정경주

1. Edge detection operators can be compared in an objective way. The performance of an edge detection operator in noise can be measured quantitatively as follows: Let n_0 be the number of edge pixels declared and n_1 be number of missed or new edge pixels after adding noise. If n_0 is held fixed for the noiseless as well as noisy images, then the edge detection error rate is

$$P_e = \frac{n_1}{n_0}.$$

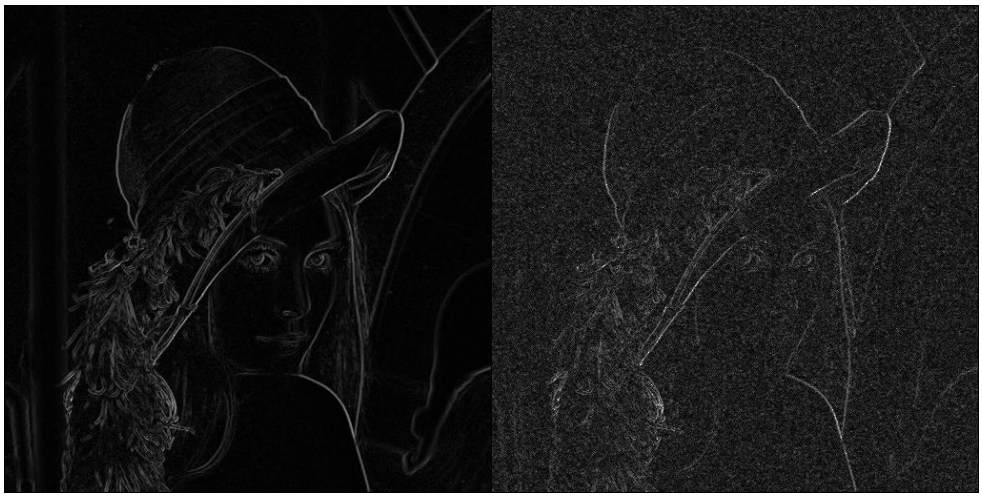
Compare the performance of the gradient operators of Roberts, Sobel, Prewitt and the 5x5 stochastic gradient on a noisy image with SNR= 8dB.

Note that the pixel location (*m*,*n*) is declared an edge location if the magnitude gradient $g(m,n) = \sqrt{g_x^2(m,n) + g_y^2(m,n)}$ exceeds a THRESH value of 150. The edge locations constitute an *edge map*. For this assignment, you can select 512x512 BMP or RAW grayscale image of Lena.

lena_bmp_512x512_new.bmp Noise_img_9dB.bmp

밑에서부터는 왼쪽은 원본 영상에 마스크 적용한 결과, 오른쪽은 9dB 노이즈 영상에 마스크를 적용한 결과입니다.

#1 Roberts Mask



Roberts.bmp Roberts_noise.bmp

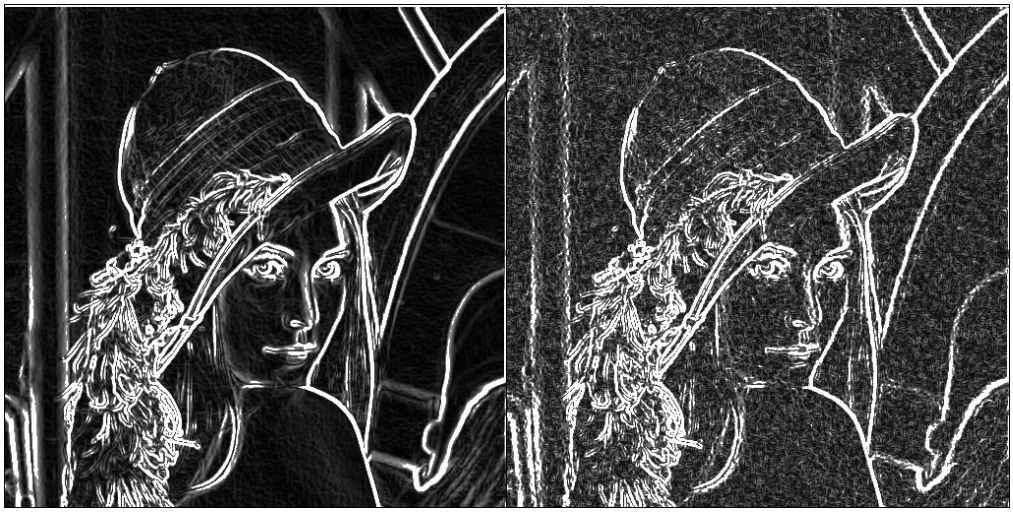
#2 Sobel Mask

Sobel.bmp Sobel_noise.bmp

#3 Prewitt Mask

Prewitt.bmp Prewitt_noise.bmp

#4 5x5 Stochastic Mask



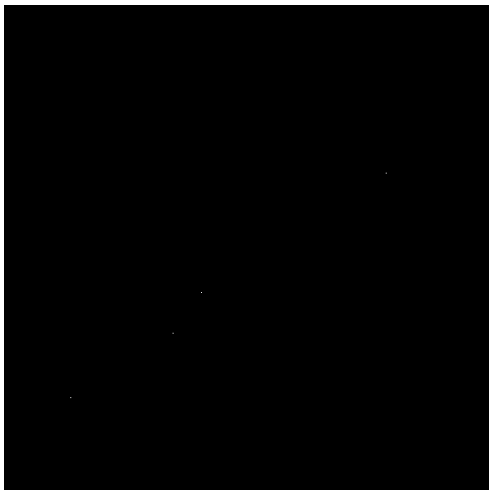
Stochastic.bmp Stochastic_noise.bmp

Discussion#1

기본의 Lena영상에서 8dB 가우시안 노이즈가 낀 영상을 만들어 주기 위해서 일단 교수님이 주신 소스에 variance 구하는 함수를 작성해줘야 한다. 512*512 픽셀 각각 마다 계산해주기 위해 이중 for문을 사용하고, 우리가 알고 있는 $V(X) = E(X^2) - \left(E(X)\right)^2$ 을 계산 하기 위해서 첫째 항에 second_order_moment를 설정하고 두번째 항엔 mean_square를 설정하고 위 식대로 계산해주면 원하는 variance 값이 나온다. variance를 sigma로 바꿔주고 AddGaussianNoise에서 클램핑이 끝난 noise_img 값을 얻을 수 있다.

원본 영상과 노이즈 영상을 가지고 이제 마스크 적용을 시작해야 되는데 그냥 함수 안에서 여러 개의 N*N 짜리 배열을 사용하면 stack overflow가 일어나기 때문에 반드시 동적할당을 통해서 2차원 배열을 생성해줘야 하고 구분하기 쉽게 마스크마다 함수로 나눠줬다. 함수 하나에 원본, 노이즈 영상의 연산이 한꺼번에 돌아가는 형태고 파일 헤더 정보도 함수 변수를 통해 함께 넘어와야 한다.

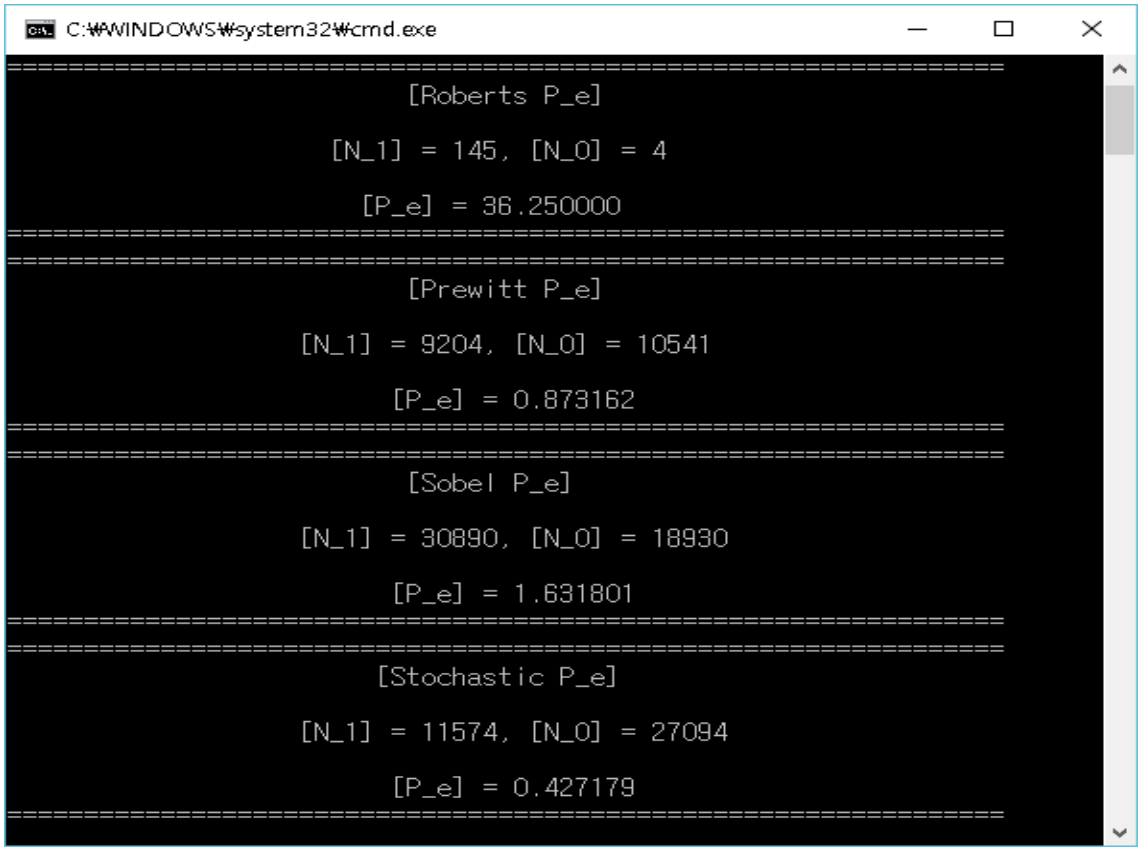
마스크 X, Y 를 사용하면서 가장 먼저 생각해야 할 점은 경계부분의 처리 문제인데 난 여기서 가장 많이 사용되는 (1, 1) 마스크 슬라이딩을 사용하기 위해서 for문의 첫 값은 1, 끝값은 511로 설정해주었다. 바깥은 i,j 이중 for문안에서 마스크 연산이 3x3 으로 이뤄지고 동적할당해준 X, Y 두 배열은 아래에 for문을 돌려서 $g(m,n) = \sqrt{g_x^{\,2}(m,n) + g_y^{\,2}(m,n)}$ 값을 구하고, Threshold =150 을 적용해주면 (값에 따리 희미한 선들을 선명하게 만들어 줄 수 있다) 원하는 영상 결과를 얻을 수 있는데 단순하게 150 이하 값을 0 으로 주고 했더니 영상이 아래와 같이 점이 4개만 보여서 엣지를 판단할 수 없게 되었다.



그래서 threshold 이하 값을 오리지날 영상의 원래 값인 value를 넣어줬는데 어짜피 나중에

P_e 값을 구할 때 255 값이 아니라서 이 값들은 count를 안해주기 때문에 엣지를 좀 더 잘 검출해내기 위해 모든 영상에 다 이렇게 적용시켰다.

동적 할당 메모리 해제 후 두 영상을 각각의 픽셀값에 대해서 비교해준다. 여기서 N_1는 엣지인데 엣지로 안나오는 (원본에서 255, 노이즈 처리 영상에서는 255가 아닌)것과 엣지가 아닌데 엣지로 나오는 (원본에서 255가 아닌데 노이즈 처리 영상에서는 255로 나오는)것 두 가지를 카운트 해서 더해준 값이다. N_0 는 원본 처리 영상에서 픽셀값이 255인 것을 찾아 카운트해준 값이다. $P_e = \frac{N_1}{N_0}$ 을 통해 모든 마스크 처리를 해준 영상에 대한 값을 구해보면

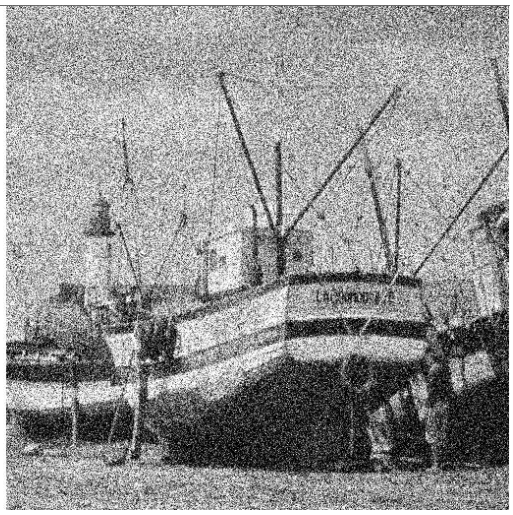


P_e 값이 클수록 좋지 않은 결과이기 때문에 Roberts Mask 결과가 가장 좋지 않고 Stochastic Mask 를 쓰면 결과가 좋다. 하지만 Stochastic은 5x5 Mask 라서 다른 것들 보다 계산량이 많아 지는 단점이 있다.

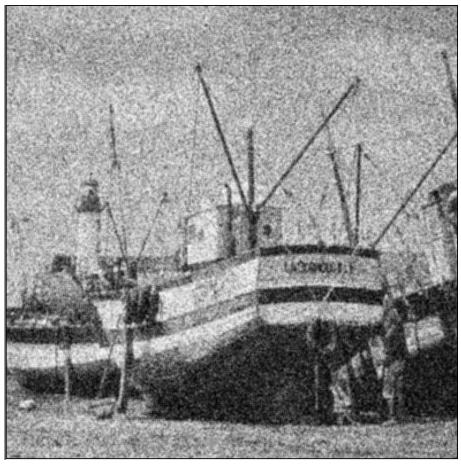
2. Compare the performance between the 3x3 Low-pass and Median filters for a noisy

image with SNR=9dB. For an objective comparison, obtain the MSE (mean square error) for each result. For this assignment, use 512x512 grayscale image of BOAT.raw.

BOAT512.raw



Boat_img_8dB.raw



Lowpass_noise.raw

Median_noise.raw

Discussion#2

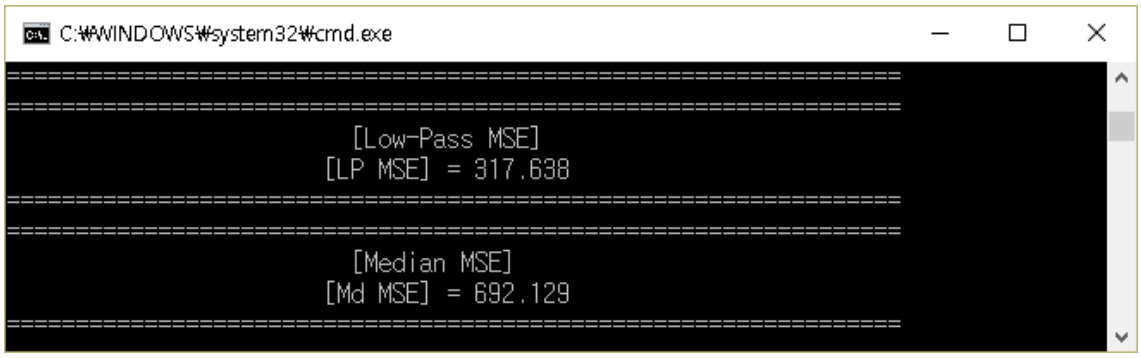
bitmap에서 raw file로 바꾸고 8dB로 바꿔서 하기 때문에 새로 노이즈 영상을 생성해야 한다. Lowpass에서는 앞에 마스크 연산과 거의 비슷하고 같은 3x3으로 하고 안의 값만 1/9 (가중치 합 1, 평균치를 낸거라고 생각하면됨)인 동일한 가중치로 계산해주면 된다. 확인해보면 Boat와 Lena 둘다 블러링 효과가 들어갔음을 알 수 있다. Median에서는 오름차순 정렬을 위해서 일단 앞에처럼 4중 for문을 써주고 해당 픽셀값을 temp에 저장해준다. 그 temp, temp_n 값을 직접 만들어준 sort() 함수(버블 정렬 이용)에 넣어서 3x3(9개)에 대해 비교하고 오름차순으로 정렬을 해주고 가운데 값인 (1,1) 에 해당하는 값을 마지막 출력값에 넣어주면 된다.

filter 에서는 Mean Square Error 를 통해서 성능을 비교할 수 있다. 아래의 식이

$$\frac{\left\{\sum\left(Original - Filtered\ image\ with\ noise\right)^2\right\}}{(N*N)}$$

필터 성능의 척도가 된다. (작을수록 성능이 좋은 것이라고 판단)

과제에서는 가우시안 노이즈를 넣어줘서 했기 때문에 임펄스 노이즈에 강한 Median filter보다는 가우시안 노이즈에 강한 Low-pass filter가 MSE 값이 작게 나왔다. (Median은 임펄스값인 0과 255를 없애버릴 수 있기 때문에 임펄스에 강하고 Low-pass는 평균치를 내면서도 임펄스에 영향을 받기 때문에 임펄스보다는 가우시안에 강함).



과제 1, 2 둘 다 사람들마다 값이 다르게 나오는데 이것은 가우시안 노이즈를 넣을 때 랜덤한 값을 적용시키는 것이기 때문에 차이가 조금씩 난다고 생각한다.

(1) Method to generate a noisy image with Gaussian noise

The signal-to-noise ratio (SNR) is expressed in decibels as

$$SNR = 10\log_{10}\frac{\sigma^2}{\sigma_e^2} \quad \text{(dB)}$$

where σ^2 is the variance of the original image and σ_e^2 is the variance of the noise signal. In order to generate a noisy image with a specified SNR by adding Normalized Gaussian Noise with distribution of $N(0,\sigma_e^2)$, you can refer to the following source codes.

```
variance=get_image_power(input_image);      /* The data type for input _image should be Int */
stddev_noise=sqrt(variance/pow(10.0,((double) SNR/10));    /* Here SNR is set at 8 */
AddGaussianNoise(input_image, noise_image, stddev_noise);

AddGaussianNoise(input_img, noise_img, sigma)   /*Add Gaussian Noise to the input
```

```
image */
int input_img[][N], noise_img[][N];
double sigma;
{
   int i,j,s;
   for(i=0;i<N;i++)
      for(j=0;j<N; j++)
      {
           s=input_img[i][j]+Gaussian(sigma);
           noise_img[i][j]= s>255 ? 255 : s<0 ? 0 : s;
      }
}
float Gaussian(sd)
float sd;
{
static int ready =0;
static float gstore;
float v1, v2, r, fac, gaus;
int r1, r2;

if(ready==0) {
   do {
         r1=rand();
         r2=rand();
         v1=2.*((float)r1/(float) RAND_MAX-0.5);
         v2=2.*((float)r2/(float) RAND_MAX-0.5);
         r=v1*v1+v2*v2;
       } while (r>1.0);
   fac=(float) sqrt((double) (-2*log(r)/r));
   gstore=v1*fac;
   gaus=v2*fac;
   ready=1;
}
else {
   ready=0;
   gaus=gstore;
```

```
}
return (gaus*sd);
}
```

(2) Method to compute an image power or variance

The signal variance can be represented as follows:

$$\sigma^2 = E[(X-\eta)^2] = E[X^2 - 2X\eta + \eta^2] = E[X^2] - 2\eta E[X] + \eta^2 = E[X^2] - \eta^2$$

where $E[X] = \eta.$

Thus, the image power for *MXN* image can be obtained by using the following equation:

$$\sigma^2 = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} X_{ij}^2 - \left(\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} X_{ij}\right)^2.$$

(3) Edge Detection Masks

1) As for the Roberts, Sobel, and Prewitt mask, please use the masks as shown in Fig. 3.16 on page 109 of the text book.

2) As for the 5x5 stochastic gradient mask, please use the mask shown below.

$$G_X = \begin{bmatrix} 0.267 & 0.364 & 0 & -0.364 & -0.267 \\ 0.373 & 0.562 & 0 & -0.562 & -0.373 \\ 0.463 & 1.000 & 0 & -1.000 & -0.463 \\ 0.373 & 0.562 & 0 & -0.562 & -0.373 \\ 0.267 & 0.364 & 0 & -0.364 & -0.267 \end{bmatrix}$$

$$G_Y \equiv G_X^T$$