ICV Assignment #2 – Non local filtering

2018-22635 임형석

1. Problem Definition

이번 과제에서는 주어진 이미지에 gaussian noise σ = (20,30,40)을 각각 추가한 후 해당 noise image에 대해서 original non local filter, rotation-invariant patch matching nlm filter, scale-invariant patch matching nlm filter를 이용해 noise image를 denoising 하는 것이 목표이다. 또한 위에서 언급된 non local filter를 denosing하고 각각의 PSNR을 구하여 denoising의 성능을 알아보는 것이 목표이다.

Non-local-filter algorithm은 주로 Speckle한 노이즈를 제거하는데 많이 사용하는데, search window은 search window의 neighborhood patch들 중 유사한 구조를 가진 지점이 존재하고, 그러한 유사한 구조를 가진 지점들을 평균해주면 한 장의 사진만을 사용해 여러 장의 사진을 평균 낸 것과 비슷한 효과를 낸다고 볼 수 있다. NLM은 사진 주변부(local) 뿐만 아니라 사진 전체를 활용한 필터라서 다른 필터에 비해 더 좋은 성능을 기대할 수 있다. 전반적인 수식 형태는 다음과 같다.

$$\mathrm{NL[v](i)} = \sum_{j \in i} w(i,j)v(j)$$

$$\mathrm{w(i,j)} = \frac{1}{Z(i)} e^{-\frac{\|v(N_i)-v(N_j)\|_{2,a}^2}{h^2}}$$

Rotation,Mirror invariant하게 만들기 위하여 각각의 patch들을 sift descriptor에서 사용한 것과 같은 방식으로 patch들의 Dominant Gradient Orientation을 구하여 denoise시키고, Scale invariant하게 만들기 위해 이미지의 gaussian pyramid를 생성한 후 다양한 scale image에 대해 patch를 수집하고, 그것을 NLM filter에 적용하게 되면 Scale invariant한 Denoising 방법이 된다.

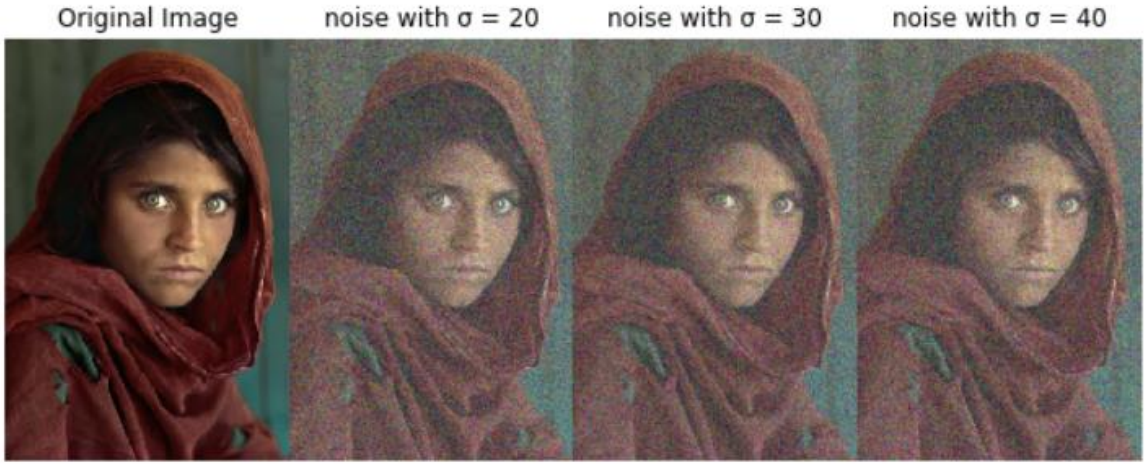
2. Dependencies

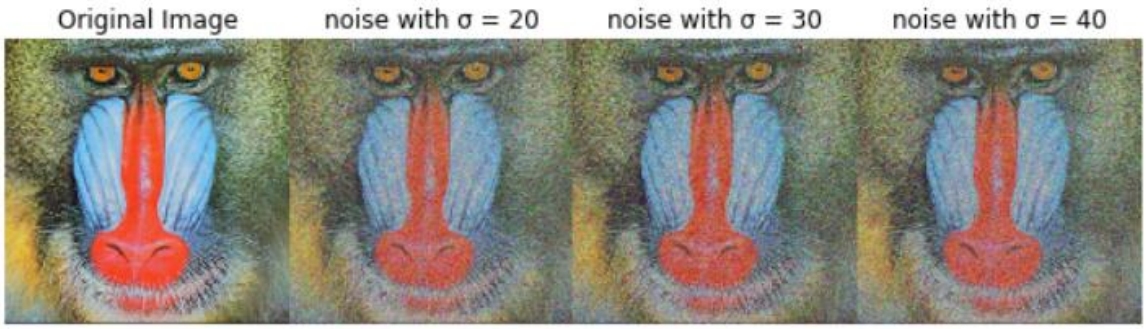
1. Ubuntu 18.04
2. Python 3.6.8
3. Numpy==1.16.2
4. Matplotlib==3.0.3
5. Opencv-python==3.4.6

6. Opencv-contrib-python=3.4.6
7. Scipy==1.2.1
8. Tqdm==4.31.1

3. Results

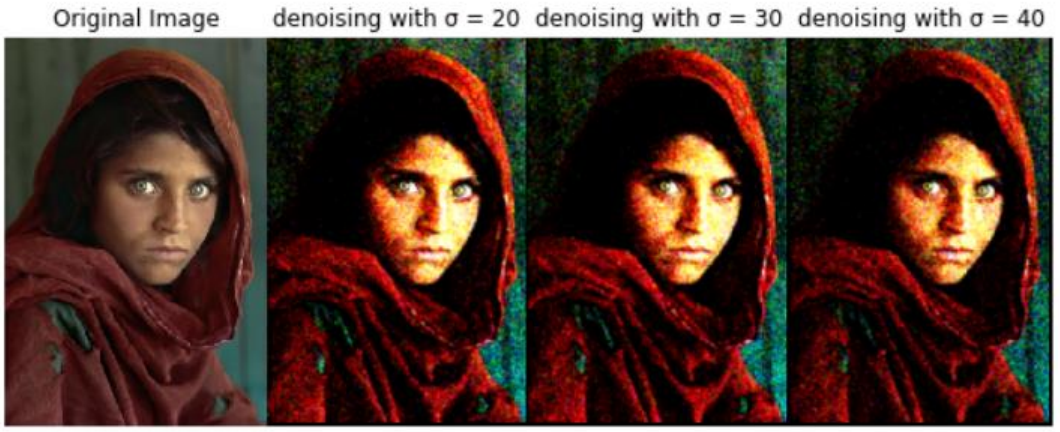
1. Noise image with gaussian noise σ = (20,30,40)
   - 원본 이미지에 mean=0, std =(20,30,40)을 각각 적용한 결과

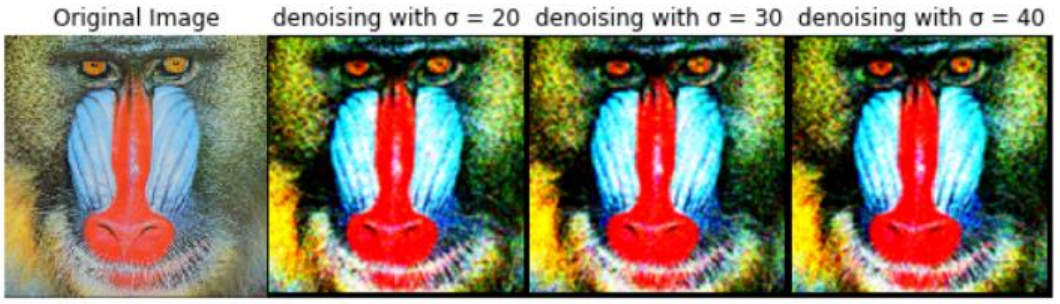




2. Non Local Means filter
   - Original NLM filter로 Denoising 한 결과이다.
   - 한 장당 약 442/6 = 76초정도 소요된다.
   - 육안상으로 noise 이미지보다는 랜덤 n

```
Time : 442 sec
```

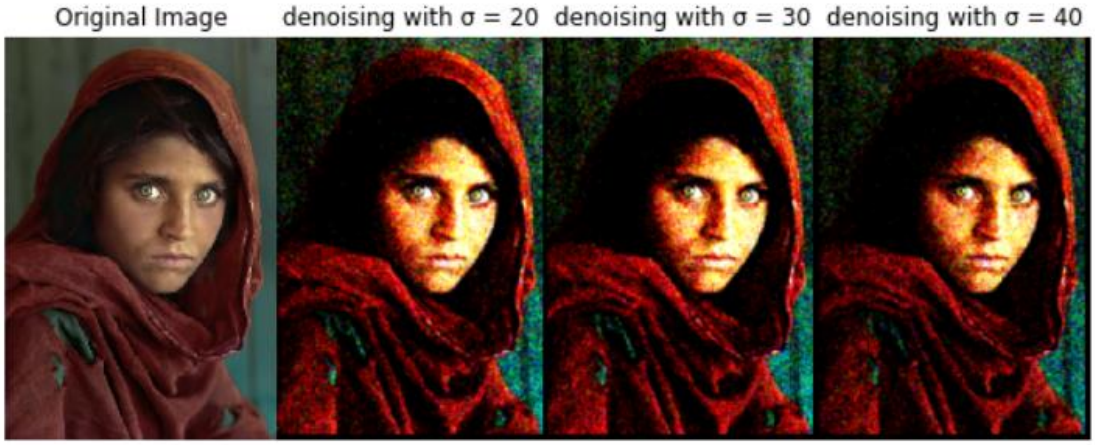


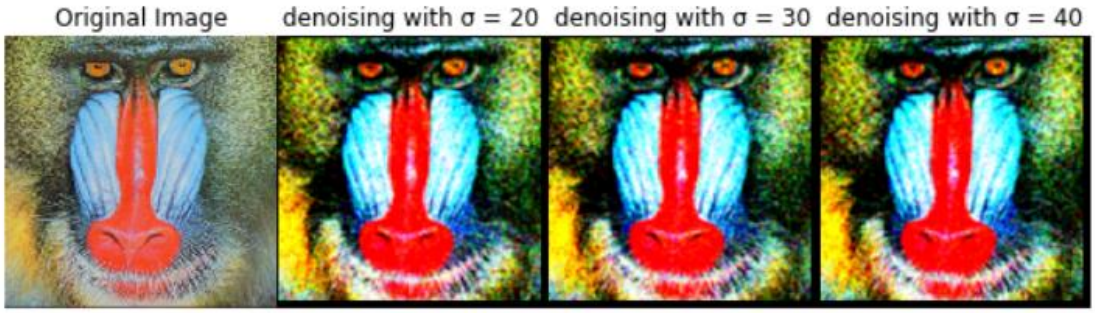


3. Rotation and Mirror Matching

- Original NLM filter에서 각 patch들의 Euclidean distance를 바로 계산하지 않고 Sift descriptor를 구한 것과 같이 dominant Gradient Orientation을 구한 다음 euclidean distance를 계산하여 patch가 Rotation invariant 하게 만든 후 denosing한다.
- 한 장당 2423/6 = 403초 가량 걸린다.
- 모든 patch에 대해서 x,y에 대한 gradient를 구하고, descriptor를 찾아내기 위해 연산이 매우 많아져 기존의 알고리즘보다 5~6배 정도 느려진 것을 볼 수 있다.
- 해당 이미지들에 대해서 육안 상으로는 기존 알고리즘과 수정된 알고리즘의 denoising 결과가 큰 차이가 없으며 실제 PSNR 값도 큰 차이가 없다.

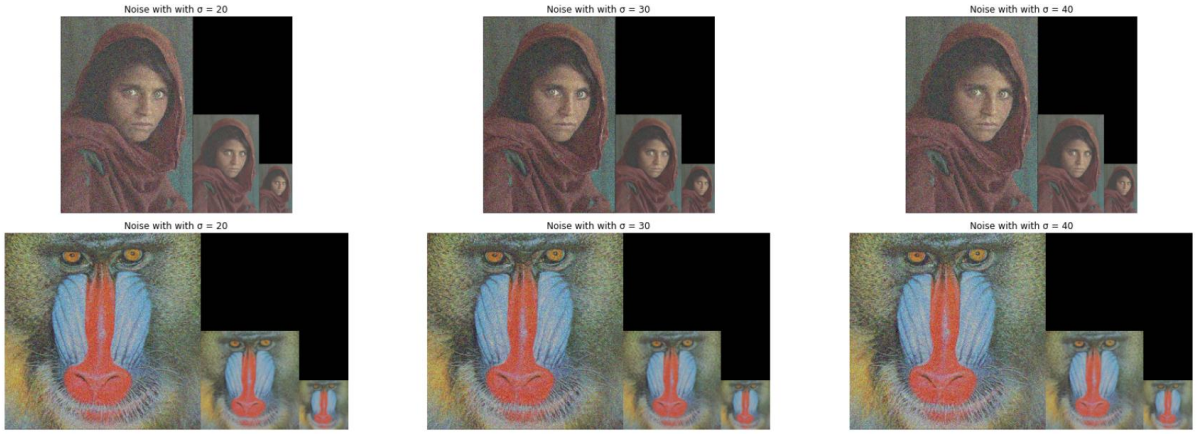
```
Time : 2423 sec
```





4. Scale-Space Search

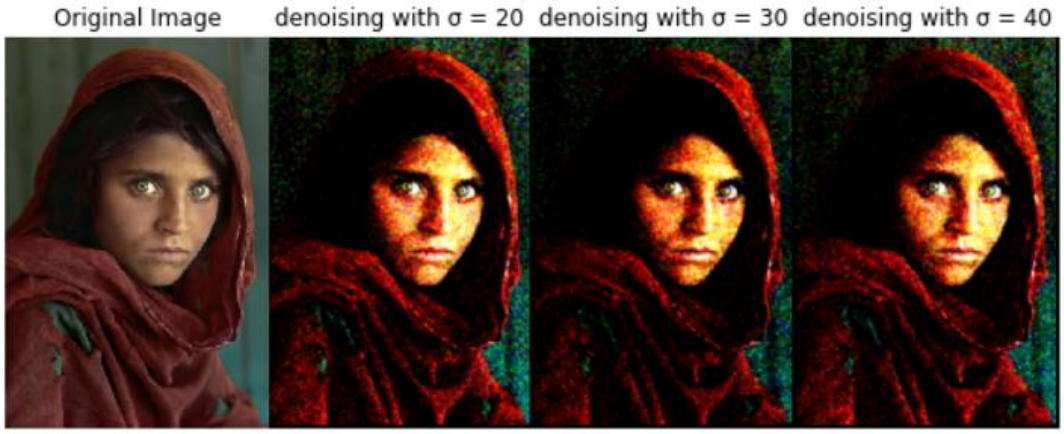
- Image에 대해 Gaussian pyramid 생성

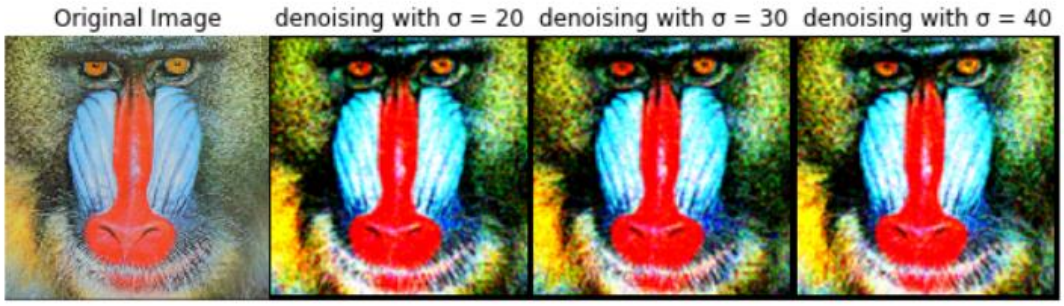


- Gaussian pyramid로 만든 모든 이미지들을 이용해 patch를 original image뿐만 아니라 scale이 달라진 image들에서도 구하여 서로 다른 scale을 가진 patch들을 구한 후, original nlm filter에서 neighborhood patch로 사용한다.
- 시간은 463/6=77초로 scale이 다른 patch들 때문에 original nlm보다 조금

연산량이 늘은 것을 알 수 있다.

```
Time : 463 sec
```





5. PSNR

- 전반적으로 알고리즘들이 비슷한 PSNR값을 가진다.

Image 1	σ = 20	σ = 30	σ = 40
Noise	18.190	18.635	18.56
NLM filter	28.324	28.272	27.097
Rotation	28.528	28.577	28.637
Scale	28.528	28.564	28.629
Image 2	σ = 20	σ = 30	σ = 40
Noise	13.089	13.050	15.996
NLM filter	27.578	27.579	27.602
Rotation	28.601	29.164	29.501
Scale	28.292	28.791	28.832

4. Discussion

1. Analyze your results. Do the modified algorithms improve the performance? If not, why?

전반적으로 수정한 denoising algorithm들은 noise image보다 좋은 PSNR 성능을 보였지만 denoising algorithm들끼리는 큰 성능차가 보이지 않았다.

Gaussian pyramid를 이용한 방식은 다양한 scale의 patch를 사용하였기 때문에, 다른 scale의 patch로 탐색하는 과정에서도 Euclidean distance가 조금이라도 있으면 평균에 더해지게 되니 노이즈가 조금이라도 더 낄 수 있어서 같은 scale의 image에 대해서는 성능 차이를 크게 느낄 수 없겠지만, 다른 scale의 image에 대해선 original nlm filter보다는 더 좋은 성능을 기대할 수 있을 것이라 생각한다.

SIFT descriptor를 이용한 방식 역시 마찬가지로 큰 성능 차이가 나지 않는데 patch 크기가 sift descriptor를 사용하기엔 너무 작기때문에 그것을 이용해distance를 구하는 방식이 크게 의미가 없을 것이라 생각한다.

2. Do you have any idea to improve the performance or enhance the running time?

기본적으로 NLM filter에서는 4중 for문으로 되어있기 때문에 각 장당 걸리는 시간이 매우 큰 편이다. 이를 해결하기 위해서는 첫 번째로 속도가 느린 편에 속하는 python 대신에 빠른 컴파일러를 가진 c,c++을 사용하거나, gpu 사용을 극대화 시킬 수 있는 플랫폼을 이용하거나 cuda언어로 코드를 작성하여 속도를 빠르게 할 수 있다. 현재에는 nlm algorithm에서는 4개의 for문, rgb channel에 대해 1개의 for문을 사용하고 있는데, matrix operation을 지금보다 더 적극적으로 사용하게 되면 지금보다 훨씬 더 빠른 속도를 낼 수 있다. Performance를 극대화 시키기 위해서는 patch size를 더 줄여서 최대한 많은 patch들이 평균 낼 수 있게 하면, 더 정교한 결과를 낼 것이라고 생각하지만 그만큼 그 안에서 도는 연산량이 많아지기 때문에 running time을 반대로 많이 길어질 것이다.