An Integrated System for Vehicle Tracking and Classification

Sebastiano Battiato, Giovanni Maria Farinella, Antonino Furnari*

University of Catania - Department of Mathematics and Compute Science

Giovanni Puglisi

University of Cagliari

Anique Snijders, Jelmer Spiekstra

Q-Free, Netherlands

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 (a tracker and a classifier) plus a controller which automatically adapts the configuration of the system to the observed conditions. The tracker component is an augmented version of the template matching which includes four modules designed to cope with the different variabilities exhibited by the data. The classification task is performed with a discriminative model learned on a training set derived from the input video sequences. Experiments show that the proposed tracker outperforms the state-of-the-art algorithms on the considered data. The performances of the classifier are assessed with respect

*Corresponding author
Email addresses: `battiato@dmi.unict.it` (Sebastiano Battiato), `gfarinella@dmi.unict.it` (Giovanni Maria Farinella), `furnari@dmi.unict.it` (Antonino Furnari), `puglisi@dmi.unict.it` (Giovanni Puglisi), `anique.snijders@q-free.com` (Anique Snijders), `jelmer.spiekstra@q-free.com` (Jelmer Spiekstra)

to both real-world and artificially introduced data to test its robustness with respect to the possible variabilities.

Keywords: traffic monitoring, vehicle tracking, vehicle classification

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 Hsieh et al. (2006); Bas et al. (2007); Zhou et al. (2007). We tackle the last two 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 literature Maggio and Cavallaro (2011); Arnold et al. (2013). 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 and Cavallaro (2011). The most straightforward approach is probably the Template Matching, where the object is assumed to be rigid and is represented as an image patch (the template) Maggio and Cavallaro (2011); Yilmaz et al. (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 template and the candidate image patches. If pose changes are consid-

[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.

ered, the Lucas-Kanade affine tracker can be used Lucas and Kanade (1981); Baker et al. (2004). 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 and Cavallaro (2011); Tomasi and 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 scene (brightness constancy assumption Horn and Schunck (1981)). The Lucas-Kanade Optical Flow algorithm Lucas and Kanade (1981) is often used to compute the optical flow. It requires the key-points to satisfy both spatial and temporal coherence constraints. The authors of Tomasi and Kanade (1991) state a criterion to choose which points may be selected as key-points in order to improve the tracker performances (specifically corners or points taken from a high texture 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 et al. (2007), human computer interaction Farinella and Rustico (2008), traffic conflict analysis Battiato et al. (2013)). In Comaniciu et al. (2003a) and in Bradski (1998) the object is represented by describing the image region in which it is contained as a n-bins histogram with respect to 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 finding the probability map relative peak in the neighborhood of the target last known position using the Mean-Shift procedure Fukunaga and Hostetler (1975); Comaniciu et al. (2003b). In the Kernel Based Object Tracking method Comaniciu et al. (2003a) a similarity measure is derived based on the Bhattacharyya coefficient providing a similarity score between the target object representation and the one of the candidate found at a given position. The localization is performed 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 et al. (2009, 2012); Hare et al. (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 modelled 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 algorithms respectively. In Section 5 we discuss the controller component. In Section 6 we report the experimental settings and discuss the results. Finally we conclude in Section 7.

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

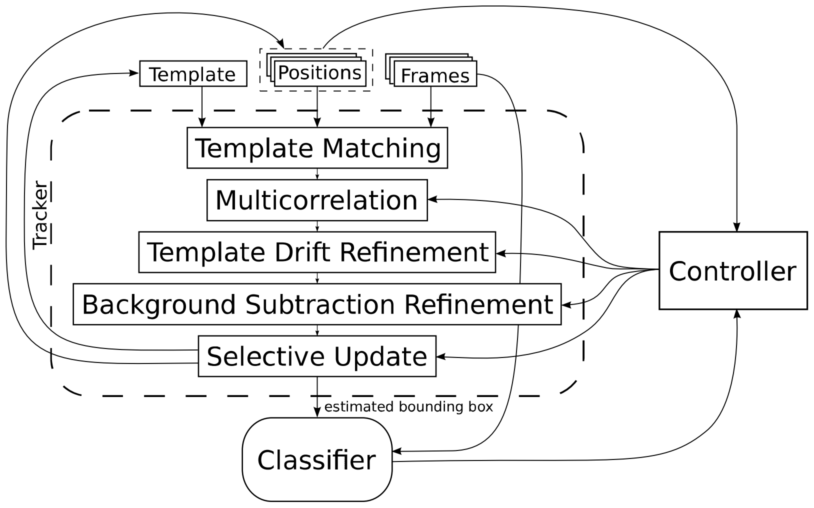


Figure 1: Diagram of the proposed system.

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 a background/foreground segmentation[2]. Both tracking and classification are performed online on real--world data. The system is composed by 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 learning. A controller is introduced to optimize the performances of the tracker by turning the introduced modules on or off on the basis of the feedback received by the different modules. Figure 1 shows the overall schema of the proposed system. The tracker component consists in four modules plus the classic template matching technique which is used to obtain an initial estimate of the vehicle bounding box. The output of the tracker component is used to update the template and to keep track of

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

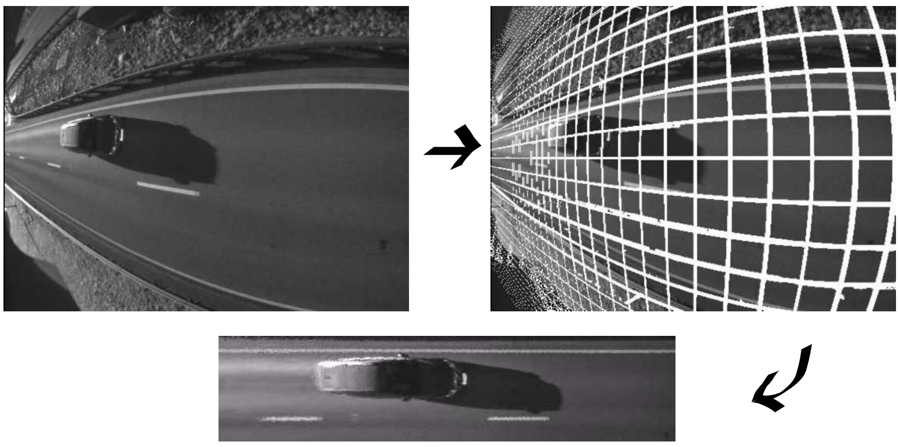


Figure 2: Preprocessing stage.

the vehicle position. The classifier component extracts an image patch of the vehicle from the current frame using the vehicle 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 module by enabling or disabling the modules making considerations on the vehicle trajectories (current and past positions) and the estimated class.

Both the tracker and the classifier 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 of 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 Figure 2. The sequences have been acquired in different places and under different lighting, weather and environment conditions and are identified by a keyword summa-

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
SEQUENCE 2	12	3
SEQUENCE 3	0	86
Total per class	1019	189
Total	1208	

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

rizing 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 tall vehicles examples. 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) since we assume uniform priors for the two classes. The effects of using a balanced dataset are discussed in the experiments (Section 6). The overall data contain 1208 vehicle transits in total.

In analyzing the application context, we highlight the following relevant

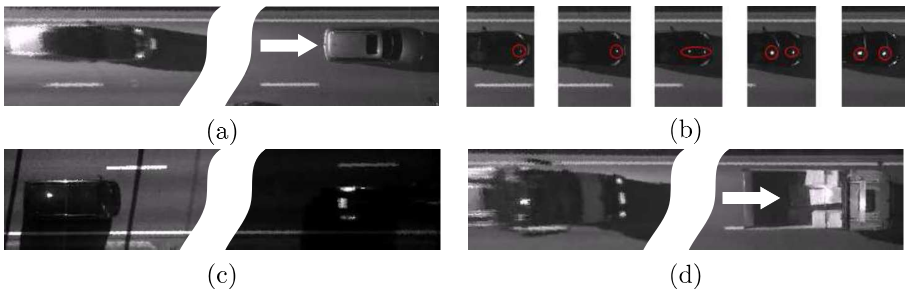


Figure 3: Data variabilities. (a) Object distortion: the figure shows the appearance of the same vehicle in the left and in the right parts of the scene. (b) Artifacts in the form of spots due to the appearance of light reflections on the vehicles. (c) Light and contrast changes: the figure shows the 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.

characteristics of the reference data:

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

3. Tracker Component

The proposed tracking algorithm (see Figure 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 bounding box center (see Figure 9 (a) for a visual example). At each frame an 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 measure. We use this general schema Maggio and Cavallaro (2011) as a baseline and augment it considering different modules which can be dynamically switched on or off by a controller. 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 artefacts (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.

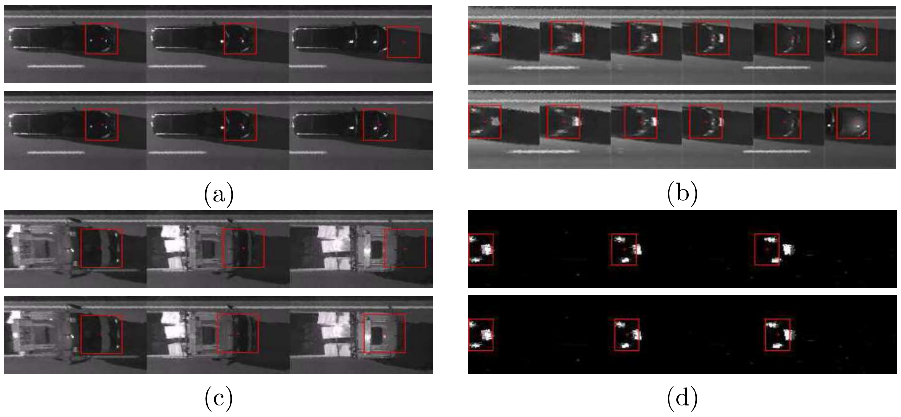


Figure 4: The issues tackled by the introduced modules (top) and the results of the proposed techniques (bottom). (a) Artifacts and multicorrelation. (b) Template drift and refinement. (c) perspective issues and background subtraction based refinement. (d) Slow motion scenes and selective update.

3.1. Multicorrelation

The presence of artifacts (see Figure 3 (b)) contributes to radical changes of the vehicles appearance 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 Figure 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. Figure 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 couple of corresponding blocks and the final score is obtained by averaging the nine subwindows similarity values. A statistical 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. Figure 4 (a) shows the effects of the artifacts on the baseline algorithm and the result of the proposed technique.

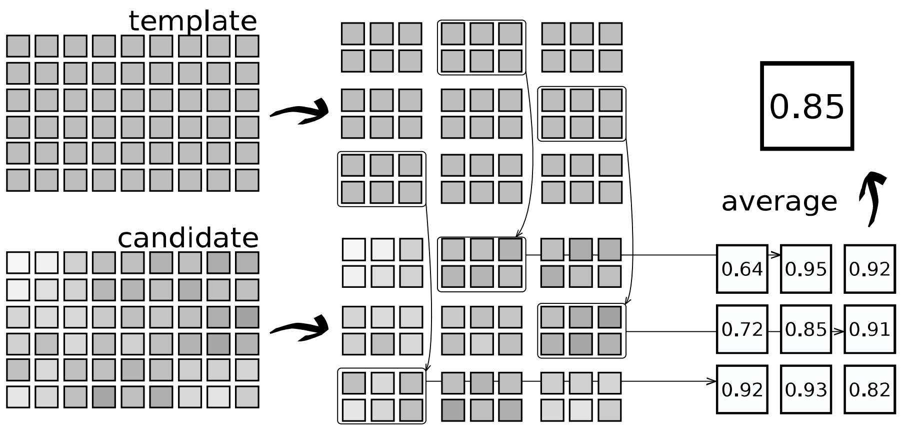


Figure 5: The multicorrelation procedure.

3.2. Template Drift Refinement Module

The presence of light, perspective, contrast changes and distortion, together with the continuous update of the template, generate the template drift problem in the form of the progressive inclusion of the background into the template. This effect is shown in Figure 4 (b). In order to reduce the template drift, a module to refine the vehicle position is introduced (see Figure 6). The refinement is based on the fact that the vehicle is stretched horizontally by effect of the distortion introduced in the preprocessing stage (see Figure 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 original template dimensions in order to enclose more information. An exhaustive search of the object at different horizontal scales would make the algorithm much slower, so, in order to improve the performances, we first perform a regular search (i.e., without any refinement) in order to obtain an initial guess. Afterwards we search for the best match among a number of candidates (called cuts) obtained by discarding the rightmost pixels (the ones which are more likely to contain background information) and horizontally-scaled versions of the template. The

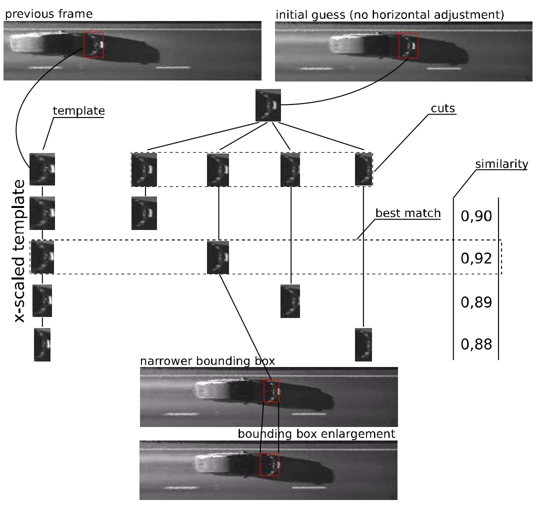


Figure 6: The refinement procedure.

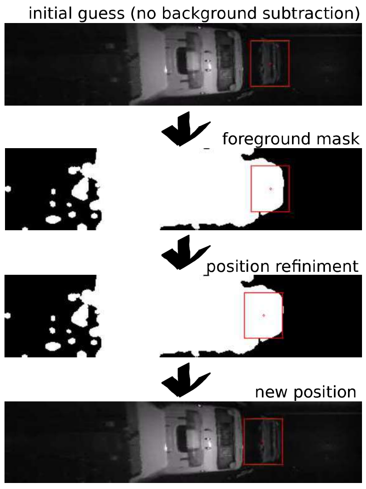


Figure 7: Background subtraction refinement.

best match identifies both the right scale factor and the correct position. The described method can be summarized in the following steps:

1. **Initial Guess:** a regular search (i.e., without any refinement) is performed in order to obtain a first guess of the vehicle position;
2. **Cuts:** for each scaling factor in a given range, a cut is obtained by discarding the rightmost pixels form the initial guess in order to build a candidate of width equals 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.

3.3. Background Subtraction Module

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

$$\Delta f = |f_i - f_{i-1}| \; ; \tag{1}$$

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

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

- 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 Figure 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 number of total 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 maximum accepted 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.

Figure 7 shows an example of applying the above 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 Figure 4 (d). Since the vehicles move very slowly and considering that the object changes of appearance between two consecutive frames are slight, 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. Figure 4 (d) shows the results of the proposed module.

4. Classifier Component

The classification is tackled as a linear classification problem where the data are image patches extracted during the tracking according to the vehicle estimated bounding box. Specifically the classification is performed when the vehicle approaches the central part of the scene, where the perspective variability discussed in Section 2 is less significant. Figure 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

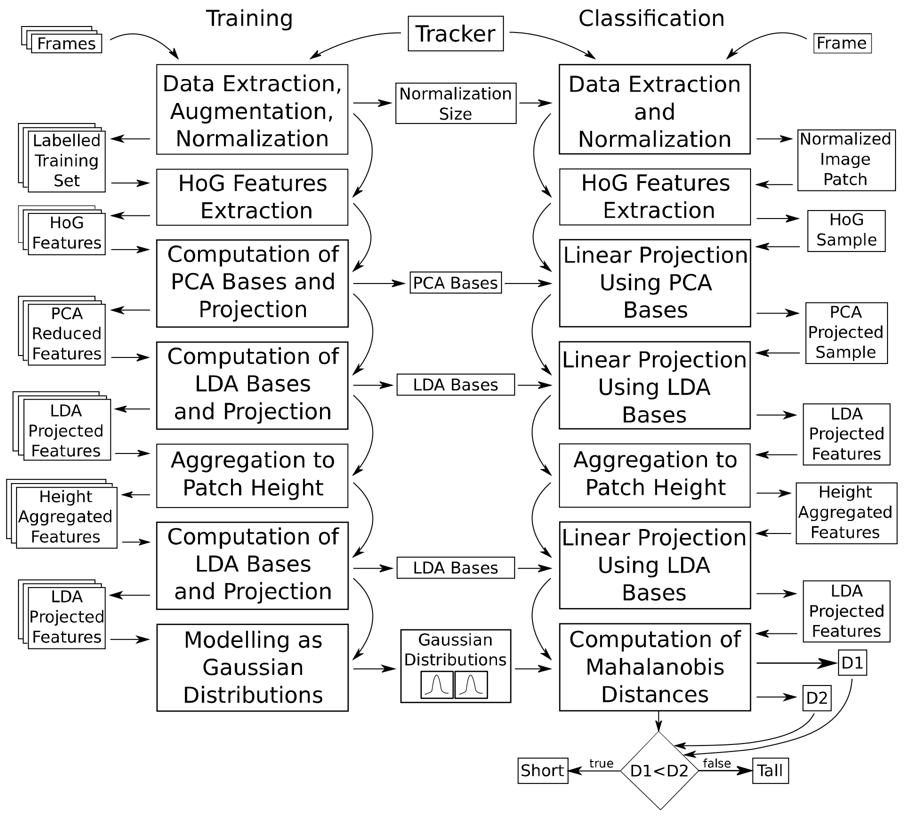

Training
Tracker
Classification
Frames
Frame
Data Extraction, Augmentation, Normalization
Normalization Size
Data Extraction and Normalization
Labelled Training Set
Normalized Image Patch
HoG Features Extraction
HoG Features
HoG Features Extraction
HoG Sample
Computation of PCA Bases and Projection
PCA Bases
Linear Projection Using PCA Bases
PCA Reduced Features
PCA Projected Sample
Computation of LDA Bases and Projection
LDA Bases
Linear Projection Using LDA Bases
LDA Projected Features
LDA Projected Features
Aggregation to Patch Height
Aggregation to Patch Height
Height Aggregated Features
Height Aggregated Features
Computation of LDA Bases and Projection
LDA Bases
Linear Projection Using LDA Bases
LDA Projected Features
LDA Projected Features
Modelling as Gaussian Distributions
Gaussian Distributions
Computation of Mahalanobis Distances
D1
D2
Short
true
D1<D2
false
Tall

Figure 8: The training/classification pipeline

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 feature vectors dimensionality 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 base. 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.

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 Figure 9 (a)-(b)). In order to make the training 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 et al. (2008). For each transit we introduce four variabilities: perspective, translation, rotation and photometric transformation. The perspective variabilities are obtained extracting for each transit three image patches when the extraction window center is found approximately on the scene horizontal central line and when it is found before or after that line at a uniform step p_s (see Figure 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 Figure 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 Figure 9 (d)). The photometric variabilities are obtained extracting two additional patches for any previously extracted patch (including all the

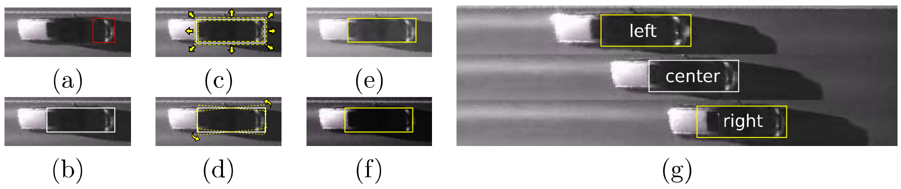


Figure 9: Data extraction (white) and augmentation (yellow). (a) Original tracking box. (b) Extraction window. (c) Alignment variabilities. (d) Rotation variabilities. (e)-(f) Photometric variabilities. (g) Perspective variabilities: the scene central vertical line is in red.

other variabilities) after photometric processing: $I_i^{ph_{1,2}} = (p_\alpha)^{\pm 1} I_i \pm p_\beta$ (see Figure 9 (e)-(f)). The combination of all the variabilities allows to obtain 99 patches per each vehicle labelled in the dataset. All the patches are normalized to the training set mean patch size $\overline{s}$.

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 classifier parameters: the normalization size $\overline{s}$, 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

Gaussian distributions means and variances $\mu_1, \mu_2, \sigma_1, \sigma_2$. The training chain is represented in the left part of Figure 8.

After the feature extraction, the Principal Component Analysis (PCA) Hotelling (1933) is used to reduce the dimensionaly 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. 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 $\overline{q}_i$ to belong to class j is then assumed as:

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

The Gaussian distributions parameters (namely, means μ_1, μ_2 and variances σ_1, σ_2) are estimated through Maximum Likelihood (ML) Bilmes et al. (1998). Figure 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 Figure 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 $\overline{s}$ (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 into the PCA-LDA space using the expression:

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

```
Input: Training Set $\{I_i\}$
Output: Classifier $(\overline{s}, W_1, \mathbf{b}_1, W_2, \mathbf{b}_2, W_3, \mathbf{b}_3, \mu_1, \mu_2, \sigma_1, \sigma_2)$
Compute the mean patch size $\overline{s}$ and normalize $\{I_i\}$ to $\overline{s}$
Extract the HOG features from $\{I_i\}$ using *cellSize* and reshape to the
     vectors $\{\mathbf{x_i}\}$
Compute $W_1$ and $\mathbf{b_1}$ using PCA on $\{\mathbf{x_i}\}$
$\mathbf{y}_i \leftarrow W_1^T \mathbf{x_i} + \mathbf{b_1}$
Compute $W_2$ and $\mathbf{b_2}$ using LDA on $\{\mathbf{y_i}\}$
$\mathbf{z}_i = W_2^T \mathbf{y}_i + \mathbf{b}_2$
Aggregate to the image patch height $\mathbf{p}_i = (\mathbf{z}_i, \mathbf{h}_i)$
Compute $W_3$ and $\mathbf{b_3}$ using LDA on $\{\mathbf{p_i}\}$
$\mathbf{q}_i = W_3^T \mathbf{p}_i + \mathbf{b}_3$
Estimate $\mu_1$ and $\sigma_1$ by ML on $\{\mathbf{q_i} | l_i = 1\}$
Estimate $\mu_2$ and $\sigma_2$ by ML on $\{\mathbf{q_i} | l_i = 2\}$
```

(a) Training Procedure

```
Input: Image patch $I$, Classifier $(\overline{s}, \tilde{W}, \tilde{\mathbf{b}}, W_3, \mathbf{b}_3, \mu_1, \mu_2, \sigma_1, \sigma_2)$
Output: Estimated class $c$
Normalize $I$ to $\overline{s}$
Extract the HOG features from $I$ and reshape to obtain the vector $\mathbf{x}$
$\mathbf{z} \leftarrow \tilde{W}^T \mathbf{x} + \tilde{\mathbf{b}}$
$\mathbf{p} \leftarrow (\mathbf{z}, \mathbf{h})$ where $\mathbf{h}$ is the height of $I$
$\mathbf{q} \leftarrow W_3^T \mathbf{p} + \mathbf{b}_3$
$d_1 \leftarrow \frac{(\mathbf{q} - \mu_1)^2}{\sigma_1}$
$d_2 \leftarrow \frac{(\mathbf{q} - \mu_2)^2}{\sigma_2}$
if $d_1 < d_2$
   $c \leftarrow 1$
else
   $c \leftarrow 2$
```

(b) Classification Procedure

Figure 10: Training (a) and Classificaiton (b) procedures.

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:

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

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 assignment can be summarized as follows:

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

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 $G(\mu_j, \sigma_j)$. Figure 10 (b) shows the pseudocode for the classification process.

5. Controller

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 vehicle displacement as $d = y_1 - y_0$, where y_1 is the last known position and y_0 is the previous one. We then define two thresholds:

- t_1 used to activate the multicorrelation module when $d > t_1$;

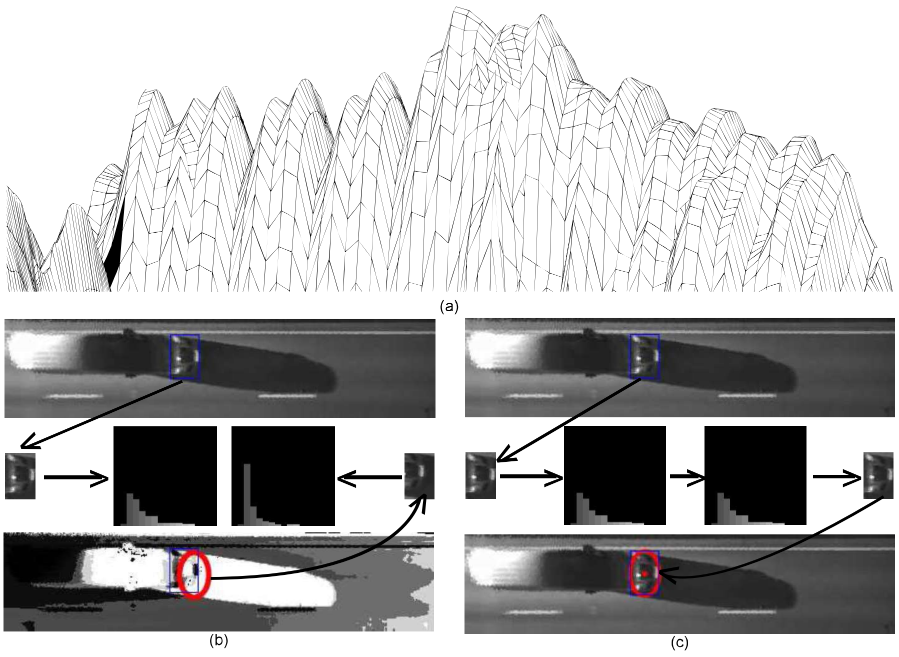


Figure 11: (a) A detail of the plot of the Bhattacharya coefficients for a given search. (b) The CAMShift probability image approach in the initialization step. (c) The Kernel-Based Object Tracking approach in the initialization step.

- t_2 used to activate the selective update module when $d < t_2$.

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

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 tracker performances are assed by manually annotating if the tracking is successful and the first frame of failure. The classification is performed once during the vehicle transit according to what discussed in Section 6.2. The performances of the classifier are assessed using k-fold cross validation techniques.

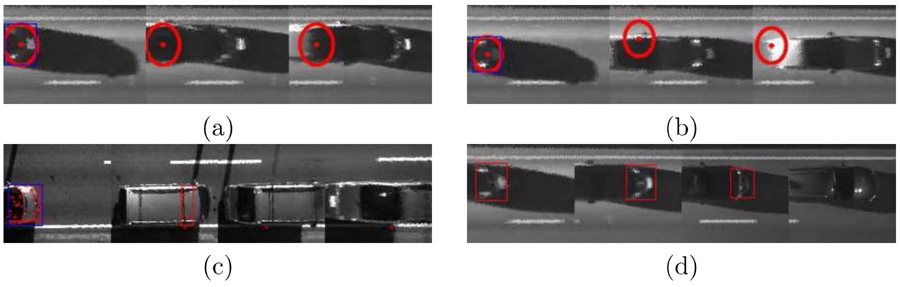


Figure 12: Example of the failures of the compared trackers. (a) Kernel Based Object Tracking in the intensity space. (b) Kernel Based Object Tracking in the edges orientation space. (c) Lucas-Kanade optical flow. (d) Tracking Learning Detection.

6.1. Tracker Component

The parameters of the tracking algorithm have been tuned through a statistical analysis 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 $px \times 12$ 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 that a statistical analysis pointed out that 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 $t_1 = t_2 = 10$ $pixels$ which correspond to the speed of $60 km/h$.

The quantitative evaluations of the tracking algorithm performances 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 methods 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 and 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 Figure 11 (b). This is due to the simplicity of the probability image which doesn't 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 Figure 11 (c) but fails in the tracking as shown in Figure 12 (a)-(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 Figure 11 (a). The Lucas-Kanade optical flow approach Lucas and Kanade (1981) gives poor results as shown in Figure 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 Figure 12 (d). It should be noted that, even if a learning component is included in the TLD algorithm, it can't cope with previously unseen appearances due to the high appearance changes caused by distortion and perspective changes.

Figure 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. Figure 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

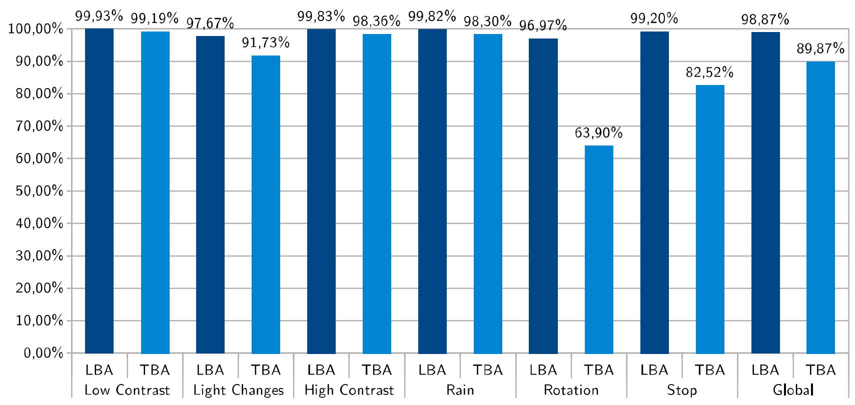


Figure 13: The results of the proposed technique on the sequences identified by corresponding keywords according to the LBA and TBA measures.

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 and Fulkerson (2010) setting the parameter $cellSize = 6$. For the augmentation step we set the perspective extraction step $p_s = 20px$, the translation $p_t = 1\ px$, the rotation step $r = 1°$ 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 doesn't 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

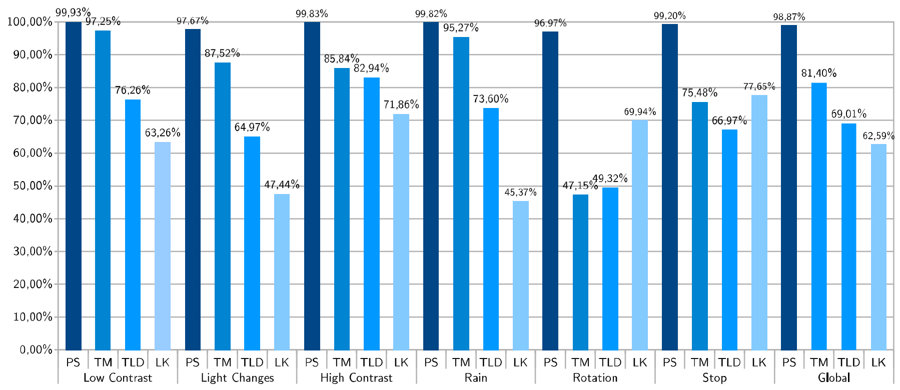
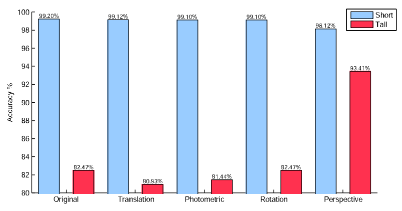


Figure 14: The results of the proposed technique (PS) vs a standard template matching pipeline (TM), the TLD algorithm (TLD) and a tracker based on the Lucas-Kanade optical flow on multiple feature points (LK). The results are related to the LBA measurement.

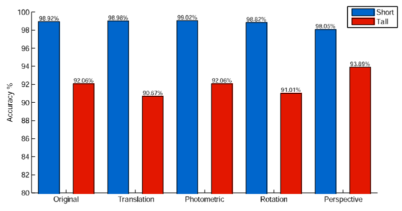
original (not augmented) instances and all the variabilities which have been introduced artificially by augmentation.

Figure 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. Figure 15 (b) shows the results for 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 Figure 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.

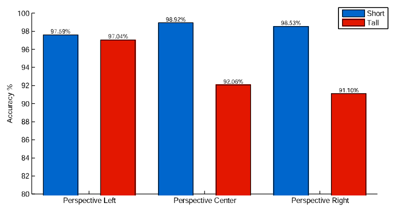
In both Figure 15 (a) and (b), the accuracy values relative to the perspective variabilities are higher than the ones relative to the original instances for the tall vehicles. This behavior can be explained looking at Figure 15 (c), where a detailed comparison between the accuracy related to instances extracted in different parts of the scene is shown (see Figure 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



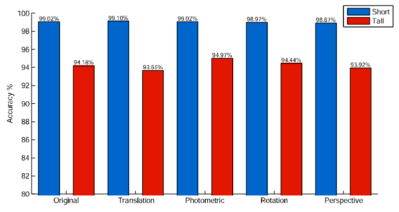
(a)



(b)



(c)



(d)

Figure 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 whole pipeline including the size information.

are more discriminative.

Finally, Figure 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 Figure 15 (d) and (b).

7. Conclusion

In this paper we have proposed a unified system for vehicle classification and tracking 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.

8. Acknowledgements

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.

References

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

Baker, S., Gross, R., Ishikawa, T., Matthews, I.. Lucas-kanade 20 years on : a unifying framework: part 2. International Journal of Computer Vision 2004;56(3):221–255.

Bas, E., Tekalp, A., Salman, F.. Automatic vehicle counting from video for traffic flow analysis. In: IEEE Intelligent Vehicles Symposium. 2007. p. 392–397.

Battiato, S., Cafiso, S., Di Graziano, A., Farinella, G.M., Giudice, O.. Road traffic conflict analysis from geo-referenced stereo sequences. In: Conference on Image Analysis and Processing. 2013. p. 381–390.

Battiato, S., Farinella, G.M., Furnari, A., Puglisi, G., Snijders, A., Spiekstra, J.. Vehicle tracking based on customized template matching. In: VISAPP International Workshop on Ultra Wide Context and Content Aware Imaging. 2014. .

Battiato, S., Gallo, G., Puglisi, G., Scellato, S.. SIFT features tracking for video stabilization. In: International Conference on Image Analysis and Processing. 2007. p. 825–830.

Bilmes, J.A., et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. International Computer Science Institute 1998;4(510):126.

Bradski, G.R.. Computer vision face tracking as a component of a perceptual user interface. IEEE Workshop on Applications in Computer Vision 1998;.

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

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

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

Farinella, G.M., Rustico, E.. Low cost finger tracking on flat surfaces. In: Eurographics Italian Chapter Conference. 2008. p. 43–48.

Fisher, R.A.. The use of multiple measurements in taxonomic problems. Annals of eugenics 1936;7(2):179–188.

Fukunaga, K., Hostetler, L.. The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory 1975;21(1):32–40.

Hare, S., Saffari, A., Torr, P.H.. Struck: structured output tracking with kernels. In: IEEE International Conference on Computer Vision. 2011. p. 263–270.

Horn, B.K.P., Schunck, B.G.. Determining optical flow. Artificial Intelligence 1981;17:185–203.

Hotelling, H.. Analysis of a complex of statistical variables into principal components. Journal of educational psychology 1933;24(6):417.

Hsieh, J., Y., S.H., C., Y.S., H., W.F.. Automatic traffic surveillance system for vehicle tracking and classification. IEEE Transactions on Intelligent Transportation Systems 2006;7(2):175–187.

Kalal, Z., Matas, J., Mikolajczyk, K.. Online learning of robust object detectors during unstable tracking. IEEE International Conference on Computer Vision Workshops, ICCV Workshops 2009;:1417–1424.

Kalal, Z., Mikolajczyk, K., Matas, J.. Tracking-learning-detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 2012;34(7):1409–1422.

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

Maggio, E., Cavallaro, A.. Video tracking: theory and practice. Wiley, 2011.

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

Nebehay, G.. Robust object tracking based on tracking-learning-detection. Master's Thesis Faculty of Informatics, TU Vienna 2012;.

Shotton, J., Johnson, M., Cipolla, R.. Semantic texton forests for image categorization and segmentation. 2008.

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

Vedaldi, A., Fulkerson, B.. VLFeat: an open and portable library of computer vision algorithms. In: Proceedings of the international conference on Multimedia. ACM; 2010. p. 1469–1472.

Yilmaz, A., Javed, O., Shah, M.. Object tracking: a survey. ACM Computing Surveys 2006;38(4):13–es.

Zhou, J., Gao, D., Zhang, D.. Moving vehicle detection for automatic traffic monitoring. IEEE Transactions on Vehicular Technology 2007;56(1):51–59.