

[디지털 영상 처리] 최종 프로젝트 미니 포토샵

컴퓨터공학부 201911228 홍지우

Contents

1. 개발 환경

2. 프로그램 실행 및 준비사항

3. 화면 구성

4. 기능 설명

1) Object recognition in video

- hand detection
- get background
- vehicle detection

2) Area processing

- average filter
- gaussian filter
- median filter

3) Edge detection

- high boost filter
- gradient edge detection
- LoG edge detection
- canny edge detection

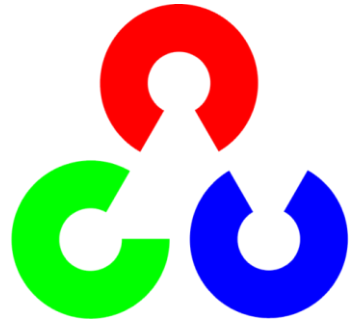
4) Histogram

- histogram equalization
- negative
- power law transformation

1. 개발 환경



PYTHON



OpenCV







ANACONDA®



- 사용 언어: python
- 영상 처리 라이브러리: open cv
- 패키지 관리 시스템: anaconda
- GUI: PyQt5

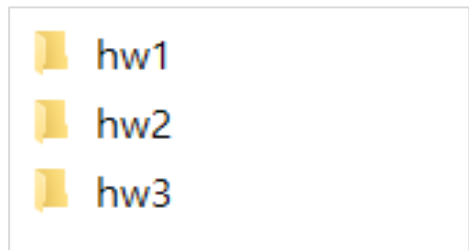
2. 프로그램 실행 및 준비사항 - [실행 방법]

 background	2021-07-01 오후 4:26	파일 폴더	
 example	2021-06-28 오후 5:29	파일 폴더	
 result	2021-07-01 오후 4:27	파일 폴더	
 RealMiniPhotoshop.py	2021-07-01 오후 4:22	Python File	25KB

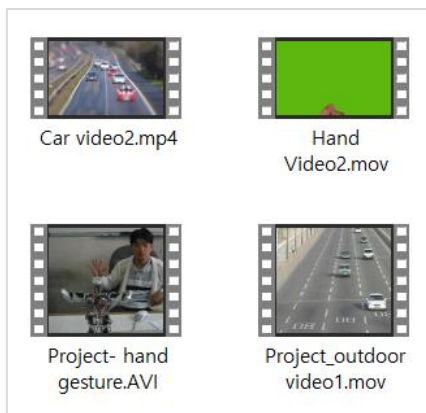
1. RealMiniPhotoshop.py 파일을 다운로드합니다.
2. 실행파일과 background, example, result 폴더는 동일한 폴더에 존재해야 합니다.
3. cv2, PyQt5 등 필요한 라이브러리를 준비합니다.
4. 변환하고 싶은 파일을 example 폴더에 저장합니다.
5. RealMiniPhotoshop.py를 통해 프로그램을 실행합니다.

(*사용자의 실행 환경에 따라 작동이 보장되지 않을 수 있습니다.)

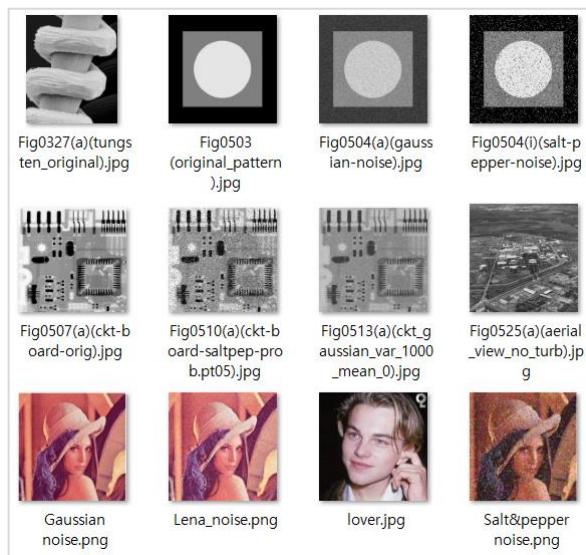
2. 프로그램 실행 및 준비사항 - [파일 준비사항]



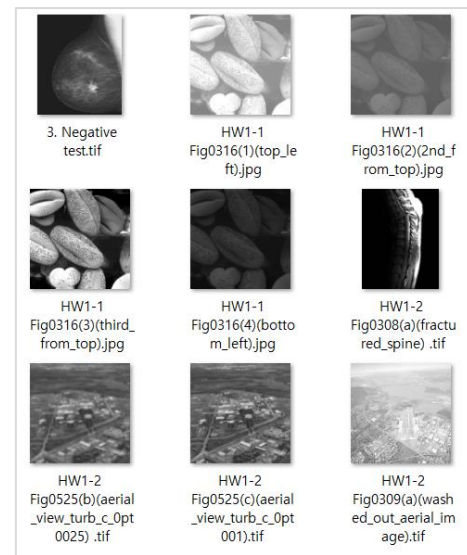
C:\Photoshop\example



C:\Photoshop\example\hw3



C:\Photoshop\example\hw2

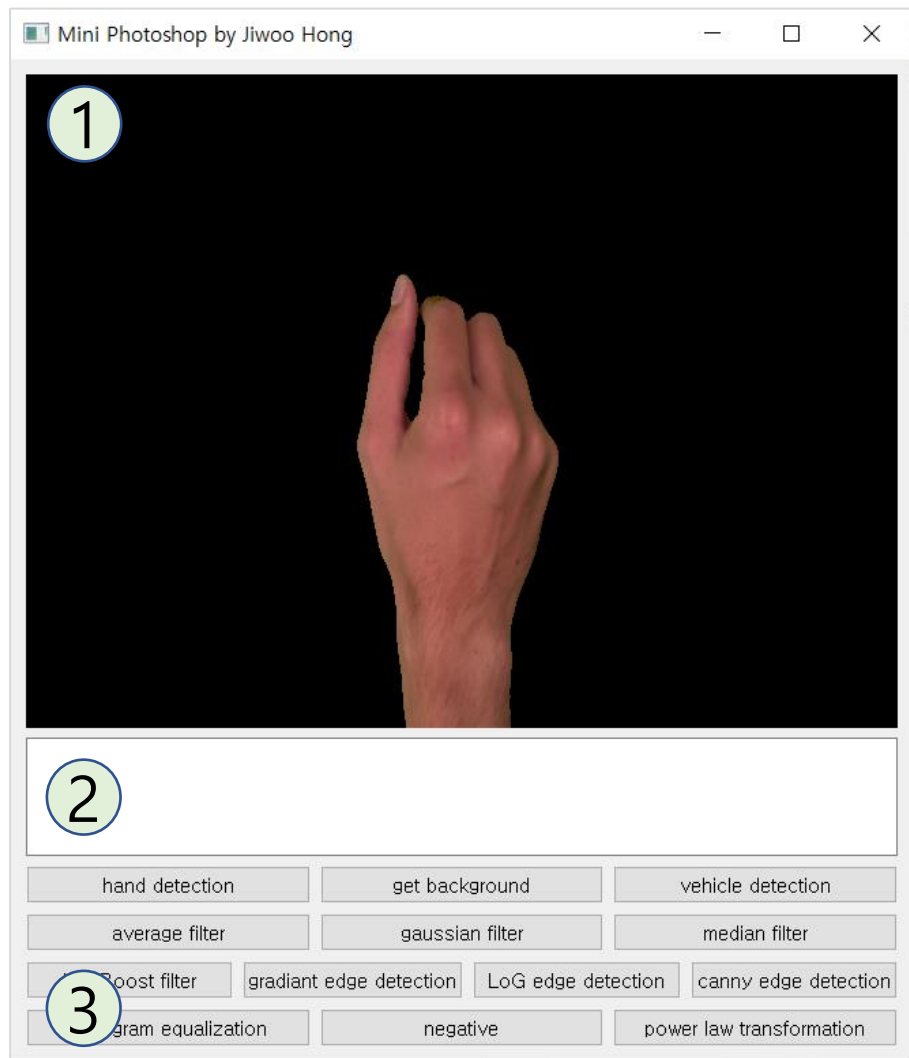


C:\Photoshop\example\hw1

- hw3 폴더에는 handDetection, get background, vehicle detection을 위한 영상 파일을 저장합니다.
- hw2 폴더에는 gaussian filter, median filter, average filter, gradient detection, high boost filter, canny edge detection, LoG edge detection을 위한 이미지 파일을 저장합니다.
- hw1 폴더에는 negative, histogram equalization, power law transformation을 위한 이미지 파일을 저장합니다.

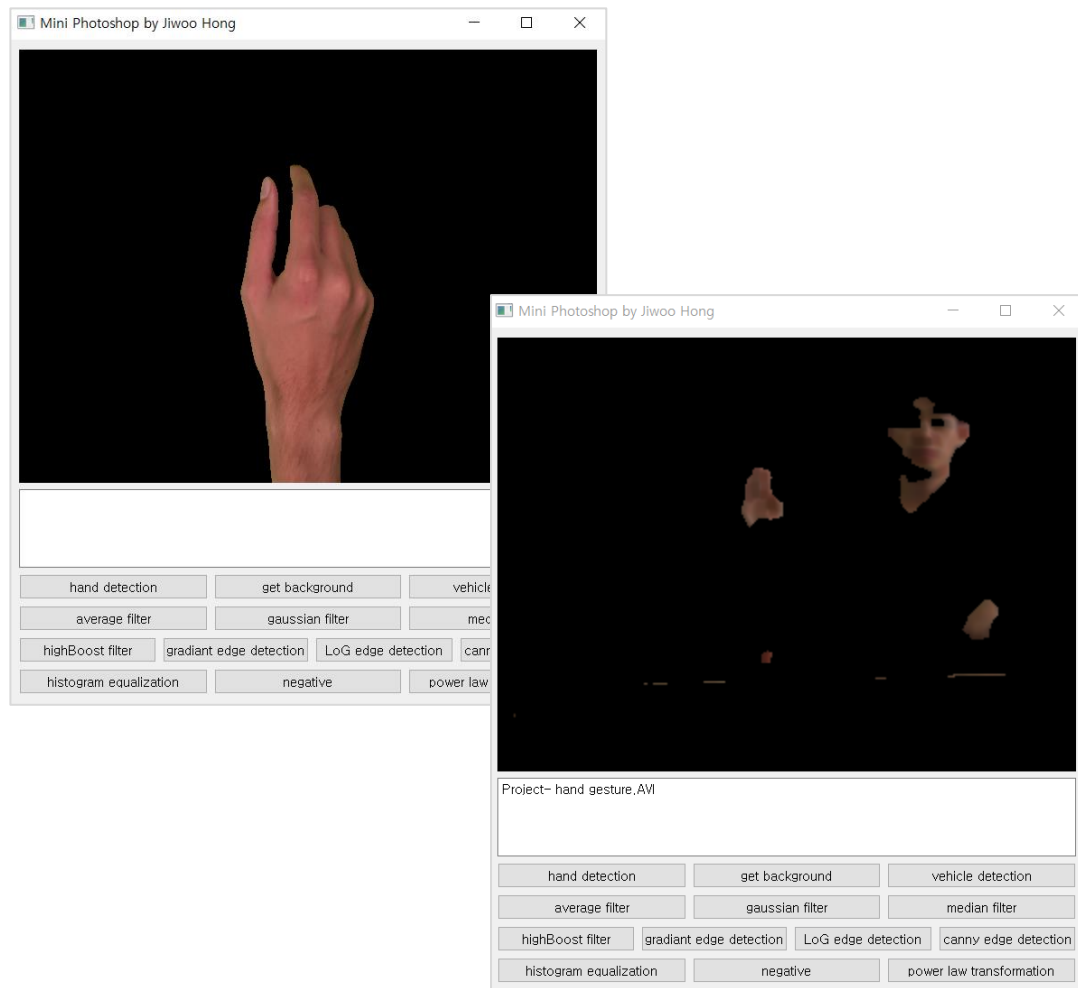
(*지정되지 않은 폴더에 영상 및 이미지를 준비할 경우 정상 작동이 보장되지 않습니다.)

3. 화면 구성



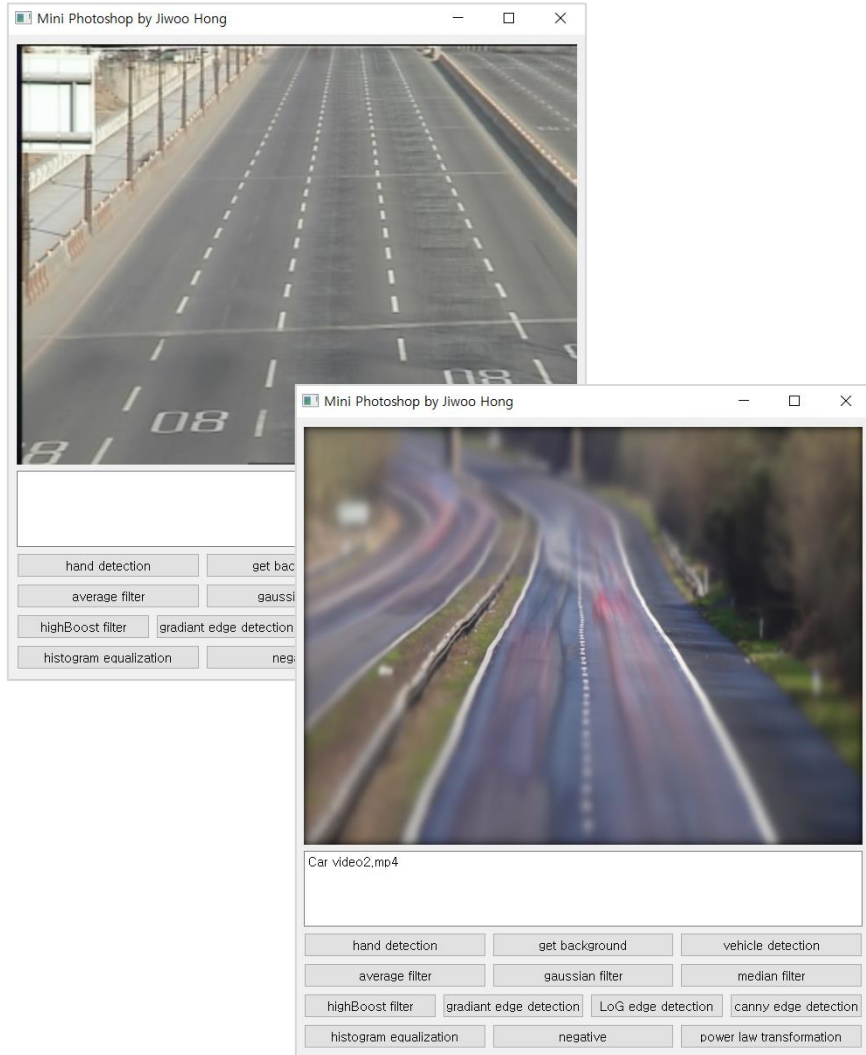
1. 가장 위쪽 레이어에서 영상 및 이미지가 변환된 모습을 확인할 수 있습니다. 또한 변환된 영상 및 이미지는 result 폴더 및 background 폴더에 저장됩니다.
2. 변환하려는 파일이 폴더에 저장되어 있을 때, 해당 파일명을 text창에 적고 하단 버튼을 통해 변환할 수 있습니다.
3. 하단 버튼을 통해 포토샵 기능을 이용하실 수 있습니다.
 - 1행: 영상 처리
(hand detection, get background, vehicle detection)
 - 2행: 이미지 처리 [color, gray 모두 가능]
(average filter, gaussian filter, median filter)
 - 3행: 이미지 edge 처리
(high-boost filter, gradient, LoG, canny edge detection)
 - 4행: 이미지 대비 향상
(histogram equalization, negative, power law transformation)

4. 기능 구성 (1) Object recognition in video – hand detection



1. 영상을 cv2.VideoCapture를 통해 가져온다.
2. 프레임마다 아래 과정을 수행한다.
 - Gaussian filter를 통해 이미지를 blurring한다.
 - 해당 프레임을 YCrCb로 색을 추출한다.
 - 피부색 범위를 포함하는 mask를 생성한다.
 - 프레임과 mask를 and연산하여 배경과 object를 구분한다.
 - 결과를 비디오로 저장하고, 화면에 결과 영상을 출력한다

4. 기능 구성 (1) Object recognition in video – get background



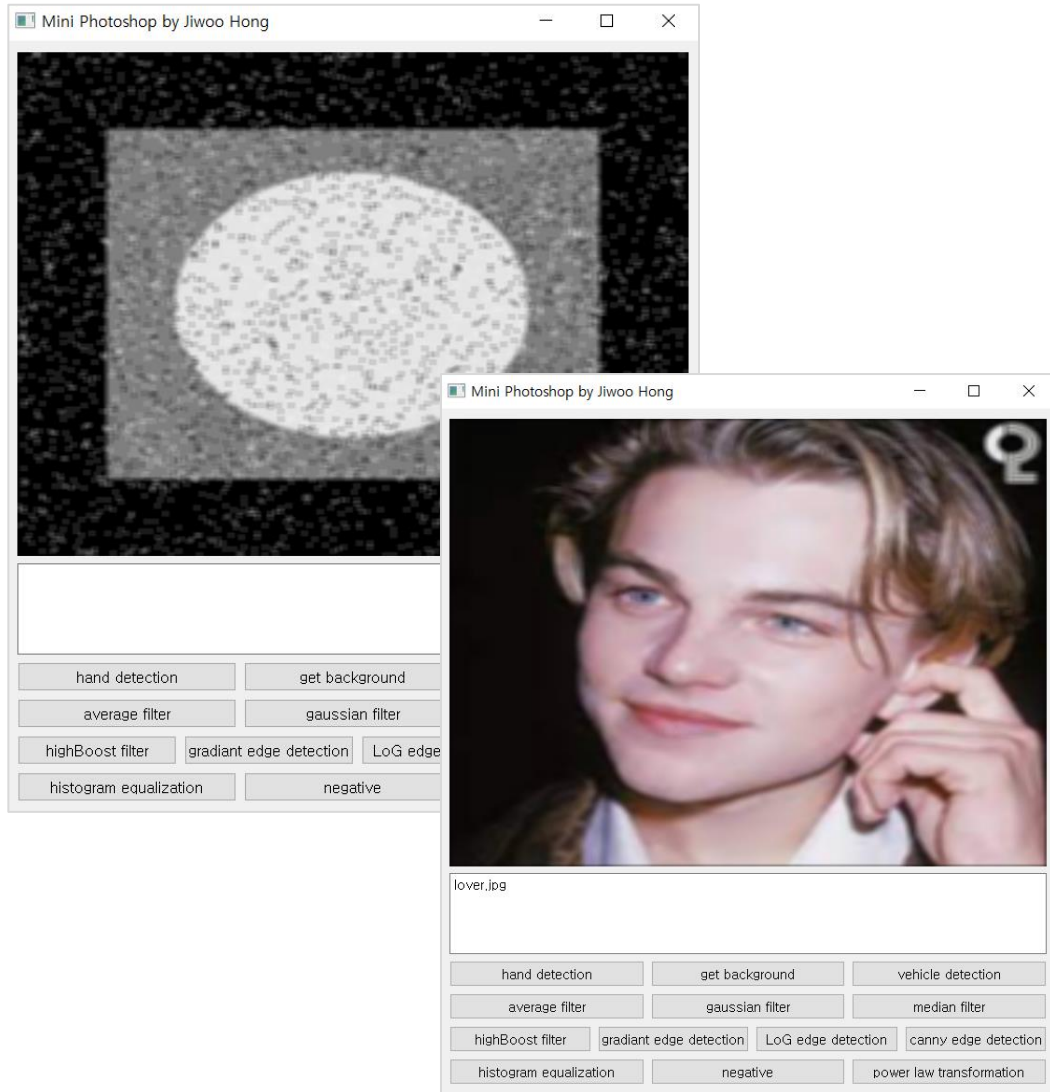
1. 영상에서 프레임을 추출한다.
2. 각 프레임마다 다음 단계를 진행한다.
 - 현재 프레임의 각 픽셀에 값을 누적한다.
 - 누적된 픽셀을 count로 나누어 평균을 취한다.
 - 값을 8비트 unsigned int로 변경하여 저장한다.
 - 결과를 이미지로 저장하고, 화면에 결과 영상을 출력한다.

4. 기능 구성 (1) Object recognition in video – vehicle detection

1. background 이미지를 가져온다
2. 영상을 프레임으로 추출한다.
3. 각 프레임마다 다음 단계를 진행한다.
 - 현재 프레임과 background 이미지를 subtraction한다.
 - RGB마다 threshold를 취해 배경은 0으로 이미지는 1로 변환한다.
 - CCL을 통해 객체의 사각형 꼭짓점을 구한다.
 - 객체의 넓이가 일정 threshold보다 큰 경우 원본 영상에 rectangle을 그린다.

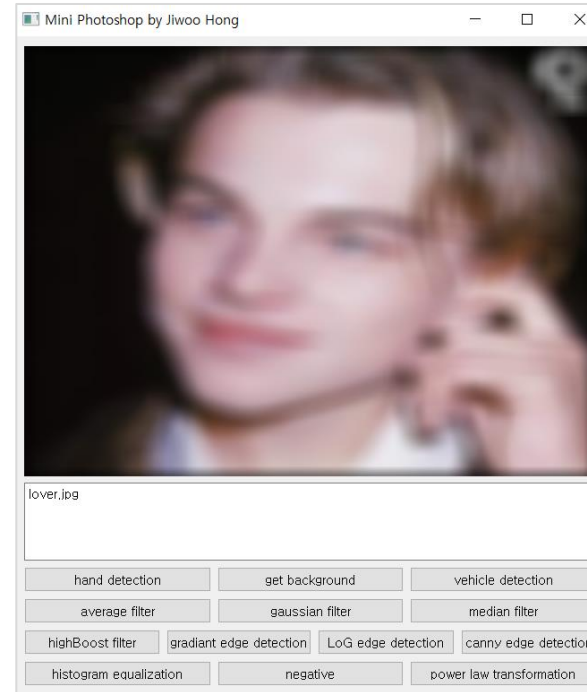
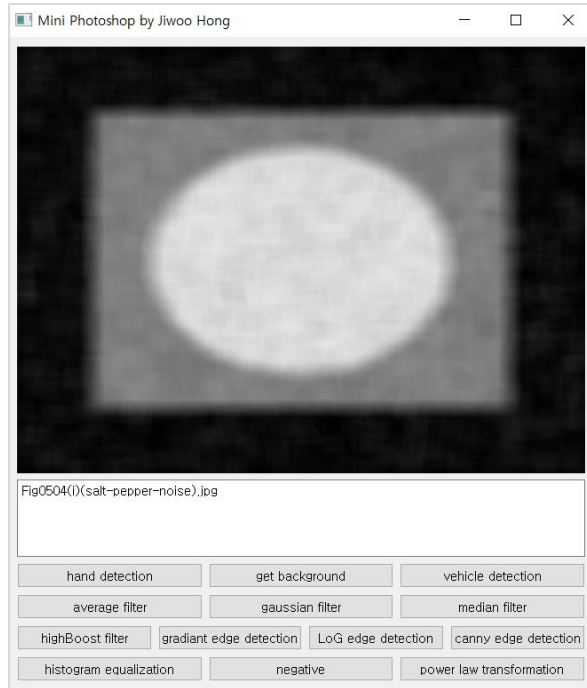


4. 기능 구성 (2) Area processing – average filter



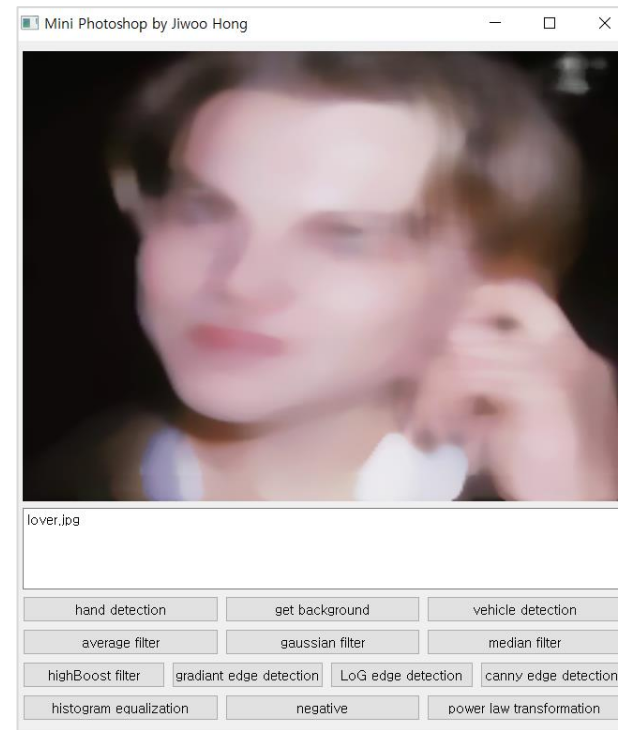
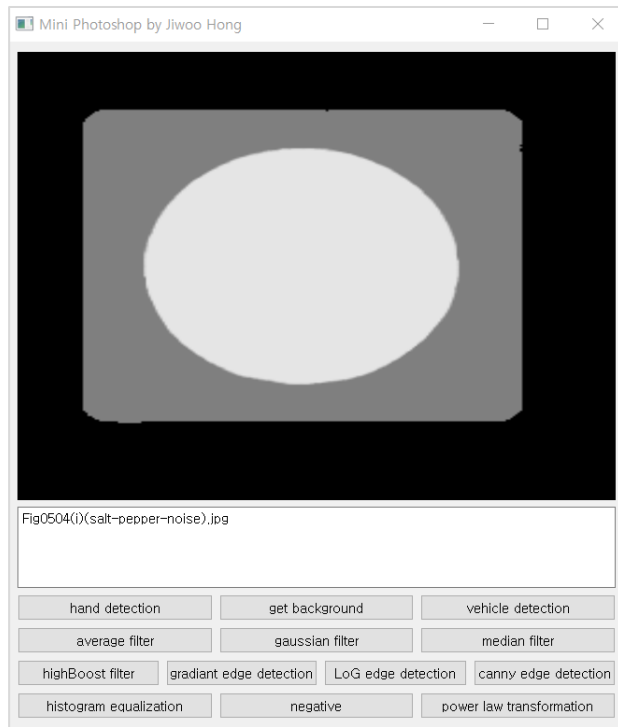
1. average를 위한 mask를 생성한다.
2. 이미지와 mask를 convolution한다.
3. 각 픽셀의 값이 0~255가 넘지 않도록 수정한다.
4. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (2) Area processing – gradient edge detection



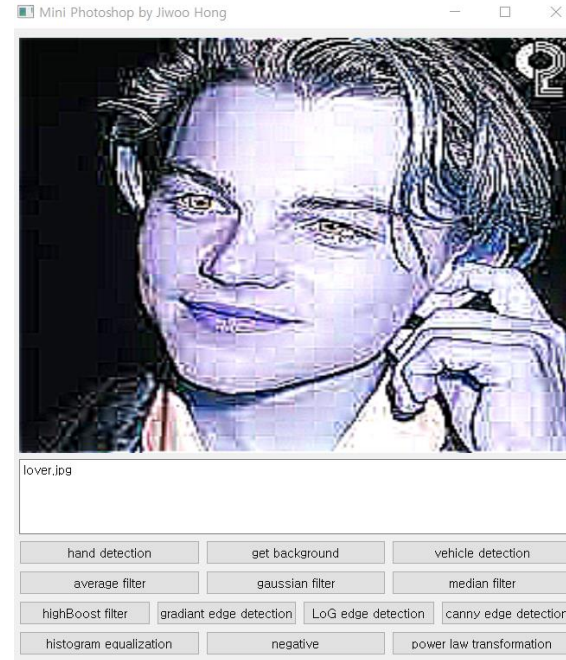
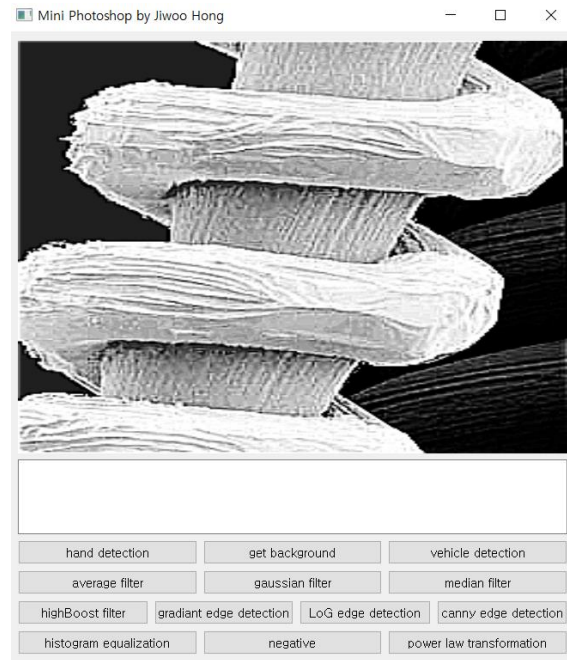
1. sigma값을 10으로 하여 1D에서의 gaussian값을 행, 열 방향으로 구하여 mask를 얻는다.
2. 컬러 이미지일 경우, RGB 3개의 영역에서 convolution을 진행한다.
3. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (2) Area processing – median filter



1. mask의 크기를 15로 설정한다.
2. 15*15 크기만큼의 픽셀을 나열해서 media값을 구한다.
3. 결과 이미지를 저장하고, 화면에 출력한다.

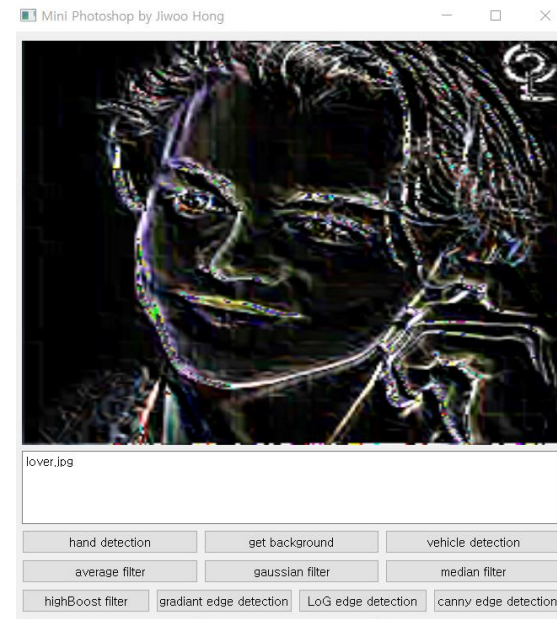
4. 기능 구성 (3) Edge detection – high boost filter



1. high boost filter mask를 생성한다.
2. mask와 이미지를 convolution한다.
3. 결과 이미지를 저장하고, 화면에 출력한다.

$$\begin{bmatrix} [-1. & -1. & -1.] \\ [-1. & 9.2 & -1.] \\ [-1. & -1. & -1.] \end{bmatrix}$$

4. 기능 구성 (3) Edge detection – gradient edge detection



```
kernel_x = np.array([[ -1,  0,  1], [ -2,  0,  2], [ -1,  0,  1]])  
kernel_y = np.array([[ 1,  2,  1], [ 0,  0,  0], [ -1, -2, -1]])
```

1. x축 및 y축으로 sobel mask를 생성한다.
2. mask와 이미지를 convolution한다.
3. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (3) Edge detection – LoG edge detection



```
kernel = np.array([[0,0,1,0,0], [0,1,2,1,0], [1,2,-16,2,1], [0,1,2,1,0],[0,0,1,0,0]])
```

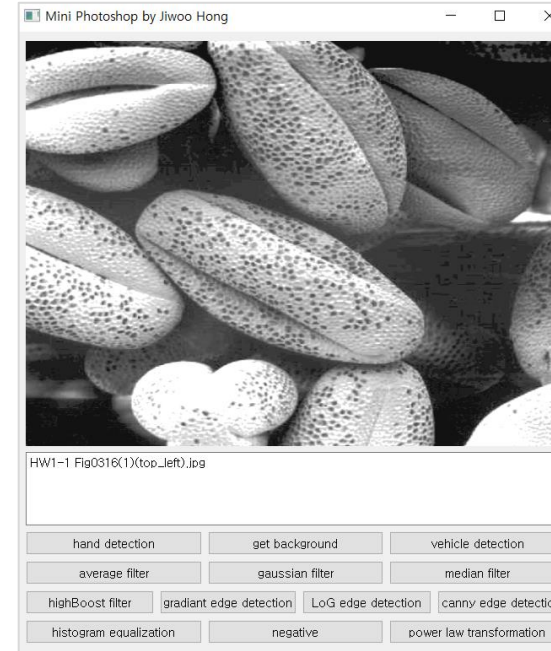
1. LoG mask를 생성한다.
2. mask와 이미지를 convolution한다.
3. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (3) Edge detection – canny edge detection



1. opencv 라이브러리인 cv2.Canny함수를 사용하여 edge를 detection한다.
2. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (3) Histogram – histogram equalization



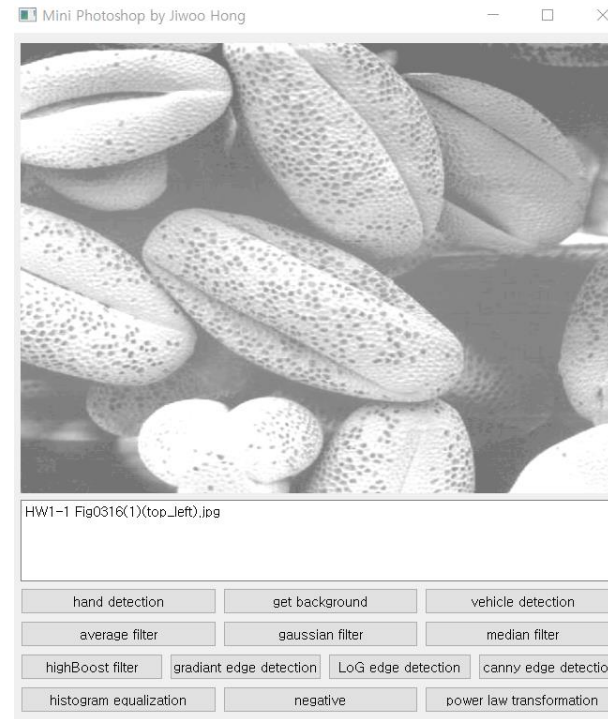
1. gray scale의 크기인 256 만큼의 배열을 생성한다.
2. 이미지를 읽으면서 histogram을 계산한다.
3. Look up table을 생성한다.
4. table을 읽으면서 이미지의 값을 변경해 대비를 향상시킨다.
5. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (3) Histogram - negative



1. gray scale 이미지의 픽셀 값을 하나씩 읽으면서 255에서 값을 빼 반전시킨다.
2. 결과 이미지를 저장하고, 화면에 출력한다.

4. 기능 구성 (3) Histogram – power law transformation



1. 이미지를 가져오고 $s = cr^\gamma$ 에서 $c = 1$ 로 고정한 뒤 다음을 각 픽셀에 대해 반복한다.
2. r 은 $[0, 1]$ 사이의 값 이므로, 255로 나누어 준 뒤 이를 감마 제공한다. 원래의 범위인 $[0, 255]$ 으로 되돌리기 위해 255.를 다시 곱해 준다.
3. 결과 이미지를 저장하고, 화면에 출력한다.

감사합니다