영상처리 프로젝트
최종 보고서

SINCE 1947
SEO KYEONG
UNIVERSITY

제출일	2024. 12. 13	**전공**	소프트웨어학과
과목	영상처리 프로그래밍	**학번**	2020301081
담당교수	장문수 교수님	**이름**	허종우

1. 개요

1.1 프로젝트 개요

1.1.1 개요

이 프로젝트는 OpenCV 라이브러리와 Numpy 모듈을 활용해 이미지를 읽고, 해당 이미지에 다양한 필터 효과를 적용하거나 밝기, 명암도를 조절하여 최종적으로 이미지 보정 기능을 수행할 수 있도록 하는 데 목적을 두고 있다. 키보드와 트랙바 이벤트를 통해 손쉽게 기능을 적용할 수 있고, 이후 편집된 이미지를 새로운 이미지 파일로 저장하는 기능을 포함하고 있다.

1.2 프로젝트 구상

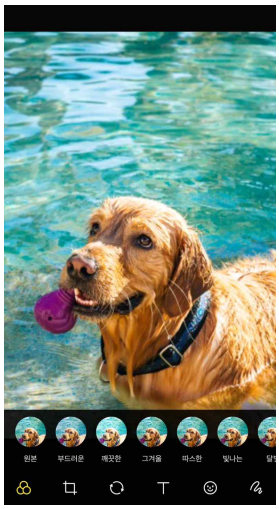
1.2.1 주제 선정

- 프로젝트 주제: 사진 보정 도구 만들기
    1) 사진 보정의 목적은 이미지를 시각적으로 더욱 훌륭하게 만드는 것
    2) 보정은 풍경, 인물 등 다양한 대상을 대상으로 함
    3) 이미지에 배경효과를 입히고, 밝기와 명암도를 조절하는 프로그램

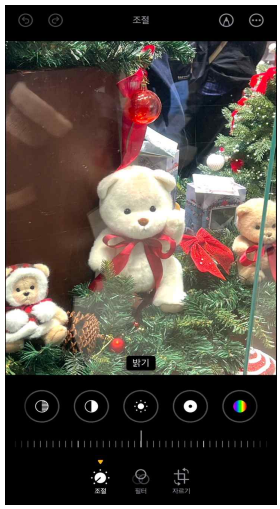
1.2.2 배경

- 아이디어: 사진 어플 또는 카카오톡 어플의 프로필 및 배경 편집 시 적용 가능한 배경효과 입히기와 밝기 및 명암도 조절 기능을 구현

예시 사진 1



예시 사진 2



- 효과: 1) 이미지에 필터를 입혀 특정 효과를 강조
    2) 이미지의 밝기를 높이거나 줄여 이미지 감상에 시각적으로 도움
    3) 명암도를 조절해 이미지 내 물체를 서로 구별할 수 있도록 함

1.2.3 사용 대상자

이 프로그램의 주요 기능은 이미지 배경효과 적용 및 밝기, 명암도 조절이다. 스마트폰이나 디지털카메라 등으로 촬영한 이미지를 소유한 사용자 누구나 사용할 수 있다.

- 사용자: 1) SNS 프로필 및 배경이미지에 특수한 효과를 더하고자 하는 사용자
    2) 이미지 필터 효과 및 편집 기능에 대해 학습하고자 하는 교육 분야 종사자
    3) 간단한 이미지 보정을 원하는 사용자

2. 구현 과정

2.1 초기 설정

```
# 이미지 읽기
image = cv2.imread("image/bear.jpg")
if image is None:
    raise Exception("영상파일 읽기 오류")
```

- OpenCV 라이브러리의 cv2.imread() 함수를 사용해 이미지 파일을 읽어오고, 해당 경로에 이미지가 존재하지 않는다면 “영상파일 읽기 오류” 예외문을 출력한다.
- 이미지 경로는 “image/파일명.jpg”로, 현재 이미지 파일 기준 상대경로다. 상대경로는 주소나 프로젝트 폴더의 위치가 바뀌더라도 내부 구조에 변함이 없다면 코드의 수정 없이 그대로 사용할 수 있다는 장점이 있다.

```
# 초기 설정
mode = None
scale = 0  # 트랙바 초기값
brightness = 50 # 밝기 초기값
contrast = 50   # 명암도 초기값
```

- 위 프로그램에는 3가지 모드가 있다. ‘f’는 배경효과 적용, ‘b’는 밝기 조절, ‘c’는 명암도 조절이다. 프로그램을 실행시킨 초기에는 이미지에 아무런 효과가 적용되지 않은 상태이므로, “현재 적용된 모드는 없음.”을 나타내기 위해 None을 mode 변수에 할당했다.
- 각 필터 효과의 강도를 트랙바에 변수 scale로 표현, 초기 이미지에는 아무런 필터가 적용되지 않은 상태이므로 변수 scale을 0으로 초기화
- 밝기, 명암도를 초기값 50에서 트랙바를 왼쪽으로 이동시키면 감소, 오른쪽으로 이동시키면 증가시키도록 만들고자 변수 초기화

```
# 텍스트 표시 위치 계산 및 출력
line_spacing = 30  # 줄 간격
start_y = 30  # 첫 번째 줄의 Y 위치
x = 20  # X 위치

# 도움말 창 생성시 텍스트의 색상 적용을 위해 선언
white = (255, 255, 255)
black = (0, 0, 0)
navy = (128, 0, 0)

# 도움말 창 생성
hp_win = np.full((200, 450, 3), white, np.uint8)
hp_win2 = np.full((250, 400, 3), white, np.uint8)
font = cv2.FONT_HERSHEY_TRIPLEX;
```

- 텍스트 표시를 위해 필요한 변수 (좌표 위치, 간격, 색상, 글씨체 등) 선언 및 도움말 창 생성

```
First_Text = [
    "Press Buttons to adjust effects",
    "F. Filtering",
    "B. Brightness",
    "C. Contrast",
    "ESC. Exit Program & Save"
]
```

```
Second_Text = [
    "Filtering Effects",
    "1. Black & White",
    "2. WarmTone",
    "3. CoolTone",
    "4. Pencil Sketch",
    "5. Sharpening",
    "6. Cartoon",
    "Others. Original",
    "ESC. Exit Program & Save"
]
```

- 두 개의 도움말 창에 표시할 텍스트 내용을 리스트 형식으로 생성함
- First_Text는 초기 실행 시 주요 단축키에 대한 설명을 hp_win으로 나타내고, Second_Text는 적용할 배경효과들을 hp_win2로 나타낸다.

2.2 구현된 기능 세부 설명

2.2.1 도움말 출력

```
# cv2.putText는 단일 줄만 출력가능, for 문 사용
for i, line in enumerate(First_Text):
    y = start_y + i * line_spacing  # Y 위치 계산
    if i == 0:
        color = navy
    else:
        color = black  # 첫 번째 줄만 강조 색상
    cv2.putText(hp_win, line, (x, y), font, 0.7, color, 1)
```

```
for i, line in enumerate(Second_Text):
    y = start_y + i * line_spacing  # Y 위치 계산
    if i == 0:
        color = navy
    else:
        color = black  # 첫 번째 줄만 강조 색상
    cv2.putText(hp_win2, line, (x, y), font, 0.7, color, 1)
```

- 텍스트의 출력 위치 조정을 위해 변수 y에 연산 과정을 더해 값에 변환을 주었다.
- i는 각 항목의 인덱스를, line은 각 항목의 문자열을 나타낸다.
- enumerate('텍스트명') 함수는 리스트를 순회하며 (인덱스, 문자열 값)을 반환한다.
- cv2.putText()는 한 번에 한 줄씩만 출력하므로, for 반복문을 통해 First_Text와 Second_Text의 도움말 창 내용을 제목은 강조된 남색, 내용은 검은색으로 반복 출력하였다.
- cv2.putText()의 매개변수 hp_win, hp_win2은 각각 문자열을 작성할 대상 행렬(영상)을 의미, line은 First_Text를 for 문으로 반복하며 작성할 문자열을 의미, (x, y)는 문자열의 시작 좌표, font는 문자열의 폰트, 0.7은 글자 크기 확대 비율, color는 글자의 색상, 1은 글자의 굵기이다.

2.2.2 밝기 및 명암도, 필터 강도 조절

```
def set_Brightness(val):
    global brightness
    brightness = val
```

```
def set_Contrast(val):
    global contrast
    contrast = val
```

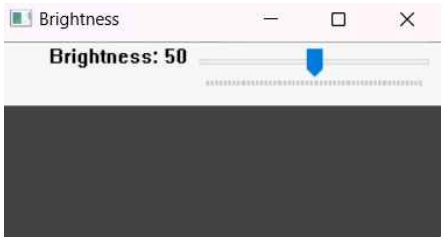
```
def set_Scale(val):
    global scale
    scale = val
```

- set_Brightness(val) 함수는 트랙바에서 입력되는 밝기값을 brightness 전역 변수에 저장한다.
- set_Contrast(val) 함수는 트랙바에서 입력되는 명암도값을 contrast 전역 변수에 저장한다.
- set_Scale(val) 함수는 트랙바에서 입력되는 필터의 강도를 scale 전역 변수에 저장한다.

2.2.3 밝기 및 명암도 적용

```
# 밝기 및 명암도 적용 함수
def btct(image):
    beta = brightness - 50
    alpha = contrast / 50
    return cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
```

- 함수 btct는 밝기, 명암도를 조절하는 함수로, 배경효과를 적용할 때에도 이전에 적용한 밝기와 명암도를 유지하기 위해 선언했다. set_함수에서 저장되는 brightness와 contrast 값을 할당해 이미지를 조정한다.
- 일반적으로 OpenCV에서 트랙바는 슬라이더 형식으로 나타나며, 정수 값 범위에서 동작한다. 밝기와 명암도에 대한 트랙바 값은 brightness와 contrast 변수에 저장되며, 두 변수의 초기값은 각각 50과 50으로 초기 설정 단계에서 초기화했다.



- cv2.convertScaleAbs는 주로 image의 밝기, 명암도 조절에 사용되는 함수이고, 기본 구문은 다음과 같다.
- dst = cv2.convertScaleAbs(src[, dst[, alpha[, beta]]]). src와 dst는 각각 입/출력 이미지를 의미하고, alpha(기본값 1)는 명암도, beta(기본값 0)는 밝기를 조절하는 매개변수이다.
- alpha = contrast / 50의 연산을 통해 명암도 트랙바 값이 50(초기값)일 경우 alpha = 1.0으로, 이는 기본값에 해당에 명암도에 변화가 없다. 마찬가지로 beta = brightness − 50의 연산을 통해 밝기의 트랙바 값이 50(초기값)일 경우 beta = 0으로, 이는 기본값에 해당해 밝기에 변화가 없다. 두 변수의 값은 트랙바의 값이 0~100으로 이동할 때 각각 변화가 생기도록 구현했다. 이는 사용자가 현재 밝기, 명암도의 강도를 쉽게 이해할 수 있도록 하는 효과가 있다.

2.2.4 재설정 함수

1) 필터의 강도 재설정

```
def reset_scale():
    global scale
    if mode == "Scale":
        scale = 0
        cv2.setTrackbarPos(mode, "Scale", 0)
```

- 현재 모드의 이름이 Scale일 경우 값을 0으로 초기화
- 밝기, 명암도 조절 과정에서 숫자 0~6을 누를 경우에서 프로그램의 비정상적 종료를 방지

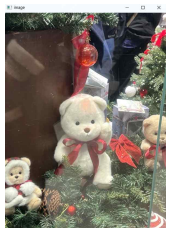
2) 트랙바 재설정

```
# 모드 변경 시 적용값 초기화
def reset_trackbar():
    global mode
    if mode:
        cv2.destroyWindow(mode)
        mode = None
```

- 모드 변경 시 이전에 생성된 트랙바 창을 닫고, 새로운 모드에 맞는 트랙바를 띄우기 위한 과정

2.2.5 필터 적용 함수

```
def filtering(num, intensity):
    global filtered_image    # 전역변수
    scale = intensity / 100
    btct_image = btct(image)
```



원본 이미지

- 이미지에 필터 효과를 적용하기 위해 구현한 'filtering'함수이다. 이는 키보드의 숫자를 나타내는 num과 현재 필터 적용 강도를 나타내는 intensity 두 변수를 매개변수로 갖는다.
- 각 필터마다 효과가 적용된 이미지를 나타내기 위해 filtered_image를 전역 변수로 선언했다.
- 트랙바에 적용되는 강도 intensity를 cv2.addWeighted()함수의 alpha 값에 할당하기 위해, intensity / 100을 다시 scale 변수로 선언한다. intensity의 값은 트랙바의 값 0~100을 scale alpha 값 0~2.0으로 나타낸다.
- cv2.addWeighted() 함수를 통해 두 이미지를 합성하고, 배경 필터 적용 강도를 트랙바로 조절하는 것이 가능하도록 구현, 연산된 값을 filtered_image 변수에 할당
- 밝기, 명암도 적용을 유지하기 위해 btct() 함수가 적용된 이미지를 btct_image 변수로 선언

```
filtered_image = cv2.addWeighted(src1, alpha, src2, beta, gamma[, dst[, dtype]])
```

- cv2.addWeighted() 함수는 영상의 산술 연산 함수 중 하나로, 기본 구문은 위 그림과 같다. src1, src2는 각각 합성할 두 이미지 영상을 의미, 서로 같은 크기 및 타입을 가져야 한다. alpha는 첫 번째 영상 가중치, beta는 두 번째 영상 가중치를 의미한다. gamma는 결과 영상에 추가로 더할 값을 나타내고, dst와 dtype은 현재 프로그램 코드에서는 모두 생략되었다.

1) 흑백 효과 적용

```
if num == '1':  # 흑백 변환
        gray = cv2.cvtColor(btct_image, cv2.COLOR_BGR2GRAY)
        gray_colored = cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR)
        filtered_image = cv2.addWeighted(btct_image, 1 - scale, gray_colored, scale, 0)
```

- 이미지를 그레이스케일(흑백)로 변환한 후 이를 다시 컬러로 변환함. 이후 트랙바로 흑백, 원본의 혼합 정도를 조절할 수 있다.
- gray 이미지는 2차원 단일 채널 이미지로, 컬러 3채널 이미지인 btct_image 변수와 바로 결합할 수 없다. 이를 해결하기 위해 기존 이미지를 흑백으로 변환한 후 다시 컬러 형식으로 변환했다.
- 기존 컬러 이미지에서 scale의 강도에 따라 점점 흑백 이미지로 변환한다.

2) 따듯한 효과(웜톤) 적용

```
elif num == '2':  # 웜톤
        bgr = list(cv2.split(btct_image))
        bgr[2] = cv2.add(bgr[2], intensity)
        bgr[1] = cv2.add(bgr[1], intensity / 3)
        warmTone = cv2.merge(bgr)
        filtered_image = cv2.addWeighted(btct_image, 1 - scale, warmTone, scale, 0)
```

- cv2.split() 함수는 BGR 채널을 각각 분리하여 튜플로 변환함. 그러나 튜플은 불변 데이터 타입이므로, 이를 리스트로 변환해 가변 데이터 타입으로 만든 후 각 B, G, R 인덱스에 수정된 값을 적용해 따듯한 효과를 만들었다. 주황색 색상의 RGB 픽셀값은 (255, 165, 0)으로, Red와 Green 채널의 픽셀값을 수정해 웜톤 효과 구현

3) 차가운 효과(쿨톤) 적용

```
elif num == '3':  # 쿨톤
        bgr = list(cv2.split(btct_image))
        bgr[0] = cv2.add(bgr[0], intensity)
        bgr[1] = cv2.add(bgr[1], intensity - 10)
        coolTone = cv2.merge(bgr)
        filtered_image = cv2.addWeighted(btct_image, 1 - scale, coolTone, scale, 0)
```

- 웜톤 효과와 같은 방식으로 Blue, Green 채널 값을 수정해 쿨톤 효과를 구현함

4) 연필 스케치 효과 적용

```
elif num == '4':  # 연필 스케치 필터
        gray = cv2.cvtColor(btct_image, cv2.COLOR_BGR2GRAY)
        gray_invert = 255 - gray
        gaus_img = cv2.GaussianBlur(gray_invert, (35, 35), sigmaX=0, sigmaY=0)
        pencil = cv2.divide(gray, 255 - gaus_img, scale=255)
        pencil_colored = cv2.cvtColor(pencil, cv2.COLOR_GRAY2BGR)
        filtered_image = cv2.addWeighted(btct_image, 1 - scale, pencil_colored, scale, 0)
```

- 위 효과는 다음 영상을 참고해 구현했다.(https://youtu.be/Cy7FDbbIzZM?si=BabWja6hmLBEG1fW)





image gray gray_invert gaus_img pencil

- 연필 스케치 효과의 핵심 아이디어는 흑백 이미지와 반전 및 블러 이미지를 활용한 나눗셈 연산이다. 기존 흑백 이미지를 흑백 반전과 블러가 적용된 이미지로 나누어 선명한 윤곽선을 생성하고, 나머지 어두운 영역은 흐릿하게 처리하도록 연산해 스케치 효과를 구현했다.

```
gaus_img = cv2.GaussianBlur(gray_invert, (35, 35), sigmaX=0, sigmaY=0)
```

- gaus_img는 흑백을 반전시킨 이미지에 가우시안 블러를 적용해 반전 이미지의 경계를 부드럽게 처리하여 노이즈를 제거하고 자연스러운 스케치 효과를 만드는 데 도움을 준다. (35, 35)는 가우시안 블러의 크기이고, 이 값이 커질수록 블러 효과가 더욱 넓고 부드럽게 처리되어 스케치효과가 강조된다.

```
pencil = cv2.divide(gray, 255 - gaus_img, scale=255)
```

- cv2.divide(src1, src2, scale)은 두 이미지를 픽셀 단위로 나누는 연산을 수행한다. src1은 gray 이미지이고, src2는 255 - gaus_img이다. 밝은 영역은 gray 이미지의 값이 상대적으로 크기 때문에 나눗셈 이후 값이 더 커진다. 어두운 영역은 255 - gaus_img의 값이 커지므로 나눌 때 값이 더욱 작아진다. scale = 255는 나눗셈 결과를 0~255 범위로 맞추기 위해 사용하였다.

```
pencil_colored = cv2.cvtColor(pencil, cv2.COLOR_GRAY2BGR)
```

- 이후 2차원 컬러 채널을 GRAY2BGR 연산을 통해 3차원으로 만들어 pencil_colored 변수에 할당했다. 기존 컬러 이미지로부터 scale 값 조정을 통해 스케치효과가 적용된다.

5) 선명한 효과(샤프닝) 적용

```
elif num == '5':  # 샤프닝
        sharpen_kernel = np.array([[-1, -1, -1],
                                   [-1,  9, -1],
                                   [-1, -1, -1]])
        sharpened_image = cv2.filter2D(btct_image, -1, sharpen_kernel)
        filtered_image = cv2.addWeighted(btct_image, 1 - scale, sharpened_image, scale, 0)
```

- 샤프닝 효과는 커널의 크기를 직접 설정하고, cv2.filter2D() 함수를 통해 적용했다. 트랙바가 오른쪽으로 점점 이동함에 따라, 그 효과가 더욱 강조되어 선명한 이미지가 나타난다.

6) 카툰화

```
elif num == '6':  # 카툰화
        gray = cv2.cvtColor(btct_image, cv2.COLOR_BGR2GRAY)  # 그레이스케일 변환
        gray = cv2.GaussianBlur(gray, (9, 9), 0)  # 가우시안 필터 -> 잡음 제거
        edges = cv2.Laplacian(gray, -1, None, 5)  # 라플라시안 필터 -> 엣지 검출
        _, sketch = cv2.threshold(edges, 70, 255, cv2.THRESH_BINARY_INV)
        blurred = cv2.medianBlur(btct_image, 5)  # Median blur로 부드럽게 처리
        cartoon_image = cv2.bitwise_and(blurred, blurred, mask=sketch)
        filtered_image = cv2.addWeighted(btct_image, 1 - scale, cartoon_image, scale, 0)
```

- 위 효과는 다음 블로그를 참고하여 구현했다.

(https://my-coding-footprints.tistory.com/156), (https://charlezz.com/?p=45040)




edges sketch blur cartoon

- 카툰 효과는 흰 바탕에 검은색 경계선으로 엣지 검출된 이미지와 블러 효과가 적용된 이미지를 서로 bitwise_and 행렬 비트 논리곱 연산하여 구현했다.

```
gray = cv2.cvtColor(btct_image, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (9, 9), 0)
edges = cv2.Laplacian(gray, -1, None, 5)
```

- 엣지 검출을 위해 기존 이미지를 그레이스케일로 변환해 gray 변수에 저장. 이후 gray에 가우시안 블러를 적용해 이미지의 작은 잡음들을 제거, 윤곽선을 부드럽게하는 효과를 적용함. (9, 9)는 커널의 크기로, 크기가 클수록 더 부드럽게 처리되지만 디테일이 감소한다.
- 라플라시안 필터를 통해 흑백이미지에서 엣지를 검출한다. 밝기가 급격히 변하는

영역(윤곽선)을 강조. -1은 출력 이미지와 입력 이미지의 데이터 타입을 동일하게 설정한다는 의미이고, None은 Scale 매개변수에 아무런 값을 지정하지 않고 그대로 사용한다는 의미이다. 5는 라플라시안 필터 커널의 크기를 의미한다.

```
_, sketch = cv2.threshold(edges, 70, 255, cv2.THRESH_BINARY_INV)
blurred = cv2.medianBlur(btct_image, 5)  # Median blur로 부드럽게 처리
cartoon_image = cv2.bitwise_and(blurred, blurred, mask=sketch)
filtered_image = cv2.addWeighted(btct_image, 1 - scale, cartoon_image, scale, 0)
```

- 이후 검출된 엣지를 이진화 과정을 거쳐 효과를 더욱 강조하고, 검은색 영역과 흰색 영역을 cv2.THRESH_BINARY_INV를 사용하여 반전시켜 sketch 변수에 해당 이미지를 할당한다. cv2.threshold() 함수는 ret, dst 두 개의 값을 반환한다. ret은 적용된 임계값, dst는 임계값이 적용된 결과 이미지를 나타낸다. 적용된 임계값이 없다는 의미의 '_'로 표현
- 기존 이미지를 메디안 블러를 통해 블러 효과를 적용해 blurred 변수에 할당하고, 블러링된 이미지에 sketch 이미지의 엣지를 마스킹 복사, 이후 비트 연산을 통해 하나의 이미지로 만든다.

```
else:  # 필터가 없으면 원본 이미지 유지
        filtered_image = btct_image.copy()
```

- 이미지에 적용된 필터가 없거나, 1~6 이외 다른 숫자 키를 누르는 경우 원본을 유지한다.

2.3 주요 실행 코드

2.3.1 키보드 & 트랙바 이벤트

```
while True:
    key = cv2.waitKeyEx(100)
    if key == 27:       # esc 누르면 종료
        break

    # 필터링 적용
    filtering(current_filter, scale)
    cv2.imshow(title, filtered_image)
```

- 이미지에 filtering 함수 적용 및 이미지 띄우기, 'ESC' 키를 누르면 반복 종료.
- ESC 키를 누르기 전엔 지속적으로 이미지를 띄우고, 이벤트에 따른 효과 적용
- scale 값은 필터 효과를 조정하는 cv2.addWeighted()에서 사용한다.
- 트랙바로 scale 값을 조정하고, 변경되는 값이 동적으로 filtering() 함수에 전달 및 필터 적용

```
# 키보드 이벤트 처리
    if ord('0') <= key <= ord('6'):  # 필터 변경
        current_filter = chr(key)  # 현재 필터 업데이트
        reset_scale()
```

- ord() 함수와 chr() 함수를 사용해 키보드 0~6키 입력 시 필터가 적용되도록 함
- 필터가 변경될 때 마다 이전 필터에 적용된 강도 값을 초기화

```
elif key == ord('f'):
        reset_trackbar()
        mode = "Scale"
        cv2.namedWindow(mode)
        cv2.imshow(mode, hp_win2)
        cv2.createTrackbar(mode, "Scale", scale, 100, set_Scale)
```

- 'f' 키를 누를 경우 현재 모드의 이름을 Scale로 바꾸고, 트랙바를 리셋. 필터링 안내문 hp_win2를 함께 출력, 이후 필터링 강도를 조절할 "Scale" 트랙바를 안내문 이미지와 함께 출력한다.

```
elif key == ord('b'):
    reset_trackbar()
    mode = "Brightness"
    cv2.namedWindow(mode, cv2.WINDOW_NORMAL)
    cv2.createTrackbar(mode, "Brightness", brightness, 100, set_Brightness)

elif key == ord('c'):
    reset_trackbar()
    mode = "Contrast"
    cv2.namedWindow(mode, cv2.WINDOW_NORMAL)
    cv2.createTrackbar(mode, "Contrast", contrast, 100, set_Contrast)
```

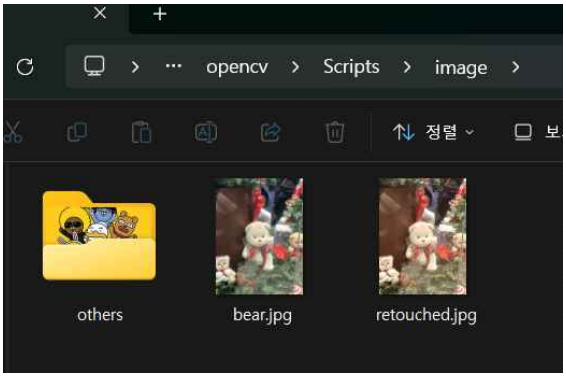
- 'b' 키를 누를 경우 현재 모드의 이름을 Brightness로 바꾸고, 트랙바를 리셋한다. 이후 밝기를 변경할 수 있는 트랙바 윈도우창이 띄워진다. 슬라이더의 위치로 동적으로 할당되는 밝기값을 최대값 100까지 반영한다. 트랙바를 옮길때마다 set_Brightness 함수가 콜백되어 값이 전달된다.

- 'c' 키를 누를 경우 현재 모드의 이름을 Contrast로 바꾸고, 트랙바를 리셋한다. 이후 명암도를 변경할 수 있는 트랙바 윈도우창이 띄워진다. 슬라이더의 위치로 동적으로 할당되는 명암도값을 최대값 100까지 반영한다. 트랙바를 옮길때마다 set_Contrast 함수가 콜백되어 값을 전달한다.

2.3.2 편집본 저장

```
# 편집본 이미지를 저장
save_path = "image/retouched.jpg"
cv2.imwrite(save_path, filtered_image)
print(f"편집본이 저장되었습니다: {save_path}")
```

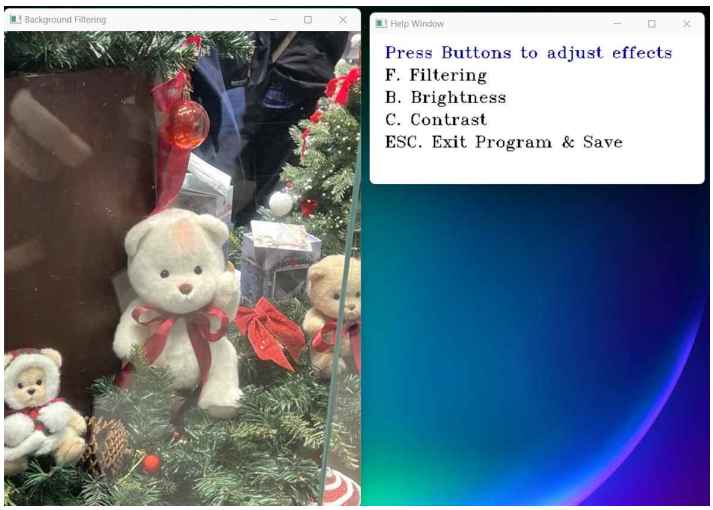
- cv2.imwrite() 함수로 "image/파일명.jpg" 상대 경로에 수정된 이미지 파일을 저장한다.



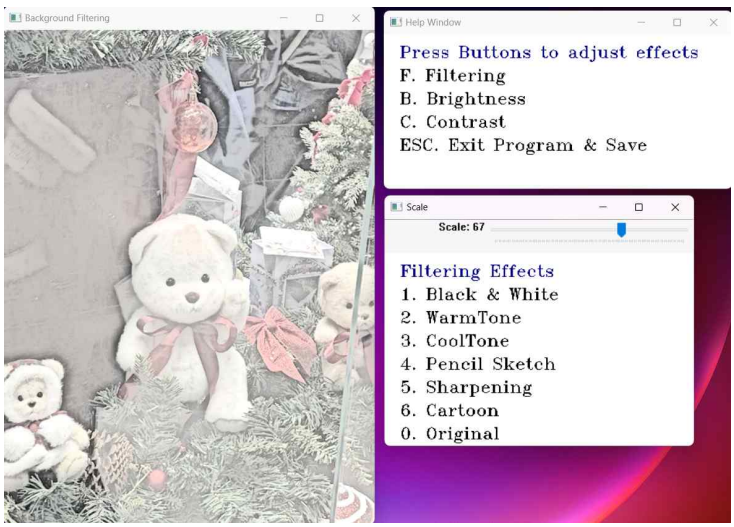
- 해당 경로에 수정된 이미지가 저장되었다.

3. 결과

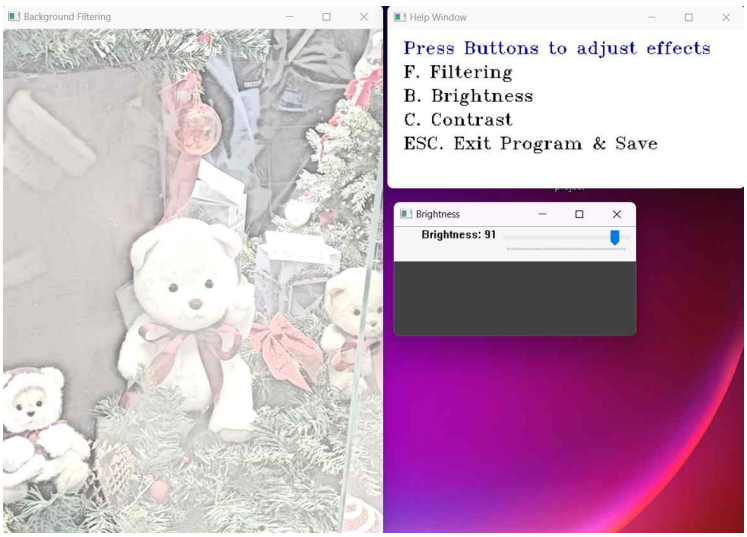
3.1 프로그램 실행 화면



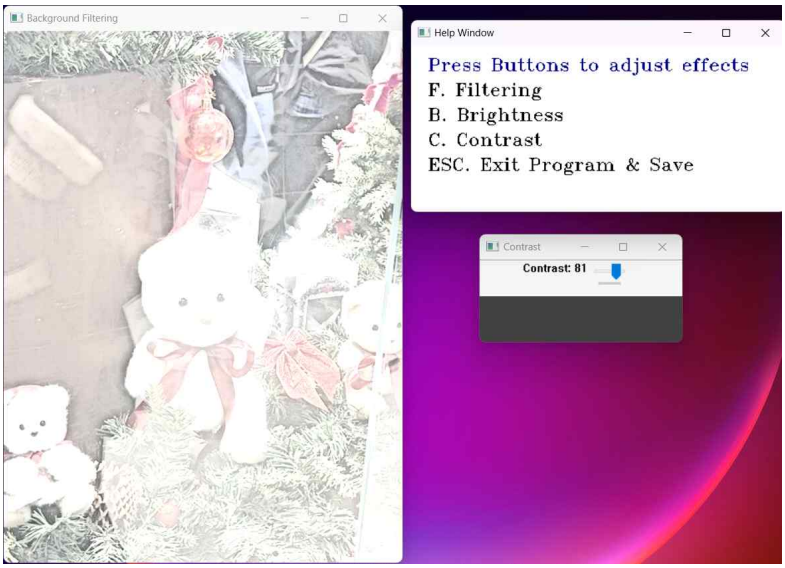
- 프로그램 실행 초기 화면



- f 키를 누른 후 4번을 눌러 스케치 필터 효과 적용 화면



- b 키를 누른 후 필터가 적용된 상태에서 밝기 조절



- c 키를 누른 후 필터 및 밝기가 적용된 상태에서 명암도 조절

- 사용자는 키보드와 슬라이더를 활용하여 이미지에 배경효과를 입히고 밝기와 명암도를 조절할 수 있다.
- 이후 조정된 이미지는 실시간으로 화면에 나타나며, 최종 편집된 이미지를 원하는 경로에 저장할 수 있다.

3.2 보완할 점

3.2.1 다양한 필터의 종류

- 이미지 필터링과 밝기, 명암도 조절이 프로그램의 핵심 코드인 만큼 적용 가능한 필터의 종류가 늘어날 필요가 있다.

3.2.2 배경효과 적용 과정

- 'f' 키를 눌러 배경효과 모드를 바꿀 때, 현재 모드를 실시간으로 나타낼 수 있다면 사용자의 편리성이 더욱 증가할 것이다.

3.2.3 강화된 예외 처리

- 잘못된 입력값이나 접근, 트랙바 및 필터 효과 적용 과정에서 발생할 수 있는 오류를 찾고 예외 처리 과정을 통해 프로그램의 안정성을 높일 필요가 있다.