JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 25 No. 1 (2017), pp. 7-21

Computational Complexity and Numerical Optimization of Adaptive Kuwahara Filter

Krzysztof Bartyzel[1]

[1]*Katolicki Uniwersytet Lubelski Jana Pawła II*
Katedra Analizy Obrazów
al. Racławickie 14, 20-950 Lublin
kbartyzel@kul.lublin.pl

Abstract. *Adaprive Kuwahara filter produces very interesting results and significantly improves the efficiency and performance of the original algorithm in the context of noise reduction without blurring the edges. This document contains the experimental and theoretical comparison of the computational complexity of the modified algorithm and the consideration of optimization methods.*
Keywords: *Noise reduction, Image processing, Image filtering, Computational complexity.*

.

1. Introduction

The issue of noise reduction in images is very important because noises can appear during image acquisition, transmission or compression and decompression. This is very complex matter and we still do not have perfect solution. The issue is still up to date and new concepts are continuously created. Some new interested approaches we can find in [1, 2, 3, 4]. Unfortunately, very often we have to deal with situations when reading new paper where the author suggests new solutions

or significant changes the upgrades are not described briefly enough. Unnecessary mental shortcuts make the recipient not able to do computer implementation or to suffer from serious technical problems. Even if he succeed, received results can be very disappointing or differing from the original paper. It should be also noted that the image analysis and processing operations are extremely compiled and require vast amounts of calculations. Although we have powerful computers but we still need more and more computing power. We cannot assume that in one day expectations of users will stop to grow. On the contrary, we should assume that the users expectations will grow continuously. In addition, growth of the computing power is often the result of multicore processors. To take advantage of the latest capabilities is necessary to adjust the algorithm to the multicore architecture. Usually, this is not a trivial task and often in itself is very valuable, worthy sharing work. But accordingly to Amdahl's law [5] we can expect significant efficiency growth, especially when comparing with single core applications. All these arguments make that the issue of implementation and optimization is crucial for usability. In this paper are presented the results of further research conducted on Adaptive Kuwahara Filter introduced in [6]. There will be presented concepts how to optimize the algorithm and the real execution times for comparing the algorithms on a considerable amount of sample data.

2. Description of the Adaptive Kuwahara filter algorithm

To ensure notation consistency to the next part of the article the Adaptive Kuwahara filter [6] will be presented. The standard Kuwahara filter requires a strictly defined window size. In the case of the proposed modification of the filter the window size is changing, just as it did in the adaptive median filter, depending on the local properties of the image. The initial window size is 3x3.

Step 1: The filter window should be divided into 4 areas by following the original algorithm. Let us denote them as θ_k where $k \in \{0, 1, 2, 3\}$. Initially, each of these areas will consist of 4 pixels. For the purposes of this algorithm, let us call these four areas the basic areas.

Step 2: For each of these areas the values of mean m_k and variance δ^2_{min} are calculated. As before, the value of a specific pixel can be the value of color intensity,

brightness, or any other calculated value. The mean and variance are calculated according to the formula:

$$m_k = \frac{1}{N_k} * \sum_{(x,y)\in\theta_k} \varphi(f(x,y)) \tag{1}$$

$$\delta_k^2 = \frac{1}{N_k} * \sum_{(x,y)\in\theta_k} \left[\varphi(f(x,y)) - m_k\right]^2 \tag{2}$$

where:

$k \in \{0, 1, 2, 3\}$,

f is the source image function,

φ is a function calculating the value of a particular pixel,

N_k is the number of pixels in the current area, in the first cycle the value is 4.

Step 3: Each of the basic areas is considered separately. For the chosen area the size of the window is increased by 1. Next, for the new window size, mean $\overline{m_k}$ and variance $\overline{\delta_k^2}$ have to be calculated according to the formulas presented in the previous step.

An example, how the resizing the filter window is performed, can be seen in Figure 1. The filter window central element and the elements included in area θ_k as it increases are highlighted.

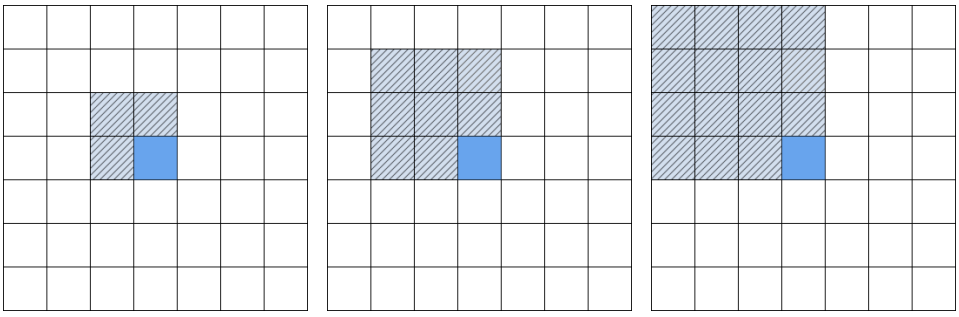

Figure 1: Resizing the filter window and items included in the new area θ_k.

If the variance of the new area ($\overline{\delta_k^2}$) is smaller than before the resizing of the

filter window (δ_k^2), then the mean and variance of the basic area k take the newly calculated values:

$$m_k := \overline{m_k} \tag{3}$$

$$\delta_k^2 := \overline{\delta_k^2} \tag{4}$$

Then we continue to increase the size of the window for the basic area selected in the current step until its size reaches the maximum allowable size or until the variance of the newly enlarged area is greater than that calculated in the previous cycle of the algorithm. In this way the minimum variance and the corresponding average value will be achieved for the basic area k. For further calculations, it is not necessary to know the size of the window for which the variance and the mean values were calculated.

The calculations shown in Step 3 must be repeated separately for each of the four basic areas.

Step 4: Finally, we compare the variance of all four areas. At this stage each of the basic areas can be made with different quantity of items. We are looking for an index of the area for which the variance is the smallest.

$$\delta_{min}^2 = \min_{k\in\{1,2,3,4\}} (\delta_k^2) \tag{5}$$

The resulting value of the output pixel is the average value of the basic area for which the variance was the smallest. Figure 2 presents an example of possible window size distribution for items included in the four basic areas and common parts of these areas. The center pixel is marked with black color.

3. Numerical optimization

A very important disadvantage of the adaptive median filter is its relatively long execution time. One of the reasons for this is the fact that, in each increasing window cycle, calculating the median value of the elements in the wider window, there is no easy way to take into account the calculation obtained in the previous cycle of the algorithm. The most complex operation is the re-establishing of the

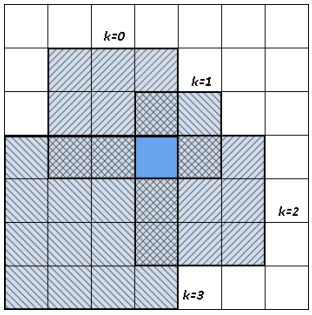

Figure 2: An example of possible window size distribution for basic areas.

proper order of the elements. In the case of the adaptive Kuwahara filter it is possible to use the results from the previous cycle, so one can significantly improve the performance of the algorithm. More precisely, it comes to improve the calculation of the mean and the variance after increasing the size of the basic area window.

Both the statistics and the probability theory provide that:

$$D^2X = E(X - EX)^2 \tag{6}$$

This is just another notation of the formula used in step 2 of the presented algorithm. Expanding this equation we get:

$$D^2X = E(X - EX)^2 = E(X^2 - 2 * EX * X + (EX)^2) = EX^2 - (EX)^2 \tag{7}$$

It means that the formulas (1 and 2) used in step 2 can be written as:

$$m_k = \frac{1}{N_k} * \sum_{(x,y)\in\theta_k} \varphi\left(f(x,y)\right) \tag{8}$$

$$\delta_k^2 = \frac{1}{N_k} * \sum_{(x,y)\in\theta_k} \left[\varphi\left(f(x,y)\right)\right]^2 - \left[\frac{1}{N_k} * \sum_{(x,y)\in\theta_k} \varphi\left(f(x,y)\right)\right]^2 \tag{9}$$

For simplicity, we can note:

$$m_k = \frac{1}{N_k} * S \tag{10}$$

$$\delta_k^2 = \frac{1}{N_k} * S^2 - \left[\frac{1}{N_k} * S\right]^2 \tag{11}$$

where:

$$S = \sum_{(x,y)\in\theta_k} \varphi\left(f(x,y)\right),$$
$$S^2 = \sum_{(x,y)\in\theta_k} \left[\varphi\left(f(x,y)\right)\right]^2.$$

Using the second version of the formulas we gain the possibility to calculate variance without knowing the mean value of the elements. Both values can be calculated simultaneously. This property is very important because it allows to calculate m_k^2 and δ_k^2 while reviewing elements of θ_k only once. This means reducing the input and output operations which in computer calculations are always very expensive. In addition, we gain the possibility to use the calculations from the previous cycle for calculating the values of the mean and variance after increasing the area window size. Increasing the area window size entails increasing the number of elements in θ_k. This means that in the next cycle the sums S and S^2 should be supplemented with elements added to collection θ_k. In other words, when calculating the mean value ($\overline{m_k^2}$) and variance ($\overline{\delta_k^2}$) for increased windows we can use previously calculated values and each pixel will be processed only once for each basic area considered separately.

An example of items that should be added to collection θ_k and to the sum S and S^2 after resizing windows is given in Figure 3.

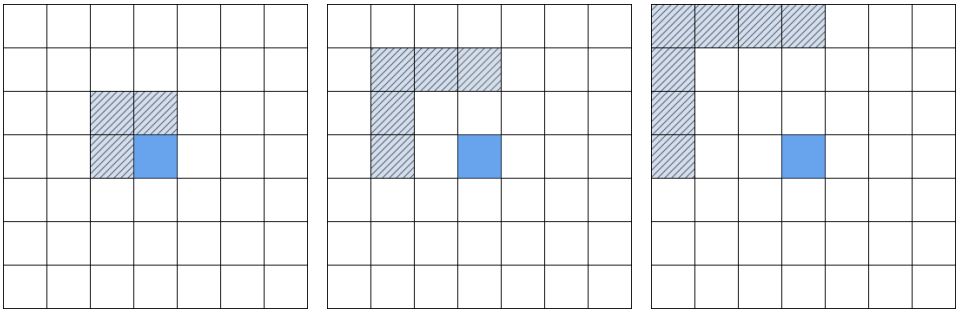

Figure 3: Items that should be added to θ_k after resizing windows.

Other concepts for improving the efficency

Another way for optimization is to comparing the variance of each area at an earlier stage rather than continuing calculation separately (until its size reaches the maximum allowable size or until the variance of the newly enlarged area is

greater than that calculated in the previous cycle of the algorithm) and compare them after completion. If one or more area, for a few window resizing cycles (on the basis of experiments it seems that the value of three is sufficient) reaches scores worse than other than with big probability we can exclude such area from further calculation. Same as maximum allowable window size this value can be treated as additional parameter for presented algorithm. An example was shown at the fig. 4. More white means more resizing cycles were needed. The performed experiments clearly demonstrate that such situations (when it was necessary to calculate the number of cycles) occurs mainly at the edges.

Figure 4: Example of window resizing cycles occurrence.

During the implementation one should also be aware of the possibilities of the latest computers and construct a particular implementation in a way that can perform calculations on multiple cores. Detailed technical specifications and instructions how one can implement image processing algorithms with examples in Microsoft .NET platform are included in [7]. The main concept assumes that the image will be divided into disjoint areas and each area will be processed separately on dedicated core. The amount of the areas depend on the number of available cores.

Table 1: Filter algorithms time comparison (in seconds) for 5 different images of size of 190x220 on the processor Intel Core 2 Duo P8700 (2 cores, 2.54 GHz).

SN	Kuwahara Filter 3x3	Kuwahara Filter 11x11	Adaptive Median Filter	Adaptive Kuwahara Filter
1	0.031	0.124	0.405	0.048
2	0.064	0.109	0.468	0.062
3	0.015	0.118	0.405	0.046
4	0.015	0.124	0.561	0.031
5	0.031	0.124	0.405	0.048
AVG	0.028	0.117	0.464	0.044
SD	0.021	0.008	0.065	0.013

4. Comparison of computational complexity

The algorithm discussed above has been implemented. In order to obtain similar results, implementation diligence in each of the compared algorithms was similar. However, willingness to use the capabilities of the latest computers forced the need to modify the algorithms in such a way as to be able to perform calculations simultaneously on multiple-core processors.

The following section presents the real execution times and theoretical considerations of the maximum computational complexity for comparing the algorithms.

The results obtained are presented in Tables 1-7. The practical experiment shows that the proposed modification is very effective and the modified filter can be up to 10 times faster than the adaptive median filter.

Tables 1-3 present a comparison of the execution times for sample images of different sizes. In addition, Tables 4 and 5 show the results of an experiment involving comparison of execution times for a large quantity sample images. The results obtained clearly demonstrate the very considerable advantage of the proposed filter.

In order to better explore the possibilities and to verify the capabilities of the proposed algorithm in comparison to the other noise reduction solutions, a second experiment was conducted. As previously, two noise algorithms were used. The first one was salt and pepper noise (adding white and black pixels) with uniform distribution on a specified percentage of the image surface. The second, more sophisticated, added noise to a specified percentage of the image with random in-

Table 2: Filter algorithms time comparison (in seconds) for 5 different images of size of 768x576 on the processor Intel Core 2 Duo P8700 (2-core, 2.54 GHz).

SN	Kuwahara Filter 3x3	Kuwahara Filter 11x11	Adaptive Median Filter	Adaptive Kuwahara Filter
1	0.156	0.702	16.239	0.608
2	0.249	0.686	15.943	0.546
3	0.187	0.670	15.990	0.468
4	0.109	0.904	15.865	0.405
5	0.109	0.686	15.679	0.421
AVG	0.162	0.730	15.943	0.490
SD	0.059	0.098	0.203	0.086

Table 3: Filter algorithms time comparison (in seconds) for 5 different images of size of 1779x2473 on the processor Intel Core 2 Duo P8700 (2-core, 2.54 GHz).

SN	Kuwahara Filter 3x3	Kuwahara Filter 11x11	Adaptive Median Filter	Adaptive Kuwahara Filter
1	1.338	7.238	48.438	3.884
2	1.076	7.394	48.890	3.822
3	1.138	7.222	50.185	3.744
4	1.435	6.942	49.865	4.056
5	1.284	6.957	49.078	3.902
AVG	1.254	7.151	49.291	3.882
SD	0.147	0.196	0.718	0.115

Table 4: Filter algorithms time comparison (in miliseconds) for 197 different images of size of 550x413 (0.23 Mpx) on the processor Intel Core 2 Duo P8700 (2-core, 2.54 GHz).

SN	Kuwahara Filter 3x3	Kuwahara Filter 11x11	Adaptive Median Filter	Adaptive Kuwahara Filter
AVG	62.479	358.071	1091.716	230.279
SD	8.091	23.251	226.006	27.220

Table 5: Filter algorithms time comparison (in miliseconds) for 166 different images of size of 3072x2304 (7 Mpx) on the processor Intel Core 2 Duo P8700 (2-core, 2.54 GHz).

SN	Kuwahara Filter 3x3	Kuwahara Filter 11x11	Adaptive Median Filter	Adaptive Kuwahara Filter
AVG[ms]	2140.029	13773.827	40688.238	7142.506
SD[ms]	340.706	2762.262	3417.319	530.776

tensity, which either lightened or darkened a specific pixel.

A single test procedure was as follows:

- noise was added to the original image
- the original image was compared with the noised image and statistics of this comparison were collected
- the noised image was filtered by the various algorithms, the results of these operations were compared with the original image and statistics of the comparisons were collected
- the test was performed for each image

Comparing two images we get the following information:

- the average error, the average intensity differences of each pixel for the entire image
- the variance and the standard deviation
- the amount of modified pixels against the original image
- the time taken to complete the operation and the comparisons

In Table 6 were placed aggregated statistics obtained in second experiment. While in Table 7 placed the same results as in Table 6 but in the percentage view, comparison of the performance of individual algorithms to proposed solution. Obtaining rate over 100% means higher performance (for particular aspect) achieved by the proposed filter.

For complete evaluation of the proposed solution, apart from the presentation of experimental data also the theoretical calculations of the maximum computational complexity have to be performed.

Table 6: Filter algorithms aggregated statistics (in seconds) obtained in second experiment for 68 different images of size of 3072x2304 (7 Mpx) on the processor Intel Core i5 3210M (2-core, 2.5 GHz).

Noise type		No Filter	Kuwahara Filter 3x3	Kuwahara Filter 11x11	Kuwahara Median Filter	Adaptive Kuwahara Filter
1% salt and pepper	Sum of AVG	26714.26	702.77	1466.61	921.95	580.23
	Sum of SD	12845.12	2088.07	3539.43	2808.80	1755.19
	Sum of Time[s]	12679	123424	672597	4044551	437120
5% salt and pepper	Sum of AVG	26714.26	702.77	1466.61	921.95	580.23
	Sum of SD	12845.12	2088.07	3539.43	2808.80	1755.19
	Sum of Time[s]	12679	123424	672597	4044551	437120
5% additive salt and pepper	Sum of AVG	6247.97	523.96	1151.93	815.69	475.35
	Sum of SD	3598.03	1442.64	2873.89	2483.09	1419.51
	Sum of Time[s]	12127	122265	676171	3212945	428545
25% additive salt and pepper	Sum of AVG	6249.06	1038.79	1276.62	1759.85	732.30
	Sum of SD	3598.06	1711.57	2870.67	3129.58	1491.86
	Sum of Time[s]	13832	122175	672745	2438302	460966

Table 7: Percentage view of aggregated statistics obtained in second experiment, comparison of the performance of individual algorithms to proposed solution.

Noise type		Kuwahara Filter 3x3	Kuwahara Filter 11x11	Adaptive Kuwahara Filter
1% salt and pepper	Sum of AVG	121,12%	252,76%	158,89%
	Sum of SD	118,97%	201,66%	160,03%
	Sum of Time[s]	28,24%	153,87%	925,27%
5% salt and pepper	Sum of AVG	208,53%	257,54%	119,51%
	Sum of SD	187,14%	187,98%	157,29%
	Sum of Time[s]	25,57%	142,79%	618,02%
5% additive salt and pepper	Sum of AVG	110,23%	242,33%	171,6%
	Sum of SD	101,63%	202,46%	174,93%
	Sum of Time[s]	28,53%	157,78%	749,73%
25% additive salt and pepper	Sum of AVG	141,85%	174,33%	240,32%
	Sum of SD	114,73%	192,42%	209,78%
	Sum of Time[s]	26,5%	145,94%	528,95%

Comparing the computational complexity of algorithms, we can skip the part related to the size of the image because the calculations must be performed for each pixel for each algorithm and therefore have a linear effect on the calculation execution time.

The most significant impact will, therefore, be that of the size of the window for which the calculation is performed. The minimum size is 3x3, and the maximum we define as $(2 * k_{max} + 1) * (2 * k_{max} + 1)$ where k_{max} is a natural number.

Kuwahara Filter

In the case of this filter the most complicated operation is the calculation of the mean and variance, therefore:

$$\Theta(f(k_{max})) = 4 * (3 * (k_{max} + 1) * (k_{max} + 1)) \approx \Theta\left(12 * {k_{max}}^2\right)$$

Adaptive median filter

The adaptive median filter calculates the median value and, therefore, requires to sort the items. Then the size of the window may be increased up to the maximum value, therefore:

$$\Theta(f(k_{max})) = \sum_{k=1}^{k_{max}} (2 * k + 1)^2 = \sum_{k=1}^{k_{max}} \left(4 * k^2 + 4 * k + 1\right) =$$

$$= 4 * \frac{k_{max} * (k_{max}+1) * (2 * k_{max}+1)}{6} + 4 * \frac{k_{max} * (k_{max}+1)}{2} + k_{max} \approx \Theta\left({}^4/_3 * {k_{max}}^3\right)$$

Adaptive Kuwahara filter

This filter requires the determination of the average and variance for the increasing windows sizes. In an extreme situation, the computational complexity will be equal to:

$$\Theta(f(k_{max})) = 4 * \sum_{k=1}^{k_{max}} 3 * (2 * k + 1) = 12 * \sum_{k=1}^{k_{max}} (2 * k + 1) =$$

$$= 12 * \left(2 * \frac{k_{max} * (k_{max}+1)}{2} + k_{max}\right) = 12 * \left({k_{max}}^2 + 2 * k_{max}\right) \approx \Theta\left(12 * {k_{max}}^2\right)$$

Comparing the computational complexity it can be concluded that the computational complexity of the Adaptive Kuwahara filter, in terms of the order of magnitude, is comparable with the standard Kuwahara filter, and much smaller than that of the Adaptive median filter. This does not quite reflect the real execution time of the algorithm, but says only about its maximum value. It is important to

also take into consideration the actual data from the experiment, where the Adaptive Kuwahara filter advantage is even more apparent. These data were presented in the earlier part of this work.

5. Summary

Very important aspect, as was mentioned previously, is the optimization of computational complexity. On the basis of the presented experiments and theoretical considerations obtained results allow to maintain the claim that Adaptive Kuwahara Filter can be successfully used. It is very fast in operation, especially compared to the adaptive median filter. To fully demonstrate that thesis the direction for further research should be a detailed Image Quality Assessment with other filters.

References

[1] Bhadouria, V. and Ghoshal, D., *A study on genetic expression programming-based approach for impulse noise reduction in images*, Signal, Image and Video Processing, 2016, pp. 575–584.

[2] Esakkirajan, S., V. T. S. A. and PremChand, C., *Removal of high density salt and pepper noise through modified decision based unsymmetric trimmed median filter*, IEEE Signal Processing Letters, 2011, pp. 287–290.

[3] Bhadouria, V.S., G. D. and Siddiqi, A., *A new approach for high density saturated impulse noise removal using decision-based coupled window median filter*, Signal, Image Video Processing, 2014.

[4] Nair, M. and Raju, G., *A new fuzzy-based decision algorithm for high-density impulse noise removal*, Signal, Image Video Processing, 2012, pp. 579–595.

[5] G.M, A., *Validity of the single processor approach to achieving large scale computing capabilities*, AFIPS spring joint computer conference, 1967, pp. 483–485.

[6] Bartyzel, K., *Adaptive Kuwahara filter*, Signal, Image and Video Processing, 2015, pp. 1–8.

[7] Bartyzel, K., *Algorithms optimization for the image processing and analysis by constructing parallel solutions*, Studia Informatica, Systems and information technology, 2015, pp. 5–14.