

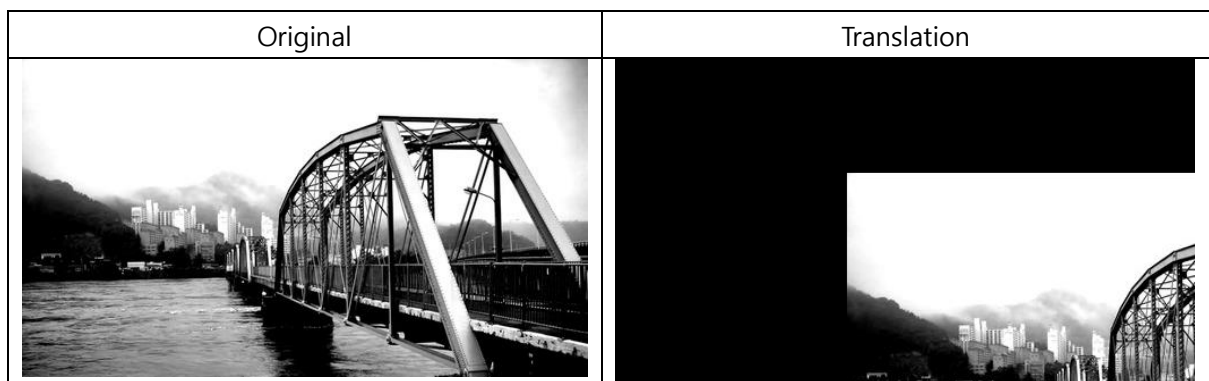
컴퓨터비전 과제 #1

조아라 2021710127

[문제 1. Image Transformations]

1-1 Translation, Rotation, Similarity Transformation 행렬을 구성해보세요. 이 때, 각 행렬의 파라미터는 자유롭게 설정하세요. 그리고, 구성된 행렬을 OpenCV Library의 cv2.warpAffine 함수를 이용하여 임의의 이미지에 적용해 변형시켜보세요.

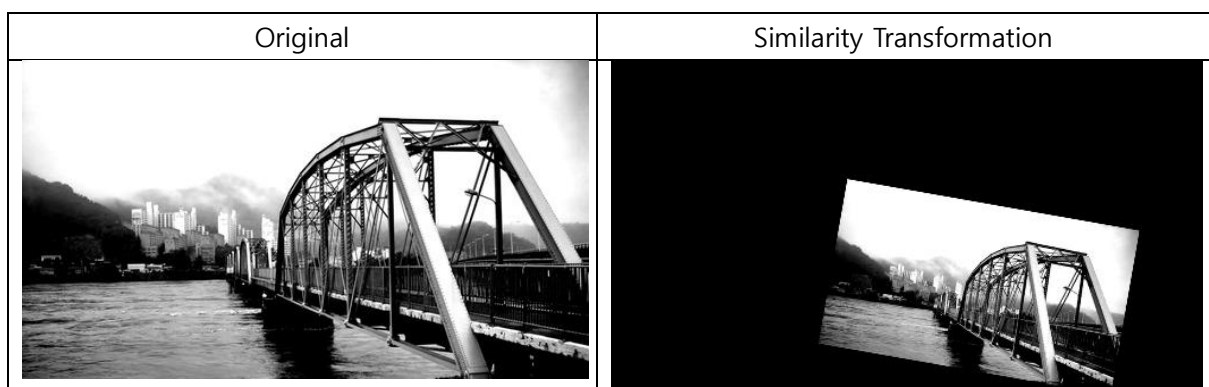
Translation : (200,100) 이동



Rotation : 15도 회전



Similarity Transformation : 10도 회전, 0.5배

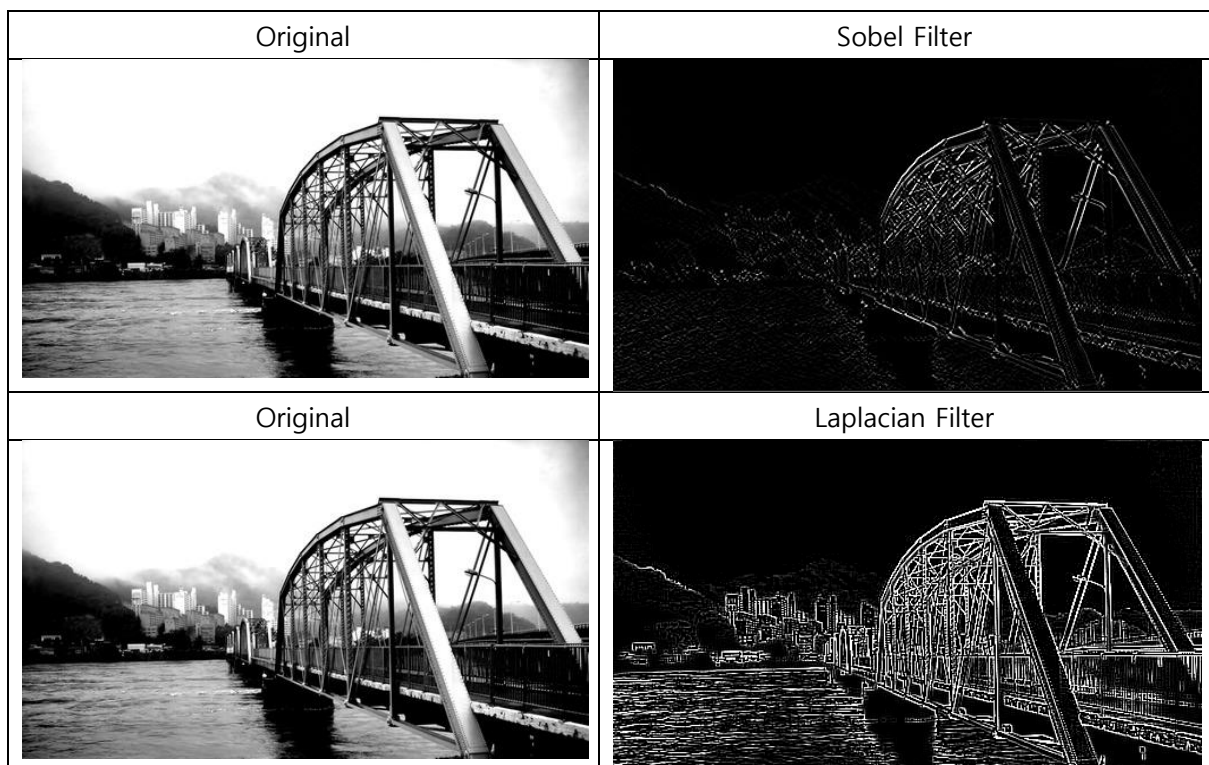


[문제 2. Filters]

2-1. OpenCV Library를 이용하여 Gaussian Filter를 임의의 이미지에 적용해보세요. 이 때, 필터의 크기/분산은 자유롭게 정해보세요.



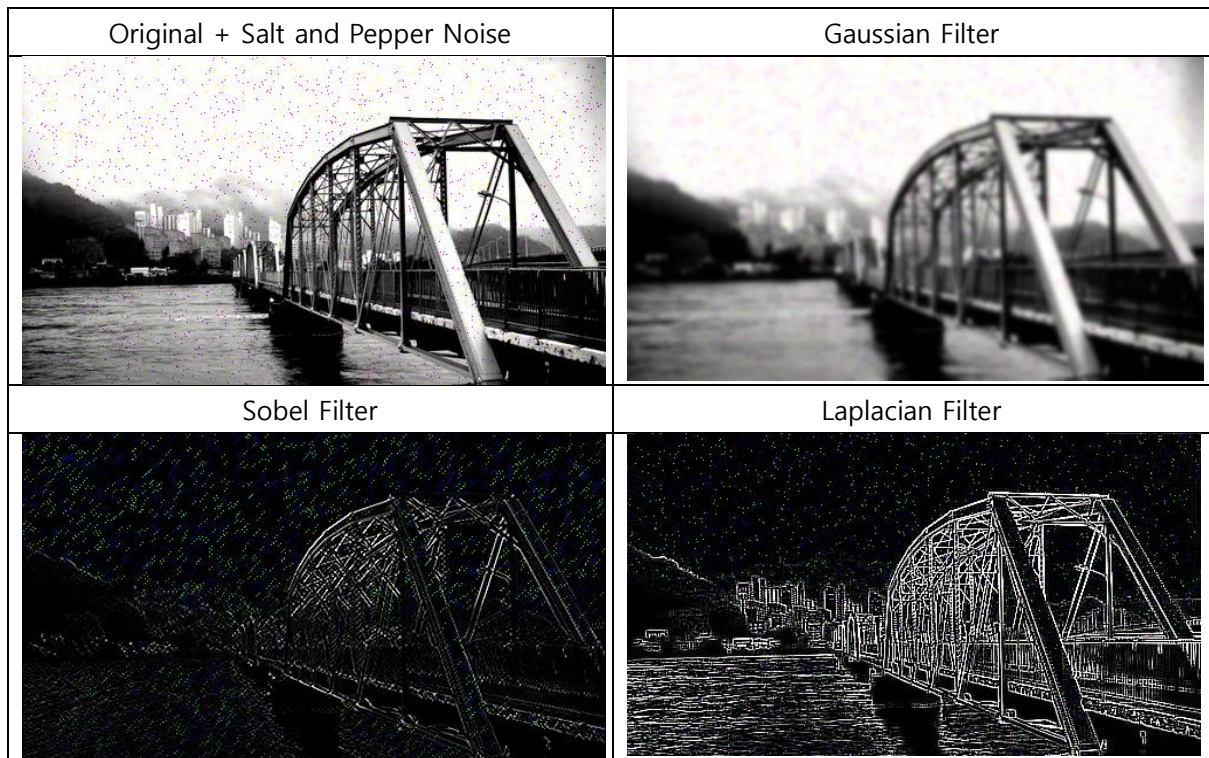
2-2. OpenCV Library를 이용하여 Sobel Filter 와 Laplacian Filter를 임의의 이미지에 적용해보세요. 이 때, 필터의 파라미터는 자유롭게 정해보세요.



2-3. 임의의 이미지에 Salt and Pepper Noise를 추가하고, 앞서 구현한 Gaussian Filter, Sobel Filter, Laplacian Filter를 적용해보고, Noise가 감소/증가 하였는지 분석해 보세요.

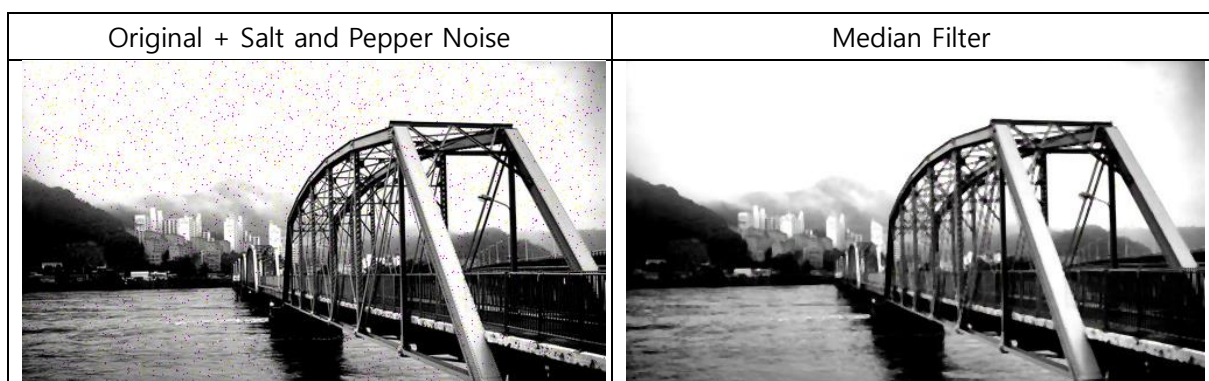
> Salt and Pepper Noise 추가 하고 각각의 Filter를 적용 하였을 경우 아래와 같은 이미지를 얻을

수 있었습니다. Noise를 추가하였을 경우 눈으로 보았을 때 Gaussian_blur < Laplacian filter < sobel filter 순서로 노이즈가 증가하는 것을 확인 할 수 있었습니다.



2-4. 앞서 Salt and Pepper Noise가 추가된 이미지에 cv2.medianBlur함수를 이용하여 Median Filter를 적용시켜보세요. 그리고 2-3에서의 Filter들의 결과와 비교해보세요.

> Salt and Pepper Noise를 적용한 이미지에 Median Filter를 사용했을 경우 앞에서 적용한 3가지 Filter 방법에 비해 Noise가 거의 제거된 결과를 얻을 수 있었습니다.



[문제 3.Image Pyramids]





3-1. 임의의 이미지를 가지고, OpenCV Library의 cv2.resize 함수를 이용하여 Up-sampling과

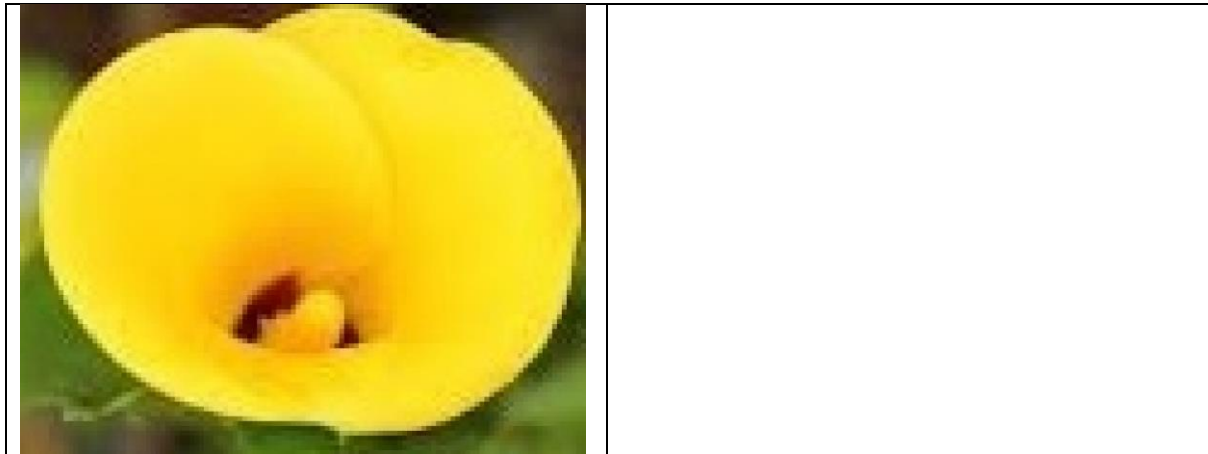
Down-sampling을 적용해 보세요. 이때, Interpolation의 여러방식(예를 들어 Bi-linear, Cubic 등)을 적용시켜보고 결과에 대한 비교를 설명해 보세요.

> 쌍 선형 보간법, 큐빅 보간법, 영역 보간법, 가장 가까운 이웃보간법 방식을 이용하여 이미지의 Up-sampling을 진행하였습니다. 그 결과 영역 보간법, 가장 가까운 이웃보간법은 이미지 모서리의 계단 현상이 발생하였으나, 이에 비해 쌍 선형 보간법, 큐빅 보간법을 적용한 이미지의 경우 Up-sampling이 잘 이루어진 것을 확인 할 수 있었습니다.

Down-sampling이미지 모서리 계단 현상이 모두 존재하였으나, 영역 보간법을 적용하였을 때 보다 나은 이미지를 얻을 수 있었습니다.

Up-sampling

Original	Cubic
	
Bi-linear	영역 보간법
	
가장 가까운 이웃보간법	





Down-sampling

Original	Cubic
	
Bi-linear	영역 보간법
	
가장 가까운 이웃보간법	
	

3-2. 임의의 이미지를 가지고 OpenCV Library의 `pyrDown` 함수를 이용하여 Gaussian Pyramid를 구성해보세요. 그리고, 이미지들의 퀄리티와 용량을 비교해보세요.

> `pyrDown`을 반복할수록 블러링 효과를 낸 듯한 이미지로 보이게 되며 이미지가 깨지는 현상이 발생하였으며, 이미지의 용량은 41.6KB -> 14.1KB -> 4.5KB -> 1.87KB 로 약 1/3정도로 변경되었습니다.



2회적용	3회적용
	

[문제 4. Median Blur 직접구현]

4-1. 앞서 적용해보았던 Median Blur를 Numpy 라이브러리를 활용하여 직접 구현해보세요. 이때, np.sort와 같은 모든 라이브러리 사용이 가능합니다. 단, cv2.medianBlur()를 사용하지 않고 구현한 후, 결과를 검증하기 위하여 cv2.medianBlur() 결과와 직접 구현한 결과를 비교해보세요.

> 직접 구현한 Median Blur와 cv2.medianBlur()를 적용한 두 이미지를 비교해보면 비슷해 보이지만 직접 구현한 Median Blur를 적용한 이미지의 경우 전반적으로 조금 더 어둡게(진하게) 표현 된 것을 확인할 수 있었습니다.

Median Blur	My Median Blur
	