Comparative Analysis of Stereo Vision Algorithms Implementation on Various Architectures

Aleksey Efimov [1], Dmitry Ustukov [1]

[1] Ryazan State Radio Engineering University named after V.F. Utkin, Gagarina st., 59/1, Ryazan, 390005, Russia

Abstract

A comparative analysis of the functionality of stereo vision algorithms on various hardware architectures has been carried out. The quantitative results of stereo vision algorithms implementation are presented, taking into account the specifics of the applied hardware base. The description of the original algorithm for calculating the depth map using the summed-area table is given. The complexity of the algorithm does not depend on the size of the search window. The article presents the content and results of the implementation of the stereo vision method on standard architecture computers, including multi-threaded implementation, a single-board computer and FPGA. The proposed results may be of interest in the design of vision systems for applied applications.

Keywords

Synthetic vision, stereo algorithms, hardware architectures, FPGA.

1. Introduction

The task of developing mobile robotic systems takes one of the leading places among the problems of modern science and technology. The current stage of development of science and technology is closely related to the widespread introduction of autonomous robotic systems. One of the most urgent problems of synthetic vision is the robot's spatial orientation.

There are several technical approaches to solving this problem: the use of lidars, ultrasonic sensors and television cameras. The first approach gives the best result, but is the most expensive. The second one is the cheapest, but does not allow you to get a full-fledged three-dimensional map of the area. Therefore, the use of television cameras is the best option for solving the problem.

The construction of a three-dimensional map using TV cameras is possible using two sensors, using stereo vision methods. Classical algorithms for solving the problem are quite time-consuming and require very powerful computing tools to provide the necessary speed. However, when considering the problem within the framework of mobile robotic systems, the issues of the dimensions of the device and its power consumption are acute. Therefore, the use of computing systems based on powerful personal computers seems impossible.

To solve the problem, it is worth paying attention to the developing class of computing devices - single-board computers. Their advantages include low power consumption, small dimensions, constant development of the hardware while maintaining the same size.

A large number of existing dense stereo vision algorithms [1, 2] demonstrate good results on reference test stereopairs. As a rule, a “perfect” result is produced by algorithms based on non-local disparity search methods. However, testing these algorithms on images obtained from sensors in real conditions gives different results.

The results of processing terrain images obtained by sensors in real conditions are critical for most algorithms. To a greater extent, errors in calculations cause low-contrast textures and the negative impact of scene illumination. Non-local disparity search algorithms can "lose" some objects in the

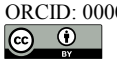
GraphiCon 2022: 32nd International Conference on Computer Graphics and Vision, September 19-22, 2022, Ryazan State Radio Engineering University named after V.F. Utkin, Ryazan, Russia
EMAIL: lexie62rus@mail.ru (A. Efimov); ustukov.mail@ya.ru (D. Ustukov)
ORCID: 0000-0002-4014-8718 (A. Efimov); 0000-0002-4848-8936 (D. Ustukov)


image. As a rule, this happens with periodic or small-sized objects, which can be a significant obstacle to the further movement of the mobile system.

In local algorithms based on the correlation search for paired pixels, such problems are eliminated by increasing the size of the search window. The processing time of one stereopair linearly depends on the window size. Under certain conditions of illumination and composition of objects detected by sensors, the calculation of disparities requires a small window. In other cases, for better results, the window size needs to be increased.

GPU and FPGA architectures impose restrictions on the use of dynamic memory and algorithms with a large number of branches. These restrictions prevent most of the non-local dense stereo computation algorithms from being implemented. For all three algorithms, it is possible to implement an algorithm based on the correlation matching of pixels. The algorithm should allow you to adjust the quality of calculating the depth map by changing the size of the correlation search window for disparities without changing the program or firmware of the device. The processing time of a pair of frames from a stereo system should, if possible, be independent of changes in the algorithm settings.

2. Description of the selected algorithm

The algorithm for calculating the depth map by correlation matching of paired pixels consists of the following steps.

The considered algorithm is described in detail in [1] and has established itself as a reliable method for the depth map calculation, which is distinguished by a sufficiently high accuracy with a low computational complexity.

Step 1. Discrepancy values d are selected from the range [0..d_{max}], where d_{max} is the maximum discrepancy for calculating the depth map, steps 2, 3 of the algorithm are performed for each value of *d*.

Step 2. The summed-area table (SAT) is calculated using the formula

$$I_{i,j,d} = \sum_{n=0}^{j} \sum_{m=0}^{i} \left| L_{m,n} - R_{m-d,n} \right|, \quad (1)$$

where i, j are the coordinates of the calculated element in buffer I, d is the disparity value for which calculations are performed, L are the brightness values of the image pixel from the left sensor, R are the brightness values of the image pixel from the right sensor.

Based on the features of the integrated image, its elements can be calculated in a different way, according to the formula

$$I_{i,j,d} = \left| L_{i,j} - R_{i-d,j} \right| - I_{i-1,j-1,d} + I_{i-1,j,d} + I_{i,j-1,d} \quad (2)$$

For each value in the SAT, only four arithmetic operations are required.

Step 3: The SAT values are used to calculate the cost plane. $P_{i,j,d}$ - an element of the array of cost planes, which is the sum of the modules of the difference in brightness of pixels in a window of a given size with the window center in coordinates i, j. Regardless of the size of the specified window, for the calculation it is necessary to read only four values from the buffer P. Thus, the sum of the modules of the pixel brightness difference in the window is calculated by the formula:

$$P_{i,j,d} = I_{i+t,j+u,d} - I_{i+t,j-u-1,d} - I_{i-t-1,j+u,d} - +I_{i-t-1,j-u-1,d} \quad (3)$$

The calculated value $P_{i,j,d}$ is compared with the best cost value stored in buffer *B* and obtained in the previous calculation step (*d-1*). If the condition $P_{i,j,d} \leq B_{i,j}$ is met as a result of the comparison , then the content $B_{i,j}$ is updated and the disparity value $D_{i,j} = d$ is fixed.

After iterating over all *d* values in buffer *B*, will represent the depth map of the current stereo pair.

The presented algorithm is well parallelized and time-independent of the variable window size used for correlation search of paired pixels. However, the calculation of SAT can be excluded from the algorithm. In this case, step 2 is excluded from the algorithm, and at step 3 calculations are performed using the formula

$$P_{i,j,d} = \sum_{n=j-(h-1)/2}^{j+(h-1)/2} \sum_{m=i-(w-1)/2}^{i+(w-1)/2} \left| L_{m,n} - R_{m-d,n} \right|, \quad (4)$$

where *h* is the window height value for pixel correlation matching, *w* is the window width [1, 3-7].

3. Algorithm Implementation

Four implementations of the algorithm are analyzed:

1. The central processor of a computer with a classical architecture on one kernel;
2. The central processor of a computer with classical architecture using OpenCL technology.
3. Graphics processor (GPU) using OpenCL;
4. Raspberry PI single board computer.

The first implementation method is characterized by a high frequency of operation (up to 5 GHz), but processing such a large amount of data in one thread is not efficient.

The second method allows you to increase the speed of the algorithm by executing on multiple kernels. Those. performance increases by about *n* (where *n* is the number of processor kernels) times compared to the first method.

A feature of the GPU architecture is the number of its kernels (up to several thousand). Working at relatively low frequencies (about 1 GHz on average), such devices is much more efficient than a central processor in processing large volumes of the same type of data. The bottleneck of this implementation is the speed of data transfer from RAM to the graphics card and vice versa. The time spent on transmission depends quite strongly on the structure of the transmitted data and the characteristics of specific equipment, and, according to average estimates, is up to 17-22% of the time spent directly on calculations.

Also of great interest is the implementation of algorithms on single-board computers. An important feature of this implementation is low power consumption, small size and low cost, however, the performance of such systems is lower than personal systems when using a processor and significantly lower when using a graphics processor.

4. Efficiency analysis

Implementation performance analysis is conducted to select the best computing architecture for the stereo of a mobile device. First of all, the performance of the implementation is evaluated, and secondly, the power consumption of the device. The analysis is carried out for three computing architectures of the classical CPU, GPU and Raspberry Pi4 single-board computer. A series of rectified images sized 640x512 pixels is used as initial data (Figure 1). Size 640x512 pixels – the size of the image from the sensor operating in the SWIR range. Such sensors provide better visibility in poor conditions compared to optical cameras. The following values of the parameters w=11, h=5, d=[0..200] are set in the implemented algorithms. The image from one of the sensors of the stereo system on the left and the depth map calculated by the presented algorithm on the right (Figure 2).

Table 1 presents a comparison of the implementation of the dense stereo vision algorithm with a window size of 11x5 on the four above algorithms. As a platform for the first three implementations, a system with the following parameters was used.

Figure 1: The testing image from one of the sensors of the stereo system on the top and the depth map calculated by the presented algorithm on the bottom

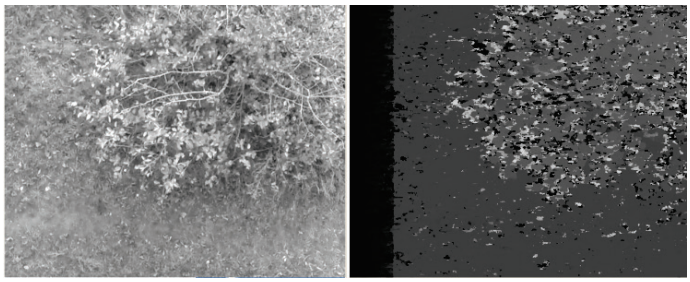


Figure 2: The image from one of the sensors of the stereo system on the left and the depth map calculated by the presented algorithm on the right

Table 1
Comparative characteristic for disparities 0-150

Architecture		Speed, FPS	Execution time, sec	Power consumption (whole module)
PC AMD FX 8300 3.3GHz, RAM – 4GB, GPU AMD RADEON HD 7870, SSD – 128GB	CPU (1 stream)	0.12	8.2348	150W
	CPU (8 streams)	0.57	1.3756	150W
	GPU (multi-stream)	4.10	0.2439	300W
Raspberry 4 b (CPU Quad core Cortex-A72 (ARM v8) 64-bit 1.5GHz RAM 8GB LPDDR4-3200 SDRAM)	CPU (1 stream)	0.06	16.3447	25W
	CPU (8 streams)	0.30	3.3236	25W
FPGA (Virtex 7)*	clock generator frequency 200 MHz	8,31	0,1203	250W

* Evaluation of preliminary implementation [10].

For the stereo system, sensors operating in the SWIR range were selected. Such sensors provide better visibility in poor conditions compared to optical cameras.

To detect and mark occlusions [8, 9], it is necessary to obtain additional D^* values - pixel disparities, by calculating them by shifting the left image relative to the right one. Such an operation requires replacing one of the terms in formula 2 with $\left|R_{i,j} - L_{i+d,j}\right|$.

In the synthesized scheme, the calculation of occlusion is performed simultaneously with the calculating values of $B_{i,j}$. This requires synthesizing duplicate buffers B and D, which we will call B^* and D^*, respectively.

When getting a value $P_{i,j,d}$, it must also be compared with the value stored in B*. If the condition $P_{i,j,d} \leq B^*_{i-d,j}$, is met, then the content $B^*_{i-d,j}$ is updated and the value $D^*_{i-d,j} = d$ is fixed.

A sign of occlusion, the inequality $D_{i,j} \neq D^*_{i-D_{i,j},j}$ happens. The determination of occlusions is made at the last step of calculations [1].

5. Conclusion

The CPU is the most versatile computer for solving technical vision problems. However, the significant size of the motherboard, the need for active cooling and significant power consumption

make it difficult to use the CPU as a computer to solve the problem. The limitation on the time of access to memory cells and the limit of the maximum frequency of the CPU kernel will not allow to significantly increase the performance of the algorithm in the future. Despite this, parallelization of the algorithm over computing kernels reduces the complexity of calculations in proportion to the number of kernels, however, in this case, the memory access time will still remain a limiting factor.

The GPU has a number of restrictions on the implementation of algorithms. Despite the significant number of universal computing kernels, the data exchange bus between the GPU and the main device remains the bottleneck. A significant time loss reduces the advantages of the GPU over the CPU in solving such problems. Despite this, the GPU has a more advanced memory manager, due to which the loss of time during mass memory access becomes not noticeable. However, GPUs with a large number of processing kernels have significant power consumption and, as a result, strong heat dissipation. Single-board computers with a GPU, such as the Jetson tx1, can be used as computers to solve the problem, but this solution is not optimal in terms of estimating the cost of the final solution.

Despite the fact that the Raspberry PI single-board computer during testing showed a twofold lag compared to the CPU, its energy efficiency turned out to be almost an order of magnitude lower than that of the PC CPU. It is worth noting that for many mobile systems, energy efficiency, size, and cost will be more important than performance, especially since at full load the Raspberry Pi aims to process a frame in 3 seconds. By simplifying the algorithm, reducing the range of disparities, skipping the filtering step of the depth map, the Raspberry Pi 4 is capable of producing more than 1 FPS (frame per second).

As directions for further research, it is worth noting the implementation of the considered stereo vision algorithms using FPGA.

Despite the fact that the FPGA in preliminary testing [10] showed almost a twofold performance gain compared to the GPU, its energy efficiency turned out to be two orders of magnitude lower.

In the current implementation on the FPGA, eight digital automata are implemented in parallel. An increase in the number of digital machines will increase productivity with an insignificant increase in energy consumption.

A small amount of used crystal resources (registers and LUTs) allows you to increase the logic of the algorithm, and makes it possible to perform optimization within the resources of a single FPGA chip. In the final implementation, the FPGA can be placed on a small board with electrical wiring and a passive cooling system.

Thus, the results of preliminary studies show that FPGAs are suitable for solving the problem of depth mapping under tight time constraints. And due to low power consumption and small dimensions, the use of such devices is most successful on various mobile devices.

In addition, one of the promising areas is the study of the performance of stereo vision algorithms implemented using CUDA computing parallelization technologies for NVIDIA GPUs and OpenCL.

6. References

[1] Ustukov, Dmitry & Muratov, Yevgeniy & Nikiforov, Michael & Melnik, Olga. (2017). Analysis of the efficiency of dense stereovision algorithm implementation on different computer architectures. 78-81. 10.1109/CCOMS.2017.8075271.

[2] D. Scharstein and R. Szeliski. Middlebury stereo evaluation. URL: http://vision.middlebury.edu/stereo/eval/

[3] Dudko I.S., Efimov A.I., Loginov A.A., Lomteva O.A., Muratov E.R. Automation of research and debugging of algorithms and image processing programs. Izvestiya TulGU. Technical Sciences. Issue 9. - Tula: 2015. pp. 87-95.

[4] Q. S. Yang. A Non-Local Cost Aggregation Method for Stereo Matching. Computer Vision and Pattern Recognition (CVPR), 2012, pp 1402 - 1409.

[5] Forsyth D.A., Ponce J. Computer Vision a modern approach. Upper Saddle River, NJ 07458, 2003, P 926.

[6] R. Szeliski. Computer Vision: Algorithms and Applications, http://szeliski.org/Book/, P 979, 2010.

[7] R. Hartley, A. Zisserman. Multiple View Geometry in Computer Vision (second edition), Cambridge university press, P. 673, 2004.

[8] Geiger, D., Ladendorf, B., and Yuille, A. Occlusions and binocular stereo. In Second European Conference on Computer Vision (ECCV'92), pp. 425–433, Santa Margherita Liguere, Italy.
[9] R. Y. Tsai, A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, IEEE J. Robotics Automat. Vol. RA-3, No. 4, pp 323-344, 1987.
[10] Ustukov D.I., Nikiforov M.B., Muratov E.R. and etc. Implementing one of stereovision algorithms on FPGA. 5th Mediterranean Conference on Embedded Computing (MECO), Bar, 2016. pp. 64-67.