Design & Implementation of Real-time Parallel Image Processing Scheme on Fire-Control System

Chang Bae Moon
ICT-Convergence Research Center
Kumoh National Institute of Technology
Gumi, South Korea
cb.moon@kumoh.ac.kr

Byeong Man Kim
Department of Computer Software Engineering
Kumoh National Institute of Technology
Gumi, South Korea
bmkim@kumoh.ac.kr

Dong-Seong Kim
Department of IT Convergence Engineering
Kumoh National Institute of Technology
Gumi, South Korea
dskim@kumoh.ac.kr

***Abstract*— In this paper, a parallel image-processing method based on OpenMP is proposed for a software application in a fire-control system(FCS). The proposed method for image-processing is composed of automatic and semi-manual detection methods respectively. In the automatic method, a target is detected automatically by the image-processing dynamically for moving target in CCD with IR for bright target. For the performance evaluation, the proposed method is compared with a CUDA and normal image processing. The simulation results show that, the proposed method has good performance in term of total image processing time in the FCS.**

Keywords—Fire-Control System; Real-time Parallel image processing; Military Application; CUDA; OpenMP

I. INTRODUCTION

In a typical weapon system, an operator can observe a distant target by using a camera from a location that is safe from enemy attack, and the target is tracked by the FCS (Fire-Control System). Common examples of such a system that have been studied include Remote Weapon Systems and the remote-controlled weapon station(RCWS)[1, 2].

The image-processing technology of an FCS can be divided into two approaches: target detection and target tracking. In the case of the target detection technology, when a target is detected in an image, the operator is notified and given the target information. The target tracking Technology has the combined function of following the target and firing at it. At all times, a Gunner Primary Sight (GPS) which is another part of a FCS is automatically taking aim at the target. Hence, the speed of image-processing is an important factor in providing real-time information with which to control the base system.

Existing methods for improving image-processing speed are also tested. The first method is based on CUDA[3, 4], while the second method uses OpenMP[5, 6]. In the CUDA, the raw image-processing speed is relatively fast because of the use of a General Purpose GPU(GPGPU). However, the overall image-processing speed of the FCS is relatively slow because time is required to copy image data from the system memory to the GPU memory. In comparison with the CUDA, the overall image-processing speed of the FCS is faster with the OpenMP method because it only uses the system memory and a multi-core CPU.

In this paper, a parallel image-processing method is proposed to improve the processing speed of an FCS by the OpenMP. The processing times of the proposed method are compared with those of both the CUDA and Normal Image-Processing(NIP).

II. PROBLEM FORMULATIONS

Studies on defense-system equipment have been carried out for both the remote weapon system[1] and the remote-controlled weapon station(RCWS)[7, 8]. The remote weapon system provides safety for the human operator and correspond the emergency by using remote firing from inside a vehicle. Therefore, the safety and viability of the human operator are reasonably guaranteed. Such operations can be conducted in day, night, and unfavorable conditions. In this paper, a high-speed image-processing technology is proposed for application in both the Remote Weapon Systems and the RCWS.

The methods for separating a target from its background are different depending on whether an infrared(IR) camera or a charge-coupled-device(CCD) camera is used. The separation method for an IR camera is the one proposed by Otsu[9, 10]. In the Otsu algorithm, the pixel-value distribution of an image is treated as two normal distributions because the target and background have similar brightness distributions. So, the smallest value between the two normal distributions is selected. Furthermore, the optimal threshold value in this method is calculated if the two distributions are clearly distinguishable. In this paper, the threshold value is also calculated by the Otsu algorithm[9, 10] for detecting a bright target in the IR image.

By using an IR camera, the target heat is expressed in the image and so the Otsu method can be applied effectively. By using a CCD camera, the target color is expressed in an image, so the method of image difference[11] is used instead for target/background separation with a CCD camera. In this method, a moving object is detected by the difference between images n and n-1.

III. Fire-Control System Structure

In this paper, a parallel processing method based on OpenMP is proposed and tested for a FCS. It is intended to improve the image-processing of an FCS at the target-detection stage. The proposed method is implemented in the image-processing module as shown in Fig. 1, the details are described in Table 1.

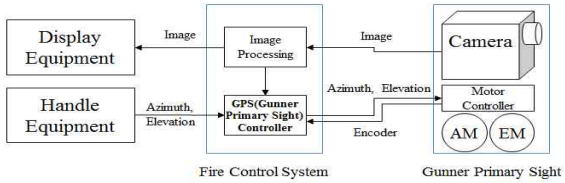


Fig. 1. Overall Architecture of Fire-Control System(FCS)

TABLE I. FCS Equipment Functions

Equipment	Module	Note
Display Equipment	-	- To provide an observed image from the GPS to an operator.
Handle Equipment	-	- To transmit an operator order to the FCS.
Fire-Control System	Image-Processing Module	- To detect and track the target using input image from the GPS camera. - To provide an image to the operator. - To transmit an azimuth and an elevation to the GPS Control Module.
	GPS Control Module	- To input elevation and azimuth information from an Image-Processing Module and a handle. - To calculate elevation and azimuth of real target using the input information for controlling the GPS motor.
Gunner Primary Sight (GPS)	Camera	- To acquire an area image for observation by an operator.
	Motor Control Module	- To control AM and EM by input data of an elevation and an azimuth.
	AM, EM	- An azimuth motor and an elevation motor

The Image-Processing Module of the FCS is divided into detection, tracking, and stabilization, as shown in Fig. 2. In the target-detection process, a target is automatically detected when it is moving. In the target-tracking process, a human operator indicates which target is to be tracked. However, it is impossible to detect and track a target if there are severe disturbances and shocks to the camera system; the stabilization technology can be used for solve this problem.

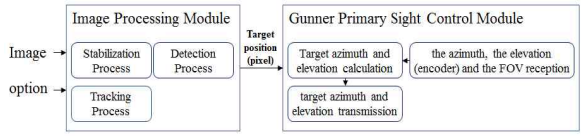
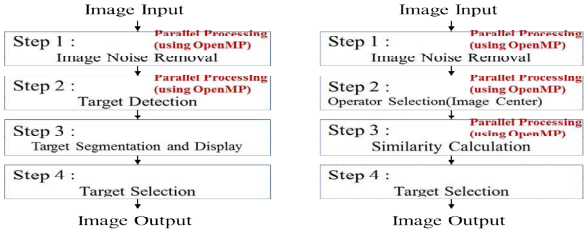


Fig. 2. Image-Processing & Control Module of FCS

The GPS control module of the FCS consists of three processes: AE&FOV, TA&EC, and TA&ET. In AE&FOV, an azimuth and elevation are received from the encoder, and the Field of View(FOV) is received from the camera. In TA&EC, the real target azimuth and elevation are calculated with the azimuth, elevation, and FOV information received from the encoder and camera, and with the position(pixel) value of a target determined by image-processing. In TA&ET, the azimuth and elevation that were calculated by TA&EC are transmitted to the motor-control module as shown in Fig. 1.

IV. Target Detection Method

The target-detection method of the FCS that is shown in Fig. 3 as two options: an automatic and a semi-manual target-detection method. In the automatic method, a target is detected automatically by the image-processing. In the other method, a human operator and machine detects the target manually. In this paper, an automatic target-detection method is described.



(a) Automatic Target Detection (b) Semi-Manual Target Detection

Fig. 3. Automatic and Semi-Manual Detection Methods

A. *Step 1: Image Noise Removal*

Various types of filter exist for removing noise, and the basic structures of these have similar structure. In this paper, the proposed method is intended to improve the processing speed. Therefore, a Gaussian filter(3×3mask) is used, which is a filter that is commonly used in image-processing.

B. *Step 2: Target Detection*

In the GPS, an IR(thermal) and a CCD camera are used for target detection. Different algorithms are used for target detection depending on the camera's characteristics(see Fig. 4). The IR camera is normally used at night; it produces a gray-scale image of the target temperature. Therefore, a method of applying an automatic threshold is likely to be effective in detecting the target. In the case of a CCD camera, it is a color image that is normally represented. Hence, applying a subtraction is more effective here than the automatic-threshold method.

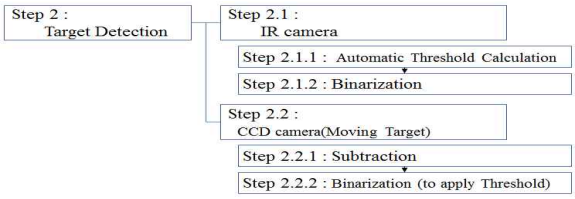


Fig. 4. The Target Detection Method on FCS

In the step 2.1(see Fig. 4), the automatic threshold and binary image are calculated by the Otsu threshold. Also, in the step 2.2, moving targets are detected by calculating the

difference between the first and nth image in a buffer of size n. The optimum value of n is sensitive to changes in the images; therefore, in this step, it is controlled by the operator.

C. Step 3: Target Segmentation and Display

In the target-detection step, when a target is detected, the resulting image and background colors are white and black respectively. When using the results, objects can be separated from their background, but the distinction between a target and other objects is ambiguous. Therefore, in this paper, Connected-Component Labeling(CCL)[13] is used to distinguish between a target and other objects.

D. Step 4: Target Prioritization

In the prioritization, as shown in Fig. 5, there are three possible methods by which a target can be selected: method 1 uses the smallest distance between the image center and objects, method 2 uses the smallest distance between the point selected by the operator and objects, and method 3 uses manual selection by the operator after objects have been automatically arranged in size order.

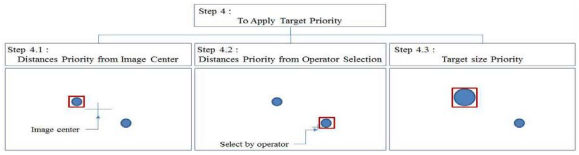


Fig. 5. Priority Calculation Methods for Target Selection

V. Performance Evaluation

In this paper, two experiments are used for validation of the proposed method. The first experiment compares the image-processing times of the CUDA, OpenMP, and NIP methods. The second experiment analyzes the processing time of the proposed method based on OpenMP.

A. Experimental Setup

The hardware experimental conditions were as follows: Windows 7 64-bit operating system, 4GB Ram, Intel i5 CPU (quad core) and the NVIDIA GeForce GT 620. Also, in the first experiment, General image-processing algorithms (Sobel, Subtraction, Template Matching) were used for measuring the image-processing speed. To test the Sobel operation, an image size of 1024 × 768 was used. To test the template matching and the subtraction, an image size of 720 × 480 was used; the template image size in the template matching was 50 × 50. The processing time was calculated 10 times in total using the system time.

B. Processing times: OpenMP vs CUDA and NIP

All the image-processing time results are shown Fig. 6. When the sobel algorithm was applied to CUDA, the average image-processing time(10 repetitions) is 44.5ms. In the CUDA, the subtraction and the template matching are 48.1ms and 1,911.8ms, respectively. When using the NIP method, the average time of the sobel algorithm, the subtraction and the template matching are 40.1ms, 8.8ms and 6,534.2ms, respectively. The average time of the sobel algorithm when using the OpenMP method is 11.4ms, the subtraction is 2.5ms, and the template matching is 2,030.3ms. When a higher time-complexity algorithm such as template matching was applied, the CUDA performed well. However, when a lower time-complexity algorithm such as the subtraction or the sobel algorithm was applied, the OpenMP method performed well.

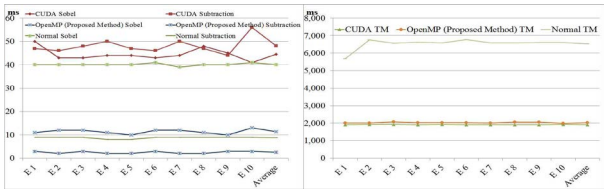


Fig. 6. Processing Time Comparison(CUDA, OpenMP, and NIP)

For CUDA, the only algorithm that showed better performance was template matching; the subtraction and Sobel took longer than NIP(by more than 39.3ms) and OpenMP(by more than 45.6ms).

C. Experimental results of target-detection algorithm using OpenMP

The image-processing time of the target-detection algorithm was measured as shown Fig. 7. The performances for Distance (the method of Step 4.1) and Size (the method of Step 4.3) were 15.16ms and 13.36ms, respectively. In the IR image, the performance for Distance was 15.53ms and the performance for Size was 16.11ms.

Fig. 7. Experimental Results of Image-Processing Time (Average of 30 frames; SC : Similarity Calculation)

VI. Conclusion

In this paper, a real-time image-processing method using OpenMP is proposed for improving the image-processing speed of a Fire-Control System(FCS). In order to provide real-time processing of Fire-Control System, the target detection and tracking information must be delivered to the base system within deadline using proposed scheme. The proposed method is compared with the existing methods of CUDA and NIP respectively.

Through the simulation and experimental tests, the image-processing time by using OpenMP is less than for the CUDA in terms of delay, computation complexity. The image-processing time per frame is less than 20ms for the OpenMP method.

The proposed method should be tested by measurements from a variety of experimental scenarios. As future works, one may consider to apply the target tracking in the FCS on moving platform such as drone.

ACKNOWLEDGMENT

This research was financially supported by the ITRC (Information Technology Research Center) support program (IITP-2017-H8601-16-1011), This research also was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2018R1C1B6001042).

REFERENCES

[1] Greene, B., "Remote Weapon Systems: The Next Generation", Land Warfare Conference 2010, Brisbane, Australia, November 15-19, 2010.

[2] B. S. Chauhan, Manvendra Singh, V. K. Sharma, P. C. Pandey , "Auto-video Tracking System: Performance Evaluation", Defence Science Journal, Vol. 58, No. 4, pp.565-572, July 2008.

[3] Victor Podlozhn , "Image Convolution with CUDA", NVidia CUDA white paper, 2007.

[4] Nathan Slegers, Andrew Brown, Jonathan Rogers, "Experimental investigation of stochastic parafoil guidance using a graphics processing unit", Control Engineering Practice, Volume 36, pp. 27-38, March 2015.

[5] Michael J. Quinn, "Parallel Programming in C with MPI and OpenMP", pp. 436-448MC Graw Hill, 2003.

[6] O. Rosén, A. Medvedev, T. Wigren, "Parallelization of the Kalman filter on multicore computational platforms", Control Engineering Practice, Volume 21, Issue 9, pp. 1188-1194, September 2013.

[7] Eugenio Po, "Turrets for AIFVs", Military Technology MILTECH, Dec. 2007.

[8] Sebastian Balos, Vencislav Grabulov, Leposava Sidjanin, "Future Armoured Troop Carrying Vehicles", Defence Science Journal, Vol. 60, No. 5, pp. 483-490, September 2010.

[9] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitive performance evalution", Journal of Electronic Imaging 13, pp. 146-165, 2004.

[10] Z. X. Li and S. W. Kim, "A Multi-thesholoding Approach Improved with Otus's Method", Journal of The institute of electronics engineers of korea, vol. 43, No. 5, pp. 29-37, 2006.

[11] Noriko Yoshiike, Yoshiyasu Takefuji, "Object segmentation using maximum neural networks for the gesture recognition system", Neuro computing, Vol. 51, pp 213–224, April 2003.

[12] Coeurjolly, D., Montanvert, A., "Optimal separable algorithms to compute the reverse euclidean distance transformation and discrete medial axis in arbitrary dimension", IEEE Trans. Pattern Anal. Mach. Intell. 29 (3), pp. 437–448, 2007.

[13] Shapiro, L., and Stockman, G., "Computer Vision. Prentice Hall", pp. 69–73, 2002.

[14] Torbert, Shane, Applied Computer Science(2nd ed.), p. 158, 2016.

[15] LIU, F., AND GLEICHER, M., "Automatic Image Retargeting with Fisheye-View Warping", In ACM UIST, pp. 153–162, 2005..