Algorithm of search and track of static and moving large-scale objects

Anatoly Kalyaev[1,a], Maxim Khisamutdinov[1]

[1]*SRI MCS SF, Taganrog, Russia*

Abstract. We suggest an algorithm for processing of a sequence, which contains images of search and track of static and moving large-scale objects. The possible software implementation of the algorithm, based on multithread CUDA processing, is suggested. Experimental analysis of the suggested algorithm implementation is performed.

1 Introduction

The problem of search and track of static and moving large-scale objects is needed in many areas of human activity such as astronomy, aviation, medicine, etc [1-7] (Fig 1,2). The problem implies processing of a single image or a sequence of images for detection coordinates of static and moving large-scale objects.

We propose new method based on calculation of the descriptors of the image frame using a graphic subsystem and comparing it with the preliminarily trained matrix of objects by means of an artificial neural network model.

Figure 1. Detecting and tracking of static large-scale objects.

The method of large-scale object search consists of 3 steps:

• preliminary processing of an image of a video sequence (conversion to monochromatic, reducing noise with the help of median filtering, increasing contrast, selection of visibility region);

• detection of the point features of the image (calculation of descriptors) (Fig. 3);

• comparison of the point features with a template (comparison of the detected point features of the image with template features of possible objects) (Fig. 4).

Figure 2. Detecting and tracking of moving large-scale objects.

[a] Corresponding author: anytroll@gmail.com

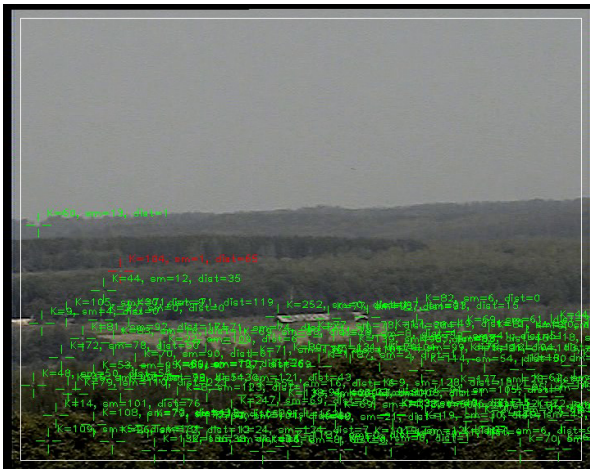

Figure 3. All points and parameters of potential targets.

Proposed algorithm is pretty hard to calculate that is why we decided to aim it at parallel computing system realization. For analysis of the suggested algorithm we have developed software implementation based on multithread loading of the CUDA-calculator.

Figure 4. Filtered results with real target.

2 Algorithm

At figure 5 you can see the structure flow chart of the developed algorithm. Let's describe it in details. At first blocks of flow-chart we show input data of the algorithm: the next frame frame of the video sequence, the vector of parameters opt_flow of the optical flow, the matrix of training descriptors sum_descr and the vector vPoints of the points, which correspond to the desired targets.

The algorithm starts with calculation of the descriptors descriptors_1GPU of the image frame using a graphic subsystem. Then the obtained descriptors are compared with the preliminarily trained matrix sum_descr by means of an artificial neural network model. Operation of the model is described in the next section of the paper.

The output results of the artificial neural network model are the matrices trainIdx, distance, matches. Further on, we select the descriptors of the vector descriptors_1GPU which most exactly coincide with the matrix sum_descr according to the maximum distance criterion MAX_DISTANCE. The next step of the algorithm consists in transformation of the obtained descriptors good_matches into the items of the vector vPointsBig. Coordinates of each group of the vector vPointsBig are averaged by means of a special algorithm. The rest items of the vector vPointsBig are added to the vector vPoints with the help of the algorithm of point adding to the vector vPoints, which is similar to the algorithm of new points adding for point targets. The vector vPoints, which now contains some added items, is updated with the help of the correspondent algorithm, similar to the algorithm of the vector vPoints update for point targets. The last step of the algorithm is output of the suspected targets obtained by analysis of the vector vPoints.

3 Implementation

For analysis of the suggested algorithm we have developed software implementation [8]. For faster execution of our software tool we use multithread loading of the CUDA-calculator (see Fig.6).

Since new versions (higher than 4.0) of the CUDA platform follow the rule "one CUDA context for one process in the system", certain complications, concerning software implementation of multithread loading of the CUDA-calculator, become a problem. It is possible to avoid such limitation, using API CUDA and generating virtual CUDA contexts for one thread in RAM. Owing to such approach, we have no limitations concerning the rule "one CUDA context for one process", and each thread will have its CUDA context. In this case, the only imitation is available RAM of the computer device.

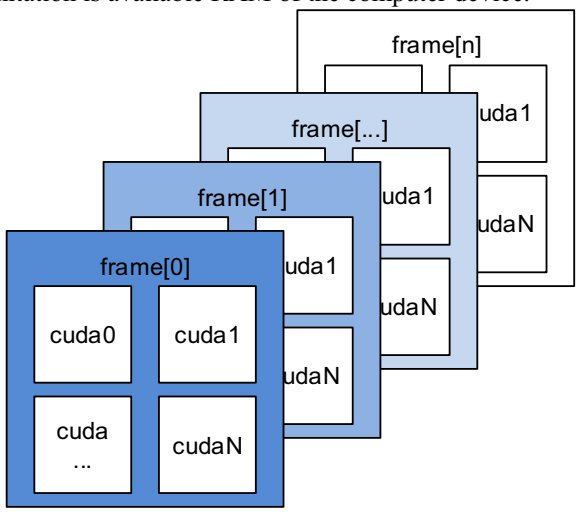

Figure 6. Multithread loading of the CUDA-calculator.

The main idea is loading a group of images into video memory and data-intensive processing of image fragments (the image is equally divided among threads). The number of threads (cuda1 ... cudaN) is chosen experimentally according to the characteristics of the used video card and the size of available RAM of the computer system (recommended utilization of the

graphics processor (GP) is not more than 80%). It is necessary to take into account utilization of the graphics processor, not the top temperature limit of the core.

To analyse the execution time of our software implementation of the suggested algorithm with multithread programming of the CUDA-calculator we used the following test [9-10]. The number of used threads is 1-8, the input number of images is 2-64. If we use 1 thread and 64 images, then utilization of the graphics processor does not exceed 10%, and if we use 8 threads and 64 images, then GP utilization is 80%. Table 1 shows the testing results. The first column contains the number of images in the sequence. The first line contains the number of threads. The execution time is measured in seconds.

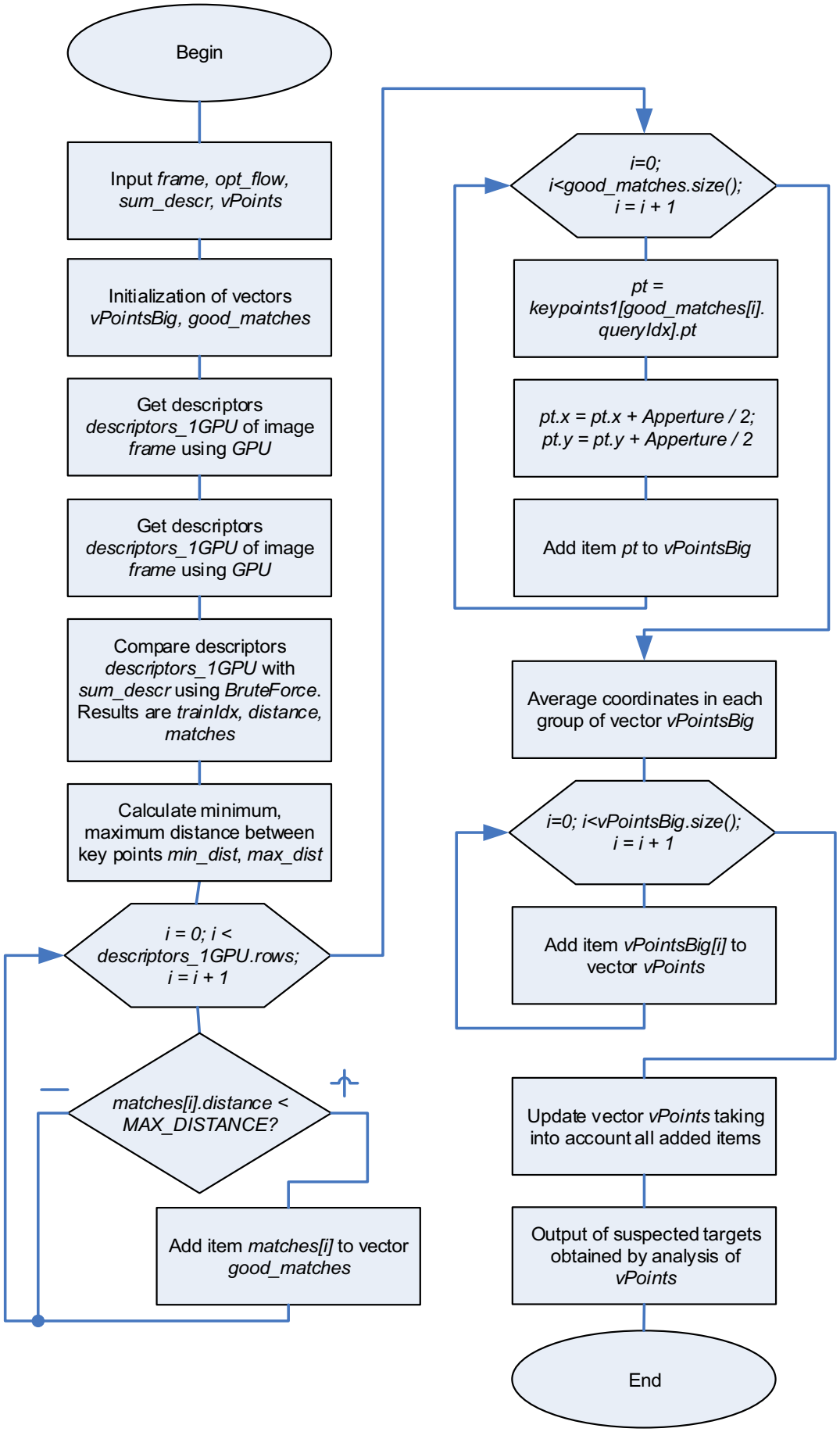

Figure 5. The structure flow chart of the algorithm of large-scale object search and detection from complex background.

Table 1. Testing results of the execution time.

	1	**2**	**4**	**8**
2	1.2	0.8	0.45	0.22
4	3.4	2.3	1.2	0.55
8	5.9	3.3	1.59	0.79
16	12.8	7.1	3.8	1.7
32	24.5	13.5	7.6	3.4
64	97.3	51.8	26.1	13.5

As you can see from the table, owing to use of multithread loading of the CUDA-calculator it is possible to reduce time costs proportionally to the number of used threads.

Acknowledgment

The project is supported by the Grant of the President of Russian Federation № MK-5463.2016.9.

References

1. Herbert Bay, Tinne Tuytelaars and Luc Van Gool, "Speeded-up robust features (SURF)," Computer vision and image understanding, vol.110, No.3, 2008, pp. 346-359. [8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. “BRIEF: Binary robust independent elementary features,” In European Conference on Computer Vision, 2010.
2. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images (PDF Download Available). Available from: https://www.researchgate.net/publication/292157133_Image_Matching_Using_SIFT_SURF_BRIEF_and_ORB_Performance_Comparison_for_Distorted_Images [accessed Jun 14, 2017].
3. B. Shan, “A Novel Image Correlation Matching Approach,” JMM, vol. 5, no. 3, 2010.
4. I.S. Korovin, M.G. Tkachenko, M.V. Khisamutdinov, A.I. Kalyaev, Artificial intelligence hybrid methods application in the task of the heavy oilfields profitability increase / Source of the Document Neftyanoe Khozyaystvo - Oil Industry (1), pp. 106-109 (2016).
5. A.I. Kalyaev, I.A. Kalyaev, Method of multiagent scheduling of resources in cloud computing environments / Source of the Document Journal of Computer and Systems Sciences International 55 (2), pp. 211-221 (2016).
6. I. Kalyaev, A. Kalyaev, I. Korovin, Decentralized approach to control of robot groups during execution of the task flow / Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **98**, 12, pp. 48-54 (2016).
7. Ya.S. Korovin, A.I. Kalyaev, Methods and algorithms of improving the efficiency of data transmission systems in oil corporations' enterprise networks. / Neftyanoe khozyaystvo - Oil Industry (9), pp. 96-100 (2013).
8. A. Kaliaev, Multiagent approach for building distributed adaptive computing system / Procedia Computer Science 18, pp. 2193-2202 (2013).
9. I.S. Korovin, M.V. Khisamutdinov, A.I. Kalyaev, Data Mining Methods Application to the Problem of Handling Corporative Dataset on Heavy Oil Production // Proceedings of the 2016 International conference on intelligent control and computer application. ACSR-Advances in Comptuer Science Research (30) p.387-389, (2016).
10. I.S. Korovin, A.I. Kalyaev, M.V. Khisamutdinov, Using a dynamic model of technological processes to reduce the cost of heavy oil extraction // Conference: 2nd Information Technology and Mechatronics Engineering Conference (ITOEC) p.21-22, (2016).