
영상처리 프로그래밍

7주차

소개

- 조교 : 김형훈
 - 연구실 : A1409 (운영체제 연구실)
 - 이메일 : khh8996@naver.com
 - 전화번호 : 010-5316-7953
- 담당 교수 : 김백섭 교수님

목차

- 마스크(Mask) 연산
- 엣지(Edge) 검출
- 모폴로지(Morphology)
- 실습
- 과제

마스크(Mask) 연산

- 마스크(Mask)를 커널(Kernel), 필터(Filter), 윈도우(Window) 라고 부름
- 이미지 필터링(Image Filtering)은 마스크를 이미지와 컨볼루션(Convolution) 연산하는 것
 - 컨볼루션 연산
 - 공간 필터링(Spatial Filtering) 라고 부름
 - 생성한 마스크를 이동시키면서 같은 이미지 영역과 곱하고 더하여 그 결과 값을 이미지의 해당 위치의 값으로 하는 새로운 이미지를 만드는 연산

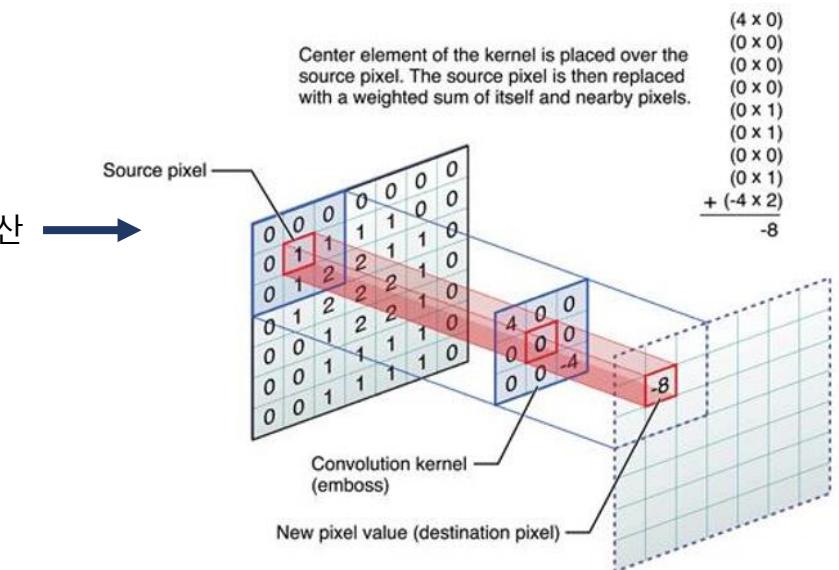


그림 출처 : <https://steemit.com/kr-dev/@jeaimetu/convolution>

마스크(Mask) 연산

- 블러링 (Blurring)
 - 이미지의 초점이 맞지 않은 것처럼 흐릿하게 만드는 것
- 모션 블러 (Motion Blur)
 - 이미지를 특정 방향으로 움직이는 것처럼 만드는 것
- 샤프닝 (Sharpening)
 - 이미지의 엣지(Edge)를 날카롭게 만드는 것
- 엠보싱 (Embossing)
 - 이미지의 화소를 가져와서 그림자나 하이라이트로 바꾸는 것

마스크(Mask) 연산

- OpenCV에서 컨볼루션 연산하는 함수를 제공함
 - cv2.filter2D(src, ddepth, kernel[, dst, anchor, delta, borderType])
 - src : 입력 이미지, Numpy 배열
 - ddepth : 출력 이미지의 dtype
 - -1 : 입력 이미지와 동일함
 - CV_8U, CV16U/CV16S, CV_32F, CV_64F
 - kernel : 컨볼루션 커널, float32의 $n \times n$ 크기의 배열
 - dst : 결과 이미지, Numpy 배열
 - anchor : 커널의 기준점, default : 중심점(-1, 1)
 - delta : 필터 적용된 결과에 추가할 값
 - borderType : 외곽 픽셀 보정 방법 지정
 - 보통, 생성한 마스크를 kernel에 입력함

마스크(Mask) 연산

- OpenCV에서 가우시안 분포(Gaussian Distribution) 커널로 블러링을 적용하는 함수를 제공함
 - cv2.GaussianBlur(src, ksize, sigmaX[, sigmaY, borderType])
 - src : 입력 이미지
 - ksize : 커널 크기, 홀수
 - sigmaX : X 방향 표준편차
 - 0 : auto, 시그마(σ) 값 지정
 - sigmaY : Y 방향 표준편차
 - default : sigmaX
 - borderType : 외곽 테두리 보정 방식

마스크(Mask) 연산

- 예시 - 마스크 연산

```
# 예제 - 블러링(Blurring)

# 이미지 파일 불러오기
img = cv2.imread('./train_input.png')

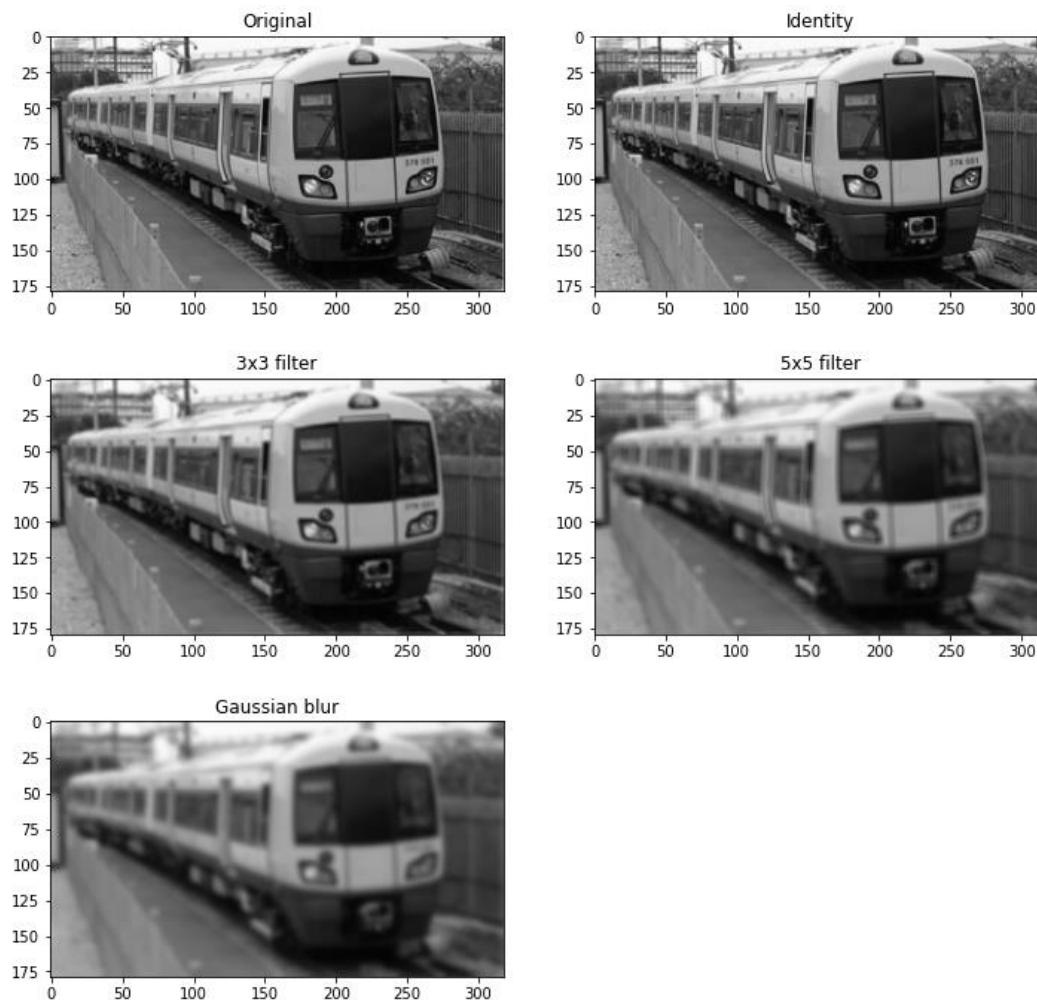
# 커널(Kernel) 생성
kernel_identity = np.array([[0,0,0], [0,1,0], [0,0,0]])
kernel_3x3 = np.ones((3,3), np.float32) / 9.0
kernel_5x5 = np.ones((5,5), np.float32) / 25.0

# 필터 적용 (컨볼루션(Convolution) 연산)
output1 = cv2.filter2D(img, -1, kernel_identity)
output2 = cv2.filter2D(img, -1, kernel_3x3)
output3 = cv2.filter2D(img, -1, kernel_5x5)

# 가우시안 블러링(GaussianBlur) 적용
# 시그마(sigma)를 2.0으로 설정
output4 = cv2.GaussianBlur(img, (0, 0), 2.0)

# 결과 출력
plt.figure(figsize=[12,12])
plt.subplot(321):plt.imshow(img[...,:-1]):plt.title("Original")
plt.subplot(322):plt.imshow(output1[...,:-1]):plt.title("Identity")
plt.subplot(323):plt.imshow(output2[...,:-1]):plt.title("3x3 filter")
plt.subplot(324):plt.imshow(output3[...,:-1]):plt.title("5x5 filter")
plt.subplot(325):plt.imshow(output4[...,:-1]):plt.title("Gaussian blur")

plt.show()
```



마스크(Mask) 연산

- 예시 - $\text{img}[:, :, :-1]$ 과 $\text{img}[:, :, ::-1]$ 비교

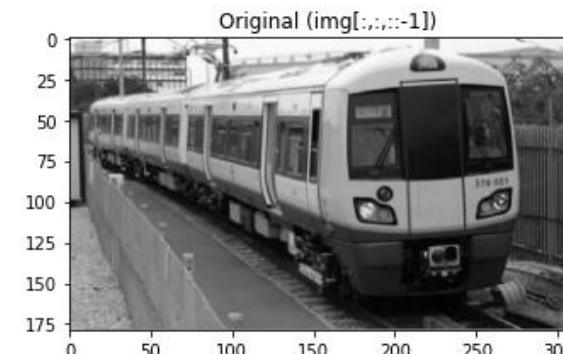
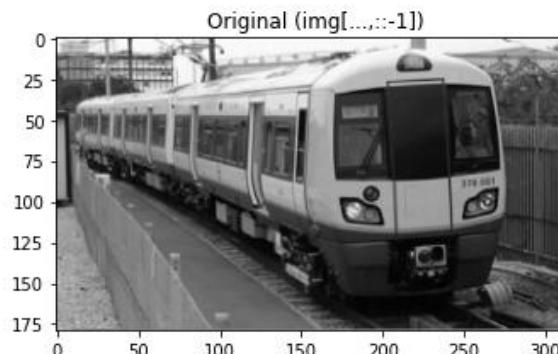
```
# img[..., ::-1]과 img[:, :, ::-1] 비교
# 동일한 결과를 출력함

plt.figure(figsize=[12,12])

plt.subplot(121)
plt.imshow(img[..., ::-1])
plt.title("Original (img[..., ::-1])")

plt.subplot(122)
plt.imshow(output1[:, :, ::-1])
plt.title("Original (img[:, :, ::-1])")

plt.show()
```



마스크(Mask) 연산

- 예시 - 모션 블러(Motion Blur)

```
# 예제 - 모션 블러(Motion Blur)

# 이미지 파일 불러오기
img = cv2.imread('./tree_input.png')

size = 9

# 커널(Kernel) 생성
kernel_motion_blur = np.zeros((size, size))
kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
kernel_motion_blur = kernel_motion_blur / size

# Numpy Float 출력 옵션 설정 (소수점 2번째 자리까지 출력)
np.set_printoptions(precision=2)
print('Generated Kernel\n', kernel_motion_blur)

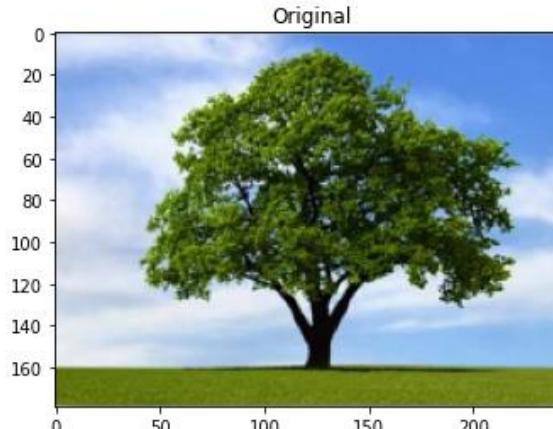
# 필터 적용 (컨볼루션(Convolution) 연산)
output = cv2.filter2D(img, -1, kernel_motion_blur)

# 결과 출력
plt.figure(figsize=[12,10])
plt.subplot(121):plt.imshow(img[...,:-1]):plt.title("Original")
plt.subplot(122):plt.imshow(output[...,:-1]):plt.title("Motion Blur")

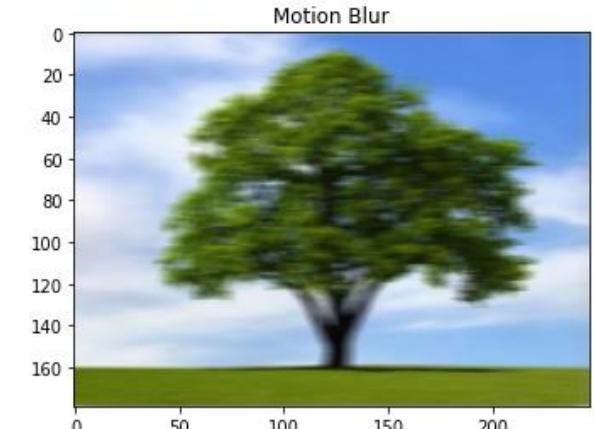
plt.show()
```

Generated Kernel

```
[[0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11, 0.11]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]
 [0., 0., 0., 0., 0., 0., 0., 0., 0.]]
```



Original



Motion Blur

마스크(Mask) 연산

- 예시 - 샤프닝(Sharpening)

```
# 예제 - 샤프닝(Sharpening)

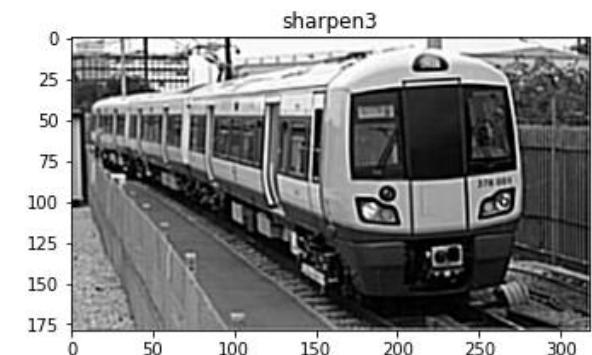
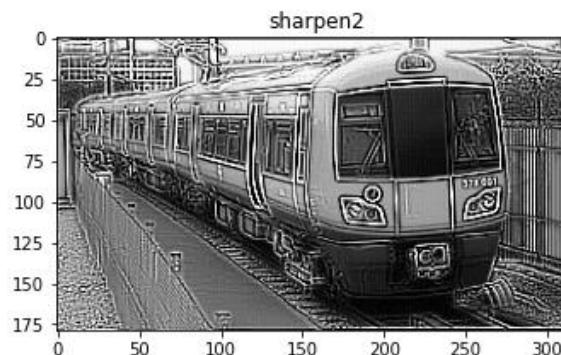
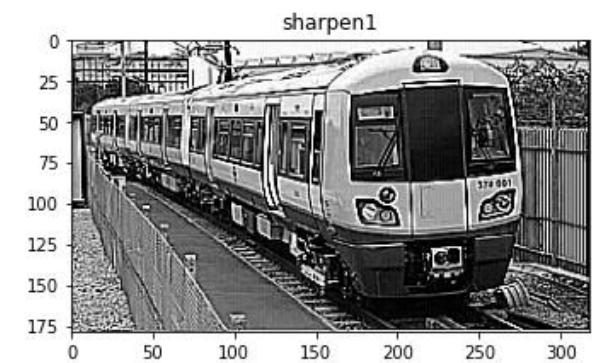
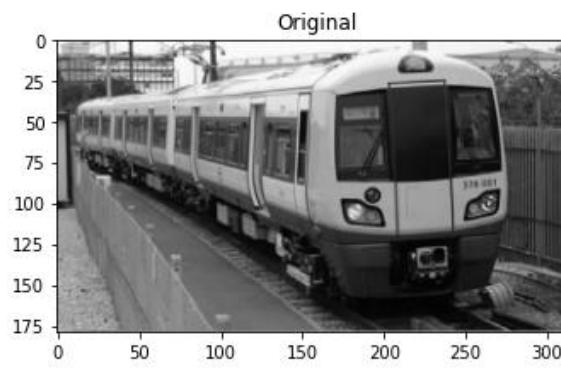
# 이미지 파일 불러오기
img = cv2.imread('./train_input.png')

# 커널(Kernel) 생성
kernel_sharpen_1 = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])
kernel_sharpen_2 = np.array([[1,1,1], [1,-7,1], [1,1,1]])
kernel_sharpen_3 = np.array([[-1,-1,-1,-1,-1],
                            [-1, 2, 2, 2,-1],
                            [-1, 2, 8, 2,-1],
                            [-1, 2, 2, 2,-1],
                            [-1,-1,-1,-1,-1]]) / 8.0

# 필터 적용 (컨볼루션(Convolution) 연산)
output1 = cv2.filter2D(img, -1, kernel_sharpen_1)
output2 = cv2.filter2D(img, -1, kernel_sharpen_2)
output3 = cv2.filter2D(img, -1, kernel_sharpen_3)

# 결과 출력
plt.figure(figsize=[12,10])
plt.subplot(221):plt.imshow(img[...,:-1]):plt.title("Original")
plt.subplot(222):plt.imshow(output1[...,:-1]):plt.title("sharpen1")
plt.subplot(223):plt.imshow(output2[...,:-1]):plt.title("sharpen2")
plt.subplot(224):plt.imshow(output3[...,:-1]):plt.title("sharpen3")

plt.show()
```



마스크(Mask) 연산

- 예시 - 엠보싱(Embossing)

```
# 예제 - 엠보싱(Embossing)

# 이미지 파일 불러오기
img = cv2.imread('./house_input.png')

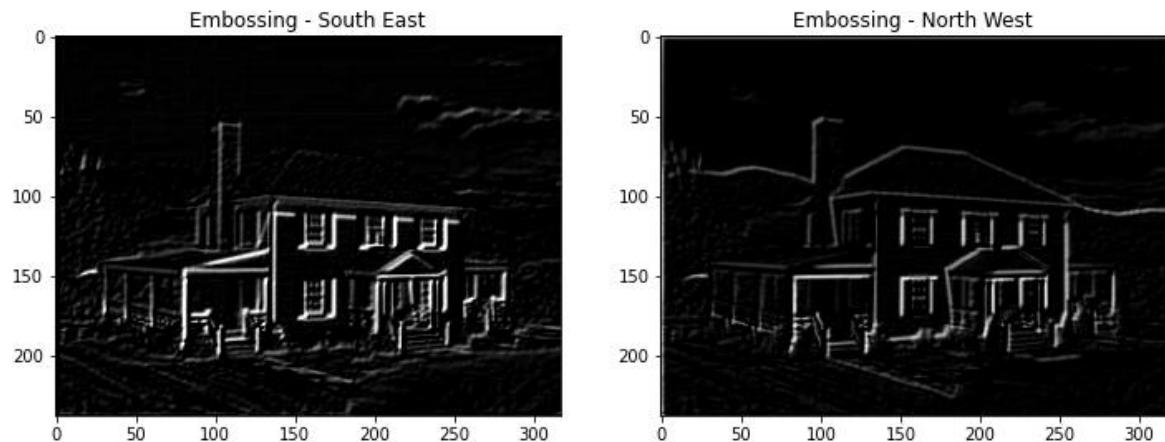
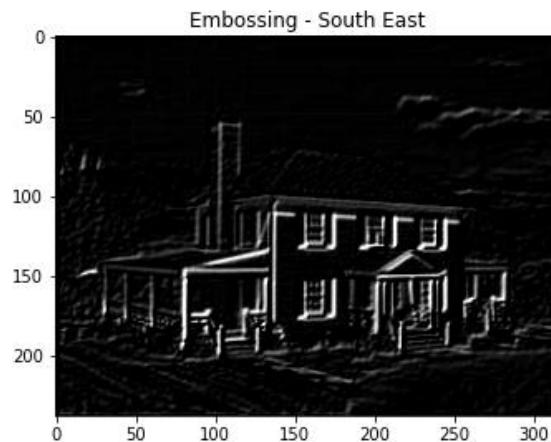
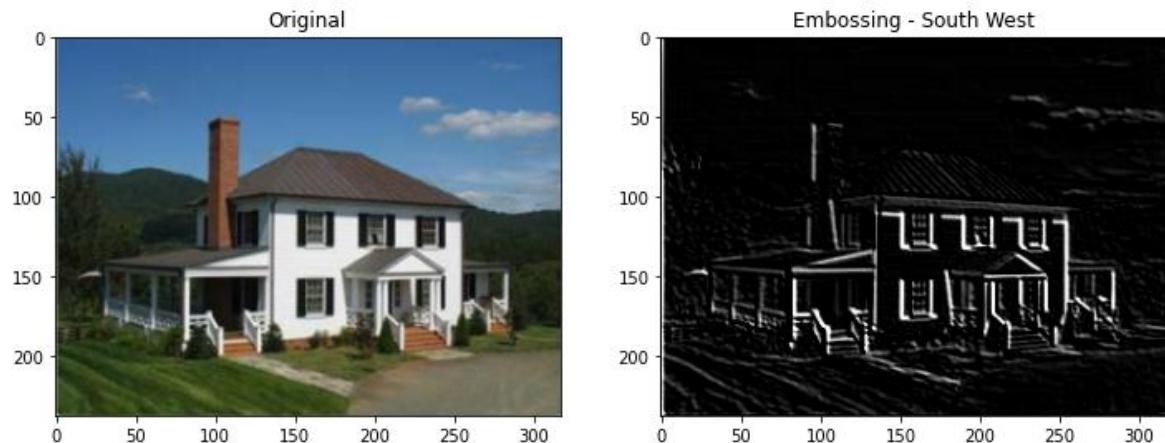
# 커널(Kernel) 생성
kernel_emboss_1 = np.array([[0,-1,-1], [1,0,-1], [1,1,0]])
kernel_emboss_2 = np.array([[-1,-1,0], [-1,0,1], [0,1,1]])
kernel_emboss_3 = np.array([[1,0,0], [0,0,0], [0,0,-1]])

# BGR에서 GRAY로 변경
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 필터 적용 (컨볼루션(Convolution) 연산)
output1 = cv2.filter2D(gray_img, -1, kernel_emboss_1)
output2 = cv2.filter2D(gray_img, -1, kernel_emboss_2)
output3 = cv2.filter2D(gray_img, -1, kernel_emboss_3)

# 결과 출력
plt.figure(figsize=[12,10])
plt.subplot(221):plt.imshow(img[:, :, ::-1]):plt.title("Original")
plt.subplot(222):plt.imshow(output1, cmap='gray'):plt.title("Embossing - South West")
plt.subplot(223):plt.imshow(output2, cmap='gray'):plt.title("Embossing - South East")
plt.subplot(224):plt.imshow(output3, cmap='gray'):plt.title("Embossing - North West")

plt.show()
```



엣지(Edge) 검출

- 소벨 (Sobel)
 - 엣지 검출의 가장 대표적인 미분 연산자로, 입력 이미지에 대해 수평과 수직 경계선을 검출
 - cv2.Sobel(src, ddepth, dx, dy[, dst, ksize, scale, delta, borderType])
 - src : 입력 이미지
 - ddepth : 출력 이미지의 dtype
 - -1 : 입력 이미지와 동일함
 - CV_8U, CV16U/CV16S, CV_32F, CV_64F
 - dx, dy : 미분 차수 (0, 1, 2 중 선택. 단, 둘 다 0일 수는 없음)
 - ksize : 커널의 크기
 - scale : 미분에 사용할 계수
 - delta : 연산 결과에 가산할 값

엣지(Edge) 검출

- 라플라시안 (Laplacian)
 - 대표적인 2차 미분 마스크
 - 2차 미분을 적용하면 경계를 좀 더 확실히 검출할 수 있음
 - cv2.Laplacian(src, ddepth[, dst, ksize, scale, delta, borderType])
 - src : 입력 이미지
 - ddepth : 출력 이미지의 dtype
 - -1 : 입력 이미지와 동일함
 - CV_8U, CV16U/CV16S, CV_32F, CV_64F
 - ksize : 커널의 크기
 - scale : 미분에 사용할 계수
 - delta : 연산 결과에 가산할 값

엣지(Edge) 검출

- 캐니 (Canny)
 - 잡음에 강한 뛰어난 엣지 검출 기법
 - cv2.Canny(img, threshold1, threshold2, [, edges, apertureSize, L2gradient])
 - img : 입력 이미지
 - threshold1, threshold2 : 이력 임계값에 사용할 최소, 최대 값
 - apertureSize : 소벨 마스크에 사용할 커널 크기
 - L2gradient : 기울기를 구할 방식 지정 플래그 (True, False)
 - edges : 엣지 결과 값을 갖는 2차원 배열

엣지(Edge) 검출

- 예시 - 엣지 검출 (Sobel, Laplacian, Canny)

```
# 예제 - 엣지 검출(Edge Detection)
```

```
# 이미지 파일 불러오기  
img = cv2.imread('./train_input.png')
```

```
# 소벨(Sobel) 필터 적용 (수직선 방향의 엣지를 검출)  
sobel_vertical = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
```

```
# 소벨(Sobel) 필터 적용 (수평선 방향의 엣지를 검출)  
sobel_horizontal = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
```

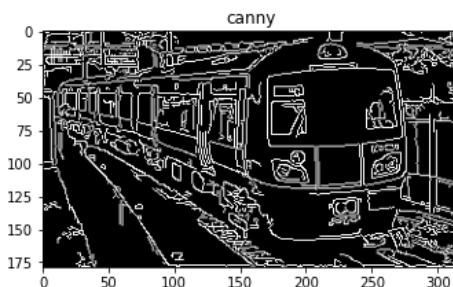
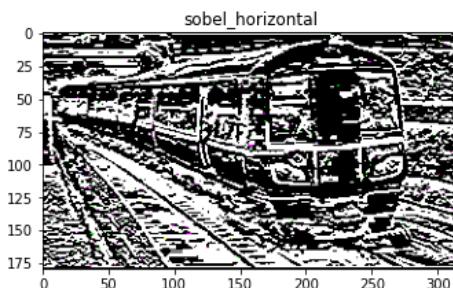
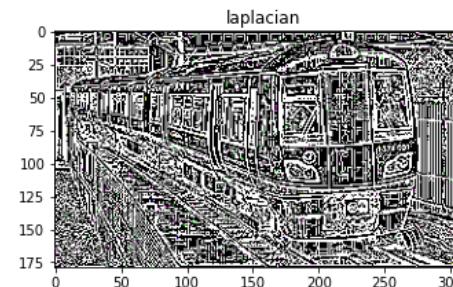
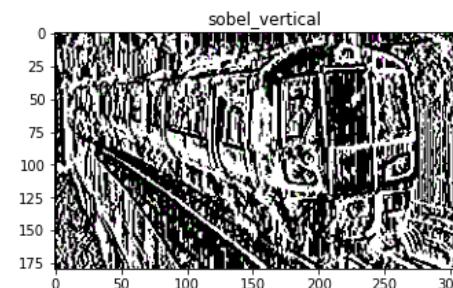
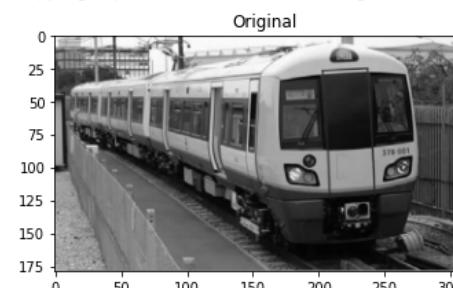
```
# 라플라시안(Laplacian) 필터 적용  
laplacian = cv2.Laplacian(img, cv2.CV_64F)
```

```
# 캐니(Canny) 필터 적용  
canny = cv2.Canny(img, 50, 240)
```

```
# 결과 출력
```

```
plt.figure(figsize=[12,12])  
plt.subplot(321):plt.imshow(img[:, :, ::-1]):plt.title("Original")  
plt.subplot(322):plt.imshow(sobel_vertical, cmap='gray'):plt.title("sobel_vertical")  
plt.subplot(323):plt.imshow(sobel_horizontal, cmap='gray'):plt.title("sobel_horizontal")  
plt.subplot(324):plt.imshow(laplacian, cmap='gray'):plt.title("laplacian")  
plt.subplot(325):plt.imshow(canny, cmap='gray'):plt.title("canny")  
  
plt.show()
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).



모폴로지(Morphology)

- 침식 (Erosion)
 - 원래 있던 객체의 영역을 깎아 내리는 연산
 - cv2.erode(src, kernel[, anchor, iterations, borderType, borderValue])
 - src : 입력 이미지
 - kernel : 구조화 요소 커널 객체
 - anchor : 구조화 요소의 기준점
 - iterations : 침식 연산 적용 반복 횟수
 - borderType : 외곽 영역 보정 방법 설정 플래그
 - borderValue : 외곽 영역 보정 값

모폴로지(Morphology)

- 팽창 (Dilatation)
 - 침식과 반대로 영상 속 사물의 주변을 덧붙여서 영역을 확장하는 연산
 - cv2.dilate(src, kernel[, dst, anchor, iterations, borderType, borderValue])
 - src : 입력 이미지
 - kernel : 구조화 요소 커널 객체
 - anchor : 구조화 요소의 기준점
 - iterations : 침식 연산 적용 반복 횟수
 - borderType : 외곽 영역 보정 방법 설정 플래그
 - borderValue : 외곽 영역 보정 값

모폴로지(Morphology)

- 예시 - 모폴로지 (Erosion, Dilatation)

```
# 예제 - 모폴로지(Morphology)

# 이미지 파일 불러오기
img = cv2.imread('./text_input.png')

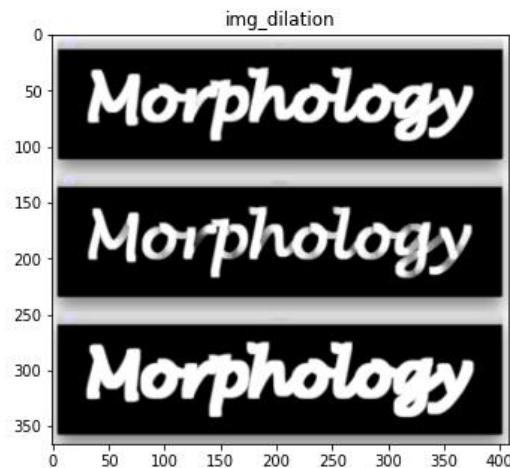
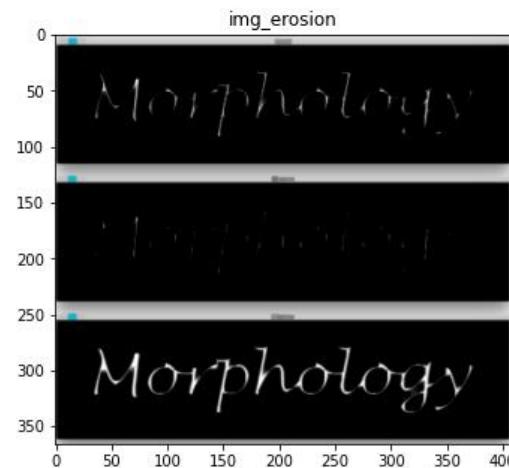
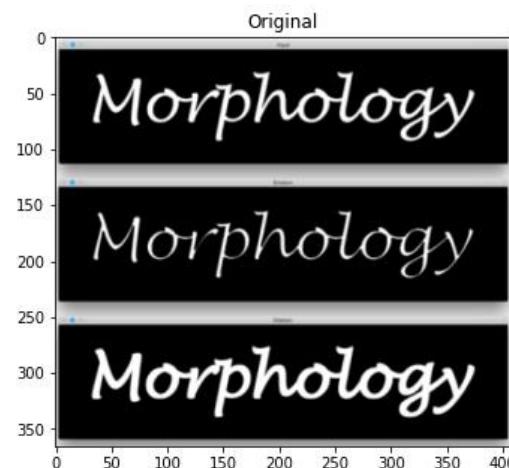
# 커널(Kernel) 생성
kernel = np.ones((5,5), np.uint8)

# 침식 연산 적용
img_erosion = cv2.erode(img, kernel, iterations=1)

# 팽창 연산 적용
img_dilation = cv2.dilate(img, kernel, iterations=1)

# 결과 출력
plt.figure(figsize=[12,12])
plt.subplot(221):plt.imshow(img, cmap='gray'):plt.title("Original")
plt.subplot(223):plt.imshow(img_erosion, cmap='gray'):plt.title("img_erosion")
plt.subplot(224):plt.imshow(img_dilation, cmap='gray'):plt.title("img_dilation")

plt.show()
```



실습

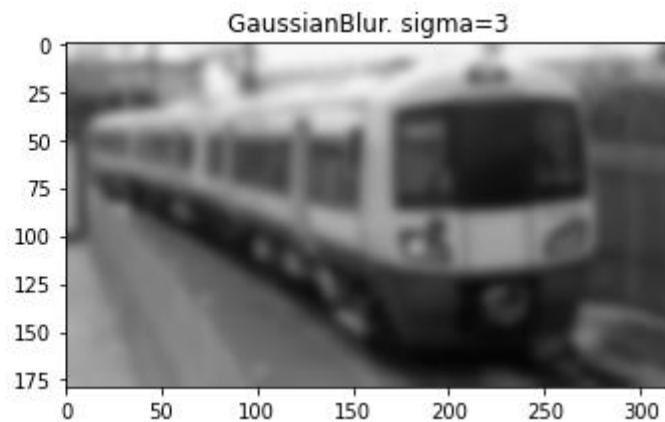
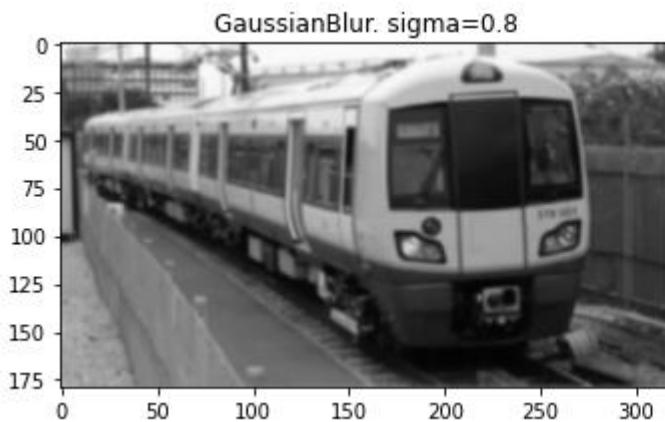
- 실습은 예시를 따라 하는 것으로 대체함

과제

- 소스 코드와 결과를 캡쳐하여 문서(워드, 한글)에 정리한 후, 스마트 캠퍼스에 제출
 - 제출 형식 : 교과목명_주차_학번_이름.docx or .hwp (제출 형식 안 맞을 시 감점)
 - 기간 내에 제출하지 못할 경우 이메일로 제출 (감점)
 - 주석 작성 필수 (주석 없을 시 감점)
 - 부정 행위 적발 시 감점 (컨닝)

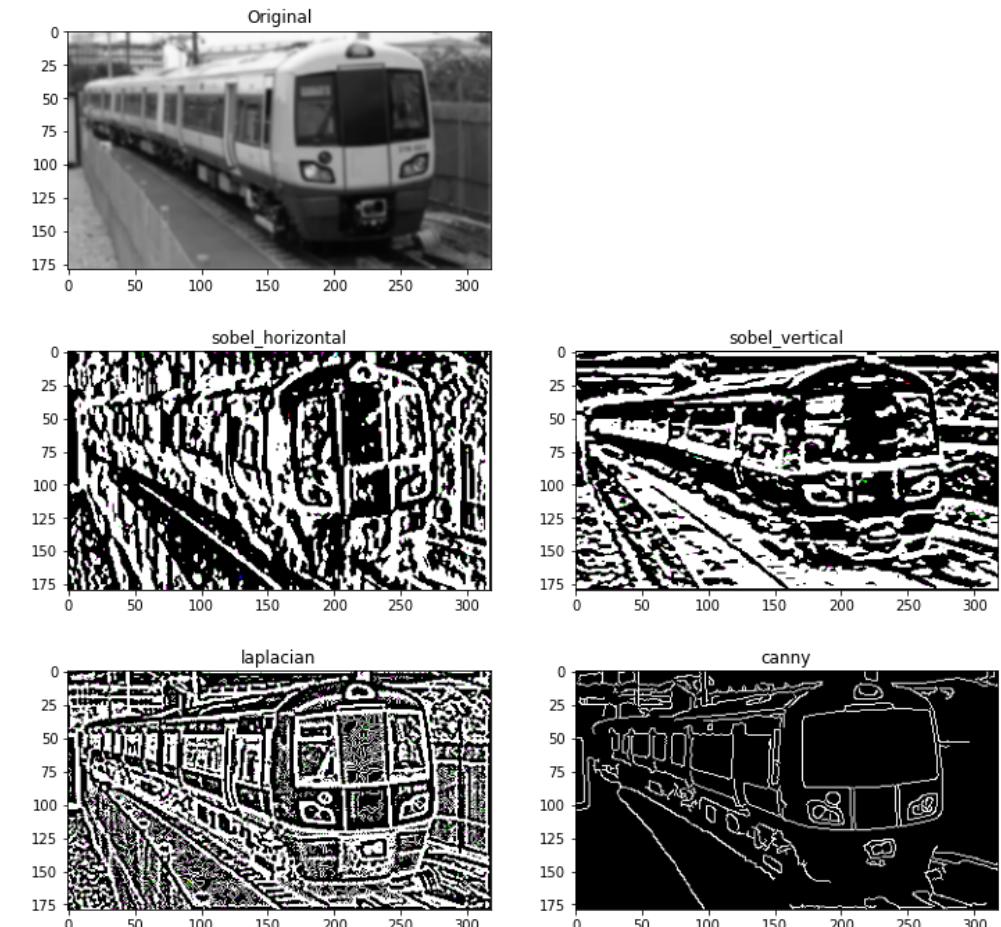
과제 1

- 이미지에 GaussianBlur를 적용하시오.
 - sigma = 0.8, 3일 때, 각각에 대해 연산하시오.
 - sigma가 0.8일 때, 3*3 필터와 5*5 필터 중 어느 것과 비슷한가?
 - 스마트 캠퍼스에서 이미지 (train_input.png) 사용
 - Week7.ipynb 파일의 Assignment 1에서 코드 작성



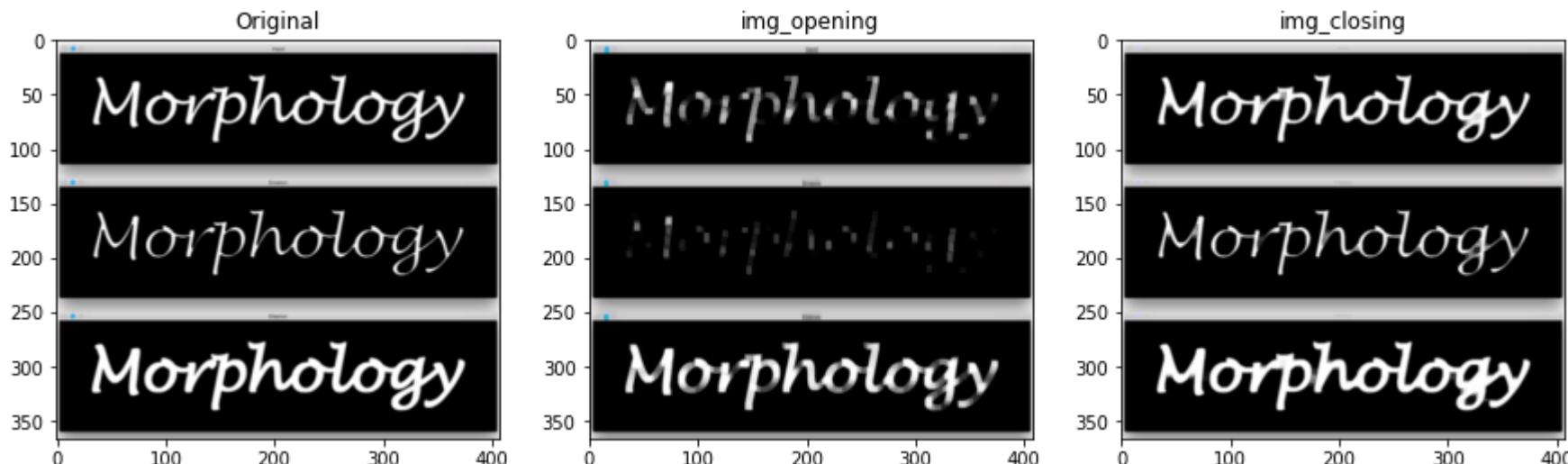
과제2

- 과제 1번의 결과에서 엣지 검출을 수행하시오.
 - Gaussian Blurring이 적용되어 있는 과제 1번에 Sobel, Laplacian, Canny를 적용하시오
 - Gaussian Blurring이 적용되어 있지 않은 것과 비교하였을 때, 어떻게 변했는지 설명하시오
 - Week7.ipynb 파일의 Assignment 2에서 코드 작성



과제3

- 이미지에 열림(opening), 닫힘(closing)의 결과를 구하시오.
 - cv2.morphologyEx() 함수를 이용
 - 스마트 캠퍼스에서 이미지 (text_input.png) 사용
 - Week7.ipynb 파일의 Assignment 3에서 코드 작성



감사합니다