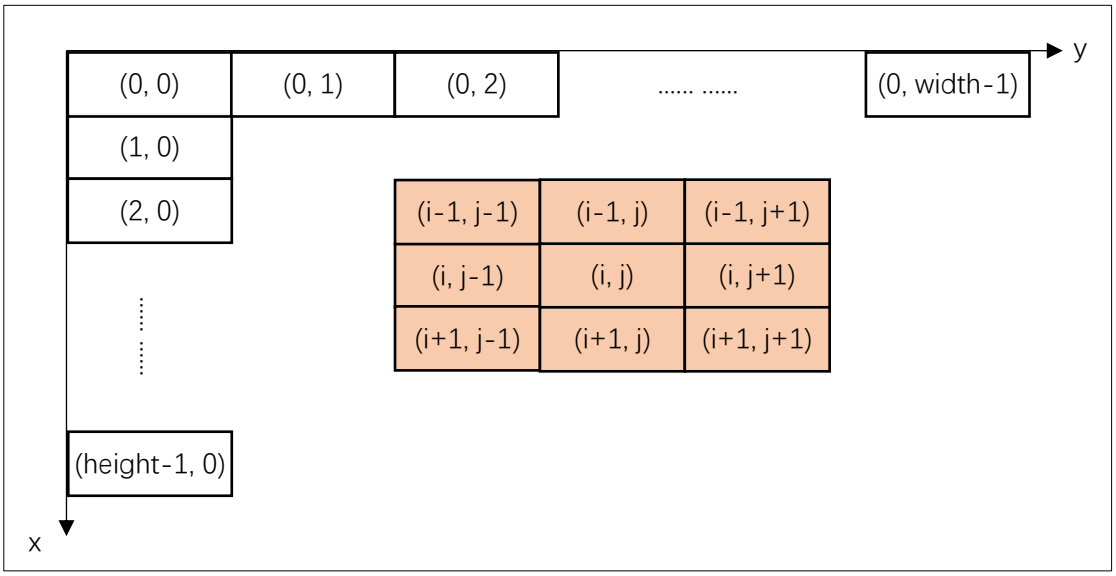
Class lab 1

Use two images for each operation to do the following operations and write down their advantages and disadvantages and explain your results:

1. Test the given pgmreader program use 3 x 3 filter:

- Average Filter

Algorithm:



$$outImage(i,j) = \sum_{k=-1}^{1}\sum_{t=-1}^{1} inImage(i+k, j+t)\ /9 \quad (1.1)$$

$$indexInData\ =\ i * width + j \quad (1.2)$$

$$outData[i * width + j] = \sum_{k=-1}^{1}\sum_{t=-1}^{1} inData[(i+k) * width + (j+t)]\ /9 \quad (1.3)$$

Results (including pictures):

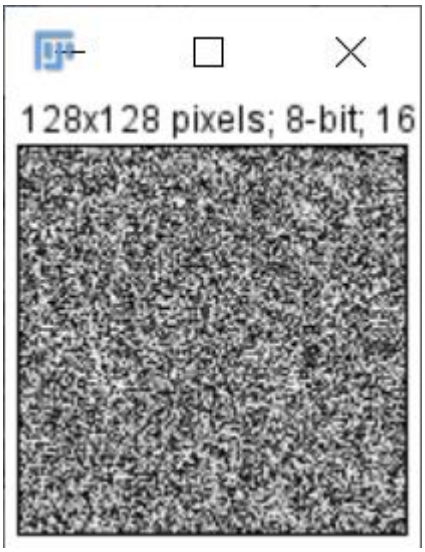
Result of processing "Lena.pgm":

Source Image:

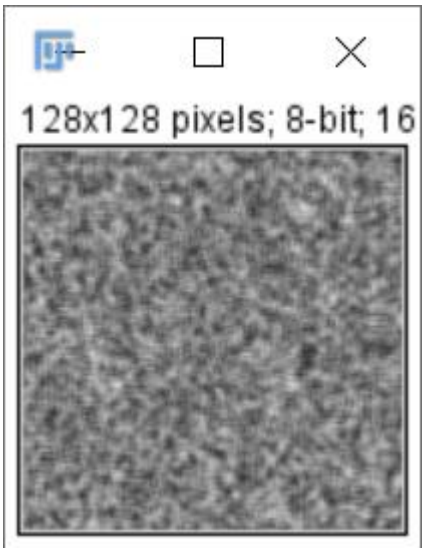
Result after average filter:

Result of processing “Noise.pgm”:

Source Image:



Result after average filter:



Discussion:

From equation (1.1), we can conclude that differences between neighbor pixels become smaller by averaging, which leads to the smoother of image.

The result images do turn out to be smoother. For example, the edges between objects are less obvious.

Codes:

```
int matrix[matrixWidth][matrixHeight];
memset(matrix, 0, sizeof(matrix));
for (int i = zero1 / 2; i < matrixWidth - zero1 / 2; i++) {
    for (int j = zero2 / 2; j < matrixHeight - zero1 / 2; j++) {
        matrix[i][j] = *tempin;
        tempin++;
    }
}

for (int i = zero1 / 2; i < matrixWidth - zero1 / 2; i++) {
    for (int j = zero2 / 2; j < matrixHeight - zero1 / 2; j++) {
        int res = 0;
        for (int m = 0; m < number1; m++) {
            for (int n = 0; n < number2; n++) {
                res += matrix[i - zero1 / 2 + m][j - zero2 / 2 + n];
            }
        }
        matrix[i][j] = res / (number1 * number2);
        *tempout = matrix[i][j];
        tempout++;
    }
}
```

- Median Filter

Algorithm:

Input: a unsigned char array that stores the source image data

Output: a unsigned char array that stores the output image data

For inImage(i, j):

Store its' 3 x 3 neighbor including itself in a unsigned char array **local**[9];

Find out the median of **local**;

Assign the median to outImage(i,j) => $outData[i * width + j] = findMedian(\boldsymbol{local}, 9)$

END

Results (including pictures):

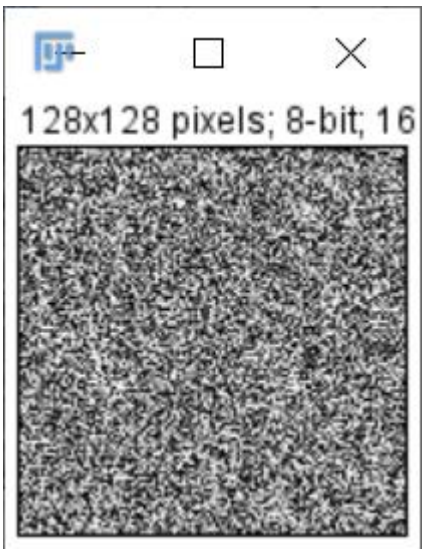
Result of processing "Lena.pgm":

Source Image:

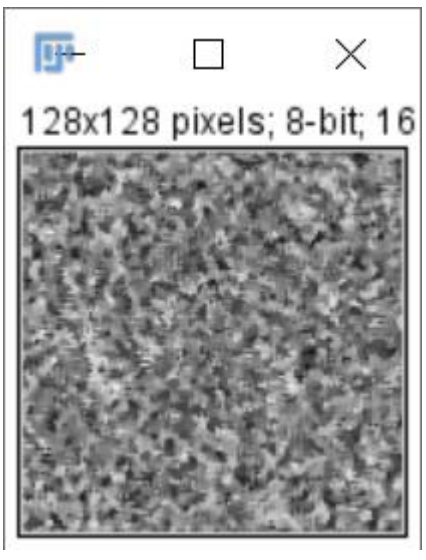
Result after median filter:

Result of processing "Noise.pgm":

Source Image:



Result after median filter:



Discussion:

The median filter also smooths the input image. But the result of median filter is sharper than that of average filter. Median filter simply substitutes the target pixel with the median of its 3 x 3 neighbor, that is, the operation does not operate all pixels in the neighbor. So the differences between pixels in the output image by median filter is larger than that by average filter.

Codes:

```
int matrix[matrixWidth][matrixHeight];
memset(matrix, 0, sizeof(matrix));
for (int i = zero1 / 2; i < matrixWidth - zero1 / 2; i++) {
    for (int j = zero2 / 2; j < matrixHeight - zero1 / 2; j++) {
        matrix[i][j] = *tempin;
        tempin++;
    }
}
```

```
for (int i = zero1 / 2; i < matrixWidth - zero1 / 2; i++) {
    for (int j = zero2 / 2; j < matrixHeight - zero1 / 2; j++) {
        int arry[number1 * number2], k = 0;
        for (int m = 0; m < number1; m++) {
            for (int n = 0; n < number2; n++, k++) {
                arry[k] = matrix[i - zero1 / 2 + m][j - zero2 / 2 + n];
            }
        }
        qsort(arry, number1 * number2, sizeof(int), cmp);
        matrix[i][j] = arry[number1 * number2 / 2];
        *tempout = matrix[i][j];
        tempout++;
    }
}
```