International Journal of
Science and Engineering Investigations vol. 7, issue 74, March 2018
ISSN: 2251-8843

Recognition Speed Increase Using Multithreading

R. A. Simonyan
IIAP NAS RA
(simonyan.r2@gmail.com)

Abstract- The quick recognition of the objects is very important, as in many cases the quick acquisition of the information on the objects has great importance [1]. In this article we are going to speak about the recognition of the object (according to the template matching based object recognition) and about the mechanisms for increasing the speed of the process.

Keywords- *Object Recognition, Video Surveillance, Recognition Speed Increase, Multithreading, Sub Threads Development, Producer-Consumer Problem*

I. INTRODUCTION

The quick recognition of objects is also important in the following cases:

1- Not to keep the user waiting to see the recognized object.

2- In fast changing environment the mechanism should manage to recognize all the objects in the sight without leaving out any object.

In some cases it can have a vital role [1, 2]. For instance, due to the quick recognition systems security workers can detect the terrorist/criminal on time and take preventing measures to save human lives.

In this particular case to ensure quick recognition multithreading has been implemented [3][4]: An optimal algorithm has been developed to increase the speed of big/array object recognition to its maximum.

Multithreading is the simultaneous transaction of Central Processor Unit or of one core consisting of multilateral processes which are appropriately supported by the operational systems. Multithreading aims at expanding the usage of one core using the paralleling of thread-level as well as of instruction-level. If one thread gets many cache omissions, then the other threads can continue using the unused resources which can lead to a faster general implementation, as those resources would be “free” if only one thread was being used. Besides, if one part cannot use all the calculating resources of CPU (as the orders depend on the results being produced by each of them), through this thread the inactivity of the other resources can be prevented.

II. METHOD DESCRIPTION

During video surveillance the camera captures an image at 24 frames/cadres in a second frequency, whereas the object detection algorithm has a certain pace. If while surveillance we take each cadre and if we apply the recognition algorithm, then in one second for each of 24 shots we will have 1 second for the execution of the algorithm, that is to say one real second will last 24 seconds. Thus, implementing the object recognizing/detection algorithm we will fall behind the time, which is not appropriate for video surveillance where it is necessary the immediate detection of objects.

During the surveillance the video acts like a stream, which works in a real time. To tackle the aforementioned issues multi-thread algorithms are being used.

We are going to discuss one of the famous methods of object detection for surveillance through multi-thread means, afterwards we will discuss the offered method and its advantages.

The algorithm working through multi-thread means first of all establishes a link with the camera and gets the information from the camera, in the format appropriate for the future development. Then the algorithm is designed to have two threads, which the computer will run simultaneously (fig. 1.) [3][4]. A buffering array will be created where each of the cadres taken during the surveillance will be stored for the further development. Through the first thread the algorithm code reads the information got from the camera, that is to say all the cadres according to the sequence and are stored in the aforementioned buffering array. This process continues till the camera is in a working mode. Consequently, every second the algorithm developed in the first thread saves 24 cadres/shots in the buffering array. Thus, the last element of the array corresponds to the last cadre of the camera at any time. Having the last cadre, we will have the information available on the events happening at the present moment.

Through the second thread the object detection algorithm will be implemented [1][2], the latter will turn to the buffering array and will take the cadre for development. As the first and the second threads start the process simultaneously, it is possible that the second thread starts its work faster than the first manages to save the first cadre got from the camera.

This will lead to a problem for the system. That is the reason that the second thread turns to the buffering array and checks the presence of the cadre. Object detection block of the second thread will start working as soon as when the first thread saves the first cadre in the buffering array.

Thus, by taking the cadre from the array the detection block will return to the system the detected objects, then, finishing the job, it will turn to the buffering array again, and will take the cadre saved the last and will execute the detection process again. This process will be executed in an infinite circle till the camera is in the working mode.

This model of multi-threading is called Producer-consumer problem. The software part, that is to say the algorithm, has been developed to carry out trials on the latter and to do time calculations. Figure 1 depicts the block-scheme of the algorithm.

It is worth mentioning that in this model the pace of the first thread should be faster than the one of the second thread, so that the system operates optimally. Otherwise, by saving the cadres very slowly in the buffering array the first thread will leave the second thread inactive, that is to say in a standby mode, as the second thread, by taking the cadre from the array, detects objects faster by emptying the array. To avoid the above mentioned non-optimal work usually some functions are implemented to provide small time delay (for instance 100m/s) for algorithms. In this case it will be used for the second thread immediately after the detection code.

Producer-consumer model will not lead to the aforementioned problem during surveillance for object detection and recognition, as the time span for detection and recognition (the detection time span is noted as t2) is always bigger than getting the cadre from the camera and saving in the buffering array (noted as t1). In contrast to this the detection time span is much bigger that of saving the cadre in the array (formula 1).

$$t2 << t1 \quad (1)$$

Thus, the first thread fills the buffering array very quickly, and the second thread manages to take only a small quantity of cadre of buffering array. More concrete trials have shown that the second thread uses 5-25% of existing cadres of the buffering array, which means that the first thread is 4-20 times faster than the second one. The slow pace of the second thread limits the usage of the majority of cadres of the first thread. This will cause a problem for the user, for instance, in case if the pace of the second thread is 20 times slower than the first one the system will be able to recognize the video in every 20 cadres, which approximately equals to 1 second per cadre frequency. In this particular case an object can happen to be found in that intermittent unobserved place and stay undetected.

To solve this issue a method has been developed which will enhance the frequency of the taking and developing process of cadres, which, in its turn, will lead to the detection of greater number of objects.

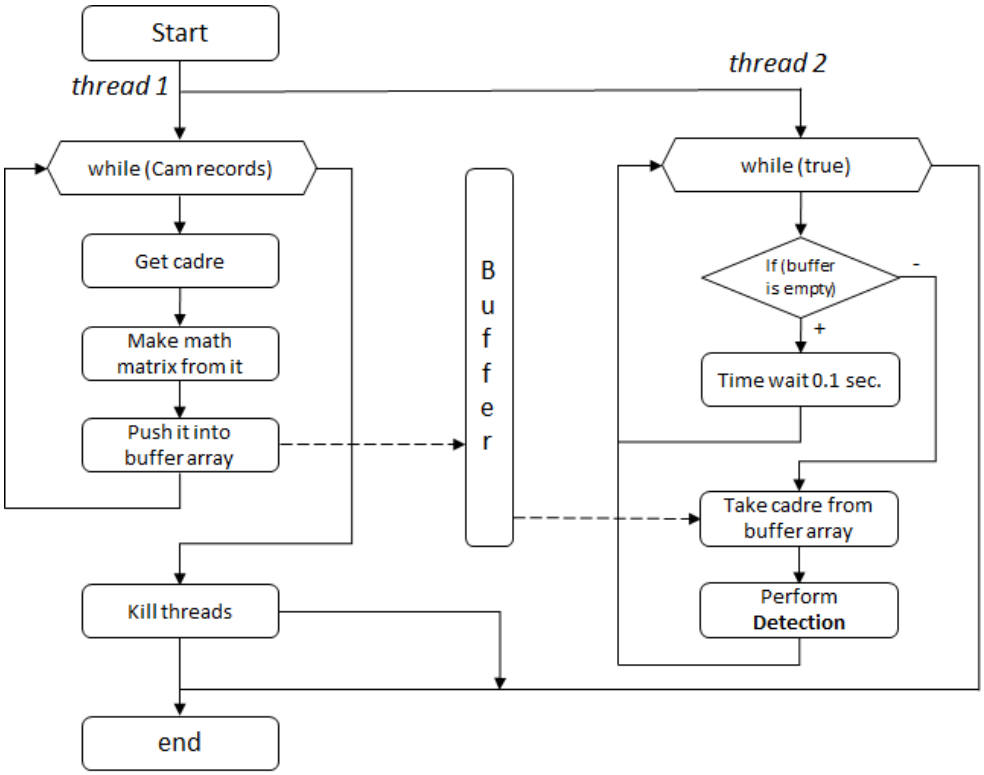

Figure 1. Producer-consumer model block –scheme in case of object detection during video surveillance

III. MODIFIED (OPTIMIZED) METHOD'S DESCRIPTION

Thus, for solving the problem of object detection and recognition during the surveillance a more optimal algorithm of multi-threading has been developed based on the method of Producer-consumer model.

In the case of the aforementioned Producer-consumer model the quantity of the producer (thread 1) and consumer (thread 2) is the same. In the case of the new method the second thread will be divided into more than one thread, as a result we will have one producer and several consumers.

This method presupposes that the parallel running of consumers will lead to the increase of the frequency of detection in the video.

To identify the utmost number of possible consumers a code has been developed which turns to and gets the quantity of free threads in the computer. The quantity is decided so that threads, working in parallel, run as simultaneously as possible, otherwise the great quantity of parallel threads will bring to the overloading of the computer core and the software code will work much slower than without multi-threading, that is to say in sequence.

The algorithmic presentation of the second thread is set in a separate function, which can be "called" by the threads working in parallel. Multi-threading will be organized so that the second thread will run the sub threads-the consumers.

Thus, every sub thread-every consumer will run the algorithmic code of object detection, consequently, by taking the last cadre from the buffering array. The threads work in parallel but not ideally in the same time span, that is to say there can be a small, millisecond difference in the time span of the threads. This asynchrony can lead to a problem. To understand the problem let us consider the following example. At a certain moment of the system working there are 7 cadres in the buffering array, in case the first thread is the producer, and the sub threads I, II, III of the second thread are the consumers (fig. 2). II sub thread, by starting its work a little faster than the first one, will take 7 cadres from the array, whereas the I sub thread, by starting faster than the III will take the penultimate 6 cadres. Afterwards, sub thread II, after implementing detection (highlighting the objects detected in the cadre), will display cadre 7 in the final video, I sub thread-6^{th} cadre, and III-5^{th} one. We will have the following sequence of the cadres-7, 6, 5, whereas we need to have sequence of 5,6,7. The sequence will be broken and the user will see on the screen a video consisting of mixed sequence of cadres.

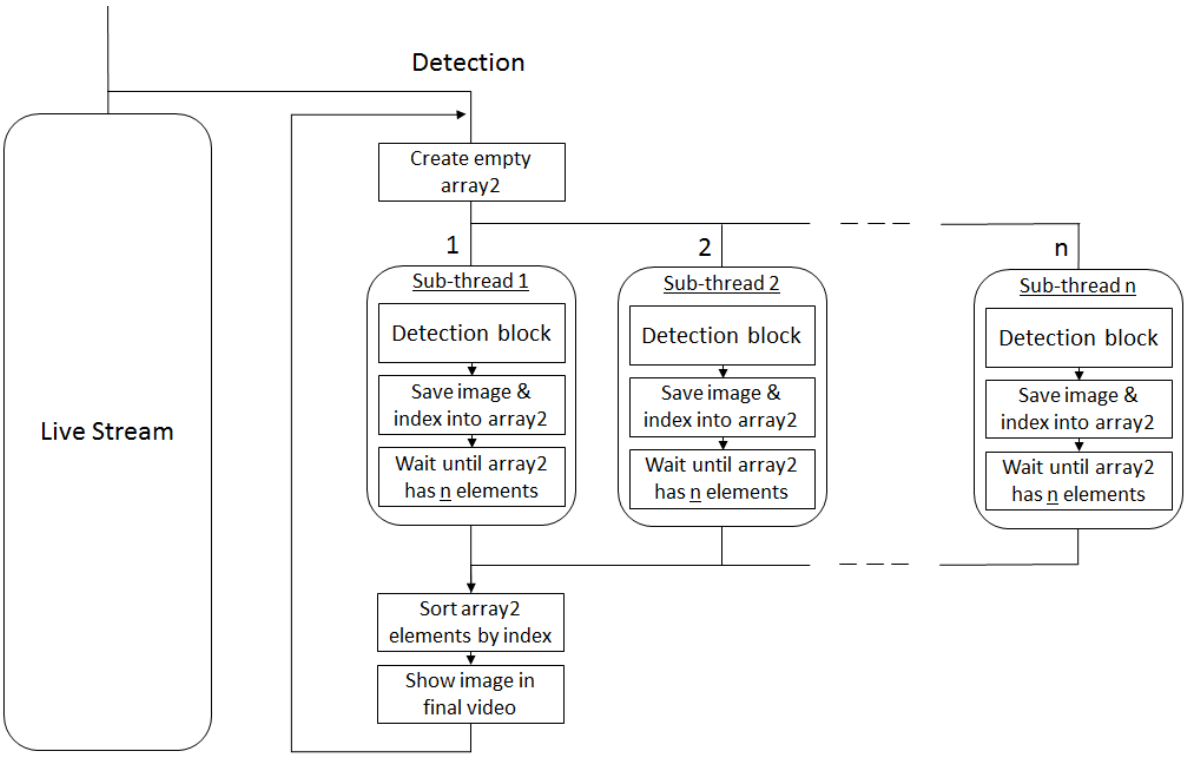

Figure 2. Modified Producer-consumer model block scheme organized through sub threads

To escape this problem first of all we will change the program/software block, which is responsible for taking cadres from the buffering array while detection. The aforementioned software/program block along with the cadre will transfer the index of the cadre from the buffering array, at the same time not removing the cadre from the array. The index shows the position of the cadre in the array, for instance the index for cadre 27 is 27. This index will allow saving each cadre in a chronological order after the consumers' activities.

During multiple threading one of the sub threads will finish the work the first-that is to say the object detection. Afterwards the sub thread will save the cadre and the index as a tuple variable and will wait for the end of the work of the other sub threads. The other sub threads will work in the same way, waiting for the end of the work of the other sub threads. This can be organized through the following checking system, by comparing the number of elements of array with previously known number of sub threads.

This way we will force all the sub threads work at the same pace, consequently the working time will be equal to the most slowly pace of the sub thread.

Every sub thread, by turning to the buffering array, locks it so that the other sub threads do not use that element.

In the new array the elements are spread randomly. We will use sort algorithm according to the indexes, as a result of which the cadres are arranged in the accurate chronological order in the array. This being so, the sequence will not be broken and the user will see a wide of accurately sequenced order of cadres. The new array is zeroed preparing a ground for the next cycle rotation.

As we know the first phase in the process of object recognition is the detection of objects, and due to the described detection method we win time which will contribute to the quickening of the pace of general recognition.

We have spoken about the quickening of the pace of the detection process, we have also discussed the method for quickening the recognition method. It is evident that in the image there can be more than one object. On the whole the recognition algorithms are organized so that after the detection the objects are identified and for each object separately recognition algorithm is implemented one after another. The recognition algorithms are responsible for the detection of only one object at a time. That is to say, it recognizes the first one, after finishing it turns to the second and so on.

To quicken this process we will organize the sub threads so that each of them divides into small sub threads, when each of them will implement the recognition code in its turn. Each time small sub threads will be generated by the number of objects, consequently each of them will be responsible for the detection of one object (fig. 3).

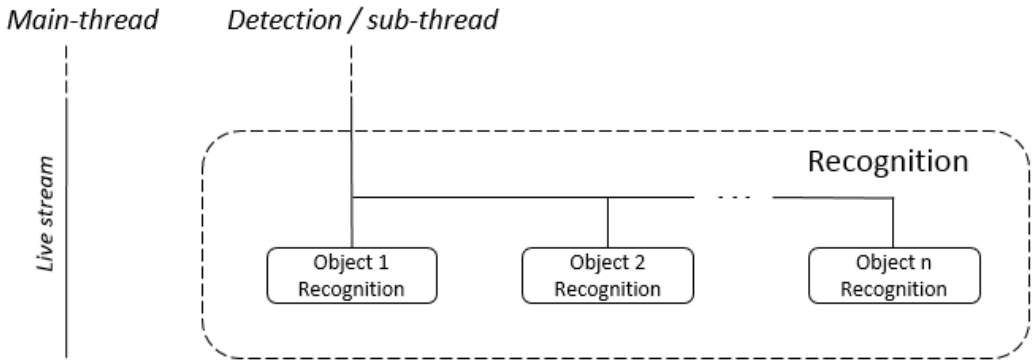

Figure 3. Recognition block and small sub threads

By organizing the recognition process in parallel we spend approximately as much time as it would be needed for the recognition of one object. Right after the detection the sub thread will be divided into smaller sub threads, that is to say, after the detection block, which will be followed by the algorithm responsible for the saving of newly created image and the saving of the index, which in its turn will be followed by the appropriate activities and after this all the cycle will be repeated (fig 4).

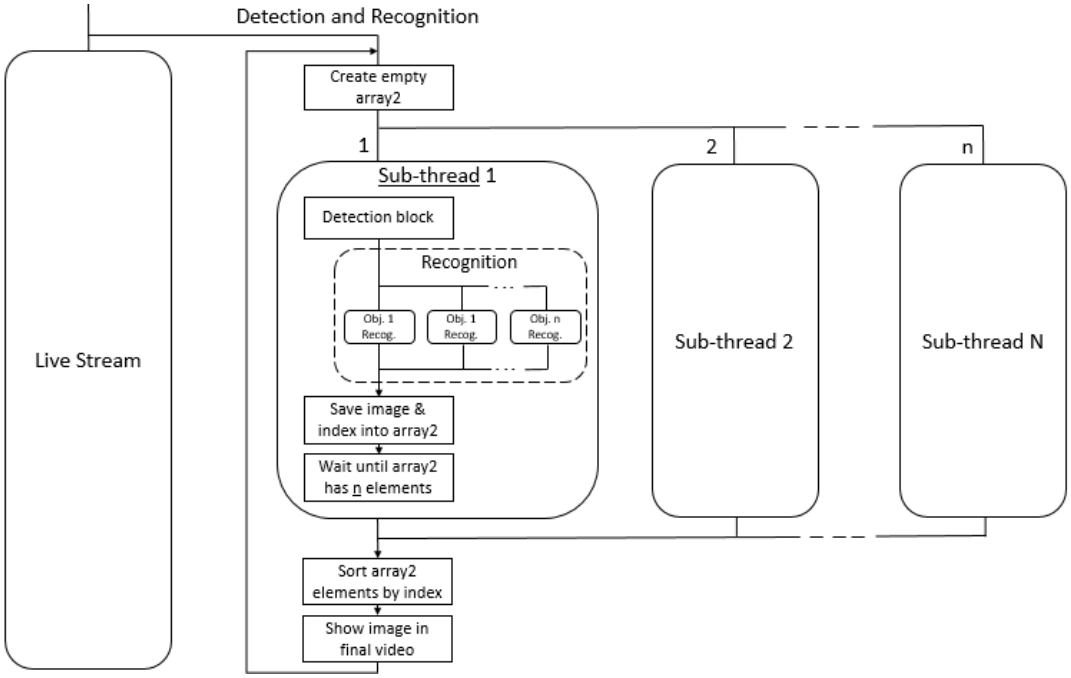

Figure 4. The block scheme of integral algorithm of detection and recognition

Due to the developed method the pace of objects recognition in video (detection and recognition) quickens substantially.

IV. CONCLUSION

Thus, to quicken the pace of object recognition process speed in videos an algorithm has been developed working through multi-threading.

To develop the algorithm Producer-consumer model has been investigated, including its advantages and disadvantages. To increase the algorithm speed the producer-consumer model has been modified.

In the first phase, an algorithm has been developed to increase the pace of object detection algorithm, which includes organizing the activities of sub threads, cycle and new array, and in the second phase the quickening of recognition processes has been ensured.

Trials/experiments have been carried out, the results of which have shown that the pace of the modified method is substantially quicker than that of the previous one, etc.

One of the advantages of the method is that it does not keep the user waiting for them to see the recognized objects on the screen. In fast changing environment the system manages to recognize all the objects in the view without leaving out any object.

REFERENCES

[1] R. A. Simonyan, D. A. Simonyan, “Detection and ignorance method of false targets during object detection”, In Computer Science and Information Technologies (CSIT) Conference 2017, pp. 372-375.

[2] R. A. Simonyan, “Hidden and Unknown Object Detection in Video”, In International Journal of New Technology and Research (IJNTR) Volume 2, Issue 11, Nov. 2016, pp. 22-25.

[3] N. Saaidon, W. Sediono, "Multicolour object detection using multithreading", In IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2015, pp. 48 - 52.

[4] S. K. Mahapatra, S. K. Mohapatra, S. Mahapatra, S. K. Tripathy, "A Proposed Multithreading Fuzzy C-Mean Algorithm for Detecting Underwater Fishes", 2016 2nd International Conference on Computational Intelligence and Networks (CINE), 2016, pp. 102 - 105

[5] L. Fan, A. C. Loui, “A Graph-Based Framework for Video Object Segmentation and Extraction in Feature Space”, In IEEE International Symposium on Multimedia (ISM), 2015, pp. 266-271

[6] Basel A. Mahafzah, "Parallel multithreaded IDA* heuristic search: algorithm design and performance evaluation", In International Journal of Parallel, Emergent and Distributed Systems, Dec 2010, pp. 61-82

[7] Khaled El-Fakih, Gerassimos Barlas, Mustafa Ali, Nina Yevtushenko, "Parallel algorithms for reducing derivation time of distinguishing experiments for nondeterministic finite state machines", In International Journal of Parallel, Emergent and Distributed Systems, Mar 2017, pp. 197-210

[8] E. S. Fraga, "Symmetric multiprocessing algorithm for conceptual process design", In Computer Aided Chemical Engineering, Volume 8, 2000, pp. 637-642

[9] P. O. Frederickson, R. E. Jones, B. T. Smith, "Synchronization and control of parallel algorithms", In Parallel Computing, Volume 2, Issue 3, Nov 1985, pp. 255-264.