A Dual-Stage Robust Vehicle Detection and Tracking for Real-time Traffic Monitoring

Jorge Batista, Paulo Peixoto, Catarina Fernandes and Miguel Ribeiro
ISR-Institute of Systems and Robotics
Department of Electrical Engineering and Computers
University of Coimbra, Coimbra, Portugal

***Abstract*— This paper addresses the important problem of detecting and tracking vehicles in outdoor dynamic scenes as part of a real-time traffic surveillance system. The proposed solution is based on a dual-stage approach, using a pixel-level stage to extract foreground object from background scenes and a block-level stage to detect and track vehicles. The pixel-level stage combines a multi-background modelling with a dynamic thresholding, using a low-scale quasi-connected-components as a first stage for image object grouping/cleaning. The block-level performs a 8x8 block-region analysis defining a block energy function that is used to label the blocks belonging to different vehicles and track them over a stack of images. This approach has proven to be very helpful for occlusion reasoning. The proposed solution has the ability to overcame some of the most difficult problem that arise in outdoor scenes such as illumination variations, shadow-casts and waving movement resulting from trees and camera vibration. The performance and robustness of the proposed algorithm is shown using real highway traffic monitoring situations.**

I. INTRODUCTION

In order to support safe and efficient driving, it is important to classify the behaviors of vehicles and to understand their interactions on typical traffic scenarios. Until recent years, this task was performed by human operators at traffic control centers, but the huge increase on the number of available cameras requires automatic traffic surveillance systems ([24], [22], [3], [23], [21], [4], [6], [8]).

In the last decades, one of the most important efforts in ITS research has been the development of visual surveillance systems that could help reduce the number of traffic incidents and traffic jams in urban and highway scenarios. Although the large number of systems based on different types of sensors and their relative performance, vision based systems are very useful to collect very rich information about road traffic.

The work presented on this paper is part of an automatic traffic surveillance system. The primary goal of the system is to detect and track potentially anomalous traffic events along the highway roads. By anomalous events we mean the detection of vehicles that stopped on the highway, vehicles driving in lane's opposite direction and also vehicle that are constantly switching between lanes. The system should be able to identify each vehicle and track its behavior, and to recognize dangerous situations or events that might results from a chain of such behaviors. The system must be robust to illumination changes and small camera movements, being able to robustly track vehicles against occlusions and crowded events.

*This Work was supported by BRISA-Auto Estradas de Portugal S.A.

To achieve these goals, a dual-stage approach was adopted, using a pixel-level approach to extract foreground objects from background scenes and a block-level to robustly detect and track vehicles.

The pixel-level combines a multi-background modelling with a dynamic thresholding, using a low-scale quasi-connected-component as a first stage to group pixels into potential vehicles ([19], [18]). One of the major problems to overcome at the pixel level is the existence of shadow-casts induced by the vehicles themselves, which tends to merge the detected blobs of nearby vehicles and the glares induced on the surfaces of the highway (pavement and limit protection barriers) by vehicle's night light system ([22], [5], [7], [10]). A simple and efficient approach has been implemented in order to minimize these type of problems.

The pixel-level stage delivers a set of object blobs that are validated and tracked at the block-level stage. At the block-level stage the image is divided into 8x8 pixel blocks. One vehicle label is assigning to each block classified as a vehicle. For the labelling process the algorithm uses a block-energy function [21] that is based on four different parameters: block-motion vectors, block-texture matching, neighbor-blocks consistency and pixels-block overlapping. Occlusion reasoning and tracking are obtained combining the labelling process with a Kalman filter tracker using a novel and simple algorithm to manage vehicle's image grouping. With this approach, the system is able to track vehicles even in the case of a partial occlusion, which happens quite often in traffic scenarios, specially with low-angle oriented cameras or with cameras mounted on the side of the road.

The algorithm has proven to work robustly under different illumination conditions and camera noise, being used in a real highway scenario.

II. PIXEL-LEVEL STAGE

A. Background/foreground detection

Background/foreground detection is an essential component of most video surveillance systems involving object

detection and tracking. Such systems require, at the same time, both robustness against illumination changes and computation feasibility. Most of the existing solutions either make strict assumptions about the scene, or fail to handle abrupt illumination changes resulting from moving clouds or camera automated gain control (AGC).

Existing background modelling methods can be classified as either single-layer or multi-layer. Single layer methods obtain a model for the color distribution of each pixel based on past observations. Usually a single gaussian is considered to model the statistical distribution of a background pixel being updated through a α-blending approach. Depending on the value of α, either the foreground objects may prematurely blended into the background, or the model becomes unresponsive to the observations.

Stauffer and Grimson [2] modelled the background with a mixture of Gaussian models (MoG). Rather than explicitly modelling the values of all pixels as one particular type of distribution, the background is obtained by a pixel-wise mixture of Gaussian distributions to support multiple backgrounds. MoG are adaptable to illumination changes with the huge advantage of handling multiple backgrounds. However, their performance deteriorates when the scene to be described is dynamic and exhibits non-stationary properties in time. Outdoor lighting variations are complex and MoG doesn't prove to be robust to sudden illumination changes that may results from moving sparse clouds.

B. Multi-layer Background Modelling

In real outdoor scenarios, like a highway, the environment is highly dynamic and the detection solution must be tailored to robustly detect non-cooperative targets. To achieve this goal, a multi-layered and adaptive background modelling was adopted using an approach based on three background image models. Out of these three models, two of them are used to model the dynamics of the background allowing the system to cope with intensity variations due to illumination changes or noise and fluttering objects moving on the scene. The third background image is used in the cleaning/validation process, being a direct copy of a past image.

For a certain time step t, let the primary background be represented by B_p and the secondary background by B_s. To reduce computational cost, at the end of each time step, the primary background holds the pixel values that are closest to the current image. If this is not the case at the current time step, pixels are swaped between the two background images to make this likely to be the case in the next time step. The background model with smaller difference D_i ($D_i = I^t - B_i^t,\ i = p,s$) is represented by B_c and the other background by $B_{\bar{c}}$.

Representing the RGB color values of a pixel by $I(\phi)$ being $\phi = (x,y,c)$, the background images $B_i(\phi)_{i=p,s}$ are initialized using a set of T consecutive object free images, such that

$$\left.\begin{array}{l} B_p(\phi) = min\{I^t(\phi), t = 1..T\} \\ B_s(\phi) = max\{I^t(\phi), t = 1..T\} \end{array}\right\} if\ I^t(\phi) \in N_t$$

The background images updates depends on feedback from upper level stages, and it is implemented recursively in a fully automatic way, using:

$$B_c^{t+1}(\phi) = \begin{cases} \eta_s I^t(\phi) + (1-\eta_s) B_c^t(\phi) & \text{if } \phi \in T_t \\ \eta_f I^t(\phi) + (1-\eta_f) B_c^t(\phi) & \text{if } \phi \in N_t \end{cases} \quad (1)$$

$$B_{\bar{c}}^{t+1}(\phi) = B_{\bar{c}}^t(\phi) \quad (2)$$

where ϕ is the pixel index, I_t is the current frame, η_s is the integration factor of a pixel classified as a target (T_t) which is much smaller than the integration factor of a pixel classified as nontarget (N_t), η_f. The labelling of a pixel as being in the target set or in the nontarget set is carried on at the block level stage. The use of different integration factor for pixels belonging to target set and nontarget set will help the system to adapt more slowly the potential target regions. However, the presence of ghosts, i.e., false target regions due to statics objects belonging to the background image (e.g., cars) which suddenly start to move, results in longer false alarms. This problem is solved using the third background image, increasing the integration factor for that region if the target region is present in that background image model.

Background updating is called at the final processing step of a frame, so it has access to both the final label image L and the original thresholded difference image $D_{th}(\phi)$. This thresholded difference image is obtained combining background subtraction with a dynamic thresholding.

C. Dynamic thresholding

Four different thresholds are used: the per-pixel threshold (T_{pp}), the low threshold (T_l), the high threshold (T_h) and the global threshold (T_g).

The T_{pp}, attempts to account the scene noise at a pixel through time, e.g., when the camera shakes, the edge pixels will have a significant intensity change; in these cases, we will increase the T_{pp}. The T_g is the minimum value to eliminate camera noise and depends on the scenario. T_g is initialized using the difference of the initial background images

$$T_g(\phi) = \left| B_p(\phi) - B_s(\phi) \right|. \quad (3)$$

The algorithm computes the per-pixel difference between the image and both background images, and if the smaller difference exceeds the low threshold T_l, i.e.,

$$\min_i = \left| I^t(\phi) - B_i^t(\phi) \right| > T_l(\phi) \quad (4)$$

the pixel is considered as active (potential target pixel). A target is a set of connected active pixels such that a subset of them verifies

$$\min_i = \left| I^t(\phi) - B_i^t(\phi) \right| > T_h(\phi) \quad (5)$$

where T_h is a high threshold. The low threshold T_l is the sum of a scenario dependent threshold T_g and a per-pixel threshold T_{pp} that dynamically adapts its value along the process, i.e.

$$T_l(\phi) = T_g + T_{pp}(\phi), \quad (6)$$

and the high threshold is defined as

$$T_h(\phi) = T_l(\phi) + C, \tag{7}$$

where C is a constant whose value depends on the scenario.

The thresholded difference image D_{th} is defined in such a way that it is non-zero only if the pixel is active, i.e.,

$$D_{th}(\phi) = \begin{cases} \min_i |I^t(\phi) - B_i^t(\phi)| & if\ active \\ 0 & if\ non-active \end{cases} \tag{8}$$

Each pixel in a new frame is classified either as a target or nontarget pixel. Target pixels can also be classified as object (O), shadow (S) or highlight (H) and ghost (G) pixel. The clustering of target pixels is based on the following validation rules

O → (target)&[∼(shadow/highlight)]&(in motion)
S/H → (target)&(shadow/highlight)
G → (target)&[∼(shadow/highlight)]&[∼(in motion)]

A critical situation occurs whenever objects stop their movement for a period or when objects modelled as being part of the background start moving. To deal with this situation, each pixel has a state transition map defining a dynamic pixel rate of adaptation. The state transition map will encode in all the moving object pixels the elapsed time since the beginning of the object movement. Different rates of adaptation are used according to

$$\eta = \begin{cases} 1.0 & : if\ [(Ghost)\&(elapsed\ time < \delta_t)] \\ \eta_s & : if\ [Object] \\ K \cdot e^{\nabla t} & : if\ [(Ghost)\&(elapsed\ time \geq \delta_t)] \\ \eta_f & : Otherwise \end{cases} \tag{9}$$

where ∇t is the elapsed time since the target stopped its movement and $\delta_t = \frac{\sqrt{w_b^2+h_b^2}}{f_r \cdot \sqrt{v_x^2+v_y^2}}$ being (w_b, h_b) the width and height of the bounding box respectively, f_r the frame rate and (v_x, v_y) the image velocity components of the bounding box center of mass.

1) Threshold Updating: Just like the background images, threshold levels are updated at the last step of the frame processing, i.e, after determining which pixels belong to the target and nontarget sets. If a pixel of D_{th} is declared as active and it is not on the target set, then it is labelled as a noisy pixel. By increasing T_{pp} of noisy pixels, the system sensibility is reduced, as well as the probability of having pure noise regions classified as targets. When pixels are classified as nontarget their T_{pp} is reduced, therefore increasing their sensibility. The per-pixel threshold T_{pp} is modified according to:

$$T_{pp}(\phi) = \begin{cases} T_{pp}(\phi) + C_u & \text{if } D_{th}(\phi) > 0\ \&\ \phi \in N_t \\ T_{pp}(\phi) - 1 & \text{if } D_{th}(\phi) = 0\ \& \\ & mod(upct(\phi), C_f) = 0\ \& \\ & T_{pp}(\phi) > 0 \\ T_{pp} & \text{otherwise} \end{cases} \tag{10}$$

where $D_{th}(\phi)$ is the thresholded difference image, $upct(\phi)$ is a matrix that holds the update count and C_f is a system parameter. The initial threshold values were obtained empirically and $C_f = 8$.

D. Regions Definition

Most systems compute the difference between the current frame and the background image and consider as targets the pixels above a certain threshold. Then, pixels close to each other are clustered to form possible targets regions. This process usually leaves gaps that might lead to erroneous targets detection. Morphology can fill these gaps but using threshold-with-hysteresis is a more accurate way because only meaningful gaps are filled.

1) Quasi-Connected Components (QCC): The QCC combines TWH with gap filling and connected component labelling. One of the techniques to keep a fast QCC process consists on a reduction in resolution that also provides gap filling.

Based on $D_{th}(\phi)$, a low resolution image P (parent image) is created. The thresholded difference image is divided in regions which area is a factor multiple of two. Each pixel value of the parent image stores the number of pixels above high and low threshold of the corresponding region in the thresholded difference image D_{th}.

Analyzing the parent image, target regions are defined by connected pixels above the low threshold where the region also contains a given fraction of its pixels above the high threshold. In this way, the region has an overall high sensitivity while also trying to ensure that at least some of the pixels are very unlikely to be false alarms (since they are above the high threshold). Only the regions that have a minimum number of pixels above T_h equal to $(0.005A)$, where A is the corresponding area in the high resolution image, that matches the previous labelled regions, in a spatial-temporal sense. These two validation criteria are extremely useful to distinguish between coherent targets from fuzzy collection of isolated points. Although, the QCC is unable to remove shadow-casts and over-illuminated regions.

2) Shadow-cast and over-illumination removal: Shadow-cast and over-illuminated regions occur due to different kind of sources. While the first occurs during daylight and is induced by the sunlight, the latter occur during the night and is induced by the vehicle's night light system. In both cases, their presence induces the labelling of S/H regions as targets, which in some cases are difficult to remove. These two type of regions share a common characteristic: they suffer an intensity variation related to the background images, being darker in the case of shadow-cast and lighter in the case of over-illumination while the remaining features remain relatively stable. We observed that most of S/H pixels can be correctly removed using a simple 8x8 block normalized cross-correlation between the image and the primary background (figure 1).

Representing the image blocks centered at each pixel of the labelled target regions as vectors, the normalized cross-correlation (NCC) between two blocks $P = (p_0, p_1, \ldots, p_{N-1})^T$ and $Q = (q_0, q_1, \ldots, q_{N-1})^T$, where N is the number of pixels in the block, is given by

$$NCC = \frac{1}{\sigma_p \sigma_q} (P - \bar{P}) \cdot \left(Q - \bar{Q}\right) \tag{11}$$

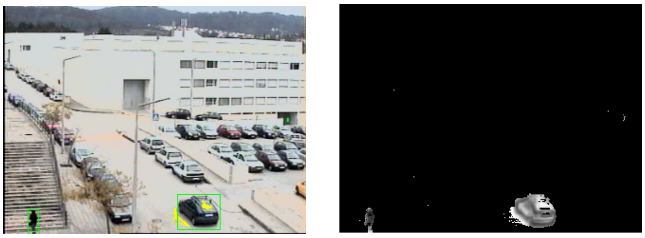

Fig. 1. Foreground/background vehicle detection. Left: Vehicle and light shadow-cast detection. Right: The NCC between the image and the primary background.

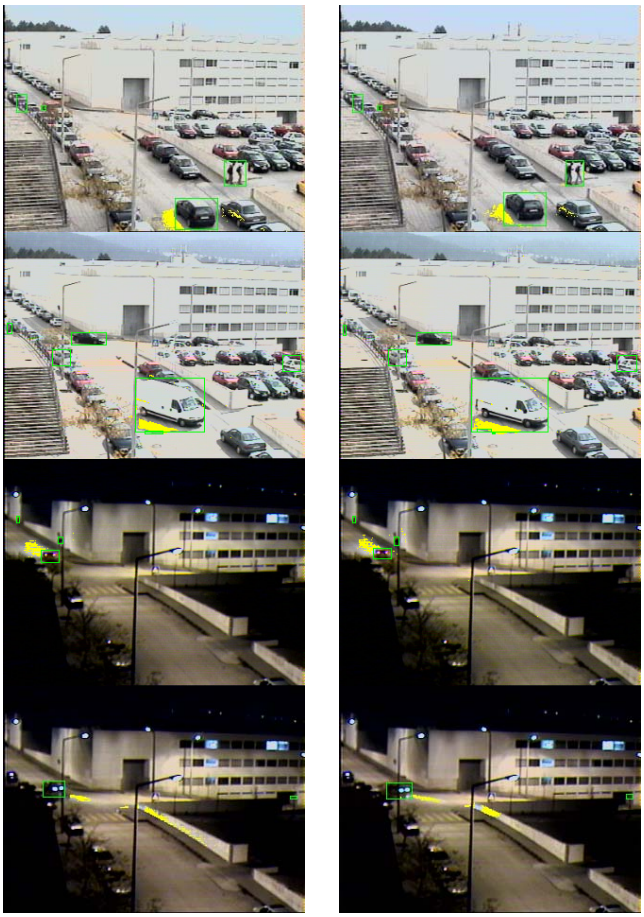

Fig. 2. Shadow-cast and over-illumination detection. (Rows 1&2 : Light shadow-cast detection. Rows 3&4 : Over-illumination detection due to the vehicle light system.

where σ_p and σ_q are the standard deviations over the blocks and $\bar{P}$ and $\bar{Q}$ are vectors containing the means:

$$\sigma_p = \sqrt{(P-\bar{P})\cdot(P-\bar{P})} \tag{12}$$

$$\bar{P} = (\bar{p},\bar{p},\ldots,\bar{p})^T \qquad \bar{p} = \frac{1}{N}\sum_{i=0}^{N} p_i \tag{13}$$

If

$$\begin{cases} NCC(x,y)_{x,y\in L} > \delta & S/H \\ NCC(x,y)_{x,y\in L} \leq \delta & Object \end{cases} \tag{14}$$

After this cleaning process, a normal connected component phase is applied followed by some morphology operations.

This very simple solution works very well for light shadow-casts and it is a very efficient process to eliminate ghost regions due to over-illumination (figure 2).

III. Block-Level Stage

The block-level stage has to accomplish two main tasks which are implemented as two different processing steps: grouping management and object clustering and occlusion reasoning.

In the first step, a group reasoning algorithm has been developed to label the occurrence of the detected targets T_n in time frame t into six distinct events: merge, split, merge& split, new, lost or update.

In the second step, the image frame is decomposed into a matrix of 8x8 blocks. The main purpose of this level step is to classify each block into vehicles, assigning a vehicle label to each block. This is accomplished by performing a 8x8 block-region analysis defining a block energy function that is used to label the blocks belonging to different vehicles and track them over a stack of images.

This labelling process outputs a target/nontarget label image L that is feedback to the pixel-level stage.

A. Single-view tracking

The single-view tracking aims to track at the image level all moving targets detected and segmented by the pixel-level stage.

The target state vector is represented by $X = [p_b\ w_b\ \dot{p_b}\ \dot{w_b}]^T$, where $p_b = (x,y)$ and $\dot{p_b} = (\dot{x},\dot{y})$ are the position and velocity of the vehicle's bounding box center of mass and w_b is the dimension of the bounding box.

The system model used is the following discrete model [9]:

$$X_k = \mathbf{f}(X_{k-1},k-1) + \mathbf{W}_k \qquad Z_k = \mathbf{h}(X_k,k) + \mathbf{V}_k \tag{15}$$

where $\mathbf{W}_k$ is a discrete-time white noise process with mean zero and covariance matrix Q, $\mathbf{V}_k$ is a discrete-time white noise process with mean zero and covariance matrix R, and $\mathbf{W}_j$, $\mathbf{V}_k$, and X_0 are uncorrelated for all j and k. We considered the assumption that trajectories are locally linear in 2D and the width of the bounding box changes linearly. In this case the resulting system model follows a linear difference equation $X_k = A\cdot X_{k-1} + W_k$ where the system evolution matrix, A_k, is based on first order Newtonian dynamics and assumed time invariant.

The measurement vector is represented by $Z_k = [p_b, w_b]^T$ and is related to the state vector via the measurement equation $Z_k = C\cdot X_k + \mathbf{V}_k$.

B. Grouping management

At this stage it is important to define the concept of an *object*. An *object* can have a single or compound nature and represent an image tracked target. It is represented by the descriptor $O_n = [T_n, \zeta_n, j, \{Ł[i]|_{i=1..j}\}]$, where T_n represents the object descriptor, ζ_n the tracker parameters and j the

number of targets associated to the object n. Ł$[i]$ is a list of pointers to the j object descriptors that form the compound object ($j > 1$).

The target model descriptor, represented by $T_n = [p_b, w_b, A_b, X_t, h[i]|_{i=1..k}]$, is composed of several primitives: the image coordinates of the bounding box center of mass (p_b), the dimension of the bounding box (w_b), the area of the blob (A_b), a block object map X_t that has the label distribution from time frame t, and the color information associated to each k target blocks $h[i]_{i=1..k}$ (block-color histogram).

To disambiguate between possible candidates of correspondence in the tracking process multiple image cues are used based on spacial and temporal estimation and color.

Assume that we have N *a posterior* estimated image position *objects* ($\hat{O}_i$) and M detected *targets* (T_j) for time frame t. The bounding boxes overlapping ratio

$$OR(\hat{O}_i, T_j) = max\left(\frac{\bigcap(\hat{O}_i, T_j)}{\bigtriangledown T_j}, \frac{\bigcap(\hat{O}_i, T_j)}{\bigtriangledown \hat{O}_j}\right) \qquad (16)$$

is used to build correspondence matrices (CM) between $\hat{O}_i$ and T_j for time frame t. $\bigtriangledown$ represent the area of the bounding box and $\bigcap$ the bounding boxes overlapping area.

The correspondence matrix (CM) is a $N \times M$ matrix, defined as follows

$$CM(i,j) = \begin{cases} 1 & OR(\hat{O}_i, T_j) > T \\ 0 & OR(\hat{O}_i, T_j) \leq T \end{cases} \quad \forall_{i\in 1...N,\ j\in 1...M} \qquad (17)$$

where T is the threshold which accounts for the overlap requirement. We define two auxiliary vectors

$$CM_L(i) = \sum_{j=1}^{M} CM(i,j)_{\forall_{i\in 1...N}} \qquad CM_C(j) = \sum_{i=1}^{N} CM(i,j)_{\forall_{j\in 1...M}} \qquad (18)$$

CM:

	T_1	T_2	...	T_M	CM_L
$\hat{O}_1$	1	0	...	1	2
$\hat{O}_2$	0	1	...	0	1
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$
$\hat{O}_N$	0	1	...	0	1
CM_C	1	2	...	1	

Detected targets (T_j) are classified according to the following rules:

$$\begin{array}{ll} 1 \longleftrightarrow 1 & \exists_i : CM_L(i) = CM_C(j) = 1 \wedge CM(i,j) = 1 \\ Merge & \exists_i : CM_C(j) > 1 \wedge CM(i,j) = 1 \\ Split & \exists_i : CM_L(i) > 1 \wedge CM(i,j) = 1 \\ Split\&Merge & \exists_i : CM_L(i) > 1 \wedge CM_C(j) > 1 \wedge CM(i,j) = 1 \\ New & \exists_j : CM_C(j) = 0 \\ Lost & \exists_i : CM_L(i) = 0 \end{array} \qquad (19)$$

From these six classification rules, the most difficult to handle are the split, merge and split&merge events. In the case of merging the algorithm has to deal with total or partial occlusion and grouping. Its goal is to cluster the blocks superimposed by the detected target T_n into different vehicles, determining the most likely block object map X_t based on the current target T_n^t, the object map X_{t-1} and the matched target in time frame $t-1$, T_n^{t-1}. For splitting events, the algorithm have to disambiguate which objects are associated to different targets.

Based on the correspondence matrices (CM), four managers, running in cascade, were used to handle the image *objects*: *split* manager, *merge* manager, *new/lost* manager and *update* manager.

The Split manager- When a split event is detected, two possible situations can occur: a compound object split (the most common case) or a single object split (when a group of vehicles enter the surveillance area and split).

To handle the compound object split, the manager creates a new correspondence matrix between the objects inside the compound object and the image targets (T_n) that are detected as split candidates. The correspondence is, this time, based on block-region analysis, by associating a detected target to each object of the compound. In the case of a compound object, the object descriptor O_n has all the relevant information of the singular objects inside the compound. Each single block of the compound object has an object label that supply a first hint to disambiguate which objects are associated to different targets.

In order to handle the split&merge event, after the correct matching of the splitting objects, the descriptors of each object inside the compound are recovered from the compound object descriptor and added to the tracked object list, associating to each object the segmented target primitives of the target they matched. The compound object descriptor is removed from the tracked objects list and discarded. In case of a split&merge event, the tracked object list is feeded into the merge manager that creates a new compound object for the new merged object.

For the case of a single split, new objects are created and added to the new born object list, associating the detected descriptor to each one of them. This new object is definitely moved to the tracked object list after being tracked for 5 consecutive frames (temporal consistency).

The Merge manager- When a merge situation is detected, a compound object descriptor is created and added to the list of object trackers, moving the object descriptors of the merged objects from the tracked object list to a dying object list, decreasing its life vitality over a period of 10 frames, being definitely discarded after this period. The new object descriptor store the descriptors of the objects merged (compound objects descriptors) and also the total number of targets that compose the compound target. If a split situation is detected before the death of the objects (ex: objects crossing), the objects descriptors are recovered from the dying objects list to the tracked list. Figure 3 shows the evolution of the split and merge manager on a split&merge event detection.

Based on the *a posterior* kalman filter estimated velocity all labelled blocks of object O_n^{t-1} in time frame $t-1$ are projected into time image frame t, I_t. To each projected block k superimposed by the detected target blob T_n, the energy function $F(k)$ is obtained. In a merging event with partial occlusion, several blocks tend to be labelled to different objects, in special those on the borderline between both

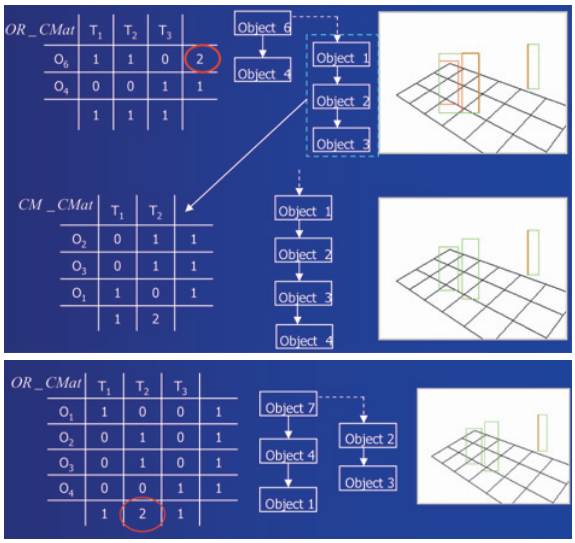

Fig. 3. A split&merge event handeled by the proposed grouping management algorithm. Top row: Split-manager; Bottom row: merge manager.

objects. These blocks are checked in order to decide to each vehicle they belong.

The New/Lost manager- When a null value is detected on the last row of the *CM* matrix this means that a new object was detected. A single object descriptor is created and included on a list of new born object increasing is life vitality over a stack of 5 frames. After this period, the descriptor is moved to the tracked object list.

If a null value is detected on the last column of the *CM* matrix a lost object is considered to happen. Its descriptor is moved from the tracked object list to a dying object list decreasing its life vitality for a period of 10 frames over which it is definitely discarded.

The Update manager- At this stage, the tracked object list has a complete *object* $\rightarrow$ *target* matching, updating the object trackers with the segmented targets (T_n) information.

C. Object clustering and occlusion reasoning

The block-energy function is defined based on spacial and temporal image correlation and block-color matching. Three measures are used in this energy function $F(k)$, where $k \in K_b$ and K_b is the set of blocks associated with target T_n.

The energy measures used are the *block neighbor consistency*, where the more neighbor blocks exist with the same label, the more likely the block is to have the object label, the *temporal block-matching*, where the neighbor condition between consecutive frames is evaluated as a degree of overlapping between the block and its re-projection into the previous frame and finally the *block-color matching* where the color likelihood of blocks is measured using the intersection of the color histograms of the block and its re-projection to the previous frame.

Using the energy function, the algorithm is able to validate the projected block map $\hat{X}_t$ using the previous block map X_{t-1}, the actual detected targets T_n and the objects O_{n-1} tracked in time frame $t-1$.

Based on the *a posterior* kalman filter estimated velocity all labelled blocks of object O_n^{t-1} in time frame $t-1$ are projected into time image frame t, I_t. To each projected block B^k superimposed by the detected target blob T_n, the energy function $F(k)$ is obtained. In a $1 \longleftrightarrow 1$ situation, the block map X_t that minimizes the energy function is obtained using the *a posterior* kalman filter estimated velocity.

In a merging event with partial occlusion, several blocks tend to be labelled to different objects, in special those on the borderline between both objects. Using the *a posterior* Kalman filter estimated velocity for each object, a predicted block map $\hat{X}_t$ is obtained projecting the block map X_{t-1} of the objects that merged into time frame t. This map is validated minimizing the energy function

$$F = \sum_{k \in T_n} F(k) \tag{20}$$

$$F(k) = c_1 \cdot (N_p^k - 8)^2 + c_2 \cdot (M_b^k - 64)^2 + c_3 \cdot (H_c^k) \tag{21}$$

where N_p^k is the number of neighbor blocks of a block B_k, M_b^k is the number of overlapping pixels of the blocks with the same vehicle label, when the block B_k is re-projected into previous time frame using the estimated velocity, and H_c^k is the normalized block color-histogram intersection ([12], [16]) defined as

$$H_c^k(B_t^k, B_{t-1}^k) = \frac{\sum_x \min(B_t, B_{t-1})}{\sum_x B_t} \tag{22}$$

where x represents the number of histogram color bins.

In the case of partial occlusion, the algorithm need to determine to which of alternative objects a block is likely to belong. In these cases, some of the borderline blocks tend to be labelled as belonging to several objects. To disambiguate this situation the algorithm estimates an energy function for all possible candidates, using as matching criteria the minimum energy function.

During the tracking of a compound object, the update of the block map X is based on the computation of block motion vectors using image optical flow. The motion vector of each object O_n inside the compound object is the average motion vector of the block labelled as object n. This motion vector is used to obtain the predicted block map $\hat{X}_t$ for time frame t. The updated map X_t is used to cluster the blocks into updated objects, and these clustered objects are used to feed information into the kalman filter trackers. In this process, several trackers are associated with a compound object: The compound object tracker and the trackers of each object inside the compound object.

The *a posterior* estimated velocity supplied by each object tracker is useful in the case of a split event, helping to disambiguate which vehicles are associated with each detected target. Using the estimated velocity, an estimated block map $\hat{X}_t$ is obtained and the correspondence matrix is obtained using the block map X_t that is updated using the energy function.

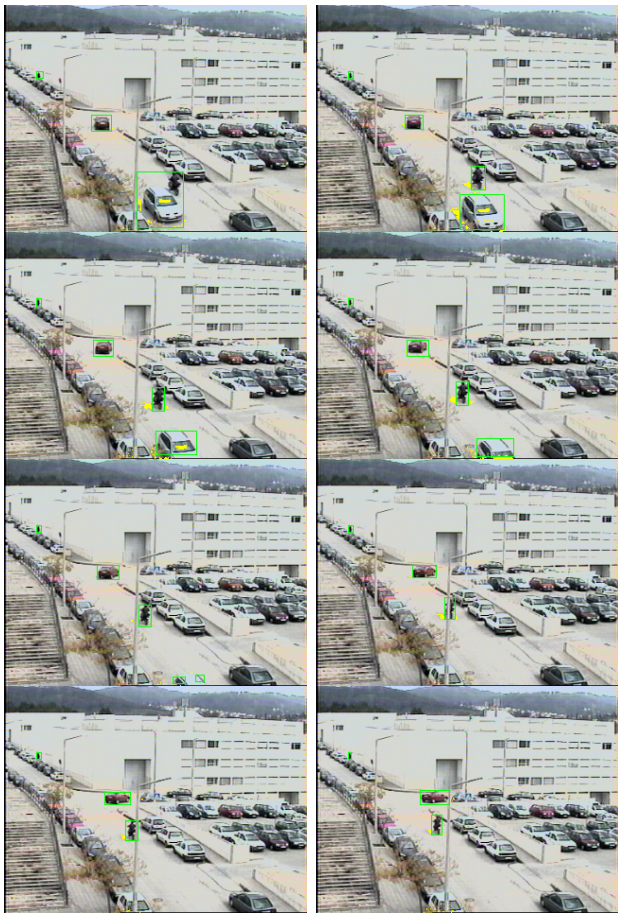

Fig. 4. Vehicle detection and tracking on a real day-light urban scenario.

Fig. 5. Vehicle detection and tracking on a real night urban scenario.

IV. Experimental results

The proposed algorithm is part of a traffic surveillance system that is currently running over a real highway scenario. The proposed algorithm was tested not just on highway scenarios, but also on urban environments where vehicles and pedestrians interact randomly. Figure 4-5 shows the result of the vehicle detection using the proposed algorithm. Figure 6 shows a small part of the tracking result using the proposed algorithm on a real highway scenario.

V. Conclusions

A simple and efficient solution to detect and track multiple vehicles on traffic scenarios is presented. The work presented on the paper is part of a traffic surveillance system currently running experimentally on real highway scenarios. The vehicle detection is based on multiple background model combined with a dynamic thresholding. The system deals with light shadow-casts and over-illuminated detected regions using a simple approach based on normalized cross-correlation working either in night-light or day-light conditions. A simple and robust solution to handle image occlusion and grouping is proposed based on the concept of correspondence matrices. The clustering and tracking of different vehicles uses a image-block vehicle labelling process based on the use of a block energy function.

References

[1] Grimson,W., Stauffer,C., Romano,R., Lee,L., Using adaptive tracking to classify and monitor activities in a site, *Proc. of the IEEE CVPR*, 1998.

[2] Stauffer,C., Grimson,W., Adaptive background mixture models for real-time tracking, *Proc. of IEEE CVPR*, 1999.

[3] Beymer,D., McLauchlan,P., Coifman,B., Malik,J., A real-time computer vision system for measuring traffic parameters, *Proc. of IEEE CVPR*, 1997.

[4] Magee,D., Tracking Multiple Vehicles using Foreground, Background and Motion Models, *Image and Vision Computing*, vol 22(2), pp143-155, 2004.

[5] Cavallaro,A., Salvador,E., Ebrahimi,T., Shadow-aware object-based video processing, *IEE Vision, Image and Signal Processing*, Vol. 152, Issue 4, August 2005.

[6] Cavallaro,A., Steiger,O., Ebrahimi,T., Tracking video objects in cluttered background, *IEEE Transactions on Circuits and Systems for Video Technology*, 15(4), April 2005, pp.575- 584

[7] Salvador, E., Cavallaro,A., Ebrahimi,T., Cast shadow segmentation using invariant colour features, *Computer Vision and Image Understanding*, vol. 95, n.2, August 2004, pp. 238-259

[8] R. Collins, et al., A System for Video Surveillance and Monitoring. *CMU-RI-TR-00-12*, Carnegie Mellon University, 2000.

[9] Bar-Shalom,Y., Fortmann,T., Tracking and Data Association. *Academic Press, Inc*, New-York 1988.

[10] Horpraset,T., Harwood,D., Davis,L., A statistical approach for real-time robust background subtraction and shadow detection, *ICCV'99 Frame Rate Workshop*, 1999.

[11] Haritaoglu,I., Harwood,D., Davis,L., Hidra-Multiple people detection and tracking using silhouettes, *IEEE Workshop on Visual Surveillance*, 1996.

[12] Swain,J., Ballard,D., Color Indexing, *IJCV, 7:1,11-32*, 1991.

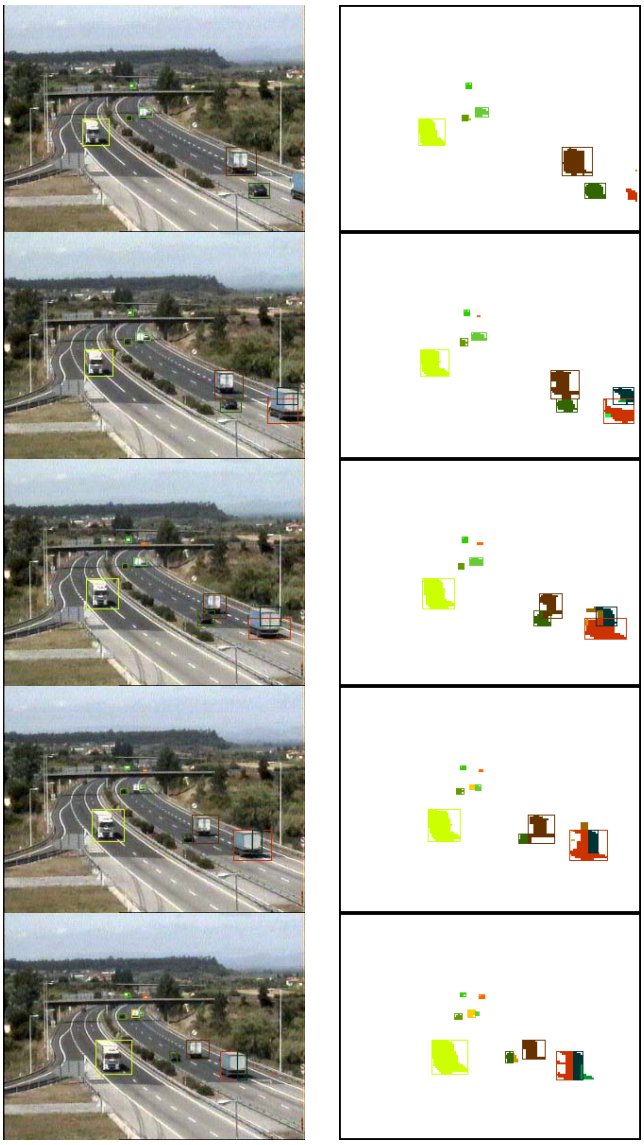

Fig. 6. Vehicle tracking on a real highway scenario. Left: Vehicles detected with the proposed algorithm. Right: The 8x8 Block Map X_t obtained using the energy function $F(k)$.

[13] McKenna,S., Raja,Y., Gong,S., Tracking Colour Objects using Adaptive Mixture Models, *Image and Vision Computing*, *17,225-231*, 1999.

[14] Black,J., Ellis,T., Multi Camera Image Tracking, *IEEE PETS2001*, 2001.

[15] Zhao,T., Nevatia,R., Lv,F., Segmentation and Tracking of Multiple Humans in Complex Situations, *IEEE CVPR*, Hawaii, 2001.

[16] McKenna,S., Jabri,S., Duric,Z., Rosenfeld,A., Tracking Groups of People, *CVIU, 80,42-56*, 2000.

[17] Mittal,A., Video Analysis Under Severe Occusions, *PhD Thesis*, University of Maryland, 2002.

[18] Boult,T., Gao,X., Michaels,R., Eckmann, M., Omni-direccional visual surveillance, *Image and Vision Computing*, n22, 2004.

[19] Boult,T., et al., Frame-rate omnidireccional surveillance and tracking of camouflaged and occluded targets, it 2nd IEEE Int. Workshop on Visual Surveillance, 1999.

[20] Porikli,F., Multiplicative Background-Foreground Estimation under Uncontrolled Illumination using Intrinsic Images, *TR2005-012*, MERL, 2005.

[21] Kamijo,S., Matsushita,Y., Ikeuchi,K., Sakauchi,M., Occlusion Robust Vehicle Detection utilizing Spatio-Temporal Markov Random Filter Model, *7th World Congress on ITS*, Torino, Italy, 2000.

[22] Cucchiara,R., Grana,C., Piccardi,M., Prati,A., Detecting moving objects, ghosts and shadows in video streams, *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 10, pp. 13371342, 2003.

[23] Koller,D., et. al., Towards robust automatic traffic scene analysis in real-time, *Proceedings of Int. Conference on Pattern Recognition*, 1994, pp. 126131.

[24] Foresti, G., Object detection and tracking in time-varying and badly illuminated outdoor environments, *SPIE Journal on Optical Engineering*, 37 (9), 1998.

[25] Brémond,F., Thonnat,M., Tracking Multiple Non-Rigid Objects in Video Sequences, *IEEE Transaction on Circuits and Systems for Video Technology Journal*,vol.8,no.5, 1998.

[26] Batista,J., Tracking Pedestrians in a Multiple Cameras System with Trajectory Prediction and Occlusion Modelling, *PETS/ECCV 2004-Sixth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance*, Prague, Czech Republic, May, 2004.

[27] Batista,J., Tracking Pedestrians under Occlusion using Multiple Cameras, *ICIAR 2004-Image Analysis and Recognition*, Porto, Portugal, Sept-Oct, 2004.