**Assignment 2 report**

**Median Filters**

Dewi Kharismawati

14231619/dek8v5

# Implementations

1. Median filtering pseudocode

Read image

For each pixel

Crop based on the window\_size aoi

Sort(aoi)

Median = take the middle value of sorted aoi

Save median to result\_image

Evaluate with RSME

Plot all plots, histogram, and image results

Executing main.m, will execute all window size from 3x3 to 25x25. It will show original, denoised image, and their diff of window size 3,15,25. It records all time elapsed and RMSE for each window size, then plot in a figure.

The denoised image will lost floor(window\_size/2) border.

1. Median Filtering with histogram

Median filtering in 1 is claimed to be slow because it uses sort algorithm, which requires O(w n log n). Therefore, instead of sorting we want to use histogram to get the median of each iteration, which only requires O(w) . This approach is claimed to be a lot faster compare to method no 1.

Read image

For each pixel

Crop based on the window\_size aoi

Histogram aoi for bin counts and grey-level values

Get the cumulative of bin counts

Find the index of the first cumulative beans with >= middle of window\_size

Extract the grey-level from that index to get median

Save median to result\_image

Evaluate with RSME

Plot all plots, histogram, and image results

Executing main\_med\_filt.m, will execute all window size from 3x3 to 25x25. It will show original, denoised image, and their diff of window size 3,15,25. It records all time elapsed and RMSE for each window size, then plot in a figure.

The denoised image will lost floor(window\_size/2) border.

1. Adaptive Median Filter

Median filter is claimed to tackle the problem that method no 1 and 2 have, they can remove noise, but the drawback is that we lose edges and details in the process. Meanwhile, adaptive median filter can adjust the size of window filter based on certain conditions. When the median value of the pixel is impulse, window size needs to be increased.

Pseucode:

Read image

define window size max and min

For each pixel

Get Zmin minimum gray level in the window kernel

Get Zmax maximum gray level in the window kernel

Get Zmedian median gray level in the window kernel %using sort instead of histogram

% Level A

While window size <= window max

A1 = Zmedian – Zmin

A2 = Zmedian – Zmax

If A1>0 and A2<0

% Level B

B1 = current pixel - Zmin

B2 = current pixel – Zmax

If B1>0 and B2<0

Output = current pixel

Else

Output = Zmed

Else

Increase window size

If window size > window max

Output = current pixel

Evaluate with RSME

Plot all plots, histogram, and image results

# Results

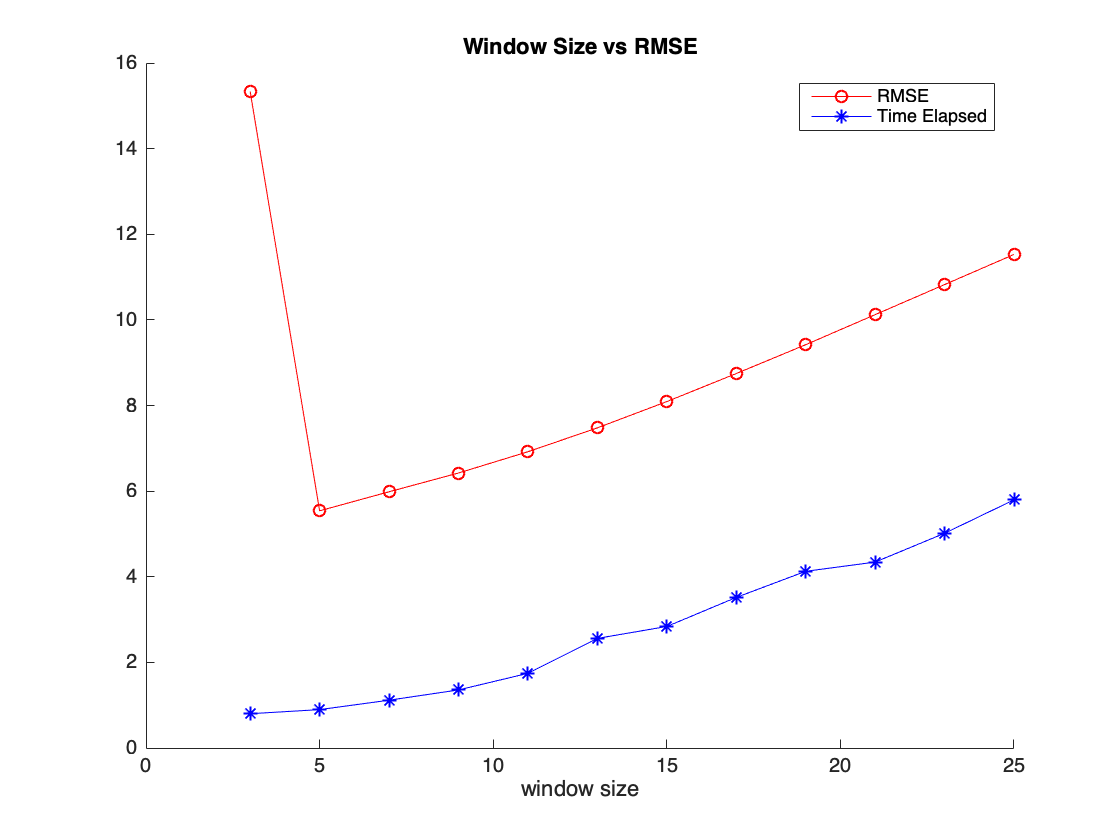
1. Regular Median Filter: Sample of Pears 30% noise results

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Window size | Time | RMSE |
|  | 3x3 | 0.804303 | 15.5024 |
| A picture containing screenshot  Description automatically generated | 15x15 | 2.560739 | 8.0910 |
|  | 25x25 | 6.278398 | 11.5459 |



Pears 60% noise

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Window size | Time | RMSE |
|  | 3x3 | 0.800223 | 15.3345 |
|  | 15x15 | 2.837979 | 8.0914 |
|  | 25x25 | 5.796041 | 11.5282 |



ABQ 60%

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Window size | Time | RMSE |
| A screenshot of a cell phone  Description automatically generated | 3x3 | 71.498193 | 61.9841 |
| A close up of a piece of paper  Description automatically generated  A screenshot of a social media post  Description automatically generated | 15x15 | 208.242502 | 14.5673 |
| A close up of a logo  Description automatically generated | 25x25 | 440.335031 | 18.2859 |

A close up of a map

Description automatically generated

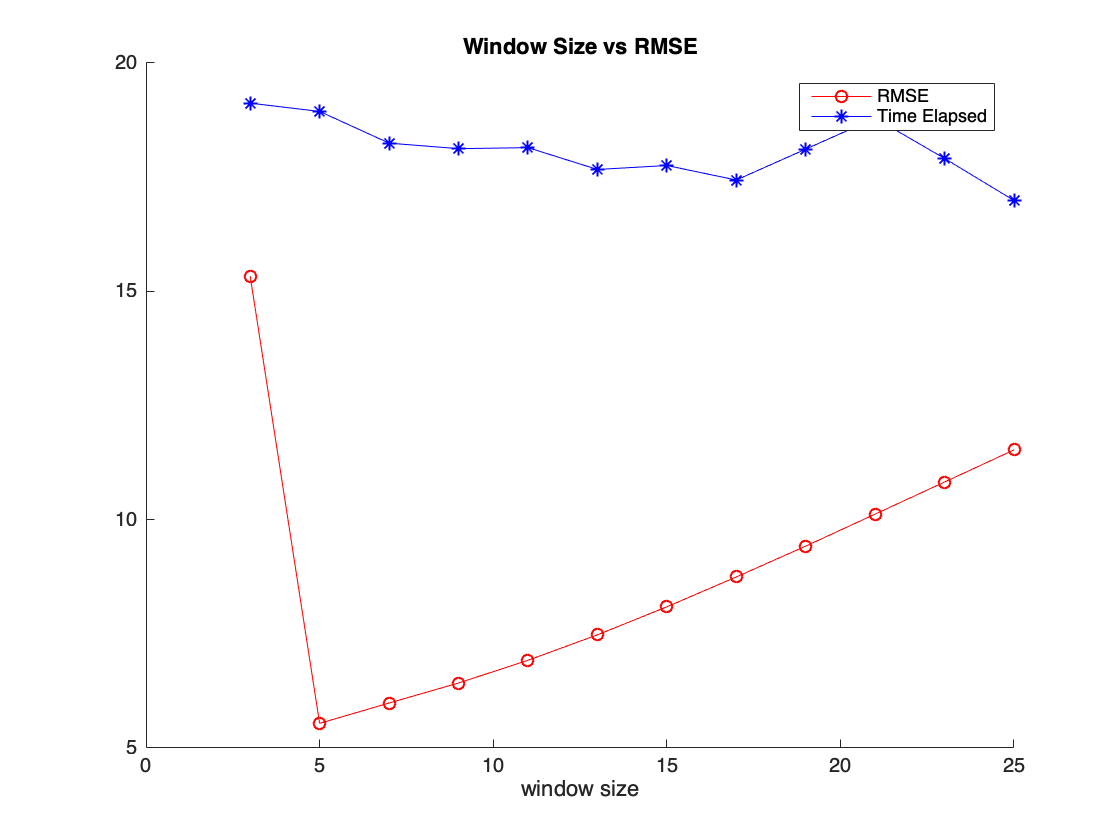
|  |  |  |
| --- | --- | --- |
| Win size | Time (s) | RMSE |
| 3 | 71.498 | 61.98 |
| 5 | 85.461 | 27.59 |
| 7 | 111.484 | 13.20 |
| 9 | 119.57 | 11.64 |
| 11 | 154.25 | 12.59 |
| 13 | 178.73 | 13.62 |
| 15 | 208.24 | 14.56 |
| 17 | 256.14 | 15.42 |
| 19 | 313.72 | 16.21 |
| 21 | 382.08 | 16.95 |
| 23 | 406.02 | 17.63 |
| 25 | 440.34 | 18.29 |

For median filter version 1, the bigger the window size, the better the filtering results are. From 3 experiments above, a bigger window size leads to a longer time to complete the process, but the smaller RMSE we get. However, visually speaking, larger window\_size removes noise, but it makes the image blurry and we lost a lot of information, especially on edges and corners.

1. Histogram Median Filter

Sample of Pears 60% noise results

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Window size | Time | RMSE |
| A screenshot of a social media post  Description automatically generated | 3x3 | 19.1206 | 15.3345 |
| A screenshot of a cell phone  Description automatically generated | 15x15 | 17.755 | 8.0914 |
| A picture containing screenshot  Description automatically generated | 25x25 | 16.989 | 11.5282 |



|  |  |  |
| --- | --- | --- |
| Win size | Time (s) | RMSE |
| 3 | 19.12 | 15.3345 |
| 5 | 18.94 | 5.5399 |
| 7 | 18.25 | 5.9848 |
| 9 | 18.12 | 6.4230 |
| 11 | 18.15 | 6.9202 |
| 13 | 17.67 | 7.4778 |
| 15 | 17.76 | 8.0914 |
| 17 | 17.44 | 8.7462 |
| 19 | 18.11 | 9.4201 |
| 21 | 18.79 | 10.1220 |
| 23 | 17.90 | 10.8246 |
| 25 | 16.98 | 11.5282 |

The RMSE for the same image in median filter using sort and histogram is identical. It clarifies that my algorithm using histogram is correct. However, an interesting finding is time elapsed to denoised the same image. In the experiment of range 3-25, sort median filter is a lot faster, it contradicts histogram’s claim. In window size 3x3, sort is 20 times faster compare to histogram. However, it does not conclude that way. In 3x3, the sort only sorting 9 elements of array, but bigger the window size will requires a lot more resources, because they are sorting more arrays.

We can see it from graph of both algorithms. Time elapsed of using sort is increasing over time, but histogram is fairly stable, even decreasing. Then, I tried to test the algorithm using window size 105 on pears image, sort median filter executes in 80.964551 seconds, but histogram only takes 15.675738 seconds to execute. This is because with 105 window size, sort needs to sort 11,025 elements, but histogram take the hist of 11,025 which requires so much less resources. Therefore, histogram is works better when the window size is bigger.

ABQ 60%

I tried to filter ABQ 60% using histogram, but it takes forever to execute. This is because ABQ image is huge. In 1 hour, it finishes 2 window\_size, which are 3 and 5.

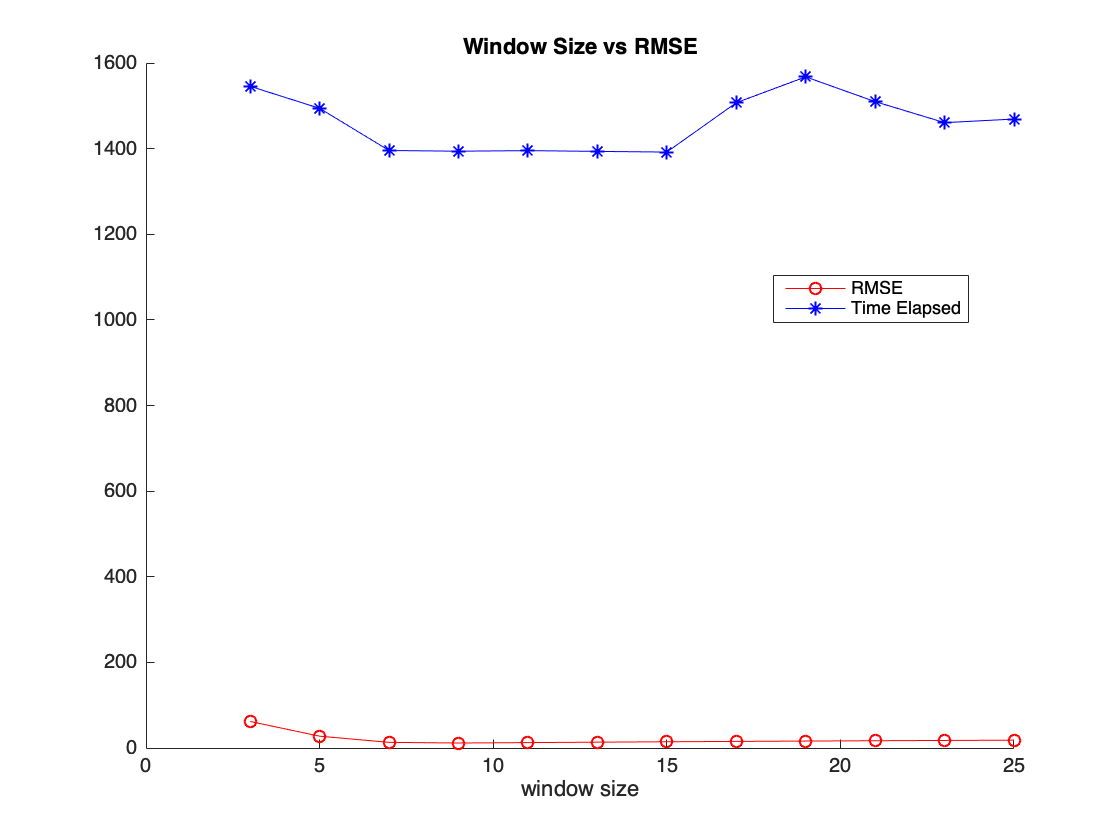
window\_size 3

Elapsed time is 1545.524351 seconds.

RMSE = 61.9841

RMSE is still identical compare to sort median filter, but it takes 26 minutes to execute window\_size=3 alone. After more than 5 hours, this is the result of window size 3 to 25

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Window size | Time | RMSE |
| A close up of a piece of paper  Description automatically generated | 3x3 | 1545.52435 | 61.9841 |
| A close up of a piece of paper  Description automatically generated | 15x15 | 1391.951115 | 14.5673 |
| A close up of a piece of paper  Description automatically generated | 25x25 | 1469.104358 | 18.2859 |



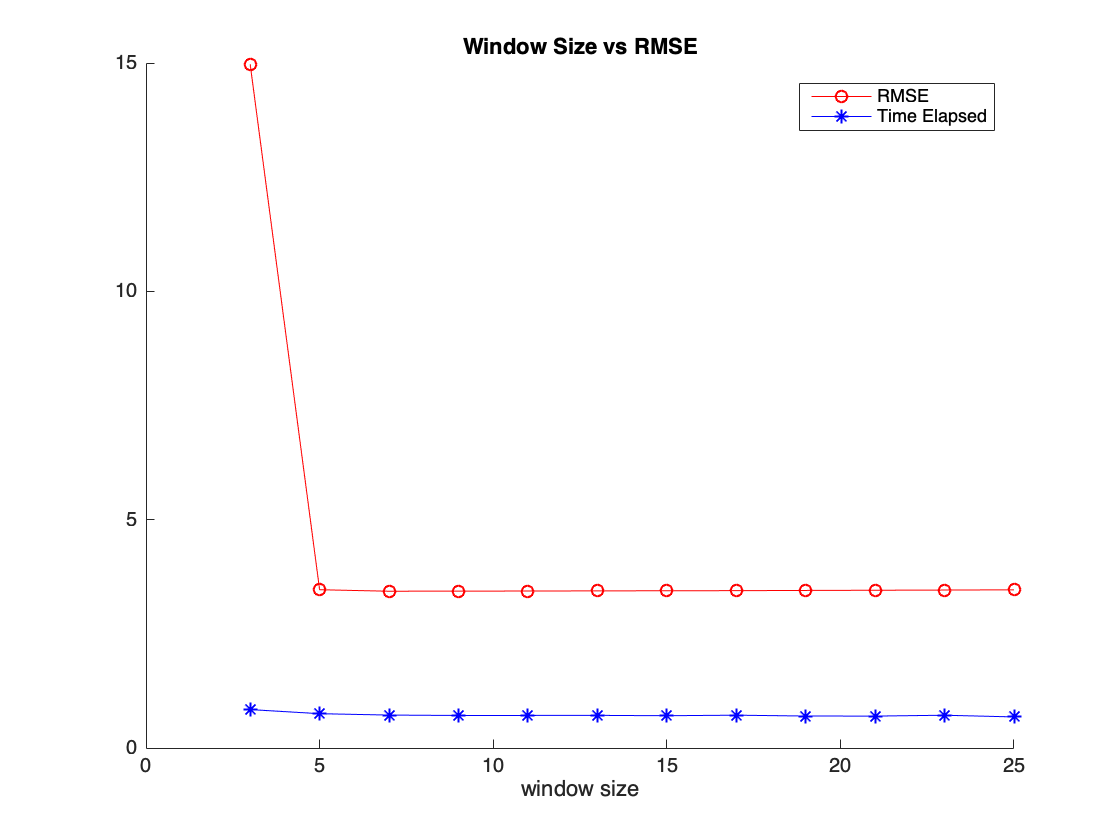
|  |  |  |
| --- | --- | --- |
| Win size | Time (s) | RMSE |
| 3 | 1545.52 | 61.98 |
| 5 | 1494.13 | 27.59 |
| 7 | 1395.63 | 13.20 |
| 9 | 1393.98 | 11.64 |
| 11 | 1395.33 | 12.59 |
| 13 | 1393.54 | 13.62 |
| 15 | 1391.95 | 14.56 |
| 17 | 1507.18 | 15.42 |
| 19 | 1567.59 | 16.21 |
| 21 | 1509.78 | 16.95 |
| 23 | 1460.67 | 17.63 |
| 25 | 1469.10 | 18.29 |

In this experiment median filter with histogram is super slow.

1. Adaptive Median Filter

Pears 60%

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Max Win size | Time(s) | RMSE |
|  | 5x5 | 0.734430 | 3.4658 |
|  | 15x15 | 0.758240 | 3.4430 |
|  | 25x25 | 1.003224 | 3.4637 |

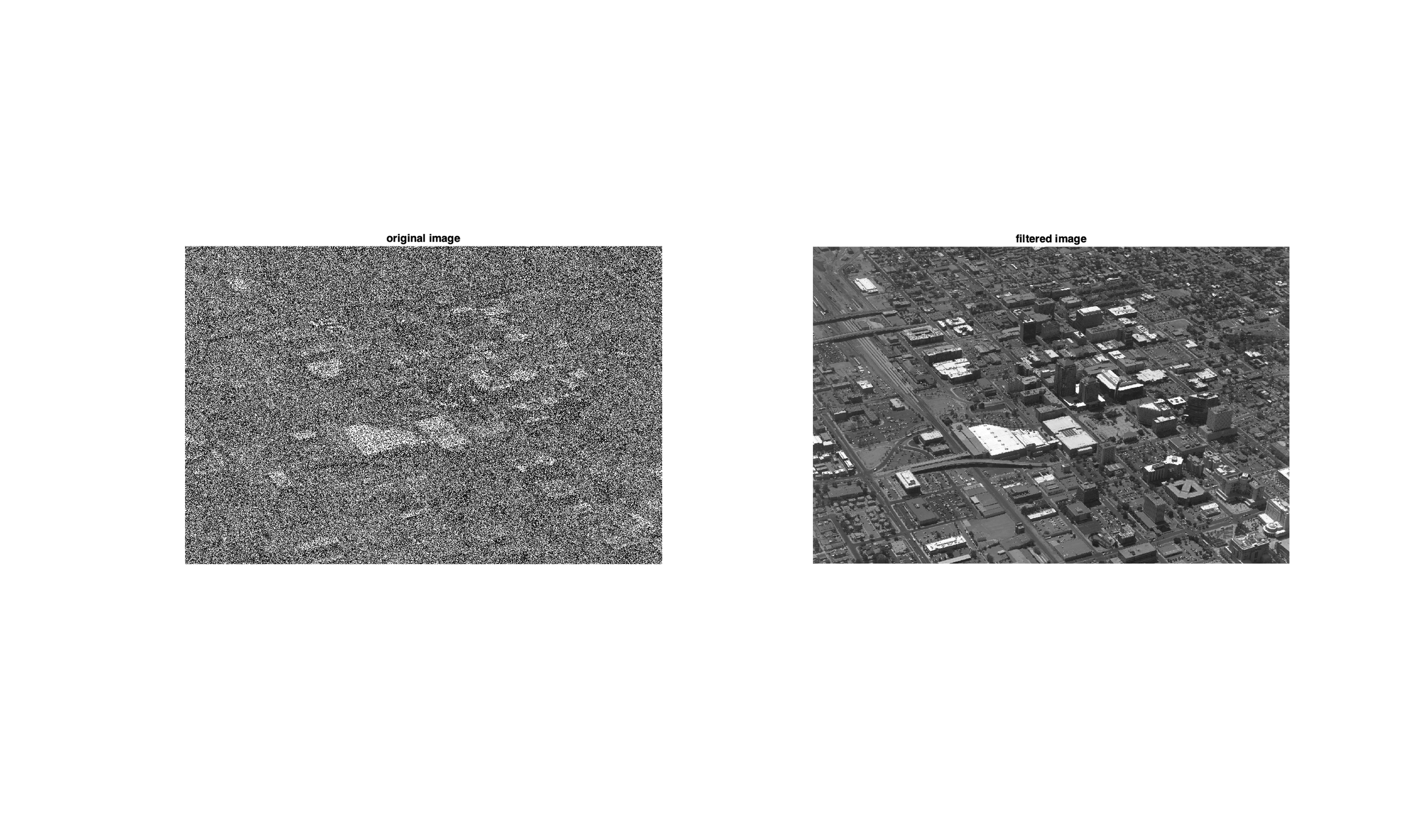


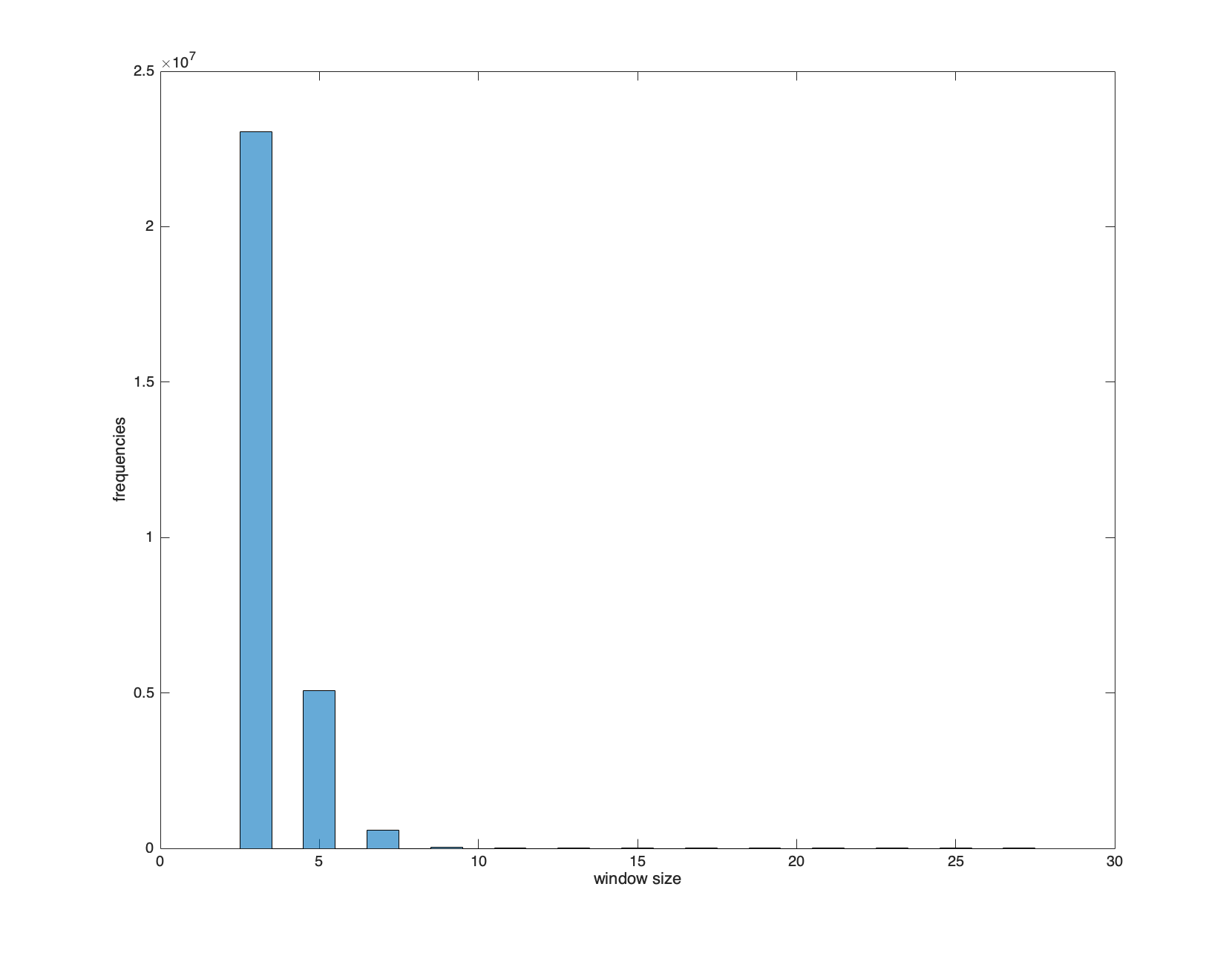
|  |  |  |
| --- | --- | --- |
| Max Win Size | Time (s) | RMSE |
| 3 | 0.727007 | 14.9734 |
| 5 | 0.734430 | 3.4658 |
| 7 | 0.753620 | 3.4330 |
| 9 | 0.741663 | 3.4361 |
| 11 | 0.737330 | 3.4369 |
| 13 | 0.743319 | 3.4409 |
| 15 | 0.758240 | 3.4430 |
| 17 | 0.764870 | 3.4454 |
| 19 | 0.791594 | 3.4502 |
| 21 | 0.810517 | 3.4545 |
| 23 | 0.768411 | 3.4586 |
| 25 | 1.003224 | 3.4637 |

As we can see, the result of using adaptive filter is impressive. Denoised image has detail and no blur. It preserves the edges really well. RMSE evaluations proves this point, adaptive median filter has the lowest RMSE compare to previous 2 method. And time evaluation shows that this method performs faster compare to other 2 methods. The reason is, based on the histogram window\_size vs frequency above, the majority of window size used in all cases window\_max is 3x3, using sort that we use, it does sort only 9 elements. Therefore, it performs fast.

ABQ 60%

Result of ABQ with 60% noise with window size min 3 and max 25



**

Elapsed time is 166.411179 seconds.

RMSE 7.2416

Result of ABQ with 60% noise based on different max window size allowed by the user

|  |  |  |  |
| --- | --- | --- | --- |
| Result | Max Win size | Time(s) | RMSE |
| A close up of a piece of paper  Description automatically generated | 5x5 | 180.591851 | 21.6884 |
| A close up of a piece of paper  Description automatically generated | 15x15 | 167.527600 | 7.2393 |
| A close up of a piece of paper  Description automatically generated | 25x25 | 182.033166 | 7.2416 |

A screenshot of a cell phone

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Max Win Size | Time (s) | RMSE |
| 3 | 166.415227 | 61.8819 |
| 5 | 180.591851 | 21.6884 |
| 7 | 183.238089 | 8.9176 |
| 9 | 180.110875 | 7.3066 |
| 11 | 175.463682 | 7.2391 |
| 13 | 166.340203 | 7.2387 |
| 15 | 167.527600 | 7.2393 |
| 17 | 166.389025 | 7.2398 |
| 19 | 166.310213 | 7.2403 |
| 21 | 166.162990 | 7.2407 |
| 23 | 168.274904 | 7.2412 |
| 25 | 182.033166 | 7.2416 |

With max window size 5x5, adaptive median filter still have some salt and pepper noise and high RMSE. But, with max window size 7 and up, it performs a lot better and RMSE seems stable at around 7 greyscale error, which is really good. Visually, the result of max window size 7 and up is giving us sharper texture, edges, and corners. This tackles the problem of 2 previous median filter that we build.

# Conclusion

Median filters proven have the ability to remove noise if we have the right window size. If the window size is too small, we will have some noise left, otherwise we will lost so many information, such as texture, edges, and corners since the image is become blurred.

Median filter using sort is good if the window size is small since we only need to sort small number of elements, but it will takes so much time when the window size is big. Therefore, if the window size that we are going to use is big, we can use histogram to find the median. This will save time since the complexity is O(window\_size) compare to sorting O(w2 log w2). But still, in small window size sorting algorithm performs faster.

Finally, we tackle the problem of losing image texture with adaptive median filter. It increases the window size if the median of the window is impulse. This method is mathematically and visually better compare to median filter using sort and histogram. The drawback is slower compare to median sort filter, but still faster compare to my histogram median filter.