Real-Time Visual Detection and Tracking System for Traffic Monitoring

Mauro Fernández-Sanjurjo, Brais Bosquet, Manuel Mucientes, Víctor M. Brea

Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS)
Universidade de Santiago de Compostela, Santiago de Compostela, Spain

Abstract

Computer vision systems for traffic monitoring represent an essential tool for a broad range of traffic surveillance applications. Two of the most noteworthy challenges for these systems are the real-time operation with hundreds of vehicles and the total occlusions which hinder the tracking of the vehicles. In this paper, we present a traffic monitoring approach that deals with these two challenges based on three modules: detection, tracking and data association. First, vehicles are identified through a deep learning based detector. Second, tracking is performed with a combination of a Discriminative Correlation Filter and a Kalman Filter. This permits to estimate the tracking error in order to make tracking more robust and reliable. Finally, the data association through the Hungarian algorithm combines the information of the previous steps. The contributions are: (i) a real-time traffic monitoring system robust to occlusions that can process more than four hundred vehicles simultaneously; and (ii) the application of the system to anomaly detection in traffic and roundabout input/output analysis. The system has been evaluated with more than two thousand vehicles in real-life videos.

Keywords: Computer Vision, Traffic Monitoring, Object Detection, Visual Tracking

Email address: `mauro.fernandez@usc.es`, `brais.bosquet@usc.es`, `manuel.mucientes@usc.es`, `victor.brea@usc.es` (Mauro Fernández-Sanjurjo, Brais Bosquet, Manuel Mucientes, Víctor M. Brea)

1. Introduction

Traffic monitoring through computer vision systems allows solving tasks like vehicle counting, accident detection, roundabout entry/exit analysis or assisted traffic surveillance. The goal of a traffic monitoring system is to provide a framework to detect the vehicles that appear on a video image, and estimate their position while they remain in the scene. A complete traffic monitoring application requires the integration between detection and tracking. Besides, in real-life traffic scenarios, two requirements are especially important: occlusion handling and real-time performance, especially when there are many vehicles in the scene.

From the computer vision point of view, we can distinguish two different approaches for tracking: *low-level* and *high-level* tracking. In this work, we consider *low-level* trackers those algorithms that, once initiated with a detection bounding box, estimate the position of the object in the new frame exploiting just the visual information. They model the appearance of the object of interest by extracting features and then searching them in the subsequent frame [1, 2, 3]. These algorithms cannot handle total occlusions and do not provide a framework for multiple object tracking. In addition, the best current solutions do not operate in real-time with more than one object on a CPU [2, 4].

High-level trackers on the other hand, use, apart from visual features, more complex information to estimate the new object position, like probabilistic motion models, data association, maps of the environment, etc. In recent years, the most standard solution to high-level tracking has been the tracking-by-detection approach [5]. This framework considers the tracking task as a data association problem between detections and trackers over time. Solutions to this approach offer a high-level of precision and robustness, some of them, with a reduced computational cost, but they assume the existence of reliable detections in every frame of a video, which is a restriction for real-time performance in a real-life application, as current state-of-the-art deep-learning based detectors operate above 75 ms per frame [6]. In addition, many of them assume that the detections are perfect, which in a real scenario is unrealistic as often there are false positives, wrong framed or not identified objects.

In this paper, we present a detection and tracking system for traffic monitoring that operates in real-time with multiple objects, and handles total occlusions. The system is composed of a deep-learning based detector, a low-level Discriminative Correlation Filter (DCF) based tracker, a high-level Kalman Filter based tracker and data association based on the Hungarian algorithm. The contributions of our proposal are:

- A traffic monitoring system that can process **more than 400 vehicles simultaneously** in videos with HD resolution in real-time.
- The system also **handles occlusions** by detecting the upcoming occlusion and searching the occluded vehicle in a zone called *Search ROI (Region-Of-Interest)* that is proportional to the error degree in the tracking process. We provide a metric for **on-line tracking failure detection** by estimating the distance between two independent tracking methods allowing us to update the system's tracking error accordingly.
- We extend our system for solving **two real-life traffic applications**: roundabout I/O (Input/Output) analysis and traffic anomalies detection. We perform experimental evaluation using state-of-the-art tracking metrics of the system and its extensions using more than 2,000 vehicles.

The rest of this paper is structured as follows. Section 2 gives an overview of closely related work. In Section 3 we explain the details of our approach. In Section 4 we perform extensive experiments to validate our proposal and introduce the traffic applications developed. Finally, conclusions are given in Section 5.

2. Related Work

Traffic monitoring systems detect and track all the vehicles in a video sequence. This task presents two main challenges: to manage total occlusions and to operate in real-time with multiple vehicles.

2.1. Detection

Given an image and a set of object categories, detection consists of identifying all the objects in the image by placing a bounding box around them, and classifying them in the corresponding category. In recent years, since the upswing of Deep Convolutional Neural Networks (CNNs or ConvNets), this technique has been the mainstream for object detection.

ConvNets are mainly based on consecutive convolution operations that extract features from the images. As the convolution layers become deeper, resultant features become more complex. The first ConvNets were adapted to classification problems triggered by the increasing performance of GPUs. Among them, some of the most relevant architectures have been AlexNet [7], which was the pioneer, followed by VGG [8] that uses smaller convolution filters to improve learning and, finally, ResNet [9] that made its structure even deeper.

The work in the field of image classification was later extended to perform object detection. The first ConvNet for object detection was R-CNN [10] (Regions with CNN features), which uses a region proposal algorithm (such as selective search [11] or edge boxes [12]) to generate possible locations, and applies a classification network to each of them. Improving the previous approach, Fast Region-based Convolutional Neural Network (Fast R-CNN) [13] calculates the deep features for the whole image and, then, projects each of the regions on the last feature map and classifies them, thus saving a lot of computing time. Finally, becoming a milestone in the object detection field, Faster Region-based Convolutional Neural Network (Faster R-CNN) [14] introduces a region proposal algorithm based entirely on a ConvNet called the Region Proposal Network (RPN). The RPN uses the information from intermediate layers of a standard classification network to provide different locations in which an object may appear.

To improve the performance of the proposal of regions in all possible scales, Lin et al. [15] replicate the RPN from Faster R-CNN in several layers of the network in which deeper feature maps are combined with shallower ones. The shallower the layer the smaller the object it will locate. This approach, called Feature Pyramid Network (FPN) obtains outstanding results as shown in the COCO detection challenge 2016 [16]. All these approaches present a high level of performance but, their main limitation is their computational cost, which makes them harder to use in applications that demand real-time performance.

Another kind of detection ConvNets are called one-shot. These methods generate candidate regions directly from feature maps instead of having a specific network (RPN) for this task. By doing this, they reduce their computational time at the expense of lower quality detections. Remarkable approaches of this type are: Single Shot MultiBox Detector (SSD) [17], You Only Look Once (YOLO) [18] and RetinaNet [19].

2.2. Tracking

2.2.1. Low-level tracking

In recent years two alternative on-line approaches for low-level trackers have dominated the state-of-the-art: Discriminative Correlation Filter (DCF) based trackers, and deep-learning based trackers. On the one hand, DCF based trackers predict the target position training a correlation filter that can differentiate between the object of interest and the background. The first DCF solutions were focused on a sole feature (commonly intensity values), and a single filter per tracked object [20]. From that point on, continuous increase in performance has been made by incorporating multi-dimensional features [21] such as Histogram of Oriented Gradients (HOG) [22] and color, implementing scale estimation [23] and other extensions like non-linear kernels [24], long-term memory components [25] and others [26, 27]. Improvements in robustness and accuracy have been made at the expense of decreasing the tracker speed, going from 172 fps in [24] to 0.3 fps for the 2017 Visual Object Tracking (VOT) winner C-COT [2]. This is a continuous trend as can be seen in VOT 2018 [28] results, where the top algorithm on the public dataset, Learning Adaptive Discriminative Correlation Filters (LADCF) [29] still operates at a speed of 1.3 fps on CPU and

10.8 fps on GPU. This increase in computational cost limits the use of the most advanced state-of-the-art tracking solutions in real-life applications.

On the other hand, deep-learning based trackers use ConvNets. SiamFC [1] is one of the first approaches of this kind. This tracker consists of two branches that apply an identical transformation —deep features extractor— to two inputs: the search image and the exemplar. Then, both representations are combined through cross-correlation, generating a score map that indicates the most probable position of the object. In [30] SiamRPN is proposed, adding a Region Proposal Network (RPN) to a siamese network in order to generate bounding box proposals that go through a classification and a regression branch. DaSiamRPN [31] improves SiamRPN, focusing the training on semantic distractors, and adding a search region strategy for long-term tracking. This trackers showed excellent performance in tracking metrics in the VOT challenge [32, 28]. However, none of these trackers can cope with occlusions, nor by themselves provide a framework to deal with multiple objects.

2.2.2. High-level tracking

Due to the increase in performance of deep learning detectors in recent years, the task of tracking is increasingly being seen as a data association problem, i.e. tracking-by-detection. In this approach, the primary concern is to assign detections to trackers over time. Some international challenges [5] have emerged to rank solutions to this problem, evaluating precision, robustness and speed among other performance metrics. In the past few years, complex solutions to this tracking approach that obtain outstanding results have appeared. Some of them focus on extending traditional high-level tracking approaches. Kim et al. [33] and Chen et al. [34] propose extensions to the classical multiple hypotheses tracking (MHT) [35]. The former introduces on-line appearance representations and the latter enhances the detection model of classical MHT. Others emphasize the need to efficiently combine multiple cues over a long period of time [36, 37, 38]. Ultimately, some work has emerged to provide algorithms specialized in tracking and identification of non-rigid objects [39, 40].

All these approaches have demonstrated good performance in classic multiple object tracking metrics as commented before. Their fundamental limitation is the speed, as none of the work discussed in this section shows performance metrics above 2.6 Hz even without accounting for the detection time. This is clearly a limitation for real-time applications in which tracking is merely one of the modules involved. Still, solutions like [41] and [42] trade simplicity for speed assuming reliable detections at every frame, which is clearly far from a real situation even with today high performance video object detectors [43].

2.2.3. Traffic monitoring

Some work in the traffic monitoring field has been done in the recent years [44]. In [45], vehicle counting is performed employing an environment segmentation strategy. In [46] a tracking approach using background subtraction and Kalman filter tracking to tackle the data collection in roundabouts is proposed. These approaches usually run at real-time speed due to the use of background subtraction for detecting mobile objects. These object identification methods could represent a limitation in scenarios that present camera movement (on-board cameras), shadows, image artifacts, or objects that appear very close to each other since they usually are identified as only one by the background subtraction algorithm. Also, in [47] a method for vehicle tracking and speed estimation with multiple cameras is proposed, although this solution does not operate in real-time due to its complexity.

3. Video Traffic Monitoring

We propose a complete traffic monitoring system that combines tracking and detection and can operate as a baseline for multiple applications.

3.1. System Overview

Our system is made up of three blocks (Figure 1): detection, tracking and data association. To detect vehicles in an image, we use a deep learning based detector. For tracking, we combine a DCF-based tracker with a Kalman-based

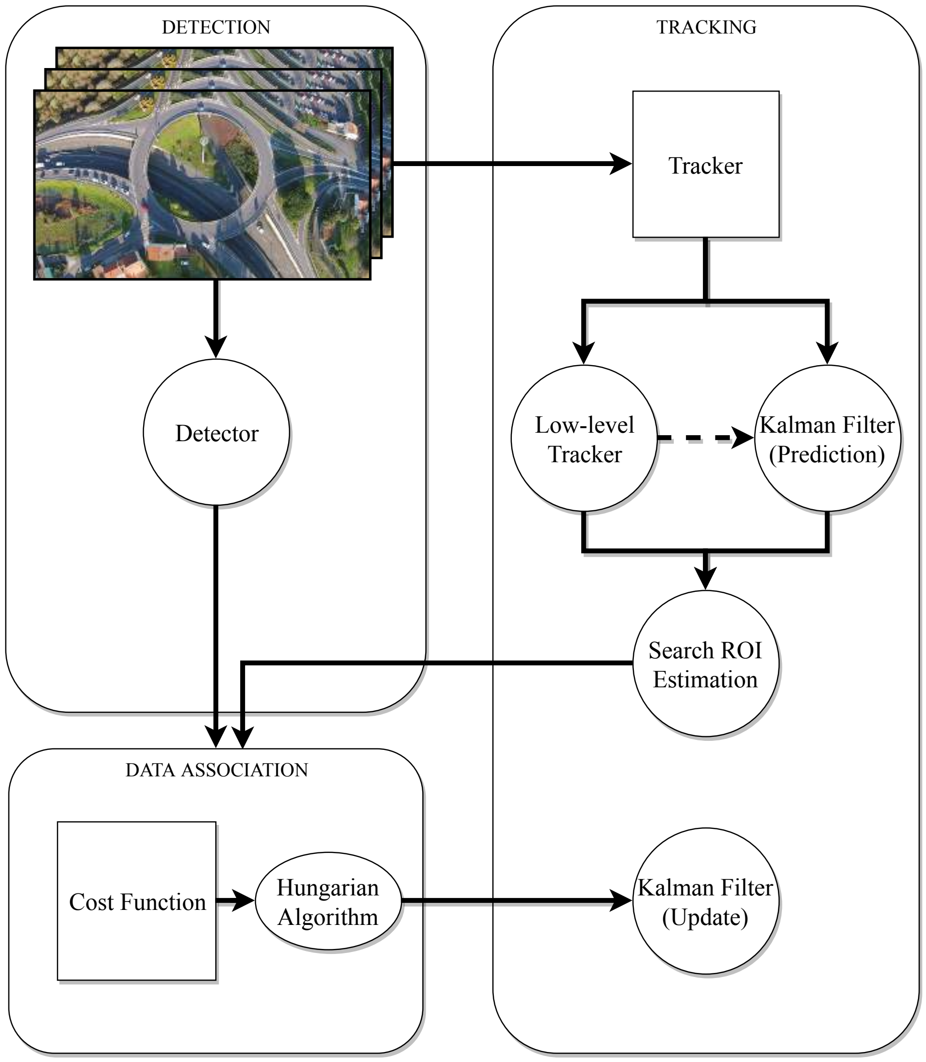


Figure 1: Traffic monitoring system.

one, which enables to calculate a failure detection metric to identify occluded vehicles. Finally, in the data association module, we assign each detection with its correspondent tracker through the Hungarian method [48, 49] and perform an update of the trackers.

Algorithm 1 presents the main steps of the system. The inputs to the system at every time instant t are the new frame (Im_t) of the video, and the set of

Algorithm 1: Traffic Monitoring System

```
Require:
    (a) $Im_t$: Image frame at current time $t$
    (b) $\Phi_{t-1} = \{\varphi^1_{t-1}, \varphi^2_{t-1}, \ldots, \varphi^n_{t-1}\}$
1 Function Main($Im_t$, $\Phi_{t-1}$):
2     $\overline{\Phi}_t$ =Tracking_Prediction($\Phi_{t-1}$,$Im_t$)
3     if $time\_elapsed > \tau$ then
4         $\Psi_t$ =ConvNet_Detect()
5         $\Phi_t$ =Tracking_Update($\overline{\Phi}_t$,$\Psi_t$,$Im_t$)
6     else
7         $\Phi_t = \overline{\Phi}_t$
8     return $\Phi_t$
```

trackers in the previous time instant (Φ_{t-1}). First, the trackers positions in the new image (Im_t) are calculated (Algorithm 1, line 2 —Alg. 1:2—). This is done by combining two trackers: a DCF and a Kalman one (Section 3.2). Then, if the time elapsed between detections is over a certain threshold τ (Alg. 1:3), the detection of the objects of interest (Ψ_t) in the current image Im_t is performed through a ConvNet (Alg. 1:4). In practice, τ is set to the minimum time value that allows the system to achieve real-time performance. Detection is performed with a fully convolutional network called FPN [15], which uses feature maps information at different scales to locate from small to large objects, through a pyramidal architecture with lateral connections between them. The FPN provides high precision at a high computational cost, taking about 135 ms to perform a full detection in an HD image. Thus, clearly, a deep learning solution for detection does not suffice to comply with real-time requirements, which makes tracking necessary. After detection, trackers are updated (Alg. 1:5, Section 3.3). If no detection is performed at current time t, tracking prediction alone ($\overline{\Phi}_t$) determines the current trackers state (Φ_t, Alg. 1:7).

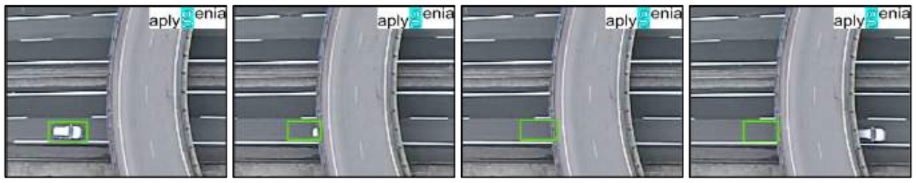
3.2. Tracking Prediction

Alg. 2 describes the trackers prediction. First, the system estimates the position of the trackers in the new frame (Im_t) using DCF tracking. Our tracker is based on the Discriminative Scale Space Tracker (DSST) [23], which is a correlation-filter-based tracker [20]. It uses HOG [22] and color as features for the correlation filter that models the tracked object.

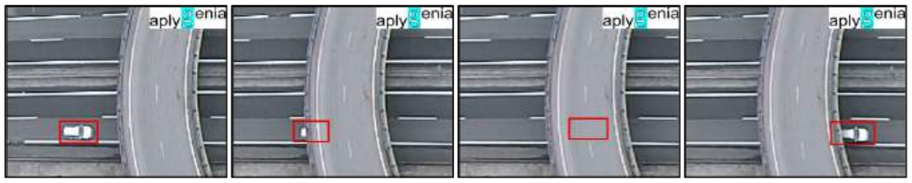
Algorithm 2: Tracking Prediction

```
   Require:
     (a) $Im_t$: Image frame at current time $t$
     (b) $\Phi_{t-1} = \{\varphi^1_{t-1}, \varphi^2_{t-1}, \dots, \varphi^n_{t-1}\}$
1  Function Tracking_Prediction($\Phi_{t-1}$,$Im_t$):
2      for $i = 1$ to $n$ do
3          $Y_t = \frac{\sum_{l=1}^{d} \overline{W^l_{t-1}} S^l_t}{X_{t-1}+\lambda}$
4          $\zeta_t = max(\mathscr{F}^{-1}\{Y_t\})$
5          $W^l_t = (1-\eta)W^l_{t-1} + \eta\overline{G}F^l_t, \quad l = 1,..,d$
6          $X_t = (1-\eta)X_{t-1} + \eta\sum_{k=1}^{d}\overline{F^k_t}F^k_t$
7          $\bar{\mu}_t = A_t\,\mu_{t-1}$
8          $\bar{\Sigma}_t = A_t\,\Sigma_{t-1}\,A_t^T + R_t$
9          $\kappa = \delta[1 - exp\{-(\zeta_t - c(\bar{\mu}))^T$
10         $(C_t\bar{\Sigma}_tC_t^T + Q_t)^{-1}\,(\zeta_t - c(\bar{\mu})\}]$
11         $Q_t = Q_t\,\kappa$
12         $K_t = \bar{\Sigma}_t\,C_t^T(C_t\,\bar{\Sigma}_t\,C_t^T + Q_t)^{-1}$
13         $\Sigma_t = (I - K_t\,C_t)\bar{\Sigma}_t$
14         $\overline{\varphi}^i_t \leftarrow < W_t, X_t, \bar{\mu}_t, \Sigma_t, Q_t >$
15     return $\overline{\Phi}_t$
```

In Alg. 2:3 the correlation scores Y_t are computed considering a sample S_t, extracted from Im_t and the previous filter information formed by W_{t-1} and X_{t-1}. λ is a weight parameter and d is the feature dimension of the correlation filter. The maximum value of Y_t —taking the inverse Discrete Fourier Transform (DFT)— will be the center of the object in the new image (ζ_t) as shown in Alg. 2:4. Then, the filter is updated at Alg. 2:5-6, where F is a target sample extracted from Im_t at position ζ_t, G is the desired correlation output and η is a learning rate parameter. In practice, we use two DCFs, one for estimating translation and another one for scale change. Tracking based just on DCF trackers has two limitations. First, we cannot handle occlusions (Figure 2). Second, it does not provide a robust tracking failure detection (i.e. knowing when the tracking fails) as the PSR (Peak to Sidelobe Ratio) value [20], which measures the spread degree of the convolution operation of the correlation filter, is not a reliable measure. As shown in Figure 3, the PSR takes different threshold values for different videos and scenarios, which makes difficult to identify when a tracker is lost.



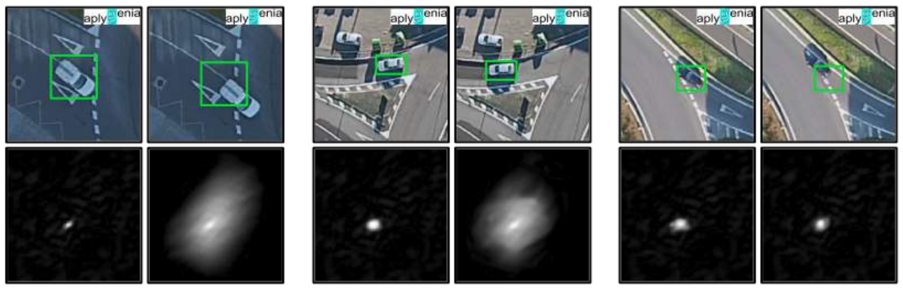
(a) DCF Tracker



(b) DCF+KF Tracker

Figure 2: (a) The low-level DCF tracker (in green) cannot recover the identity of the object once occluded as it only relies on appearance. (b) The combination of a DCF and a KF manages occlusions, as it also takes into account the object motion model. Images courtesy of Aplygenia S.L.

To provide a solution to both problems, we introduce a Kalman Filter (KF) tracker that, by modeling the movement of the object can handle occlusions and,



(a) (b) (c)

Figure 3: PSR values are poor predictors of tracking failures for the DCF tracker.(a) An ideal case in which PSR values predict correctly a tracking failure. (b) A vehicle is being tracked correctly but the PSR distribution indicates a tracking loss. (c) Tracking loss is not detected by PSR as the values do not change in both situations. Images courtesy of Aplygenia S.L.

in combination with the DCF tracker, can estimate the error in the tracking process. So, once the vehicle's new position is calculated by the DCF tracker (ζ_t), we estimate the position using the Kalman filter. We use a linear constant velocity model in the KF, so the state of each vehicle is modeled as:

$$\mu := [x, y, v_x, v_y] \tag{1}$$

Here x and y represent the position of the object, and v_x and v_y represent the linear velocity in both axes. We perform Kalman prediction in Alg. 2:7-8: $\bar{\mu_t}$ is the predicted state mean, A is the transition matrix, $\bar{\Sigma}$ is the predicted covariance of the system and R is the process noise covariance matrix. At this point we have two independent estimations of the new vehicle position: DCF (ζ_t) and Kalman ($\bar{\mu_t}$).

In order to measure the agreement degree of both estimators, we use the Mahalanobis distance, which measures the distance between a point and a distribution. We express this distance as the uncertainty that represents how unlikely both positions correspond to the same tracker (Alg. 2:10), where $c(\bar{\mu})$ is the expected position of the object estimated by the KF, C_t is the Jacobian matrix of c (measurement model), Q_t is the measurement noise covariance matrix, and δ is a normalization factor. With this uncertainty (κ), we update accordingly the covariance of the measure Q_t (Alg. 2:11): the higher the uncertainty in both positions corresponding to the same object, the higher the covariance. When one of the estimators fails to track an object (i.e. occlusion), the distance between the positions of the trackers increases and so does the uncertainty. This is an important feature because it permits to detect tracking failures, thus, maintaining a more reliable estimation of the object position during the time elapsed through a larger covariance matrix. Then, we perform a partial correction (Alg. 2:12-13) of the covariance Σ not only every time we integrate a measure performing a full correction (Alg. 3), but with every prediction as well.

At this point we integrate in every tracker $\overline{\varphi}_t^i$ the new calculated information, which is represented in a new set $\overline{\Phi}_t$ (Alg. 2:14-15).

3.3. Tracking Update

The tracking update process is shown in Alg. 3. First, the algorithm estimates the area in which the vehicle might be in the current frame. We call this area our search ROI. This search ROI is a rectangle centered at $\bar{\mu}_t$ and with a size proportional to Σ_t (Alg. 3:3, Figure 4).

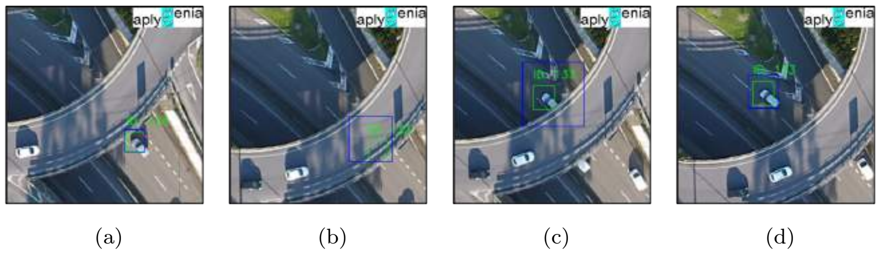


Figure 4: Creation of a search ROI for occlusion handling. (a) Both tracking methods agree on the object position. (b) As the DCF fails to track the occluded object, the distance between both estimations increases and so it does the search ROI. This process continues in (c) and finally in (d), when the detector finds the vehicle at the other side of the road and the tracker recovers. Images courtesy of Aplygenia S.L.

Next, data association assigns every detection to its corresponding tracker. The first step is to identify the possible detections that are candidates to be

Algorithm 3: Data Association and Tracking Update

```
Require:
    (a) $Im_t$: Image frame at current time $t$
    (b) $\overline{\Phi}_t = \{\overline{\varphi}_t^1, \overline{\varphi}_t^2, \ldots, \overline{\varphi}_t^n\}$
    (c) $\Psi_t = \{\psi_t^1, \psi_t^2, \ldots, \psi_t^m\}$
1  Function Tracking_Update($\overline{\Phi}_t$,$\Psi_t$,$Im_t$):
2      for $i$=1 to $n$ do
3          $SROI_t^i$ =Calc_SearchROI($\bar{\mu}_t$,$\Sigma_t$)
4          for $j$=1 to $m$ do
5              if $\frac{SROI_t^i \bigcap \psi_t^j}{SROI_t^i \bigcup \psi_t^j} > 0$ then
6                  $z_t = \psi_t^j.center$
7                  $\omega(i,j) = [(z_t - c(\bar{\mu}_t))^T$
8                  $(C_t \Sigma_t C_t^T + Q_t)^{-1} (z_t - c(\bar{\mu}_t))]$
9              else
10                 $\omega(i,j) = \infty$
11     $\{< \overline{\varphi}_t^\alpha, \psi_t^\beta >\}$ =Hungarian ($\omega$)
12     for every $\alpha, \beta$ in $\{< \overline{\varphi}_t^\alpha, \psi_t^\beta >\}$ do
13         DCF_Reset ($\psi_t^\beta$)
14         $K_t = \bar{\Sigma}_t \, C_t^T (C_t \, \bar{\Sigma}_t \, C_t^T + Q_t)^{-1}$
15         $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$
16         $\Sigma_t = (I - K_t \, C_t)\bar{\Sigma}_t$
17         $\varphi_t^i \leftarrow < W_t, X_t, \mu_t, \Sigma_t, Q_t >$
18     return $\Phi_t$
```

assigned to a particular tracker. In our case, a detection (ψ_t^j) that has an IoU (Intersection over Union) $>$ 0 [50] with a tracker's search ROI ($SROI_t^i$) is considered a candidate for the assignment (Alg. 3:5). Once a detection has already been established as a candidate, we calculate the cost of association

$\omega(i,j)$ as the Mahalanobis distance between the position predicted by the tracker ($c(\bar{\mu}_t)$) and the center of the measurement provided by the ConvNet (z_t) (Alg. 3:6-8). If there is no overlap (IoU) the association between the tracker $\bar{\varphi}_t^i$ and a detection ψ_t^j is not possible, and the cost of association is set to infinite (Alg. 3:10). With this information, a cost matrix is generated. Every element of the matrix represents the cost of associating a detection (ψ_t^j) with a tracker (φ_t^i). That results in an assignation problem that is solved in polynomial time by the Hungarian Method (Alg. 3:11). For every successful assignation ($< \varphi_t^\alpha, \psi_t^\beta >$) we reset the DCF tracker of φ_t^α at position ψ_t^β, and perform an update of its KF, obtaining the new mean (μ_t) and covariance (Σ_t) of the filter (Alg. 3:13-16). Finally, every tracker φ_t^i in Φ_t (Alg. 3:17-18) is constructed with W_t, X_t and Q_t from the previous prediction step, and μ_t and Σ_t from the current update step.

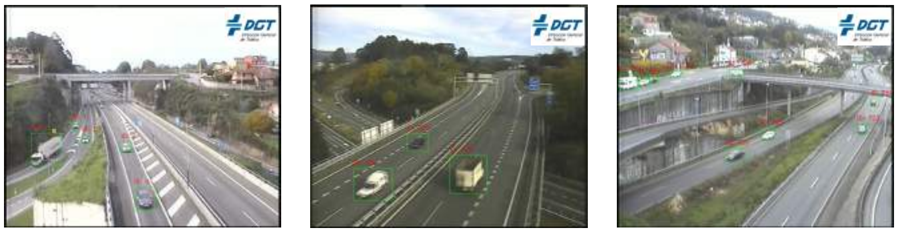
4. Results

In this section we present the results obtained by our tracking system. First, we show an ablation study of the system with a set of traffic videos provided by the Spanish Traffic Authority (DGT) in Section 4.1. In Section 4.2 we evaluate our system in the MOT15 public dataset showing how it performs with respect to the ideal case. In Section 4.3 we analyze the computational cost of the different parts of our system and its implementation for real-time performance. In Sections 4.4 and 4.5 we extend the proposed system for two real-life traffic applications: roundabout monitoring and anomaly detection in traffic roads.

4.1. Comparison between DCF and DCF+KF

In this section we compare the core of our tracking system —the DCF tracker plus the Kalman Filter— with a DCF baseline, in order to show the increase in robustness metrics by adding a motion predictor even in scenarios that do not present occlusions. In order to guarantee a fair comparison we will remove the following components from our complete system: (i) the ConvNet detector —we will use the ground truth detections; (ii) the tracking detection failure module; (iii) the data association module —association is based on IoU [51].

The dataset has three real-life videos provided by the *DGT* [1] with more than 1,000 vehicles with a frame rate of 15 fps and without total occlusions, for a fair comparison (Figure 5 and Table 1). We measure the performance with standard multiple object tracking metrics [51].



(a) (b) (c)

Figure 5: Example frames of our video dataset to evaluate tracking metrics. These videos are from traffic monitoring cameras. Images courtesy of DGT.

Table 1: Dataset with more than 1,000 different vehicles in more than 20 minutes of video to evaluate tracking metrics.

Video	**Frames**	**Vehicles**	**Time**
DGT_34001	4,900	221	0:05:26
DGT_39132	15,050	575	0:16:43
DGT_39151	2,345	244	0:02:36
Total	***22,295***	***1,040***	***0:24:46***

In every frame we have a set of hypotheses (our trackers) and a set of objects (labeled ground truth). We assign to every hypothesis its nearest object with IoU > 0. For every successfully assigned pair, we measure the IoU between the two bounding boxes to estimate the multiple object precision (MOTP). To estimate the multiple object accuracy (MOTA) or robustness we count every

[1]Dirección General de Tráfico (DGT) is the Spanish Traffic Authority [52].

hypothesis without an associated object as a *false positive*, every object that has no associated hypothesis as a *miss* and, finally, every assignation of identity (ID) that differs from the last frame's ID as a *mismatch*.

Also, in order to analyze the influence of the elapsed time between detector calls, we varied the frequency of the detections (obtained from the ground truth) from once every 5 frames to once every 30 frames. To identify tracking failure, we use the PSR value. As the PSR is not a reliable metric, we repeated the experiments for different PSR thresholds. Finally, when detections are available, all trackers are reinitialized.

Results for MOTP are shown in Figure 6. As we can see, DCF+KF achieves more precision than DCF independently of the PSR threshold, but the differences are negligible. This is an expected result as the objective of adding a KF is to increase robustness.

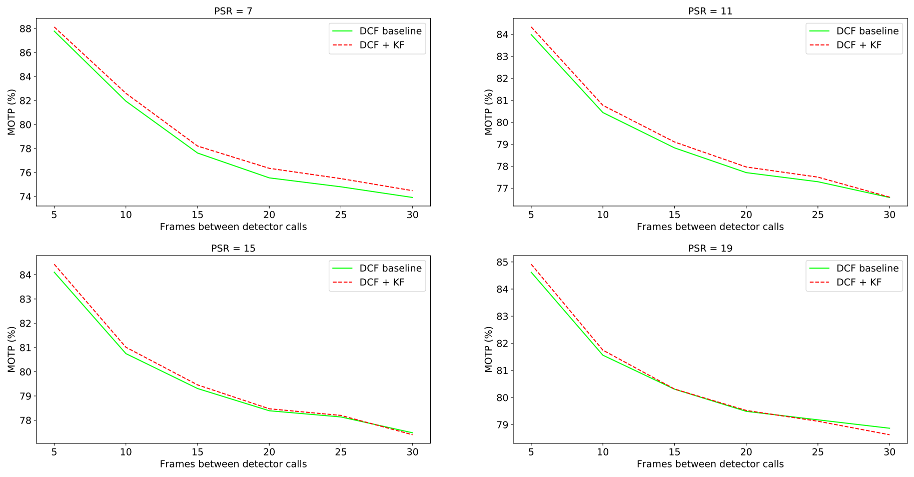


Figure 6: MOTP results for DCF and DCF+KF with several PSR thresholds and different number of frames between detector calls for the videos outlined in Table 1.

MOTA metrics are shown in Figure 7. DCF+KF shows more robustness than DCF for every combination of PSR value and number of frames between detector calls. The higher the number of frames between detector calls, the higher the improvement of DCF+KF over DCF —5% of improvement for 25

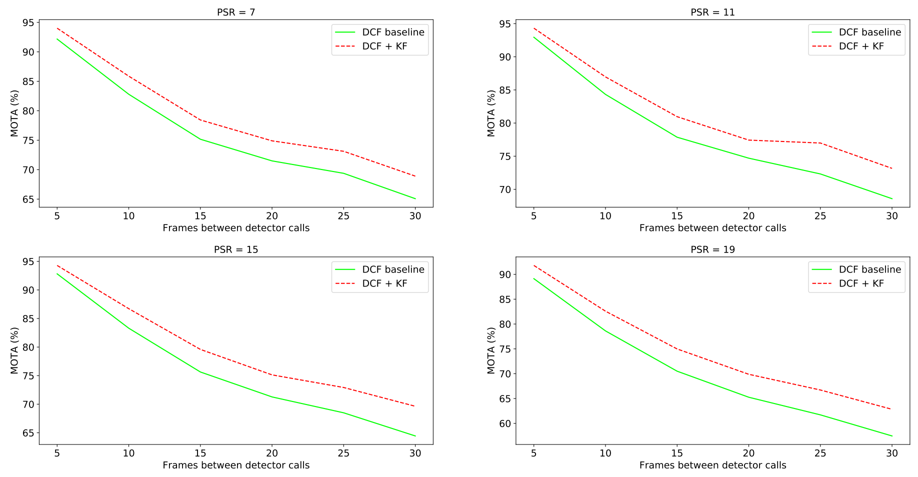


Figure 7: MOTA results for DCF and DCF+KF with several PSR thresholds and different number of frames between detector calls.

Table 2: Reinitializations (Reinit), Misses (M), false positives (FP) and mismatches (MM) in the videos of Table 1 for a PSR threshold of 13 (standard). The right columns show the improvement of DCF+KF over DCF.

Version	**Reinit.**	**M**	**FP**	**MM**	**M(%)**	**FP(%)**	**MM(%)**
DCF	5	168	1,091	84	17.2%	30.9%	402.9%
DCF+KF	5	143	834	17			
DCF	10	639	2,102	164	34.9%	24.0%	381.3%
DCF+KF	10	474	1,695	34			
DCF	15	1,285	2,540	205	40.7%	12.5%	387.2%
DCF+KF	15	913	2,258	42			
DCF	20	1,949	2,806	209	45.8%	10.0%	353.4%
DCF+KF	20	1,336	2,551	46			
DCF	25	2,704	2,788	201	42.9%	13.0%	277.5%
DCF+KF	25	1,892	2,468	53			
DCF	30	3,309	2,951	197	43.7%	3.7%	205.6%
DCF+KF	30	2,302	2,846	65			

and 30 frames. This is important for a real-time application because the deep learning detector can be executed only at some time instants.

This increase in robustness is detailed in Table 2, analyzing the misses, false positives and mismatches. The largest reduction of failures is in the number of *mismatches*. This is because the KF allows keeping the identities of the objects for longer periods, which results in an improvement between 200% and 400% compared with DCF. This improvement is of key importance for roundabout monitoring as each mismatch represents a failure and has a negative impact on the success rate in the final system. Also, for detecting anomalies on traffic roads, we need to maintain the identity of the vehicles as long as possible to ensure that the correct alarm is triggered. In conclusion, the results obtained support the addition of a Kalman filter to a DCF tracker as the core of our tracking system due to the increase in robustness while keeping precision.

4.2. Results in MOT benchmark

In this section we evaluate our tracking system in the MOT 15 benchmark [5], and compare it with two other approaches. The first one represents the ideal scenario in which the detector can be called at every frame, thus the low level tracking would not be necessary, and a simple data association based on IoU would be enough to match the ground truth in consecutive frames (tracking-by-detection approach). This approach is named *Ideal Case* and, as explained before, is not a valid solution taking into account current state-of-the-art detector inference times. The second approach is the real case in which the detector can only be called five times per second on average to guarantee real-time processing in the application —an average call of FPN takes 0.135 seconds, but a maximum runtime of 0.230 seconds is reached. Data association is made by IoU as in the previous case. This approach is named *D+IoU*. In order to guarantee a fair comparison, we modify the MOT benchmark as follows:

1. Ground truth is used as detections in order to evaluate the tracking module. For the *Ideal Case* all the detections are provided, while for *D+IoU* and our proposal only 5 detections per second are available.

2. In those cases in which a new object appears in the scene, it is added to the trackers list even though there is no detection at that particular frame. This applies to *D+IoU* and our approach.

Table 3 shows the results. We can observe how our approach outperforms the *D+IoU* achieving 21 % more MOTA using the same detections. Also, it gets a reduction in the number of mismatches of 58.58% which is crucial for a wide range of traffic applications.

Table 3: Table that shows the MOT15 results obtained comparing our system (*Our Approach*) with the *Ideal Case* and *D+IoU*.

Name	**Misses**	**False Positives**	**Mismatches**	**MOTA**	**MOTP**
Ideal Case	0	0	50	99,874	99,994
D+IoU	6160	6547	1067	65,205	84,126
Our Approach	2415	2307	442	86,955	85,836

4.3. Implementation details

The proposed system (Figure 1) runs on a server with an Intel Xeon E52623v4 2.60 GHz CPU, 128 GB RAM and an Nvidia GP102GL 24GB [Tesla P40] as GPU. Figure 8 shows how the system works for a 30 fps HD video. The system performs tracking in one of every 3 frames and detection in one of every 6 frames. Also, these two tasks are completely parallelized by threads. With these frequencies, the robustness of our system is not affected.

Table 4 shows the times of the two most computational expensive operations of our system: detection and tracking —computing times of other tasks are negligible. As explained before, for a 30 fps HD video, the tracking module processes 1 of 3 frames, which gives 0,1 seconds per frame. Using 15 threads for parallelization, the system is able to process more than 400 objects in the image while maintaining real-time performance, *i.e.* 30 fps —the maximum number of objects at any given time in our videos was 60. As mentioned before, detection is the slowest part of our system, taking an average 0.135 seconds in an HD

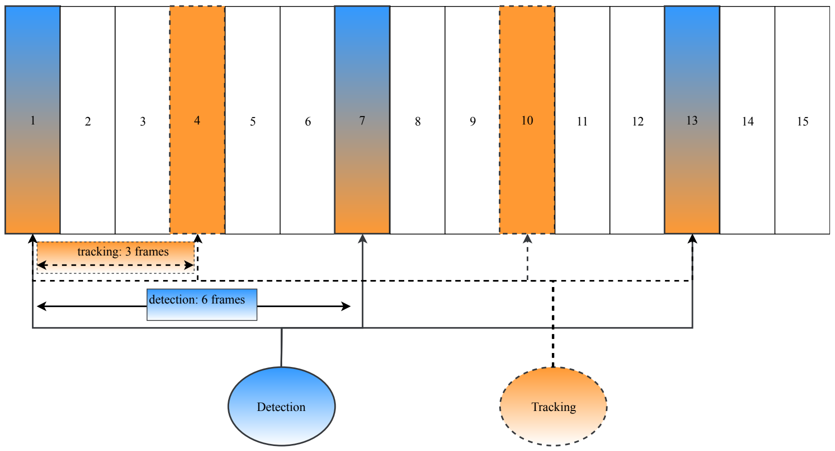


Figure 8: Frame processing of our system with an input video. We show the first 15 frames as an example. The system performs tracking one of every 3 frames (orange), and detection one of every 6 frames (blue).

image and 0.075 seconds in VGA resolution. These values are below the 0.2 threshold required by the system for the detection module.

4.4. Roundabout Monitoring

In this section, we analyze our complete system (Figure 1) for roundabout monitoring. The objective of the system is to identify the entry and the exit a vehicle takes, maintaining its identity while it remains in the roundabout. The final goal is to provide the I/O matrix R, in which every element $(R(i,j))$ represents the number of vehicles that joined the roundabout taking entry i and exit j. If a vehicle enters the roundabout and exits it with the same ID we count that as a tracking success. On the contrary, if the identity changes along the video, then we count that vehicle as a tracking failure.

For performing the metrics, we use a video dataset which consists of eight videos of roundabouts recorded from an Unnamed Aerial Vehicle (UAV) at 30 fps with HD resolution. The videos have different conditions that are challenging for traffic monitoring: shadows, total occlusions (two level roads), camera

Table 4: Computational times for the tracking and detection modules of the traffic monitoring system.

Tracking	
Frames processed by second	10 frames of 30
Total max. time with parallel computing	0.0121 sec (60 objects, 15 threads)
Max. number of objects in 0.1 sec	492 objects
Detection	
Frames processed by second	5 frames of 30
Average time per HD image	0.135 sec

movement, etc. Figure 9 shows a snapshot of some of these videos [2].

To evaluate the performance, for every video, the entries and exits of the roundabout were marked with a line. Then, the system processes the video completely and generates the result input/output matrix. This result is compared with the manually calculated ground truth to measure the success rate of the system. Table 5 shows the results obtained from processing the I/O matrix of eight videos with more than 1,000 vehicles in total. An average success rate of 93% is obtained. Results also show our system's ability to handle occlusions as two of the videos are scenarios with a high rate of total occlusions: in one of them the 50% of the vehicles are totally occluded nearly twice on average.

[2]A demonstration video can be downloaded from: `http://bit.ly/roundabout_sample_video`

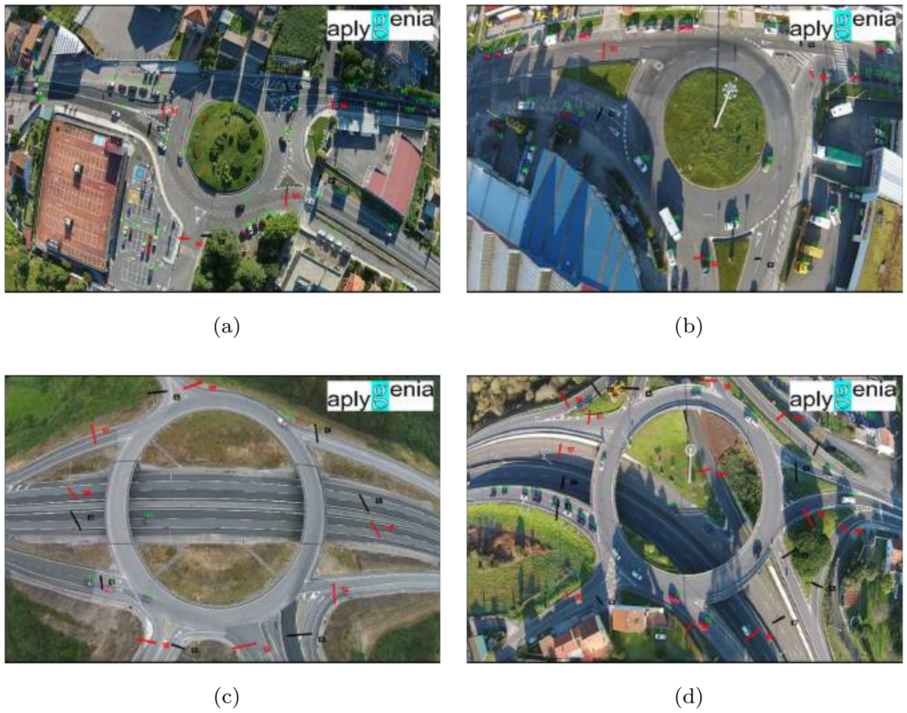


Figure 9: Example frames of some videos of the roundabout monitoring dataset. These videos are recorded from an UAV flying over a roundabout. Images courtesy of Aplygenia S.L.

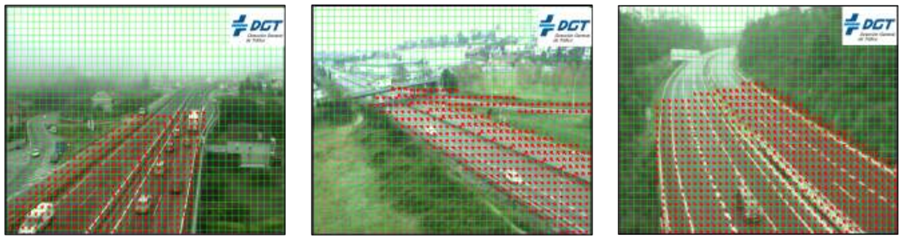
4.5. Anomaly Detection in Traffic Monitoring

Another application of the traffic monitoring system is anomaly detection. The objective is to identify possible anomalies on a road, like: sudden stops (sometimes produced by crashes), vehicles that circulate at an inappropriate velocity or vehicles that move in a direction different to the usual on that road.

First, the system has to generate the movement map of the road. We do that by dividing the image into grid cells. Each one of the cells has two moving averages: one for the angle and the other for the velocity. We update the movement map with the tracking information that our traffic monitoring system provides. Figure 10 shows the movement map calculated for three traffic videos. The alarms are fired based on a probabilistic Gaussian model of the velocity and direction in the cells of the map occupied by the vehicle. Thus, when the velocity

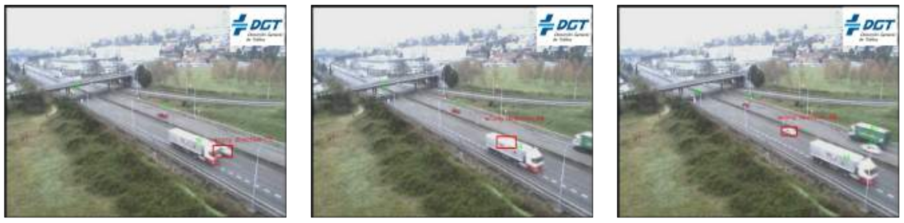
Table 5: Results in the video dataset for roundabout monitoring. The columns are: video, number of occlusions (#*occ*), number of vehicles occluded (#*vocc*), duration of video, total number of vehicles (#*vehicles*) and success rate obtained by our tracking system.

Video	**#occ**	**#vocc**	**Time (min:sec)**	**#vehicles**	**Success**
usc_vr_1	308	160	05:11	320	86,50%
usc_yt_1			01:43	13	100%
usc_yt_2			00:30	15	93%
usc_yt_3			00:45	14	100%
usc_pl_1			11:12	138	88%
usc_rb_1			11:48	230	95%
usc_sx_1			09:26	255	91%
usc_ou_1	22	11	02:49	52	96%
Total	***330***	***171***	***43:40***	***1,037***	***93,36%***



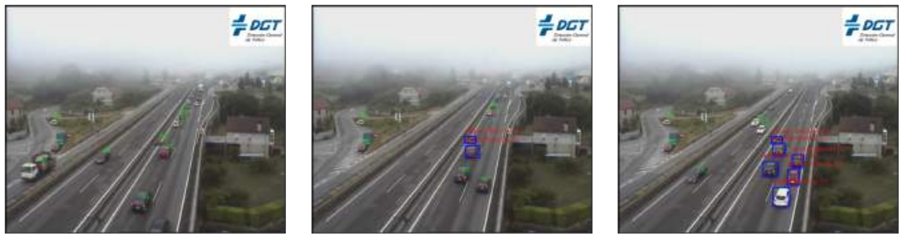
(a) (b) (c)

Figure 10: Examples of the movement map generated in traffic videos for accident detection. For each cell we show just the mean angle. Images courtesy of DGT.



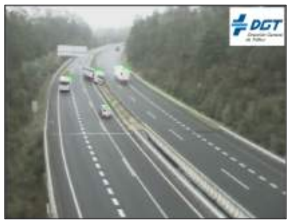
(a) (b) (c)

Figure 11: Example of an alarm for a vehicle moving in a wrong direction. In (a) the system detects an object moving with a direction contrary to the usual one in that lane, so an alarm is launched. In (b) the system keeps tracking the object despite being totally occluded. In (c) the system continues the tracking after the occlusion, keeping the same id and the alarm. Images courtesy of DGT.

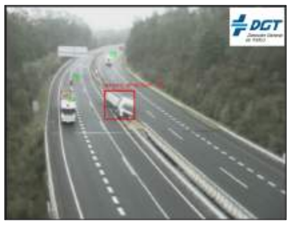


(a) (b) (c)

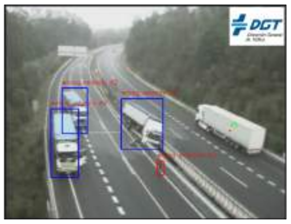
Figure 12: Example of an alarm for an accident. In (a) a motorcycle crashes with a car. In (b) the system detects both vehicles as static in a lane that has movement, so it triggers an alarm. Finally in (c) the system keeps triggering alarms for every car that stops in that area. Images courtesy of DGT.



(a)



(b)



(c)

Figure 13: Frames of a video that has both alarms. In (a) a truck starts to derail. In (b) the system detects an anomaly in the second lane when the truck invades it so it triggers the wrong direction alarm. In (c) when the crashed truck remains static the system triggers the velocity alarm and it does the same for every truck that moves with abnormal velocity (reduced speed). Images courtesy of DGT.

or direction of the vehicle differs from the standard ones, an alarm is triggered. The model also takes into account that the velocity and/or direction in the area can change, and an alarm might be switched off due to this.

To evaluate the system for anomaly detection, first we generated the traffic flow for every processed video. For the experiments, we used 4×4 cell grids with a moving average of 20 periods. We compare the tracking results and alarms with the manually annotated ground truth frame by frame. Figures 11, 12 and 13 show detected alarms in three different traffic scenarios. Figure 14 shows the results obtained with this application in three videos provided by the Spanish Traffic Authority (DGT). These videos present accidents, sudden stops and vehicles circulating in the wrong direction of a lane. Our system is able to track successfully (without IDs losses) more than 90% of the objects and launch the corresponding alarm in more than 90% of the cases.

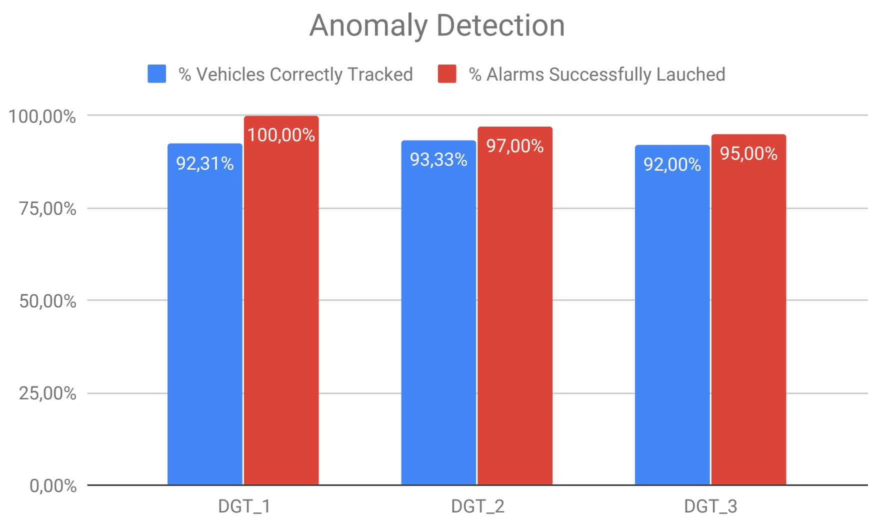


Figure 14: Results obtained by our anomaly detection system. For calculating the percentage of vehicles successfully tracked (blue), every tracker miss, false positive or mismatch is considered a failure. For the percentage of alarms successfully launched (red), every alarm not triggered in the corresponding frame is considered a failure. Videos courtesy of DGT.

5. Conclusions

We have presented a traffic monitoring system that combines a ConvNet detection, DCF and Kalman trackers, and the Hungarian data association. The system is able to track hundreds of objects in real-time while being robust to occlusions. The combination of the DCF and Kalman filters allows to: (i) improve both the precision and, especially, the robustness, obtaining an improvement of up to 402,9% in the number of tracking mismatches; (ii) estimate the error of each tracker, thus increasing the robustness and reliability of the system. We have applied the traffic monitoring system to two different real-life applications. First, in roundabout monitoring, our system achieves a 93% success rate for the I/O matrix, even in cases with high occlusion rates, shadows and movement of the UAV onboard camera. Second, for anomaly detection in traffic monitoring, the system identifies accidents on roads, sudden stops, abnormal speeds and vehicles that move in the wrong direction, triggering an alarm in any of these scenarios. As future work, we plan to extend our traffic monitoring system to perform multi-camera vehicle tracking and multi-camera vehicle re-identification. Also, we find interesting to estimate the velocity of the vehicles being tracked to improve anomaly detection with speed limits information.

Acknowledgments

This research was partially funded by the Spanish Ministry of Science and Innovation under grants TIN2017-84796-C2-1-R and RTI2018-097088-B-C32, and the Galician Ministry of Education, Culture and Universities under grant ED431G/08. Mauro Fernández is supported by the Spanish Ministry of Economy and Competitiveness under grant BES-2015-071889. These grants are co-funded by the European Regional Development Fund (ERDF/FEDER program). We thank the company Aplygenia S.L. and Dirección General de Tráfico (DGT) for their collaboration.

References

[1] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, P. H. S. Torr, Fully-convolutional siamese networks for object tracking, in: European Conference on Computer Vision Workshops, 2016.

[2] M. Danelljan, A. Robinson, F. S. Khan, M. Felsberg, Beyond correlation filters: Learning continuous convolution operators for visual tracking, in: European Conference on Computer Vision (ECCV), 2016.

[3] M. Danelljan, G. Bhat, F. S. Khan, M. Felsberg, ECO: Efficient convolution operators for tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[4] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[5] MOTChallenge the multiple object tracking benchmark, `https://motchallenge.net/`, accessed: 2018-12-18.

[6] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, J. Sun, Light-head R-CNN: in defense of two-stage object detector, arXiv preprint arXiv:1711.07264.

[7] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[8] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556.

[9] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[11] J. R. Uijlings, K. E. Van De Sande, T. Gevers, A. W. Smeulders, Selective search for object recognition, International Journal of Computer Vision (IJCV), 2013.

[12] C. L. Zitnick, P. Dollár, Edge boxes: Locating object proposals from edges, in: European Conference on Computer Vision (ECCV), 2014.

[13] R. Girshick, Fast R-CNN, in: IEEE International Conference on Computer Vision (ICCV), 2015.

[14] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems (NIPS), 2015.

[15] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision (ECCV), 2014.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: European Conference on Computer Vision (ECCV), 2016.

[18] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: IEEE International Conference on Computer Vision (ICCV), 2017.

[20] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[21] M. Danelljan, F. Shahbaz Khan, M. Felsberg, J. Van de Weijer, Adaptive color attributes for real-time visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[22] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

[23] M. Danelljan, G. Häger, F. S. Khan, M. Felsberg, Discriminative scale space tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (8) (2017) 1561–1575.

[24] J. F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (3) (2015) 583–596.

[25] C. Ma, X. Yang, C. Zhang, M.-H. Yang, Long-term correlation tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[26] M. Danelljan, G. Hager, F. Shahbaz Khan, M. Felsberg, Adaptive decontamination of the training set: A unified formulation for discriminative visual tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[27] H. Kiani Galoogahi, T. Sim, S. Lucey, Correlation filters with limited boundaries, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[28] M. Kristan, A. Leonardis, J. Matas, M. Felsberg, R. Pflugfelder, L. ehovin, T. Vojir, G. Bhat, A. Lukei, A. Eldesokey, G. Fernandez Dominguez,

A. Garcia-Martin, . Iglesias-Arias, A. Alatan, A. Gonzalez-Garcia, A. Petrosino, A. Memarmoghadam, A. Vedaldi, A. Muhi, The Sixth Visual Object Tracking VOT2018 Challenge Results, 2019, pp. 3–53.

[29] T. Xu, Z.-H. Feng, X.-J. Wu, J. Kittler, Learning adaptive discriminative correlation filters via temporal consistency preserving spatial feature selection for robust visual tracking, arXiv preprint arXiv:1807.11348.

[30] B. Li, J. Yan, W. Wu, Z. Zhu, X. Hu, High performance visual tracking with siamese region proposal network, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[31] Z. Zhu, Q. Wang, L. Bo, W. Wu, J. Yan, W. Hu, Distractor-aware siamese networks for visual object tracking, in: European Conference on Computer Vision (ECCV), 2018.

[32] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, R. Pflugfelder, The visual object tracking vot2017 challenge results, in: IEEE International Conference on Computer Vision Workshops, 2017.

[33] C. Kim, F. Li, A. Ciptadi, J. M. Rehg, Multiple hypothesis tracking revisited, in: IEEE International Conference on Computer Vision (ICCV), 2015.

[34] J. Chen, H. Sheng, Y. Zhang, Z. Xiong, Enhancing detection model for multiple hypothesis tracking, in: IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2017.

[35] D. Reid, et al., An algorithm for tracking multiple targets, IEEE Transactions on Automatic Control 24 (6) (1979) 843–854.

[36] W. Choi, Near-online multi-target tracking with aggregated local flow descriptor, in: IEEE International Conference on Computer Vision (ICCV), 2015.

[37] A. Sadeghian, A. Alahi, S. Savarese, Tracking the untrackable: Learning to track multiple cues with long-term dependencies, in: IEEE International Conference on Computer Vision (ICCV), 2017.

[38] R. Duan, C. Fu, E. Kayacan, Tracking–recommendation–detection: A novel online target modeling for visual tracking, Engineering Applications of Artificial Intelligence 64 (2017) 128–139.

[39] S. Tang, M. Andriluka, B. Andres, B. Schiele, Multiple people tracking by lifted multicut and person reidentification, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[40] R. Henschel, L. Leal-Taixé, D. Cremers, B. Rosenhahn, Fusion of head and full-body detectors for multi-object tracking, in: Computer Vision and Pattern Recognition Workshops, 2018.

[41] A. Bewley, Z. Ge, L. Ott, F. Ramos, B. Upcroft, Simple online and realtime tracking, in: IEEE International Conference on Image Processing (ICIP), 2016.

[42] E. Bochinski, V. Eiselein, T. Sikora, High-speed tracking-by-detection without using image information, in: IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS), 2017.

[43] C. Feichtenhofer, A. Pinz, A. Zisserman, Detect to track and track to detect, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[44] S. R. E. Datondji, Y. Dupuis, P. Subirats, P. Vasseur, A survey of vision-based traffic monitoring of road intersections, IEEE Transactions on Intelligent Transportation Systems 17 (10) (2016) 2681–2698.

[45] J. I. Engel, J. Martín, R. Barco, A low-complexity vision-based system for real-time traffic monitoring, IEEE Transactions on Intelligent Transportation Systems 18 (5) (2017) 1279–1288.

[46] H. Dinh, H. Tang, Development of a tracking-based system for automated traffic data collection for roundabouts, Journal of Modern Transportation 25 (1) (2017) 12–23.

[47] Z. Tang, G. Wang, H. Xiao, A. Zheng, J.-N. Hwang, Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018.

[48] H. W. Kuhn, The hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1-2) (1955) 83–97.

[49] B. Wu, R. Nevatia, Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors, International Journal of Computer Vision 75 (2) (2007) 247–266.

[50] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[51] K. Bernardin, R. Stiefelhagen, Evaluating multiple object tracking performance: the clear mot metrics, Journal on Image and Video Processing (2008) 1.

[52] DGT: Dirección General de Tráfico, `http://www.dgt.es/es/`, accessed: 2018-12-18.