CUDA Based CAMshift Algorithm for Object Tracking Systems

Ji Hoon Jo, Sang Gu Lee
Department of Computer Engineering,
Hannam University
133 Ojung-dong, Daeduk-gu, Daejon
KOREA
invers83@nate.com, sglee@hnu.kr

Abstract: - In this paper, we present an image object tracking system for GPGPU based CAMshift algorithm. For image object tracking, we use the parallel CAMshift tracking algorithm based on the HSV color image distribution of detected moving objects. In this, RGB-to-HSV color conversion, image masking such as open and close operation for image morphology, and computing of centroid are executed in parallel. CAMshift algorithm is very efficient for real-time tracking because of its fast and robust performance. In this system, CUDA environment and C++ program are used for image processing and accessing the PTZ protocol and RS-485 communication for controlling the position of PTZ camera in order to arrange the moving object images in the middle part of the monitor screen. This system can be applied to an effective and faster image surveillance system for continuous object tracking in a wider area and real time.

Key-Words: - CUDA, Object tracking, PTZ camera, CAMshift algorithm, parallel processing

1 Introduction

The area of object tracking in the image processing is an attractive research subject and has many applications on the field of computer vision [1]. In this paper, we implement an image object tracking system for GPGPU based CAMshift algorithm. For this, we use CUDA environment and C++ program. PTZ cameras can control the pan, tilt and zoom operations of the camera lens through a surveillance DVR or computer system. PTZ cameras have the ability to moving right, left, up, down and even zoom. Generally, PTZ cameras cover very large areas compared with fixed cameras which would not be able to do. Therefore, recently PTZ cameras have been used in many applications for security operations of moving object tracking through manually operating or automatic control. In this paper, the main idea is that moving objects are positioned nearly in the middle part of the monitor screen.

In this paper, we use the CAMshift tracking algorithm based on the HSV color model image distribution of moving object. CAMshift algorithm is very efficient for real-time image tracking because of its fast and robust performance. In this system, we use C++ and CUDA kernels for detecting moving object and accessing the PTZ protocol and RS-485 communication for controlling the position of PTZ camera in order to put the position of the moving objects in the middle part of the window screen. The proposed system can be applied to an effective and fast image surveillance system for continuous object tracking in a wider area.

The organization of this paper is as followings. Section 2 tackles CAMshift algorithm for image segmentation. In section 3, the proposed system is discussed. In section 4, parallelization of the CAMshift algorithm is shown. Experimental results are discussed in section 5. And finally, section 6 draws a conclusion.

2 CAM shift algorithm

CAMshift (Continuously Adaptive Mean shift) algorithm is a modified version of the mean shift algorithm. CAMshift tracking adapts the size of the tracking window to the object using the knowledge of the aspect-ratio of the desired object and the 0-th moment of the kernel.

CAM shift tracking algorithm is as follows.

```
CAM –shift tracking( )
kernel = kernel_initialize( );
for  image_frame = 1: n
features = object_features(image_frame);
repeat
[$M_{00}$ $M_{01}$ $M_{10}$]  = moments(kernel, features);
Cw = centroid(kernel);
Cg = [$M_{01}$ / $M_{00}$ , $M_{10}$ / $M_{00}$];
window_shift(kernel, Cg – Cw);
kernel_size(kernel, width($M_{00}$), height($M_{00}$));
until  |Cg – Cw| <= e;
end
```

Here, M_{00}, M_{01}, and M_{10} mean 0-th moment, first moment(x) and first moment(y), respectively.

3 Proposed System

In the proposed system, object images are captured from PTZ camera. These images are transferred to PC via image grabber board, and parallel CAMshift algorithm extracts moving objects and eliminates the back-ground image. Then, the pre-processing procedures such as filtering and morphological computations (erosion, dilation, open and close operations) are performed. C++ program sends the packets such as P, T and Z data to the PTZ camera using RS232 to RS485 converter. In the monitor, the moving objects are displayed in the middle part of the window screen. Fig. 1 shows the entire hardware structure in this system. We use SPD-1000 PTZ Dome camera (Samsung) [4]. A frame image size of 640 x 480 image data is used. We get 30 frames each second.

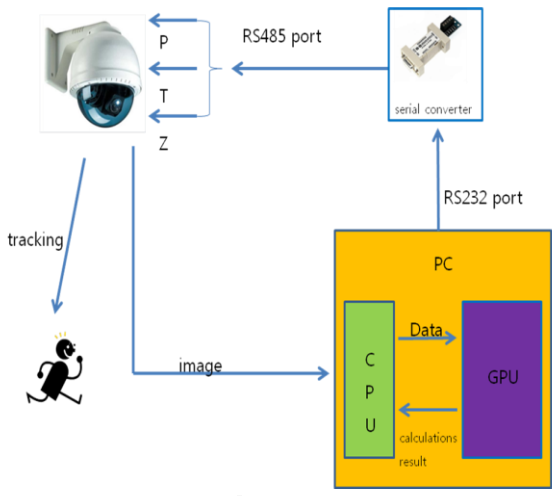


Fig. 1 Hardware system

Table 1 shows the packet format and protocol for controlling the step motors of PTZ camera. For example, if we send the packet in the Table 2, pan of PTZ camera is moving to right direction in speed 34.

Table 1. Command packet format

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
STX	Cam ID	Host ID	Cmd-1	Cmd-2	Data 1
Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	
Data2	Data 3	Data 4	ETX	Check-sum	

STX: A0H, ETX: AFH

Check-sum = ~((byte2 + byte3 …+ byte9) & FFH)

Table 2. Protocol sample

command	protocol	
Pan Right	A0 01 00 00 02 22 00 00 00 AF DA	speed 34

We designed and implemented a serial converter circuit. Fig. 2 shows RS232 to RS485 converter hardware circuit.

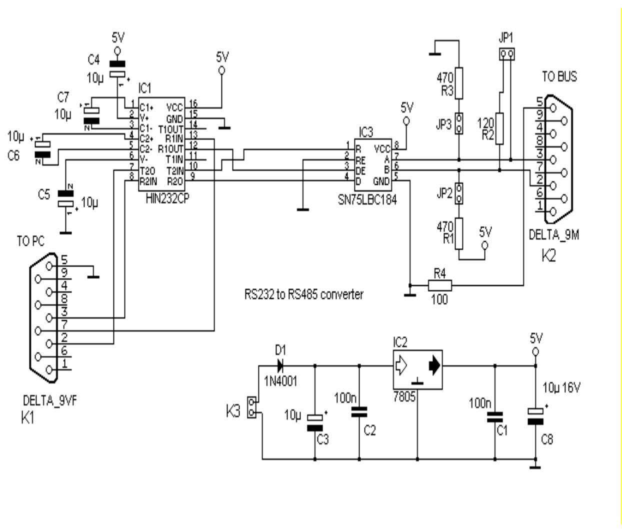


Fig. 2 RS232 to RS485 converter hardware

4 Parallelization of the CAMshift algorithm

The procedures of parallelization of CAMshift algorithm are as followings as

① Parallel processing of a conversion of RGB color into HSV

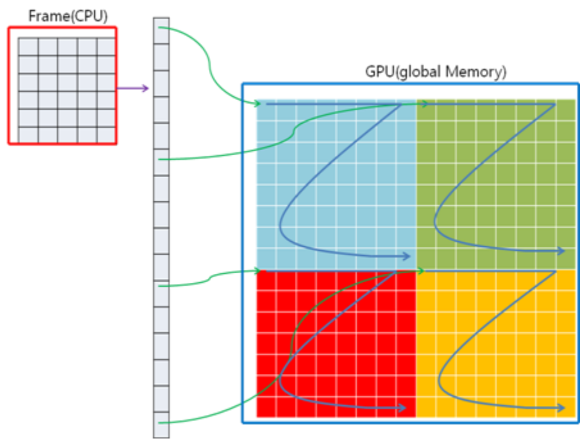


Fig. 3 Parallelization of RGB– HSV conversion

① Parallel processing of image masks such as open and close operations for image morphology as shown in Fig 4.

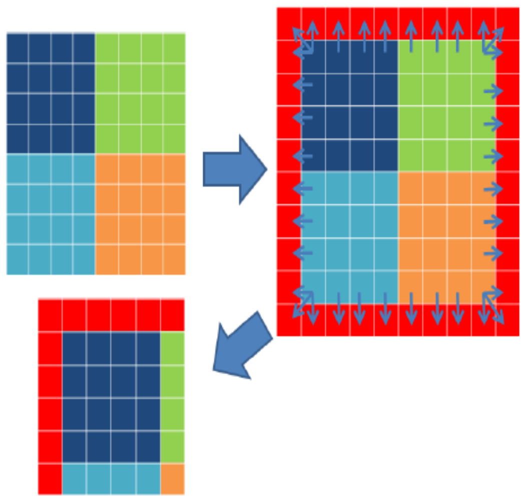


Fig. 4 Parallelization of image masks

② After taking each Hue value of pixel in the HSV color domain, getting the range of threshold between 0.8 and 1.0.

③ For the centroid of ROI, M00, M01, and M10 are computed. This procedure is composed of summation of vectors and summation of weighted multiplication of vectors. Figure 5 and Figure 6 show the mechanism of summation and reduction procedure, respectively.

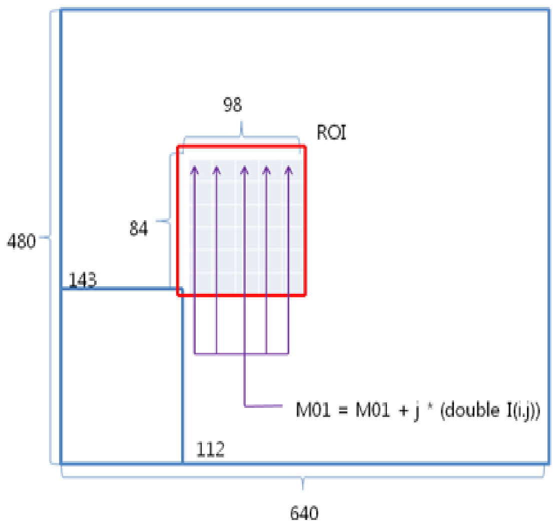


Fig. 5 M_{01} moments calculations

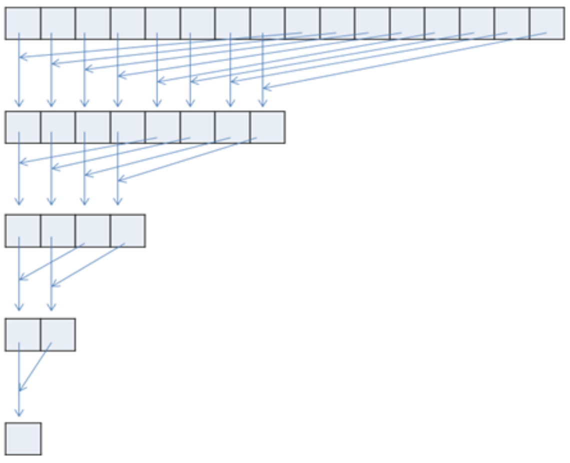


Fig. 6 Reduction procedure

④ After computing the weighted sums in GPU and transferring these values to CPU, calculate the centroid and decide the convergence condition.

5 Experimental Results

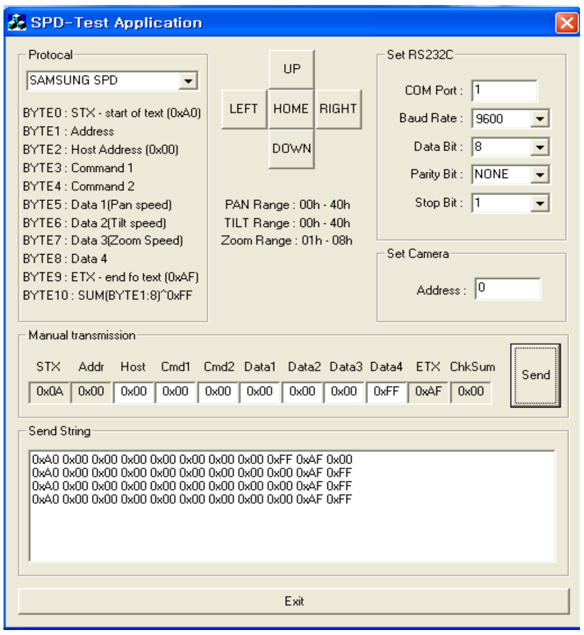
Experimental development environment is as followings as Table 3.

Table 3. Development environment in PC

	Development environment
Hardware	CPU: Pentium (R) D 3.00GHz Memory: 2GB Graphic card : GTX 560
OS	Microsoft Windows 7
Software tool	Microsoft Visual Studio 2008, C++ NVIDIA CUDA Tool kit 4.2 CUDA visual profiler

Fig. 7 shows a GUI window display to transfer packets for controlling motors of PTZ camera for pan, tilt and zoom operation ranges and directions (up, down, left and right).

Fig. 8 shows the flowchart of this system. Fig. 9 shows the object frame images for camera tracking in the PTZ dome camera.



Protocal: SAMSUNG SPD

```
BYTE0 : STX - start of text (0xA0)
BYTE1 : Address
BYTE2 : Host Address (0x00)
BYTE3 : Command 1
BYTE4 : Command 2
BYTE5 : Data 1(Pan speed)
BYTE6 : Data 2(Tilt speed)
BYTE7 : Data 3(Zoom Speed)
BYTE8 : Data 4
BYTE9 : ETX - end fo text (0xAF)
BYTE10 : SUM(BYTE1:8)^0xFF
```

PAN Range : 00h - 40h
TILT Range : 00h - 40h
Zoom Range : 01h - 08h

COM Port : 1
Baud Rate : 9600
Data Bit : 8
Parity Bit : NONE
Stop Bit : 1

Address : 0

STX	Addr	Host	Cmd1	Cmd2	Data1	Data2	Data3	Data4	ETX	ChkSum
0x0A	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0xFF	0xAF	0x00

```
0xA0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xAF 0x00
0xA0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xAF 0xFF
0xA0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xAF 0xFF
0xA0 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xAF 0xFF
```

Fig. 7 GUI window display

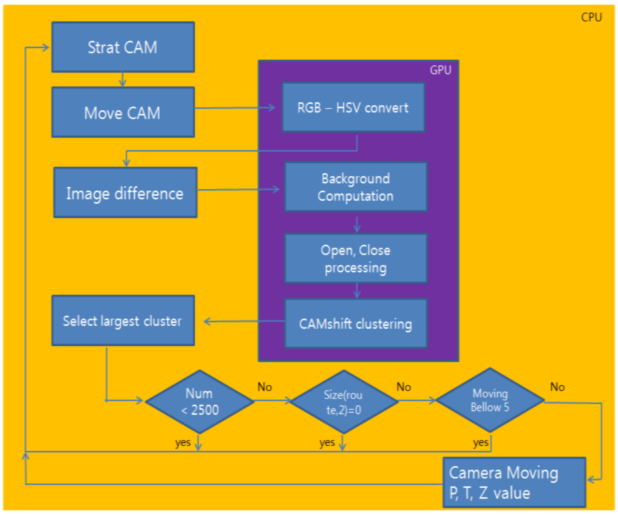
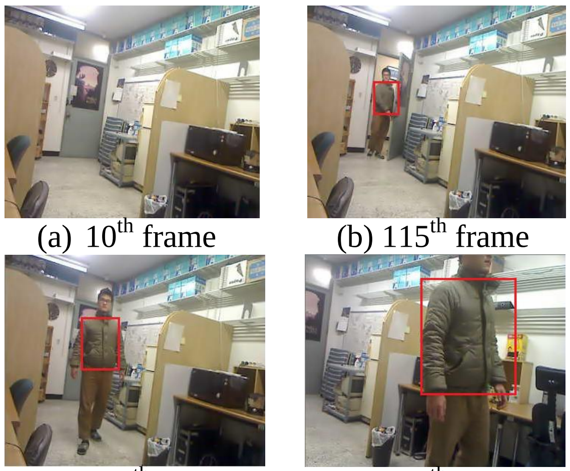
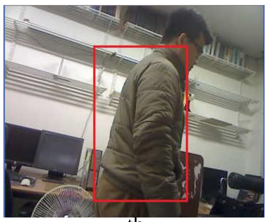


Fig. 8 Flowchart



(a) 10th frame (b) 115th frame

(a) 168th frame (d) 199th frame


(e) 231th frame

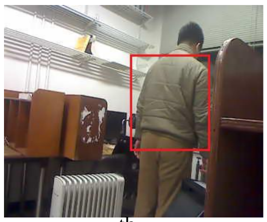

(f) 276th frame

Fig. 9 Object tracking frame images

Table 4. Values of pan, tilt and zoom

Frame #	Pan	Tilt	Zoom
10th	316	13	1.0
115th	271	15	1.0
168th	246	19	1.0
199th	192	24	1.0
231th	159	22	1.0
276th	114	18	1.0

Table 4 shows each value of pan, tilt, and zoom, respectively given in the Fig. 9.

Table 5. speedup of the CPU and GPU

	CPU	GPU (CUDA)	speedup
RGB-to-HSV	12.87	0.75	17.16
Histogram generation	1.81	0.2	9.05
Probability backprojection	1.62	0.18	9
Total	16.31	1.14	14.3

Table 5 shows the performance evaluation of the proposed CAMshift algorithm of each frame in CPU and GPU, respectively. The size of image resolution is 640 x 480. As shown in Table 4, we got the speedup of about 17 times in RGB-to-HSV operation. In the operations of histogram generation and backprojection, we also had the speedup of about 8 ~ 10 times faster.

5 Conclusion

We have implemented an image object tracking system for PTZ cameras using parallel CAM shift algorithm in CUDA environment. By CUDA kernels and C++, we controlled the PTZ camera and positioned moving object nearly in the middle part of the screen. Parallel CAM shift algorithm showed fast, stable and robust performance. In this paper, we gained the speedup of about 17 times in RGB-to-HSV conversion. In the operations of histogram generation and backprojection, we also had the speedup of about 8 ~ 10 times faster as compared as CPU processing. For the future study, we will develop a new tracking system that can be utilized in the multi-GPU board in order to speed-up the total performance and to build that system in a stand-alone prototype. As a further research, we will also implement several extensions to the system's ability for an integrated smart surveillance camera system having multiple cameras with networked system.

Acknowledgement

This work was supported by Security Engineering Research Center, granted by the Korea Ministry of Knowledge Economy.

References:

[1] M. Sonka, V. Hlavac and R. Boyle, Image Processing, Analysis, and Machine Vision, 3rd ed., Thomson, 2008

[2] D. Comaniciu and P. Meer, "Mean shift: A Robust Approach Toward Feature Space Analysis", IEEE tr. on PAMI, Vol.24, no.5, pp 603-619, May 2002

[3] P. Li and L. Xiao. Mean shift parallel tracking on GPU. In H. Arajo, A. M. Mendona, A. J. Pinho, and M. I. Torres, editors, IbPRIA, volume 5524 of Lecture Notes in Computer Science, pages 120–127. Springer, 2009.

[4] Samsung Techwin, PTZ Dome Camera SPD-1000 Manual, 2010

[5] S. Huang; and J. Hong; , "Moving object tracking system based on CAMshift and Kalman filter," International Conference on Consumer Electronics, Communications and Networks (CECNet), pp.1423-1426, 2011.

[6] W. Xiangyu, and L. Xiujuan, "The study of moving target tracking based on Kalman-CAMshift in the video," 2nd International Conference on Information Science and Engineering (ICISE), pp.1-4, 2010.