ISSN: 2053-3578 I.F. 9.1

ALGORITHMS FOR MANAGING GRAPHICAL DATABASES IN DIGITAL SIGNAL PROCESSING

Rakhimov Bakhtiyar Saidovich[1], Rakhimova Feruza Saidovna[2] Khujamov Elyor Jumanazarovich[3], Saidova Zarina Bakhtiyar qizi[4]

[1]Head of the Department of Biophysics and information technologies of Urgench branch of Tashkent Medical Academy, Uzbekistan, bahtiyar1975@mail.ru

[2]Assistant - professor, Tashkent University of Information Technologies named after Muhammad al Khwarizmi, Uzbekistan, feruzarakhimova@mail.ru,

[3]student of Urgench state University, Uzbekistan, elyorkhujamov@mail.ru,

[4]master of Tashkent University of Information Technologies named after Muhammad al Khwarizmi, Uzbekistan, ootabek2001@mail.ru

Abstract. Computer vision as a scientific discipline refers to the theories and technologies for creating medical database systems that receive information from an image. Despite the fact that this discipline is quite young, its results have penetrated almost all areas of life. Computer vision is closely related to other practical areas like image processing, the input of which is two-dimensional images obtained from a camera or artificially created. This form of image transformation is aimed at noise suppression, filtering, color correction and image analysis, which allows you to directly obtain specific information from the processed image. This information may include searching for objects, feature points, segments, etc. All scalar processors operate in SIMD mode, while executing a block of threads in the G80, the number of threads in a block is 32, called a warp.

Key words: Grafic processors, algorithm, memory, hardware, software.

Introduction.

For the first time, the G80 architecture was presented in November 2006 and became widespread in several versions of graphic processors, differing in the number of multiprocessors installed on them [1]. If we remove all the elements of the architecture associated with the

processing of graphics, we get the following structural diagram.

The main elements of the G80 architecture are:

1) A flow control block designed to generate a schedule and control the execution of flows. This block is controlled by the main processor of the system (CPU), which delegates a parallel task consisting of many threads to the GPU;

2) Computing unit, consisting of multiple streaming multiprocessors that receive and process parallel streams (work in MIMD mode);

3) Memory hierarchy, in which the main element is video memory (graphics adapter memory). It is accessed through L1 and L2 caching. However, the memory access operation is very expensive and can cost from 400 to 600 multiprocessor clock cycles.

The multiprocessor consists of the following components [4]:

1) 8 unified scalar processors (SP, Stream Processor), which allow performing both operations on integers and on floating point numbers;

2) 2 blocks for calculating transcendental functions with single precision (SFU, Super Function Unit);

3) shared memory block;

4) flow control unit

5) constant memory cache;

6) general purpose registers (RF, Register File).

The basis of the search for objects in the image using the Viola-Jones method is the Haar features of a rectangular shape. Computing one such feature in OpenCV requires finding three rectangle intensities multiplied by a weighting factor:

$$h = \sum_{t=1}^{3} \alpha_t \cdot ii_t \quad . \qquad (1)$$

The coordinates of the rectangles for a sliding window of minimum size are set when training a cascade of classifiers. Let the dimensions of the sliding window be defined as follows:

$$w' = f^{i} \cdot w_0$$
$$h' = f^{i} \cdot h_0 \quad , \qquad (2)$$

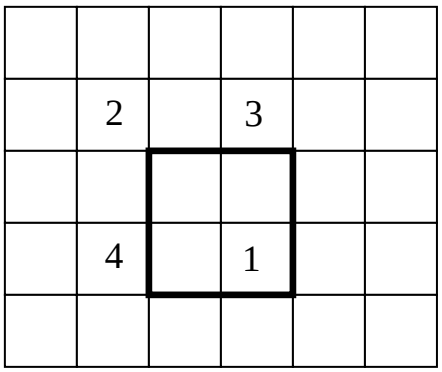
here w_0 and h_0 - minimum (initial) dimensions of the sliding window;

f - window scaling factor;

i - iteration at which the window dimensions are calculated, while the maximum iteration is equal to:

$$i_{\max} = ceil\left(\min\left(\log_f \frac{w+1}{w_0}, \log_f \frac{h+1}{h_0} \right) \right). \qquad (3)$$

To calculate the intensity of one rectangle by the integral, you need to know the values of the integral in 4 pixels (see Fig. 1). Thus, to calculate the intensity of one rectangle, 3 addition / subtraction operations, one multiplication are required, and to calculate the entire Haar sign, 11 addition / subtraction operations and 3 multiplication operations.



$$ii = ii_1 + ii_2 - ii_3 - ii_4$$

Fig.1. Determination of the intensity of the image rectangle by its integral

Further, the Haar signs form a cascade that already determines whether the object is in

the specified area or not. In OpenCV, when training a cascade, each Haar feature has its own response threshold, at which one or another value is added to the sum of the cascade depending on the value of the feature (in this case, the threshold value itself is multiplied by the light correction factor [1,3,11]:

$$s = \sum_{c=1}^{c_{\max}} \begin{cases} h_c < thres_c \cdot l, a1_c \\ h_c \geq thres_c \cdot l, a2_c \end{cases}, \quad (4)$$

where the values $a1_c, a2_c, thres_c, c_{\max}$ are determined when training a cascade of classifiers, as well as weights in the classifiers themselves. Thus, to calculate the cascade of classifiers, it is necessary to execute $12 \cdot c_{\max}$ additions, $4 \cdot c_{\max}$ multiplications and $c_{\max}$ comparisons. Then this sum is compared with the threshold value of the cascade, and if it is greater than it, then it is concluded that the cascade has detected the object. The sequence of such cascades makes it possible to identify complex objects in the image. If for $c_{\max}$ accept the total number of Haar features in all cascades, then in total they require $21 \cdot c_{\max}$ flop (provided that one multiplication requires 2 flop additions). Then the total number of operations required to detect an object by the integral of the image, provided that an object is detected in each pixel, can be estimated by the following formula:

$$W_S(N) = \sum_{i=0}^{I-1} (w - f^i \cdot w_0 + 1) \cdot (h - f^i \cdot h_0 + 1) \cdot (21 \cdot c_{\max}) . \quad (5)$$

It is clear that this situation will not occur on real images, but nevertheless, an estimation of the algorithm execution time for the CPU is necessary in order to find out whether it is expedient to execute the algorithm on the central processor:

$$T_{CPU}(N) = \frac{W_S(N)}{S_{CPU}} . \quad (6)$$

Graphs of the theoretical and experimental time estimates are shown in fig. 2. (sliding window size $w_0 = 24$ на $h_0 = 24$, scaling factor $f = 1.2$, $c_{\max} = 2913$ classifiers). In an image with dimensions of 1024 by 1024, 637 possible areas of the object were found in total.

The next architectural solution for NVIDIA graphics processors was GT200, presented in

June 2008 [9]. The main differences from the G80 in terms of general calculations are:

1) the number of TPCs (Thread Processing Cluster, multiprocessor clusters) has increased;

2) in each TPC, the number of multiprocessors has increased to 3 per cluster, while the L1 cache has also increased;

3) the number of registers for program instructions has doubled;

4) each multiprocessor has a block for computing operations on double precision floating point numbers.

Thus, this architecture made a breakthrough in the number of scalar processors on the graphics adapter and the amount of data processed, but the structural elements and their arrangement remained the same.

Unlike the GT200 architecture, the third generation Fermi architecture [8], presented in September 2009, has significantly redesigned the structural diagram of the GPU.

The central element of the architecture is the L2 cache, which is used to access video memory. Accordingly, its volume has also increased significantly. Streaming multiprocessors are formed around this cache, which have also changed from previous architectures:

1) the number of scalar processors has increased to 32, moreover, they are optimized for working with 64-bit data;

2) it became possible to execute two competing bundles of streams on one multiprocessor;

3) the number of blocks for calculating transcendental functions has increased to 4;

4) it became possible to manage the amount of shared memory and the first-level cache for data;

5) there were blocks for calculating data addresses located in a single address space.

Objective Statement

Obviously, not all computer vision algorithms can be parallelized on GPUs. Any artificial computer vision system, regardless of its area of application, should include the following typical stages of work:

1) image acquisition (photo or video filming);

2) preliminary processing;

3) highlighting characteristic features;

4) detection or segmentation;

5) high-level processing.

Almost all stages can be realized with the help of parallel computer vision algorithms executed on modern parallel computing devices. Some of them can use data parallelism, which is advisable to use on the general-purpose GPUs discussed above. Consider the main groups of computer vision algorithms using data parallelism (this classification is a generalization of groups from):

1) image transformation algorithms - input and output data are two-dimensional images, the coordinates of the output image element differ from the coordinates of the input element. These algorithms include: affine transformations, coordinate system transformations, etc .;

2) filtering algorithms - input and output data are two-dimensional images. Each pixel in the output image is the result of an operation on a group of pixels in the input image that fall into a window of a certain size (filter). Filters can be represented by various mathematical devices: cellular automata, neural networks, functions, etc .;

3) statistical algorithms - input data are two-dimensional images, output data are arrays with statistical information. Each element of the original data can be used for statistical calculations if it meets the requirements of the algorithm. An example of such algorithms can be the calculation of histograms, Hook's transformation, grouping of elements, etc. Parallelization of such algorithms requires communication between processors to combine the accumulated statistical data;

4) recursive algorithms - input and output data are two-dimensional images. Each element of the original image contributes to the formation of all the image elements in the output. These algorithms include: calculation of the integral of the image, distance transformation, etc. This type of algorithms is very difficult to parallelize, since in most cases, computation of a single output item requires the result of previous input items.

Materials nad Methods Bulk Synchronous Parallel (BSP, Valiant, 1990) is an extension of the PRAM model [8]. Later it was transformed into a parallel computing standard [6]. In this model, the main attention is paid to communication and synchronization of processors. Therefore, the model consists of the following components [7]:

1) Processors, each of which has fast local memory and can execute multiple virtual threads;

2) A communication network that allows sending and receiving communication messages from processors;

3) A mechanism for synchronizing all processors at certain points in time.

The algorithm in BSP has a vertical and horizontal structure [10]. The vertical structure is a sequence of supersteps, each of which consists of three main steps:

1) Local calculations on each processor using only the data that was stored in the processor's local memory. Calculations are performed asynchronously;

2) Communication between processes using a communication network (messaging);

3) Barrier synchronization of processes. Once a process reaches a synchronization point (a barrier), it suspends computation and waits for other processes to reach that synchronization point. the results of the program before sending them to the host (in conventional DRAM).

The entire R600 architecture for parallel computing can be divided into three blocks: a flow control block, an array of vector processors, and a memory controller.

The flow control block receives commands from the central processor to perform parallel processing of data and generates a schedule for the execution of threads. It consists of a command processor, a thread generator, and a command dispatcher. The command processor generates interrupts for the CPU during the execution of parallel computations and gives the job to the thread generator. Based on the information received, the thread generator compiles a schedule for blocks of threads executed on multiprocessors and converts it into an array of VLIW (very long instruction word, very long machine instruction [7]) instructions. Thus, all calculations on the graphics adapter are performed according to the EPIC architecture (Explicitly Parallel Instruction Computing, a system of instructions with explicit parallelism [6]). This architecture allows you to schedule the load of all processors on each cycle [4]. In this case, the schedule is compiled statically with the most efficient loading of multiprocessors. The commands are then passed to the command dispatcher for execution, which has access to both the array of vector processors and the memory controller in order to execute the schedule correctly.

Results and Discussion

The models of parallel programming discussed above are intended primarily for the programmer to have an idea about the main structural elements of graphic processors, which include memory and computing elements, and the relationships between them. In addition, both CUDA and OpenCL have some algorithm abstraction that assumes that input and output data are represented as an array of elements, each of which is processed independently of each other, due to which data parallelism is achieved. We highlight the main disadvantages of these models:

1) there is no mathematical description of the abstract model of the GPU, so it is impossible to estimate the running time of a particular algorithm on different GPUs;

2) significant parameters of parallel algorithms have not been identified, thanks to which it is possible to analyze and compare these algorithms in terms of execution time on a GPU with a certain configuration;

3) despite the fact that the models are heterogeneous (i.e. they take into account not only graphics processors, but also central ones), they do not have methods for making a decision about the target computing system;

4) for these models, general principles for optimizing parallel computing have not been developed (in [2, 6], optimization methods are given for a specific platform, but not for a model);

5) there is no general methodology for the development of parallel algorithms.

Conclusions

If the GPU is the Device, then the size of the partition and blocks is limited. Each block is executed on a separate multiprocessor independently of other blocks. Therefore, the width and height of the block are determined by the maximum number of simultaneously processed threads on the multiprocessor. In turn, when executed, the blocks are divided into bundles of 32 threads, which are launched in accordance with the commands of the multiprocessor thread control unit. Communication between threads in a block is done using shared memory and barrier thread synchronization. Series of numerical experiments have been carried out on the research of piecewise-quadratic bases. With use of the offered algorithm of calculation of coefficients in Haar and Harmut's piecewise-quadratic bases, "Table-1" is achieved. Here the factor of compression Kc is defined by the formula:

$$Kc=N/(N-N1),$$

Where N - Quantity of readout of function N1 - Quantity of the zero factors received as a result of use of offered algorithm.

Computer vision as a scientific discipline refers to theories and technologies for creating

artificial systems that receive information from an image. Despite the fact that this discipline is quite young, its results have penetrated almost all spheres of life. Computer vision is closely related to other practical areas [3]: 1) image processing, the input data of which are two-dimensional images obtained from a camera or artificially created. This form of image transformation is aimed at noise suppression, filtering, color correction, etc.; 2) image analysis, which allows obtaining certain information directly from the processed image. Such information may include the search for objects, characteristic points, segments, etc.; 3) vision of the robot, designed to orient the robot in space by modeling the environment from images received from video cameras; 4) machine vision, which is used in production and industry for automatic product quality control, product defect detection, measurement control, etc. Typical applied problems of computer vision are [3,4]: 1) Detection of objects in the image. Despite the fact that a person can easily select specific objects from an image, this problem has not yet been completely solved for artificial systems. Most of the solutions are of a particular nature, based on the specific properties of the object being searched for, and, accordingly, are not suitable for searching for objects that do not have them. There are several universal algorithms for object detection (neural networks, Viola-Jones, etc. [8]), which are slow and have a serious detection error with slight deviations of objects from the desired ones specified during training, but, nevertheless, their simplified versions widely used for small images. Therefore, an important task while maintaining the accuracy of detection is to speed up calculations [5]; 2) Recognition of objects in the image [1].

References

1. European Association of Urology [internet]. Available from: http://uroweb.org/wp-content/uploads/EAU-Guidelines-Non-NeurogenicMale-LUTS-Guidelines-2016.
2. Wang X, Wang X, Li S, Meng Z, Liu T, Zhang X. Comparative effectiveness of oral drug therapies for lower urinary tract symptoms due to benign prostatic hyperplasia: a systematic review and network meta-analysis. PLoS One, 2014 Sep 12, 9: e107593.
3. Reich O., Gratzke C., Bachmann A. et al. UrologySection of the Bavarian Working Group for Quality Assurance. Morbidity, mortality and early outcome of transurethral resection of the prostate: A prospective multicenter evaluation of 10,654 patients. J Urol. 2008; 180b(1):
4. Foley S.J. and Bailey D.M.: Microvascular density in prostatic hyperplasia. BJU Int, 85:70, 2000.
5. Mebust W.K., Holtgrewe H.L., Cockett A.T.K., Peters P.C. and Writing Committee:

Transurethral prostatectomy: immediate and postoperative complications. A cooperative study of 13 participating institutions evaluating 3,885 patients. J Urol, 141: 243, 1989.

6. Rakhimov BS, Mekhmanov MS, Bekchanov BG. Parallel algorithms for the creation of medical database. J Phys Conf Ser. 2021;1889(2):022090. doi:10.1088/1742-6596/1889/2/022090
7. Rakhimov BS, Rakhimova FB, Sobirova SK. Modeling database management systems in medicine. J Phys Conf Ser. 2021;1889(2):022028. doi:10.1088/1742-6596/1889/2/022028
8. Rakhimov B, Ismoilov O. Management systems for modeling medical database. AIP Conference Proceedings 2022, 2432, 060031doi:10.1063/5.0089711
9. Rakhimov BS, Khalikova GT, Allaberganov OR, Saidov AB. Overview of graphic processor architectures in data base problems. AIP Conference Proceedings, 2022, 2467, 020041 doi:10.1063/5.0092848
10. Allaberganov, O.R., Rakhimov, B.S., Sobirova, S.K., Rakhimova, F.B., Saidov, A.B. Problem for Medical System with Infinite Zone Potential in the Half Line AIP Conference Proceedings, 2022, 2647, 050025
11. RB Saidovich, SA Bakhtiyarovich, BB Farkhodovich, KDA Ugli, MMZ Qizi Analysis And Using of the Features Graphics Processors for Medical Problems Texas Journal of Medical Science 7, 105-110
12. Rakhimov, Bakhtiyar; Rakhimova, Feroza; Sobirova, Sabokhat; Allaberganov, Odilbek; ,Mathematical Bases Of Parallel Algorithms For The Creation Of Medical Databases,InterConf,,,,2021,
13. Rakhimov, Bakhtiyar Saidovich; Rakhimova, Feroza Bakhtiyarovna; Sobirova, Sabokhat Kabulovna; Kuryazov, Furkat Odilbekovich; Abdirimova, Dilnoza Boltabaevna; ,Review And Analysis Of Computer Vision Algorithms,The American Journal of Applied sciences,3,5,245-250,2021,
14. Saidovich, Rakhimov Bakhtiyar; Bakhtiyarovich, Saidov Atabek; Farkhodovich, Babajanov Boburbek; Ugli, Karimov Doston Alisher; Qizi, Musaeva Mukhtasar Zayirjon; ,Analysis And Using of the Features Graphics Processors for Medical Problems,Texas Journal of Medical Science,7,,105-110,2022,
15. Rakhimov, BS; Ismoilov, OI; Ozodov, RO; ,Russian “Software and automation of forensic examination”,METHODS OF SCIENCE Scientific and practical journal,11,,28-30,2017,
16. Rakhimov, Bakhtiyar Saidovich; Saidov, Atabek Bakhtiyarovich; Shamuratova, Inabat Ismailovna; Ibodullaeva, Zarnigor Ollayor Qizi; ,Architecture Processors in Data Base

Medical Problems,International Journal on Orange Technologies,4,10,87-90,2022,Research Parks Publishing

17. Rakhimov, Bakhtiyar Saidovich; Saidov, Atabek Bakhtiyarovich; Allayarova, Asal Akbarovna; ,Using the Model in Cuda and Opencl for Medical Signals,International Journal on Orange Technologies,4,10,84-86,2022,Research Parks Publishing
18. Rakhimov, BS; Allabeganov, OR; Saidov, AB; ,Processor means for the spectral analysis of medical signals on the of polynomial walsh bases epra,International Journal of Research and Development (IJRD),5,7,10-11,2020,
19. Forsyth DA, Pons, J. Computer vision. Modern approach / D.A. Forsyth, J. Pons: Trans. from English - M .: Publishing house "Williams", 2004. - 928 p .: ill. - Parallel. tit. English
20. Frolov V. Solution of systems of linear algebraic equations by the preconditioning method on graphic processor devices
21. Brodtkorb A.R., Dyken C., Hagen T.R., Hjelmervik J.M., Storaasli O.O. State-of-the-art in heterogeneous computing / A.R. Brodtkorb, C. Dyken, T.R. Hagen, J.M. Hjelmervik, O.O. Storaasli // Scientific Programming, T. 18, 2010. - S. 1-33.