GPU Based Real-time Floating Object Detection System

Yang Jie[1 , a], Meng Jian-min [2,b※]

1. Faculty of Machinery &Transportation, Southwest Forestry University, KunMing, YunNan. 650224, China

2. Patent Examination Cooperation Center of SIPO, Beijing 100081, China

[a]351725623@qq.com,

[b※]corresponding author: mengjianmin@biguozhi.cn

Keywords: Object detection; GPU; Motion Estimation; Parallel Processing

Abstract. A GPU-based floating object detection scheme is presented in this paper which is designed for floating mine Detection tasks. This system uses contrast and motion information to eliminate as many false positives as possible while avoiding false negatives. The GPU computation platform is deployed to allow detecting objects in real-time. From the experimental results, it is shown that with certain configuration, the GPU-based scheme can speed up the computation up to one thousand times compared to the CPU-based scheme.

Introduction

Floating sea mines have been a major threat to navel and civilian ships for the last several decades. Of the 18 U.S. ships damaged by air and naval weapons since 1950, 14 were by mines [1]. In the early 1980s, the U.S. Navy began development of a new mine countermeasures (MCM) forces [2]. The successful detection and classification of floating objects, especially mines, ahead of the ship's course will be essential to it security. Moreover, to decrease the on-board manpower, partial or extensive automation of tedious, full-attention tasks such as floating mine detection and classification is important for ships to operate at maximum efficiency.

Sonar and computer vision systems [3-5] can be applied to detect floating mines. Unfortunately, the wide variation in surface mine and other target signatures, combined with ship motion and the greater impact in shallow water of surface acoustic interactions, reverberation, bubble fields, mixed salinity and currents, organic matter and debris, high amounts of clutter due to bottom features, and other phenomena on the performance of sonar systems, limit the ability of both the system and its operator to detect and classify floating objects with a sufficiently high probability of detection and low probability of false alarm. Further, unlike sonar, computer vision system is passive and it will not have negative impact on the environment. Therefore, a computer vision system which is able to automatically detect floating mine will be very important to ship safety. In [6], a background subtraction based method was used to detect small objects on the sea surface.

In this paper, a floating mine detection algorithm and a GPUs based computation framework was presented. Our previous works in [7] presented an algorithm which reliably extracts suspicious buoy-like floating objects from a video sequence. This algorithm extracts the candidates by examining the appearance and motion information. It is the first step in the whole mine detection system as shown in Fig.1. However, this algorithm requires extensive computation for detecting the floating objects which limits its application to many real-time applications. Meanwhile, in recent decades, many new computation platforms have been applied to image processing and computer vision areas. These technologies include Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs). FPGAs allow high levels of parallelism and often use simple statically-scheduled control and modest arithmetic. The GPU is a massively parallel computing device, originally developed to operate on the many pixels of an image in parallel. GPU device are being used more and more for applications outside computer graphics.

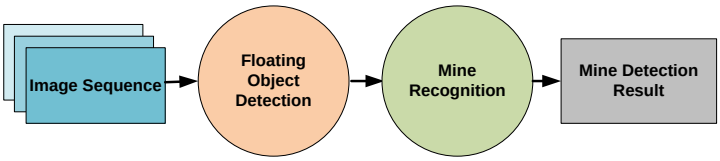

Fig.1 Overall System Diagram

By optimizing current algorithm using the GPU platform, it is possible to attach more computation intense modules after the floating object detection module and therefore improves the overall systems performance.

The paper is organized as follows. In section 2, the floating object detection algorithm is reviewed. In section 3, the computation scheme based on GPU is discussed. In section 4, some experimental results are given to show its effectiveness. Conclusion and future work are presented in section 5.

Algorithm Review

The objective of the mine detection algorithm is to identify regions of interest in video of ocean scenes under a variety of circumstances, including variable sea and lighting conditions, variable target shapes and colors, and distractors such as

wave glare and debris in the scene. The algorithm must have a miss rate very close to zero, and within that constraint should attempt to minimize the false positive rate. Reducing false positives further is the goal of the subsequent processing modules.

The approach used in this paper relies on contrast and motion based on two assumptions we made to simplify the problem.

1) A floating object is distinguishable from the background because its appearance is different.

2) A floating object has distinct motion pattern from distractors.

The diagram for the mine detection algorithm is shown in Fig.2. The algorithm is briefly reviewed in this paper. One can refer to [7] for further details of the algorithm. At each scale, the algorithm includes several stages: candidate pixel selection, motion estimation/analysis, and spatio-temporal smoothing. Results from different scales are fused together to reach the final decision.

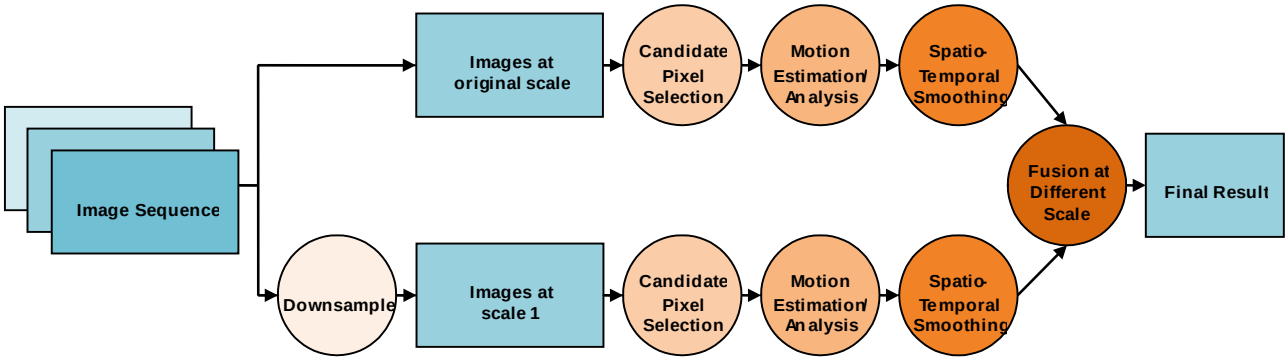

Fig.2 Floating Object Detection System

Multi-scale Scheme.

A multi-scale scheme (two scales in this paper) is applied to accommodate different object sizes and improve efficiency in computation. The down-sampled image is reduced by a factor of 2. Then the same algorithm is applied to both scales.

Candidate pixel selection.

Based on the above assumption, candidate pixels can be selected by locating pixels with a certain minimum amount of contrast to the average of a small region. In particular, we determined that grayscale intensity contrast is adequate, thus avoiding the extra computational cost for color processing. Contrast information is far more reliable than simple intensity thresholding because of

large variations in lighting and mine brightness under anticipated conditions.

Another consideration is the block size. Our assumption is that the object is small compared to the block size and its intensity value is different from the average intensity of pixels in the block. If the object is large and occupies most of the block, the average intensity of the block will be close to the average mine intensity. In this case, it is very likely that the pixels of the object will not be selected as candidates. The multi-scale scheme is helpful here as well.

Motion Estimation/Analysis.

Contrast information alone is not sufficient. Distractors such as sun glint and non-floating objects may also exhibit significant intensity contrast from the immediate background. Motion estimation and analysis is capable of filtering out glare or glint. An individual glare element has a very short visibility cycle, and also shows large changes in brightness. Thus its motion trajectory does not stay constant and it can be distinguished from floating objects. Wave elements exhibit consistent motion and cannot be isolated from targets from that factor alone. However, they also exhibit more generally horizontal movement, whereas floating objects, including mines, exhibit primarily vertical motion. In the current version of the algorithm, a minor bias is given to vertical motion in the filtering of targets.

Existing classical motion estimation algorithms such as block matching [8,9] and optical flow [10,11] have difficulty in this case because of the variant size of the object and abrupt brightness changes. A motion correspondence algorithm called the “motion projection algorithm” is proposed in [7] to estimate motion with high accuracy. Essentially, this algorithm aims at finding the best correspondence of one pixel across two neighboring frames. On the other side, the motion estimation is on a pixel-wise basis which requires extensive computation.

Spatio-temporal Smoothing.

After candidate pixel selection and motion analysis, the result still contains noise and non-object distractors, for example moving waves. According to the observation of motion difference between the object and background, spatio-temporal smoothing is performed to retain only the motion regions which are consistent in spatio-temporal domain. Filtering is performed by simply counting the number of pixels exhibiting consistent motion from the motion estimation and analysis, and removing pixels that have a score below a threshold. Currently, the threshold is one-half of the mask size. For example, if the mask is an 11×11×11 volume, there must be at least five pixels with consistent motion to pass the filter.

GPU Implementation

GPU is a massively parallel computing device, originally developed to operate on the many pixels of an image in parallel. The GPU is a co-processor, meaning that a host program prepares data, copies it to the GPU memory, and then launches computational kernel which run in the GPU. The resulting data are then copied back to the PC memory from the GPU device memory for further analysis or display.

The GPU kernels are written in an extended version of C and compiled by the NVIDIA-provided cross compiler. A kernel’s computation is subdivided into blocks, each of which executes entirely within one of the multiprocessors and independently from other blocks in the kernel. Blocks are further subdivided into lightweight threads, groups of which are executed in a multiprocessor’s SIMD elements. Threads within a single block communicate via shared memory and synchronization calls.

Global memory is large and visible to all the threads across blocks. It also takes more clock cycles to access the global memory which is optimized for block transfers. At same time, each thread has its own shared memory which is smaller but takes less clock cycles to access. Shared memory is optimized for concurrent access. The programmer, in general is responsible for manually managing the memory hierarchy.

After profile analysis, the most time consuming modules in the algorithm are indentified and then selected for GPU optimization. These modules include candidate pixel selection, motion estimation analysis and spatio-temporal smoothing.

To take advantage of the parallel architecture of GPU, each incoming image is divided into tiles of equal size as shown in Fig.3 which are processed by the cores in the GPU. In the figure, the tile size is the same as the block size for the candidate pixel selection which is 80×80. The choice of tile size doesn't affect the accuracy of algorithm; instead the speed of the GPU based design which is shown in the experimental results. The reason to choose the same tile size as the blocks is that as long as the tile and the block are aligned, one tile won't need information from other tiles to calculate the average and deviation of intensity value inside a block. Therefore, the processing for each tile is independent and it does not require synchronization between threads.

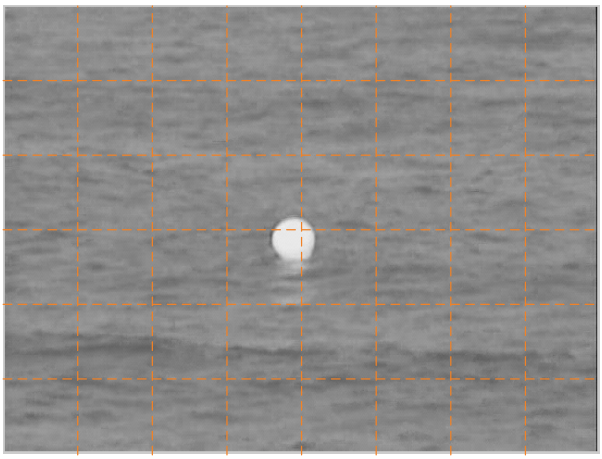

Fig.3 Tile partition for one image (tile size 80×80)

At the beginning of each frame, the image tiles are copied to the corresponding shared memory locations. Each thread can determine, at run time, its tile and thread index and thus which subset and thus which subset of the data to operate on. At the end of processing, the results from each thread are combined into the final result.

Experimental Results

The objective of this algorithm is to detect all possible mine candidates and provide a good candidate list for further processing such as high-level motion analysis and object recognition. Due to the nature of the application and the fact that this algorithm is a pre-processing component of the overall solution; false positives are more tolerable than false negatives. Although this approach generates a large number of false positives, many are filtered out in the subsequent motion estimation/analysis module as well as in the spatio-temporal smoothing module. From the binary mask indicating objects locations, we treat connected set of pixels as a single object. The output of the overall algorithm is a set of centroids, bounding boxes, and number of pixels of each of these potential mine objects. Currently, the proposed algorithm works well on most videos. The miss rate is 0% for objects up to 1000m, 3% for 1500m, and 11% for 2000m. Fig.4 shows some examples of detection result including two IR video frames. Detected objects are highlighted in small blue boxes.

The proposed algorithm was implemented on two different platforms for comparison. The first scheme is based on general purpose CPU. The second scheme is based on the GPU. The computing time for the selected modules on each platform was listed to illustrate the effectiveness of the optimization. The image reading/scaling routines and other interface logics have been excluded from the time calculation.

Fig. 4. Mine detection result examples

CPU-based scheme.

The algorithm is implemented in the MATLAB environment; major parts of the algorithm, such as candidate pixel selection, motion estimation/analysis, spatio- temporal smoothing are optimized using the C-Mex code; the experiment runs on a Dell Desktop equipped with an Intel Core 2 Quad Q6600 CPU running at 3.2 GHz and 3GB system memory.

GPU-based scheme.

We use the CUDA toolkit 4.0 from NVIDIA to implement an optimized version of the proposed algorithm. The GPU platform we used is the NVIDIA Tesla C1060 computing platform. This system has 240 processor cores and each core is clocked at 1.296 GHz. It is equipped with 4GB of GDDR3 device memory at 102 GB/s communication bandwidth and capable of processing 933 GFLOPs per second.

The results on computation time for the CPU and GPU based schemes are compared in Tab.1. The CPU-based scheme is denoted as Conf1. The choice of block size doesn't affect the CPU-based scheme as much from the experimental results which agree with our intuition. The data listed for Conf1 are calculated when block is set to 80×80. Three GPU-based schemes are tested using three sizes of tile and block. They are Conf2 (40×40 pixels), Conf3 (80×80 pixels), and Conf3 (120×120 pixels). These small tiles are redistributed to parallel thread blocks. The number of threads wrapped in each GPU block is configured to 40, 80, and 120 individually for Conf1, Conf2 and Conf3. Each thread in each GPU block processes one row in image. To minimize the bias because of image content, three image sequences are chosen and the average running time for each sequence is listed.

Tab.1 Performance Comparison

Sequence	Conf1 (ms)	Conf2(us)	Conf3(us)	Conf4(ms)
seq1	188.5	209.1	520.4	24.3
seq2	225.9	222.1	517.3	34.1
seq3	234.1	224.2	524.6	31.3

As shown in the table, the GPU-based scheme is around 8 to 1000 times faster than the CPU-based schemes depending on the configuration. The GPU-based scheme with Conf2 (40 threads per block) achieves the fastest speed, around 1000 times faster than Conf1. When

decreasing the number of GPU blocks and increasing the number of threads wrapped in each GPU block, we notice the performance gets worse. In Conf2, the GPU speedup drops to around 500 times faster than Conf1; the Conf3 only achieves less than 10 times speedup. The main reason for the performance difference is mostly due to the memory structure in GPU. The local/shared memory has a much faster accessing speed than the GPU device/global memory. Increasing the number of threads in each block causes more frequent conflict in shared-memory access and lack of shared-memory space. As a result, it takes longer to synchronize all the threads wrapped in one GPU block. Also, to solve the problem of lack of shared-memory space, part of the data used during processing has to be moved to device memory and thereby results in higher memory-access latency.

Conclusions and Future Work

In our previous works, a floating object detection algorithm and its general purpose CPU based implementation can achieve a robust and accurate result. However, the computation cost keeps it from being applied to real-time applications. In this paper, an efficient GPU-based scheme is proposed to speed up the computation. By taking advantage of the massive parallel architecture in the GPU processor, computation of the proposed algorithm can be decreased to less than one millisecond which is short enough for most of real-time applications.

The floating object detection is the first part of the whole mine detection system. To fully detect floating mines automatically, the system will need a mine recognition module which can post process the results and lower the false positive rate. The mine recognition module can use the features such as intensity, shape, motion and etc from the floating object detection module.

References

[1] D. Skinners, "Mine Countermeasures (MCM) Sensor Technology Drivers". *SPIE Proceedings Detection Technologies for Mines and Minelike Targets*, Vol. 2496, 1995.

[2] http://en.wikipedia.org/wiki/Avenger_class_mine_countermeasures_ship

[3] C. Zimmerman, M. Coolidge: "The Forgotten Threat of Attack by Sea: Using 3D Sonar to Detect Terrorist Swimmers and Mines", *IEEE Conference on Technologies for Homeland Security*, Nov., 2002.

[4] G. J. Dobeck, J. Hyland, "Sea mine detection and classification using side-looking sonars", *SPIE Proceedings Annual International Symposium on Aerospace/Defense Sensing, Simulation and Control*, 1995, pp. 442-453.

[5] Y. Chen, T. Q. Nguyen, "Sea Mine Detection Based on Multiresolution Analysis and Noise Whitening", *Technical Report*, 1999.

[6] A. Borghgraef, O. Barnich, F. Lapierre, M. Van Droogenbroeck, W. Philips, M. Acheroy, "An Evaluation of Pixel-Based Methods for the Detection of Floating Objects on the Sea Surface,", *EURASIP Journal on Advances in Signal Processing*, vol. 2010, Article ID 978451, 11 pages, 2010. doi:10.1155/2010/978451

[7] Z.Y. Wei, D.J. Lee, D. Jilk, R. Schoenberger, "Motion projection for floating object detection", *3rd International Symposium on Visual Computing*, Lake Tahoe, CA, U.S.A., Nov, 2007, p.152-161.

[8] Y. Huang, C. Chen, C. Tsai, C. Shen, L. Chen, "Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results", *The Journal of VLSI Signal Processing*,vol. 42,2006,pp 297-320

[9] N. S. Love, C. Kamath, "An Empirical Study of Block Matching Techniques for the Detection of Moving Objects", *LLNL Technical Report UCRL-TR-218038*, 2006.

[10]B. K. P. Horn, B. G. Schunck, “Determining Optical Flow”, *Artificial Intelligence*, vol. 17, 1981, pp 185-203.

[11]B. D. Lucas, T. Kanade, “An iterative image registration technique with an application to stereo vision”, *Proceedings of Image Understanding Workshop*, 1981, pp 121-130.

Received Date:2016-05-8
Fund:CSC Foundation(2014(3033))
About the author: YangJie, female, associate professor, mainly engaged in computer vision applications, and worked in computer vision and the remote sensing group of Technical University Berlin as a visiting scholar in Sep.2014-Sep.2015,her Dr. graduated from Beijing Institute of Technology.

※Corresponding author: Meng Jian-min, male, Corresponding address: Chinese Patent Building, No.11, Gaoliang qiao xiejie, Haidian District, Beijing .Email:mengjianmin@biguozhi.cn Tel:010-82246431, cell:13521535745, Fax:010-82246300