

Speed Performance Improvement of Vehicle Blob Tracking System

Sung Chun Lee and Ram Nevatia

University of Southern California, Los Angeles, CA 90089, USA
sungchun@usc.edu, nevatia@usc.edu

Abstract. A speed performance improved vehicle tracking system on a given set of evaluation videos of a street surveillance system is presented. We implement multi-threading technique to meet the requirement of real-time performance which demanded in the practical surveillance systems. Through multi-threading technique, we can accomplish near real-time performance. An analysis of results is also presented.

1 Introduction

Detection and tracking of vehicles is important for traffic analysis and video surveillance. Our objective is to develop techniques for automatic vehicle tracking to enable the image analysts to cope with exploiting the exponentially growing volume of video imagery available to them.

There are several factors that make this task highly challenging: camera and scene instabilities. Firstly, sometimes, cameras shake, creating false foreground detections and automatic gain control abruptly changes the intensity of the video image causing multiple false detections. Secondly, there exist ambient illumination changes such as due to passing clouds and occluding objects such as trees and traffic signs as illustrated in Fig. 1.



Fig. 1. Illustration of camera and scene instabilities: (a) A white car (top right) occluded behind a tree and a traffic sign, (b) Intensity of scene dramatically changes due to automatic camera gain change as clouds pass.

In our previous work [5], we have tried to address these issues and described a robust detection and tracking vehicle system. One of limitations of the work is slow computational performance. Practical surveillance applications demand real time processing performance as an important factor. We applied a multithreading programming approach to improve the computational performance which is the focus of this paper.

2 Vehicle Tracking and Detection

Before explaining the speed enhancement, we briefly describe the detection and tracking algorithm from the previous work.

The task is to evaluate the performance of our vehicle detection and tracking algorithm on a set of surveillance videos provided by the VACE project [2]. Objective includes accurate detection, localization and tracking while maintaining the identities vehicles as they travel across different frames.

The videos are of street scenes captured by cameras mounted at light pole heights looking down towards the ground. There is one road running from top to bottom of the image and another one from left to right near the top of the image. Provided videos are from two different cameras at several different times.

We assume vehicles move on a ground plane; we use vehicle motion and vanishing points from scene features to compute an approximate camera model [4]. This process is performed once, in training phase. To distinguish vehicles from walking humans and other motion, we set a minimum size for an object to be considered as a vehicle; this size is set in 3D, its size in image is computed by using the camera parameters.

2.1 Background subtraction

A common way to detect moving objects has been to learn a pixel-wise model of the background and to flag significant variations from it as a moving object occludes the pixel. This approach works well when the camera is stationary. We have used this method to detect and track vehicle blobs in surveillance videos.

We use an adaptive background subtraction method [4]. We first detect temporal color changes in the input video stream and classify as corresponding to moving objects. Then, we filter out isolated points and cluster the detected pixels to form foreground objects. The background model is updated at each frame. In an ideal case, every extracted foreground object would correspond to one vehicle object. However, there might be other cases: merged vehicles, split vehicle, or other moving objects (e.g. human).

2.2 Vehicle tracking and Detection

Our tracking method processes the frames sequentially. We analyze the extracted foreground blobs from background subtraction process as described in Section 2.1. At each new frame, we do following:

1. Apply tracking object's dynamic model to predict its new position
2. Generate an association matrix based on the overlap between the predicted object rectangle and the detected blob rectangle
3. In the association matrix, there are following cases:
 - (a) If the track-blob match is one-to-one, we simply update the position of the track.
 - (b) If a blob has no match with current tracking objects and its size is comparable to a regular vehicle, a new tracking object is created.
 - (c) If there is no blob match for more than a certain number of frames, the track of this object ends.
 - (d) If an object matches with multiple blobs, we combine the split blobs into one to match with the object being tracked.
 - (e) If multiple objects merge into one blob, we segment the blob based on the appearance model of each involved object. Specifically, we apply a meanshift color tracking method [1] to locate the vehicle in the merged blob.

Some example results are given below in Fig. 2.



Fig. 2. Results of vehicle tracking and detection (Yellow box: DCO (Dont Care Object), Green box: Detected ground truth, White box: Detected system output, Blue box: Missed ground truth, Black box: Detected but overlapped with DCO, Magenta box: Occluded ground truth (DCO), Cyan box: Stationary ground truth (DCO))

3 Computation Time Improvement

One of the limitations in our approaches was that computation performance was up to 10 times slower than real-time. Our analysis indicated that the adaptive background subtraction task is the most time consuming process in our system. One alternative is the use of a General Processing Unit (GPU); GPUs are very powerful and relatively inexpensive. However, GPU programming can be difficult. Instead, we have chosen to explore the use of multi-processors available in a modern workstation (dual-core dual processors are now common, quad-core machines should become available shortly).

3.1 Multithreading Background Subtraction

A *thread* is a unit of execution in an operating system. A process (or application program) consists of one or more threads and the scheduler in the operating system controls thread task distribution. In our system, the user can control the number of threads and the system divides input stream image into small piece of images based on the number of threads as illustrated in Fig. 3.

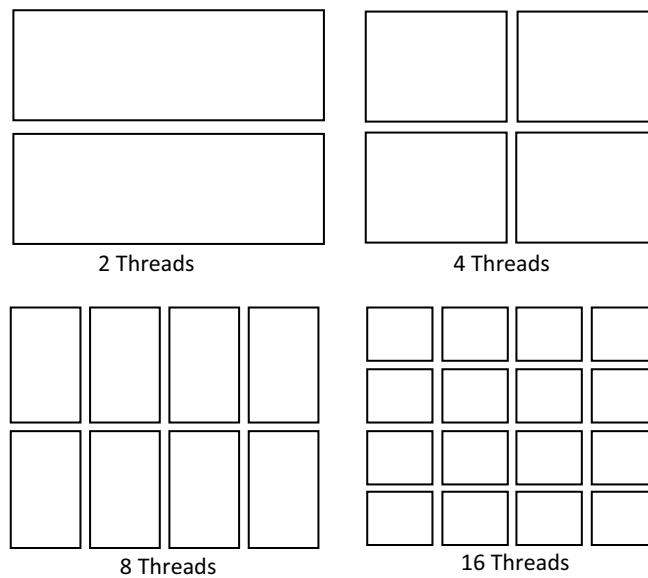


Fig. 3. Image division based on the number of threads.

Then, the system distributes each image region to the threads to perform the adaptive background subtraction operation for smaller size images. Lastly, the system collects and combines the results of background subtraction after all threads finish their tasks. The procedural description of the multi-threaded

adaptive background subtraction algorithm is shown below:

Capture a video frame and do the followings:

1. Divide the frame image into multiple images with configuration of Fig. 3 according to the number of threads (user's input)
2. Distribute the segmented image to each thread and dispatch threads to perform the adaptive background subtraction task.
3. Wait until all threads have finished their tasks.
4. Combine all pieces of background subtracted images into one image.

Our preliminary experiments indicate that we can achieve a speed up of between 2.5 and 4 by using various number of threads. The main limitation seems to be in limited Cache memory that is shared between two processors in dual core chip.

4 Experiments and Results

In 2006, we tested our systems on the videos provided by the VACE project. The size of each frame is 720×480 . The experiments were finished on a regular PC with Intel Pentium 2.6GHZ CPU. The average processing time was 2.85 (frame per second). By using a multithreading background subtraction, We are able to achieve average of 11.9 (frame per second) as shown in Table 1.

Num of Threads	Background Subtraction Speed with Multithreading (second / frame)	Tracking Speed (second / frame)	Final Speed (second / frame)	Final Speed (second / frame)
1	0.200	0.020	0.220	4.55
2	0.110	0.020	0.130	7.69
4	0.085	0.020	0.105	9.52
8	0.064	0.020	0.084	11.90

Table 1. Computation Time using multithreading skill with Intel Xeon CPU with two dual core 64 bit 3.73GHZ CPUs.

We quantitatively evaluated our system according to the requirements of the test process. The metrics shown evaluate both the detection and tracking performances. We summarize the metric definitions below; more details may be found in [2].

1. MODP (Multiple Object Detection Precision) measures the position precision of single frame detections;
2. MODA (Multiple Object Detection Accuracy) combines the influence of miss detections and false alarms;
3. MOTP (Multiple Object Tracking Precision) measures the position precision at tracking level;
4. MOTA (Multiple Object Tracking Accuracy) is MODA at tracking level with consideration of ID switches.

Table 2 lists the scores on 50 test video sequences. One observation from the table is that: the difference of MODP and MOTP, or MODA and MOTA is very small for all the test video sequences. It is mainly because the penalty on object ID change is relatively small. Actually, there is a difference in the number of ID changes in the outputs of different subsets. However, the current defined MOTA and MOTP are not able to reflect this tracking error very well.

Average MODP	Average MODA	Average MOTP	Average MOTA
0.602	0.679	0.616	0.640

Table 2. Evaluation scores on 50 test video sequences.

5 Conclusions

We have presented evaluation results of the detection and tracking accuracy and precision and the speed performance improvement of our vehicle detection and tracking system on a provided set of surveillance videos. The data contains many highly challenging features. The performance of our system is promising though many shortcomings exist and shows near real-time operation.

Acknowledgements: This research was funded by VACE program of the U.S. government.

References

1. D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift *IEEE Conference on Computer Vision and Pattern Recognition*, 1:511-518, 2001.
2. R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, M. Boonstra, and V. Korzhova. Performance Evaluation Protocol for Face, Person and Vehicle Detection Tracking in Video Analysis and Content Extraction (VACE-II) CLEAR - Classification of Events, Activities and Relationships [http://www.nist.gov/speech/tests/clear/2006/CLEAR06-R106-EvalDiscDoc/Data and Information/ClearEval_Protocol_v5.pdf](http://www.nist.gov/speech/tests/clear/2006/CLEAR06-R106-EvalDiscDoc/Data%20and%20Information/ClearEval_Protocol_v5.pdf).
3. Liyuan Li, Weimin Huang, Irene Y.H. Gu, and Qi Tian. Foreground Object Detection from Videos Containing Complex Background *ACM Multimedia*, 2003.
4. Fengjun Lv, Tao Zhao and Ramakant Nevatia. Self-Calibration of a Camera from Video of a Walking Human *Proceedings of IEEE International Conference on Pattern Recognition (ICPR)*, 2002.
5. Xuefeng Song and Ram Nevatia. Robust Vehicle Blob Tracking with Split/Merge Handling *CLEAR'06 Evaluation Campaign and Workshop in conjunction with FG'06* 2006.