Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2017

Homework 1 – Image Median Filtering

Due: 22/02/2017, Wednesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism and homework trading will not be tolerated!

Introduction

Image filtering is a useful tool for image processing in computers. It is generally used for noise removal in the digital images. A noisy image and the filtered image are given in Figure 1. The main idea behind the image filtering is to update a pixel's value in accordance with the pixels in its neighborhood.

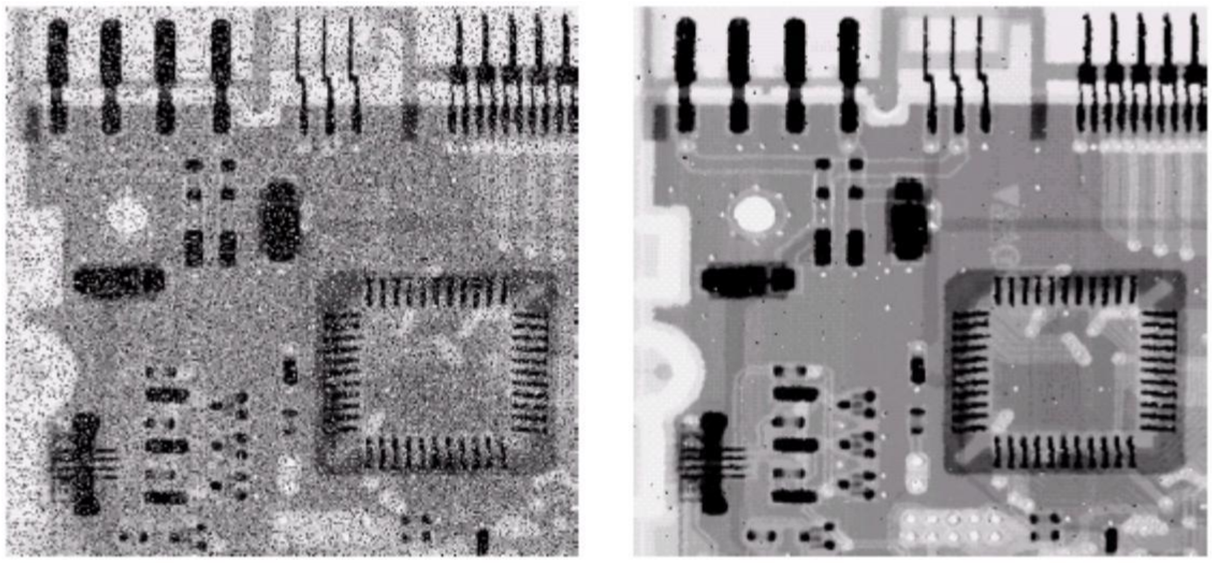


Figure 1: (left) Noisy image, (right) Image after noise removal

The image is a $m \times n$ matrix, where each cell of the matrix is called a *pixel*. For this homework, we will work on a simple matrix representation of a grayscale image, which will have pixels containing integer values between 0 and 255.

Your task is to implement the image filtering operation. First, your program is going to ask the *filename* that holds the image as a matrix of integers. Then, the user will be prompted to enter the *filter size*. After providing the inputs, your program should perform a *median filter operation* with the given *filter size* on the input matrix. The details are given in the following sections.

Inputs, Output and Program Flow

First, your program should prompt for the *filter size* and read it. *filter size* must be an **odd positive integer** number. If it is not entered accordingly, the input entry should continue until a proper value is entered.

Then, the program prompts for the data file name. Then, it reads the file name from the keyboard until the correct file name is entered.

A sample image matrix with 6 rows and 5 columns is as follows. Of course, your program should work with matrices with arbitrary number of rows and columns.

```
10 20 13 12 40
5 200 120 255 123
18 21 19 17 20
16 13 160 200 255
0 14 16 19 58
100 150 175 18 23
```

Figure 1. A Sample 6×5 Matrix

You need to use a multidimensional vector (a vector of vectors) in order to save the matrix in the memory. Also, you have to make sure that the matrix cells only contain values between 0 and 255, all the rows should be of equal size, and etc. In such cases, your program should prompt an appropriate error message and quit immediately.

After appropriate inputs are entered, your program should perform a median filtering operation with the given *filter size* on the input matrix. Finally, your program should output the matrix after filtering operation.

In the next section, details of the median filtering are explained.

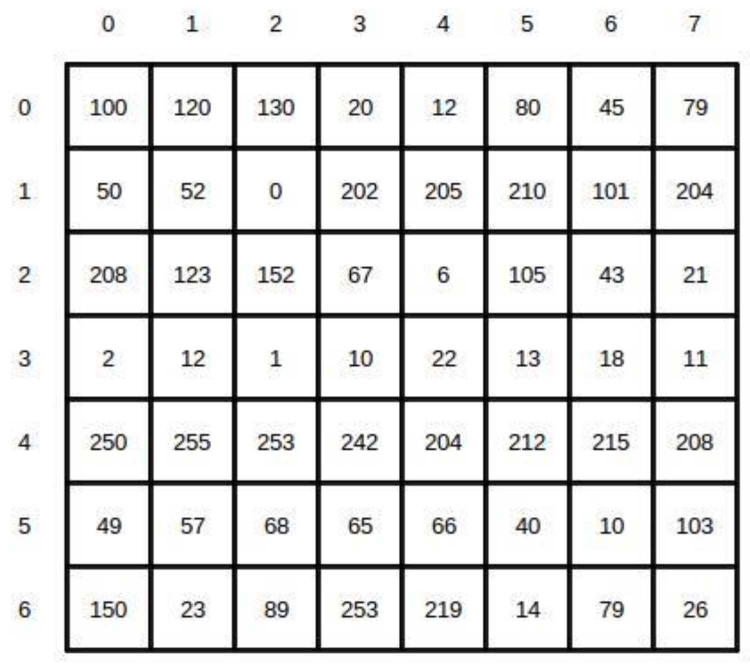
Median filtering

Median filtering is the process of updating each pixel of a given image with the median value of the pixels within a neighborhood of that pixel. Neighborhood is determined by the parameter *filter size.*

Let us assume that our *filter size* value is k, where k is an odd number as we mentioned above. In the filtering process, each pixel at position (x, y) is replaced with the median value of the elements of a $k \times k$ submatrix, of which the central element is located at (x, y)

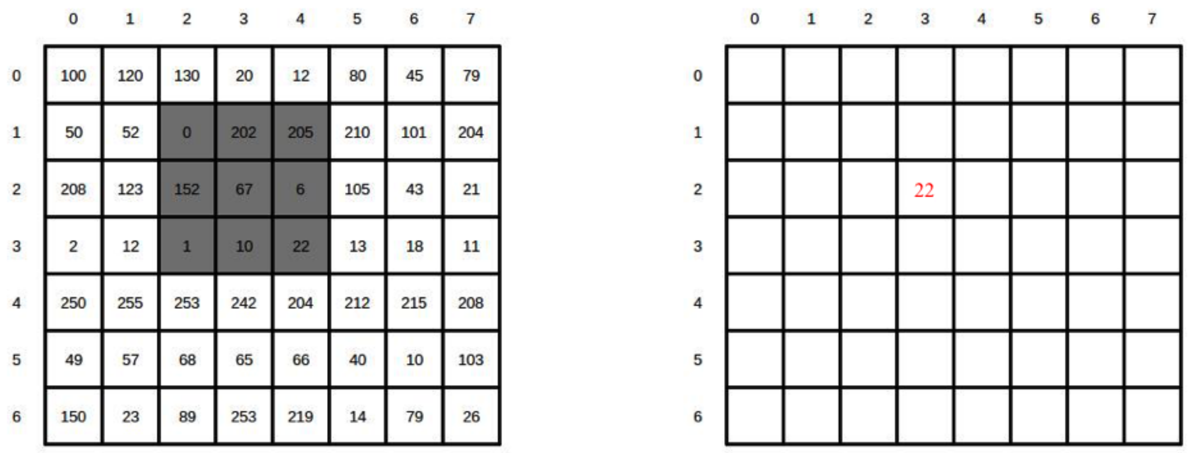
position. For some pixels close to matrix boundary, some of submatrix elements may go out of the actual image matrix. In such cases, the elements out of boundary must be ignored and only the cells that fall inside the boundary of the matrix should be included in the filtering.

We are going to detail median filtering over some examples. Let us assume that we are given a 7×8 image as shown in Figure 2 and *the filter size* is 3. Figure 3 shows the process of filtering applied to the pixel at (2,3). The submatrix in this case becomes a 3×3 matrix with central element at (2,3). This is shown as the gray area in the left side of Figure 3. The cells in this neighborhood contains values {0, 1, 6, 10, 22, 67, 152, 202, 205} when sorted from smallest to largest. The median value in this set is 22, since it is the middle element of the sorted list. This median value should be written into the central cell, the cell at location (2,3). Please note that the pixel at row 2 and column 3 in the new image takes the value of 22.



	0	1	2	3	4	5	6	7
0	100	120	130	20	12	80	45	79
1	50	52	0	202	205	210	101	204
2	208	123	152	67	6	105	43	21
3	2	12	1	10	22	13	18	11
4	250	255	253	242	204	212	215	208
5	49	57	68	65	66	40	10	103
6	150	23	89	253	219	14	79	26

Figure 2: A sample 7 × 8 image

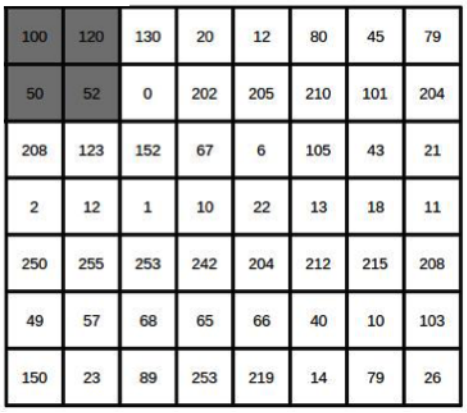


	0	1	2	3	4	5	6	7
0	100	120	130	20	12	80	45	79
1	50	52	0	202	205	210	101	204
2	208	123	152	67	6	105	43	21
3	2	12	1	10	22	13	18	11
4	250	255	253	242	204	212	215	208
5	49	57	68	65	66	40	10	103
6	150	23	89	253	219	14	79	26

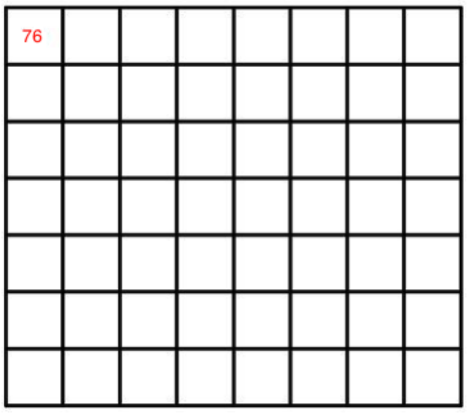
	0	1	2	3	4	5	6	7
0								
1								
2				22				
3								
4								
5								
6								

Figure 3: 3 × 3 filter (represented by gray field) is put on the image pixel at (2, 3). The median value of the gray area is written to row 2 and column 3 pixel of the new image (right).

A special case occurs in filtering at the border (or close to border if filter size is greater than 3) pixels of the image. This case is exemplified in Figure 4. In this example, we want to apply filtering to the pixel that lies at row 0 and column 0. In this case, we use a filter submatrix with the central element at (0,0), which is shown as the gray area in Figure 4 (left side). Some elements of the filter submatrix go beyond the image matrix boundaries. These elements that are out of the image matrix are ignored while finding the median value; as shown in the left part of Figure 4. The neighborhood cells contain values {50, 52, 100, 120} when sorted from smallest to greatest. Since there are even number of values, the median value should be the average of the two values that are closest to middle of the list. For this example, we find the median to be 76, the average of values 52 and 100. Finally, we write the value to the corresponding cell in the new image as illustrated in the right part of Figure 4. <u>If the filtered value you found is not an integer, you need to round it down.</u>



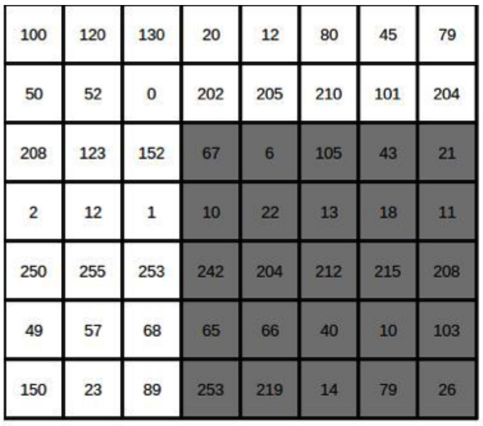
100	120	130	20	12	80	45	79
50	52	0	202	205	210	101	204
208	123	152	67	6	105	43	21
2	12	1	10	22	13	18	11
250	255	253	242	204	212	215	208
49	57	68	65	66	40	10	103
150	23	89	253	219	14	79	26



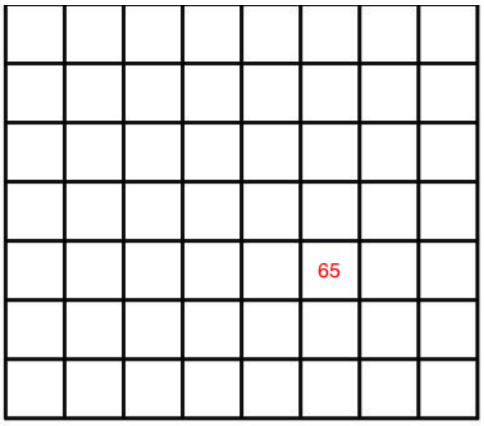
76							

Figure 4: 3 × 3 filter is applied to pixel at row 0 and column 0, which is the corner element.

The user can provide any odd number for *filter size*. Figure 5 shows the case where 5 × 5 filter is applied to the pixel at row 4 and column 5. Similar to the previous examples, we take the median value in the gray area and write this value to the corresponding location in the new image.



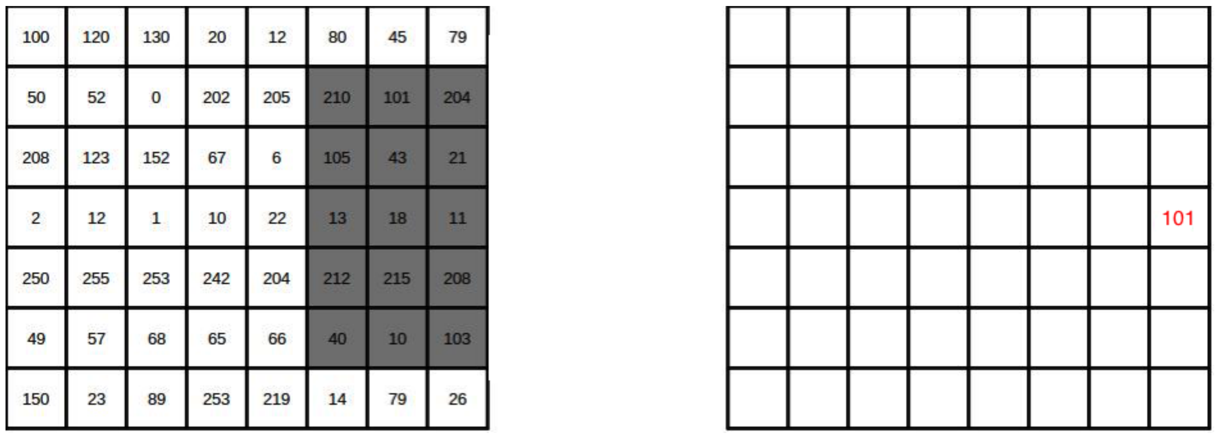
100	120	130	20	12	80	45	79
50	52	0	202	205	210	101	204
208	123	152	67	6	105	43	21
2	12	1	10	22	13	18	11
250	255	253	242	204	212	215	208
49	57	68	65	66	40	10	103
150	23	89	253	219	14	79	26



					65		

Figure 5: 5 × 5 filter is applied to pixel at row 4 and column 5.

Figure 6 shows another application of 5 × 5 filter, which is for pixel at (3, 7). In this example, some elements of the filter submatrix are not in the image matrix and therefore ignored.



100	120	130	20	12	80	45	79
50	52	0	202	205	210	101	204
208	123	152	67	6	105	43	21
2	12	1	10	22	13	18	11
250	255	253	242	204	212	215	208
49	57	68	65	66	40	10	103
150	23	89	253	219	14	79	26

							101

Figure 6: 5 × 5 filter is applied to pixel at row 3 and column 7.

CAUTION!

1. **Please note that for an entire median filtering process, you must apply the filter to each pixel of the given image and fill each pixel in the new image accordingly. In the examples above, we only applied averaging filter to particular pixels for the sake of brevity.**
2. **In the examples, filters of size 3 and 5 are used. Your program should work for any positive odd number filter size (1, 3, 5, 7, 9, 11, ...).**

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark.

The sample input files are provided in the .zip package of this homework.

Sample Run 1

File: input1.txt

```
3 255 10 8
0 43 12 1
7 12 0 10
12 4 6 9
5 6 200 17
```

Output:

```
This program performs median filtering operation
on an input matrix given in a text file
```

```
Enter the filter size (must be a positive odd integer)
-3

Error. Enter the filter size (must be a positive odd integer)
2

Error. Enter the filter size (must be a positive odd integer)
3

Enter the name of the file
input.txt

Error: Could not open the file input.txt

Enter the name of the file
input1.txt


FILTERED MATRIX

  23   11   11    9
   9   10   10    9
   9    7    9    7
   6    6    9    9
   5    6    7   13
```

Sample Run 2
File: input2.txt

```
4 43       12    1 4
12 0 7 10   14
12 4       6 9 200
```

Output:

```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
3

Enter the name of the file
input2.txt


FILTERED MATRIX

   8    9    8    8    7
   8    7    7    9    9
   8    6    6    9   12
```

Sample Run 3
File: input3.txt

```
100 120 130 20 12 80 45 79
50 52 0 202 205 210 101 204
208 123 152 67 6 105 43 21
2 12 1 10 22 13 18 11
250 255 253 242 204 212 215 208
49 57 68 65 66 40 10 103
150 23 89 253 219 14 79 26
```

Output:
```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
5

Enter the name of the file
input3.txt


FILTERED MATRIX

 120  110  100  105   80   79   79   80
  76   59   51   59   44   44   44   62
 120  110  100  105   80   79   79   80
  57   66   66   67   67   67   83  101
  89   78   68   67   67   65   41   40
  62   66   67   65   67   65   53   33
  89  119  150   89   89  103   91   79
```

Sample Run 4
File: input4.txt

```
31 10 8 120
10 43 12 1
7 12 0 10
4 6 9 250
55 63 20 17
```

Output:
```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
1

Enter the name of the file
input4.txt
```

```
FILTERED MATRIX

  31   10    8  120
  10   43   12    1
   7   12    0   10
   4    6    9  250
  55   63   20   17
```

Sample Run 5
File: input5.txt

```
3 a 10 8
0 43 12 1
7 12 0 10
12 4 6 9
5 6 200 17
```

Output:

```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
3

Enter the name of the file
input5.txt
Error: Input file contains invalid characters
Program Exiting...
```

Sample run 6
File: input6.txt

```
3 250 10 8 14 21
0 43 12 1
7 12 0 10
12 4 6 9
5 6 200 17
```

Output:

```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
5

Enter the name of the file
input6.txt
Input file is not in matrix format
Program Exiting...
```

Sample run 7
File: input7.txt

```
3 256 10 8
-1 43 12 1
7 12 0 10
12 4 6 9
5 6 200 17
```

Output:

```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
5

Enter the name of the file
input7.txt
Pixel with value: 256 is not in range [0, 255]
Program Exiting...
```

Sample run 8
File: input8.txt

```

3 255 10 8
0 43 12 1

7 12 0 10

12 4 6 9
5 6 200 17
```

Output:

```
This program performs median filtering operation
on an input matrix given in a text file

Enter the filter size (must be a positive odd integer)
5

Enter the name of the file
input8.txt
Input file is not in matrix format
Program Exiting...
```

Some Important Rules
In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

When we grade your homework we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

What and where to submit (PLEASE READ, IMPORTANT)
You should prepare (or at least test) your program using MS Visual Studio 2012 C++. We will use the standard C++ compiler and libraries of the abovementioned platform while testing your homework. It'd be a good idea to write your name and last name in the program (as a comment line of course).

Submissions guidelines are below. Some parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade. Name your solution, project, cpp file that contains your main program using the following convention (the necessary file extensions such as .sln, .cpp, etc, are to be added to it):

"SUCourseUserName_YourLastname_YourName_HWnumber"

Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw1

In some homework assignments, you may need to have more than one .cpp or .h files to submit. In this case add informative phrases after the hw number. However, do not add any other character or phrase to the file names.

Now let us explain which files will be included in the submitted package. Visual Studio 2012 will create two *debug* folders, one for the solution and the other one for the project. You should delete these two *debug* folders. Moreover, if you have run your program in release mode, Visual Studio may create *release* folders; you should delete these as well. Apart from these, Visual Studio 2012 creates a file extension of *.sdf*; you will also delete this file. The remaining content of your solution folder is to be submitted after compression. Compress your solution and project folders using WINZIP or WINRAR programs. Please use "zip" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension. Check that your compressed file opens up correctly and it contains all of the solution, project and source code files that belong to the latest version of your homework. Especially double-check that the zip file contains your cpp and (if any) header files that you wrote for the homework.

Moreover, we strongly recommend you to check whether your zip file will open up and run correctly. To do so, unzip your zip file to another location. Then, open your solution by clicking the file that has a file extension of .sln. Clean, build and run the solution; if there is no problem, you could submit your zip file. Please note that the deleted files/folders may be regenerated after you build and run your program; this is normal, but do not include them in the submitted zip file.

You will receive no credits if your compressed zip file does not expand or it does not contain the correct files. The naming convention of the zip file is the same. The name of the zip file should be as follows:

SUCourseUserName_YourLastname_YourName_HWnumber.zip

For example zubzipler_Zipleroglu_Zubeyir_hw1.zip is a valid name, but

Hw1_hoz_HasanOz.zip, HasanOzHoz.zip

are **NOT** valid names.

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Albert Levi, Ömer Mert Candan