REPORT

“ 디지털영상처리 1차 과제 ”

과 목 명	디지털영상처리
담당교수	김진헌 교수님
학 과	컴퓨터공학과
학 번	2016305078
이 름	최영환
제 출 일	2021.11.06

목 차

Ⅰ. 과제 소개 ····· 1

Ⅱ. 프로그램 개요 ····· 1

Ⅲ. 프로그램 구현 ····· 2

Ⅳ. 결과화면 ····· 5

Ⅴ. 결론 및 후기 ····· 8

Ⅵ. 평가표 ····· 9

< 과제 소개 및 설명 >

1. 영상을 읽어 들여 이 영상의 밝기를 트랙바로 제어하는 모습을 보인다.

2. 원본 영상과 밝게 만든 영상을 나란히 배열하여 출력한다. 좌측 : 원본영상, 우측 : 변환영상

3. 트랙바는 0~30의 값을 가진다. 이는 원본 영상의 10%, 20%, ... , 90%, 300% 만큼 더 밝게 만들어 표현함을 의미한다. 이 처리는 아래와 같이 원본 영상에 대한 곱셈 배수를 곱해서 구현한다.
   10% => 1+0.1, 20% => 1+0.2, 30% => 1+0.3, ... ,90% => 1+0.9, 300% => 1+3.0

4. 트랙바를 오른쪽으로 움직였다가 다시 왼쪽으로 만들어 어둡게 만든다. 이때, 최종 단계에서 0을 선택할 경우, 원본 영상과 같아야 한다.

5. 각 화면의 좌측 상단(50, 50)에 'Original' 과 'Brighter' 문자를 출력해야한다. 이들 문자는 밝기에 영향 받지 않는다.

6. 's' 키 입력 시, 밝아진 영상을 현재의 폴더에 지정된 파일 이름으로 저장한다. 이때, 문자정보('Brighter')는 포함되지 않는다. 형식은 아래와 같다.
   원본 파일의 이름 앞에 'tmpxx_'를 추가로 붙임.(xx 는 트랙바의 값을 의미한다.)
   ex) dark1.png가 원본일 때 tmp00_dark1.png, tmp01_dark1.png, ...,
   tmp29_dark1.png, tmp30_dark1.png,

7. esc 키를 입력하면 종료한다.

< 프로그램 요약 >

※ 본 과제를 구현한 프로그램의 흐름을 요약하면 아래와 같다.

1. 영상을 읽은 후, 트랙바 및 출력 윈도우를 생성한다.

2. 원본영상과 변환영상을 함께 출력하고, 트랙바의 값을 사용하여 밝기를 제어한다.

3. cv.waitKey() 함수를 사용하여 사용자의 키 입력을 기다린다.

4. esc가 입력되면 프로그램을 종료하고, 's' 가 입력되면 변환영상을 저장한다. 두 경우가 아니라면, 영상에 텍스트를 삽입한다.

※ 본 페이지부터는 프로그램에 관한 설명입니다.

< 프로그램 구현 >

1. 영상을 읽은 후, 트랙바 및 출력 윈도우 생성 (+ 라이브러리 import)

```
# 사용된 라이브러리
import cv2 as cv
import numpy as np

# 변수 선언
Path = './'
Name = 'dark1.png'
```

본 과제에서 사용된 라이브러리는 numpy 와 cv2 라이브러리이다. 또한, 과제의 조건이 경로와 파일변수 명을 Path 와 Name으로 사용하는 것이므로, 위와 같이 변수를 선언하였다.

```
# 트랙바의 콜백함수
def onChange(x):
    pass


# 영상 로드
FullName = Path + Name              # 원본 파일의 전체 경로
img = cv.imread(FullName)           # 원본 파일 로드
assert img is not None, 'Failed to load image file!'    # 입력영상 로드 실패. NULL 반환

imgC = img.copy()               # 원본영상
out = img.copy()                # 변환영상
dst = cv.hconcat([imgC, out])   # 출력영상

# 윈도우 생성
cv.namedWindow('image')

# 트랙바 생성
cv.createTrackbar('scale', 'image', 0, 30, onChange)
```

원본 파일의 전체경로표현을 위한 변수로 FullName을 사용하였다. 또한, imgC, out 변수는 각각 원본영상과 변환영상으로 사용되었다. copy() 함수를 사용하여, 각 변수들의 포인터가 다르도록 한다. dst 변수는 최종적으로 출력될 영상으로, 과제에서 요구한 결과를 얻기 위해 cv.hconcat() 함수를 사용하여 나란히 배열한다. 영상 로드 시, 영상의 자료형에 대한 조건은 없었기 때문에 영상의 자료형은 고려하지 않았다. 윈도우를 미리 생성하고, 해당 윈도우에 트랙바를 추가한다. 이때, 트랙바의 콜백함수에서 아무런 처리를 하지 않도록 한 것은 이후 결론 및 후기 부분에 작성해두었다.

< 프로그램 구현 >

2. 원본영상과 변환영상을 함께 출력하고, 트랙바를 통해 밝기를 제어한다.

```
30    # 영상 출력 및 밝기 제어
31    while True:
32        cv.imshow('image', dst)      # 출력 영상
33        outC = out.copy()            # 변환 영상 복사
34
35        # 트랙바 위치 획득
36        scale = cv.getTrackbarPos("scale", "image")
37
38        # 영상의 밝기 변환
39        s = 1 + (scale * 0.1)        # 원영상에 곱해질 곱셈 배수
40        outC = (np.clip(255*(outC/255 * s), 0, 255)).astype('uint8')
```

dst 에 담긴 영상을 출력하고, cv.getTrackbarPos() 함수를 사용하여 트랙바의 값을 scale 변수에 저장하고, 이 값을 사용하여 제시된 예시를 바탕으로 영상의 밝기 변환을 위한 곱셈 배수를 구한다. 제시된 예시는 아래와 같다.

10% => 1+0.1, 20% => 1+0.2, 30% => 1+0.3, ..., 90% => 1+0.9, 300% => 1+3.0

트랙바의 값은 0~30 이므로, 0~300% 에 대입하여 생각한다. 위 식에 근거하여 계산을 했다. 위 과정을 통해 x% => 1 + (0.01 * x) 라는 식을 얻고, 이 식에 트랙바의 값을 적용하여 s = 1 + (0.1 * scale) 이라는 식으로 변환한 다음, 이 값을 다시 변환영상의 복사본 (이하 변환영상)에 곱해서 밝기를 변환한다. 이때, 변환영상을 255로 나누어 정규화하고, 여기에 곱셈배수 s 의 값을 곱하면 영상 각 화소의 값은 1.s배 된 값으로 바뀌게 된다.

현재 영상은 각 화소의 값이 0~1 사이인 부동소수 영상이다. 이를 원래 자료형인 uint8 로 되돌리기 위해, 이 값에 255를 곱하고, np.clip() 함수로 각 화소의 값을 0~255 사이로 제한한 다음, astype() 함수를 통해 원영상과 같이 자료형이 uint8 인 영상으로 바꿔 outC 변수에 저장하였다.

< 프로그램 구현 >

3. cv.waitKey() 함수를 사용하여 사용자의 키 입력을 기다린다.

4. esc가 입력되면 프로그램을 종료하고, 's' 가 입력되면 변환영상을 저장한다. 두 경우가 아니라면, 영상에 텍스트를 삽입한다.

```
42      # 키 입력 대기
43      key = cv.waitKey(1)
44      # s 입력 시 지정된 형식의 이름으로 영상 저장
45      if key == ord('s'):
46          saveName = f'tmp{int(scale):02d}_{Name}'  # 저장 파일의 이름
47          savePath = './' + saveName                # 저장 파일의 전체 경로
48          cv.imwrite(savePath, outC)                # 파일 저장
49          print(f"File name is {saveName}")
50          print(f'Image successfully wrote to {savePath}')
51      # esc 입력 시 프로그램 종료
52      elif key == 27:
53          print('Terminating program...')
54          cv.destroyAllWindows()
55          exit(0)
56      # 영상에 글자 삽입
57      else:
58          org = (50, 50)
59          cv.putText(imgC, "Original", org, 0, 1, (0, 0, 255), 2)
60          cv.putText(outC, "Brighter", org, 0, 1, (0, 0, 255), 2)
61
62      # 최종 출력 영상
63      dst = cv.hconcat([imgC, outC])
64
```

cv.waitKey() 함수를 통해 키 입력을 대기한다. 's' 키 입력 시, 지정된 형식의 이름으로 영상을 현재 폴더에 저장한다. 저장한 파일의 이름과 경로를 출력한다. 과제 공고에서 '현재의 폴더에 저장한다.' 라고 나와 있었으므로, Path 변수가 아닌, 상대경로 './' 를 사용하여 저장 경로를 지정해주었다.

다음으로 esc 입력 시, 프로그램 종료를 출력문을 통해 알리고, 모든 창을 닫고 프로그램을 종료한다.

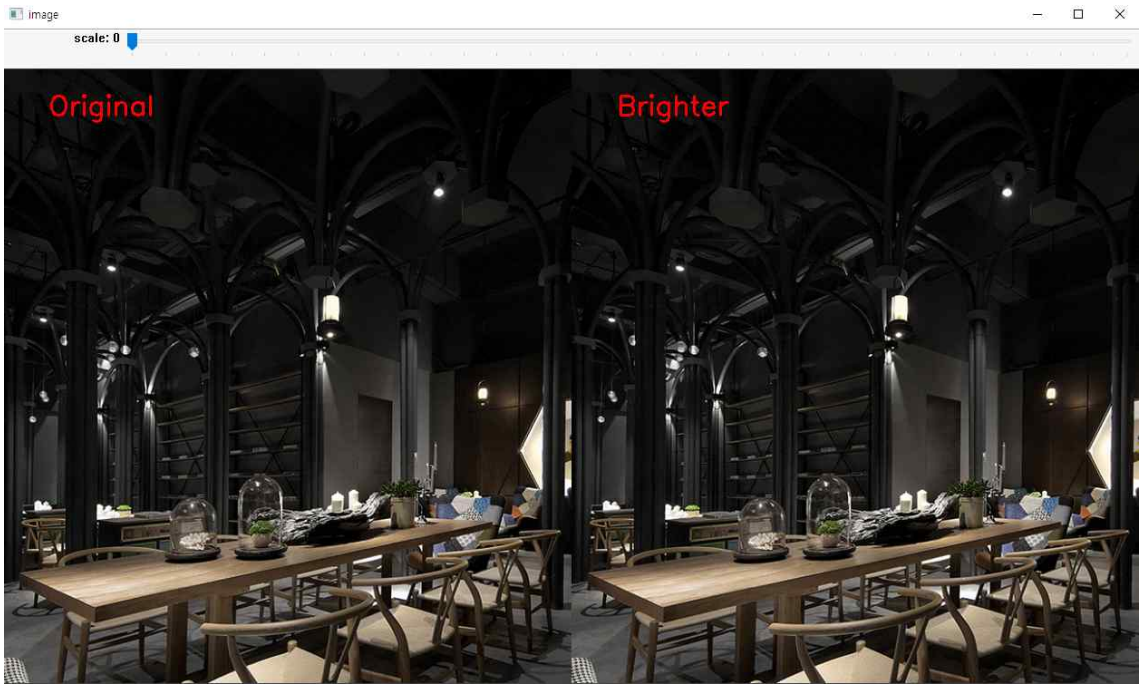
위 두 경우가 아닌 경우, 원본영상과 변환영상 각각에 'Original' 과 'Brighter' 라는 텍스트를 각 영상의 좌측 상단에 출력한다.
위 조건문을 통하여, 텍스트가 추가되기전에 밝기 변환이 끝나기 때문에, 과제의 조건인 '텍스트는 밝기 변화에 영향을 받지 않을 것'이 충족되었고, '텍스트는 제외하고 변환영상만 저장할 것'이 충족되었다.
마지막으로 영상에 변화가 생겼으므로, 최종 출력 영상인 dst 변수에 변경사항이 적용된 영상을 넣어준다. 이로써 프로그램 구현이 완료되었다.

※ 본 페이지부터는 결과 관측입니다.

< 결과 화면 >

1. 트랙바를 0으로 선택했을 때의 처리 결과

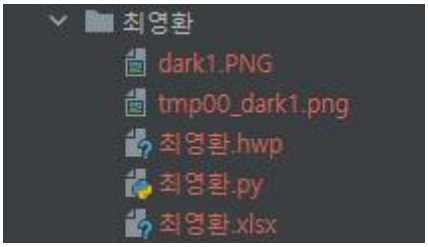
1) 출력 결과



트랙바를 0으로 선택했을 때의 결과화면이다. 트랙바의 값이 0이므로, 곱셈배수는 1 + 0.0 이 된다. 따라서, 밝기 변환 또한 수행되지 않아야하므로, 원본과 같은 영상이 출력되어야 한다. 원본과 변환 영상이 같으며, 텍스트가 추가되어 출력됨을 확인할 수 있다.

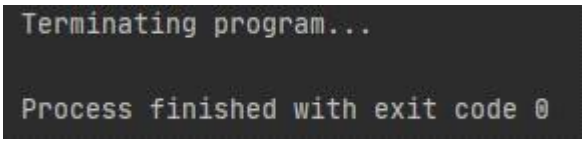
2) 's' 키 입력 전 현재 폴더의 상태

3) 's' 키 입력 후 현재 폴더의 상태



지정된 형식과 같은 이름으로, 현재의 폴더에 저장되었음을 확인하였다.

4) esc 키 입력 시 출력 창

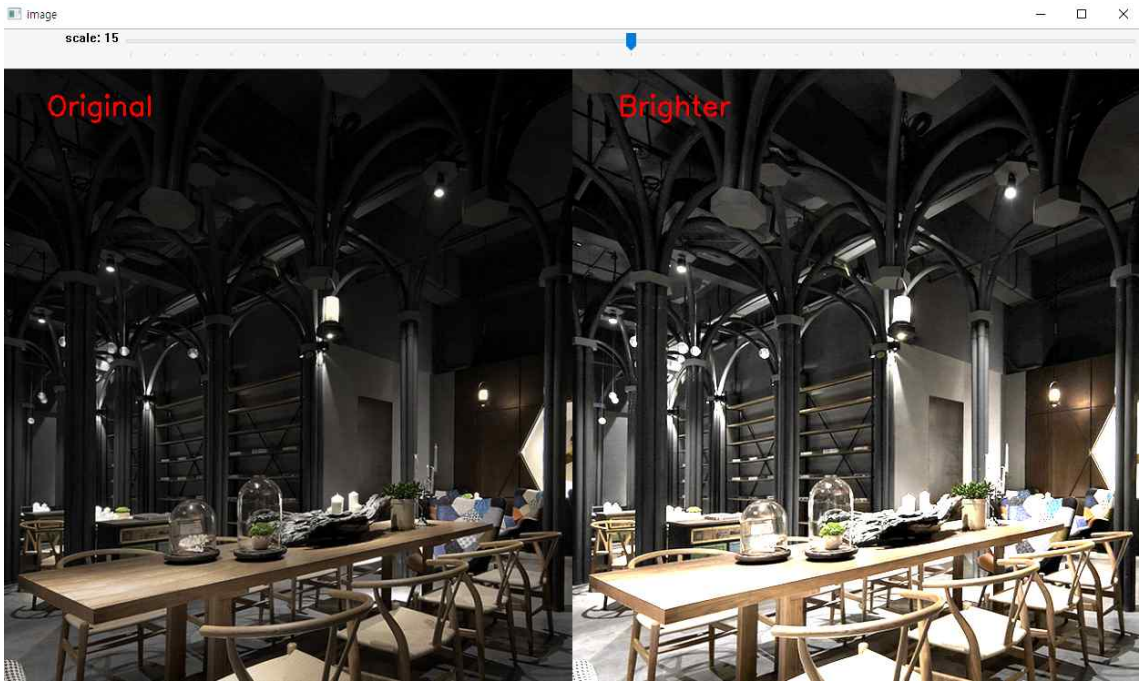


esc 입력 시 프로그램이 종료되었음을 확인하였다.

< 결과 화면 >

2. 트랙바를 15로 선택했을 때의 처리 결과

1) 출력 결과



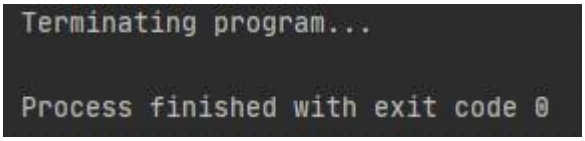
트랙바를 15로 선택했을 때의 결과화면이다. 트랙바의 값이 15이므로, 곱셈배수는 1 + 1.5 가 된다. 따라서, 원 영상보다 더 밝은 영상이 출력되었으며, 텍스트도 추가되어 출력됨을 확인할 수 있다. 앞서 본 식에 의하면 이는 150%이므로, 1.5배 밝아진 영상이라고 추측할 수 있다.

2) 's' 키 입력 전 현재 폴더의 상태

3) 's' 키 입력 후 현재 폴더의 상태

지정된 형식과 같은 이름으로, 현재의 폴더에 저장되었음을 확인하였다.

4) esc 키 입력 시 출력 창

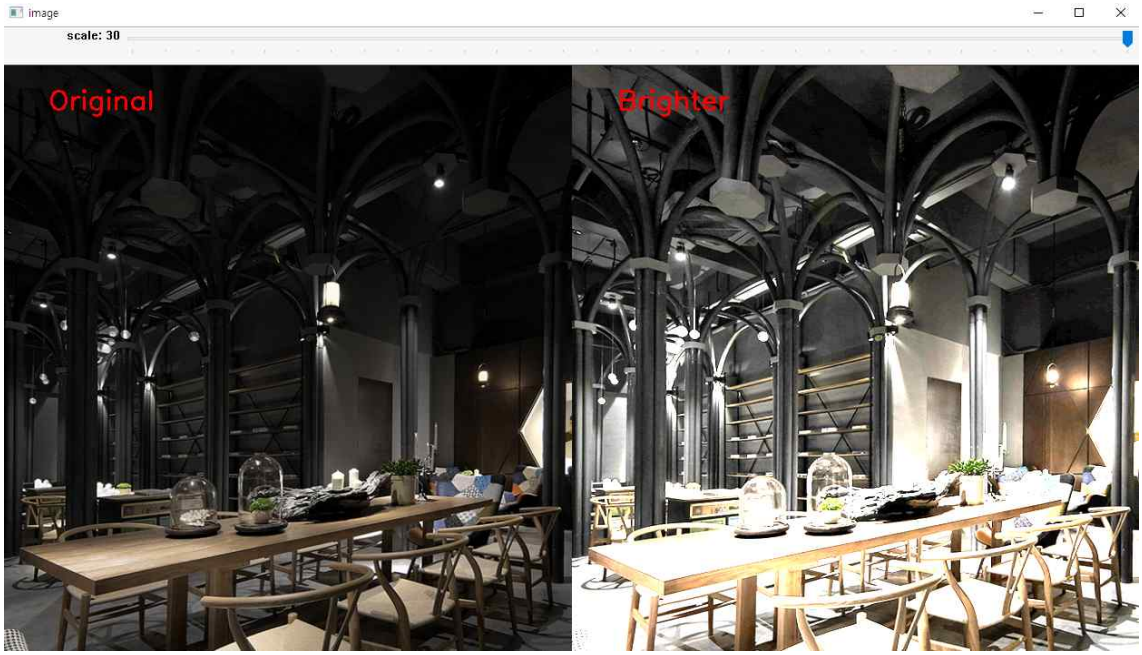


esc 입력 시 프로그램이 종료되었음을 확인하였다.

< 결과 화면 >

3. 트랙바를 15로 선택했을 때의 처리 결과

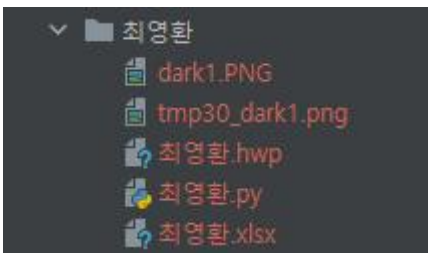
1) 출력 결과



트랙바를 30으로 선택했을 때의 결과화면이다. 트랙바의 값이 30이므로, 곱셈배수는 1 + 3.0 이 된다. 따라서, 원 영상보다 더, 앞서 15를 선택한 경우보다 더 밝은 영상이 출력되었다.
앞서 본 식에 의하면 300%이므로, 3배 밝아진 영상이라고 추측할 수 있다.

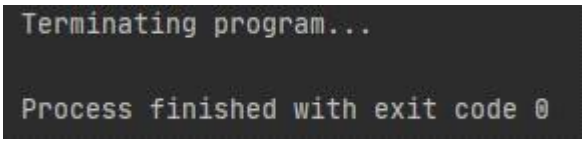
2) 's' 키 입력 전 현재 폴더의 상태

3) 's' 키 입력 후 현재 폴더의 상태



지정된 형식과 같은 이름으로, 현재의 폴더에 저장되었음을 확인하였다.

4) esc 키 입력 시 출력 창



esc 입력 시 프로그램이 종료되었음을 확인하였다.

※ 결과 관측에서 사진과 글로만 작성된 보고서를 통해서는 명확하게 설명 불가능한 부분이 있다고 판단되어, 영상에서 상세히 설명하였습니다.

< 결론 및 후기 >

1. 결론

앞서 프로그램 구현 부분에서 기술한 바와 같이, 밝기 조절 과정에서 과제에서 주어진 예시를 참고하여, 원본 영상에 곱해야 할 곱셈배수가 1 + (트랙바의 값 * 0.1) 이라는 사실을 얻어내었고, 이 곱셈배수를 이용하여 밝기 조절을 하기 위해, 기존에 배포되었던 예제 프로그램을 참조하여, out = (np.clip(255*(img/255 * s), 0, 255)).astype('uint8') 와 같이 코드를 작성했다. (org : 변환영상, img : 원본영상) 이 과정을 간단하게 예시를 들어 설명하자면 아래와 같다.

c1	c2	c3
c4	c5	c6
c7	c8	c9

위와 같은 영상이 있다고 가정한다. (예시이고, 간략하게 해야하므로 3x3으로 표현하였다.)
c1 ~ c9 은 각 픽셀의 계조값이며, 자료형은 0 ~ 255의 값을 갖는 uint8 의 정수형으로 가정하였다.
c1 ~ c9를 각각 255로 나누어주면 각 값은 0 ~ 1 사이의 부동소수가 되고, 이 값에 위에서 얻은 곱셈배수를 곱하면 아래와 같아진다. (각 값에 대한 연산은, numpy array 연산에 의하여 array * (곱셈배수) 를 하면 자동으로 처리된다.)

c1 * s / 255	c2 * s / 255	c3 * s / 255
c4 * s / 255	c5 * s / 255	c6 * s / 255
c7 * s / 255	c8 * s / 255	c9 * s / 255

위와 같은 연산이 시행됨에 따라, 현재 각 값은 0 ~ 1 사이의 값이 아닐 수도 있으며, 원본영상과 자료형도 맞지 않는 상태이다. 따라서, 각 값에 다시 255를 곱하고, np.clip() 함수와 .astype() 함수를 사용하여 각 값을 0 ~ 255 사이의 값을 갖는 uint8 형의 영상으로 변환하면, 목표했던 바와 같이 원 영상에 비해 밝아진다. (과제 공고에서 주어진 예시에 의하면, (트랙바의 값 * 10)% 만큼 밝아진다.) 트랙바의 값이 15라면, 변환영상은 1.5배 밝아진 영상이라고 추측할 수 있다.

위 과정을 통하여 밝기 조절에 성공하였으며, 자료형도 원본과 맞게 되돌려 주었으므로, 원본 영상과 같이 출력하여도 오류가 발생하지 않는다. (정수형으로 되돌리는 과정을 하지 않을 경우 원본영상과 이어붙여서 저장할 때(cv.hconcat() 함수 사용 시) 오류가 발생한다.)

< 결론 및 후기 >

2. 후기

(1) 영상파일에 텍스트 추가 관련 후기

과제의 조건 중 하나로, 이에 대한 처리를 위하여 최초에는 변환 영상에 글자를 넣기 전에 따로 저장에 쓰일 복사본을 생성하고, 그 영상을 저장하도록 하였는데, 이 방법은 코드의 가독성이 떨어지고, 사용하지 않는 복사본이 사용된다는 생각이 들었다. 따라서, 앞서 프로그램 구현에서 기술한 바와 같이 지정된 키 입력인 's' 와 esc 키 미입력 시, 텍스트를 삽입하도록 변경하였다. 추가된 텍스트가 밝기에 영향을 받지 않는 것은 비교적 쉽게 처리가 되었다. 단순히 밝기 조절 처리 이후에 텍스트를 삽입하면 되는 것이었다.

(2) 트랙바의 콜백함수 처리 관련 후기

앞서 프로그램 구현 부분에서 기술한 바와 같이, 트랙바의 콜백함수는 아무것도 처리하지 않도록 하였는데, 최초 프로그램 구상 당시, 트랙바의 값을 자동으로 얻어오는 콜백함수를 통해 영상에 관한 처리를 다 하려고 하였으나, 밝기 처리에 대한 소스코드는 짧고, 프로그램의 크기 자체가 그리 크지 않고, 위와 같이 콜백함수로 모든 동작을 처리하는 것에서 여러가지 문제점이 발생하였다. (ex. 트랙바의 위치만 이동하고, 트랙바의 값을 얻지 못하는 문제, 텍스트 추가 관련 처리가 원할하지 않은 문제 등등...) 따라서, 콜백함수를 통하여 트랙바의 값을 조정하는 것 대신, 현재 트랙바의 값을 얻어오는 getTrackbarpos() 함수를 통하여 트랙바의 값을 얻고, while 루프 내에서 이를 처리하는 것이 더 간단하고 효율적이라는 판단하에 위와 같이 추가 함수 제작이나, 트랙바의 콜백함수는 아무런 처리도 하지 않도록 프로그램을 제작하였다.

(3) 최종 후기

예제 프로그램을 참조하였으며, 또한 프로그램의 규모 자체도 크지 않아 큰 어려움 없이 제작이 가능했다고 판단된다.

< 평가표 >

※ 각 미션에 대한 평가만 작성하였습니다.

1. 영상을 읽어 들여 이 영상의 밝기를 트랙 바로 제어하는 모습을 보인다. 답변 : 프로그램 구현에서 해당 과정을 상세히 설명하였습니다.
2. 원본 영상과 밝게 만든 영상을 나란히 배열하여 출력한다. 좌측: 원본 영상, 우측: 밝게 만든 영상. 답변 : 프로그램 구현 부분에서 해당 과정을 상세히 설명하였으며, 출력 결과에서 해당 부분을 확인하였습니다.
3. 트랙 바는 0~30의 값을 갖는다. 이는 원본 영상의 10%, 20%, 30% ... 90%, 300% 만큼 더 밝게 만들어 표현함을 의미한다. 이 처리는 아래와 같이 원본 영상에 대한 곱셈 배수를 곱해서 구현 한다. 답변 : 프로그램 구현 부분에서 해당 과정을 상세히 설명하였으며, 출력 결과에서 해당 부분을 확인하였습니다.
4. 트랙 바를 오른쪽으로 움직였다가 다시 왼쪽으로 만들어 어둡게 만들면 최종 단계에서 0을 선택하면 원본 영상과 같아야 한다 답변 : 출력 결과에서 해당 부분을 확인하였으며, 영상에서 더 명확하게 확인이 가능합니다.
5. 각 화면의 좌측 상단(50, 50)에 'Original'과 'Brighter' 문자를 출력해야한다. 이들 문자는 밝기에 영향 받지 않는다. 답변 : 프로그램 구현 부분에서 해당 과정을 상세히 설명하였으며, 출력 결과에서 해당 부분을 확인하였습니다.
6. key 's'를 누르면, 밝아진 영상을 현재의 폴더에 지정된 파일이름으로 저장한다. 이때 문자정보('Brighter')는 저장 영상에 포함되지 않는다. 답변 : 프로그램 구현 부분에서 해당 과정을 상세히 설명하였으나, 출력 결과에서 해당 부분에 대한 명확한 확인은 불가하므로, 영상에서 해당 내용을 명확히 확인할 수 있습니다.
7. esc 키를 입력하면 종료한다. 답변 : 프로그램 구현 부분에서 해당 과정을 상세히 설명하였으나, 출력 결과에서 해당 부분에 대한 명확한 확인은 불가하므로, 영상에서 해당 내용을 명확히 확인할 수 있습니다.