|  |
| --- |
| 01. Images  Noise & Filtering |

|  |
| --- |
| **과제 요약** |
| **1. 이미지에 Salt-and-pepper, Gaussian Noise 생성.**  **2. Average, Gaussian, Median 필터 적용.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **제출 일자** | 2018. 10. 02 | **전 공** | 컴퓨터공학 |
| **과 목** | 영상처리 특론 | **학 번** | 1004660 |
| **담당 교수** | 이 준 재 | **이 름** | 박 준 수 |

# ***요 약***

*이미지에는 다양한 변수들로 인해 원하지 않는 신호, 노이즈(noise)가 생길 수 있다. 이를 해결하는 테크닉은 블러(blur), 필터(filter), 스무딩(smoothing) 등으로 알려진 기법이 있다. 이는 각각의 알고리즘의 처리 방식 마다 대상을 선정하는 방법은 다르지만, 일반적으로 근처의 대상을 선정해 노이즈가 발생한 지점의 값을 보정하는 방법들을 취한다.*

*본 레포트에서는 이미지에 Gaussian, Salt-and-pepper 노이즈를 발생시키고, 각각의 이미지들에 몇가지 필터들을 적용해봄으로써, 각 필터들의 노이즈 제거 효과에 대해 실험해보고 그 결과를 검증했으며, 결과적으로 Salt-and-pepper 노이즈에는 Median 필터가 효율적이며, Gaussian 노이즈는 각각의 필터가 육안으로 그 차이를 확인하기 힘든 정도의 차이만 보임을 알 수 있었다.*

# **1. 서 론**

신호처리 분야에서 노이즈(잡음, noise)는 원치 않는 측정, 저장, 전송, 처리 및 변환 등의 과정 중 변형된 신호를 총칭하는 말이다. 즉, 노이즈의 원래 의미는 “원하지 않는 신호”이다 [1]. 이를 이미지 처리 분야에 적용하면 밝기, 색 등의 정보가 임의로 변형된 것으로 말할 수 있으며, 이미지 처리 분야에서 일반적으로 알려진 노이즈는 다음과 같다:

* Gaussian noise
* Salt-and-pepper noise
* Shot noise
* Quantization noise (uniform noise)
* Film grain
* Anisotropic noise
* Periodic noise

이들 중, Gaussian 노이즈는 Gaussian 분포를 가지는 통계적인 노이즈(statistical noise)이며[2], 노이즈의 값들이 가우스 분포를 가진다.

|  |
| --- |
| ììì´ ìë¤. With Gaussian noise |

Figure 1. 원본 이미지(좌)와 Gaussian 노이즈 이미지(우)[3]

Gaussian 노이즈는 일반적으로 고온, 조명 불량, 회로 잡음 등의 이유로 이미지 취득 중 발생한다. 이를 해결하기 위해서는 일반적으로 Median, Gaussian 필터를 사용한다.

임펄스 노이즈라고도 불리는 Salt-and-pepper 노이즈는 이미지 신호에 임펄스 신호(날카롭게 튀는 신호)가 생겨 픽셀의 값이 극단적(0/255, black/white)으로 변하는 노이즈다.

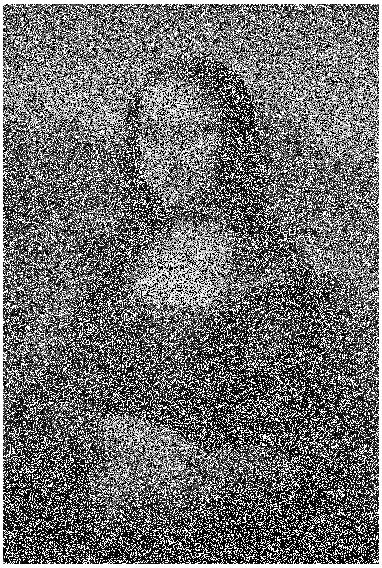


Figure 2. 임펄스 노이즈 이미지[4]

임펄스 노이즈에 대한 효과적인 방법으로는 메디언 필터나 형태 연산 필터(morphological filter) 등이 알려져 있다[5].

# **2. 노이즈 필터링**

본 과제에서는 임의의 이미지에 두가지 종류의 노이즈(Gaussian, Salt-and-pepper noise)를 발생시키고, 노이즈가 생긴 이미지에 몇가지 필터(Average, Gaussian, Median)를 적용해 보고자 한다.

원본 이미지는 이미지 이미지 처리 알고리즘에 가장 많이 사용되는 이미지 중 하나인 Lenna 이미지를 선택했다(figure 3 참고).



Figure 3. Lenna 원본 이미지(512 x 512)

**2.1. 노이즈 이미지 생성**

아래의 CODE 1은 원본 이미지(Figure 3)에 Salt-and-pepper 노이즈를 발생시키는 함수를 보이며(Salt-and-pepper 노이즈는 Open CV[6]에 관련 함수가 없다), Figure 4는 이를 호출해 노이즈가 발생한 이미지를 보인다.

CODE 1. Salt-and-pepper noise 생성 함수[7]

|  |
| --- |
| 1. void salt\_pepper(cv::Mat src\_img, double noise\_percentage) 2. { 3. srand((int)time(NULL)); 4. int noise\_points = (int)(((double)src\_img.rows \* 5. **src\_img.cols \* src\_img.channels()) \* noise\_percentage / 100.);** 6. for (int count = 0; count < noise\_points; count++) { 7. int row = rand() % src\_img.rows; 8. int col = rand() % src\_img.cols; 9. int channel = rand() % src\_img.channels(); 11. uchar\* pixel = src\_img.ptr<uchar>(row) 12. + (col \* src\_img.channels()) + channel; 14. \*pixel = (rand() % 2 == 1) ? 255 : 0; 15. **}** 16. } |

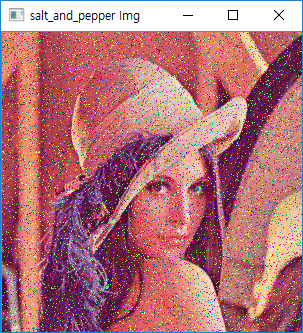


Figure 4. Salt-and-pepper 노이즈가 발생한 이미지

아래 CODE 2는 원본 이미지(Figure 3)에 Gaussian noise를 적용하는 함수를 보이며, Figure 5는 이를 호출해 Gaussian noise가 발생한 이미지를 보인다.

CODE 2. Gaussian noise 생성 함수[7]

|  |
| --- |
| 1. cv::Mat gaussian\_nosie(cv::Mat src\_img, double average, double sigma) 2. { 3. cv::Mat noise\_img(src\_img.size(), CV\_16SC3); 4. cv::randn(noise\_img, cv::Scalar::all(average), cv::Scalar::all(sigma)); 5. **cv::Mat temp\_img;** 6. src\_img.convertTo(temp\_img, CV\_16SC3); 7. cv::addWeighted(temp\_img, 1.0, noise\_img, 1.0, 0.0, temp\_img); 8. temp\_img.convertTo(temp\_img, src\_img.type()); 9. return temp\_img; 10. **}** |

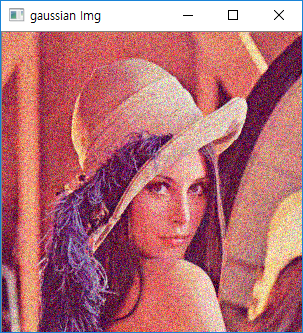


Figure 5. Gaussian 노이즈가 발생한 이미지

**2.2. 필터 적용**

2.1절에서 Salt -and-pepper 노이즈와 Gaussian 노이즈를 같은 이미지에 발생시켰다.

노이즈가 발생한 이미지에 대해 필터를 적용하기 위해서 Open CV[6]가 지원하는 다음의 함수들을 사용하였다(라이브러리 사용 방법은 [8]을 참고함):

* cv::blur(Average 필터)
* cv::medianBlur(Median 필터)
* cv::GaussianBlur(Gaussian 필터)

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure 6. Salt-and-pepper가 발생한 이미지(좌상)와 Average(우상), Median(좌하), Gaussian(우하) 필터 처리 후 이미지

위 Figure 6은 Salt-and-pepper 노이즈를 발생시킨 이미지와 그 이미지에 3가지 필터(Average, Median, Gaussian 필터)를 적용한 이미지들을 보인다.

|  |  |
| --- | --- |
|  |  |
|  |  |

Figure 7. Gaussian 노이즈가 발생한 이미지(좌상)와 Average(우상), Median(좌하), Gaussian(우하) 필터 처리 후 이미지

아래 Figure 7은 Gaussian 노이즈를 발생시킨 이미지와 그 이미지에 3가지 필터(Average, Median, Gaussian 필터)를 적용한 이미지들을 보인다.

# **3. 결론**

이 과제에서 본인은 원본 이미지에 Salt-and-pepper, Gaussian 노이즈를 발생시키고, 노이즈가 발생된 이미지에 몇가지 필터들을 적용해 봤다.

일반적으로 Salt-and-pepper 노이즈를 해결하기 위해서는 Median 필터가 효율적이라고 알려져 있으며, 이를 확인하기 위해 Salt-and-pepper 노이즈를 발생시키고 필터를 적용해본 결과, 다른 두가지 필터에 비해 Median 필터가 보다 원본에 가까운 이미지로 복원됨을 알 수 있었다. 하지만, 원본과 대비했을 때, 번짐(blur) 현상이 일어나, 수채화 같은 이미지로 변함을 알 수 있었다.

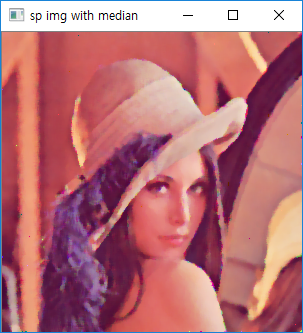


Figure 8. 40%의 발생 비율로 생성된 Salt-and-pepper 노이즈에 Median 필터 적용

위 실험에서 Salt-and-pepper 노이즈의 발생 비율은 10% 였으나, 이를 40%까지 증가 시킨 후, Median 필터를 적용하면, 아래 Figure 8에서 보이는 바와 같이 부분적으로 0, 255 값이 남게 된다. 또한, 이를 적절한 수준으로 임계 값을 지정해 필터를 지정함으로써 수채화 같은 느낌의 이미지 필터를 제작하고 판매하는 애플리케이션의 원리로 이용할 수 있을 것으로 생각한다.

일반적으로 Gaussian 노이즈를 해결하기 위해서는 Median, Gaussian 필터가 효율적이라고 알려져 있으며, 이를 확인하기 위해 Gaussian 노이즈를 발생시키고 세가지 필터를 적용해본 결과, 육안으로는 Gaussian 이 가장 효율적으로 보이지만, 사람마다 이를 다르게 평가할 수 있을 만큼 성능이 비슷했다. 이를 확인하기 위해서는 이미지 유사도 판별이 필요할 것으로 생각된다.

# **4. 참고문헌**

[1] V. Tuzlukov, *Signal processing noise*. CRC Press, 2002.

[2] T. Barbu, "Variational image denoising approach with diffusion porous media flow," in *Abstract and Applied Analysis*, 2013, vol. 2013: Hindawi.

[3] *Gaussian noise - Wikipedia*. Available: <https://en.wikipedia.org/wiki/Gaussian_noise>

[4] *Salt and Pepper Image Denoising Using MATLAB | Wirebiters*. Available: <http://www.wirebiters.com/salt-pepper-image-denoising-using-matlab/>

[5] O. Marques, *Practical image and video processing using MATLAB*. John Wiley & Sons, 2011.

[6] *OpenCV library*. Available: <https://opencv.org/>

[7] K. Dawson-Howe, *A practical introduction to computer vision with opencv*. John Wiley & Sons, 2014.

[8] OpenCV. *OpenCV: Smoothing Images*. Available: <https://docs.opencv.org/3.1.0/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html>