Efficient Multi-Camera Detection, Tracking, and Identification using a Shared Set of Haar-Features

Reyes Rios Cabrera[1], Tinne Tuytelaars[1] and Luc Van Gool[1,2]
[1]K.U.Leuven, ESAT – PSI, Belgium. IBBT, Belgium
[2]ETH-Zurich BIWI Switzerland
{reyes.rioscabrera,tinne.tuytelaars,luc.vangool}@esat.kuleuven.be

Abstract

This paper presents an integrated solution for the problem of detecting, tracking and identifying vehicles in a tunnel surveillance application, taking into account practical constraints including realtime operation, poor imaging conditions, and a decentralized architecture. Vehicles are followed through the tunnel by a network of non-overlapping cameras. They are detected and tracked in each camera and then identified, i.e. matched to any of the vehicles detected in the previous camera(s). To limit the computational load, we propose to reuse the same set of Haar-features for each of these steps. For the detection, we use an Adaboost cascade. Here we introduce a composite confidence score, integrating information from all stage of the cascades. A subset of the features used for detection is then selected, optimizing for the identification problem. This results in a compact binary 'vehicle fingerprint', requiring very limited bandwidth. Finally, we show that the same set of features can also be used for tracking. This haar features based 'tracking-by-identification' yields surprisingly good results on standard datasets, without the need to update the model online.

1. Introduction

This paper addresses the problem of detecting, tracking, and identifying vehicles in a tunnel using multiple cameras with non-overlapping views, as illustrated in Figure 1. This is a challenging task given the harsh illumination conditions usually found in tubular passages with artificial illumination [1], [2]. Also the image quality is often relatively poor, with limited resolution, interlacing effects, motion blur, as well as compression artifacts being common phenomena. This makes it difficult to find informative features in the scene. Color, the most widely used feature in traffic applications, is not reliable in tunnels since the artificial lighting often affects the natural colors of objects. Texture information is also very limited, again due to poor illumination, but also because of the low resolution of surveillance cameras (see e.g. the lack of detail on the detected vehicles shown in Figure 2). Motion

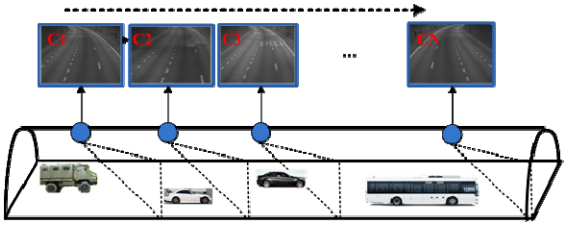


Figure 1: Multi-camera system in tunnel surveillance. Most tunnels are equipped with multiple surveillance cameras. Our goal is to use these to keep track of the vehicle through the tunnel.

information can be reliably used for detection and tracking, but only if the traffic has not come to a stand-still, if vehicles are not too close to each other, and if the road surface is not too reflective (reflecting the headlights over a long range). Moreover, it cannot be used for matching the vehicles between different cameras (also referred to as multi-camera tracking), which is the main goal of this paper.

Vehicle detection, tracking and matching are often addressed separately, as consecutive steps. In this paper, we show how these tasks can work together in an optimal way, sharing the same set of features so as to limit the computational overhead and get acceptable results.

Ideally, detection and tracking should be done on or near the camera using embedded hardware. This means we are on a tight budget as far as computing power is concerned. Another issue is the communication between the cameras and the central server. To limit the bandwidth consumption, we only send a compact, binary 'fingerprint' of each vehicle over the network. Finally, the camera system should work with existing surveillance cameras and not require heavy calibration or parameter tuning for each setup separately ('plug and play' being the ideal).

The main contributions of this paper can be summarized as follows: 1) We show it is possible to build a system for integrated vehicle detection, tracking, and identification for tunnel surveillance; 2) We show this can be done efficiently by reusing the same set of features for all three steps; 3) For the detection, we introduce a novel composite confidence score, integrating information from all stages of the detection cascade; 4) For the identification, we propose a compact, binary vehicle 'fingerprint' that can be

used for vehicle matching across cameras; 5) Finally we introduce Haar features based 'tracking-by-identification', in analogy to 'tracking-by-detection'. This scheme allows taking the specific appearance of the object being tracked into account, without the need to update a model online as in [6], [7].

The remainder of this paper is organized as follows. First, we briefly describe related work. Then, section 2 explains our object detection scheme, with special emphasis to the composite confidence score computation. Section 3 describes the computation of a vehicle fingerprint and how it can be used for vehicle matching. Next, in section 4, the same features are used for tracking. Section 5 describes the integrated system and the experimental results, and section 6 concludes the paper.

1.1. Related work

Sharing of features has received quite some attention in the context of multi-class object recognition [11]. The reuse of features across different tasks, on the other hand, is less studied.

Probably most related to our work is the work of Yuan and Sclaroff [4]. They show that features boosted during cascade training for object detection are not only good for detection, but can also be used as a filter for the subsequent task of within foreground object classification. The selected weak classifiers can be shown to be random bipartitioning hyperplanes following the definition of hashing functions in Locality Sensitive Hashing. As such, they can be used to construct a Hamming distance, approximating nearest neighbor search in the Euclidean feature space. Yuan and Sclaroff use this Hamming distance to quickly select a few candidate classes, which are then further evaluated with a more complex classifier. Here, we use the same Hamming distance for object matching, as well as for object tracking, albeit not in a filter-and-refine framework.

Our vehicle fingerprint, on the other hand, is similar to the small codes proposed by [13] in the context of very large scale image retrieval and classification.

In the context of tracking, tracking-by-detection [12] has become popular recently, since it can recover from errors and temporal occlusions and it is not affected by drift. On the downside, it does not adapt to the object being tracked, as opposed to mentioned appearance-based trackers that can exploit the specific appearance of both the object and the background. Recently, several online classifier-based tracking methods have been proposed, that refine their model during tracking. However, these require extra computation time and cannot be used in our application. The tracking-by-identification we propose is a way to incorporate instance-specific information in the tracking process, without the need to update a model online.

2. Detection – composite confidence score

The poor illumination conditions in tunnels and low quality images make detection very hard. To detect the vehicles we use an implementation of the Viola- Jones detector [3]. This consists of a cascade of strong classifiers, each of which is a combination of several weak classifiers selected using the AdaBoost framework. Using a cascade ensures that most background samples are rejected at early stages with minimal computational effort. At each stage, we aim at rejecting 50% of the background samples while keeping 99.9% of the positive samples. We also use the integral images scheme proposed by Viola-Jones, which allows fast computation of the features independent of the scale or location of the window being evaluated.

To create the cascade, for each stage a strong classifier is constructed based on many Haar features

$$H(x) = \sum_{t=1}^{n} \alpha_t h_t(x) \quad (1)$$

where: $H(x)$ represents a strong classifier, $h_t(x)$ is a weak feature, n is the number of features and α is the weight assigned by the Adaboost algorithm. Normally with threshold a $\theta=0$, if $H(x) < \theta$, the sample is classified as background, while $H(x) >= \theta$ indicates a vehicle. However, for the cascade scheme, θ is tuned so that it lets subwindows classified as background pass to the next stages of the cascade (negative $H(x)$ are considered also vehicles), so as to ensure that almost all positives make it to the next stage.

In our experiments, we found that keeping track of the local output accuracies of each stage, and adding them together in a composite confidence score 'G' (as for *Global*) gives better detections than the standard strategy of only looking at the score of the last stage [3], We define G as:

$$G = \frac{1}{S} \sum_{i=1}^{T} H_i(x) \quad (2)$$

where T is the number of stages of the cascade, and S is defined as:

$$S = \sum_{i=1}^{T} \sum_{t=1}^{n_i} |\alpha_{it}| \quad (3)$$

The variable S functions as a normalization factor of all the weights of the cascade, which can be considered as the global maximum possible response, therefore $-1 \leq G \leq 1$.

The score G can be seen as a way of relaxing the hard decisions made at each stage, distinguishing between those samples that were 'just good enough' versus those in which the classifier was 'very confident'. The computation

of this score is straight forward and it does not need further normalization for each stage, since its value dependents already on the number of classifiers of the stage itself.

Starting from all subwindows that successfully get to the end of the cascade, we then select the final detections by thresholding the score G and applying Non-Maximal Suppression. Additionally, we use the variance of a subwindow as a prefilter before detection. We observed that subwindows containing vehicles typically have a variance between 15 and 60 whereas the variance of most background tunnel subwindows is below 15. We use this information to speed up the detection by dropping subwindows with very low/high variance before the cascade detection. The same variance is also used to variance-normalize the window prior to detection.

3. Matching – the vehicle fingerprint

Matching of vehicles in traffic scenarios is typically done by comparing their appearances and using their kinematics together with inter-camera distances and spatial constraints (e.g. occupied lane), to reduce the number of possible matches. However, matching of vehicle appearances in tunnels is challenging, as illustrated by the examples of vehicles extracted from our database in Figure 2.

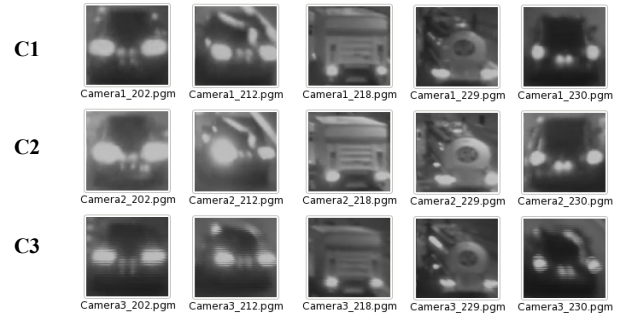


Figure 2. The poor quality of the cameras and hard imaging conditions make vehicle identification in tunnels a difficult task, as illustrated by these four example cars recorded by three different cameras in the same tunnel.

As a result, features typically used for object recognition, like color, local features, edges or PCA-projections have limited success in tunnel applications. Color is not reliable; calculation of invariant local features (such as SIFT or SURF) is computationally demanding and their amount in low resolution images is insufficient; edges are difficult to extract; and PCA-projections may be camera-specific.

Here, we investigate the use of Haar features selected from the pool of weak classifiers computed by the detection cascade. Reusing the features of the detection step avoids wasting time on the computation of new features. Moreover, [4] has shown that these features lend themselves well as Hamming embedding for approximating Euclidean distance, this gives a binary descriptor, which can be sent to the central server where we have strong bandwidth constraints. We experiment with both the standard Hamming distance as well as the weighted Hamming distance (where each bit is weighted by a real value). Note that this optimization scheme is also similar to some approaches used for metric learning [14].

Following the procedure proposed in [4] we start from a training set $S = \{(q_1, a_1, b_1),\ldots,(q_t, a_t, b_t)\}$ of t triples of positive examples. q_i, a_i and b_i are all positive examples. In each triple, a_i is a more preferable neighbor of q_i than b_i. In our case, a_i and q_i represent the same vehicle, while b_i represents a different vehicle. Additionally, we have a set of binary functions $B = \{h_1,\ldots,h_n\}$, where $h_k(x) \in \{-1,+1\}$. Each h_k induces a distance measure:

$$d_k(x,y) = |h_k(x) - h_k(y)|/2 \tag{4}$$

and a weak classifier f_k (f_k is defined on triples, different from h_k):

$$f_k(q_i,a_i,b_i) = d_k(q_i,b_i) - d_k(q_i,a_i) \tag{5}$$

where $d_k(x, y) \in \{0,1\}$ and $f_k(q_i, a_i, b_i) \in \{-1, 0, +1\}$. Our goal in training is to find a strong classifier

$$F(q,a,b) = \sum \beta_j f_j(q,a,b) \tag{6}$$

such that $F(q, a, b) > 0$ for all triples (q, a, b). If we define a new distance measure

$$D_w(x,y) = \sum \beta_j d_j(x,y) \tag{7}$$

and plug Eqn.(5) into Eqn.(6), we have

$$F(q,a,b) = D_w(q,b) - D_w(q,a) > 0 \tag{8}$$

As proposed by [4], we again use AdaBoost to train the strong classifier $F(q,a,b)$, selecting a subset of the features h_k, with corresponding weights β, and a subset of features all with the same weight. Since the weights are fixed, they do not need be sent to the server, only the selected binary values $\{h_k$, resulting in a compact yet performant descriptor.

4. Haar features based Tracking-by-identification

Finally, we propose to use the same set of selected

features used for identification also for the tracking. This may seem a weird choice at first. However, those features have proven to focus on what remains constant over different detection windows, in spite of small variations between the different cameras (viewpoint, pose, lighting, etc.). As such, they can be expected to be robust for tracking as well. Moreover, tracking-by-identification allows taking the appearance of the specific object being tracked into account, without the need for an online boosting scheme, which would be computationally too expensive in our setting.

Tracking-by-identification works just like tracking-by-detection, except that we do not use the confidence of an object detector, but the Hamming distance to the object as seen in the first frame. Moreover, we do not compute all features, but only the optimized subset. In our implementation, we simply compute the Hamming distance for all subwindows close to the detection position in the previous frame.

5. Experimental results

Before we report results on detection (section 5.2), matching (5.3), tracking (5.4), and our integrated system (5.5), we first describe our dataset (5.1).

5.1. Our dataset

For our experiments, we use a set of three annotated video sequences, corresponding to three spatially consecutive but non-overlapping cameras from a tunnel surveillance system in operation. The image resolution is 576x768. All sequences have been manually annotated, assigning to each vehicle a unique identifier and delimiting its bounding box in at least 15 frames. The first part of each video is used for testing, while the second part is used for training. Traffic flow used for testing is composed of around 160 vehicles (mostly cars and trucks) passing by during 3 minutes, the number used for training was about 300.

5.2. Detection

To train the detection cascade, we use the 3 annotated videos as well as 100 additional tunnel images without vehicles downloaded from the internet. Moreover, all positive samples are mirrored horizontally to enlarge the training set. All training data is variance normalized. During testing, the variance normalization is performed online using the squared integral image. The training size is 50x50 pixels for all vehicles.

Figure 3 shows the performance (precision vs. recall) of our detector, once using the score of the last stage of the cascade, as is usually done, and once considering our composite confidence score *G*.

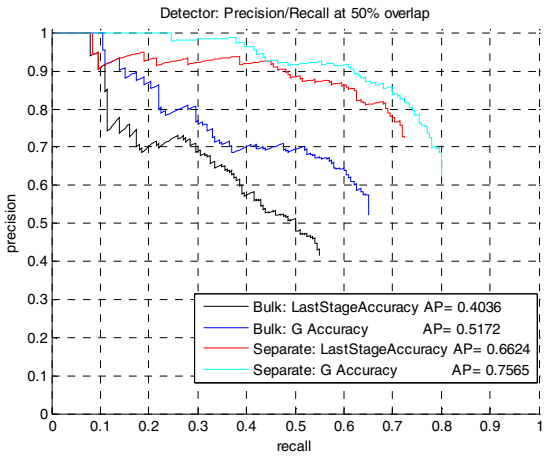


Figure 3: Precision/recall of our detector tested for a bulk cascade (cars and trucks together) and separate cascades (one for cars and other for trucks applying Non-Maximal Suppression).

To define a correct detection, we use 50% overlap (intersection divided by the union) as criterion, as in the Pascal VOC Challenge [8]. To generate the precision/recall plot, a sliding window was run on several annotated frames using the same parameters for both methods. Both the precision as well as the recall benefit significantly from the new confidence measure, with the average precision increasing from 40% to almost 52% and from 66 to 75%. As operating point for further processing, we can select a high threshold, since misdetections can be covered in another frame.

5.3. Matching

Next, we select an optimal subset of features from the pool of features used in the *bulk* detection cascade which contains a total of 890 features. These are used as 'vehicle fingerprints' and sent to the central server, where they are compared with the fingerprints sent by other camera(s) using Hamming distance.

For this experiment, we have manually annotated 400 vehicles in all three cameras, always assigning the same identifier to corresponding vehicles. 200 of these (each with 15 samples with the same ID) are used to optimize the features, while the remaining 200 are used for testing. From the training samples we construct 20,000 triples. Optimization took about 10 seconds in our matlab script.

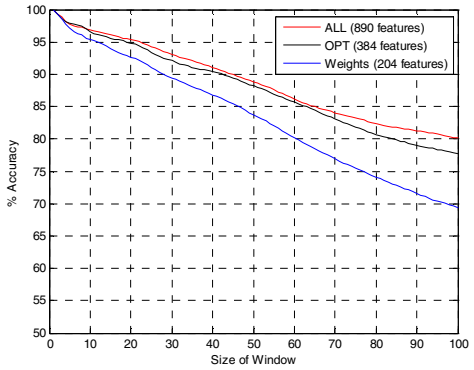
When matching over different cameras, there are usually side constraints that can be exploited limiting the number of matching candidates. For instance, given an estimated time lag between two cameras, we can reduce the search to a small time window. Moreover, the different matchings are not independent. If e.g. the first vehicle of camera 1 matches to the second vehicle of camera 2, no other vehicle of camera 1 can match to that same vehicle. This kind of disambiguation has a big impact on the

matching accuracy. In our experiments, we use the Munkres algorithm for this purpose which is a combinatorial optimization algorithm that solves the assignment problem in polynomial time.

Given all the vehicles detected in one camera in a chronologically ordered list, we use a sliding window running over this list to select matching candidates. The optimal size of this window needs to be determined by experiment.

In figure 4a, we report the accuracy of the matching as a function of the number of distractors, i.e. with the sliding window always centered on the correct detection.

a)



b)

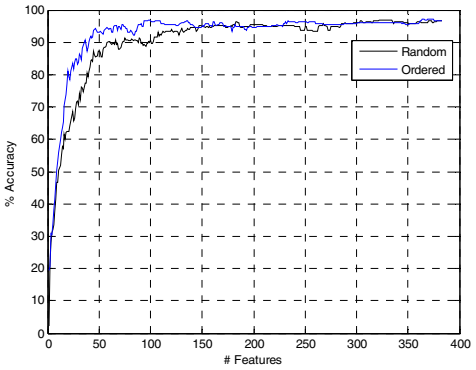


Figure 4. a) Average matching accuracy of Cam1-Cam2, Cam1-Cam3, Cam2-Cam3, with windows centered on correct detection. b) Adding features to improve identification. Using the features in the order that they were boosted (blue) increases accuracy faster than adding them randomly (black). The accuracy saturates very early, at about 100 features.

Since the correct match for all elements of a specific window, is always in the test set, we start with a matching accuracy of 100% (for the case of 0 distractors, window size=1). Clearly, in our setting the weighted version did not improve over the standard Hamming distance, on the contrary. The non-weighted version is almost as good as the result obtained with all the features, while using less than half the number of features. Normally, due to physical tunnel constraints, we consider a maximum of 20 vehicles groups, for which accuracy is 95%.

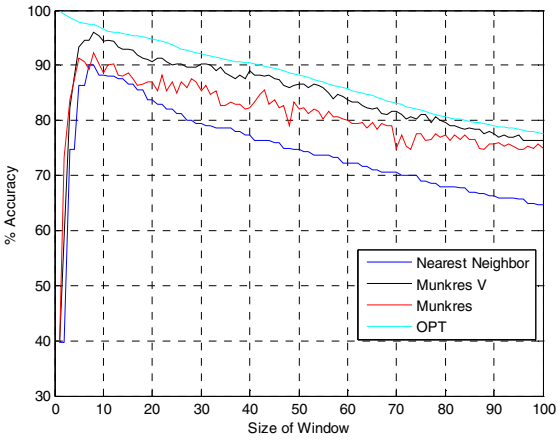


Figure 5. Matching accuracy under more realistic circumstances. The worst performance is the one using only Nearest Neighbor. Munkres, is adding a set of vehicles (+=window/2) and immediately selecting the winner without any voting. Munkres V additionally includes the voting mechanism, and comes very close to the theoretical optimal OPT, which is the same curve as shown in Figure 4a.

Next, we evaluate the effect of the number of selected features on the matching accuracy, see Figure 4b. This plot shows the average for window sizes 10, 15, 20, and 25. Already for 100 features, we obtain an accuracy of 95%. Hence we use only the first 100 features as our vehicle fingerprint. For comparison, we repeated the same experiment also for randomly selected weak classifiers (from the same pool selected by the detection cascade). This also goes up to 95%, but not that fast.

In practice, we obviously do not know the corresponding vehicle beforehand and, as a result, we cannot center the sliding window on the correct detection.

As a result, the size of the sliding window is a tradeoff between two factors: making sure the correct vehicle falls within its range, while limiting the amount of distractors falling in its range. On the other hand, a particular vehicle also falls within multiple sliding windows and, as a result, gets matched several times. To exploit this redundancy, we propose to use a voting algorithm. We add vehicles until an *N*-size buffer is full (with *N* the size of the window). So each vehicle is assigned several times with the Munkres algorithm. If it is assigned to the same vehicle as in the previous iteration, voting for that particular pair of vehicles increases. At each step the pair of vehicles with highest voting value is counted as a valid match and removed from the buffer. In Figure 5, we compare different algorithms for matching the vehicles, namely nearest neighbor based matching, the Munkres algorithm, and the improved Munkres algorithm (including voting). As a reference, we also added the curve of Figure 4a, which can be considered as a theoretical maximum. For all these experiments, we used

ground truth bounding boxes as well as the ground truth order of appearance of vehicles in the tunnel.

5.4. Tracking

For evaluating the tracking, we use the Overlap-Criterion of the Pascal VOC Challenge, which computes the overlap score as

$$(roi_T \cap roi_{GT}) / (roi_T \cup roi_{GT}) \quad (9)$$

with roi_T the tracker detection and roi_{GT} the ground truth.

In Figure 6, we show the tracking accuracy as a function of the number of features. We can see that we obtain a tracking accuracy of 80%. Moreover, the tracking results do not improve much after 100 features, while the processing time increases linearly with the number of classifiers used. The figure shows adding features until 250, in three different ways: ***OPT***: selected in the order they were optimized. ***Random***: selected randomly from optimized set. ***OPT filter***: selected in the order they were optimized but filtering features with size of width OR height < 3 pixels (in the 50x50 training size) and features with width & height < 6 pixels. We can see that small features have a negative impact for the tracking of vehicles.

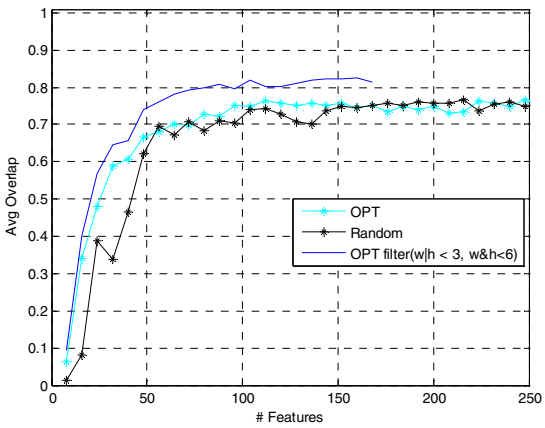


Figure 6. Tracking accuracy versus the number of features used. Average of tracking 20 vehicles, during 30 frames, computing at every 5 frames.

Additionally, in order to further evaluate the proposed method for tracking, we use the benchmark videos from [6], [7] to compare our results. [6], [7] have proposed an online tracker based on multiple instance learning and multiple instance boosting. We found that using a simple set of identification-optimized features without online model updating, can already achieve reasonably good results. For these experiments, we use the 144 first features from the same set of 384 features optimized for vehicle identification.

Our tracker is clearly competitive with state of the art obtaining the best and second best score for 2 of the sequences. It showed considerable lower results in videos where the starting position was not very representative for the whole sequence, see Table 1, Figure 7. In the tiger1, coke1 videos, tracking is carried out properly the first frames, but then the object is lost and hardly found again. Since we are using only the first fingerprint, it cannot perform well on these two videos. However for our aimed application (the tunnel surveillance) success rate is very good and we can successfully track any vehicle.

Sequence	OUR	MILSER	MIL	OSB	OAB
sylv	0.53	**0.63**	0.61	0.46	0.50
david	0.48	**0.71**	0.54	0.31	0.32
faceocc1	**0.73**	0.68	0.63	0.71	0.38
faceocc2	0.65	**0.78**	0.65	0.63	0.64
coke1	0.10	0.18	**0.29**	0.12	0.20
tiger1	0.32	**0.60**	0.51	0.17	0.27

Table 1: Comparison of different tracking methods on the videos: our method, MILSERboost, MILboost, Online Semi-boost and online Adaboost, table from [6]. Bold=1st, Underline= 2nd place

Figure 7. Snapshots of some tracking results: our tracking-by identification method (blue) is often competitive with the more complex MILBoost algorithm of [7] (red).

5.5. Our integrated system

We have integrated a system for detecting, identifying and tracking vehicles along a tunnel pipe corresponding to the range of view of the 3 cameras used for the experiments. Figure 8 shows an image of the system in operation. To increase performance, other information was integrated into the system, such as motion estimation as well as the lane where the vehicle is in.

Figure 8. Snapshots of our system in operation. Tunnel camera 1, 2, 3.

When computing the Hamming distance of vehicle x and vehicle y, we adapt the distance if vehicles are from different lanes.

$$D_{new}(x,y) = D(x,y) + \alpha|l_x - l_y|D(x,y) \qquad (10)$$

where D is the distance and l is the lane number, and α is a parameter determined empirically. With this we integrate the low probability of a car changing lane from camera I to $I+1$, into our identification matrix.

6. Conclusions and discussion

We presented a general framework for tunnel surveillance using non-overlapping cameras. Reusing the same set of features for the detection, the tracking as well as the matching over different cameras allows keeping the computation time under control, and yields promising results. In particular, we manage to improve the detection accuracy over a standard Adaboost cascade framework by using a novel confidence score. We show inter-camera matching accuracies of 95 % and up using a compact (100bit) vehicle fingerprint and we obtain tracking results competitive with the state of the art, in spite of the simplicity of our algorithm. Different components (composite confidence score, tracking framework) have been used successfully in other contexts as well. This suggests that our proposal could be applied to other multi-camera tracking scenarios.

An issue not addressed in this paper is the recovery of errors from one camera to another. With many cameras it is possible to recover such errors using other detections and matchings in a similar way as the voting scheme, which can reduce missed/misclassified vehicles.

We evaluated our multi-camera tracking experiments on one tunnel only. Further validation is required to proof transferability to other tunnels. Also a hybrid combination between tracking-by-detection and tracking-by-identification would lead to better tracking results. This is left as future work.

Acknowledgement:

Part of this work was funded through the IBBT-VICATS project and ERC grant COGNIMUND.

References

[1] Kastrinaki, V., Zervakis, M., Kalaitzakis, K.: A survey of video processing techniques for traffic applications. Image and Vision Computing 21(4):359 – 381, 2003.

[2] Alper Yilmaz, Omar Javed , Mubarak Shah, Object tracking: A survey, ACM Computing Surveys (CSUR), v.38 n.4, p.13-es, 2006.

[3] P. Viola, M. Jones, Rapid Object Detection Using a Boosted Cascade of Simple Features, IEEE CVPR 2001.

[4] Quan Yuan; Sclaroff, S., Is a detector only good for detection?, IEEE ICCV 2009.

[5] Jianyu Wang, Xilin Chen, Wen Gao, Online Selecting Discriminative Tracking Features Using Particle Filter, IEEE CVPR 2005, vol. 2, pp.1037-1042,

[6] Zeisl Bernhard, Leistner Christian, Saffari Amir, Bischof Horst, Online Semi-Supervised Multiple-Instance Boosting, IEEE CVPR 2010.

[7] Boris Babenko, Ming-Hsuan Yang, Serge Belongie, Visual Tracking with Online Multiple Instance Learning, IEEE CVPR 2009, Miami, Florida.

[8] Everingham, M. and Van Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A.,The PASCAL Visual Object Classes Challenge 2010 (VOC2010)

[9] H. Grabner, C. Leistner, and H Bischof. Semi-supervised on-line boosting for robust tracking. In ECCV, 2008

[10] S. Stalder, H. Grabner, L. van Gool, Beyond Semi-Supervised Tracking: Tracking Should Be as Simple as Detection, but not Simpler than Recognition. In IEEE ICCV, WS on On-line Learning for Computer Vision, 2009.

[11] A. Torralba, K. P. Murphy and W. T. Freeman, Sharing visual features for multiclass and multiview object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence , vol. 29, no. 5, pp. 854-869, May, 20

[12] Kenji Okuma, Ali Taleghani, Nando De Freitas, James J. Little, and David G. Lowe, A Boosted Particle Filter: Multitarget Detection and Tracking, ECCV 2004

[13] Torralba, A.; Fergus, R.; Weiss, Y.; Small codes and large image databases for recognition, IEEE CVPR2008

[14] Distance Metric Learning: A Comprehensive Survey. Liu Yang. Michigan State University, May 19, 2006.