인터페이스 화면에 꽉 찰 수 있도록 cv2.resize()를 통해 크기 변경을 진행한다. cv2.imwrite()는 추후 UNDO기능을 위해 임시 파일을 저장한다.

```
elif mode == OPEN_IMAGE:
    print('open file')
    root = tk.Tk()
    filePath = filedialog.askopenfilename(initialdir='', title='Select file', filetypes=(('jpg files', '*.jpg'), ('png files', '*.png'), ('all files', '*.*')))
    root.destroy()

    color_img = cv2.imread(filePath)
    color_img_resize = cv2.resize(color_img, (680, 500))
    gray_img = cv2.cvtColor(color_img, cv2.COLOR_BGR2GRAY)
    gray_img = cv2.cvtColor(gray_img, cv2.COLOR_GRAY2BGR) # 색상 대입을 위해 3차원화
    gray_img_resize = cv2.resize(gray_img, (680, 500))
    hsv_img_resize = cv2.cvtColor(color_img_resize, cv2.COLOR_BGR2HSV)
    background[:, 120:] = gray_img_resize

    cv2.imwrite('../temp/log.jpg', background)
```

ii. 색 추출

```python
def BGR2H(color):
    V = max(color)
    B, G, R = color
    minGBR = min(color)
    if V == B:
        result = 240 + (60*(R-G)) / (V - minGBR)

    if V == G:
        result = 120 + (60*(B-R)) / (V - minGBR)

    if V == R:
        result = (60*(G-B) / (V - minGBR))

    result = (result/360) * 170

    if result < 0:
        result += 170

    return result
```

구현된 HUE와 팔레트를 이용해 색을 지정하면 그 RGB값을 통해 Hue를 구하는 BGR2H 함수를 구현한다. (BGR2H 함수) openCV에서의 H값은 기존 0~360이 아닌 0~170이므로 스케일링 작업까지 진행하였다.

Hue변환이 완료되면 iii기능인 STACK모드 여부를 확인하고, 픽셀을 순회하며 변환된 Hue에 +-20의 오차범위 내에 들은 픽셀값의 RGB값을 대입한다. 즉, 비교대상은 HSV로 변환된 이미지이다. 추후 추출된 색을 변경하기 위한 tempPos를 통해 추출된 픽셀값을 전부 저장한다. **이 프로젝트의 가장 핵심적인 기능이다.**

```python
elif mode == EXTRACT_COLOR:
    HUE = BGR2H(Color)
    print(HUE)

    if usingStack:
        background[:, 120:] = gray_img_resize

    for i in range(500):
        for j in range(120, 800):
            p = hsv_img_resize[i][j-120]
            if HUE - 20 < p[0] <= HUE + 20 and p[1] != 0 and p[2] != 0:
                background[i, j] = color_img_resize[i, j-120]
                tempPos.append([i, j])

    cv2.imwrite('../temp/log.jpg', background)
```

iii. 스택 모드 & 색상 변경 & reset

```python
elif mode == STACK_COLOR:
    if usingStack:
        usingStack = False
        print('Change stack mode : ON')

    else:
        usingStack = True
        print('Change stack mode : OFF')


elif mode == CHANGE_COLOR:
    cv2.imwrite('../temp/log.jpg', background)
    for pos in tempPos:
        r, c = pos
        background[r, c] = Color
        tempPos = []


elif mode == RESET_IMAGE:
    print('reset')
    cv2.imwrite('../temp/log.jpg', background)
    background[:, 120:] = gray_img_resize
```

STACK_COLOR는 색상을 여러 번 추출 할 때 기존 추출된 색상을 유지할지, 새로그릴지 정할 수 있는 모드이다.

CHANGE_COLOR는 추출된 모든 픽셀값에 대해 특정한 색으로 변환을 시켜주는 기능을 한다. 팔레트에 지정된 색으로 변경된다. 추출된 픽셀들은 위에서 설명했듯 tempPos에 모두 저장된다.

RESET_IMAGE는 잘못 추출하거나 마음에 들지 않을 때 초기 흑백사진으로 돌아가게 한다.

iv. 원본 보기(컬러) & 뒤로가기 & 블러링 & 샤프닝 & 저장 & 종료

```
126     elif mode == SHOW_ORIGIN:
127         print('show origin')
128         cv2.imwrite('../temp/log.jpg', background)
129         for i in range(500):
130             for j in range(120, 800):
131                 background[i, j] = color_img_resize[i, j-120]
132         background[:, 120:] = color_img_resize
133 
134     elif mode == UNDO:
135         background = cv2.imread('../temp/log.jpg')
136 
137 
138     elif mode == BLURRING_IMAGE:
139         filter = np.ones((5, 5), np.float32) / 25
140         background[:, 120:] = cv2.filter2D(background[:, 120:], -1, filter)
141 
142     elif mode == SHARPNING_IMAGE:
143         filter = np.array([[1, -2, 1], [-2, 5, -2], [1, -2, 1]], np.float32)
144         background[:, 120:] = cv2.filter2D(background[:, 120:], -1, filter)
145 
146     # elif mode == INVERSION_COLOR:
147     #     background[:, 120:] = 255 - background[:, 120:]
148 
149     elif mode == SAVE_IMAGE:
150         cv2.imwrite('result.png', background[:, 120:])
151         print("save!")
152 
153     elif mode == EXIT:
154         cv2.destroyAllWindows()
155         return
156 
157     cv2.imshow("PaintCV", background)
```

SHOW_ORIGIN은 원본 컬러영상을 출력시켜준다. color_img_resize를 불러오면 된다.

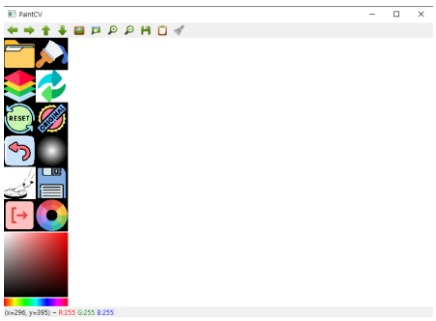
UNDO는 각 기능을 실행할 때 마다 저장해놓은 팔레트 전체의 사진을 다시 불러와 한 단계 복구를 할 수 있는 기능이다.

BLURRING_IMAGE는 이미지에 블러링을 적용하는 것이다. 필터는 5*5의 1/25로 이루어진 필터를 이용하였다. 블러링 기능으로 추출된 색을 제외한 아웃포커싱 기능도 수행한다. 블러링 아이콘을 여러 번 누르고 색 추출을 다시 누르게되면 색 추출되는 부분은 다시 칠해지므로 그 부분만 선명해지는 효과를 볼 수 있다.

SHARPNING은 블러링과 마찬가지인데 필터가 다를 뿐이다. save이미지는 현재까지 적용된 변환 사진을 저장할 수 있는 기능이며 EXIT는 프로그램을 종료하는 기능이다.

3) 프로그램 시연 사진

A. 프로그램 초기화면

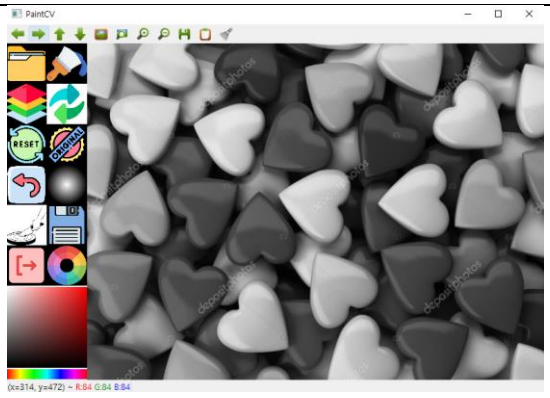


왼쪽에 색상을 지정하는 팔레트와 각종 기능을 수행하는 아이콘이 있으며, 오른쪽 흰색 화면에 사진이 출력된다.

B. 사진을 불러올 경우

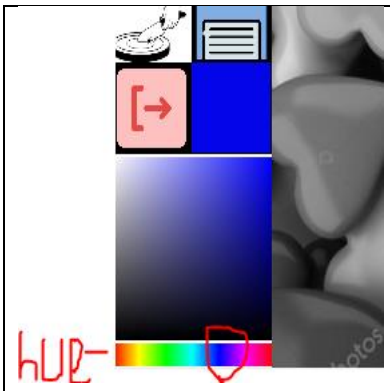
흑백화면으로 시작을 한다. 원본을 보려면 구현된 ORIGIN아이콘을 누르면 볼 수 있다.

원본 사진

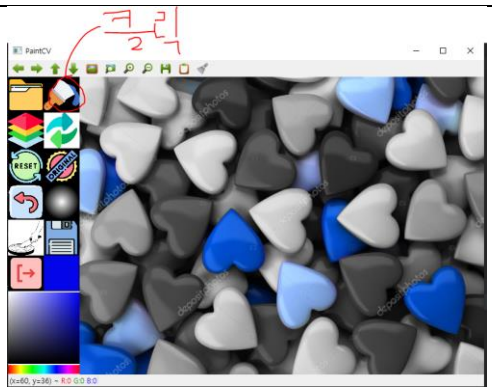


파일 열기 기능을 통해 불러올 경우

C. 파란색 하트만 추출

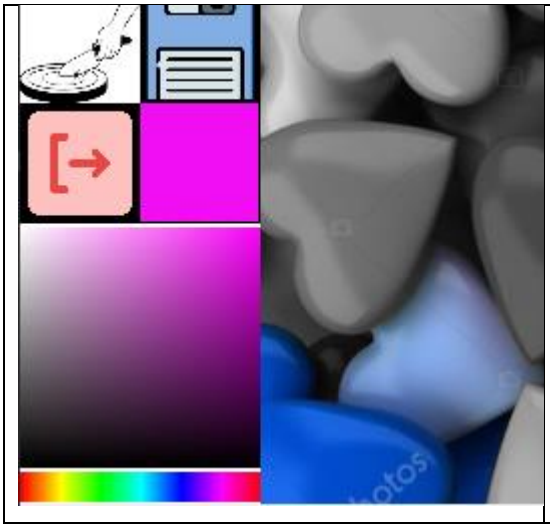


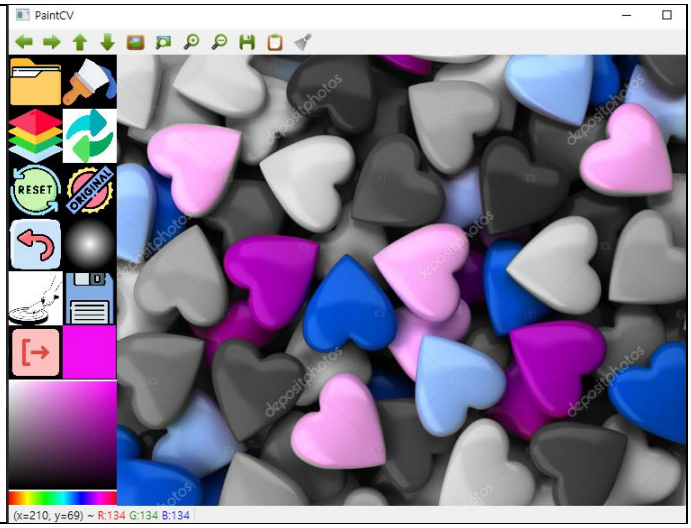
색상 지정 (마우스 클릭 이용)



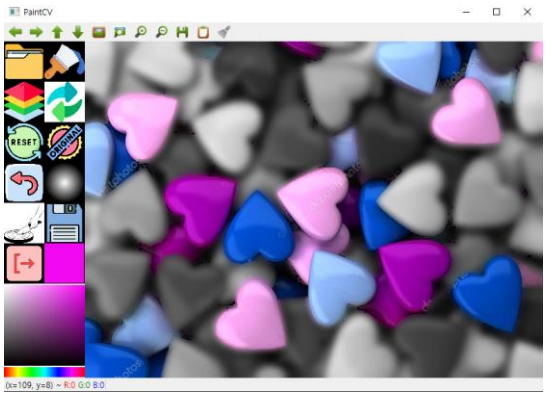
지정된 색상만 추출 완료

D. 이어서 핑크색 하트도 추출 (STACK mode가 ON인 상태)

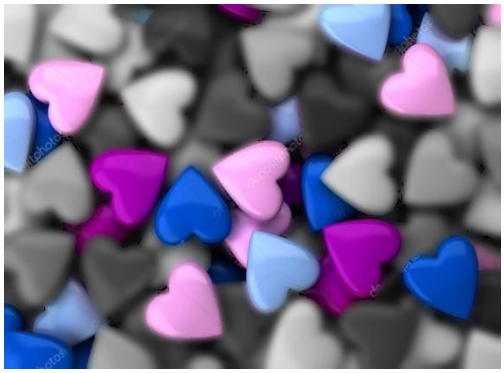




E. 블러링을 이용한 아웃포커싱 (파란, 핑크 하트 제외 블러) 그리고 저장(saveIcon 클릭)한 사진



편집화면



저장된 사진