Expert Systems with Applications 42 (2015) 7263–7275

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

An integrated system for vehicle tracking and classification

Sebastiano Battiato [a], Giovanni Maria Farinella [a], Antonino Furnari [a,*], Giovanni Puglisi [b], Anique Snijders [c], Jelmer Spiekstra [c]

[a] University of Catania, Department of Mathematics and Computer Science, Italy
[b] University of Cagliari, Department of Mathematics and Computer Science, Italy
[c] Q-Free, The Netherlands

ARTICLE INFO

Article history:
Available online 29 May 2015

Keywords:
Traffic monitoring
Vehicle tracking
Vehicle classification
Data-driven
Template matching

ABSTRACT

We present a unified system for vehicle tracking and classification which has been developed with a data-driven approach on real-world data. The main purpose of the system is the tracking of the vehicles to understand lane changes, gates transits and other behaviors useful for traffic analysis. The discrimination of the vehicles into two classes (cars vs. trucks) is also required for electronic truck-tolling. Both tracking and classification are performed online by a system made up of two components (tracker and classifier) plus a controller which automatically adapts the configuration of the system to the observed conditions. Experiments show that the proposed system outperforms the state-of-the-art algorithms on the considered data.

1. Introduction

Video traffic monitoring is a popular application domain in Computer Vision. In this context, algorithms are often designed to detect, re-identify, count, track or classify vehicles (Bas, Tekalp, & Salman, 2007; Hsieh, 2006; Zhou, Gao, & Zhang, 2007), while others are designed to improve the safety of the driver (Aouaouda, Chadli, Boukhnifer, & Karimi, 2014; Aouaouda, Chadli, & Karimi, 2014; Dahmani, Chadli, Rabhi, & El Hajjaji, 2013; Saifia, Chadli, Karimi, & Labiod, 2014). We tackle the specific tracking and classification tasks presenting a unified system for the online tracking and classification of vehicles. The system has been designed and tested to work with real-world data acquired by Q-Free[1] and can be used for a series of traffic-related applications ranging from road charging to law enforcement, electronic toll collection and truck tolling.

Many approaches to visual object tracking are available in the literature (Arnold et al., 2013; Maggio & Cavallaro, 2011). Each strategy is formulated by making assumptions on the application domain and choosing a suitable object representation and a frame-by-frame localization procedure. A method to update the target representation during the tracking is usually required, especially when the target is subject to geometric and photometric transformations (pose changes, deformations, illumination changes, etc.) (Maggio & Cavallaro, 2011). The most straightforward approach is probably the Template Matching, where the object is assumed to be rigid and it is represented as an image patch (the template) (Maggio & Cavallaro, 2011; Yilmaz, Javed, & Shah, 2006). If no pose changes are considered, the object is searched in the neighborhood of the last known position by maximizing a chosen similarity function (e.g., Sum of Squared Differences (SSD), Normalized Cross Correlation (NCC), etc.) between the template and candidate image patches. If pose changes are considered, the Lucas–Kanade affine tracker can be used (Baker, Gross, Ishikawa, & Matthews, 2004; Lucas & Kanade, 1981). In this case the pose changes are modeled as a set of affine transformations and the target is localized by estimating the transformation parameters which maximize the Sum of Squared Differences (SSD) between the template and the transformed version of the candidate. In other approaches the object is represented as a set of local feature points which are tracked independently (Maggio & Cavallaro, 2011; Tomasi & Kanade, 1991). This allows the algorithm to naturally deal with the object deformations since no global rigid coherence is required among the key-points. In order to track each key-point, the sparse optical flow can be computed assuming that the changes of the pixel intensities are about entirely due to motion and not to possible lighting changes in the

* Corresponding author.
E-mail addresses: battiato@dmi.unict.it (S. Battiato), gfarinella@dmi.unict.it (G.M. Farinella), furnari@dmi.unict.it (A. Furnari), puglisi@unica.it (G. Puglisi), anique.snijders@q-free.com (A. Snijders), jelmer.spiekstra@q-free.com (J. Spiekstra).

[1] Q-Free (http://www.q-free.com/) is a global supplier of solutions and products for Road User Charging and Advanced Transportation Management having applications mainly within electronic toll collection for road financing, congestion charging, truck-tolling, law enforcement and parking/access control.

http://dx.doi.org/10.1016/j.eswa.2015.05.055
0957-4174/

scene (brightness constancy assumption Horn & Schunck, 1981). The Lucas–Kanade optical flow algorithm (Lucas & Kanade, 1981) is often used to compute the optical flow. It requires the key-points to satisfy both spatial and temporal coherence constraints. In Tomasi and Kanade (1991) it is stated a criterion to choose which points may be selected as key-points in order to improve the performances of the tracker (specifically corners or points taken from a highly textured area of the image). In some cases the set of feature points can be directly "tracked" for specific application contexts (e.g., video stabilization (Battiato, Gallo, Puglisi, & Scellato, 2007), human computer interaction (Farinella & Rustico, 2008), traffic conflict analysis (Battiato, Cafiso, Di Graziano, Farinella, & Giudice, 2013)). In Comaniciu, Ramesh, and Meer (2003a) and in Bradski (1998) the object is represented by describing the image region in which it is contained as a n-bins histogram in the hue feature space. The object is then localized by maximizing a similarity function between the object representation of the current frame and the representation of the target candidate with respect to its position. In Bradski (1998) the CAMShift algorithm is proposed. A probability image is built back-projecting the target object hue histogram onto the current frame in order to obtain a map of the most probable object positions. The object is localized searching for local maxima of the probability map in the neighborhood of the last known position of the target using the Mean-Shift procedure (Comaniciu, Ramesh, & Meer, 2003b; Fukunaga & Hostetler, 1975). In Comaniciu et al. (2003a) the Kernel Based Object Tracking method is presented. A similarity measure based on the Bhattacharyya coefficient is derived. This measure provides a similarity score between the representation of the target object and the one of the candidate found at a given position. The localization is performed by maximizing the similarity measure with respect to the target candidate position using the Mean-Shift procedure. Other methods consider an extended appearance model and solve the tracking task as a classification problem: Arnold et al. (2013), Kalal, Matas, and Mikolajczyk (2009, 2012) and Hare, Saffari, and Torr (2011). In Kalal et al. (2012) TLD is proposed, a hybrid approach capable of tracking the object, learning its appearance and detecting it after its eventual disappearance from the scene. The tracker component is a Lucas–Kanade based tracker which tracks a set of feature points obtained using a regular grid constructed over the target object. The trajectory in the feature space is modeled by two parallel processes that extend and refine an online model (the learning component). A detector component runs in parallel with the tracker in order to enable re-initialization after its failure.

This paper is the extension of our previous work (Battiato et al., 2014) where a first version of the algorithm for vehicle tracking was presented. Here we discuss the tracking algorithm in more details and provide a comparative analysis with respect to the state-of-the-art. Moreover we add a module for online vehicle classification and integrate the two components into a unified system for traffic monitoring purposes.

The rest of the paper is organized as follows: in Section 2 we provide an overview of the system and analyze the reference data. Sections 3 and 4 present the proposed tracking and classification components respectively. In Section 5 we discuss the controller component. In Section 6 we report the experimental settings and discuss the results. Section 7 draws the conclusions.

2. System overview and reference data

The goal of the proposed system is to correctly track the vehicles during their transit through the road. We also want to classify the vehicles into two main classes: tall vehicles (e.g., trucks, buses, etc.) and short vehicles (e.g., cars, vans, etc.). We assume that the detection of the vehicles is performed by an external module based on plate detection and recognition plus background/foreground segmentation.[2] Both tracking and classification are performed online on real-world data. The system is composed of two main components: a tracker and a classifier. The tracker is based on template matching and is augmented with four additional modules tailored to cope with the specific variabilities exhibited by the data. The classifier is based on a supervised machine learning technique trained on a dataset containing both real and artificial examples in order to consider a number of variabilities during the learning process. A controller is introduced to optimize the performances of the tracker by turning the modules on or off on the basis of the feedback received by them. Fig. 1 shows the overall schema of the proposed system. The tracker component consists of four modules plus the classic template matching technique which is used to obtain an initial estimate of the bounding box of the vehicle. The output of the tracker component is used to update the template and to keep track of the vehicle position. The classifier component extracts an image patch of the vehicle from the current frame using the estimated bounding box. This is done once in the whole vehicle transit as explained in Section 4. Finally, the controller optimizes the performances of the tracker component by enabling or disabling each module on the basis of the estimated trajectories (current and past positions) and the predicted class.

The overall components have been designed using a data driven approach, hence an analysis of the application context is necessary before discussing the details of the developed system in the next sections. The reference data consists in video sequences related to real video traffic monitoring which have been acquired by Q-Free. The sequences exhibit high variability in terms of lighting changes, contrast changes and distortion.

Specifically the input data are the result of a preprocessing stage on sequences originally acquired through cameras mounted on the top of the road. The preprocessing stage produces a normalized, low resolution representation of the scene where the distance between neighboring pixels is constant in the real world. An example of the preprocessing results is shown in Fig. 2. The sequences have been acquired in different places and under different lighting, weather and environment conditions and are identified by a keyword summarizing the main variabilities that the system should cope with, namely: Low Contrast, Light Changes, Leading Shadows, Stop And Go + Turn, Rain and Stop And Go. These sequences are considered for both tracking and classification. Three more sequences are introduced for classification purposes only and are identified by the keywords: Sequence 1, Sequence 2, Sequence 3. These sequences are useful to learn new variabilities for the classification and allow to get a larger number of examples of tall vehicles. In order to perform quantitative evaluations, the sequences have been manually labeled, annotating for each vehicle transit the number of the starting frame, the initial bounding box, the number of the final frame and the vehicle class. Specifically each transit T_i is associated to a label l_i, where $l_i = 1$ for the short vehicles (e.g., cars), while $l_i = 2$ for the tall vehicles (e.g., trucks).

Table 1 shows the number of vehicles which have been labeled in each sequences and the corresponding classes. It should be noted that the ratio between the number of tall vehicles and the number of short ones is approximately equals to $1:5$, while ideally we would like to work with a balanced (i.e., $1:1$ ratio) set. The effects of using a balanced dataset are discussed in Section 6. The overall data contain 1208 vehicle transits in total.

In analyzing the application context, we highlight the following relevant characteristics of the reference data:

[2] The plate detection and recognition module is already commercialized by Q-Free.

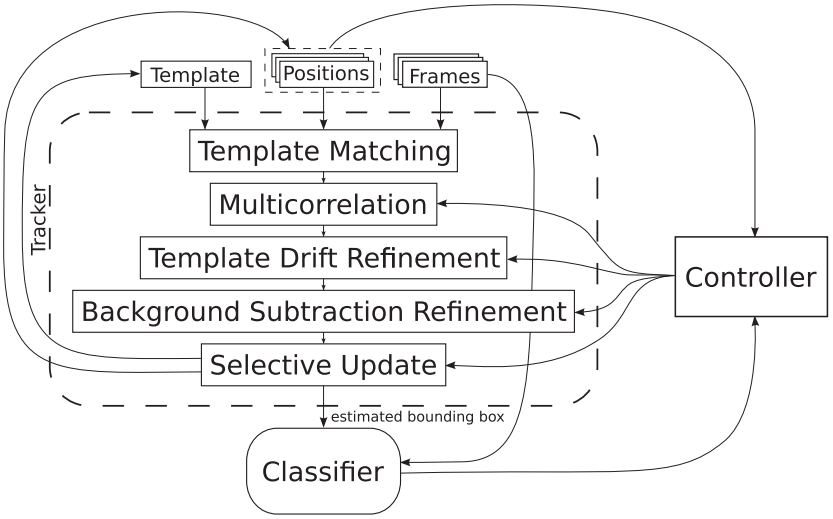

Fig. 1. Diagram of the proposed system.

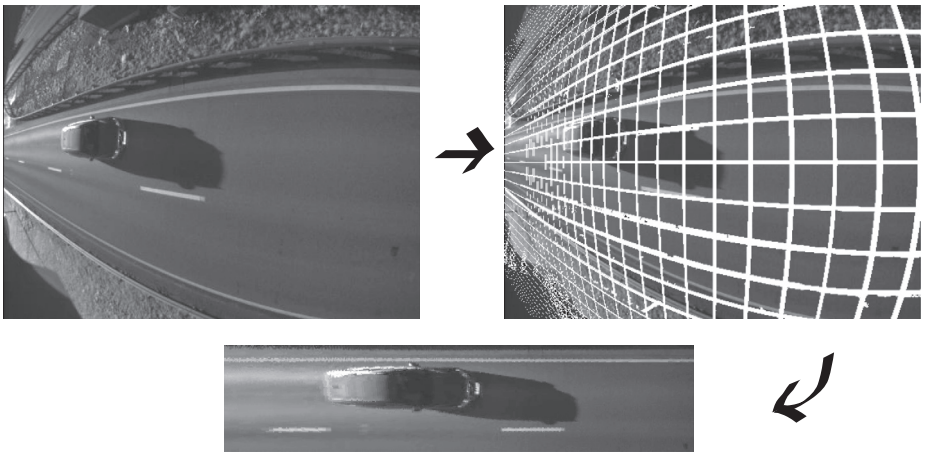

Fig. 2. The preprocessing stage.

Table 1
The number of vehicles labeled in each sequence and their related classes.

Sequence	# Short	# Tall
Low Contrast	231	15
Light Changes	238	19
Leading Shadows	160	23
Stop And Go	95	7
Stop And Go + Turn	109	18
Rain	162	15
Sequence 1	12	3
Ssequence 2	12	3
Sequence 3	0	86
Total per class	1019	189
Total	1208	

- *Grayscale sequences*: all the sequences are grayscale, which makes color-based representations unfeasible.
- *Distortion*: the vehicles are subject to distortion as show in Fig. 3(a).
- *Rigidity of the vehicles*: despite the presence of the distortion, the vehicles are rigid objects in the real world.
- *Artifacts*: light reflections cause the appearance of artifacts in the form of white spots on the vehicles as shown in Fig. 3(b).
- *Geometric and photometric variabilities*: the sequences are subject to lighting and contrast changes (Fig. 3(c)). Moreover the appearance of the vehicles can rapidly change due to perspective reasons especially for tall vehicles (Fig. 3(d)).
- *Slow scenes*: in some cases the motion of the vehicles is very slow, which makes a template updating strategy necessary in order to avoid the propagation of a drifted version of the template.

3. Tracker component

The proposed tracking algorithm (see Fig. 1) is based on the general template matching schema. At the initialization step, the plate detection module returns the bounding box of the frontal part of the current vehicle, then the template is extracted as a part of the current frame and the object position is set to the center of the bounding box (see Fig. 9(a) for a visual example). At each frame a local exhaustive search is performed: a search window is centered at the object last known position and a number of candidates centered at each point of the search window and having the same size as the template are extracted. The object position is then set to the one which maximizes the similarity score between the target template and the candidate one according to a selected similarity

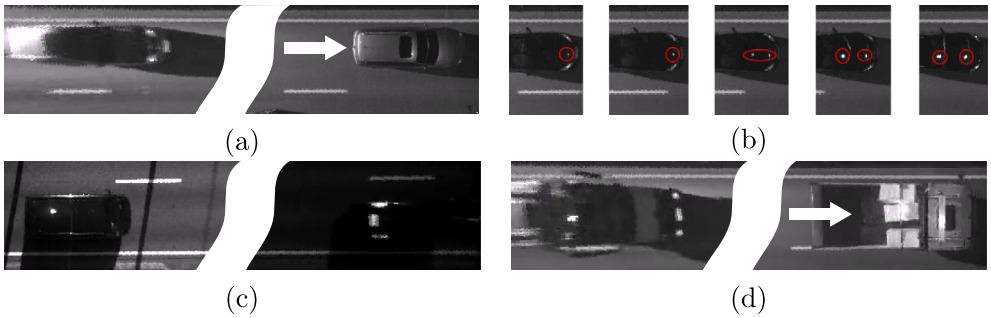

Fig. 3. Variabilities exhibited by the data. (a) Object distortion: the figure shows the appearance of the same vehicle in the left and in the right parts of the scene. (b) Artifacts in the form of spots due to the appearance of light reflections on the vehicles. (c) Light and contrast changes: the figure shows the effects of illumination changes on two different vehicles in two near frames of the same sequence. (d) Perspective changes: the figure shows how the appearance of a given vehicle changes due to perspective.

measure. We use this general schema (Maggio & Cavallaro, 2011) as a baseline and augment it considering different modules which can be dynamically switched on or off by a controller. Each module is designed to cope with one of the variabilities discussed in Section 2. In the following we briefly summarize the modules and motivate their introduction:

- The "multicorrelation" module is introduced to cope with partial occlusion due to the appearance of artifacts in the form of white spots on some parts of the vehicles.
- The "template drift refinement" module is introduced to cope with the template drift caused by the presence of distortion, light, contrast and perspective changes.
- The "background subtraction" module is introduced to cope with the sudden changes of appearance due to perspective variations.
- The "selective update" module is introduced to cope with the template drift issued by the continuous update of the target representation in slow scenes.

The proposed algorithm can be summarized by the following steps:

1. perform a regular template matching search to get an initial guess of the vehicle bounding box;
2. if the similarity score is under a given threshold t_m, use the multicorrelation method to reduce the influence of the artifacts (see Section 3.1);
3. refine the position to correct the template drift caused by the distortion (see Section 3.2);
4. refine the position to correct the template drift caused by the perspective changes (see Section 3.3);
5. if the similarity score is under a given threshold t_u, update the vehicle representation (selective update). This prevents from storing a wrong representation of the object in the scenes characterized by slow motion (see Section 3.4).

In the following sections we summarize the scope of each module used to extend the basic template matching procedure providing the related details. In order to help the reader to keep track of the mathematical symbols and parameters used in this section, we provide Table 2.

Table 2
Table of symbols related to the tracking component.

Parameter	Description
T_i	ith vehicle transit
f_i	ith frame in a video sequence
t_m	Threshold on the similarity score used to activate the multicorrelation module
t_u	Threshold on the similarity score used to activate the selective update module
M	Foreground mask
t	Threshold used to compute the initial foreground mask
s_M	Size of the median filter used to remove noise from the foreground mask
s_d	Size of the ellipsoidal structuring element used to perform the dilation operation on the foreground mask
d	Minimum foreground percentage

3.1. Multicorrelation

The presence of artifacts (see Fig. 3(b)) contributes to radical changes of the appearance of the vehicles between consecutive frames. In such cases the similarity between the current instance of the object and its representation can be low, thus making the template matching based tracker less accurate and possibly leading to a failure. An example of this problem is shown in Fig. 4(a). In order to avoid the influence of the artifacts, we act as if it were an occlusion problem introducing an alternative way to compute the similarity between two image patches. Fig. 5 summarizes schematically such computation: both the template and the candidate image patches are divided into nine regular blocks. A similarity score is computed between each pair of corresponding blocks and the final score is obtained by averaging the nine subwindows similarity values. An analysis of the similarity values highlighted that when this issue arises, the similarity measure computed in the regular way tends to be lower than a given threshold t_m. So we use the multicorrelation similarity measure only when the regular similarity score is under the given threshold. Fig. 4(a) shows the effects of the artifacts on the baseline algorithm and the result of the proposed technique.

3.2. Template drift refinement module

The presence of light, perspective, contrast changes and distortion, together with the continuous update of the template, cause the template drift problem in the form of the progressive inclusion of the background into the template. This effect is shown in the top row of Fig. 4(b). In order to reduce the template drift, a module to refine the position of the vehicle is introduced (see Fig. 6). The refinement is based on the observation that the vehicle is stretched horizontally by effect of the distortion introduced in the preprocessing stage (see Fig. 3(a)). According to this assumption, we adopt the following strategy: given the current frame and the template found at the previous frame, we search for a version of the object at a smaller horizontal scale, obtaining a smaller tracking box. The smaller tracking box is properly enlarged backward to fit the dimensions of the original template in order to enclose more

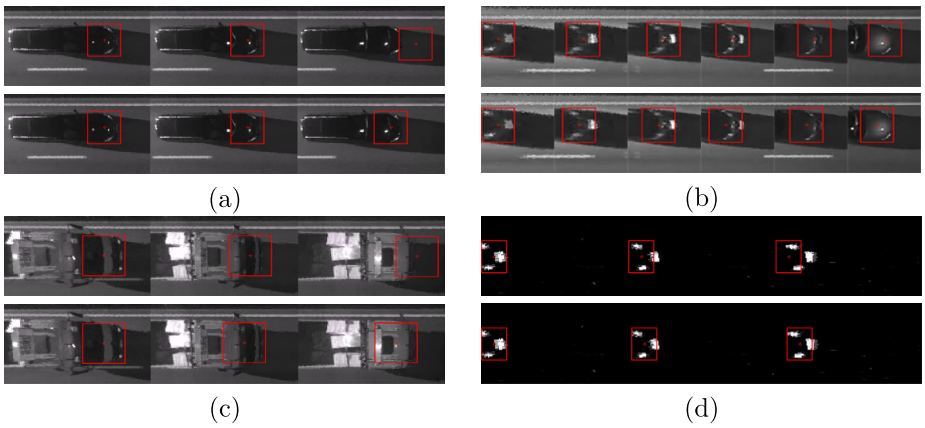

Fig. 4. The issues tackled by the introduced modules (top) and the results of the proposed techniques (bottom). (a) Artifacts and multicorrelation. (b) Template drift and refinement. (c) Perspective issues and background subtraction based refinement. (d) Slow motion scenes and selective update.

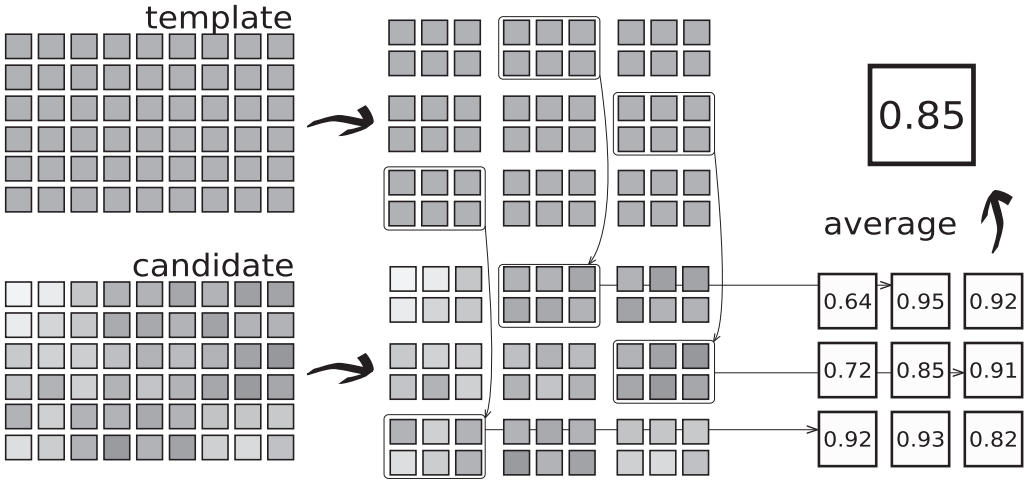

Fig. 5. The multicorrelation procedure.

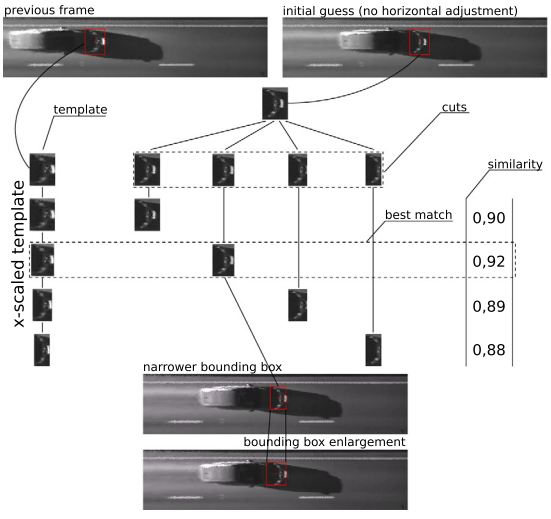

Fig. 6. The template drift refinement procedure.

information. An exhaustive search of the object at different horizontal scales would make the algorithm much slower, so, in order to speed up the computation, we first perform a regular search (i.e., without any refinement) in order to obtain an initial guess. Afterwards we consider a number of candidates (which we call cuts) obtained by discarding the rightmost pixels (the ones which are more likely to contain background information) and search for the best match with horizontally-scaled versions of the template. The best match identifies both the right scale factor and the correct position. The described method can be summarized as follows:

1. *Initial guess:* a regular search (i.e., without any refinement) is performed in order to obtain a first guess of the position of the vehicle.
2. *Cuts:* for each scale factor in a given range, a cut is obtained by discarding the rightmost pixels from the initial guess in order to build a candidate of width equal to $cut\_width = initial\_guess\_width * scaling\_factor$.
3. *Similarity:* the similarity scores between the x-scaled templates and the cuts are computed.
4. *Best match:* the best match identifies a narrower bounding box (the background pixels are removed).
5. *Bounding box enlargement:* in order to enclose more information, the narrower bounding box is enlarged including the leftmost pixels to fit the original bounding box dimensions.

Fig. 6 shows a schema of such computation.

3.3. Background subtraction module

When tracking tall vehicles, the perspective issue shown in Fig. 4(c) arises: the radical change of the appearance of the vehicle

in consecutive frames leads to the progressive inclusion of the background inside the template model up to a possible failure of the tracker. In order to correct this behavior we perform a background aware position refinement on the basis of a rough background subtraction technique based on the subtraction of subsequent frames. In order to identify the background pixels, we build a foreground mask using the following simple procedure:

- Let f_i be the current frame and let f_{i-1} be the previous frame. Compute:

$$\Delta f = |f_i - f_{i-1}|. \quad (1)$$

- Let $t \in [0, 255]$ be a given threshold. Compute the initial foreground mask as following:

$$M(x,y) = \begin{cases} 255 & \text{if } \Delta f(x,y) > t \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

- Apply a median filter of size s_M to M in order to remove noise (e.g., rain);
- Apply a morphological dilation with an ellipsoidal structuring element of size s_d to M in order to fill the holes in the mask.

Note that the foreground mask is not perfect but it is already useful for the case (see Fig. 7). The second stage consists in moving the tracking box backwards using the information coming from the foreground mask in order to exclude the background pixels. We define a foreground region as an area in the foreground mask which has at least $N * d$ non-zero pixels, where N is the total number of pixels in the region and $d \in [0, 1]$ is the minimum foreground percentage. To refine the tracking box position we use the following procedure:

1. Let $d \in [0, 1]$ be the minimum foreground percentage and let M be the foreground mask corresponding to the current tracking box.
2. Consider the rightmost p-pixels wide column of the M mask and let N be the total number of pixels in the considered row.
   - If the row contains less than $d * N$ non zero values, the tracking box is shifted p pixel backward in the horizontal direction, go to step 1.
   - Otherwise stop.

Fig. 7 shows an example of such procedure.

3.4. Selective update module

The continuous update of the vehicle representation (i.e., the template) induces the template drift problem in those sequences in which the motion is slow. The problem is similar to the one tacked in Section 3.2 but it is caused by a different variability. An example of this problem is shown in Fig. 4(d). Since the vehicles move very slowly and considering that appearance of the vehicle between two consecutive frames changes slightly, a shifted version of the template still returns a high similarity score, while the continuous update favorites the propagation of a wrong representation of the vehicle. In order to correct this behavior, we update the object representation only when it is significantly different from the old one, i.e., when the similarity score is lower than a fixed threshold t_u. Fig. 4(d) shows the results of the proposed module.

4. Classifier component

The vehicle classification is tackled as a linear classification problem where the data consists of image patches extracted during the tracking according to the estimated bounding box of the vehicle. Specifically the classification is performed when the vehicle approaches the central part of the scene, where the perspective variabilities discussed in Section 2 are less significant. Fig. 8 shows a general schema of the training/classification pipeline which is briefly presented in the following, whereas the details are discussed in the next sections. In order to build the classifier, a training set is obtained considering the image patches extracted from the input sequences discussed in Section 2. To make the learning procedure more robust, the training set is augmented by generating artificial image patches aimed at introducing translation, perspective, rotation and photometric variabilities. The patches are normalized to the training set mean patch size and the HOG (Histogram of Oriented Gradients) features are extracted Dalal and Triggs (2005). The dimensionality of the feature vectors is reduced through the Principal Component Analysis (PCA) Hotelling (1933). The Linear Discriminant Analysis Fisher (1936) is performed on the PCA reduced data to project the samples to the most discriminant dimension. This unidimensional feature is aggregated to the image patch height in pixels, obtaining a two-dimensional vector. A new LDA projection is hence performed on the two-dimensional data and the projected populations are modeled as distinct unidimensional Gaussian distributions. In the classification step, the sample is projected using the previously learned PCA and LDA bases and is aggregated to the image height feature as in the training phase. The Mahalanobis distances (Mahalanobis, 1936) between the projected sample and the Gaussian distributions related to the two classes are computed. The sample is assigned to the class giving the smallest distance according to the Maximum A Posteriori (MAP) criterion. Table 3 reports a list of the symbols used in this section.

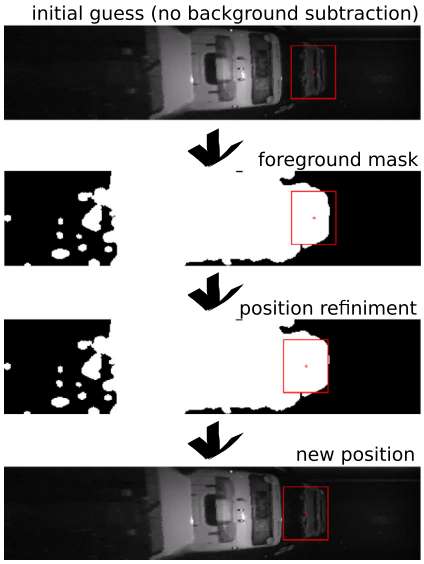

Fig. 7. Background subtraction refinement.

4.1. Data extraction, augmentation and normalization

The image patches for the classification step are automatically extracted from the video frames during the tracking. Since we track the front part of the vehicle (see Section 3), we can assume that meaningful information is contained on the left of the tracking box. On the basis of this assumption an image extraction window is obtained enlarging the tracking box backwards. Specifically, the window has the same height as the tracking box but it is three times larger (see Fig. 9(a) and (b)). In order to make the training

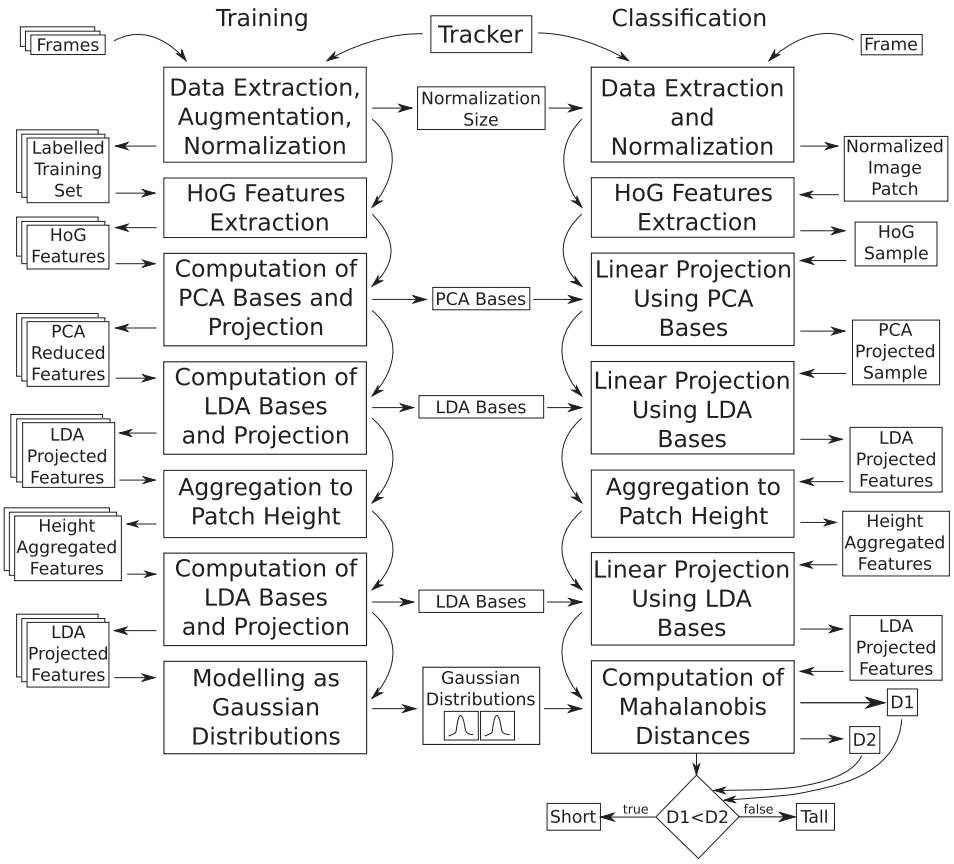

Fig. 8. The training/classification pipeline.

phase robust to some variabilities and in order to get more data, we augment the dataset, which is a common practice in machine learning (Shotton, Johnson, & Cipolla, 2008). For each transit we introduce four variabilities: perspective, translation, rotation and photometric transformations. The perspective variabilities are obtained extracting for each transit three image patches when the center of the extraction window is found approximately on the vertical central line of the scene and when it is found before or after that line at a uniform step p_s (see Fig. 9(g)). The translation variabilities are obtained extracting 8 additional patches for any previously extracted patch shifting the window by p_t pixels in the main directions: top-left, top, top-right, right, bottom-right, bottom, bottom-left, left (see Fig. 9(c)). The rotation variabilities are obtained by extracting two additional patches for any previously extracted patches (including the ones related to perspective and translation variabilities) rotating the extraction window about its center by $\pm p_r$ degrees (see Fig. 9(d)). The photometric variabilities are obtained extracting two additional patches for any previously extracted patch (including all the other variabilities) after photometric processing: $I_i^{ph_{1,2}} = (p_\alpha)^{\pm 1} I_i \pm p_\beta$ (see Fig. 9(e) and (f)). The combination of all the variabilities allows to obtain 99 patches per each vehicle labeled in the dataset. All the patches are resized to the training set mean patch size $\bar{s}$.

4.2. Feature extraction

The feature extraction step allows to obtain a representation of the vehicle which is suitable for describing the main characteristics of the vehicle class. In order to obtain robustness to misalignment, the HOG (Histogram of Oriented Gradients) features are considered Dalal and Triggs (2005). In the HOG extraction process the input image is divided into blocks of a given size (*cellSize* parameter) and the histograms of the gradient orientations are computed for each block. A post-processing procedure which considers contrast normalization is employed. The HOG features provide a representation of the object which is robust to misalignment since local spatial information is lost, but valuable global spatial information is still considered. The output of the HoG feature extraction is a table of histograms (one per block) which is properly reshaped to a vector $\mathbf{x}$ $[n \times 1]$.

4.3. Training chain

In this section we discuss the training chain, which is the process used to learn the parameters of the classifier: the normalization size $\bar{s}$, the PCA bases and bias W_1, $\mathbf{b}_1$, the LDA bases and bias W_2, $\mathbf{b}_2$, the LDA bases and bias W_3, $\mathbf{b}_3$, the means and variances of the Gaussian distributions μ_1, μ_2, σ_1, σ_2. The training chain is represented in the left part of Fig. 8.

After the feature extraction, the Principal Component Analysis (PCA) (Hotelling, 1933) is used to reduce the dimensionally of the HOG feature vectors. This is done by computing the matrix W_1 $[n \times m]$ and the vector $\mathbf{b}$ $[m \times 1]$ which are used to project the vectors $\mathbf{x}_i$ into the m principal components:

$$\mathbf{y_i} = W_1^T \mathbf{x_i} + \mathbf{b_1} \tag{3}$$

the number of the principal components m can be obtained by choosing how much variability (i.e., information) to discard.

The Linear Discriminant Analysis (LDA) (Fisher, 1936) is then used on the reduced vectors $\mathbf{y_i}$ to project them to the most discriminant dimension, i.e., the dimension which maximizes the between-class variance and minimizes the within-class variance.

Table 3
Table of symbols related to the classifier component.

Parameter	Description
p	Width of the portion of the tracking box to be checked by the foreground subtraction module at each iteration
p_s	Distance from the central line at which the image patches related to the perspective variability are extracted
p_t	Number of pixels by which the extraction window is shifted in the main directions in order to extract the image patches related to the translation variability
p_r	Number of degrees by which the image is rotated in order to extract the image patches related to the rotation variability
$p_\alpha,\ p_\beta$	Parameters of the photometric processing performed to extract the image patches related to the photometric variability
$\bar{s}$	Training set mean patch size
$W_1,\ \mathbf{b}_1$	Learned PCA bases and bias
$W_2,\ b_2$	Learned LDA bases and bias
h_i	The height of the extracted image patch
$W_3,\ b_3$	Learned LDA bases and bias after aggregating the PCA-LDA projected sample with the patch height
$\mu_1,\ \sigma_1$	Estimated mean and variance of the Gaussian distribution modeling the "short vehicles" population
μ_2, σ_2	Estimated mean and variance of the Gaussian distribution modeling the "long vehicles" population
$\mathbf{x}_i$	The extracted image samples
$\mathbf{y}_i$	The image samples projected to the PCA space
z_i	The image samples projected to the PCA-LDA space
$\mathbf{p}_i$	The PCA-LDA projected samples aggregated with the patch height
q_i	The samples aggregated with the patch height projected to the final LDA space
l_i	The label associated to the image sample. $l_i = 1$ if the image patch depicts a "short vehicle", $l_i = 2$ otherwise
$\mathcal{P}_1,\ \mathcal{P}_2$	The unidimensional populations of labeled projected samples
$\tilde{W},\ \tilde{b}$	Combined PCA-LDA bases and biases
$d_1,\ d_2$	Mahalanobis distance from the sample to the two Gaussian populations

The outputs of the LDA analysis are the matrix W_2 $[m \times 1]$ and the bias b_2 $[1 \times 1]$. Since LDA is a supervised procedure, the data labels l_i corresponding to the reduced feature vectors $\mathbf{y}_i$ are involved. The projection to the most discriminative LDA dimension is performed using the formula:

$$z_i = W_2^T \mathbf{y}_i + b_2. \tag{4}$$

It should be noted that the result of this projection consists in unidimensional features z_i since LDA projects the data to a $k-1$ dimensional space, where k is the number of classes ($k = 2$ in our case).

Meaningful information is encoded in the image patch size. This information depends on the way the detection of the vehicle is performed. Assuming that the plate position is used as a starting point and that a segmentation technique is used to infer the initial bounding box, the image patch height is dependent on the vehicle frontal width. We use this information to get a better separation of the classes. To do so we simply concatenate the most discriminative LDA features z_i and the respective patch height h_i, obtaining the two dimensional features $\mathbf{p}_i = (z_i, h_i)$.

The Linear Discriminant Analysis is again applied in order to project the $\mathbf{p}_i$ data to the most discriminant LDA dimension of the new two dimensional feature space:

$$q_i = W_3^T \mathbf{p}_i + b_3, \tag{5}$$

where W_3 $[2 \times 1]$ and b_3 $[1 \times 1]$ are computed using LDA and considering the labels l_i.

Two unidimensional populations $\mathcal{P}_1 = \{q_i | l_i = 1\}_i$ and $\mathcal{P}_2 = \{q_i | l_i = 2\}_i$ are defined. In order to be able to classify new instances, a probability model is built fitting two Gaussian distributions $G(\mu_1, \sigma_1)$ and $G(\mu_2, \sigma_2)$ to the data. The probability of a given sample $\bar{q}_i$ to belong to class j is then assumed as:

$$p(q = \bar{q}_i | class = j) = G(\mu_j, \sigma_j). \tag{6}$$

The Gaussian distributions parameters (namely, means $\mu_1,\ \mu_2$ and variances $\sigma_1,\ \sigma_2$) are estimated through Maximum Likelihood (ML) Bilmes (1998). Fig. 10(a) shows the pseudocode for the training process.

4.4. Classification chain

The classification algorithm operates on image patches extracted from the video stream. Although in the data extraction process (Section 4.1) three instances are extracted for each transit and then augmented, here we classify the vehicle using just the patch extracted from the left part of the scene (see Fig. 9(g)). Motivations for this choice are supplied in Section 6.

When a new image patch I is extracted, it is first resized to the size $\bar{s}$ (see Section 4.1). The HOG features are then extracted and the $\mathbf{x}$ feature vector is obtained. The feature vector is then projected directly to the PCA-LDA space using the expression:

$$z = W_2^T (W_1^T \mathbf{x} + \mathbf{b}_1) + b_2 = \tilde{W}^T \mathbf{x} + \tilde{b}, \tag{7}$$

where $\tilde{W} = W_2^T \cdot W_1^T$ and $\tilde{b} = W_2^T \mathbf{b}_1 + b_2$.

The z value is hence concatenated with the image patch height h and the two-dimensional $\mathbf{p} = (z, h)$ vector is projected to the final LDA space:

$$q = W_3^T \mathbf{p} + b_3. \tag{8}$$

The classification is performed applying the Maximum A Posteriori (MAP) criterion over the probability model defined in (6) using the Bayes rule:

$$p(class = j | q) = \frac{p(class = j) \cdot G(\mu_j, \sigma_j)}{p(q)}, \quad j = 1, 2. \tag{9}$$

The MAP criterion assigns the vector $\mathbf{p}$ to the class j for which $p(class = j | \mathbf{p})$ is maximum. Since we assume uniform priors (i.e., $p(class = j) = \frac{1}{2},\ j = 1, 2$) the inference can be summarized as follows:

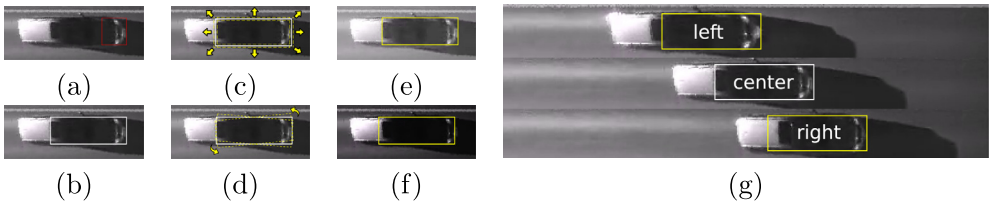

Fig. 9. Data extraction (white) and augmentation (yellow). (a) Original tracking box. (b) Extraction window. (c) Translation variabilities. (d) Rotation variabilities. (e) and (f) Photometric variabilities. (g) Perspective variabilities. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

```
Input: Training Set {I_i}
Output: Classifier (s̄, W_1, b_1, W_2, b_2, W_3, b_3, μ_1, μ_2, σ_1, σ_2)
Compute the mean patch size s̄ and normalize {I_i} to s̄
Extract the HOG features from {I_i} using cellSize and reshape to the
    vectors {x_i}
Compute W_1 and b_1 using PCA on {x_i}
y_i ← W_1^T x_i + b_1
Compute W_2 and b_2 using LDA on {y_i}
z_i = W_2^T y_i + b_2
Aggregate to the image patch height p_i = (z_i, h_i)
Compute W_3 and b_3 using LDA on {p_i}
q_i = W_3^T p_i + b_3
Estimate μ_1 and σ_1 by ML on {q_i | l_i = 1}
Estimate μ_2 and σ_2 by ML on {q_i | l_i = 2}
```

(a) Training Procedure

```
Input: Image patch I, Classifier (s̄, W̃, b̃, W_3, b_3, μ_1, μ_2, σ_1, σ_2)
Output: Estimated class c
Normalize I to s̄
Extract the HOG features from I and reshape to obtain the vector x
z ← W̃^T x + b̃
p ← (z, h) where h is the height of I
q ← W_3^T p + b_3
d_1 ← (q−μ_1)^2 / σ_1
d_2 ← (q−μ_2)^2 / σ_2
if d_1<d_2
   c ← 1
else
   c ← 2
```

(b) Classification Procedure

Fig. 10. Training (a) and classification (b) procedures.

$$class(q) = \arg\max_j G_{\mu_j,\sigma_j}(q). \tag{10}$$

It should be noted that, taking the logarithms, discarding a constant, squaring and dealing with a change of sign, the above expression is equivalent to:

$$class(q) = \arg\min_j \mathcal{M}^2(q, \mu_j, \sigma_j) \tag{11}$$

where $\mathcal{M}^2(q, \mu_j, \sigma_j) = \frac{(q-\mu_j)^2}{\sigma_j}$ is the square Mahalanobis distance between the sample q and the Gaussian distribution $G(\mu_j, \sigma_j)$. Fig. 10(b) shows the pseudocode for the classification process.

5. Controller

In this Section we discuss the controller component. Table 4 reports a list of symbols using in this section.

During the experiments we found that the performances of some modules depend on the speed of the tracked vehicles. This is mainly due to the dependence of the operations involved in the specific modules on the way the information changes between consecutive frames. Moreover the background subtraction module has been introduced specifically to deal with perspective issues related to the tall vehicles. In order to maximize the performances of the overall system on the data, we distinguish between high-speed (60 km/h or more) and low-speed (less than 60 km/h) transits and introduce a controller component which dynamically enables or disables the multicorrelation and selective update modules depending on the estimated speed. The background subtraction module is activated only when a tall vehicle is detected. To get a rough estimation of the vehicle speed at each frame we estimate the displacement of the vehicle as $\delta = \|\mathbf{y}_1 - \mathbf{y}_0\|_2$, where $\mathbf{y}_1$ is the last known position and $\mathbf{y}_0$ is the previous one. We then define two thresholds:

- t_1 used to activate the multicorrelation module when $\delta > t_1$;
- t_2 used to activate the selective update module when $\delta < t_2$.

The performances of the template drift and refinement module are found to be independent from the speed of the vehicle.

Table 4
Table of symbols related to the controller component.

Parameter	Description
$\mathbf{y}_0$, $\mathbf{y}_1$	Positions of the vehicle in the last frame and in the previous one respectively
δ	Estimated displacement of the vehicle
t_1, t_2	Thresholds on the speed of the vehicle used by the controller to activate the modules

6. Experimental settings and results

All the experiments have been performed on the dataset introduced in Section 2 (see Table 1). When the first frame of a given vehicle transit is processed, the labeled tracking box is used to initialize the tracker component. The tracker is then executed in the subsequent frames till the last frame of the transit. The performances of the tracker are assed by manually annotating whether the tracking is successful or not and, in the latter case, annotating also the first frame of failure. The classification is performed once during the transit of the vehicle according to what discussed in Section 6.2. The performances of the classifier are assessed using k-fold cross validation techniques.

6.1. Tracker component

The parameters of the tracking algorithm have been tuned in order to maximize the performances on the reference data. The Normalized Cross Correlation is used as similarity measure for the template matching, the search window size dimensions are 20 px × 12 px, in order to handle vehicles with a maximum horizontal speed of 381 km/h and a maximum vertical speed of 32 km/h. The exhaustive search is performed using an asymmetrical window (forward only) in order to reduce the computation (the vehicles can only move forward or stay still). Since in the given context a scaling factor equals to 0.02 corresponds to less than 1 px and considering in most cases the best scaling factor is in the range $[0.90, 1]$, the scaling factors are taken form this range at step of 0.02. Both the multicorrelation and the selective update thresholds are set to $t_m = t_u = 0.8$. The background subtraction refinement parameters are set to: $t = 4,\ s_m = 7,\ s_d = 5,\ d = 0.2$, while p is set to $p = 1$. The two controller displacement thresholds are set to $t_1 = t_2 = 10$ px which correspond to the speed of 60 km/h.

The quantitative evaluations of the performances of the tracking algorithm are obtained by manually marking each tracked transit either as "successful" or "failed" according to the visually assessed performances. We have also annotated the first frame of failure. In order to analyze the performances of the tracker component, two evaluation measures are used:

Transit Based Accuracy (TBA):
focused on the ability to correctly track the vehicle in all the frames of his transit. This measure is defined as:

$$TBA = \frac{1}{N}\sum_{i=0}^{N-1} s_t(T_i) \tag{12}$$

where N is the total number of transits, $\{T_i\}_{i\in[0,N-1]}$ are the transits and

$$s_t(T_i) = \begin{cases} 1 & \text{if the tracking has no errors} \\ 0 & \text{otherwise} \end{cases}; \tag{13}$$

Longevity Based Accuracy (LBA):
focused on the tracker longevity, i.e., the mean transit percentage correctly tracked before a possible failure. This measure is defined as:

$$LBA = \frac{1}{N}\sum_{i=0}^{N-1} s_l(T_i), \tag{14}$$

where N and T_i are defined as above,

$$s_l(T_i) = \frac{m_i}{n_i}, \tag{15}$$

m_i is the number of frames in which the vehicle is tracked correctly in transit T_i and n_i is the total number of frames in T_i.

We compare the performances of the proposed tracker module with the ones of some relevant approaches discussed in Section 1, namely, CAMshift (Bradski, 1998), Kernel Based Object Tracking (Comaniciu et al., 2003a), Lucas–Kanade optical flow (Lucas & Kanade, 1981) and Tracking Learning Detection (TLD) (Kalal et al., 2009). The CAMShift algorithm gives poor results since the initialization step in the intensity domain fails as shown in Fig. 11(b). This is due to the simplicity of the probability image which does not ensure the maximization of the similarity measure between the target representation and the candidate one. The Kernel-Based Object Tracking algorithm (Comaniciu et al., 2003a) succeeds in the initialization step as shown in Fig. 11(c) but fails in the tracking as shown in Fig. 12(a) and (b). Both CAMShift and Kernel-Based Object Tracking do fail in the gradient orientations feature space since the similarity measure is not a smooth function (no gradient based optimizations are possible) as shown in Fig. 11(a). The Lucas–Kanade optical flow approach (Lucas & Kanade, 1981) gives poor results as shown in Fig. 12(c) due to the violation of the brightness constancy and the spatial coherence constraints caused by the varying light, contrast condition and by the object distortion. Finally, Tracking Learning Detection (Kalal et al., 2009) fails in the last frames of the transits due to the sudden change of appearance as shown in Fig. 12(d). It should be noted that, even if a learning component is included in the TLD algorithm, it cannot cope with previously unseen appearances due to the large changes of appearance caused by distortion and perspective changes.

Fig. 13 shows the results of the proposed approach for each sequence (identified by its relative keyword as described in Section 2) and the global accuracy according to the TBA and the LBA measuring methods. The introduction of the two measuring methods can be justified observing that they measure two different qualities of the tracker. In the Stop and Rotation sequences, it can be noticed that the TBA values are consistently lower than the related LBA values. This happens because the tracker correctly tracks the object for the most part of the scene (obtaining a high LBA score) systematically failing in the last frames of the transit due to poor lighting. Fig. 14 compares the results of the proposed technique with respect to the results of a standard template matching pipeline (as described in Section 1), the TLD algorithm and a tracker based on the estimation of the optical flow for multiple feature points using the Lucas Kanade algorithm. The results are related to the LBA measurement method in order to have a fair comparison since the TBA methods yields low results for the competitor algorithms. The results of the TLD algorithm are related to the implementation in Nebehay (2012). A video showing the results of the compared algorithms can be found at the following link: http://iplab.dmi.unict.it/download/VehicleTracking.avi.

6.2. Classifier component

In the classification experiments a standard implementation of the HOG extraction algorithm is used (Vedaldi & Fulkerson, 2010) setting the parameter *cellSize* = 6. For the augmentation step we set the perspective extraction step $p_s = 20$ px, the translation $p_t = 1$ px, the rotation step $r = 1^\circ$ and the photometric processing values $\alpha = 1.2,\ \beta = 10$. The discarded variability in the PCA computation amounts to the 10%.

The robustness and generalizability of the classifier are assessed with respect to the original data and to the introduced variabilities (i.e., the augmented patches) using k-fold cross validation tests. Moreover we analyze the effect of using balanced data, changing the position where the data is extracted in the scene and considering the patch height. Hence we consider both the proposed pipeline and a simplified one which does not make use of the patch height information. Similarly we consider both the discussed dataset and an unbalanced dataset obtained removing a number of tall vehicle instances in order to get a tall-to-short vehicles ratio approximately equals to 1 : 10. We perform 10-fold tests on the data and evaluate the per-class accuracy considering both the original (not augmented) instances and all the variabilities which have been introduced artificially by augmentation.

Fig. 15(a) shows the accuracy values for the classifier on the unbalanced set (i.e., ratio 1 : 10) when no patch height information is exploited. It can be noted that the accuracy values for the tall vehicles are consistently low. We argue that this is due to the unbalanced nature of the dataset. Fig. 15(b) shows the results for

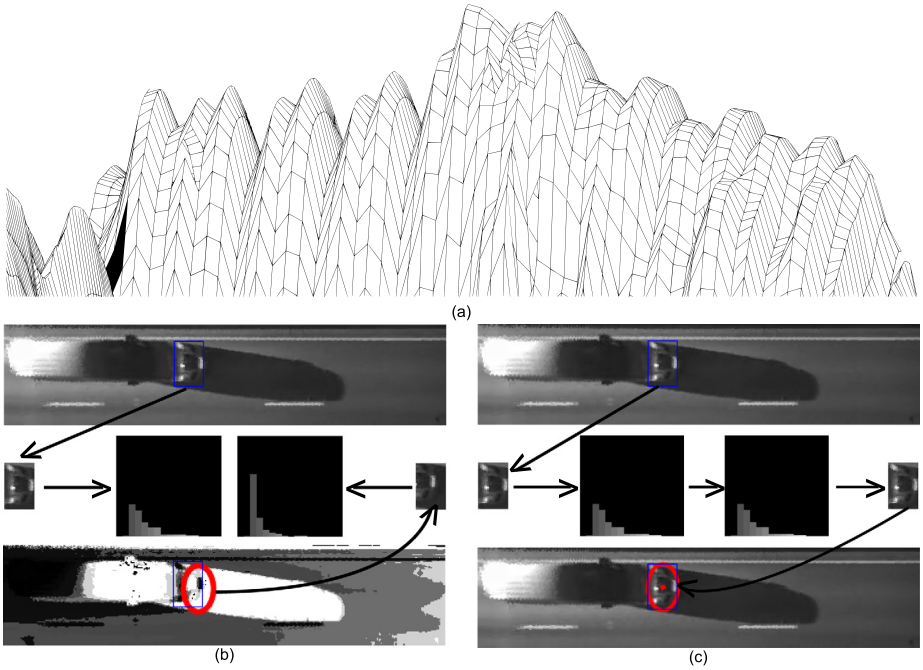

Fig. 11. (a) A detail of the plot of the Bhattacharya coefficients for a given search. (b) The CAMShift probability image approach in the initialization step. (c) The Kernel-Based Object Tracking approach in the initialization step.

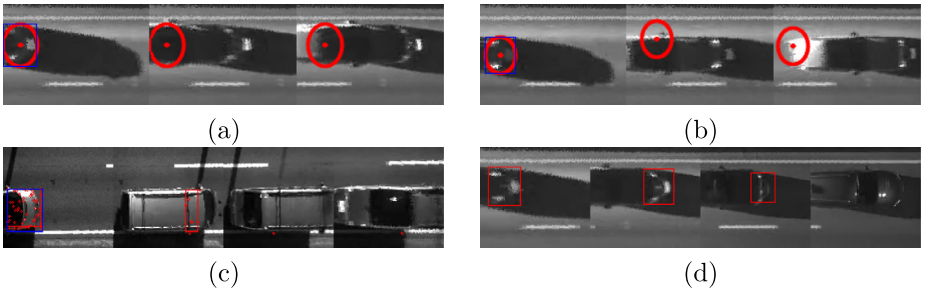

Fig. 12. Example of the failures of the compared trackers. (a) Kernel Based Object Tracking in the intensity space. (b) Kernel Based Object Tracking in the edges orientation space. (c) Lucas–Kanade optical flow. (d) Tracking Learning Detection.

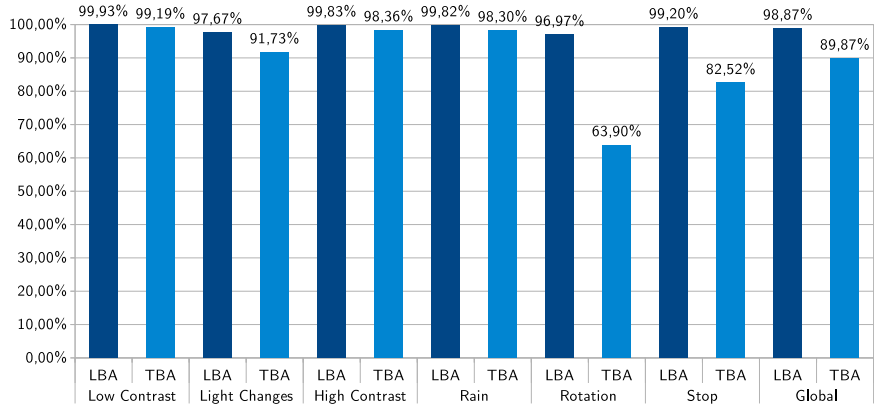

Fig. 13. The results of the proposed technique on the sequences identified by corresponding keywords according to the LBA and TBA measures.

the same classifier when the input dataset has a tall-to-short vehicles ratio approximately equals to 1 : 5 (i.e., the dataset discussed in Section 2). A comparison between Fig. 15(a) and (b) highlights the effect of balancing the dataset and how even better results could be achieved if more data concerning tall vehicles was available.

In both Fig. 15(a) and (b), the accuracy values relative to the perspective variabilities are higher than the ones relative to the

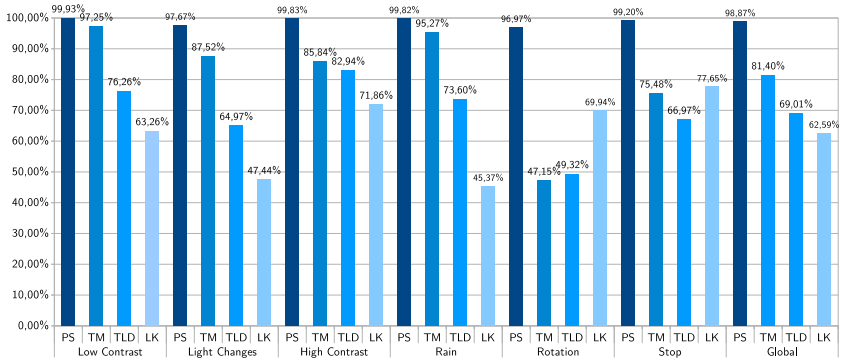

Fig. 14. The results of the proposed technique (PS) vs a standard template matching pipeline (TM), the TLD algorithm (TLD) and a tracker based on the Lucas–Kanade optical flow on multiple feature points (LK). The results are related to the LBA measurement.

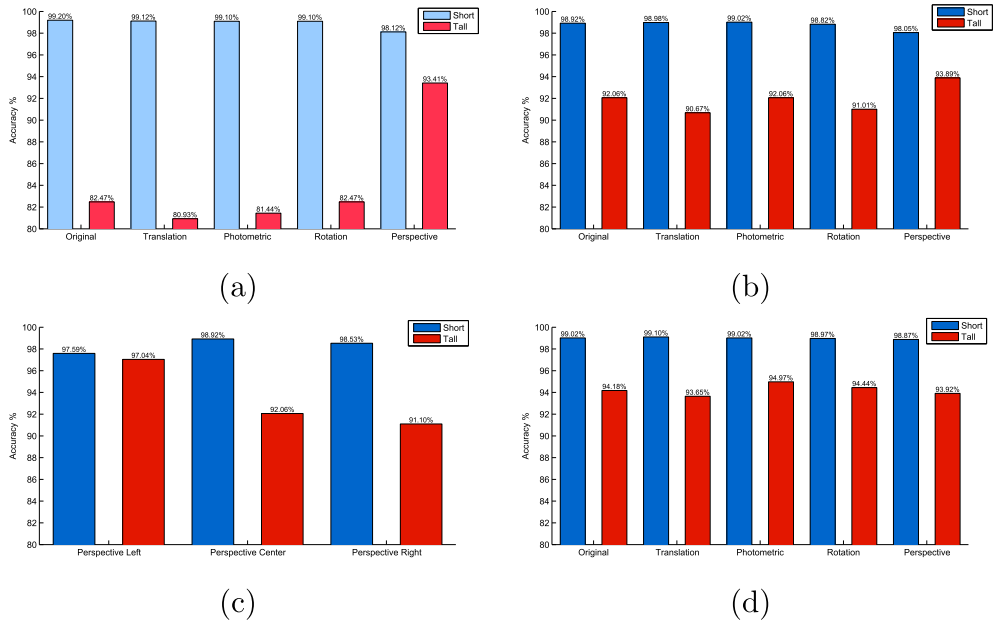

Fig. 15. The diagrams show the results of a series of 10-fold tests on the data. Different color shades indicate different datasets. (a) The accuracy values of the classifier on the unbalanced dataset when no patch height is considered. (b) The accuracy values of the classifier on the balanced dataset when no patch height is considered. (c) A detail of the perspective accuracy of (b) for the different perspective variabilities. (d) The accuracy values for the proposed pipeline exploiting the size information.

original instances for the tall vehicles. This behavior can be explained looking at Fig. 15(c), where a detailed comparison between the accuracy related to instances extracted in different parts of the scene is shown (see Fig. 9(g)). The best accuracy value is found when the patches are extracted from the left part of the scene, which means that in that point, due to perspective reasons, the appearances of vehicles are more discriminative.

Finally, Fig. 15(d) shows the results of the proposed pipeline (including the feature related to the image patch height) on the balanced dataset. An increment in the tall vehicles classification accuracy approximately equals to +2% can be inferred comparing Fig. 15(d) and (b).

7. Conclusion

In this paper we have proposed an integrated system for vehicle tracking and classification suitable for traffic monitoring purposes. The tracking component is based on the template matching method augmented to be able to cope with a series of challenging conditions related to real word sequences such as high variability in perspective, light and contrast changes, object distortions and artifacts in the scene. The effectiveness of our approach has been demonstrated through a series of experiments in critical conditions and comparisons with standard and recent techniques. The classification component is built on the basis of a training set derived from the reference data. In order to get more data for the training phase, the dataset is augmented with artificially introduced patches. The performances of the classifier have been tested considering the real variabilities and the artificial ones, as well as with respect to the variation of different parameters. A controller has been introduced to optimize the performances of the tracker component on the basis of the feedback received by the other modules. The results show that our system outperforms the state-of-the-art algorithms on the considered application domain. Future works could be devoted to test the tracker component on additional sequences including new variabilities (e.g., occlusion). The speed of the tracker could also be improved substituting the exhaustive search mechanism by a meanshift-like optimization algorithm in order to enable real time tracking of multiple vehicles. Moreover, the binary classifier presented in Section 4 could be extended to a multi-class scenario coping with the following classes: cars, vans, trucks, buses and motorbikes. Finally, the performances of the classifier could be improved gathering a new dataset comprising a balanced number of examples for each considered class.

Acknowledgments

This work has been performed in the project PANORAMA, co-funded by Grants from Belgium, Italy, France, the Netherlands, and the United Kingdom, and the ENIAC Joint Undertaking.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.eswa.2015.05.055.

References

Aouaouda, S., Chadli, M., Boukhnifer, M., & Karimi, H. (2014). Robust fault tolerant tracking controller design for vehicle dynamics: A descriptor approach. *Mechatronics*.

Aouaouda, S., Chadli, M., & Karimi, H. (2014). Robust static output-feedback controller design against sensor failure for vehicle dynamics. *IET Control Theory & Applications, 8*, 728–737.

Arnold, W. M. S., Dung, M. C., Cucchiara, R., Calderara, S., Dehghan, A., & Shah, M. (2013). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 99*, 1.

Baker, S., Gross, R., Ishikawa, T., & Matthews, I. (2004). Lucas–kanade 20 years on: A unifying framework: Part 2. *International Journal of Computer Vision, 56*, 221–255.

Bas, E., Tekalp, A., & Salman, F. (2007). Automatic vehicle counting from video for traffic flow analysis. In *IEEE intelligent vehicles symposium* (pp. 392–397).

Battiato, S., Gallo, G., Puglisi, G., & Scellato, S. (2007). Sift features tracking for video stabilization. In *ICIAP international conference on image analysis and processing* (pp. 825–830).

Battiato, S., Cafiso, S., Di Graziano, A., Farinella, G. M., & Giudice, O. (2013). Road traffic conflict analysis from geo-referenced stereo sequences. In *Conference on image analysis and processing* (pp. 381–390).

Battiato, S., Farinella, G. M., Furnari, A., Puglisi, G., Snijders, A., & Spiekstra, J. (2014). Vehicle tracking based on customized template matching. In *VISAPP international workshop on ultra wide context and content aware imaging* (pp. 755–760).

Bilmes, J. A. et al. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute, 4*, 126.

Bradski, G. R. (1998). Computer vision face tracking as a component of a perceptual user interface. In *IEEE workshop on applications in computer vision*.

Comaniciu, D., Ramesh, V., & Meer, P. (2003a). Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25*, 564–577.

Comaniciu, D., Ramesh, V., & Meer, P. (2003b). Mean shift: A robust approach toward feature space analysis. *IEEE Transaction on Pattern Analysis and machine Intelligence, 25*, 564–577.

Dahmani, H., Chadli, M., Rabhi, A., & El Hajjaji, A. (2013). Road curvature estimation for vehicle lane departure detection using a robust Takagisugeno fuzzy observer. *Vehicle System Dynamics, 51*, 581–599.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1*, 886–893.

Farinella, G. M., & Rustico, E. (2008). Low cost finger tracking on flat surfaces. In *Eurographics Italian chapter conference* (pp. 43–48).

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*, 179–188.

Fukunaga, K., & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory, 21*, 32–40.

Hare, S., Saffari, A., & Torr, P. H. (2011). Struck: Structured output tracking with kernels. In *IEEE international conference on computer vision* (pp. 263–270).

Horn, B. K. P., & Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence, 17*, 185–203.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology, 24*, 417.

Hsieh, J., Yu, S. H., Chen, Y. S., & Hu, W. F. (2006). Automatic traffic surveillance system for vehicle tracking and classification. In *IEEE transactions on intelligent transportation systems* (Vol. 7, pp. 175–187).

Kalal, Z., Matas, J., & Mikolajczyk, K. (2009). Online learning of robust object detectors during unstable tracking. In *IEEE international conference on computer vision workshops, ICCV workshops* (pp. 1417–1424).

Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*, 1409–1422.

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. *Imaging, 130*, 674–679.

Maggio, E., & Cavallaro, A. (2011). *Video tracking: Theory and practice*. Wiley.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences, 2*, 49–55.

Nebehay, G. (2012). *Robust object tracking based on tracking-learning-detection* (Master's thesis). TU Vienna: Faculty of Informatics.

Saifia, D., Chadli, M., Karimi, H., & Labiod, S. (2014). Fuzzy control for electric power steering system with assist motor current input constraints. *Journal of the Franklin Institute*.

Shotton, J., Johnson, M., & Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *IEEE conference on computer vision and pattern recognition* (pp. 1–8). IEEE.

Tomasi, C., & Kanade, T. (1991). Detection and tracking of point features. School of Computer Science, Carnegie Mellon Univ.

Vedaldi, A., & Fulkerson, B. (2010). VLFeat: An open and portable library of computer vision algorithms. In *International conference on multimedia* (pp. 1469–1472). ACM.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys, 38*, 13.

Zhou, J., Gao, D., & Zhang, D. (2007). Moving vehicle detection for automatic traffic monitoring. *IEEE Transactions on Vehicular Technology, 56*, 51–59.