Computer Vision HW8

National Taiwan University

r02944002 王瀚宇

Description

I am generating gaussian noise with amplitude of 10 and 30, salt-and-pepper noise with probability 0.1 and 0.05. I use the 3x3, 5x5 box filter and median filter, opening-then-closing and closing-then opening filter (using the octogonal 3-5-5-5-3 kernel, value = 0) on those images. I produced a total of 28 images (preprocessed and postprocessed).

Algorithm

Gaussian Noice :

$$I(nim,i,j) = I(im,i,j) + amplitude * N(0,1)$$

$N(0,1)$: Gaussian random variable with zero mean and st. dev. 1

amplitude determines signal-to-noise ratio, try 10, 30

Salt&Peper

- Generate salt-and-pepper noise

$$I(noiseImage,i,j) = \begin{cases} 0 & \text{, if } \ uniform(0,1) < 0.05 \\ 255 & \text{, if } \ uniform(0,1) > 1 - 0.05 \\ I(srcImage,i,j) & \text{, otherwise} \end{cases}$$

$uniform(0,1)$: random variable uniformly distributed over [0,1]

try both 0.05 and 0.1

principal code

```
public static ArrayList<Integer> GaussianNoise( ArrayList<Integer> origin,int
amplitude,int headerLength, int width, int height)
      {
            ArrayList<Integer> results =
InitWhite(origin,headerLength,width,height);
            for(int y = 0 ; y < height; y++)
            {
                  for(int x = 0 ; x < width ; x++)
                  {
                        int index = headerLength + y*width + x;
                        int pixel = origin.get(index) + (int)
(amplitude*random.nextGaussian());
                        if(pixel<0)pixel = 0;
                        if(pixel>255)pixel = 255;
                        results.set(index, pixel);
                  }
            }
            return results;
      }

public static ArrayList<Integer> SaltAndPeper(ArrayList<Integer>
origin,float threshold,int headerLength, int width, int height)
{
ArrayList<Integer> results =
InitWhite(origin,headerLength,width,height);
            for(int y = 0 ; y < height; y++)
            {
                  for(int x = 0 ; x < width ; x++)
                  {
                        int index = headerLength + y*width + x;
                        float randomNum = random.nextFloat();
                        int pixel = origin.get(index);
                        if(randomNum < threshold)
                        {pixel = 0;}
                        else if(randomNum>(1-threshold))
                        {pixel = 255;}
                        results.set(index,pixel);
                  }
            }
            return results;
      }
```

```
public static ArrayList<Integer> BoxFilter(ArrayList<Integer> origin,int 
boxWidth,int boxHeight,int headerLength, int width, int height)
	{
		ArrayList<Integer> results = 
InitWhite(origin,headerLength,width,height);
		
		for(int y = 0 ; y < height; y++)
		{
			for(int x = 0 ; x < width ; x++)
			{
				int centerX = (boxWidth - 1)/2;
				int centerY = (boxHeight - 1)/2;
				
				ArrayList<Integer> candidate = new 
ArrayList<Integer>();
				
				for(int boxY = 0; boxY < boxHeight ; boxY++)
				{
					for(int boxX = 0; boxX < boxWidth ; boxX++)
					{
						int globalX = x + boxX - centerX;
						int globalY = y + boxY - centerY;
						
						if(globalX<0)continue;
						if(globalX>=width)continue;
						if(globalY<0)continue;
						if(globalY>=height)continue;
						
						int pixel = origin.get(headerLength + 
globalY*width + globalX);
						
						candidate.add(pixel);
					}
				}
				
				results.set(headerLength + y * width + x, 
Average(candidate));
				
			}
		}
		
		return results;
	}
```

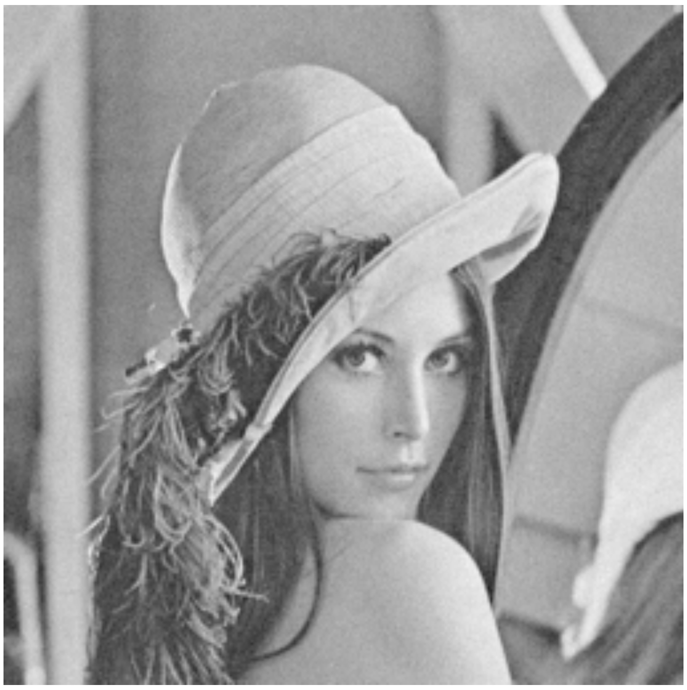
```
public static ArrayList<Integer> MedianFilter(ArrayList<Integer>
origin,int boxWidth,int boxHeight,int headerLength, int width, int
height)
	{
		ArrayList<Integer> results =
InitWhite(origin,headerLength,width,height);

		for(int y = 0 ; y < height; y++)
		{
			for(int x = 0 ; x < width ; x++)
			{
				int centerX = (boxWidth - 1)/2;
				int centerY = (boxHeight - 1)/2;

				ArrayList<Integer> candidate = new
ArrayList<Integer>();

				for(int boxY = 0; boxY < boxHeight ; boxY++)
				{
					for(int boxX = 0; boxX < boxWidth ; boxX++)
					{
						int globalX = x + boxX - centerX;
						int globalY = y + boxY - centerY;

						if(globalX<0)continue;
						if(globalX>=width)continue;
						if(globalY<0)continue;
						if(globalY>=height)continue;

						int pixel = origin.get(headerLength +
globalY*width + globalX);

						candidate.add(pixel);
					}
				}
				results.set(headerLength + y * width + x,
Median(candidate));
			}
		}
		return results;
	}
```

```
public static ArrayList<Integer> Opening_Closing(ArrayList<Integer>
origin,int headerLength, int width, int height,Kernel kernel)
     {
          ArrayList<Integer> erosion =
Erosion(origin,headerLength,width,height,kernel);
          ArrayList<Integer> dilation =
Dilation(erosion,headerLength,width,height,kernel);
          dilation =
Dilation(dilation,headerLength,width,height,kernel);
          erosion = Erosion(dilation,headerLength,width,height,kernel);

          return erosion;
     }

public static ArrayList<Integer> Closing_Opening(ArrayList<Integer>
origin,int headerLength, int width, int height,Kernel kernel)
     {
          ArrayList<Integer> dilation =
Dilation(origin,headerLength,width,height,kernel);
          ArrayList<Integer> erosion =
Erosion(dilation,headerLength,width,height,kernel);
          erosion = Erosion(erosion,headerLength,width,height,kernel);
          dilation =
Dilation(erosion,headerLength,width,height,kernel);

          return dilation;
     }
```

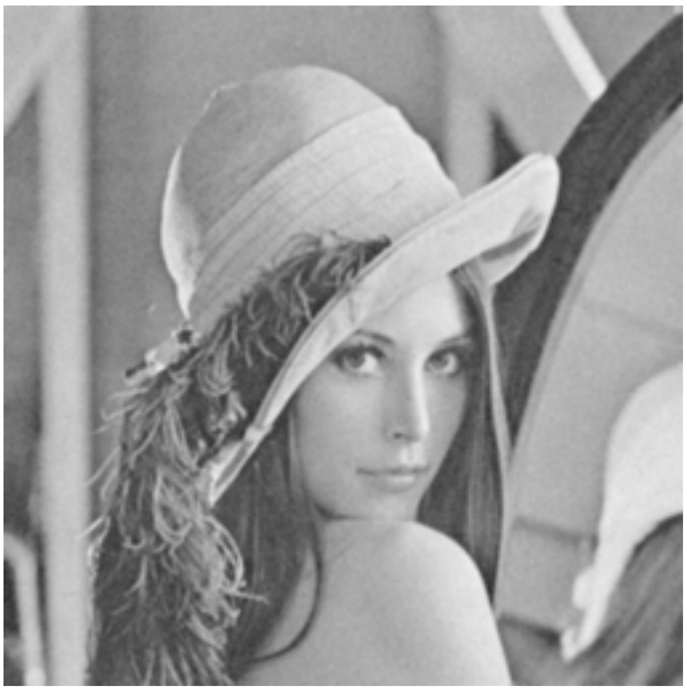
```
public static float CalculateSNR(ArrayList<Integer>
originImage,ArrayList<Integer> filterImage,int headerLength)
	{
		float result = 0;

		int total = originImage.size() - headerLength;

		// calculate us
		float us = 0;
		for(int i = headerLength ; i<originImage.size() ; i++)
		{
			us += originImage.get(i);
		}
		us /= total;

		//calculate vs
		float vs = 0;
		for(int i = headerLength ; i<originImage.size() ; i++)
		{
		vs += (originImage.get(i) - us)*(originImage.get(i) - us) ;
		}
		vs/=total;

		//calculate un
		float un = 0;
		for(int i = headerLength ; i<originImage.size() ; i++)
		{
			un += (filterImage.get(i) - originImage.get(i)) ;
		}
		un/=total;

		//caculate vn
		float vn =0;
		for(int i = headerLength ; i<originImage.size() ; i++)
		{
			float data = filterImage.get(i)-originImage.get(i)-un;
			vn += data*data;
		}
		vn /= total;

	result = (float) (20 * Math.log10(Math.sqrt(vs)/Math.sqrt(vn)));

		return result;
	}
```

Results

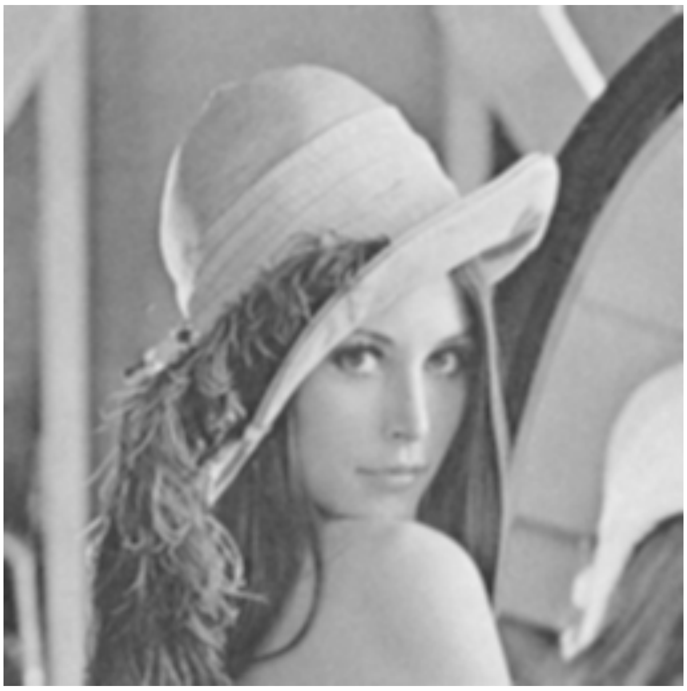
Gaussian Noise (Amplitude=10)



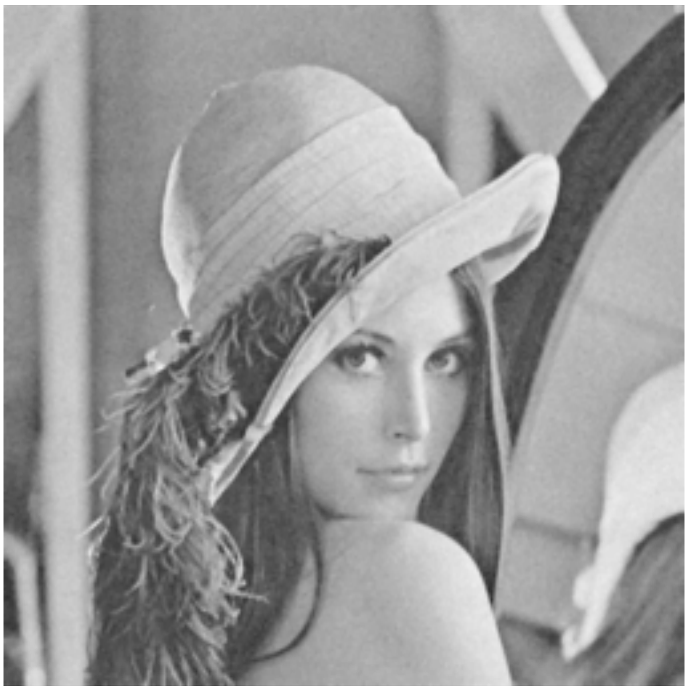
Gaussian Noise Image



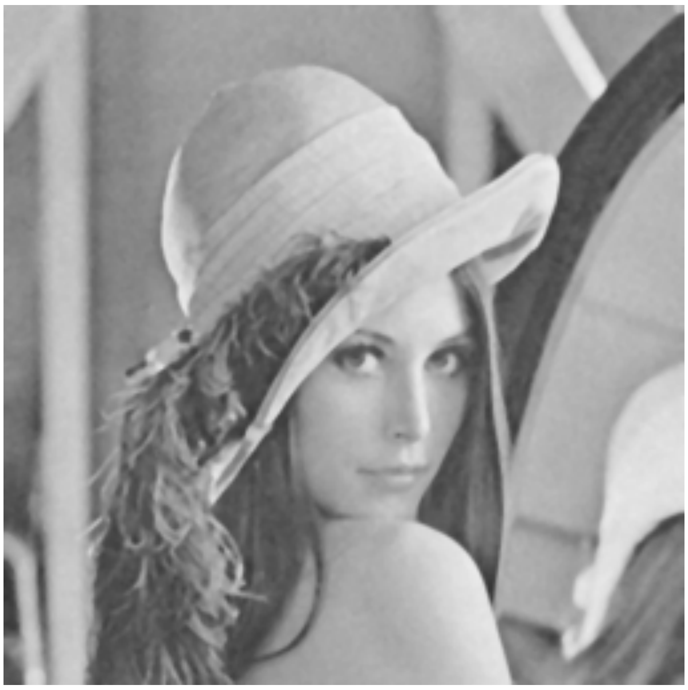
Box Filter 3x3 (SNR=17.83043)

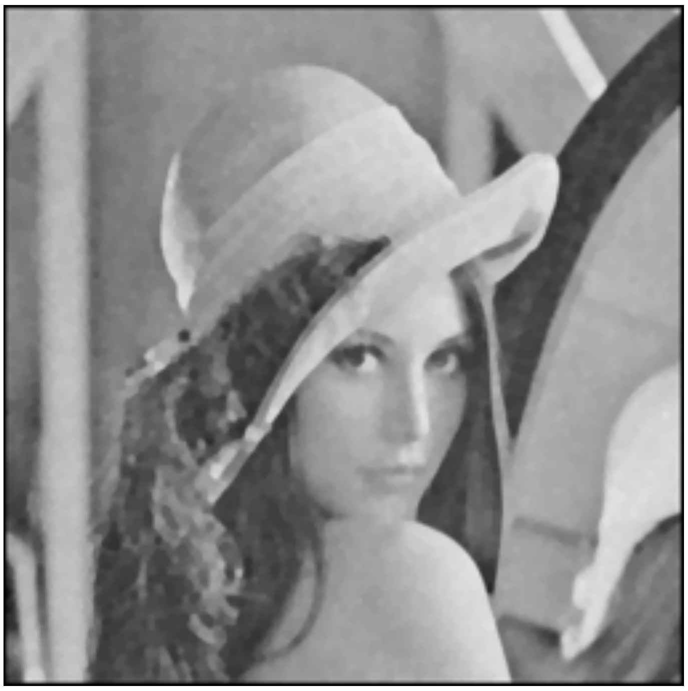


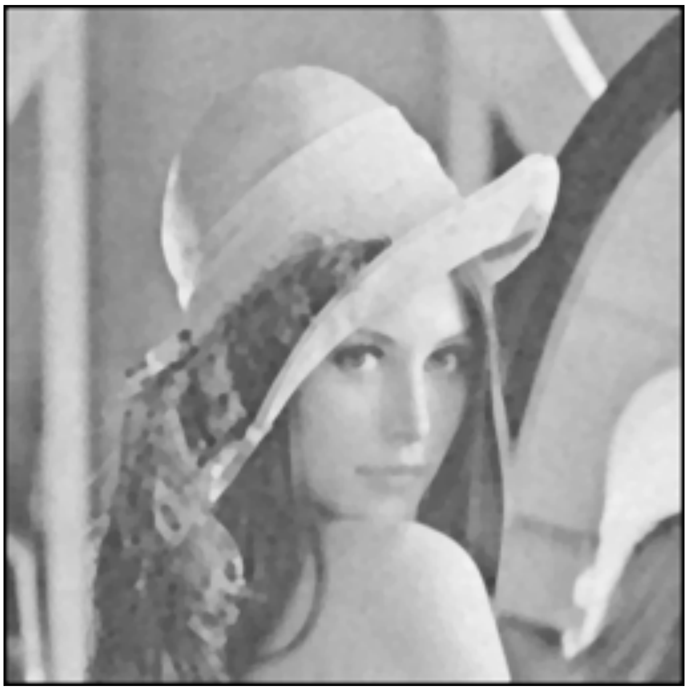
Box Filter 5x5 (SNR=14.884616)



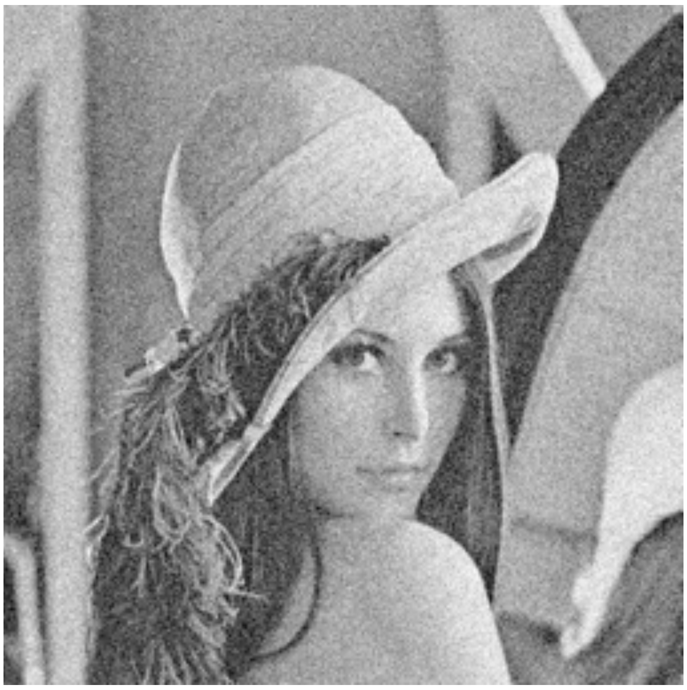
Median Filter 3x3 (SNR= 17.946619)

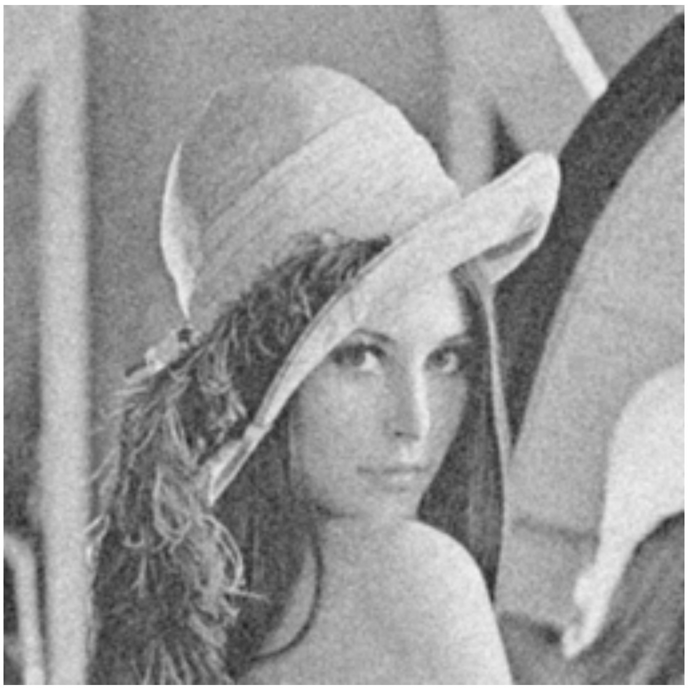

Median Filter 5x5 (SNR= 16.087215)

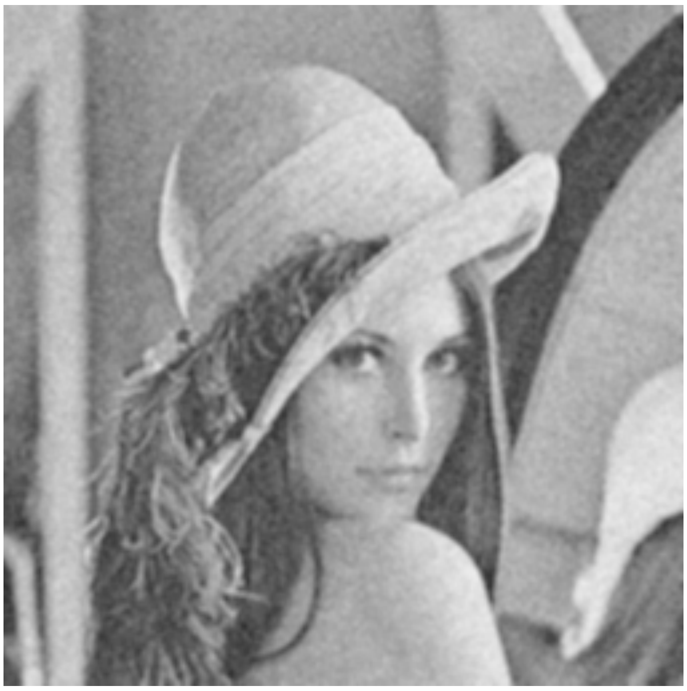

Opening then Closing (SNR= 8.581979)

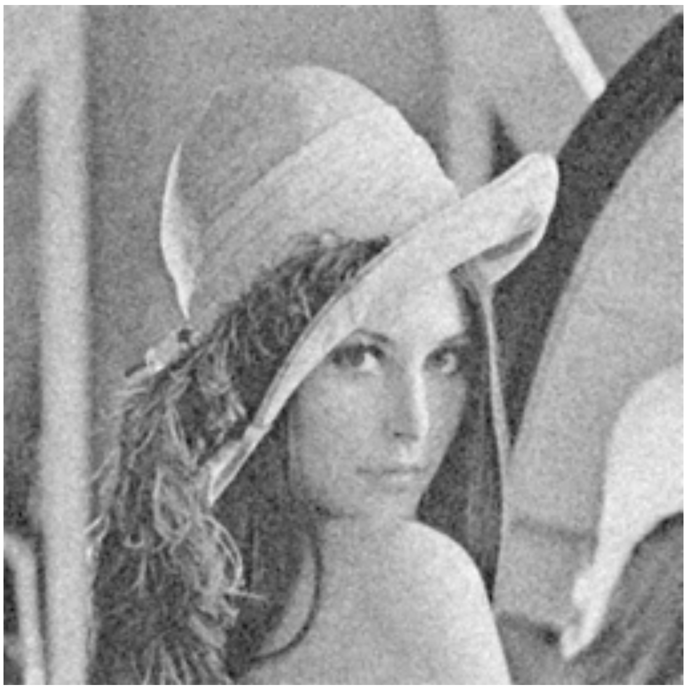

Closing the Opening (SNR= 7.6555357)

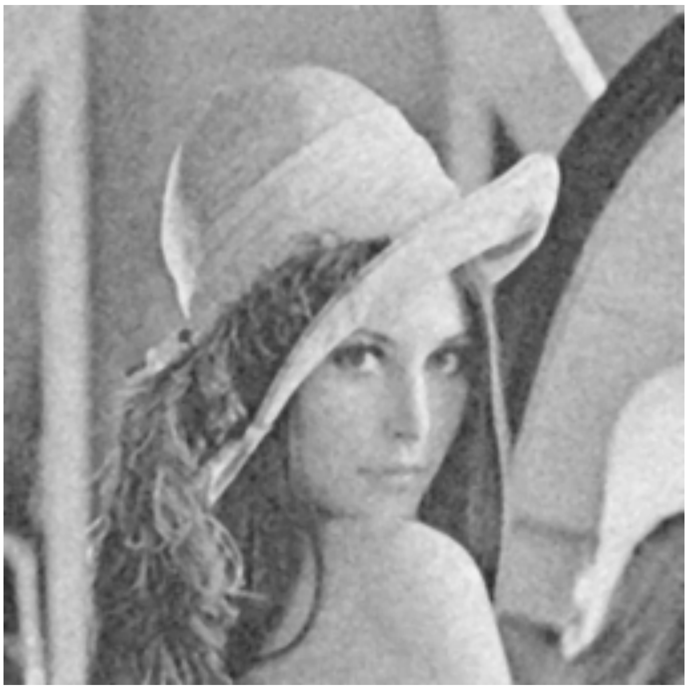
Gaussian Noise (Amplitude=30)


Gaussian Noise Image

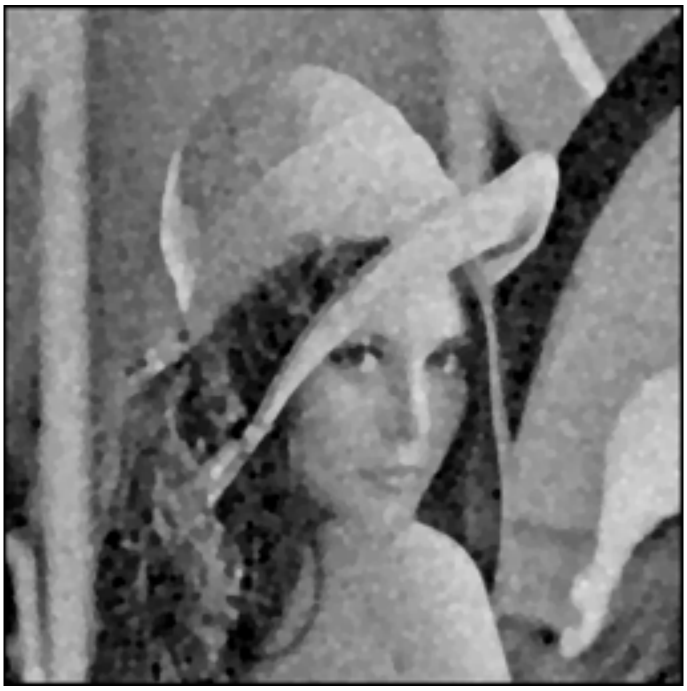

Box Filter 3x3 (SNR= 12.648438)


Box Filter 5x5 (SNR= 13.348516)

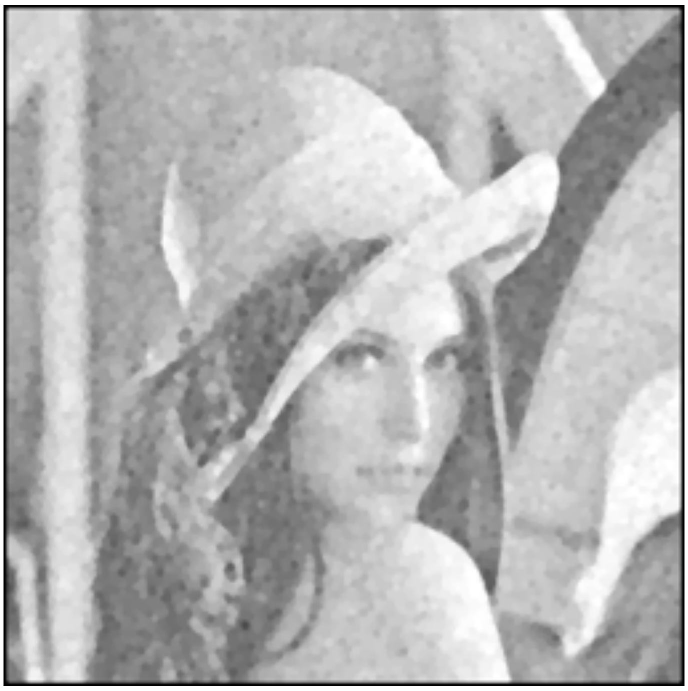

Median Filter 3x3 (SNR= 11.263369)



Median Filter 5x5 (SNR= 13.085466)

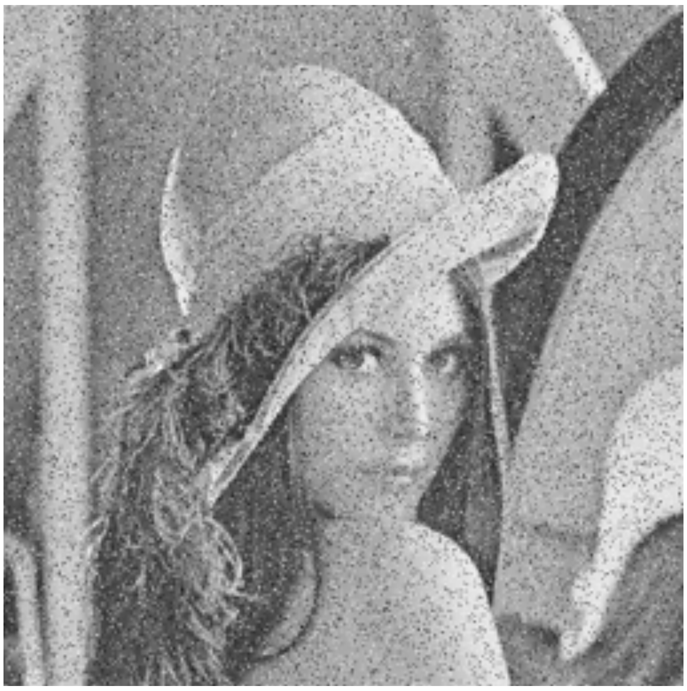


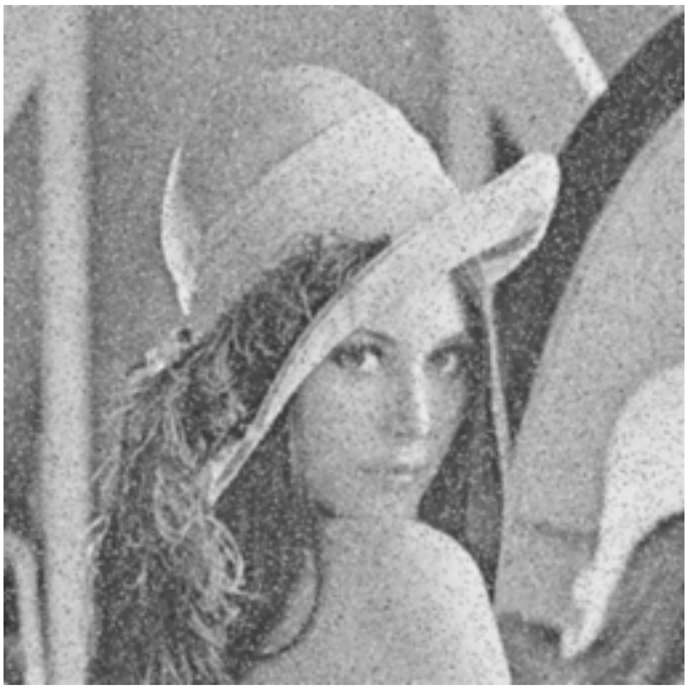
Opening then Closing (SNR= 8.579728)

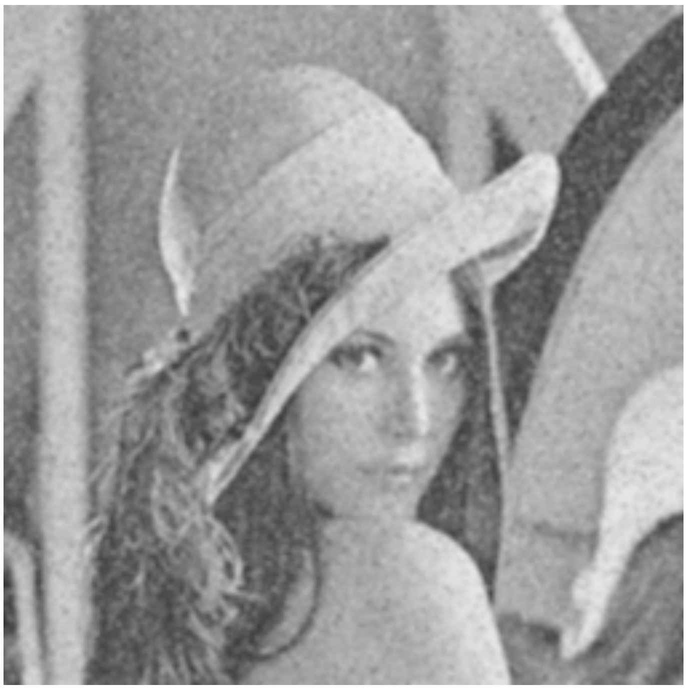


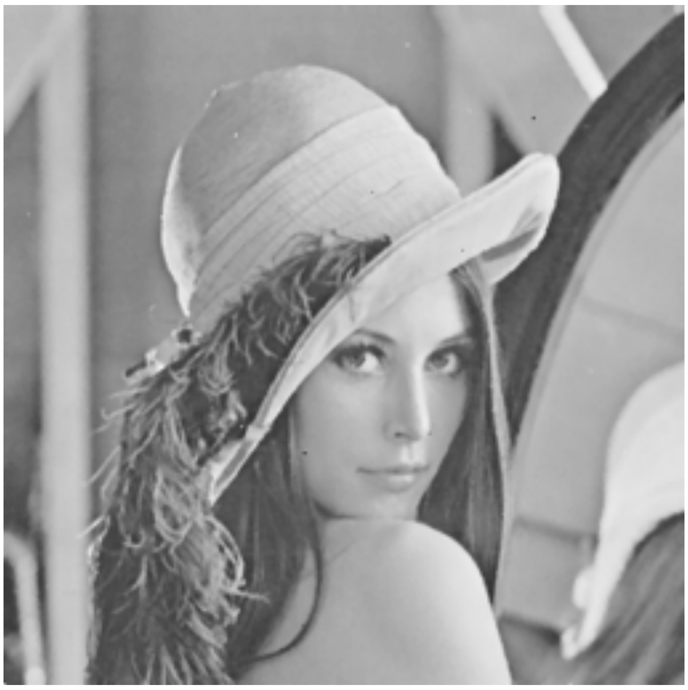
Closing the Opening (SNR= 6.0653872)

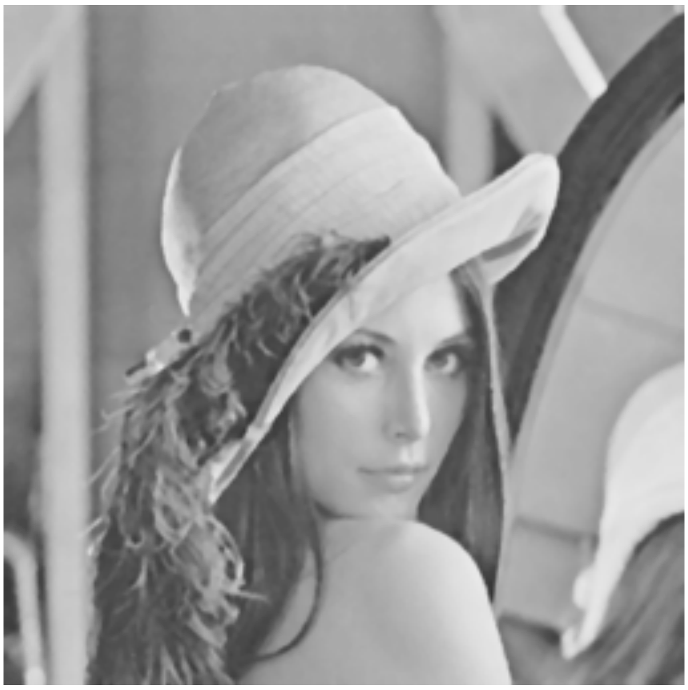
SaltAndPepper (threshold = 0.05)

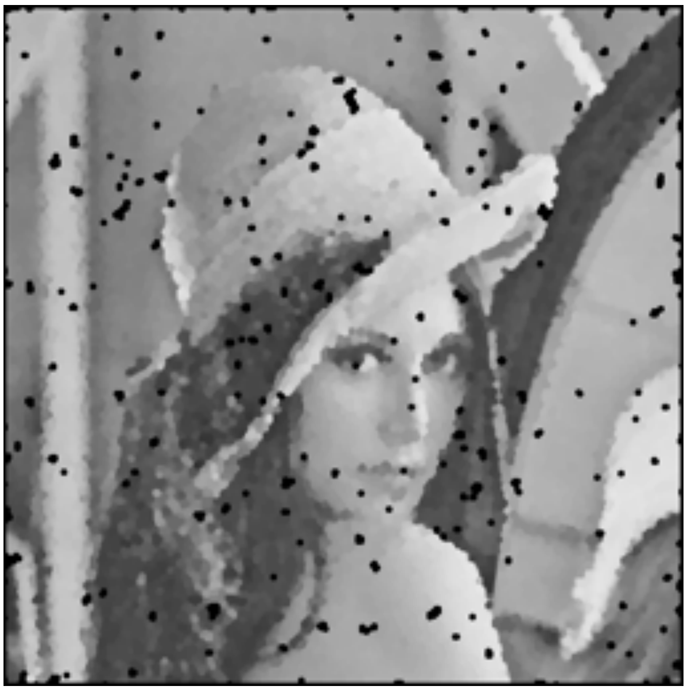

Salt And Peper Noise Image

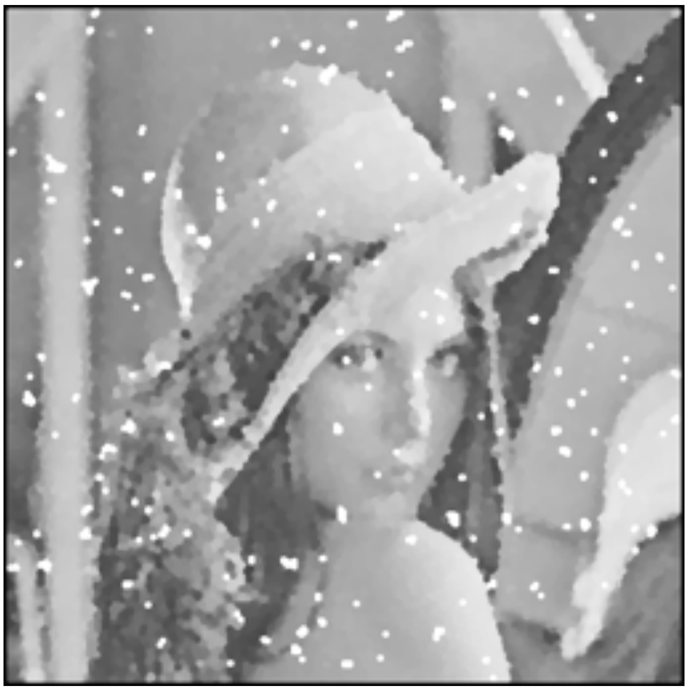

Box Filter 3x3 (SNR= 9.468221)


Box Filter 5x5 (SNR= 11.184831)

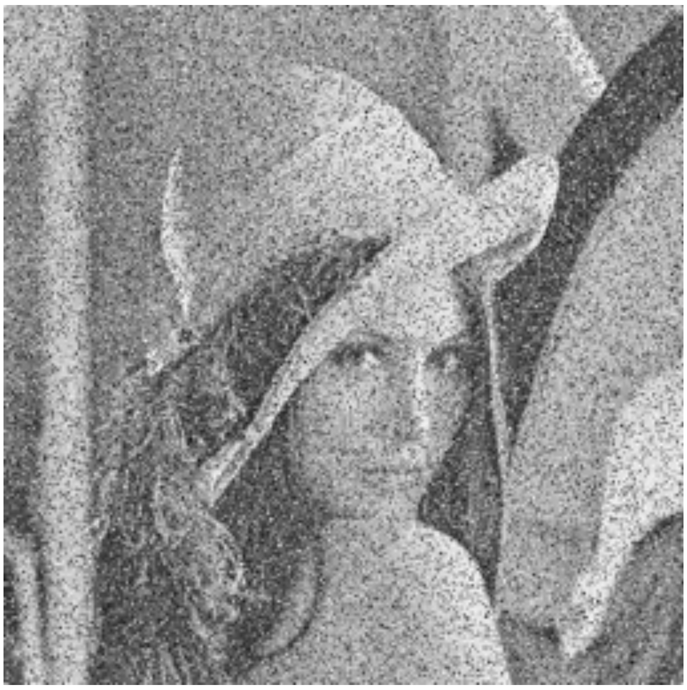

Median Filter 3x3 (SNR= 19.097776)

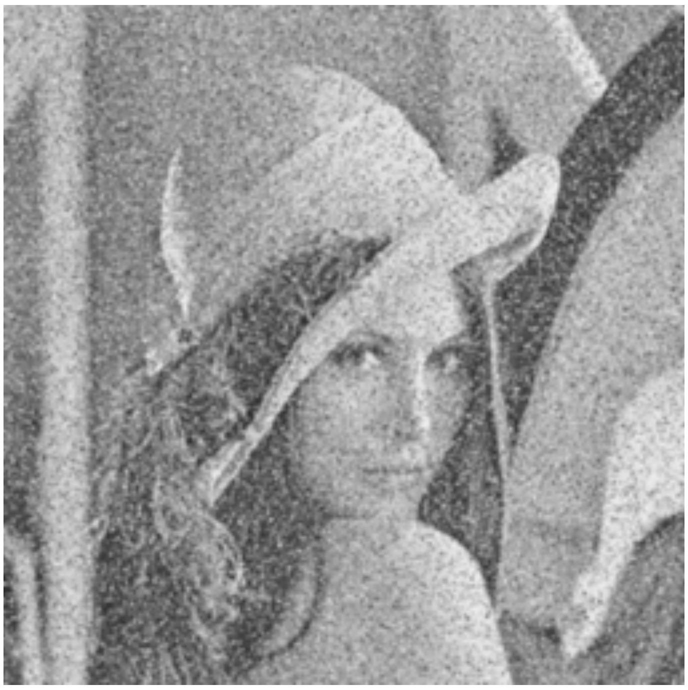

Median Filter 5x5 (SNR= 16.332739)

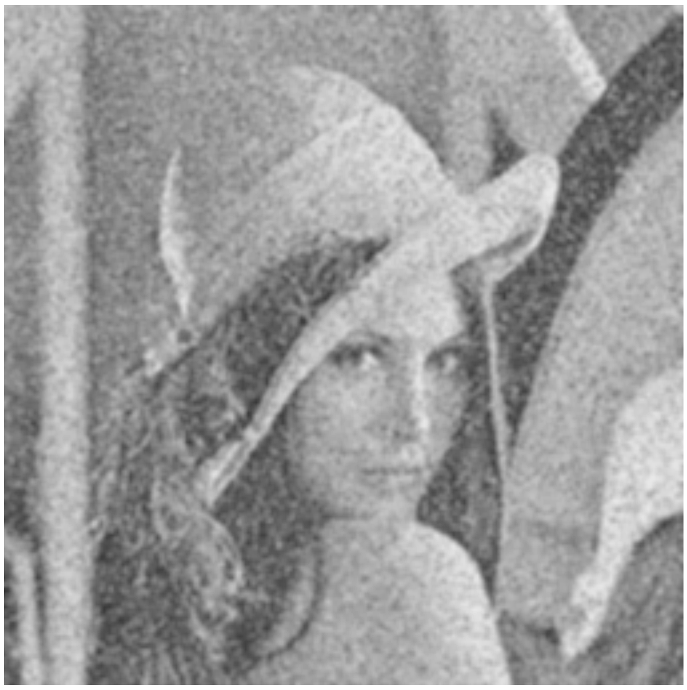

Opening then Closing(SNR= 4.3230243)

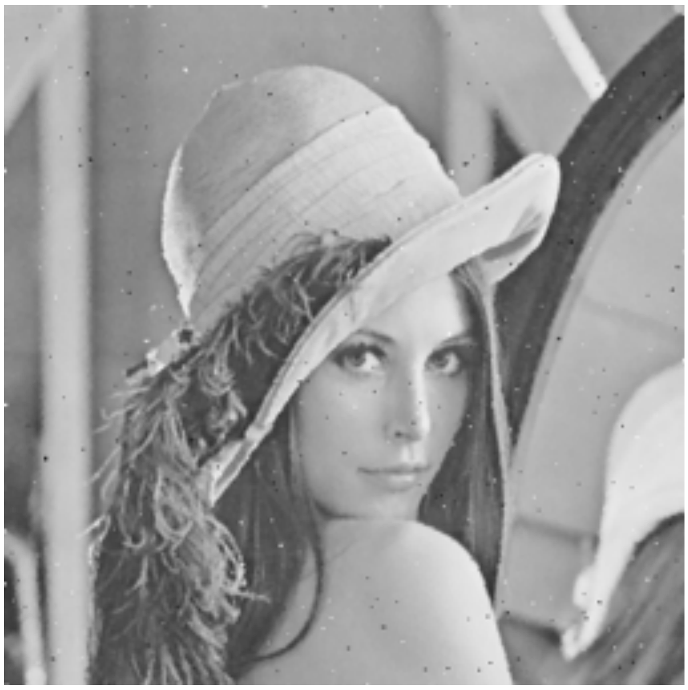

Closing the Opening (SNR= 3.8816488)

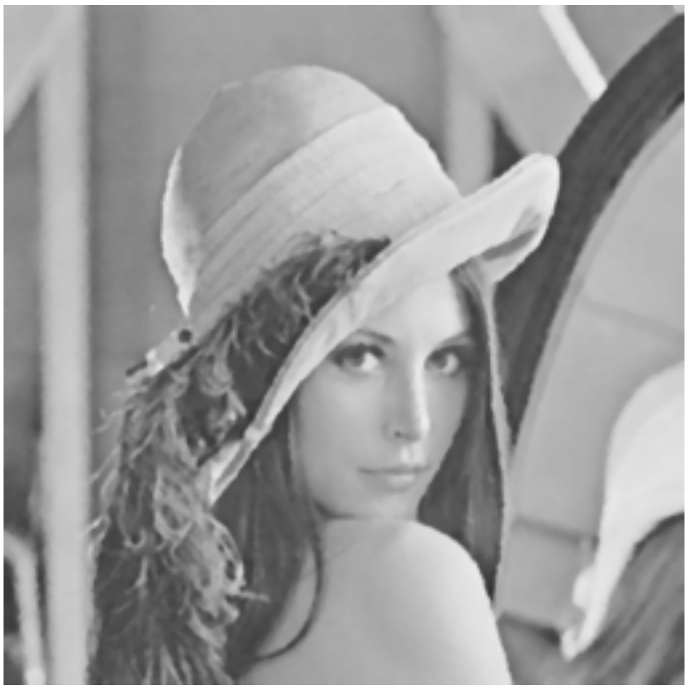
SaltAndPepper (Threshold =0.1)


Salt And Peper Noise Image

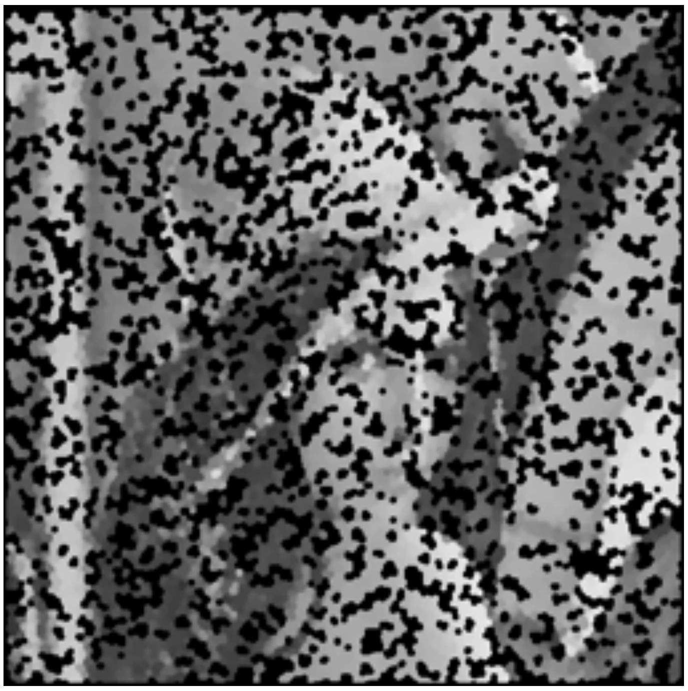

Box Filter 3x3 (SNR= 6.338612)


Box Filter 5x5 (SNR= 8.50791)

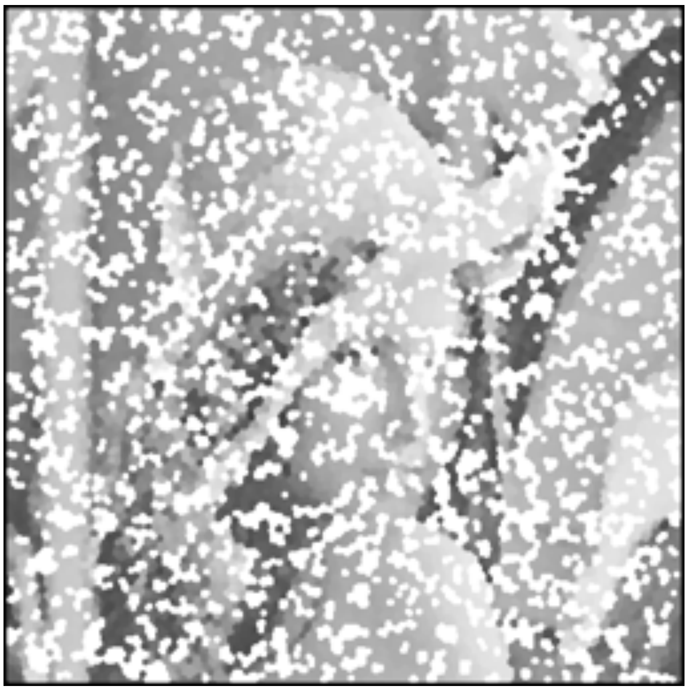

Median Filter 3x3 (SNR= 14.863498)



Median Filter 5x5 (SNR= 15.747424)



Opening then Closing(SNR= -2.188105)



Closing the Opening (SNR= -2.777967)

Discussion

目前看起來OpeningThenClosing 跟 ClosingThenOpening不論在Gaussian Noise或Pepper&Salt Noise表現都不佳。看起來不適合處理這兩種Noise。

而Gaussian Noise使用Box Filter及Median Filter效果都差不多。

而Pepper&Salt Noice的情況下Median Filter的效果較Box Filter佳。