機器視覺

HW 4

資工三 109590049 林敬翰

1. Mean Filter

3X3 Code：

```
Mat MeanFilter3X3(Mat img) {
	int row = img.rows;
	int col = img.cols;
	int o_row = row + 2;
	int o_col = col + 2;
	Mat calculateMatrix(o_row, o_col, CV_8UC3, Scalar(0));
	//re-construct matrix
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			calculateMatrix.at<Vec3b>(i+1, j+1) = img.at<Vec3b>(i, j);
		}
	}
	//calculate median filter
	Mat output = Mat::zeros(row, col, CV_8UC3);
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			int avg1 = 0;
			int avg2 = 0;
			int avg3 = 0;
			for (int k = -1; k <= 1 ; k++) {
				for (int l = -1; l <= 1; l++) {
					Vec3b index = calculateMatrix.at<Vec3b>(i + 1 + k, j + 1 + l);
					avg1 += index[0];
					avg2 += index[1];
					avg3 += index[2];
				}
			}
			output.at<Vec3b>(i, j)[0] = avg1 / 9;
			output.at<Vec3b>(i, j)[1] = avg2 / 9;
			output.at<Vec3b>(i, j)[2] = avg3 / 9;
		}
	}
	return output;
}
```

7X7 Code：

```
Mat MeanFilter7X7(Mat img) {
	int row = img.rows;
	int col = img.cols;
	int o_row = row + 6;
	int o_col = col + 6;
	Mat calculateMatrix(o_row, o_col, CV_8UC3, Scalar(0));
	Mat x(row, col, CV_8UC1, Scalar(0));
	//re-construct matrix
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			calculateMatrix.at<Vec3b>(i + 3, j + 3) = img.at<Vec3b>(i, j);
		}
	}
	//calculate median filter
	Mat output = Mat::zeros(row, col, CV_8UC3);
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			int avg1 = 0;
			int avg2 = 0;
			int avg3 = 0;
			for (int k = -3; k <= 3; k++) {
				for (int l = -3; l <= 3; l++) {
					Vec3b index = calculateMatrix.at<Vec3b>(i + 3 + k, j + 3 + l);
					avg1 += index[0];
					avg2 += index[1];
					avg3 += index[2];
				}
			}
			output.at<Vec3b>(i, j)[0] = avg1 / 49;
			output.at<Vec3b>(i, j)[1] = avg2 / 49;
			output.at<Vec3b>(i, j)[2] = avg3 / 49;
		}
	}
	return output;
}
```

做 MeanFilter 我先創了一個比原本大一點的 mat 來去做 mask 的計算，3X3 的

長寬各加 2，7X7 的長寬各加 6，並把所有的空值都設成 0，之後把 mask 內的值總和後再除以 mask 的 pixel 數量，最後再把這些值丟到新的 mat 中。

2. Median Filter

3X3 Code：

```
Mat MedianFilter3X3(Mat img) {
	int row = img.rows;
	int col = img.cols;
	int o_row = row + 2;
	int o_col = col + 2;
	Mat calculateMatrix(o_row, o_col, CV_8UC1, Scalar(0));
	//re-construct matrix
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			calculateMatrix.at<uchar>(i + 1, j + 1) = img.at<uchar>(i, j);
		}
	}
	printf("1\n");
	//calculate median filter
	Mat output = Mat::zeros(row, col, CV_8UC1);
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			int counter = 0;
			std::vector<uchar> list(9);
			for (int k = -1; k <= 1; k++) {
				for (int l = -1; l <= 1; l++) {
					list.at(counter) = calculateMatrix.at<uchar>(i + 1 + k, j + 1 + l);
					counter++;
				}
			}
			std::sort(list.begin(), list.end());
			output.at<uchar>(i, j) = list.at(5);
		}
	}
	printf("2\n");
```

```
    return output;
}
```

7X7 Code：

```
Mat MedianFilter7X7(Mat img) {
    int row = img.rows;
    int col = img.cols;
    int o_row = row + 6;
    int o_col = col + 6;
    Mat calculateMatrix(o_row, o_col, CV_8UC1, Scalar(0));
    //re-construct matrix
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            calculateMatrix.at<uchar>(i + 3, j + 3) = img.at<uchar>(i, j);
        }
    }
    //calculate median filter
    Mat output = Mat::zeros(row, col, CV_8UC1);
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            int counter = 0;
            std::vector<uchar> list(49);
            for (int k = -3; k <= 3; k++) {
                for (int l = -3; l <= 3; l++) {
                    list.at(counter) = calculateMatrix.at<uchar>(i + 3 + k, j + 3 + l);
                    counter++;
                }
            }
            std::sort(list.begin(), list.end());
            output.at<uchar>(i, j) = list.at(25);
        }
    }
    return output;
}
```

MedianFilter 的方式跟做 MeanFilter 時類似，差別在我將鄰近 mask 的值儲存進

一個 vector 中，再做 sort 來找出中位數，再輸入進新的 mat 中。

3. Gaussian Filter

```
Mat GaussianFilter5X5(Mat img) {
	int row = img.rows;
	int col = img.cols;
	//create 5X5 gaussian filter
	double GaussianFilter[5][5];
	double sigma = sqrt(1/2);
	for (int i = -2; i <= 2; i++) {
		for (int j = -2; j <= 2; j++) {
			GaussianFilter[i+2][j+2] = exp(((-1.0) * (i * i + j * j)) / (2 * sigma * sigma)) / (2 * PI
* sigma * sigma);
		}
	}
	int o_row = img.rows + 4;
	int o_col = img.cols + 4;
	Mat calculateMatrix(o_row, o_col, CV_8UC3, Scalar(0));
	//re-construct matrix
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			calculateMatrix.at<Vec3b>(i + 2, j + 2) = img.at<Vec3b>(i, j);
		}
	}
	//Calculate Gaussian Filter
	Mat output(row, col, CV_8UC3, Scalar(0));
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < col; j++) {
			double avg1 = 0;
			double avg2 = 0;
			double avg3 = 0;
			double sum = 0;
			for (int k = -2; k <= 2; k++) {
				for (int l = -2; l <= 2; l++) {
					Vec3b index = calculateMatrix.at<Vec3b>(i + 2 + k, j + 2 + l);
					avg1 += index[0] * GaussianFilter[k + 2][l + 2];
					avg2 += index[1] * GaussianFilter[k + 2][l + 2];
					avg3 += index[2] * GaussianFilter[k + 2][l + 2];
					sum += GaussianFilter[k + 2][l + 2];
```

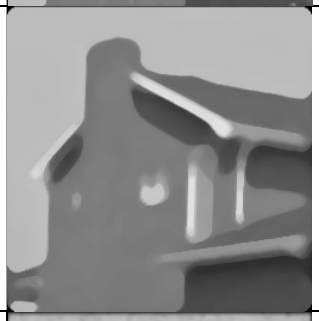
```
                    }
                }
                output.at<Vec3b>(i, j)[0] = avg1 / sum;
                output.at<Vec3b>(i, j)[1] = avg2 / sum;
                output.at<Vec3b>(i, j)[2] = avg3 / sum;
            }
        }
        return output;
}
```

我先用了一個 5X5 的 array，套入了 gaussian filter 的公式，我 sigma 使用的值是 sqrt(1/2)，來創出 gaussian filter 的 mask，一樣創出了一個新的長寬各加 4 的 mat 來做計算，並將 pixel 鄰近的點套入 mask 值並相加最後除以 mask 的總值來得出最後的值，最後套入新的 mat 中。

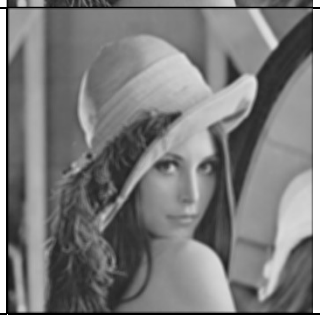
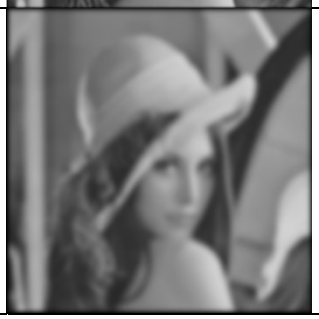
輸出

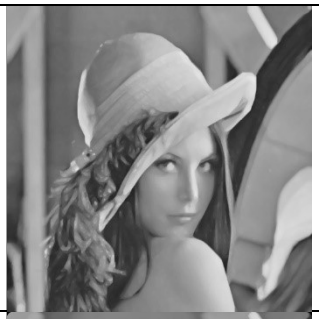
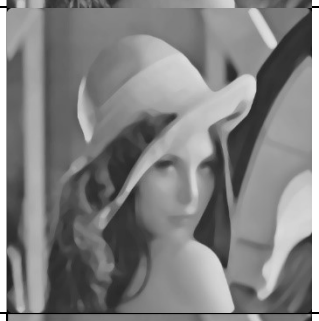
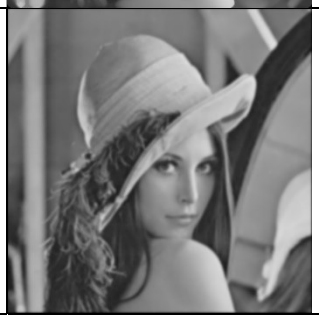
1. House256_noise

	1	7
Mean3X3		
Mean7X7		

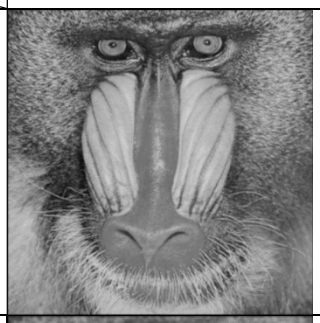
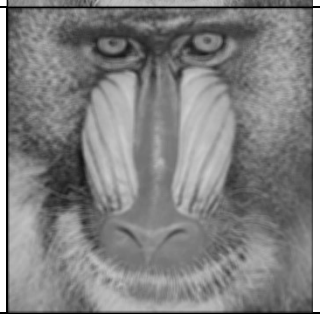
Median3X3		
Median7X7		
Gaussian5X5		

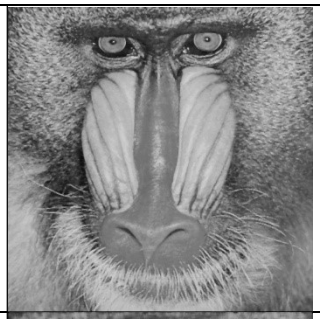
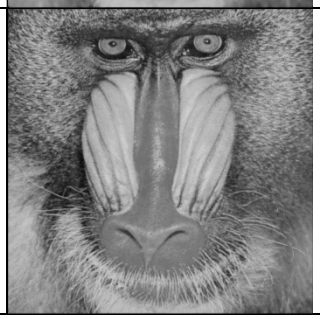
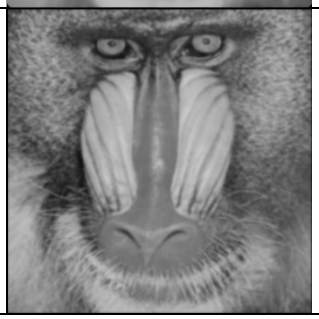
2. Lena_gray

	1	7
Mean3X3		
Mean7X7		

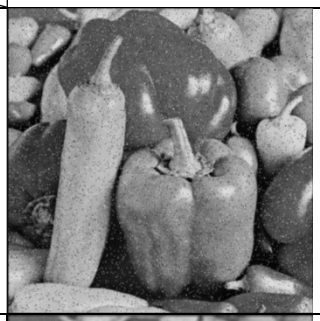
Median3X3		
Median7X7		
Gaussian5X5		

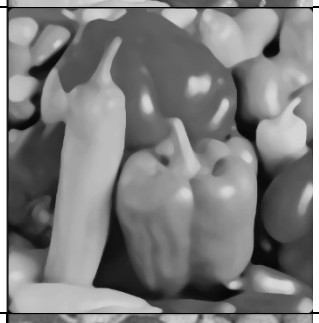
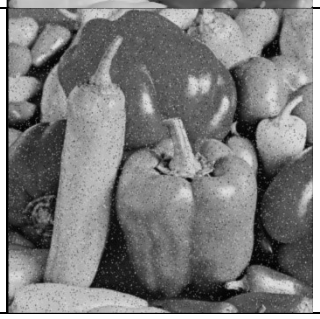
3.Mandrill_gray

	1	7
Mean3X3		
Mean7X7		

Median3X3		
Median7X7		
Gaussian5X5		

4.Peppers_noise

	1	7
Mean3X3		
Mean7X7		

Median3X3		
Median7X7		
Gaussian5X5		

我發現好像因為我在計算時用擴充 pixel 的方式，導致在做 mean 和 gaussian 的時候最外面一圈會特別黑。我發現 mean filter 是讓影像模糊化，7X7 的模糊效果比 3X3 的好。median filter 的畫面我覺得有點像其他程式的動畫濾鏡，而我覺得他的去雜訊效果是最好的。Gaussian Filter 我看來在模糊的同時保留了更好的細節，雖然在有雜訊的時候雜訊沒有被完全的濾掉。