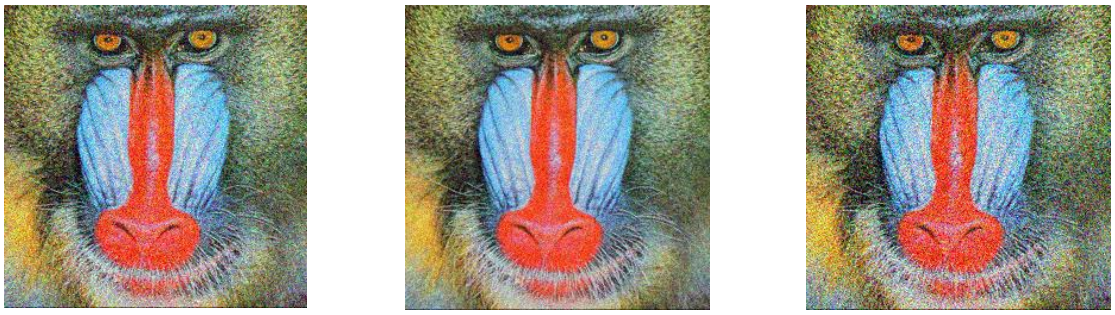
Introduction to Computer Vision Assignment #2

2019-23191 박상욱

1. Noise image generation

Clean 이미지에 mean값 0과 standard deviation인 $\sigma = (20, 30, 40)$를 이용해 Noisy 이미지를 생성하였다. './data/data_clean' 위치의 각 이미지 파일에 대해 './data/data_nosiy' 폴더 내에 Noisy 이미지를 생성한다.

$\sigma = 20$ $\sigma = 30$ $\sigma = 40$



$\sigma = 20$ $\sigma = 30$ $\sigma = 40$

Figure 1. Noisy images

[1] A non-local algorithm for image denoising, by Antoni Buades, Bartomeu Coll, Jean-Michel Morel, CVPR 2005

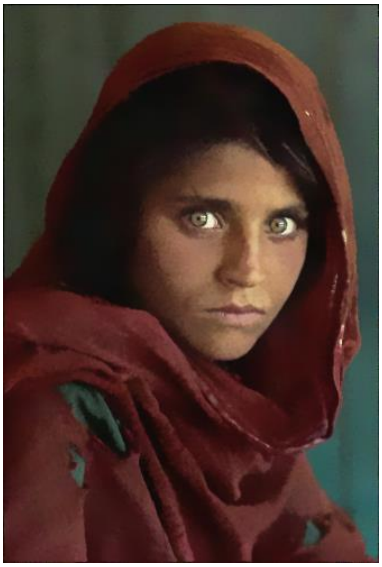
2. Implementation

Pixel의 Intensity를 Neighborhood와 Patch를 이용해 계산하는 기본의 NLM 알고리즘을 구현하였다. Denoise에 사용되는 σ는 Noisy 이미지를 생성하는데 사용한 값을 사용하였고, Neighborhood의 block size 21, Patch의 size 7, h는 10 * σ을 사용하였다.([1]의 Parameter를 이용하였다.) 최종적인 Pixel의 Intensity는 Neighborhood에서 유사한 Patch의 중심 Pixel Intensity에 큰 영향을 받아 생성된다.

구현한 필터를 이용하여 Denoising 한 결과는 Fig2와 같다. Denoise한 이미지 결과는 'res/naïve_res' 폴더에 저장하였다. PSNR 값은 Denoise한 이미지와 Clean 이미지를 이용해 계산하였다. Uint8의 Color 이미지의 PSNR 계산을 위해, 아래의 수식을 이용하였다.

$$\text{PSNR} = 20 \times \log_{10}\left(\frac{255}{\sqrt{MSE}}\right)$$

$$\text{MSE} = \frac{1}{3mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\sum_{c=0}^{2}[I(i,j,c) - \bar{I}(i,j,c)]^2$$



PSNR 33.39
Running Time 1349.45s

PSNR 31.37
Running Time 4013.13s

PSNR 26.60
Running Time 3948.86s

PSNR 23.92
Running Time 530.16s

PSNR 24.60
Running Time 1582.53s

PSNR 22.66
Running Time 1569.41s

Figure 2. Denoised images

3. Rotation and Mirror matching

NLM 알고리즘은 기본적으로 Noisy image에서 유사한 패치를 찾아서 더 유사한 pixel이 큰 영향을 주므로 유사한 패치를 많이 찾는 것이 유리할 수 있다. 이를 위해 Patch를 Rotation, Mirror하여 NLM 필터의 성능이 좋아지는지 살펴보고자 하였다. 이를 위해 SIFT descriptor의 아이디어를 이용하였다. 먼저 이미지의 각 픽셀에서 gradient의 direction과 magnitude를 계산한다. 계산한 gradient로부터 NLM에서 사용되는 각 패치에서 8개로 binning 된 Orientation feature를 찾을 수 있다. 이로부터 Orientation을 Normailization하면, 8개로 binning 된 vector는 각 패치의 feature vector가 된다. 이 feature vector를 이용해 각 패치의 feature vector의 차이를 Euclidean distance로 사용하여 NLM 식에 적용하였다. 이런 경우 Patch의 Orientation 정보가 비슷하면 해당 픽셀의 Intensity가 결과 Intensity에 큰 영향을 줄 수 있도록 의도하고자 하였다.

다만 결과 이미지는 제대로 나오지 않았고, Patch의 feature 정보가 올바르지 않은 방식으로 구하여 의도하지 않은 결과가 나온 것으로 생각한다. 8차원의 feature vector는 Orientation 정보만 가지고, Intensity의 정보를 담고 있지 않아 발생하는 문제로 보인다.

4. Scale-space Search

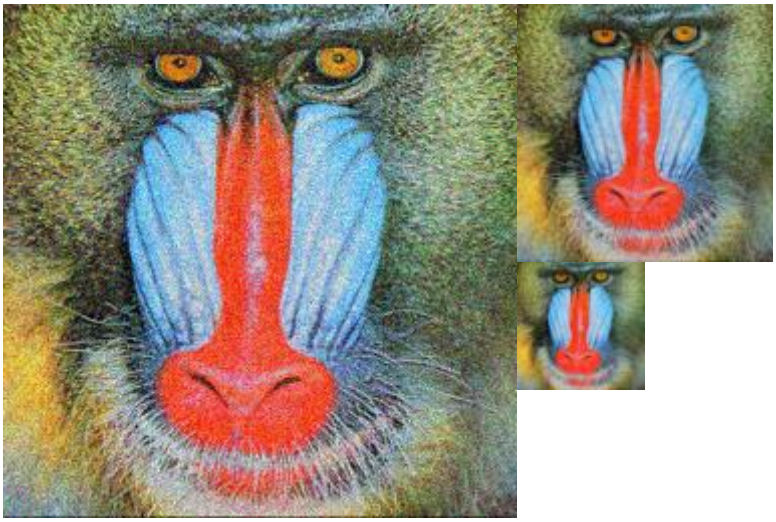


Figure 3. Image Pyramid for monkey with gaussian noise(sigma = 20)

Scale-space search를 위해 3 level Gaussian Image Pyramid를 구축하였다. Image Pyramid에는 원본 이미지를 1/2배, 1/4배 subsampling 한 것을 사용하였다. NLM 알고리즘에서 구축한 Pyramid에서도 Patch 탐색을 수행하도록 하였다.

PSNR 31.31 PSNR 29.03 PSNR 26.64

PSNR 24.62 PSNR 23.77 PSNR 22.99

Figure 4. Denoised image with scale-space search

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

유사한 Patch를 많이 찾는다고 해서 기대만큼 결과가 좋아지지 않았다. Scale-space search에서도 눈으로 보기에 개선되었다고 생각했으나, PSNR 값을 구하면 실제로 큰 효과가 없는 것을 확인할 수 있었다. 다만 조금 더 이미지가 sharpen해지는 효과가 있는 것으로 보인다. Similarity 값이 더욱 큰 Patch를 많이 찾았음에도 큰 효과가 없는 것은 Euclidean distance에 Gaussian을 사용하면서 Patch가 매우 유사할수록(distance가 0에 가까움) 큰 영향(weight)를 받고, 유사도가 크게 떨어지는 Patch의 영향은 거의 받지 않게 된다. 따라서 유사한 Patch를 더 많이 찾는다고 하더라도, 유사도가 크게 떨어지는 Patch의 영향(weight)은 크게 바뀌지 않게 되어 성능 개선에 큰 도움이 되지 않는다.

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

1) Denoising의 성능 향상을 위해서는 현재 구현한 방법은 Patch의 중심 Pixel의 Intensity만 영향을 주고 있다. 이를 해당 Patch 전체에 영향을 줄 수 있도록 수정하면 조금 더 Clear한 이미지를 얻을 수 있을 것이다. 또한 Patch 간의 distance를 단순히 Euclidean distance로 계산하고 있는데, Patch내의 pixel-wise 연산에서 Gaussian function을 이용해 Patch의 중심에서 멀어질수록 Intensity 차이의 영향을 줄이는 방법을 수행할 수 있을 것 같다.

2) 현재 연산의 속도가 매우 느린 것은 각 픽셀마다 Neighborhood 영역을 구하고, 그 Neighborhood내에서 각 neighbor마다의 Patch를 구해 현재 픽셀의 Patch와 Euclidean distance를 구하기 때문이다. 즉 Neighborbood의 픽셀 수를 n개, Patch의 픽셀 수를 p개라고 하면 O(nXpXheightXwidth) 알고리즘이라고 할 수 있다. 연산수를 줄이기 위해, Euclidean distance를 구하는 것이 아닌 각 Patch의 Mean값을 이용해 계산할 수 있을 것이다. Patch 마다의 Mean은 미리 계산할 수 있으므로, O(pXheightXwidth)의 방법으로 생각할 수 있다.