資工所碩一 R07922164 林群崴

本次作業採用 **C++配合 OpenCV** 完成

1. Generate additive white Gaussian noise

```
Mat Gaussian_noise(Mat img, int amplitude) {

	Mat result(img.rows, img.cols, CV_8U, Scalar(0));

	// random device class instance, source of 'true' randomness for initializing random seed
	random_device rd;
	// Mersenne twister PRNG, initialized with seed from previous random device instance
	mt19937 gen(rd());
	// standard deviation affects the dispersion of generated values from the mean
	normal_distribution<float> d(0, 1);

	for (int i = 0; i < img.rows; i++) {
		for (int j = 0; j < img.cols; j++) {
			result.at<uchar>(i, j) = img.at<uchar>(i, j) + amplitude * d(gen);
		}
	}

	return result;
}
```

結果：

	
with *amplitude = 10* SNR: 22.4551	with *amplitude = 30* SNR: 10.7918

2. Generate salt-and-pepper noise

```
Mat salt_and_pepper_noise(Mat img, double threshold) {

    Mat result = img.clone();

    random_device rd;  //Will be used to obtain a seed for the random number engine
    mt19937 gen(rd()); //Standard mersenne_twister_engine seeded with rd()
    uniform_real_distribution<> dis(0, 1);

    for (int i = 0; i < img.rows; i++) {
        for (int j = 0; j < img.cols; j++) {
            double d = dis(gen);
            if (d < threshold)
                result.at<uchar>(i, j) = 0;
            else if (d>1- threshold)
                result.at<uchar>(i, j) = 255;
        }
    }

    return result;

}
```

結果：

	
with *threshold = 0.05* SNR: 9.54243	with *threshold = 0.1* SNR: 6.0206

3. Run box filter *(3X3, 5X5)* on all noisy images

```
Mat box_filter(Mat img, int f_size) {
     Mat result(img.rows - f_size + 1, img.cols - f_size + 1, CV_8U, Scalar(0));

     for(int i = 0; i < result.rows; i++) {
          for (int j = 0; j < result.cols; j++) {
               Rect rl(j, i, f_size, f_size);
               result.at<uchar>(i,j) = mean(img(rl))[0];
          }
     }
     return result;
}
```

結果：

Run box filter 3X3 on gauss10

Run box filter 3X3 on gauss30

Run box filter 3X3 on salt-and-pepper 0.05

Run box filter 3X3 on salt-and-pepper 0.1

	
Run box filter 5X5 on gauss10	Run box filter 5X5 on gauss30

	
Run box filter 5X5 on salt-and-pepper 0.05	Run box filter 5X5 on salt-and-pepper 0.1

4. Run median filter *(3X3, 5X5)* on all noisy images

算中位數：

```
double medianMat(Mat Input) {


	vector<int> v;
	for (int i = 0; i < Input.rows; i++) {
		for (int j = 0; j < Input.cols; j++) {
			v.push_back(Input.at<uchar>(i, j));
		}
	}


	nth_element(v.begin(), v.begin() + v.size() / 2, v.end());
```

```
    return v[v.size() / 2];
}
```

功能實現：

```
Mat median_filter(Mat img, int f_size) {
    Mat result(img.rows - f_size + 1, img.cols - f_size + 1, CV_8U, Scalar(0));
    for (int i = 0; i < result.rows; i++) {
        for (int j = 0; j < result.cols; j++) {
            Rect rl(j, i, f_size, f_size);
            result.at<uchar>(i, j) = medianMat(img(rl));
        }
    }
    return result;
}
```

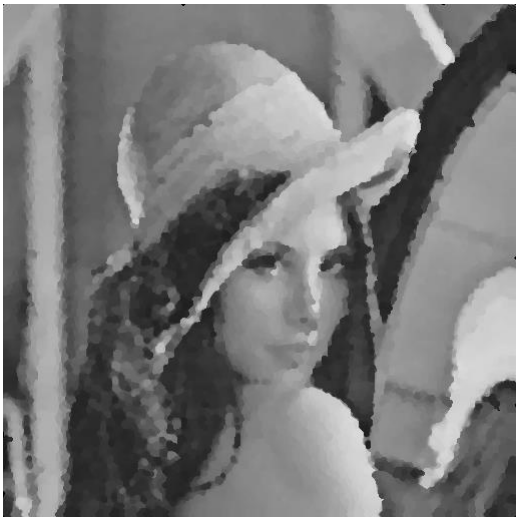
結果：

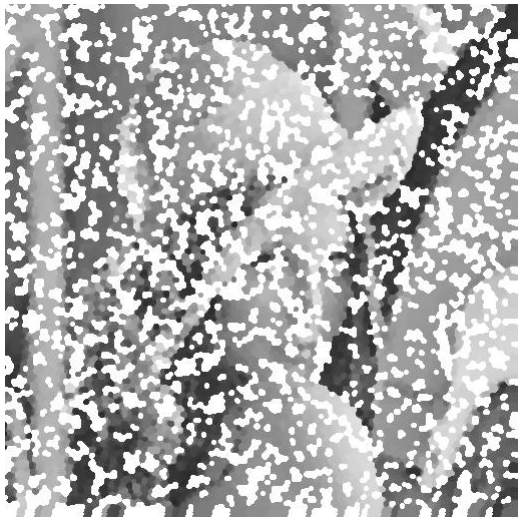
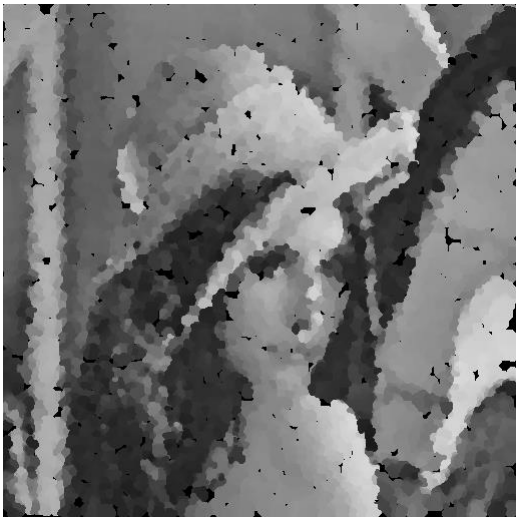
	
Run median filter *3X3* on gauss10	Run median filter *3X3* on gauss30

	
Run median filter *3X3* on salt-and-pepper 0.05	Run median filter *3X3* on salt-and-pepper 0.1
	
Run median filter 5X5 on gauss10	Run median filter 5X5 on gauss30
	
Run median filter 5X5 on salt-and-pepper 0.05	Run median filter 5X5 on salt-and-pepper 0.1

5. Run opening followed by closing and closing followed by opening

依照 HW5 所寫的功能，以標頭檔 HW5.h 匯入

	
gauss10 opening followed by closing	gauss10 closing followed by opening
	
gauss30 opening followed by closing	gauss30 closing followed by opening
	
salt-and-pepper 0.05 opening followed by closing	salt-and-pepper 0.05 closing followed by opening

	
salt-and-pepper 0.1 opening followed by closing	salt-and-pepper 0.1 closing followed by opening

檔案名稱說明：

依照所作的操作放在各自的資料夾中，

gauss10_close_open

代表 Generate additive white Gaussian noise with amplitude = 10

且進行 closing 後再進行 opening 操作