Comparison of the Efficiency of Parallel Algorithms KNN and NLM Based on CUDA for Large Image Processing

Lesia Mochurad [a], Roman Bliakhar [a]

[a] *Lviv Polytechnic National University, 12 Bandera street, Lviv, 79013, Ukraine*

Abstract

Digital image processing is widely used in various fields of science, such as medicine – X-ray analysis, magnetic resonance imaging, computed tomography, cosmology – collecting information from satellites, their transmission and analysis. Image noise accompanies any of the stages of image processing – from obtaining them to segmentation and object recognition. In order to process large images in real time, the paper proposes to use CUDA technology to parallelize KNN and NLM algorithms. The subject area of the satellite images is selected for noise reduction. A comparative analysis of the effectiveness of the proposed approach. It is investigated how the computation time of successive versions of algorithms changes with increasing the size of the image itself. Based on a series of numerical experiments, it was possible to achieve an acceleration of 40 times using CUDA technology. It is shown that the calculations on the CPU significantly exceed the time spent on execution compared to the GPU. This is due to the fact that in tasks of this type, several threads on the CPU are not able to compete with thousands of threads of the graphics core. We can also observe that as the number of GPU threads increases, the time decreases significantly. This study is especially relevant in current trends in video cards. that in tasks of this type, several threads on the CPU are not able to compete with thousands of threads of the graphics core. We can also observe that as the number of GPU threads increases, the time decreases significantly. This study is especially relevant in current trends in video cards. that in tasks of this type, several threads on the CPU are not able to compete with thousands of threads of the graphics core. We can also observe that as the number of GPU threads increases, the time decreases significantly. This study is especially relevant in current trends in video cards.

Keywords

Computer vision, noise reduction, graphics processor, computational process optimization, acceleration.

1. Introduction

Most images are affected by various types of noise caused by equipment failure, problems with data transmission or compression, and natural factors. Therefore, the first stage of image processing is filtering. The presence of noise in the image can cause inaccuracies and distortions in the segmentation and recognition phase. For example, the system may perceive noise for individual objects, which, in turn, will negatively affect further research. That is why noise removal is an important problem in the field of image processing, which has many applications in various fields, such as medicine (magnetic resonance imaging, computed tomography), industry, military applications, space research, communications (satellite television) [1-3].

Most often, noise reduction is used to improve visual perception, but can also be used for specialized purposes – for example, in medicine to increase the clarity of images on X-rays [4], in pre-processing images for further recognition and more. In addition, noise reduction plays an important role in compressing video and images.

CMIS-2022: The Fifth International Workshop on Computer Modeling and Intelligent Systems, Zaporizhzhia, Ukraine, May 12, 2022
EMAIL: lesia.i.mochurad@lpnu.ua (L. Mochurad); bliakharr@gmail.ua (R. Bliakhar)
ORCID: 0000-0002-4957-1512 (L. Mochurad); 0000-0001-8478-5308 (R. Bliakhar)

Despite the fact that noise reduction is a classic problem and has been studied for a long time, this task is still difficult and open. The main reason is that from a mathematical point of view, noise reduction is an inverse problem, so its solution is not unique. As a result, there are many different methods, the advantages and disadvantages of which will be discussed and analyzed in the next section.

Today there is a relevant area of image processing from satellites [5]. They are used by both cartographers and private companies to analyze land data. Since these images are used in further analysis by artificial intelligence methods, in particular such as computer vision, an important step is part of their pre-processing. This process often involves reducing noise in the image, which could be caused by various interferences, and therefore lose some information. That is why the subject area of satellite images is chosen for this work, because these images are large in size, and therefore require large computing power. Therefore, in this case it will be effective to take advantage of parallel calculations.

Based on the above, we can formulate the purpose of this work, namely, to conduct a comparative analysis of the effectiveness of parallel algorithms KNN and NLM based on CUDA technology to accelerate the noise of large-scale images obtained from satellite.

2. Analysis of literature sources

Significant results in the field of image noise reduction have been achieved in recent decades. This is due to the wide range of applications in everyday life, as we are surrounded by more and more digital devices: smartphones, webcams, surveillance cameras, drones and more. Accordingly, this problem occurs in such areas as: satellite television, computed tomography [4], magnetic resonance imaging, medicine in general, and in the field of research, which is actually the basis of many applications, such as object recognition and technology, such as geographic information systems, astronomy and many others [2].

Thus, image noise has remained a pressing issue for many researchers in recent times. To date, many classical filtration algorithms have been proposed (Median Filtering, Linear Filtering, Anisotropic Filtering) [3, 6, 7]. Median, linear, anisotropic and other classical filtering algorithms are usually able to remove noise, but have disadvantages, because their use loses a lot of information and suffers image geometry, resulting in objects in the image may take a completely different shape.

All linear filtering algorithms are optimal for Gaussian distribution of signals, interference and observed data, and lead to the smoothing of sharp differences in the brightness of the processed images. However, real images, strictly speaking, are usually not subject to this probability distribution.

Some of these problems are solved using non-parametric image processing methods. Studies have shown that good results in image processing were obtained using median filtering [8]. Note that the median filtering is a heuristic method of processing, its algorithm is not a mathematical solution of a strictly formulated problem. In the last two decades, nonlinear algorithms based on ranking statistics for the recovery of images damaged by various types of noise have been actively developed in digital image processing.

One of the algorithms without the above disadvantages is Non-Local Means (NLM) [8]. Because the principle is simple and straightforward, and the information contained in the image is better stored, NLM has become a popular noise reduction algorithm in recent years. Although NLM has achieved good results, there are some areas for improvement [9]. Mainly in the following aspects: first, although good results are achieved, the efficiency of the algorithm is slightly lower than traditional algorithms; secondly, it is difficult to determine the weight of similarity between image blocks, which in turn affects the efficiency of noise reduction [10].

Another effective algorithm, in terms of reducing noise and maintaining the informativeness of the original image, is K-Nearest Neighbors [11]. This algorithm is conceptually similar to the above NLM, but, as indicated in studies, with lower computational complexity [12]. Therefore, in theory, it should work faster than NLM, but the result of noise reduction should be compared in more detail.

So let's highlight the problem statement for our study:

- Develop a parallel algorithm of KNN and NLM methods based on CUDA technology and implement them programmatically.
- Test and compare the noise reduction results for each algorithm, as well as compare both algorithms with each other.
- Measure the execution time of each algorithm with a different configuration of the computer system for further calculation of acceleration and comparative analysis of efficiency.

3. Methods and tools

Mathematically, the problem of image noise reduction can be modeled as follows: $y = x + n$, where y – noisy input image, x – unknown clear image without noise, and n – is an additive of white Gaussian noise with standard deviation σn, which can be estimated in practice by various methods, such as the median of absolute deviations (MAD) [**Ошибка! Источник ссылки не найден.**3] or the principal component algorithm (PCA) [**Ошибка! Источник ссылки не найден.**4].

The purpose of image noise reduction is to reduce distortion while minimizing the loss of original features of the input image and increase the signal-to-noise ratio (SNR) [15].

The following main challenges can be identified for this task:

- flat areas of the image should be smooth;
- the edges must be protected from erosion;
- textures must be preserved;
- new distortions and artifacts should not be formed.

The principle of the first methods of noise reduction of images was quite simple: the color of the pixel was replaced by the average color of neighboring pixels. The law of variance in probability theory ensures that if nine pixels are averaged, the standard deviation of the mean noise is divided by three [8]. Thus, if we can find for each pixel nine other pixels in the image with the same color (before distortion due to noise), we can divide the noise into three (or four with 16 similar pixels, etc.) [9].

In fact, the most similar pixels will not always be the closest, on the contrary almost never. For example, you can give images with periodic patterns (when a certain fragment may occur several times). So the solution is to scan a huge part of the image for all the pixels that really resemble the pixel we want to replace (mute). The noise reduction task is then performed by calculating the average color of the most similar pixels found. This results in much greater clarity after filtering and less loss of image detail compared to average local algorithms. This algorithm is called non-local means and its definition can be written as follows:

Let Ω – image area, and p and q are two dots (pixels) within the image, then:

$$NLu(p) = \frac{1}{C(p)} \int_{\Omega} f\left(d\big(B(p), B(q)\big)\right) u(q),$$

where $u(p)$ is the filtered value of the image at the point p; $u(q)$ is the unfiltered value of the image at the point q; $C(p)$ – this is the normalizing factor; f is a weighing function in which $d(B(p), B(q))$ is the Euclidean distance between image patches focused on p and q in accordance.

When calculating the Euclidean distance $d(B(p), B(q))$ all pixels in the patch $B(q)$ have the same weight (importance), so the function $f(d\big(B(p), B(q)\big))$ can be used to mute all pixels in the patch $B(p)$ and not just for p.

This algorithm can be performed in two implementations: pixelwise and patchwise. In our work we use pixel implementation, because in fact there is not much difference between the two options, so the overall quality in terms of saving details is not improved with the use of patchwise implementation.

Let's introduce the noise reduction of the color image $u = (u_1, u_2, u_3)$ and a specific pixel p as follows: $\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q) w(p,q)$, $C(p) = \sum_{q \in B(p,q)} w(p,q)$, where $i = 1, 2, 3$ and $B(p,r)$ denote the environment centered at the point p with dimensions $(2r+1) \times (2r+1)$ pixels. The search area is limited by a square circumference of a fixed size, which is due to the limitation of

calculations. This is a window 21×21 for small and medium values σ. The window size increases to 35×35 for large values σ due to the need to detect more similar pixels to reduce additional noise.

Weight $w(p,q)$ will depend on the Euclidean distance $d^2 = d^2(B(p,f), B(q,f))$ for $(2r+1) \times (2r+1)$ color patches centered accordingly p and q:

$$d^2\big(B(p,f), B(q,f)\big) = \frac{1}{3(2f+1^2)} \sum_{i=1}^{3} \sum\nolimits_{j \in B(0,f)} (u_i(p+j) - u_i(q+j))^2.$$

That is, each pixel value is restored as the average of the most similar pixels, where this similarity is calculated in the color image. Therefore, for each pixel, each channel value is the result of the average of the same pixel [15].

To calculate the weight $w(p,q)$ we will use the exponential function: $w(p,q) = e^{-\frac{\max(d^2 - 2\sigma^2,\ 0.0)}{h^2}}$, where σ indicates the standard deviation of the noise and h is the filtering parameter set relative to the value σ [12, 15]. The weight of the function is set to average similar noise patches. That is, patches with square distances less than $2\sigma^2$ are installed to 1, while larger distances decrease faster due to the exponential function.

The weight of the reference pixel p on average, is set as the maximum weight in the vicinity $B(p,r)$. This parameter avoids excessive weighing of the reference point in the middle. Otherwise $w(p,q)$ should be equal to 1 and as a result, more value will be required to reduce noise h. Therefore, using the above procedure, we will be able to restore the noiseless value for each pixel p.

To objectively assess the basic metrics of our noise reduction using the NLM algorithm, consider an additional algorithm – KNN [**Ошибка! Источник ссылки не найден.**6]. The main principle of KNN is that the category of the data point is determined according to the classification of the nearest K neighbors. The KNN (k-Nearest Neighbors) filter was developed to suppress white noise, which is based on the Gaussian Blur Filter algorithm [17]. Consider how it works.

Let $u(x)$ – the value of the input noisy image at the point x, a $f(y)$ – the result obtained by the KNN filter with parameters h and r at some point y. Ω is the space of a given size around the selected pixel, i.e. it is a block of pixels in size $N \times$ N, so that the selected pixel (dot y) is located in the center of this block. Then the filtration formula can be represented as: $f(y) = \frac{1}{C(x)} \sum_{\Omega(x)} u(y)\, e^{-\left(\frac{(y-x)^2}{r^2}\right)} e^{\frac{(u(y)-u(x))^2}{h^2}}$, where $C(x)$ – normalizing factor.

Since the purpose of our study is to accelerate the noise reduction of images using the GPU, for such purposes we will use CUDA [**Ошибка! Источник ссылки не найден.**]. Is a software-hardware architecture of parallel computing, which allows you to significantly increase computing performance through the use of graphics processors from Nvidia. CUDA allows you to reduce the load on the processor, which gives significant benefits over time. Given the power of modern video cards, we can get a significant acceleration in the calculations. And because we will be working with images, this technology should be ideal for this type of task.

Both methods can be easily implemented using CUDA technology. Since this technology uses the SIMD parallel computing model [19], we do not need to parallelize the algorithm itself (NLM, KNN). It is enough to implement a standard version of the algorithm in the form of a kernel function. The kernel function is actually a normal sequential program, without organizing the launch of threads. Instead, the GPU runs many copies of a given function in different threads.

Consider in more detail the mechanism for organizing parallel calculations:

- Host – CPU. Performs a control role – runs tasks on the device, allocates memory on the device, moves memory to/from the device.
- Device – GPU. Performs the role of "subordinate" – does only what the CPU tells him.
- Kernel – a task (NLM, KNN), which runs the host on the device.

Threads are grouped into blocks, in the middle of which they have shared memory and run synchronously. The blocks are combined into grids, the main task of which is to serve as an isolated container to which the kernel function applies (a separate function can be applied to each grid). As part of our problem, the calculation creates three grids for a color image, and one for black and white (because algorithms split images into RGB color vector). The number of threads is equal 2^n, where $1 < n \leq 10$ ($2^{10} = 1024$, the maximum number of threads per block). The input noisy image is

divided by k blocks where $k_{ij} = n_i \times n_j$, $i \in \frac{N}{n}$, $j \in \frac{M}{n}$, where N – width, a M – image height, respectively. Everyone k_{ij} the grid block corresponds k_{ij} an image block, where for each pixel of the image the weight in a separate stream is calculated.

To conduct experiments, it was decided to develop a software product in the Python programming language. Accordingly, the PyCuda library was used to use the CUDA parallel computing architecture. As well as an OpenCV library for image processing. All calculations were performed in Google Colab [20],which allows you to write and execute Python code in a browser with a minimum of settings and free access to powerful computing capabilities. The configuration includes a graphics professor NVIDIA Tesla T4 with 2560 CUDA cores and 16 Gb of video memory and an Intel Xeon CPU with a clock speed of 2.20Ghz with two cores and two threads.

4. Results of numerical experiments

First, we evaluate the quality of both algorithms and evaluate the effect of noise reduction of various images. First we test the work of KNN.

Figure 1: Comparison of 1 image fragment before (left) and after (right) noise reduction using KNN algorithm

Figure 2: Comparison of 5 fragments of the image before (left) and after (right) noise reduction using the KNN algorithm

Now let's perform the same tests for the NLM algorithm.

Figure 3: Comparison of 5 fragments of the image before (left) and after (right) noise reduction using the NLM algorithm

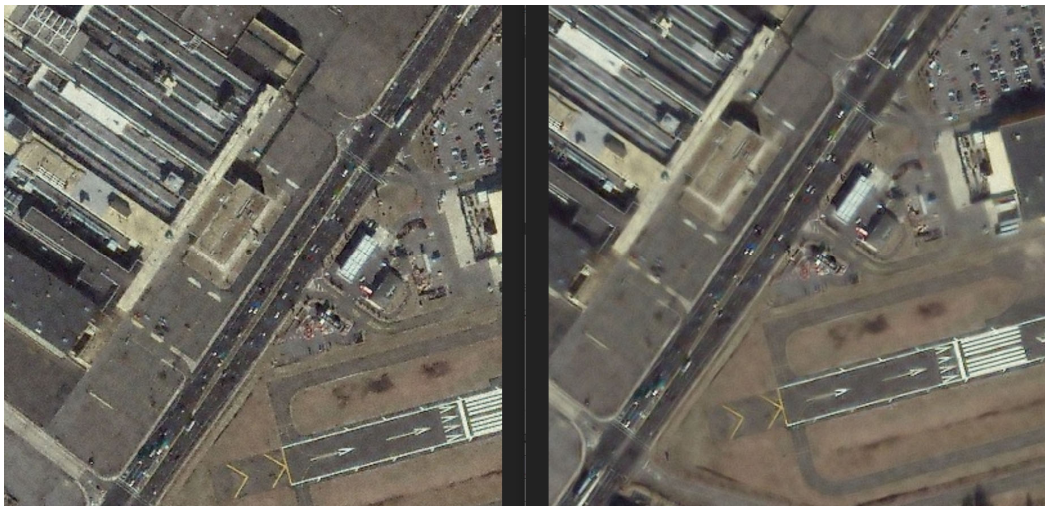

Figure 4: Comparison of 6 fragments of the image after noise reduction using KNN (left) and NLM (right)

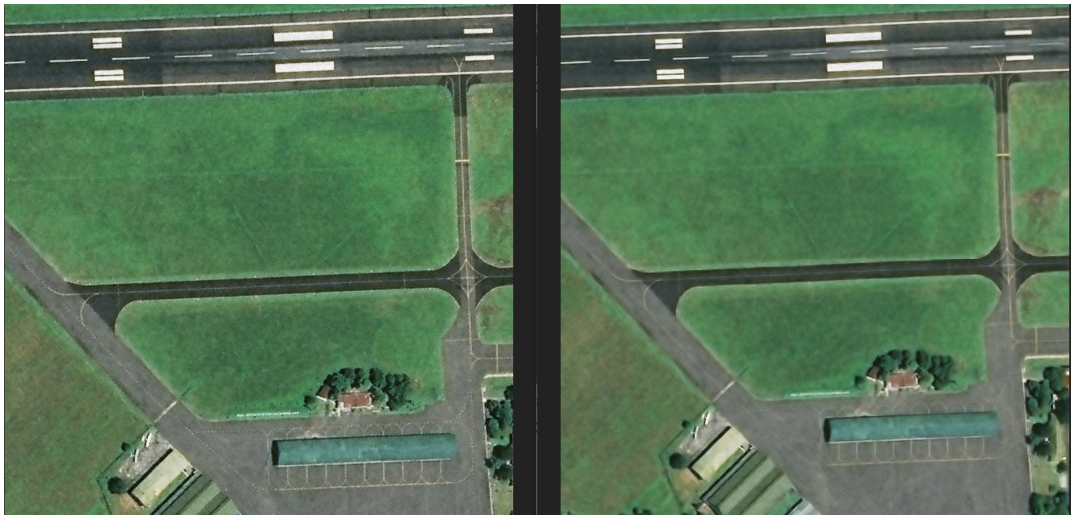

Figure 5: Comparison of 8 fragments of the image after noise reduction using KNN (left) and NLM (right)

In addition, studies were performed on images of different dimensions and with different configurations of computing power to measure the runtime of algorithms with CPU/GPU comparison. The calculations were performed for images of different dimensions, where P – the number of pixels in the image $n \times m$ pixels, and for GPUs with different number of threads per block (4, 16, 64, 256 and 1024 threads, respectively).

Table 1

Program execution time in seconds for KNN algorithm with different CPU/GPU startup configuration

P	CPU	GPU 4 tpb	GPU 16 tpb	GPU 64 tpb	GPU 256 tpb	GPU 1024 tpb
175k	0.056724	0.009772	0.005263	0.005370	0.005531	0.004922
700k	0.204899	0.030974	0.014606	0.011442	0.013098	0.011224
3kk	0.799452	0.102840	0.048849	0.038600	0.037616	0.033931
11kk	3.366155	0.405643	0.144978	0.118630	0.124715	0.121243
45kk	12.349476	1.169624	0.490328	0.418505	0.433192	0.396563
81kk	21.875035	2.154644	0.869768	0.721381	0.657146	0.594174

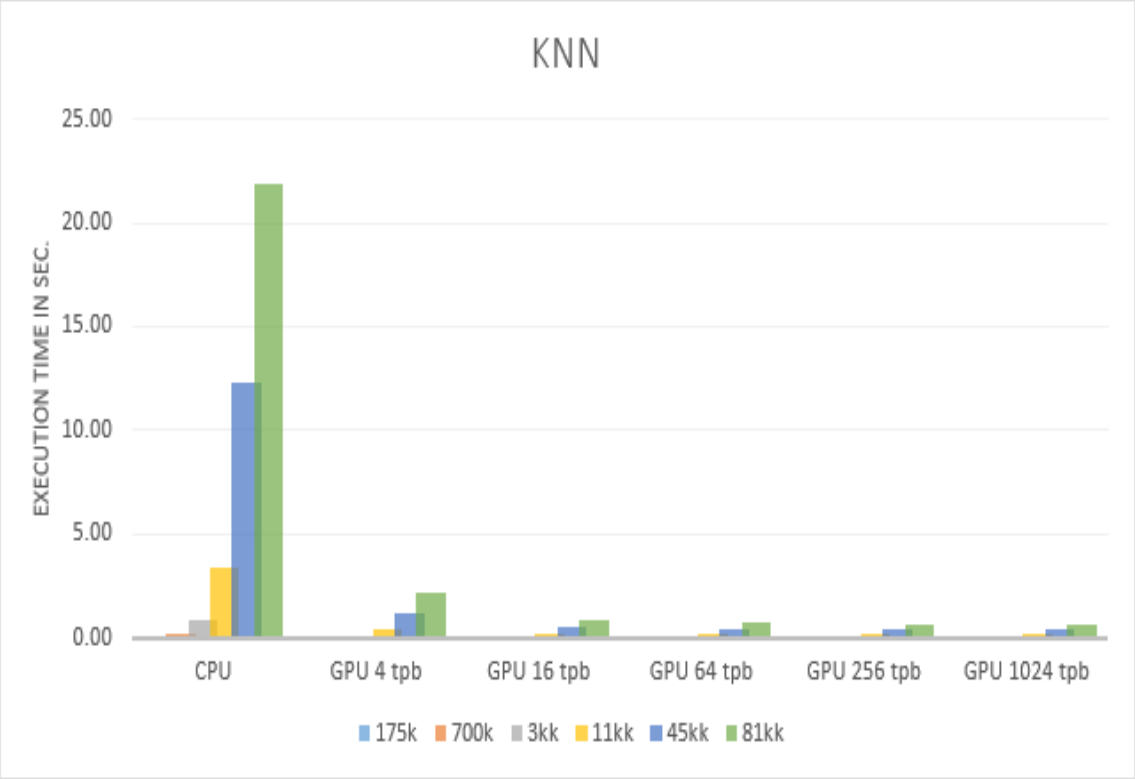

Figure 6: Execution time of parallel KNN algorithm using CPU/GPU in seconds for different image dimensions and with different program startup configuration

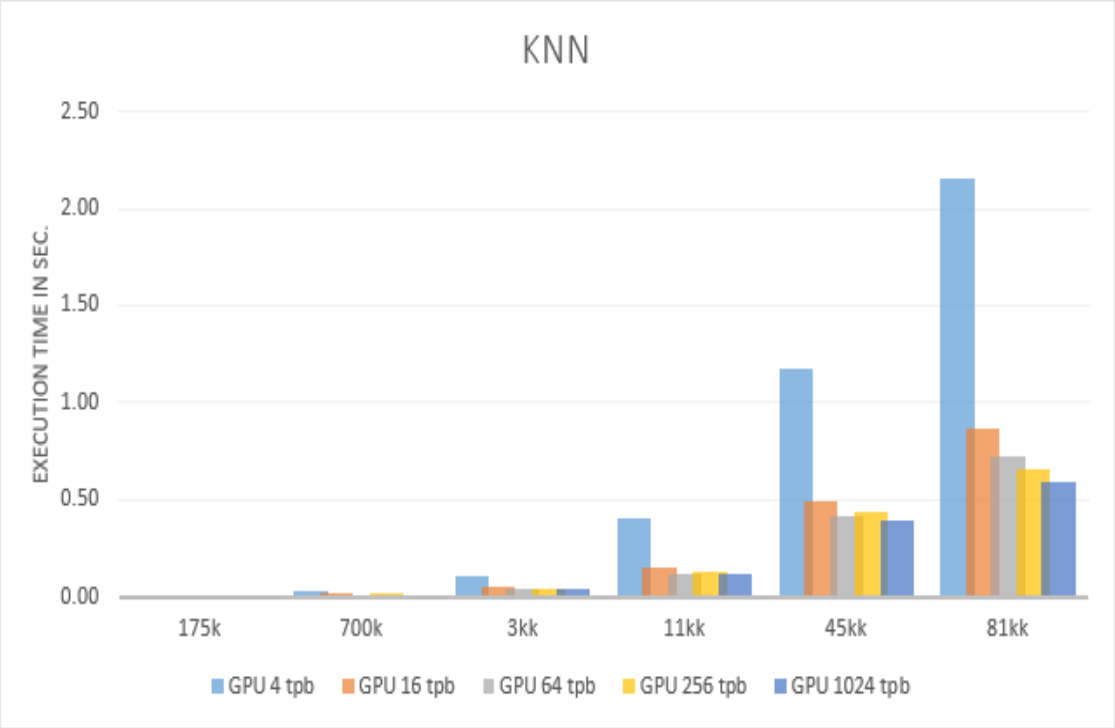

Figure 7: Comparison of runtime results in seconds for KNN algorithm executed using GPU

Table 2
NLM algorithm execution time in seconds with different CPU/GPU startup configuration

P	CPU	GPU 4 tpb	GPU 16 tpb	GPU 64 tpb	GPU 256 tpb	GPU 1024 tpb
175k	0.062403	0.023235	0.009618	0.007488	0.005829	0.004538
700k	0.230475	0.083275	0.028574	0.017382	0.010574	0.006732
3kk	0.853824	0.316450	0.066979	0.047274	0.033365	0.023549
11kk	3.264484	0.800226	0.204644	0.157260	0.120847	0.087866
45kk	13.059060	2.331046	0.811966	0.607054	0.453854	0.339316
81kk	23.748522	3.996524	1.333901	1.009624	0.720390	0.543901

We calculate the acceleration for the measurements of each algorithm. To do this, divide the execution time by CPU by the execution time by GPU.

Table 3
Acceleration factor for the KNN algorithm performed using the GPU

P	GPU 4 tpb	GPU 16 tpb	GPU 64 tpb	GPU 256 tpb	GPU 1024 tpb
175k	5.804566	10.777158	10.562708	10.255388	11.524214
700k	6.615167	14.028813	17.907669	15.643454	18.254952
3kk	7.773751	16.365716	20.711111	21.252815	23.561176
11kk	8.298323	23.218389	28.375263	26.990804	27.763618
45kk	10.558499	25.186156	29.508552	28.508098	31.141290
81kk	10.152507	25.150428	30.323849	33.287953	36.815880

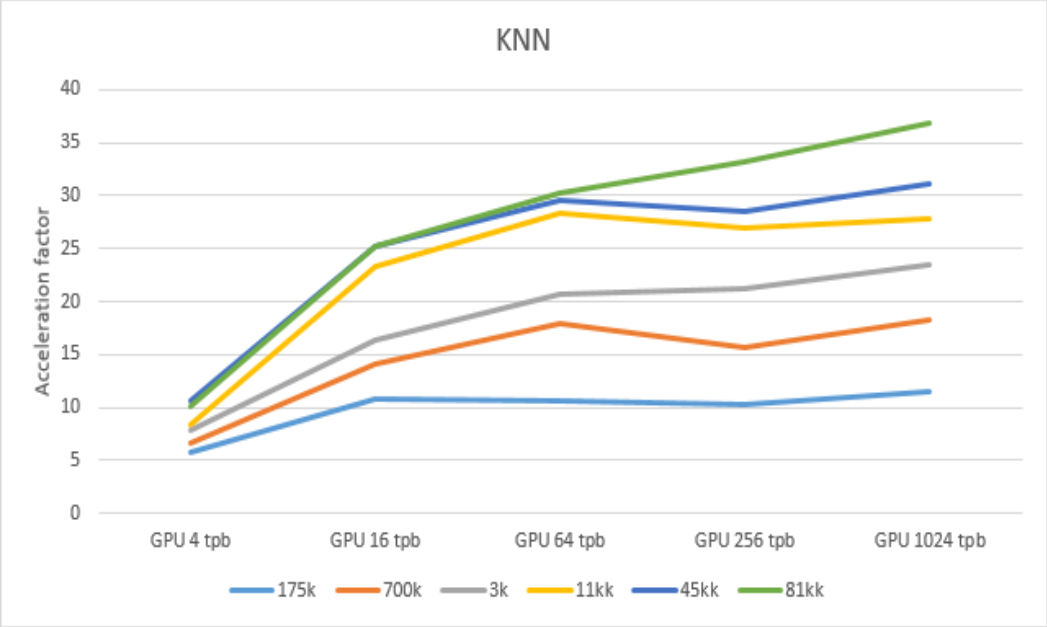

Figure 8: Dependence of the acceleration factor on the GPU on the number of threads per block at different image dimensions for the KNN algorithm

Table 4
Acceleration factor for the NLM algorithm performed using the GPU

P	GPU 4 tpb	GPU 16 tpb	GPU 64 tpb	GPU 256 tpb	GPU 1024 tpb
175k	2.685736	6.488113	8.334059	10.705201	13.750961
700k	2.767646	8.065955	13.259593	21.797397	34.235835
3kk	2.698137	12.747567	18.061340	25.590138	36.257286
11kk	4.079453	15.952002	20.758530	27.013322	37.153126
45kk	5.602232	16.083252	21.512204	28.773715	38.486371
81kk	5.942295	17.803809	23.522147	34.966196	43.663300

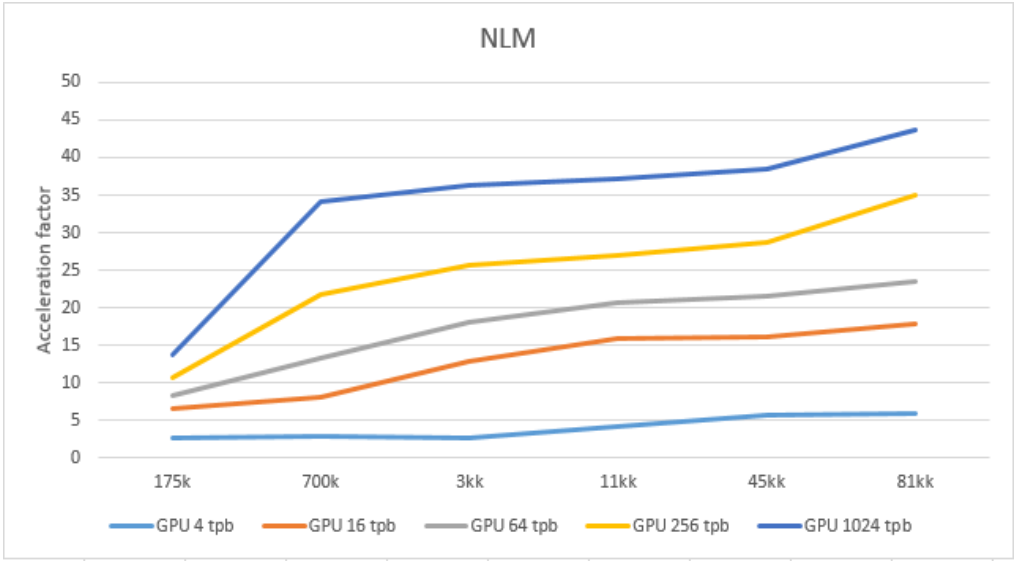

Figure 9: Dependences of the acceleration factor on the GPU on the number of threads per block at different image dimensions for the NLM algorithm

5. Discussion

Now let's analyze our experiments. To begin with, let's summarize the results of the noise reduction using each of the NLM and KNN algorithms.

As we can see from the above Figures 1-3, the algorithms reduce noise well and restore the original image even with loud noise. For comparison, in Figure 2 using the KNN algorithm, the image fragment after clearing loses the clarity of the contours, the image becomes slightly pixelated. At the same time in Figure 3 algorithm NLM visually makes less noise, so the image is still quite blurred, but the boundaries of the contours are not as violated as when noise reduction using the KNN algorithm. However, this is just one example, so don't judge by just one piece.

In Figure 4 shows that the image obtained with NLM, less clear compared to KNN. This is best seen in detail when looking at cars parked in the parking lot in the upper right corner of the image. Similar results can be observed in Figure 5. The image obtained during the KNN algorithm is clearer and richer. An example is the road marking lines at the bottom of the image, which in the first case, retained their geometry, not to mention the image on the right, where the lines are almost invisible, the image obtained by NLM was "blurred".

A similar problem is mentioned in [21], where the authors emphasize that in high-detail images, after the noise reduction with NLM, the geometry of objects deteriorates.

The next step is to analyze our program execution time metrics and GPU acceleration factors. For clarity, tests were performed on different image sizes. Therefore, it is clear from Table 1 and Table 2, that the largest time difference between CPU/GPU we get for scale images. This confirms the correctness of the choice of subject area for our task of accelerating noise reduction.

Considering the visualization in Figure 6 and Figure 7, respectively, we can see that the calculations on the CPU significantly exceed the time spent on execution compared to the GPU. This is due to the need for a large number of parallel calculations for image processing. For tasks of this type, multiple threads on the CPU are not able to compete with thousands of graphics core threads. We can also observe that as the number of GPU threads increases, the time decreases significantly.

In Table 3 and Table 4 there is a tendency to increase the rate of acceleration with increasing number of flows per unit. This is because the more threads, the fewer blocks, and therefore less time spent on memory operations.

Comparing the acceleration rates of both algorithms. We can see that the parallel NLM algorithm loses with a small number of threads, but wins with the largest number of threads per block. We can also see that the KNN algorithm does not receive such a large increase in acceleration as the NLM algorithm when increasing the image size for noise reduction from 45 million pixels to 81. This is well observed in Figure 8 and Figure 9, where the acceleration coefficient graph for the NLM algorithm grows faster than the KNN. Although KNN at 4 threads gets twice as fast as NLM.

If we compare the time spent on both algorithms, we can conclude that there is no clear favorite, although KNN shows better results with a small number of threads per block, due to the peculiarity of the algorithms. This can affect the results if the calculations are performed using older generation video cards, where the maximum possible number of threads per unit is many times less. In the conditions of use of modern video cards NLM, with the maximum number of threads, is ahead of the competitor.

6. Conclusions

This paper compares KNN and NLM algorithms in the context of parallel computations to accelerate the noise reduction of large images. To do this, using the Python programming language, implemented two algorithms with versions for CPUs and GPUs using the CUDA architecture.

In the course of the work, the results of each algorithm were analyzed and it was found that both algorithms cope well with the task of noise reduction and recovery of the original images obtained as satellite images.

Comparative tables are also constructed for each algorithm with a different configuration of the program start, in which you can evaluate the obtained time measurements and calculated acceleration rates. These data were visualized on graphs, which allowed us to objectively compare the obtained

metrics. Regarding the obtained time metrics and acceleration metrics, we can conclude that parallel computing with CUDA on GPUs is many times better than computing on CPU. It is for these algorithms that the GPU and CUDA technology can achieve an acceleration of approximately 40 times. Thanks to these results, we can use algorithms such as KNN and NLM in real-time video processing. It will also save a lot of time on processing images obtained from telescopes and satellites, which are high resolution.

7. References

[1] L. Guo, H. Zhang, H. Zhai, Ch. Gong, B. Song, Sh. Tong, Zh. Cao, X. Xu, Research on image processing and intelligent recognition of space debris, Proceedings Volume 10988, Automatic Target Recognition XXIX;1098817 (2019). doi:10.1117/12.2525058.

[2] Unsupervised Denoising for Satellite Imagery using Wavelet Subband CycleGAN [Electronic resource]. – 2020. – Resource access mode:https://arxiv.org/pdf/2002.09847.pdf.

[3] L. Fan, F. Zhang, H. Fan, et al, Brief review of image denoising techniques, Vis. Comput. Ind. Biomed. Art 2, 7 (2019). doi:10.1186/s42492-019-0016-7.

[4] Sameera V. Mohd Sagheer, Sudhish N. George, A review on medical image denoising algorithms, Biomedical Signal Processing and Control, volume 61 (2020), 102036. doi:10.1016/j.bspc.2020.102036.

[5] A. Asokan, J. Anitha, M. Ciobanu, A. Gabor, A. Naaji, D. Hemanth, J. Image Processing Techniques for Analysis of Satellite Images for Historical Maps Classification — An Overview. Appl. Sci. (2020), 10, 4207. doi:10.3390/app10124207.

[6] C. He, K. Guo, H. Chen, An Improved Image Filtering Algorithm for Mixed Noise, Appl. Sci. (2021), 11, 10358. doi: 10.3390 / app112110358.

[7] S. Kushwaha, R. Singh, Performance Comparison of Different Despeckled Filters for Ultrasound Images, Biomed Pharmacol J., (2017), 10(2). doi:10.13005/bpj/1175.

[8] G. Shweta, Meenaksh, A Review and Comprehensive Comparison of Image Denoising Techniques, International Conference on Computing for Sustainable Global Development. (2014), P. 975.

[9] K. Chen, X. Lin, X. Hu, et al, An enhanced adaptive non-local means algorithm for Rician noise reduction in magnetic resonance brain images, BMC Med Imaging 20, 2 (2020). doi:10.1186/s12880-019-0407-4.

[10] X. Zhang, Two-step non-local means method for image denoising, Multidim Syst Sign Process (2021). doi:10.1007/s11045-021-00802-y.

[11] Haiyan Wang, Peidi Xu, Jinghua Zhao, Improved KNN Algorithm Based on Preprocessing of Center in Smart Cities, Complexity, vol. 2021, Article ID 5524388, 10 pages, (2021). doi:10.1155/2021/5524388.

[12] M. Colom, A. Buades, Analysis and Extension of the Ponomarenko et al. Method, Estimating a Noise Curve from a Single Image [Electronic resource], IPOL. (2013) Resource access mode:http://www.ipol.im/pub/art/2013/45//article_lr.pdf.

[13] Chunjie Wu, Yi Zhao & Zhaojun Wang, The median absolute deviations and their applications to shewhart $\bar{X}$ control charts, Communications in Statistics – Simulation and Computation, 31: 3, 425-442, (2002). doi:10.1081/SAC-120003850.

[14] Sidharth Mishra, Uttam Sarkar, Subhash Taraphderet al, Principal Component Analysis, International Journal of Livestock Research. (2017). doi:10.5455 / ijlr.20170415115235.

[15] Suarez-Perez Alex, Gabriel Gemma, Rebollo Beatriz et al, Quantification of Signal-to-Noise Ratio in Cerebral Cortex Recordings Using Flexible MEAs With Co-localized Platinum Black, Carbon Nanotubes, and Gold Electrodes. Frontiers in Neuroscience, 12. (2018). doi:10.3389 / fnins.2018.00862.

[16] Yanhui Guo,Siming Han,Ying Li,Cuifen Zhang,Yu Bai, K-Nearest Neighbor combined with guided filter for hyperspectral image classification, Procesia Computer Science, volume 129. (2018), pp. 159-165. doi:10.1016/j.procs.2018.03.066.

[17] P. Singhal, A. Verma and A. Garg, "A study in finding the effectiveness of Gaussian blur filter over bilateral filter in natural scenes for graph based image segmentation," 2017 4th International

Conference on Advanced Computing and Communication Systems (ICACCS). (2017), pp. 1-6. doi:10.1109 / ICACCS.2017.8014612.
[18] L. Mochurad, G. Shchur, Parallelization of Cryptographic Algorithm Based on Different Parallel Computing Technologies,. Proceedings of the Symposium on Information Technologies & Applied Sciences (IT&AS 2021). Bratislava, Slovak Republic, March 5. (2021), Vol-2824, 20-29 p.
[19] L. Mochurad, N. Boyko, Technologies of distributed systems and parallel computation: monograph. Lviv: Publishing House “Bona”, (2020), 261 p.
[20] Google Colaboratory [Electronic resource] – Resource access mode:https://research.google.com/colaboratory/faq.html.
[21] A. Buades, B. Coll, and J. Morel, On image denoising methods, Technical Report 2004-15, CMLA. (2004).