Real-time Detection of Moving Vehicles

R. Cucchiara[1], M. Piccardi[2], A. Prati[1], N. Scarabottolo[3]

[1] DSI, University of Modena, Via Campi 213/b - 41100 Modena, Italy
[2] DI, University of Ferrara, via Saragat 1 - 44100 Ferrara, Italy
[3] Polo Didattico e di Ricerca di Crema - University of Milano, Italy

Abstract

Computer vision-based traffic flow monitoring is of major importance for enforcing traffic management policies. Information such as the number of vehicles passing on a road per time unit, or vehicles' turning rates at intersections are exploited by traffic management policies to supervise traffic-light timings.
Computer vision-based traffic flow monitoring requires extraction of moving vehicles from traffic scenes in real time. To accomplish this task, efficient algorithms must be used and effective, low-cost hardware implementation must be pursued. This paper first describes the algorithms used in VTTS *(Vehicular Traffic Tracking System*) to achieve segmentation of moving vehicles. Then, hardware implementation on a re-programmable FPGA-based board is described in detail.

1. Introduction

Traffic flow monitoring based on computer vision aims to extract information about the traffic flow from traffic scenes acquired with cameras. This information is required to substantially support traffic management policies with regularly updated data such as the number of vehicles passing on a road per time unit, vehicles' turning rates at intersections, queue length measurement, and many others.

Computer vision is only one of the possible approaches to extract traffic flow data: other approaches have been followed based on inductive loops, sound detectors, pneumatic sensors. Nevertheless, none of these sensors is able to extract information as detailed as computer vision is.

Consider for instance the problem of traffic monitoring at intersections. The intersection is defined by a set of input and output ways: turning rates and average crossing time for each valid combination of input and output ways are required in order to achieve a dynamic description of traffic flow at the intersection. Therefore, each single vehicle must be tracked while crossing the intersection.

Vehicle tracking requires the analysis of image sequences, in order to trace the evolving position of single vehicles. Most approaches are based on motion extraction from images: on one hand, many analytical methods of motion analysis have been adopted, like the optical flow [1], or the matching of moving features [2,3]; these methods are able to extract not only moving points, but also information on the motion direction and velocity, but are generally very expensive in terms of computational time. For example, a real-time motion detection system is proposed in [4], running on a HyperSparc workstation equipped with a Datacube MV200. On the other hand, real-time and low-cost requirements demand approximated, more efficient solutions. Low-cost, real-time systems exist, capable of performing analysis of pixel strips for measuring vehicular flow, called Inductive Loop Emulators [5]. However, these systems - likewise physical loops - are not able to track single vehicles. Our proposal is based on a low-level module, performing vehicle segmentation with simple algorithms (easily portable in hardware) on the whole image, and a high-level module performing vehicle tracking. One major goal is to provide a hardware solution to the problem of *detecting moving objects* in outdoor scenes in real time.

Therefore the low-level module definition has to cope with these issues: *simple processes*, in order to be able to develop a cheap and reliable hardware implementation; *real-time issues*, to achieve frame-rate or real-time processing, as required by the application; *flexible* approach, where system reconfigurability could be exploited; *on-site implementation* without the need for installation of large cumbersome systems in conjunction with monitoring sites. To achieve all these goals, a prototype module based on SRAM-based Field Programmable Gate Arrays (FPGAs) has been developed.

The approach we follow, oriented to traffic monitoring and also surveillance applications, extracts moving points from images with simple image processing techniques, then segments them in order to identify and locate moving objects; this information will be used by the tracking module [6, 7]. In particular, we adopt a spatio-temporal filter consisting of the integration of motion information from sequences of frames with the information of grey-level variation in each single frame.

In the next section we describe the architecture of the Vehicular Traffic Tracking System (VTTS). In section 3, we outline the algorithms proposed for extracting moving objects (vehicles, in the context of traffic monitoring); and in section 4 we describe the hardware solution based on reconfigurable-hardware. Finally, we present performance results of the developed prototype.

2. The Vehicular Traffic Tracking System

Moving objects in outdoor scenes can be perceived by an observer since both motion and luminance contrast concur to pop the object shape out of the background. While this is quite true for daytime images, it is also well known that, in night illumination condition, luminance contrast and motion are not as easy to perceive [8].

Accordingly, we have defined two different sets of image analysis algorithms for extracting vehicles from image sequences acquired in such different illumination conditions. In daylight, information on motion and high gradient points are exploited jointly; instead, at night features such as headlights must be detected and associated with vehicles; headlights must be discriminated from reflections, beams, street-lamps and horizontal signals such as zebra crossings. Fig. 1 shows two frames under different luminance conditions: Fig. 1a is acquired under daylight, while Fig. 1b at night.

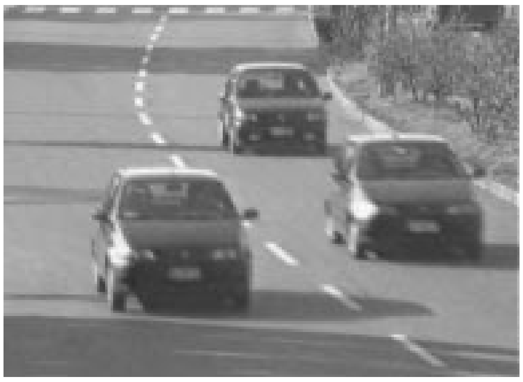

Fig. 1a. Example images in day conditions

In the architecture of VTTS (Vehicular Traffic Tracking System) two different modules for daytime and night vehicle detection are present. The algorithms for daytime vehicle extraction are [7]:

1) detection of moving points by performing a difference on three consecutive frames;

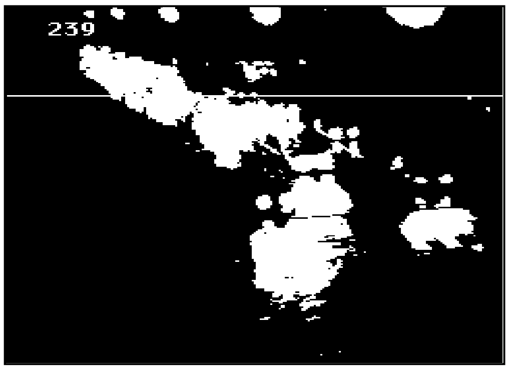

Fig. 1b. Example images in night conditions

2) detection of high contrast points in the image, i.e. points with high gradient, as possible edges of moving objects;
3) execution of a Moving Edge Closure, that is a morphological closure between moving points and sharp edges in order to extract moving objects.

In the literature, there exist at least three different kinds of efficient time-differential algorithms to extract moving points from image sequences: *difference with background*, *two-frame difference* and *three-frame difference* (otherwise called *double-difference*). In the first, object motion with respect to background is exploited; in the other two, object motion with respect to previous positions is computed, based on the hypothesis that some object points overlap in two consecutive frames. We have compared these algorithms, and then chosen the one based on the difference of three consecutive frames [7, 9]. We adopted this method since in our tests we have found it particularly robust to noise due to camera movements and able to avoid detection of very small moving objects in the scene.

At night, the approach differs substantially: headlights must be separated from beams, but their motion vectors are similar.

Thus, the algorithms for vehicle extraction at night are not based on motion detection, but mainly aim to identify headlight shapes in each frame [8]. The algorithms are:

1) image thresholding (the bimodal histogram allows easy separation of objects from background);
2) template matching on the image with a headlight template, scaled in accordance with the image region;
3) cross-correlation on headlight pairs, with correlation parameters scaled in accordance with the image region.

The last step aims at making vehicle detection more robust while discriminating vehicles with respect to motorbikes, since they have different impact on the traffic flow and different relevance for traffic management strategies.

The objects extracted from the low-level modules are finally classified as moving vehicles according to some heuristic rules and a rule-based tracking system is proposed. The system has been tested on several image sequences from traffic scenes, and assessed good performance: typical precision rates in vehicle detection for the low-level modules range from 88% to 91% [8], while the high-level tracking module is able to compensate low-level errors and nearly zeroes erroneous detection.

While the high-level module reasons on relatively few symbolic data, the low-level modules must perform vehicle segmentation at very high speed, in order to meet real-time requirements of applications. Therefore, most or all frames must be processed.

From day-time requirements arises the need for a system able to segment moving objects at frame rate: the aforementioned three steps take about one second on a high performance PC, with images already stored in main memory, thus without considering any transfer time for the frame grabber acquisition. This time, even if it is not high in itself, is still far from real-time requirements. Moreover, in many real applications, the adoption of a ruggedized, industrial PC is not affordable for many reasons, first of all, cost: in many distributed applications, such as road traffic control, a suitable solution should be to equip all traffic-lights with an intelligent camera able to detect and measure the vehicular flow.

Finally, an alternative could be the real-time transfer of all frames from the road station to a possible processing center: but also this solution is not affordable for the current costs of bandwidth (for example, in the traffic control system installed and running in Bologna, Italy, cameras transfer rates are of 0.2 frames/s only).

3. Moving vehicles detection algorithms

In this section, we outline the algorithms used to perform vehicles segmentation in detail, pointing out the day-time condition.

To explain the double-difference operator, we can consider the sequence of binary images $\{I_m\}$: the difference-image D_m is defined as:

$$D_n(i,j) = \left| I_n(i,j) - I_{n-1}(i,j) \right|$$

The *double-difference image* is obtained by performing a logical AND between pixels belonging to two subsequent difference-images, thresholded by a threshold T:

$$\mathrm{DD}_n(\mathrm{i},\mathrm{j}) = \begin{cases} 1 & \text{if } (\mathrm{D}_{n+1}(\mathrm{i},\mathrm{j}) > \mathrm{T}) \wedge (\mathrm{D}_n(\mathrm{i},\mathrm{j}) > \mathrm{T}) \\ 0 & \text{otherwise} \end{cases}$$

We observed that this operator is quite immune to noise, due to the non-repeatability of noise in three subsequent frames. Moreover, the double-difference image detects motion on the central frame I_n, where the image luminance gradient will be computed.

VTTS performs motion detection and luminance variation detection in parallel. In order to obtain the spatial-temporal extraction of the targets, both of these data have to be joined. We propose a new operator called *MEC* (*Moving Edge Closure*): a pixel of the image is labelled as moving if its gradient magnitude (∇I_n) is high enough and if a pixel previously labelled as *moving* belongs to the pixel's neighbourhood :

$$\mathrm{MEC}_n^0(\mathrm{i},\mathrm{j}) = \begin{cases} 1 & \text{if } \mathrm{DD}_n(\mathrm{i},\mathrm{j}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{MEC}_n^r(\mathrm{i},\mathrm{j}) = \begin{cases} 1 & \text{if } \mathrm{MEC}_n^{r-1}(\mathrm{i},\mathrm{j}) = 1 \vee (\nabla \mathrm{I}_n(\mathrm{i},\mathrm{j}) \geq \mathrm{T}_G \\ & \quad \wedge \ \exists (\mathrm{k},\mathrm{l}) \in \mathrm{X} \mid \mathrm{MEC}_n^{r-1}(\mathrm{k},\mathrm{l}) = 1 \\ 0 & \text{otherwise} \end{cases}$$

where T_G is the value of the threshold applied onto the gradient image and X is the pixel's neighbourhood.

Nevertheless, an approximation of the MEC operator suited for hardware implementation is required. In fact, as shown in the definition of the operator (reported above), MEC needs a recursive algorithm to be performed. As a matter of fact, raster-scan algorithm able to achieve real-time behaviour can not fulfil recursive algorithm. Therefore, MEC algorithm has been partitioned in two steps: direct scan and reverse scan. In the former, the image is analysed from the upper left corner to the lower right, while in the latter from the lower right to the upper left. This approximation is effective, except when a strongly concave object is present: in such a case some contours of the moving object could not be caught. Nevertheless, this is quite a rare situation in real image sequences under commonly used points of view.

A further drawback of the approximation introduced is the wrong link established between close objects connected by a stationary area with high gradient (such as shadows). To overcome this, MEC approximation has to be reviewed.

4. Hardware implementation

The proposed approach outlines two different abstraction levels characterised by a very different computational load, data structure and cost/speed requirements. While the high level tasks do not exhibit heavy workloads, handling a limited amount of symbolic data (the vehicles in the scene), low-level computation works on very large amount of data (image frames) and has to perform computation possibly at frame rate. For this reason we propose a dedicated hardware solution based on SRAM Field Programmable Gate Arrays (FPGA) for providing low-level computation. The selection of FPGA technology is due to the following issues:

1) implementing special-purpose logic with FPGAs is a very common approach in industrial contexts, since they provide low-cost reliable and easy-to-develop integrated solutions;
2) FPGAs represent now a good trade-off between the great resource availability of ASICs and the flexibility of DSP-based solutions. This feature is increased by current development tools supporting high-level hardware description (in particular, VHDL has been adopted).
3) SRAM-based FPGAs (e.g., of Xilinx 4000 family) enables run-time programming. In such a manner, different hardware functions can be downloaded and changed as a function of the external luminance condition.

The first issue is absolutely essential for a possible spreading of the final product (we could think of a low-level module installation for each traffic-light station).

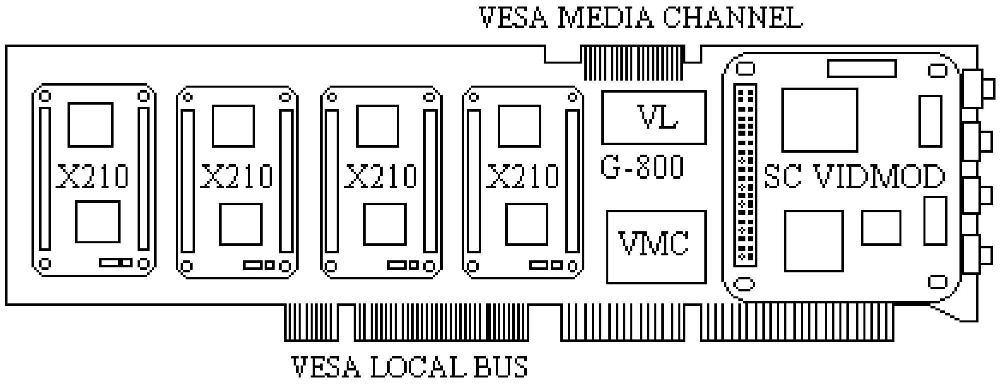

Fig. 2. The GigaOps board

The second point concerns in particular the development phase: in fact, porting the image processing tasks from software to hardware by means of high-level hardware description languages is straightforward. Lastly, the third consideration is particularly suitable for the VTTS approach: the final goal is to switch between low-level processing modules when luminance condition changes, by reprogramming the hardware at run-time. To this aim, we performed the experiments using the GigaOps G800 Spectrum board, a very powerful hardware programmable board [10], designed for imaging and video processing. The structure of G800 is schematically shown in Fig. 2: it contains a programmable device for designing the interface with standard PAL cameras and video display (in the right part of the figure), up to eight FPGA-based programmable modules (four, in the board used for the experiments) and a bus interface with standard PC. Each programmable module is composed of two programmable Xilinx FPGAs and 8 Mbytes of RAM memory. Consequently to the flexibility of this platform we can code different "hardware programs" in VHDL language, compile them (Synopsys Tools are used [11]) in order to create a netlist and finally a bitstream that can be downloaded into FPGAs at execution time by means of the PC interface. In a few milliseconds, the board is able to change its hardware functionality and thus switch between hardware programs. In this working environment, we have implemented the algorithms for day-time vehicle detection, and tested different versions of differential algorithms for the moving points extraction with two-frame difference, three-frame-difference and difference with background approaches [9] in order to compare the execution time and efficacy under various external conditions.

As well as moving point extraction, we perform concurrently edge detection and moving edge closure in real-time.

The resource availability of the board allow efficient parallelization of tasks in order to perform all computation at frame rate.

In particular we exploit task parallelism by implementing edge and motion detection in parallel on two different processing blocks (i.e., two FPGAs) and a further synchronization stage in another block that executes the MEC algorithm. Finally, a morphological closure is performed by a fourth FPGA.

Table 1 reports the global FPGA resource allocation for the prototype. It is possible to outline that the Edge Detection module has the highest computational load (expressed by the equivalent gate count). In fact edge detection is a near-neighbour operator requiring many flip-flops for internal storage of the pixels involved in convolution (expressed in terms of logic blocks, called CLBs). However, this module does not require multiple accesses to external memory banks, thus limiting the IOBs (Input/Output Blocks) required.

On the other hand, the Double Difference operation needs a lower number of logic resources, but more IOBs to interface the internal logic resourches with the two memory banks, in order to store the two previous sequence frames.

5. Performance evaluation of the hardware solution

The performance upper-bound is the execution of all the vehicle detection steps at frame-rate. Nevertheless, the detection rate should be adequate to the moving objects' speed: in order to catch motion of vehicles driving from 40 to 70 km/h, as normal in urban traffic, we consider one frame every five.

More particularly, the European PAL video standard provides a frame every 40 ms: catching one frame every five and using three frames for performing the double-difference operator, we spend 200 ms to obtain input information and other 40 ms for sending frames onto the output display.

All operations, from acquisition to moving edge closure are executed during input acquisition and therefore *are hidden by the I/O time*.

Moreover, the optional morphological closure needs additional 160 ms, that we have also hidden in a pipeline implementation. Thus real-time requirements are completely fulfilled.

Fig. 3 shows one example frame of four possible video outputs of our system: the original color sequence, without any processing (Fig. 3a), the moving points (Fig. 3b), the edge points (Fig. 3c) and the *moving objects* (Fig. 3d), obtained with MEC algorithm and one step of morphological closure.

Therefore, real-time processing is achieved as well as a severe compression of the images (from colour pixels to 1-bit/pixel images) keeping only the important information about motion, while at the same time requiring very limited bandwidth.

This work is part of a project held with the Bologna Provincia government for a city control center with vision based traffic monitoring [12].

6. Conclusion and future work

In this paper, we have presented the Vehicular Traffic Tracking System (VTTS), and the implementation of day-time vehicle detection by using FPGAs. The system acts as a complete urban traffic monitor, able to track vehicles, to count/classify vehicles (for special applications, such as reserved lanes for buses, which need vehicle classification) and to extract extra-information such as turning rates, position and length of queues, and others.

In [7], the algorithms for the day-time condition and the high-level tracking module have been presented, while [8] describes the algorithms for vehicle detection at night.

.

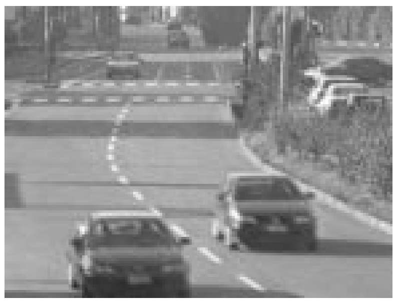

Fig. 3a. Original image

Fig. 3b. Moving points image

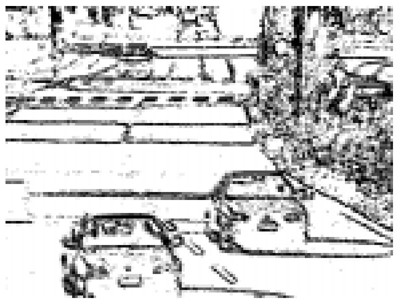

Fig. 3c. Edge points image

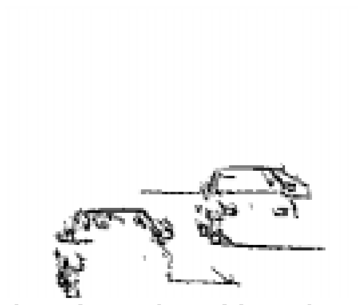

Fig. 3d. Moving objects image

This paper describes the hardware implementation of the day-time low-level module in detail. Results prove the high performance that can be achieved with use of FPGAs. Experiments performed show that vehicle detection under different light conditions requires very different image processing algorithms depending on the different visual cues that have to be detected. This analysis suggests exploiting a reconfigurable system able to adaptively change vehicle detection algorithms.

References

[1] Beauchemin, S. S., Barron, J. L., “The computation of optical flow”, ACM Computing Surveys, 27(3), 433-466, 1995.

[2] Tomasi, C., Kanade, T., “Detection and tracking of point features”, Technical Report CMU-CS-91-132, Carnegie Mellon University, Apr. 1991.

[3] Smith, S. M., Brady, J. M., “Asset-2: real-time motion segmentation and shape tracking” IEEE Trans. PAMI, 17(8), 814-820, 1995.

[4] Liu, H., Hong, T., Herman, M., Chellappa, R., “Motion-Model-Based Boundary Extraction and a Real-Time Implementation”, Computer Vision and Image Understanding, 70(1), 87-100, 1998.

[5] Fathy, M., Siyal, M. Y., “Real-time measurement of traffic queue parameters by using image processing techniques”, IEE Proc. - Image Processing and its Applications n. 410, 450-453, 1995.

[6] Koller, D., Weber, J., Huang, T., Malik, J., Ogasawara, G., Rao, B., Russel, S., “Towards Robust Automatic Traffic Scene Analysis in Real-Time.”, Proc. Int’l Conf. Pattern Recognition, 126-131, 1994.

[7] Barattin, M., Cucchiara, R. , Piccardi, M., “A Rule-based Vehicular Traffic Tracking System”, Proc. of CVPRIP’98, First International Workshop on Computer Vision, Pattern Recognition and Image Processing, NC, US, 1998.

[8] Cucchiara, R., Piccardi, M., “Vehicle Detection under Day and Night Illumination”, to appear on Proc. of ISCS-IIA99, Special Session on Vehicle Traffic and Surveillance, Genoa, Italy, 1999.

[9] Yoshinari, K., Michihito, M., “A human motion estimation method using 3-successive video frames”, Proc. of Int. Conf. on Virtual Systems and Multimedia GIFU, 135-140, 1996.

[10] Giga Operations Corporation. SPECTRUM™ Reconfigurable Computing Platform – Documentation, Release 3.01.

[11] Synopsis, FPGA Compiler User Guide v 3.5, Mountain View (US), 1996.

[12] “Intelligent transportation systems tackle the year 2000 traffic challenge in Bologna” Ertico News n.4. 1999, 4-6.