2022학년도 1학기 디지털영상처리I

프로젝트 보고서

주제명	네 주변을 블링블링		
개발자	학번	성명	E-mail
	20203218	임미선	soup7912@gmail.com

〈 주제 설명 〉

입력 영상	목표 정의
이미지	1. 이미지를 받아와서 전처리 2. 배경, 전경 분리 후 배경에 이미지 삽입

〈 대표 출력 결과 〉

〈 구현 목표 1 〉

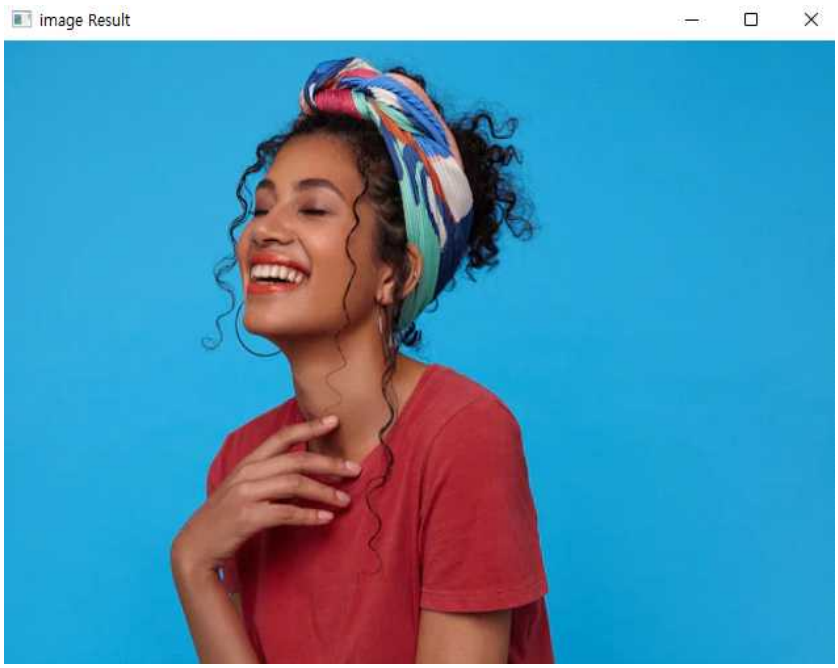
<table>
<tr><td>내용</td><td>이미지를 받아와서 전처리</td></tr>
<tr><td>접근
방법</td><td>● 필요한 input : 이미지, 배경 이미지
● 회색조 영상이어야 이미지 전처리가 쉬움 -> 복사본으로 생성한 이미지를 회색조 영상으로 변환하여 노이즈를 제거해보자!</td></tr>
<tr><td>적용
결과

및

코드
설명</td><td>

```
# 이미지 입력
image = cv2.imread('image.jpg')
image = cv2.resize(image, (640, 480))

# 이미지 적용
scene = cv2.imread('640x480-image.jpg')
scene = cv2.resize(scene, (640, 480))


# 이미지 복사본 생성 후, salt pepper noise를 제거한다.
cloneImage = image.copy()  # 현재 이미지의 복사본 생성
grayImage = cv2.cvtColor(cloneImage, cv2.COLOR_BGR2GRAY)  # 복사본을 회색조 영상으로 변환
grayImage = cv2.medianBlur(grayImage, 9)  # 블러링으로 노이즈 제거
```

● cv2.imread를 통해 이미지 input을 받음
● 현재 이미지와 배경 이미지의 size를 동일하게 지정함
● 이미지 전처리를 위해 회색조 영상으로 변환 후, 블러링을 수행하여 노이즈를 제거함



▲ 전처리 적용 이미지 출력

</td></tr>
</table>

(지면 부족시 추가하여 작성)

〈 구현 목표 2 〉

<table>
<tr><td>내용</td><td>배경, 전경 분리 후 배경에 이미지 삽입</td></tr>
<tr><td>접근
방법</td><td>
● 어떻게 배경과 전경을 분리할 것인가?
=> 커널을 생성하여 여러 영상처리 기법을 사용해보자!
● edge를 어떻게 검출할 것인가?
=> Canny Edge Detection 기법을 사용해보자!
● edge 검출의 임계치를 어떻게 조절할 것인가?
=> 팽창, 침식 수행 시 사용되는 커널의 크기를 조절해보자!

(URL 참고하였음)
● 분리된 배경에 어떻게 이미지를 삽입할 것인가?
=> 배경을 0, 전경을 255로 지정하여 배경 이미지의 화소를 하나씩 대입해주자!
</td></tr>
<tr><td>적용
결과

및

코드
설명</td><td>
<pre>
# 커널 생성
kernel = np.ones((2, 2))
thresholdImage = cv2.threshold(grayImage, 127, 255,
cv2.THRESH_OTSU)[1] # 공부 필요
thresholdImage = cv2.adaptiveThreshold(thresholdImage, 255,
cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 9, 5)
thresholdImage = cv2.dilate(thresholdImage, kernel, iterations=6) # 팽창
thresholdImage = cv2.erode(thresholdImage, kernel, iterations=20) # 침식

cloneCopy = cloneImage.copy() # 복사본의 복사본 생성
cloneCopy[thresholdImage != 0] = scene[thresholdImage != 0] # 임계치 적용
</pre>
● 배경과 전경을 분리하기 위해 임계치를 적용하여 영상 이진화를 수행함

<pre>
# edge 검출 함수
edges = cv2.Canny(grayImage, 30, 100)

# 커널 생성 후edge 팽창, 침식
kernel = np.ones((4, 4))
edges = cv2.dilate(edges, kernel, iterations=5)
edges = cv2.erode(edges, kernel, iterations=2)
</pre>
● Canny Edge Detection을 통해 영상의 edge를 검출함
● 팽창과 침식 수행 시, 커널의 크기를 조절하여 검출되는 edge의 정도를 다르게 할 수 있음
</td></tr>
</table>

▲ edge가 검출된 이미지

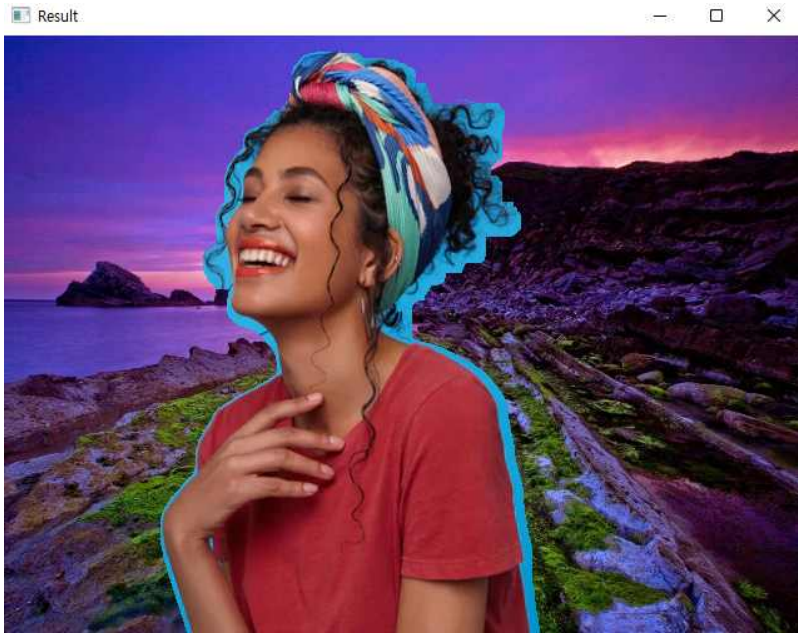
```
# image 외곽선 정보 저장
(cnts, _) = cv2.findContours(edges.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)

dm = np.zeros_like(edges)
if len(cnts) > 0:
# 배경: 0, 전경: 255 => 배경과 전경이 분리된 mask image를 그림
mcnt = max(cnts[:], key=cv2.contourArea)
dm = cv2.fillConvexPoly(dm, mcnt, (255))
```

- 검출된 edge 정보를 통해, edge를 기반으로 한 Contour의 내부를 채워줌 -> 배경과 분리된 mask image를 그림

▲ edge 내부 Contour을 채운 mask image

```
# 배경: 0, 전경: 255
c[dm != 255] = scene[dm != 255] # 전경이 아닌 경우에만 배경 이미지의 화소를 대입함
cv2.imshow("Result", c)
```

- 배경이 0, 전경이 255 -> 전경이 아닌 경우(dm != 255)에만 배경 이미지의 화소를 대입해줌으로써 배경 삽입 가능



▲ 배경에 이미지 화소를 대입한 최종 결과

(지면 부족시 추가하여 작성)

〈 참고 문헌: URL, 도서 등 〉

(!! 스스로 작성하지 않은 코드의 출처는 반드시 밝힐 것 !!)

(참고 내용은 각 요구사항의 접근 방법에 명시할 것)

1) https://faun.pub/real-time-background-changing-in-python-a34511418077