International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 16, No. 1, pp. 55 – 67, 2019

FASTER IMAGE FILTERING VIA PARALLEL PROGRAMMING

KAMIL KSIĄŻEK, ZBIGNIEW MARSZAŁEK

Institute of Theoretical and Applied Informatics, Polish Academy of Science

*Institute of Mathematics, Silesian University of Technology, Kaszubska 23,
Gliwice, 44-100, Poland
kamilksiazek95@gmail.com, zbigniew.marszalek@polsl.pl*

GIACOMO CAPIZZI, CHRISTIAN NAPOLI

*Department of Electrical, Electronics and Informatics Engineering, University of Catania,Viale A. Doria 6,
Catania, 95125, Italy
beritelli@dieei.unict.it,capizzi@dieei.unict.it, glosciuto@dii.unict.it,coco@diees.unict.it*

DAWID POŁAP, MARCIN WOŹNIAK

*Institute of Mathematics, Silesian University of Technology, Kaszubska 23,
Gliwice, 44-100, Poland
dawid.polap@polsl.pl, marcin.wozniak@polsl.pl*

Various computer methods are sourced in parallel programming. Advances in methods and techniques with their appropriate usage are beneficial for multimedia applications. Parallelization can significantly decrease the time of calculations. In this article, we analyze how the speed of calculations is influenced by the usage of parallel algorithms in image filtering processes. Additionally, we define a novel cluster selection method which helps in more efficient processing the images. We present a method based on multithreading and the division of the image for rectangles, while as cluster selector we have used k-means algorithm. Proposed filtering is applied parallel on each part of the image, where each of threads is working for each task. Results show that our proposition can give positive results and faster filtering of images when compared to the classical approach. Furthermore, the algorithm connected with k-means and filtering can be very helpful in objects detection.

Keywords: Parallel programming, Image filtering, Laplacian, K-means, Multithreading.

1. Introduction

Parallel programming is currently a dynamically growing field of computer science. Modern multicore processors enable a significant reduction of computation time. Proper use of computing power is an enormous challenge for programmers. A skillful preparation of parallel instructions that are to be carried out causes a lot of problems. However, benefits of parallel programming are obvious. Some multimedia applications require a huge amount of time - it is visible for instance in image processing. Filtering the images containing the several thousand pixels can take a lot of time, especially, when there is a substantial number of images to be filtered. Therefore an intelligent methods that improve image processing are very important.

It is clear that image filtering has a lot of applications. It is possible to improve the quality of photos with blur, noise or other undesirable effects. Moreover, sharpening the edges can be helpful in the objects detection. Parallel methods can be very useful in graphics processing and cloud computing directed to multithreading. High-quality images are also crucial in medicine in diagnosis of diseases (for instance in X-ray pictures). Therefore, capturing the details is very important. In this article we present multithreading in image filtering, and its impact on the whole process. Our approach is designed to equally distribute work among all the threads. The input image is divided into equal rectangles and each thread filters only the designated area. Therefore by the proposed algorithm we construct a method which uses all available cores. Depending on the number of CPU threads in the computer we can significantly decrease time of processing, reaching even 90% of improvement. It has a significant importance for HD multimedia systems where all the images and multimedia streams are very complex structures. Therefore our proposed method may reduce time and improve the efficiency of processing. For the research we have used an architecture with 32 CPU threads and 320 GB of memory.

The main part of this article is following: Section II describes some related works, Section III presents a theoretical background of the image filtering and three Laplacian filters applied in the research. In Section IV it is shown a detailed description of the tested parallel method. Section V gives a results of measurements and Section VI contains conclusions and remarks after studies.

2. Related Works

Very important application of multitasking is connected with data processing. In [1] it was shown how to parallelize fast sorting algorithm, while in [2] it was proposed a new more efficient parallel merge sort algorithm. These decrease computation time in large databases for instance in case of data analysis. In [3] an overview on intelligent systems for data retrieval was discussed. Graphics processing is frequent and very important topic of many publications. In [4] it was proposed a method how to more efficiently analyse the information from images for detection, and in [5] a segmentation of images based on graph analysis of the semantic image structure was presented. The image decomposition method which combines information from the infrared and visible images is presented in [6]. This method can be helpful for instance in target recognition. In [7] and [8] was presented a system for image data classification by the use of fast selection methods based on shapes comparisons. Authors in [9] propose a Weighted Guided Image Filtering algorithm (WGIF) which prevents so called "halo artefacts" effect. In many cases combination of different data ensures more efficient analysis. Such approach in medical images is shown in [10]. The method intended for filter identification is presented in [11]. An interesting problem connected with underwater imaging is introduced in [12]. Authors in [13] show the Core algorithm designed for document's identification from images. It is compared with classical detectors like ORB, SIFT and SURF-BRISK. In our paper it will be shown a method which can speed up the calculations on images. In extension to initial research [20] we proposed a k-means method which additional boost clustering. A properly and quickly processed image streamlines further work. The presented method is intended for initial processing of the images.

3. Image filtering

A very common situation is that the analysis of the original image is difficult due to noise, blur or other factors. Furthermore, if the number of details is too large, a detection of the crucial parts of the image (for instance, the contours of presented objects) is impossible. Therefore it is necessary to pre-process the image. Further analysis will be easier thanks to such tools as the image filtering.

3.1. *Theoretical background*

One of the most popular colour models is RGB model [14]. Each pixel consists of three components: R (red), G (green) and B (blue). They can be integers from the range {0, 1, ..., 255}. For instance [0, 0, 0] represents black, [255, 255, 255] determines white, [255, 255, 0] yellow, etc.

In this paper we assume that calculations will be performed by using the RGB model. Let $R_{n\times m}$ be a two-dimensional array with the values of pixels of a given image (nis the width of an image and m is the height of an image, R[i, j] represents the position (i, j) on the image, $i \in (1, \dots, n)$, $j \in (1, \dots, m)$). The idea relies on modification of the image by moving the convolution mask over the pixels. The new values of three components R, G, B depend on the pixels in the nearest neighbourhood of the calculated one. This process is illustrated in Fig. 1, by the example of 3 on 3 mask. The new value of the position (x,y) depends on 9 pixels. Of course, other sizes are also allowed.

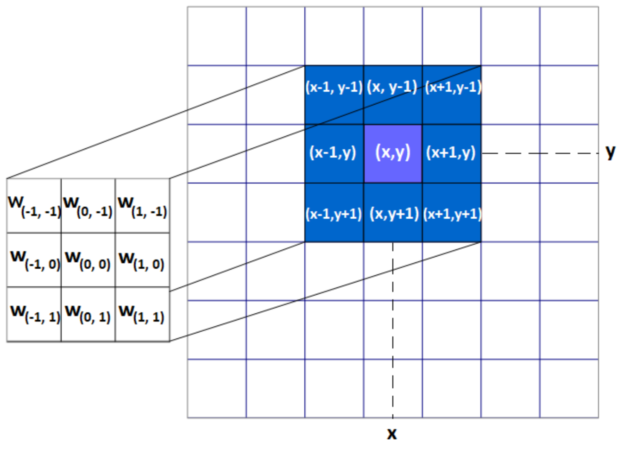

Fig.1.The sample of using a 3 on 3 convolutional mask.

The greater the mask, the larger number of pixels is taken into account during processing. The impact of each pixel is defined by the table of weights, called a filter. In our research, there were applied three types of convolution masks: 3 on 3, 5 on 5 and 9 on 9. The larger the filter is, the more details are lost [15]. During calculations on larger masks it is necessary to create an auxiliary image with borders filled with black pixels (the convolution masks exceeds the original one). This operation has an no significant influence on the final image but it enables the filtering.

The pattern for the new value of each component of the pixel located at position (i,j) in the case of 5 on 5 mask is as follows[16]:

$$R'(i,j) = \frac{1}{M} \sum_{k=-2}^{2} \sum_{l=-2}^{2} w(k,l) \cdot R(i+k, j+l)$$

where M is the sum of all values in the array (the convolution mask), w(k,l) is the weight of pixel located at position (i+k, j+l) and R(i+k, j+l) is the previous value of the pixel at given position. Sometimes M may be equal to 0. In that situation, the factor $\frac{1}{M}$ is omitted. Patterns for other convolution masks are created similarly.

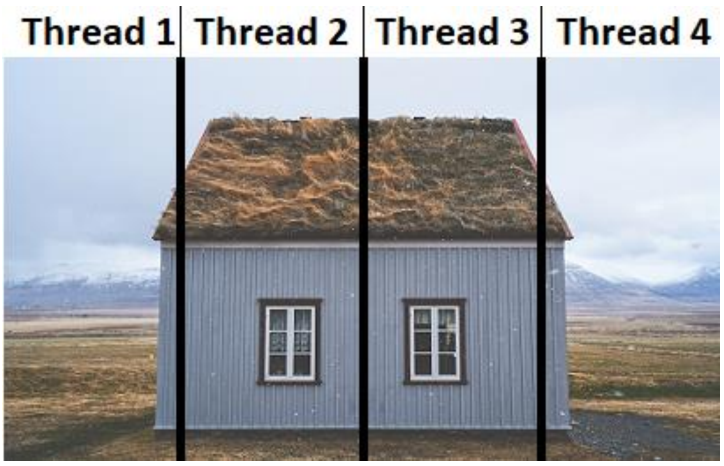

Fig. 2. Illustration of the image split into 4 rectangles. (The original photo was taken by Jonathan Andreo, and is available at unsplash.com).

3.2. *The applied filters*

During further calculations, three Laplacian filters will be used (Fig. 3) [17]. Their main task is sharpening the edges of the objects and hence, loosing of irrelevant details. The picture which has been filtered, is presented in Fig. 4. It is possible to see how the Laplacian filters influence the original image. The edges are therefore definitely more visible than the rest of the image. This kind of filters facilitates the detection of shapes.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

(a) 3×3 Laplacian filter

$$\begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

(b) 5×5 Laplacian filter

$$\begin{pmatrix} 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \\ 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\ 1 & 4 & 5 & 3 & 0 & 3 & 5 & 4 & 1 \\ 2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -24 & -40 & -24 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -24 & -12 & 3 & 5 & 2 \\ 1 & 4 & 5 & 3 & 0 & 3 & 5 & 4 & 1 \\ 1 & 2 & 4 & 5 & 5 & 5 & 4 & 2 & 1 \\ 0 & 1 & 1 & 2 & 2 & 2 & 1 & 1 & 0 \end{pmatrix}$$

(c) 9×9 Laplacian filter

Fig. 3. The presentation of Laplacian filters applied in the paper.

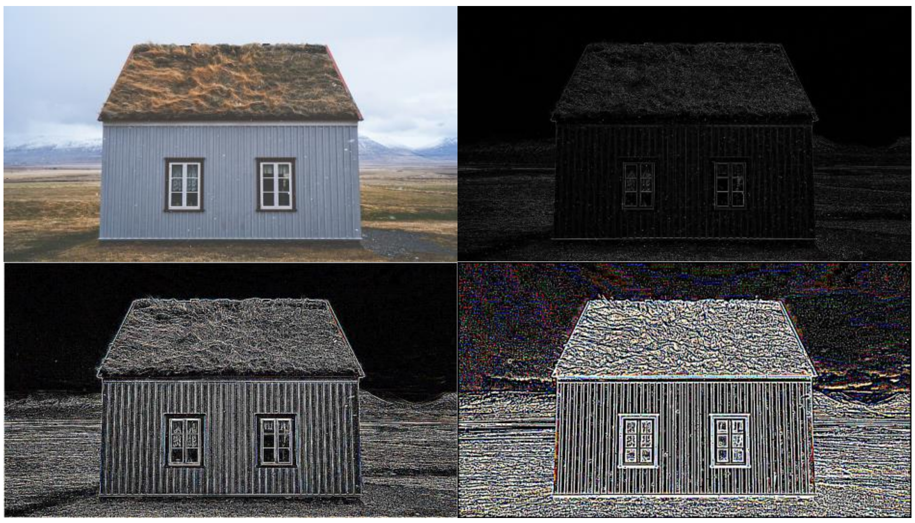

Fig. 4. The figure presents as follows: the original image and the image after 3 on 3 Laplacian filter (the first row) and images after 5 on 5 and 9 on 9 Laplacian filter (the second row). (The original photo was taken by Jonathan Andreo, and is available at unsplash.com).

4. Parallelization

Image filtering involves a lot of calculations. The larger the photo is, the greater the time of filtering is. In case of analyzing a large number of pictures, minimizing the computation time is crucial. The idea presented in this paper relies on moving the mask parallel over the image. The photo can be divided into t rectangles, where t is the number of threads (an exemplary division into 4 rectangles is presented in Fig. 2). Then each thread applies the filter to the allocated area and does not influence any other parts. Finally, all filtered fragments are combined into one image.

The only issue is to determine the way of assigning suitable areas for t threads. Each thread after running receives its own index (the numbering starts from zero). Let w be the width of the processed image (in pixels). Then the number of pixels per one thread is equal to $\left\lceil \frac{w}{t} \right\rceil$. The ceiling function is necessary because $\frac{w}{t}$ may not be an integer. In such a situation the last thread can have slightly more pixels to calculate. The Algorithm presents the pseudocode of the proposed method.

Algorithm 1 Pseudocode of the parallel method of the image filtering.

```
Input: the image for filtering, the size of the image: width $w$, height $h$, the convolution mask, number of threads $t$
  Calculate the number of pixels per one thread: $\lfloor \frac{w}{t} \rfloor$.
  Create $t$ threads.
  for $i = 0$ to $t - 1$ do
    Set the range for the thread from $i \cdot \lfloor \frac{w}{t} \rfloor + 1$ to $i \cdot \lfloor \frac{w}{t} \rfloor + \lfloor \frac{w}{t} \rfloor$.
    if $i = t - 1$ then
      Set the range for $(t-1)$-th thread from $(t-1) \cdot \lfloor \frac{w}{t} \rfloor + 1$ to $w$.
    end if
    Filter the determined area according to the convolution mask.
  end for
  Merge all the parts into the one image.
```

4.1. *Proposed K-means selection algorithm*

In order to make the object detection easier, a new approach is proposed. Before filtering, it is possible to use K-means algorithm [18]. This idea relies on isolating similar parts of the image and saving them in different files. After that operation, Laplacian filter can be applied on each new image. K-means is a well-known and widely used algorithm. Let S be a set of m-dimensional points$x_i = \{x_1, x_2, \dots, x_m\}$, where $i \in \{1, \dots, n\}$, n is the number of points in S. The main aim is a division of points in S on K disjoint sets called clusters. The number of clusters is fixed before the start of calculations. During the first step of the algorithm one has to choose the initial centers of clusters (for instance randomly). Then, the distances between points and clusters are calculated according to the following pattern:

$$d(x_i, c_j) = \|\vec{x_i} - \vec{c_j}\|$$

where$\vec{c_j}$ is the center of j-th center, $j \in \{1, \dots, K\}$,$\|\cdot\|$is a metric (for instance Euclidean, Manhattan or Mahalanobis distance). Each point is assigned to the cluster which is located nearest to the calculated point. The next step relies on calculating new centers of K clusters ($\vec{c_j}, j \in \{1, \dots, K\}$)

$$c_j = \frac{1}{l}(\vec{x_1} + \vec{x_2} + \dots + \vec{x_l})$$

where l is a number of points assigned to j-th cluster. Afterwards, new distances between points and clusters are calculated, until the stabilization (i.e. the situation in which all points are staying in the assigned subsets) or the fixed number of iterations. Thanks to this, the value of inertia is minimized:

$$I = \sum_{j=1}^{K} \sum_{x_i \in c_j} \|\overrightarrow{x_i} - \overrightarrow{c_j}\|^2$$

5. Experimental results

As was said before, application of three Laplacian filters was tested. In all cases, 100 measurements were performed and the results were averaged. The investigated image is 1200 pixels wide and 676 pixels high. For each convolution mask the image was filtered by using 1, 2, 4, 8, 16 and 32 threads. The algorithm was implemented in C\# language. The testing parallel architecture was Quad-Core AMD Opteron 8356 8p (32 CPU threads). Detailed results of experiments are presented in Tab. 1 and shown in Fig. 5 - 7. On all graphs the horizontal axis represents the number of threads and the vertical axis represents time.

It is possible to observe that even the division of the image into 2 rectangles speeds up significantly the calculations (53\%-57\% of the calculation time for one thread). In the case of the 9 on 9 convolution mask, the time was decreased from 141 to 75 seconds. Fig. 5 - 7 show a hyperbolic decline of the computing time. The differences between consecutive cases are getting smaller but still the most beneficial application is while using 32 threads (the maximum number of available CPU threads). The larger the convolution mask is, the more time can be saved thanks to multithreading (7-10 seconds in the case of the 3 on 3 mask, 22-42 seconds in the matter of the 5 on 5 mask and 66-131 seconds regarding to the 9 on 9 mask). As we have proposed multithreading method can be a very useful tool in speeding up the calculations.

TABLE I
RESULTS OF IMAGE FILTERING BY USING LAPLACIAN FILTERS (100 AVERAGED MEASUREMENTS)

threads	average time (seconds)	percentage	standard deviation	average time (CPU ticks)	standard deviation
3 on 3 convolution mask					
1	17.87600	100%	0.23354	40153258	524569
2	10.18082	56.95%	0.30065	22868252	675319
4	6.03999	33.79%	0.28318	13567088	636091
8	3.84961	21.54%	0.25843	8647022	580477
16	2.93947	16.44%	0.21808	6602667	489848
32	2.42189	13.55%	0.08085	5440081	181596

5 on5 convolution mask					
1	46.05641	100%	1.08065	103452367	2427362
2	24.74928	53.74%	0.26110	55592085	586472
4	14.43128	31.33%	0.26109	32415682	1912613
8	8.82084	19.15%	0.97053	19813459	2180018
16	6.05791	13.15%	0.49399	13607337	1109609
32	4.32978	9.40%	0.15660	9725596	351746
9 on9 convolution mask					
1	141.72927	100%	1.55923	318353710	3502354
2	75.59142	53.34%	1.27017	169794198	2853071
4	43.08413	30.40%	4.14568	96776003	9312067
8	25.00103	17.64%	2.58287	56157567	5801661
16	17.28079	12.19%	1.61762	38816286	3633518
32	10.67084	7.53%	0.51039	23968946	1146444

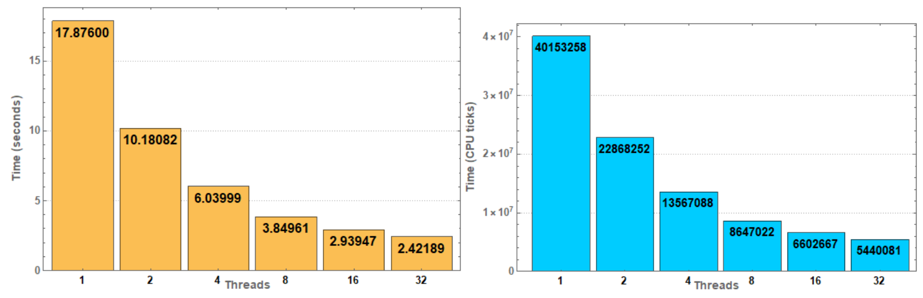

Fig. 5. Results for 3 on 3 convolution mask.

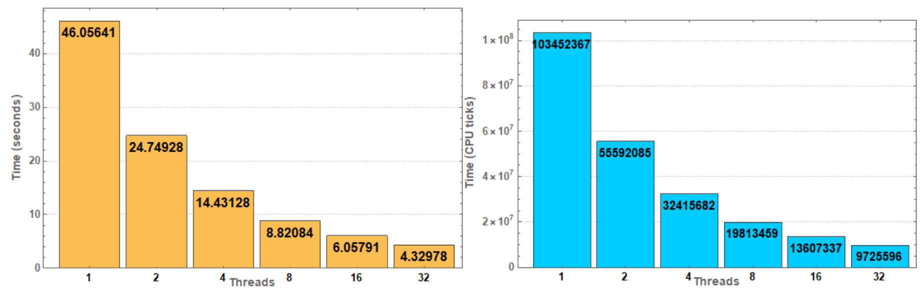

Fig. 5. Results for 5on5 convolution mask.

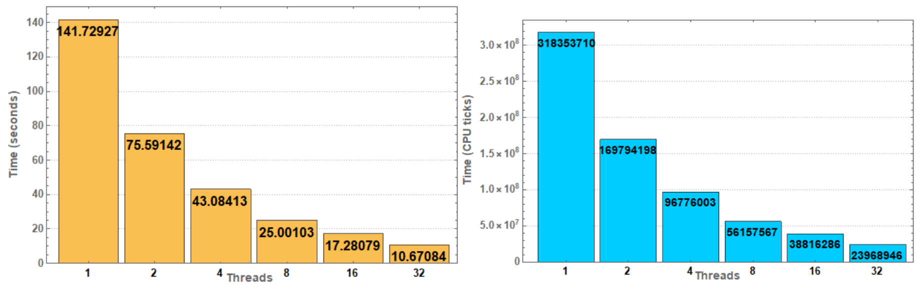

Fig. 5. Results for 9on9 convolution mask.

5.1. *K-means selection algorithm results*

The K-means algorithm is used to preliminary separating the objects from the background. During the experiments Euclidean metric was applied and initial centers were generated randomly. Calculations were performed by using scikit-learn library [19] (Python language) on the same image as before. The choice of the optimal number of clusters was based on the inertia value and the elbow criterion, what is presented in Fig. 8.

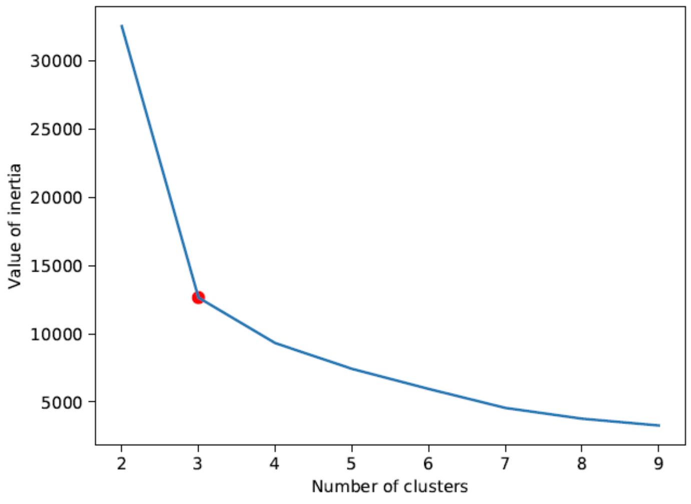

Fig. 8. The elbow criterion for choice the right number of clusters.

What is visible, the best number is 3. Of course, this value can be also selected experimentally. Calculating the silhouette (other criterion helpful in finding the appropriate number of clusters), is computationally expensive what is unacceptable for greater images like the presented one. Fig. 9-11 confirm that this approach for objects detection can be very useful.

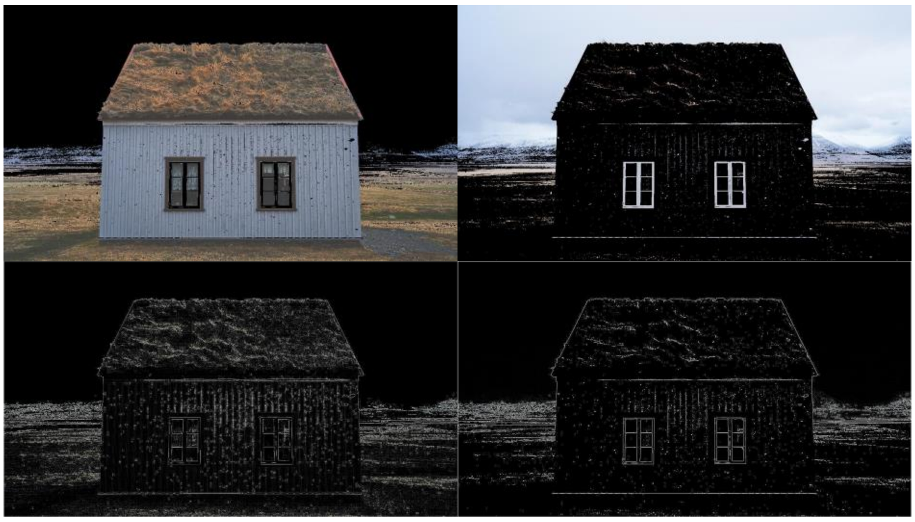

Fig. 9. The results for 2 clusters.

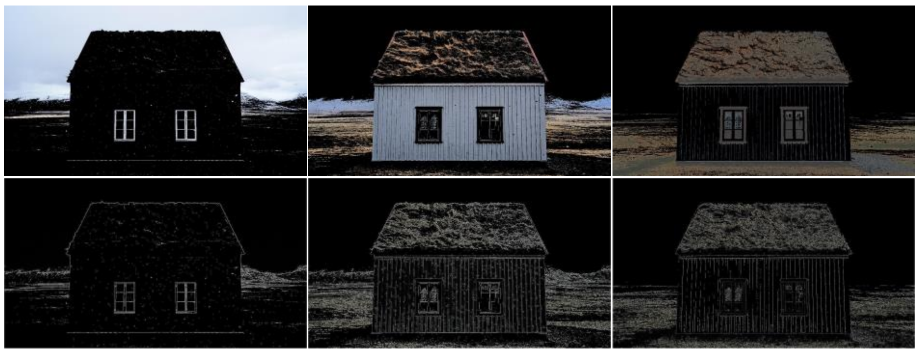

Fig. 10. The results for 3 clusters.

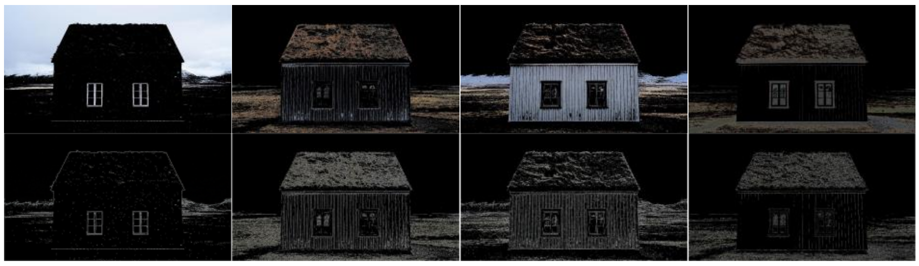

Fig. 11. The results for 4 clusters.

Already application of K-means algorithm gave promising results - the home visible on the original image was separated from the background. Output of the K-means is an input for the filtering by using the Laplace matrix (in this case, the size of 3 on 3). The edges of the object are more visible after K-means and filtering than just after application of Laplacian. This achievement can be helpful in feature recognition and shapes detection.

Of course, the optimal number of clusters is dependent on the complexity of the image. In the future, it is also possible to parallelize K-means algorithm with filtering what helps in saving some time necessary to computations.

6. Conclusions and final remarks

The research has shown that proposed parallelization significantly decreases the time of calculations. The profit is best visible when all CPU threads are used. The larger the picture and the computation mask is, the more important reduction in time spent for calculations is visible. It is worth to mention that the time necessary to filter the image of a size 1200 on 676 pixels by using only one thread and the 9 on 9 convolution mask exceeds two minutes. It can be concluded that filtering larger image (for instance of a size 6000 on 6000 pixels or even higher) involves several minutes, especially in the case of large convolution masks, also bigger than 9 on 9. Striving for a significant reduction of the calculation time is essential.

The method can be further developed by examining other parallel algorithms or trying to process many images at the same time. It would be very interesting to see how GPU acceleration methods like CUDA or OpenCL influence on total time computations. Nowadays, it is being created more and more high resolution photographs so it is essential to find efficient image processing algorithms. Therefore it is worth taking advantages of available computing power (for instance by using not only CPU but also a graphic card). It should enable to reduce calculations time what is especially important when a large number of high resolution images need to be analyzed. In our future research we will also investigate this methodology in movie processing, since this application can be the most important for HD multimedia systems.

Acknowledgment

Authors acknowledge contribution to this project from the program "Best of the Best 3.0" from the Polish Ministry of Science and Higher Education.

References

1. Z. Marszałek, "Parallel fast sort algorithm for secure multiparty computation" J. UCS, vol. 24, no. 4, pp. 488–514, 2018.
2. Z. Marszalek, "Parallelization of modified merge sort algorithm" Symmetry, vol. 9, no. 9, p. 176, 2017, DOI: 10.3390/sym9090176.
3. J. Protasiewicz, "Inventorum: A platform for open innovation," inSystems, Man, and Cybernetics (SMC), 2017 IEEE International Conferenceon. IEEE, 2017, pp. 10–15.
4. D. D. Burdescu, L. Stanescu, M. Brezovan, F. Slabu, and D. Ebanca, "Multimedia data for efficient detection of visual objects," in Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, ser. IMCOM '17. New York, NY, USA: ACM, 2017, pp. 61:1–61:8, DOI: 10.1145/3022227.3022287.
5. D. D. Burdescu, M. Brezovan, L. St˘anescu, C. S. Spahiu, and D. C. Ebânc˘a, "Graph-based semantic segmentation for 3d digital images," in 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2017, pp. 114–119, DOI: 10.1109/WAINA.2017.69.

6. Y. Jia, C. Rong, C. Wu, and Y. Yang, “Research on the decomposition and fusion method for the infrared and visible images based on the guided image filtering and gaussian filter,” in 2017 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 1797–1802, DOI: 10.1109/CompComm.2017.8322849.
7. S. Deniziak and T. Michno, “New content based image retrieval database structure using query by approximate shapes,” in Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3-6, 2017.,2017, pp. 613–621, DOI: 10.15439/2017F457.
8. S. Deniziak and T. Michno, “Query-by-shape interface for content based image retrieval,” in 8th International Conference on Human System Interaction, HSI 2015, Warsaw, Poland, June 25-27, 2015, 2015, pp. 108–114, DOI: 10.1109/HSI.2015.7170652.
9. R. Karumuri and S. A. Kumari, “Weighted guided image filtering for image enhancement,” in 2017 2nd International Conference on Communication and Electronics Systems (ICCES), 2017, pp. 545–548, DOI: 10.1109/CESYS.2017.8321137.
10. N. Dhengre, K. P. Upla, H. Patel, and V. M. Chudasama, “Biomedical image fusion based on phase-congruency and guided filter,” in 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017, pp. 1–5, DOI: 10.1109/ICIIP.2017.8313792.
11. C. Chen and M. C. Stamm, “Image filter identification using demosaicing residual features,” in 2017 IEEE International Conference on Image Processing (ICIP), 2017, pp. 4103–4107, DOI: 10.1109/ICIP.2017.8297054.
12. S. K. Dewangan, “Visual quality restoration enhancement of underwater images using hsv filter analysis,” in 2017 International Conference on Trends in Electronics and Informatics (ICEI), 2017, pp. 766–772, DOI: 10.1109/ICOEI.2017.8300807.
13. E. Royer, J. Chazalon, M. Rusiñol, and F. Bouchara, “Benchmarking keypoint filtering approaches for document image matching,” in 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 01, 2017, pp. 343–348, DOI: 10.1109/ICDAR.2017.64.
14. R. Jain, R. Kasturi, and B. G. Schunck, Machine vision. McGraw-HillNew York, 1995, vol. 5.
15. R. C. Gonzalez and R. E. Woods, Digital image processing. Upper Saddle River, NJ: Prentice Hall, 2012.
16. Z. Czech, Wprowadzenie do obliczen równoległych. Wydawnictwo Naukowe PWN, 2013.
17. T. Zielinski, Cyfrowe przetwarzanie sygnałów: od teorii do zastosowan. WydawnictwaKomunikacjiiŁacznosci, 2007.
18 A.K. Jain, Data clustering: 50 years beyond K-Means, Pattern recognition letters 31.8, 2010, pp. 651-666.
19. F. Pedregosa, G. Varoquaux et al., Scikit-learn: Machine learning inPython, Journal of Machine Learning Research, vol. 12, 2011, pp. 2825-2830.
20. K. Książek, Z. Marszałek, G. Capizzi, C. Napoli, D. Połap, M. Woźniak : The impact of parallel programming on faster image filtering. Federated Conference on Computer Science and Information Systems – FedCSIS 2018, September 9-12 Poznan, Poland 2018, pp. 545-550.