Moving Object Detection in Traffic Videos using Motion Vectors

Georgios Bardas, Stergios Poularakis and Ioannis Katsavounidis
Department of Electrical and Computer Engineering
University of Thessaly
Greece

***Abstract*— In this work, we propose a novel approach for detection and tracking of moving objects in 2D video sequences, based on Motion Estimation. Our method modifies Connected Component Analysis, grouping image macroblocks based on both spatial and motion characteristics. Computational efficiency is achieved using fast motion estimation techniques as well as additional optimization techniques on the configurable Tensilica Xtensa processor core.**

Keywords— object detection; tracking; motion vectors

I. INTRODUCTION

Recently, automatic traffic surveillance became an active area in Computer Vision, including applications such as traffic statistics extraction, law violations detection and car accident spotting. Current solutions use background subtraction or frame difference for a rough vehicle detection and then refine the first segmentation result based on prior or learnt knowledge about vehicle's characteristics, such as shape and size. Another core element of such systems is vehicle tracking, i.e. constructing vehicle's trajectory over consecutive video frames.

In this work, we propose a novel approach for detection and tracking of moving objects in 2D video sequences, based on fast motion estimation techniques [1]. These methods describe motion characteristics (direction and magnitude) of small image parts (macroblocks) using motion vectors. Subsequently, a modified Connected Component Analysis (CCA) method labels similarly-moving neighbouring macroblocks as belonging to the same object. Besides standard computer implementation, we also experimented with the configurable Tensilica Xtensa processor core, which resulted into significant computational improvement.

The remaining of this paper is organized as follows. In Sec. II we provide a short overview of some recent approaches. In Sec. III we present in detail our approach for object detection and tracking while in Sec. IV we present our performance optimization procedure and experimental results. Finally, Sec. V concludes this paper and addresses our plans for future work.

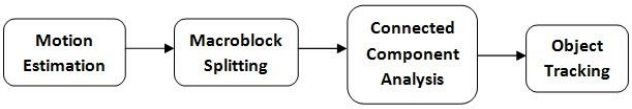

Fig. 1. General structure of our proposed system.

II. RELATED WORK

Recently, Kostia [2] detected vehicles during night-time by finding vehicle headlights and tracked them using a Kalman filter. In a similar work, Chen, Wu, Huang and Fan [3] detected both headlights and taillights, achieving real-time performance on a TI DM642 DSP-based embedded platform. Kafai and Bhanu [4] used background subtraction to detect vehicles and then recognized vehicle's type (*Sedan, Pickup truck*, *SUV/Minivan* and *unknown*), based on Hybrid Dynamic Bayesian Networks. Liu, Lu and Xu [5] determined the best vehicle detection frame by finding abrupt intensity changes on video frames, which resulted in traffic-scene-invariance. Zhou, Gao and Zhang [6] performed adaptive background subtraction and then classified the macroblocks of significant luminance change as vehicle or not vehicle parts, using Support Vector Machines (SVMs). Pan, Guo and Men [7] combined background subtraction and edge information to detect the road lanes, which allowed for easier vehicle counting. Li et al. [8] detected vehicle-parts through template matching and then used a part-based model for the final vehicle detection.

III. OUR APPROACH

The general structure of our approach is shown in Fig. 1. Below we explain in detail all four components of our system.

A. Motion Estimation

Our method begins by splitting the video frames into smaller blocks of size $p \times p$, better known as *macroblocks*. A very important property of most videos is that a macroblock at frame n (*current frame*) and position (x,y) will most probably be very similar to a macroblock at frame $n-1$ (*reference frame*) and position $(x - \Delta x, y - \Delta y)$, where Δx, Δy represent a small translation. Thus, we can define the *motion vector* of a macroblock as:

$$v = (\Delta x, \Delta y) \quad (1)$$

Although there can be many possible motion vectors for each macroblock, one usually keeps only the one minimizing the Sum of Absolute Differences (SAD):

$$SAD(\Delta x, \Delta y) = \sum_{x=0:p-1} \sum_{y=0:p-1} ||curr(x,y) - ref(x - \Delta x, y - \Delta y)|| \quad (2)$$

where *curr* and *ref* denote the current and reference frames respectively.

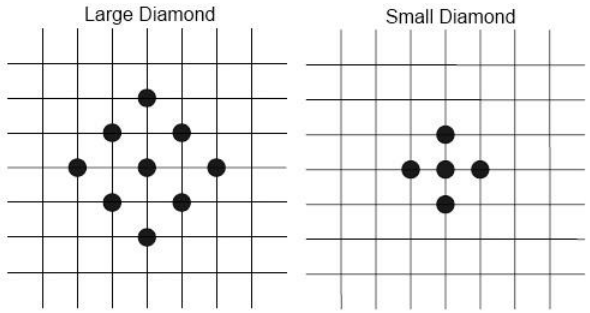

Fig. 2. The small and big diamond patterns used in PMVFASTalgorithm [1]. Black circles denote the points where SAD will be computed.

Since exhaustive evaluation of SAD for all possible translations (*Full Search*) is computationally expensive, we decided to use Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) of Tourapis, Au and Liou [1], which finds a very good solution (although sometimes not the best) at a reasonable time. Inspired from the Diamond Search (DS) algorithm [9], PMVFAST searches specific points in a diamond-shaped area (Fig. 2) and moves towards the direction of current minimum SAD, until the minimum is found in the center. Moreover, PMVFAST uses some heuristics to end the search process earlier than DS, while it also supports two diamond patterns, one big and one small (Fig. 2), which result in a better, i.e. lower SAD solution, in less operations. In our implementation, we further improved computational performance through Intel's SSE intrinsics [10], which allow for Single Instruction, Multiple Data (SIMD) computations on massively stored data, such as images and videos.

B. *Macroblock splitting*

The exact number of macroblocks at each frame depends on parameter *p*. In our approach, we used $p=16$, which is a typical choice in most popular video compression standards, such as MPEG2 [11] and H.264 [12]. However, a fixed-*p* analysis can only offer a rough view of the true motion field and moving objects. For example, a 16x16 macroblock may contain one moving object along with some static background, or even two distinct moving objects. For such reasons, we split some macroblocks to four smaller 8x8 *sub-macroblocks*, based on specific criteria, and recompute their motion vectors (Fig. 3).

The first criterion splits a macroblock if there exists a wide range in the values of the four partial SADs, corresponding to the 4 quadrants, as shown on Fig. 3 (e.g. if $S_1 = 4\ S_3$).

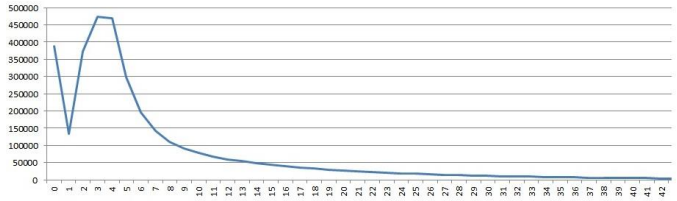

Fig. 4. Distribution of residual variances for the ten VQEG videos [13].

Fig. 3. 16x16 SAD computation based on four 8x8 SADs.

The second criterion examines macroblock's *residual*, defined as the difference between current and reference macroblock. Intuitively, we can expect that the residual of a double-object macroblock will present high variance σ^2 in its values, since the two objects contribute values of different ranges. Thus, we split only the macroblocks whose variance exceeds a predefined threshold T_σ. More specifically, we ran PMVFAST on 10 widely used videos from the Video Quality Experts Group (VQEG) collection [13] and chose $T_\sigma = \mu + 3\sigma$, where ($\mu=6$, $\sigma=4$) denote the mean and standard deviation of the variances distribution, as shown in Fig. [4]. In our experiments we noticed that combining the two criteria performed better than applying them separately. One can repeat this process to smaller-sized blocks, such as 4x4, or even 2x2 blocks. In our work, we limited the smallest block to be 4x4.

C. *Connected Component Analysis*

This step merges neighbouring macroblocks to form connected components corresponding to moving objects. Two neighbouring macroblocks M_z, M_w with motion vectors *z*,*w* are merged if $D(z, w) \leq T_v$, where

$$D(z, w) = ||z_x - w_x|| + ||z_y - w_y|| \quad (3)$$

and T_v is a small threshold. In our experiments, we used $T_v = 10$. Finally, we ignored small noisy objects that have a small number of pixels, for example those consisting of less than 10 small, i.e. 4x4, blocks.

A standard morphological filter, such as dilation, is performed to smooth-out resulting objects and close apparent "holes" in the moving object.

D. *Object Tracking*

According to Hu, Tan, Wang and Maybank [14], *"The goal of object tracking is to determine the position of the object in images continuously and reliably against dynamic scenes"* [15]. Tracking typically outputs the object's trajectory through time, which allows for basic or advanced inference in traffic videos, such as checking for red-light violations, speed control [16] and detection of dangerous driving attitude.

In this work, we use a simple and efficient tracking method, which associates two objects of consecutive frames *current* and *future*) based on their spatial proximity and motion similarity. The top *K* candidate *future* objects are further evaluated and we choose the object maximizing the overlap with *current* object. This process is repeated for all objects of the current frame.

(a) (b)

Fig. 5. (a) A frame from a traffic video, showing a car moving towards the camera. (b) The moving objects found by our approach, shown in different colors.

(a) (b)

Fig. 6. (a) A car detected in a traffic video. Note the macroblocks of zero motion vector on the object's surface. (b) Correction by painting the whole car as a rigid object through morphological dilation.

IV. IMPLEMENTATION

For our experiments, we performed parameter estimation using the ten VQEG videos [13], and tested our system on various traffic videos downloaded from YouTube. We implemented and optimized our approach both on a standard PC system and a Tensilica Xtensa processor core, as explained in sections IV-A and IV-B respectively.

A. Optimization on a standard PC system

We implemented our approach in C/C++ using Microsoft Visual Studio ™ 2010, executing on a standard PC system. Choosing PMVFAST for Motion estimation is known to provide a computational speed-up of 100x up to 2000x over Full Search, depending on the video characteristics [1]. In our implementation, we noticed a speed-up of around 120x for traffic videos. An additional improvement of 4x4 was achieved using Intel's SSE2 intrinsics, which support fast vector operations. Specifically, we split a 16x16 macroblock to four 8x8 sub-macroblocks, as shown in Fig. 3, compute the partial SADs using the *_mm_sad_epu8* intrinsic [17], and combine the four SADs into the final SAD.

B. Optimization on Tensilica Xtensa

We further optimized our implementation on Tensilica Xtensa ™, *"a configurable, extensible and synthesizable processor core for embedded System-On-Chip (SoC), focusing on design through the processor and not through hardwired RTL"* [18]. Typical development procedure on Tensilica Xtensa processor involves optimizing software and hardware repeatedly, until a target goal is achieved, as shown in Fig. 7. Using Tensilica Instruction Extension (TIE) language, a developer can define new CPU instructions (TIEs) and use them directly in software as built-in functions, usually for the most elaborate tasks *(Hotspots)*. Computational improvement comes from three main optimization techniques [19]:

- *Fusion*. Multiple processing operations can be fused into one instruction, e.g. addition and shift to the right, required when computing the average of two integers.
- *SIMD/Vector Transformation*. Similar to Intel's SSE, one can define TIEs that perform the same processing operation on multiple data of same type (Single Instruction - Multiple Data). Such optimizations are very frequent in image and video processing, where pixels are conveniently aligned in continuous memory addresses.
- *FLIX*. Flexible Length Instruction Xtensions (FLIX) allows designer to merge multiple unrelated operations in one instruction, thus offering great flexibility.

In this work, we used TIEs to speed-up the following tasks:

- *SAD*. Computing SAD between two numbers requires 4 clock cycles on a standard implementation (1 fetch, 1 addition, 1 subtraction and 1 absolute value), i.e. 4 x 256 = 1024 clock cycles for two 16x16 macroblocks. Using SIMD, we computed the SAD between two lines of 16 pixels simultaneously, which required 4 x 16 = 256 clock cycles (16 fetch, 1 additions, 1 subtractions and 16 absolute values). Finally, using FLIX, we combined fetch, addition, subtraction and absolute value in one TIE, which required 16 clock cycles and resulted in a total improvement of 1024/16 = 64x. Multiple processing operations can be fused into one instruction, e.g. addition and shift to the right, required when computing the average of two integers.
- *Residual between 16x16 macroblocks*. This computation requires initially 512 memory loads, 256 subtractions and 256 memory stores, i.e. 1024 cycles. Using SIMD and FLIX, we achieved 32x speed-up.
- *Variance of residual values*. Using similar optimizations, we achieved 46x speed-up.

V. DISCUSSION AND FUTURE WORK

In this work we proposed a novel approach for object detection and tracking on traffic videos. Our method performs fast Motion Estimation techniques (PMVFAST) and then combines neighbouring image macroblocks based on their spatial and motion features. We thoroughly optimized the execution performance of our approach, both on a standard PC system, using SIMD and PMVFAST, as well as on a Tensilica Xtensa configurable processor core, achieving real-time performances. Our goals for future work include extensive experiments on standard Computer Vision datasets, as well as further improvements of detection and tracking accuracies.

ACKNOWLEDGMENT

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: ARCHIMEDES III.

Investing in knowledge society through the European Social Fund.

REFERENCES

[1] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) – enhancing block based motion estimation," in SPIE Conf. Visual Communications and Image Processing, pp. 883–892, 2001.

[2] R. Kostia, "Night-time traffic surveillance: A robust framework for multi-vehicle detection, classification and tracking," in 6th IEEE Int'l Conf. on Advanced Video and Signal Based Surveillance, pp. 1–6, 2009.

[3] Y. L. Chen, B. F. Wu, H. Y. Huang, and C. J. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," IEEE Trans. on Industrial Electronics, vol. 58, no. 5, pp. 2030–2044, 2011.

[4] M. Kafai and B. Bhanu, "Dynamic bayesian networks for vehicle classification in video," IEEE Trans. on Industrial Informatics, vol. 8, no. 1, pp. 100–109, 2012.

[5] Yan Liu, Xiaoqing Lu, and Jianbo Xu, "Traffic scenes invariant vehicle detection," in 9th Asian Control Conf. (ASCC), pp. 1–6, 2013.

[6] J. Zhou, D. Gao, and D. Zhang, "Moving vehicle detection for automatic traffic monitoring," IEEE Trans. on Vehicular Technology, vol. 56, no. 1, pp. 51–59, 2007.

[7] X. Pan, Y. Guo, and A. Men, "Traffic surveillance system for vehicle flow detection," in 2nd Int'l Conf. on Computer Modeling and Simulation, vol. 1, pp. 314–318, 2010.

[8] Y. Li, B. Tian, B. Li, G. Xiong, F. Zhu, and K. Wang, "Vehicle detection with a part-based model for complex traffic conditions," in IEEE Int'l Conf. on Vehicular Electronics and Safety (ICVES), pp. 110–113, 2013.

[9] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," IEEE Trans. on Image Processing, vol. 9, no. 2, pp. 287–290, 2000.

[10] Intel Corporation, "Intel 64 and ia-32 architectures software developers manual," September 2013,

http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-softwaredeveloper-manual-325462.pdf.

[11] ISO/IEC JTC1/SC29/WG11 13818, "Information technology-generic coding of moving pictures and associated audio," Nov. 1994.

[12] Rec. ITU-T H.264 ISO/IEC 14496-10, "Advanced video coding for generic audiovisual services," June 2011.

[13] "Video Quality Experts Group (VQEG) YUV sequences," http://www.vqeg.org/.

[14] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 34, no. 3, pp. 334–352, 2004.

[15] H. Zhou, Y. Yuan, and C. Shi, "Object tracking using SIFT features and mean shift," Computer Vision and Image Understanding, vol. 113, no. 3, pp. 345 – 352, 2009.

[16] "A real-time computer vision system for vehicle tracking and traffic surveillance," Transportation Research Part C: Emerging Technologies, vol. 6, no. 4, pp. 271 – 288, 1998.

[17] "Microsoft MSDN," http://msdn.microsoft.com."Microsoft MSDN," http://msdn.microsoft.com.

[18] Tensilica, "Xtensa LX - Product Brief," http://www.tensilica.com.

[19] Tensilica, "Xtensa LX Microprocessor - Overview Handbook," 2004, http://www.tensilica.com.