# A Real-Time Processing Stand-Alone Multiple Object Visual Tracking System*

Mauro Fernández-Sanjurjo, Manuel Mucientes, and Víctor M. Brea

Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS)
Universidade de Santiago de Compostela, Santiago de Compostela, Spain
mauro.fernandez@usc.es, manuel.mucientes@usc.es, victor.brea@usc.es

**Abstract.** Detection and tracking of multiple objects in real applications requires real-time performance, the management of tens of simultaneous objects, and handling frequent partial and total occlusions. Moreover, due to the software and hardware requirements of the different algorithms, this kind of systems require a distributed architecture to run in real-time. In this paper, we propose a vision based tracking system with three components: detection, tracking and data association. Tracking is based on a Discriminative Correlation Filter combined with a Kalman filter for occlusions handling. Also, our data association uses deep features to improve robustness. The complete system runs in real-time with tens of simultaneous objects, taking into account the runtimes of the Convolutional Neural Network detector, the tracking and the data association.

**Keywords:** Multiple object tracking · Convolutional Neural Network · Data association.

## 1 Introduction

Real-life computer vision applications like traffic monitoring, autonomous vehicles, or surveillance in general usually require to detect and track tens of objects under the constraint of real-time processing in high resolution video like full HD and 4K formats without losing accuracy. All the above leads to the challenge of integrating seamlessly many heterogeneous solutions and hardware platforms into a stand-alone multiple object visual tracker. This does not mean a mere plug and play connection between trackers running on either CPU or GPU on the one hand, and top object detectors based on Convolutional Neural Networks (ConvNets) running on high end desktop GPUs on the other hand, but their

* This research was partially funded by the Spanish Ministry of Economy and Competitiveness under grants TIN2017-84796-C2-1-R and RTI2018-097088-B-C32 (MICINN/FEDER), 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 Dirección General de Tráfico (DGT) for their collaboration.

adaptation into a general architecture or framework combined with the insertion of decision-making processes along the data path to maximize performance metrics. Besides, motion prediction through, for instance, Bayesian filtering —Kalman filters, Particle filters, etc—, could also be included to deal with total occlusions.

In the last years, top trackers from the Visual Object Tracking (VOT) challenge [15] are based on two approaches: Discriminative Correlation Filters (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 [13,14,12]. On the other hand, deep-learning based trackers use ConvNets. SiamFC [10] 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 [16] they propose SiamRPN, 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 [20] improves SiamRPN, focusing the training on semantic distractors, and adding a search region strategy for long-term tracking. All these trackers cannot cope with occlusions, nor by themselves provide a framework to deal with multiple objects.

In contrast to VOT, the Multiple Object Tracking (MOT) contest [5] focuses on data association, as they assume that detections are available in all time instants without computational cost and, therefore, perform tracking by detection. In [18] they solve the data associations by proposing extensions to the classical Multiple Hypothesis Tracking approach. In [11] they combine Kalman filtering with the Hungarian algorithm for data association, and in [19], they improve the proposal using deep features to make data association more robust to occlusions.

In summary, we find the following limitations in the proposed approaches for real-life situations: (i) high performance object detectors based on deep learning are too slow to perform tracking-by-detection, and so does with deep learning trackers when dealing with several objects in the scene; (ii) the number of simultaneous real-time tracked objects is low; and (iii) the time response constraint dictated by the application is severely compromised by the communication among processes on CPU and GPU platforms and/or different libraries or frameworks like Caffe2 [1] , Pytorch [7] or OpenCV [6].

This paper aims at all the challenges outlined above. The main contributions of the paper are:

- A distributed architecture for a vision based tracking system which permits combining components of different technologies in a modular way for the correct exploitation of the available hardware resources, i.e. CPU and GPU.
- A complete system for visual tracking that can process tens of objects in real-time. It combines a Discriminative Correlation Filter (DCF) low-level tracker and a Kalman filter for the visual and motion information respectively. Difficult data association is performed using deep features correlation

while simple associations are solved based on overlap. The system shows a significant improvement in MOTP and MOTA metrics compared to the baseline version despite performing detection only once every second.

## 2 Stand-Alone Multiple Object Visual Tracking Approach

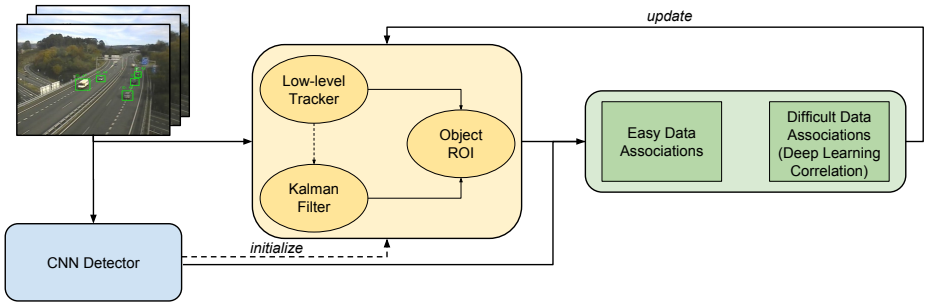


Fig. 1: Stand-alone multiple object visual tracking architecture. The blue box is the detection component. The orange block is the tracking. The green box is the data association module.

Fig. 1 shows the architecture of our system. It comprises three main components or visual tasks, namely, detection, tracking and data association. The proposed architecture serves two purposes. On the one hand, it allows the integration of the different parts of the system regardless of their underlying technologies. On the other hand, it is designed to provide a maximum use of the available hardware resources, that is, CPU and GPU. Each of the three modules of the architecture consists of a Docker [3] container with the specific dependencies necessary for their execution. In order to allow a modular integration of the different parts, Robot Operating System (ROS)[8] has been used as the framework for communication between processes for its flexibility, as well as for its support for computer vision tasks.

Algorithm 1 presents an overview of the steps of our approach. At every time instant $t$, the system processes two inputs: the current video frame ($Im_t$) and the set of trackers in the last time frame $\Phi_{t-1}$. First, the system calculates the new trackers position using two independent estimators: a discriminative correlation filter (DCF) tracker and a Kalman filter (Algorithm 1, lines 3-4 — Alg. 1:3-4—). With the bounding boxes proposed by both methods, we estimate the region of interest (ROI) in which the object might be located (Alg. 1:5). The larger the difference between the two trackers, the larger the ROI. Occlusions can be determined in cases where both predictors propose very different bounding

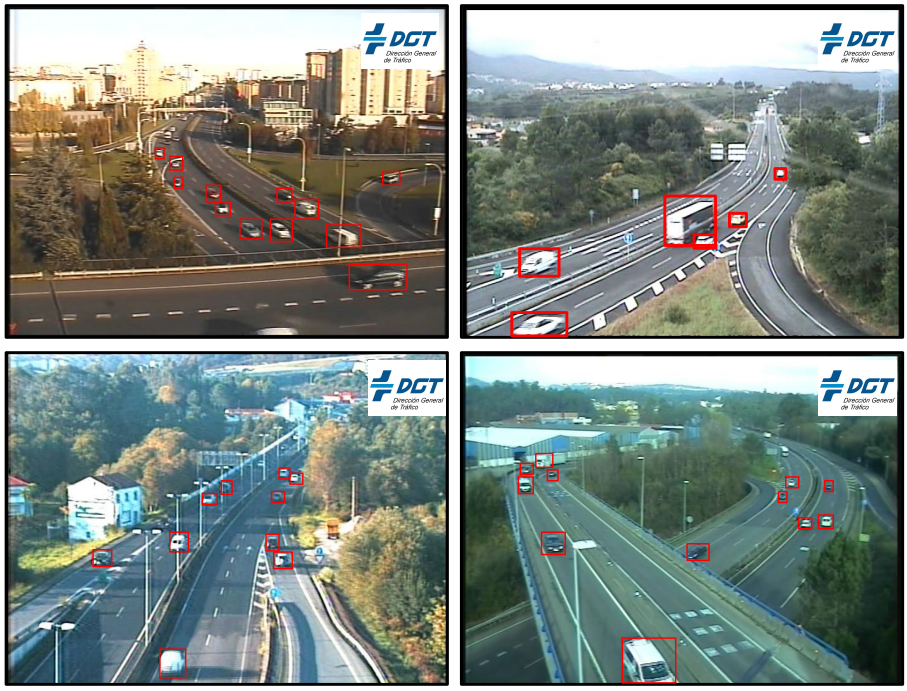


Fig. 2: Four images from the videos processed by our stand-alone multiple object visual tracking system. Videos are provided by the Spanish Traffic Authority (DGT) [2].

**Algorithm 1:** Our Stand-Alone Multi-Object Tracking System

**Require:**
(a) $Im_t$: Video frame at current time $t$
(b) $\Phi_{t-1} = \{\varphi^1_{t-1}, \varphi^2_{t-1}, \ldots, \varphi^n_{t-1}\}$

**Function** `Track`($Im_t$, $\Phi_{t-1}$)**:**
  **for** *i=1 to n* **do**
    $dcf\_ROI^i_t$ =`DCF_Track` $(\varphi^i_{t-1})$
    $kalman\_ROI^i_t$ = `Kalman_Predict` $(\varphi^i_{t-1})$
    $\overline{\varphi}^i_t \leftarrow <ROI^i_t>$=`Estimate_ROI` $(dcf\_ROI^i_t, kalman\_ROI^i_t)$
  **if** *time_elapsed* $> \tau$ **then**
    $\Psi_t \leftarrow \{\psi^1_t, \psi^2_t, \ldots, \psi^m_t\}$ =`ConvNet_Detect()`
    **for** *i=1 to n* **do**
      **for** *j=1 to m* **do**
        $IOU^{i,j}_t = \frac{\overline{\varphi}^i_t \bigcap \psi^j_t}{\overline{\varphi}^i_t \bigcup \psi^j_t}$
    $\{<\overline{\varphi}^\alpha_t, \psi^\beta_t>\}$ =`Get_Easy_Associations` $(IOU_t)$
    **for** *every* $\alpha, \beta$ *in* $\{<\overline{\varphi}^\alpha_t, \psi^\beta_t>\}$ **do**
      `update_tracker` $(\overline{\varphi}^\alpha_t, \psi^\beta_t)$
      `remove` $(\overline{\varphi}^\alpha_t, \overline{\Phi}_t)$
      `remove` $(\psi^\beta_t, \Psi_t)$
    **for** *i=1 to size*$(\overline{\Phi}_t)$ **do**
      **for** *j=1 to size*$(\Psi_t)$ **do**
        $\rho^{i,j}_t, \overline{\varphi}^i_t$ =`Deep_Learning_Correlation` $(\varphi^i_{t-1}, \psi^j_t)$
        $newIOU^{i,j}_t = \frac{\overline{\varphi}^i_t \bigcap \psi^j_t}{\overline{\varphi}^i_t \bigcup \psi^j_t}$
        $\omega(i,j) = 1 - (\rho^{i,j}_t \cdot newIOU^{i,j}_t)$
    $\{<\overline{\varphi}^\alpha_t, \psi^\beta_t>\}$ =`Hungarian` $(\omega)$
    **for** *every* $\alpha, \beta$ *in* $\{<\overline{\varphi}^\alpha_t, \psi^\beta_t>\}$ **do**
      `update_tracker` $(\overline{\varphi}^\alpha_t, \psi^\beta_t)$
      `remove` $(\overline{\varphi}^\alpha_t, \overline{\Phi}_t)$
      `remove` $(\psi^\beta_t, \Psi_t)$
    **for** *i=1 to size*$(\overline{\Phi}_t)$ **do**
      `delete_tracker` $(\overline{\varphi}^i_t)$
    **for** *j=1 to size*$(\Psi_t)$ **do**
      `new_tracker` $(\psi^j_t)$
  $\Phi_t = \overline{\Phi}_t$
  **return** $\Phi_t$

boxes, since the bounding boxes provided by DCF will remain static, while those from the Kalman filter will follow the previous movement pattern of the object.

Our system is robust enough as not to need detections in every frame. If the time elapsed since the previous detection is greater than or equal to $\tau$, detection is performed using a convolutional neural network (Alg. 1:6-7), which returns a set of detections $\Psi_t$. The aim of the detection component is twofold. First, it initializes every tracker or object of interest in the scene. Second, it refines the location and size of the bounding boxes of the trackers along their trajectories inside the data association component (see Fig. 1), improving tracking performance metrics. Thus, the frequency of detection calls sets a trade-off between tracking performance metrics and the number of objects that can be tracked at a given frame rate.

The data association block aims to assign each detection to its corresponding tracker and to identify objects that enter or leave the scene. In so doing, we build up the cost matrix $IOU_t$ (see Alg. 1:8-10), where every entry is the Intersection Over Union ($IOU$) between a tracker $\overline{\varphi}_t^i$ and a detection $\psi_t^j$. Our approach makes assignments or data association in two steps. First, the system solves easy associations between trackers and detections (see Fig. 1). In so doing, the maximum $IOU$ value is found ($maxIOU_t$). Then we iterate along the row and column of the cell with $maxIOU_t$ in order to find if there is any other $IOU_t^{i,j}$ complying with $IOU_t^{i,j} \geq \gamma \cdot maxIOU_t$. If not, we perform the assignment (Alg. 1:11). If so, this means that the $IOU$ metric is not sufficiently discriminative, yielding a difficult association (see Fig. 1). Finally, easy associations between trackers and detections are updated through Alg. 1:12-15.

Second, difficult associations between trackers and detections are solved with a deep learning based tracker. This solution features high robustness, but with a much higher computational cost than that based on $IOU$. The deep learning tracker calculates the bounding boxes in the current frame and their score (correlation) from the last frame's trackers position (Alg. 1:16-18). With this information the cost of association for the tracker $\overline{\varphi}_t^i$ with the detection $\psi_t^j$ is established as $\omega(i,j)$, using the $IOU$ between $\overline{\varphi}_t^i$ and $\psi_t^j$, namely, $newIOU_t^{i,j}$, along with their correlation score $\rho_t^{i,j}$ (Alg. 1:19-20). Once the cost matrix is generated it is solved by the Hungarian method (Alg. 1:21). For every successful assignation ($< \varphi_t^\alpha, \psi_t^\beta >$), tracker $\varphi_t^\alpha$ is updated with detection $\psi_t^\beta$ (Alg. 1:23). Finally, trackers not updated in the data association phase are candidates for being deleted, and detections not assigned are initialized as new trackers (Alg. 1:26-29).

## 3 Implementation details

Our experimental configuration consists of a server with an Intel Xeon E52623v4 2.60 GHz CPU, 128 GB RAM and an NVidia GP102GL 24GB (Tesla P40) as GPU. Nevertheless, our architecture allows for a complete distributed set up in which every module resides in a different machine.

In this current version of the architecture, we have included the ConvNet FPN with ResNeXt101 as backbone for object detection, as FPN is the basis for the three top entries in the last edition of the COCO detection challenge

[17]. A Caffe2 implementation of FPN with ResNeXt101 takes 0.39 s in full HD video, so that tracking by detection through the overlap of successive detections throughout the video is discarded.

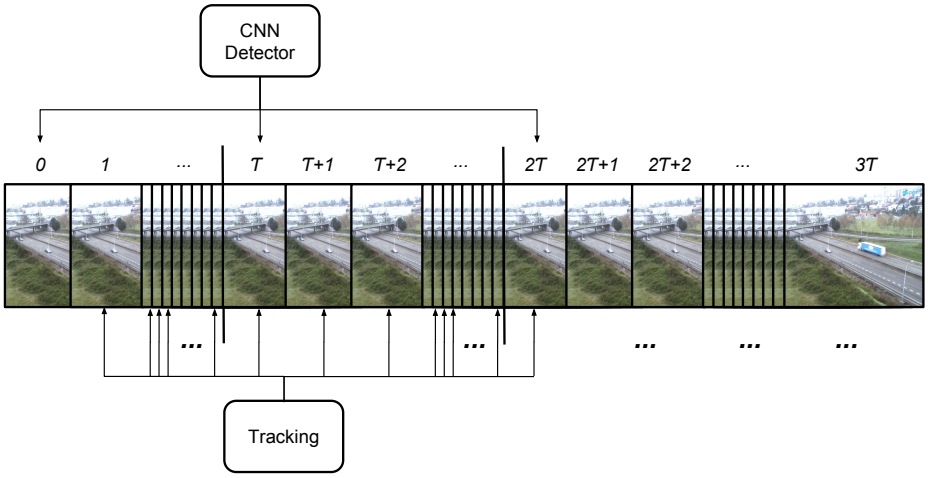


Fig. 3: Frame processing for our stand-alone multiple object visual tracker.

As DCF tracker we opt for KCF (Kernelized Correlation Filters) [14] as low-level tracker for its speed and for its implementation on a CPU, which can be parallelized to use all the available threads, thus increasing the number of tracked objects. As deep learning tracker for difficult associations between trackers and detections we build on DaSiamRPN, which offers very good results in long term tracking [15] with a speed of 200 fps [4]. However, its computation time does not scale well for several objects. Indeed, as far as we know, there is not any multiple object tracker in real-time with a deep learning solution. In order to cut processing time and cope with tens of objects at video frame rate, we have modified DaSiamRPN to initialize the network only the first time an object is identified. This is possible storing the feature vector and the states of all previously tracked objects. Still, we set an upper limit for the number of difficult data associations between trackers and detections solved with DaSiamRPN. If necessary, unsolved associations are handled during the next detection call. Finally, computation time is negligible for easy data associations.

Figure 3 illustrates how the different components of our architecture shown in Fig. 1 operate and interact with one another. Real-time processing of 30 fps in full HD video is met by cutting down the detector calls to only once per second, as our implementation of the FPN detector lasts 0.39 s in full HD video. Also, detections are being run or in parallel with the tracking. Detections during the first frame initialize the tracking module (see Fig. 1). The detection step produces a latency $\tau$ with respect to the original video as tracking in the first

frame needs to wait for the detection to finish, unlike in the rest of frames with detection, in which no waiting will be required. Finally, it should be noted here that the tracking is run on CPU, while the detection and difficult associations between trackers and detections do it on GPU. Our stand-alone multiple object tracker succeeds in dealing with this situation.

Table 1 shows a time breakdown of our stand-alone approach for 30 fps full HD video. Detection times are not included since they are run in parallel while tracking the objects in the scene. Our system is able to track 60 objects simultaneously at 30 fps HD video. In so doing, 90% of the time frame of one second in a video has been reserved for tracking, while difficult data associations have been assigned the remaining 10%. The data association is only performed once the next detection arrives, in our case, once per second. The times of the rest of the parts of the system are disregarded as they are not sufficiently relevant for the calculations. The time it takes our system to track an object during a second of video (DCF + Kalman) is 0.21 seconds. If we leave 90% of the time for tracking we would have 0.9 seconds for the operation and with a single thread we could process 4 trackers at 30 fps full HD video. When having 15 threads for this task (as one is reserved for the detection module) on a single CPU, the number rises to 60. Once the next detection is reached, it is necessary to perform data association. As mentioned above, we set an upper limit of 10% for difficult associations, which is a realistic average. In this case, it would take 0.005 seconds per object, which amounts to 0.03 seconds for 6 objects, since, as previously mentioned, DaSiamRPN is not a scalable solution. This would be within the 0.1 seconds of available time.

We have also developed a fast version of our stand-alone tracking system that for each three consecutive frames, makes an execution of the tracking system in one frame and keeps the trackers unchanged in the next two. This fast tracking system has a good performance for real-life applications, and is able to track up to 180 simultaneous objects.

Table 1: Time breakdown of our stand-alone multiple object tracker at 30 fps full HD video.

| **Tracking Times** | |
|---|---|
| Max. number of objects tracked in real-time | 60 objects |
| Time unit | 30 frames in 1 second |
| Available time for tracking assuming 90% | 90% = 0.9 seconds |
| Available time for difficult data association assuming 10% | 10%= 0.1 seconds |
| Time to track 1 object during 1 second | 0.21 seconds |
| Max. number of trackers in 0.9 seconds | 0.9/0.21 = 4 trackers |
| Using 15 threads | $4 \times 15 = 60$ objects |
| Assuming 10% of complex data association | 10% of 60 = 6 objects |
| Time of DaSiamRPN | 0.005 s per tracker |
| Average time in 6 objects | $0.005 \times 6 = 0.03$ s $< 0.1$ s |

# 4 Results

Table 2: Approach, video, number of objects in the video (n_o) Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA) for the three compared approaches.

| **approach** | **video** | **no** | **MOTP** | **MOTA** |
|---|---|---|---|---|
| 1. Ideal_case | | | 100,00% | 100,00% |
| 2. Real_case | video 1 | 589 | 54,73% | 56,22% |
| 3. Ours | | | 80,23% | 97,87% |
| 1. Ideal_case | | | 100,00% | 100,00% |
| 2. Real_case | video 2 | 218 | 58,27% | 84,74% |
| 3. Ours | | | 76,47% | 92,48% |
| 1. Ideal_case | | | 100,00% | 100,00% |
| 2. Real_case | video 3 | 244 | 66,56% | 67,38% |
| 3. Ours | | | 78,48% | 92,31% |

Table 3: Approach, video, misses (n_m), false positives (n_fp), mismatches (n_mm), percentage of increase in MOTP (%_p), percentage of increase in MOTA (%_a), percentage of decrease in misses (%_m), percentage of decrease in false positives (%_fp) and percentage of decrease in mismatches (%_mm) obtained with our system with respect to the real case in the videos of Table 2.

| **approach** | **video** | **n_m** | **n_fp** | **n_mm** | **%_p** | **%_a** | **%_m** | **%_fp** | **%_mm** |
|---|---|---|---|---|---|---|---|---|---|
| 1. Ideal_case | | 0 | 0 | 0 | | | | | |
| 2. Real_case | video 1 | 506 | 624 | 106 | | | | | |
| 3. Ours | | 38 | 21 | 1 | 46,59% | 74,10% | 92,49% | 96,63% | 99,06% |
| 1. Ideal_case | | 0 | 0 | 0 | | | | | |
| 2. Real_case | video 2 | 994 | 1,579 | 139 | | | | | |
| 3. Ours | | 391 | 903 | 42 | 31,23% | 9,14% | 60,66% | 42,81% | 69,78% |
| 1. Ideal_case | | 0 | 0 | 0 | | | | | |
| 2. Real_case | video 3 | 2,440 | 2,979 | 730 | | | | | |
| 3. Ours | | 390 | 1,018 | 42 | 17,91% | 36,99% | 84,02% | 65,83% | 94,25% |

We have tested our system for a traffic monitoring application. Fig. 2 displays some images of the videos processed in this section. In order to assess the performance of our stand-alone multiple object visual tracker we construct our own dataset using three videos with more than 1,000 objects. As performance metrics we have used Multiple Object Tracking Precision (MOTP) and Multiple Object Tracking Accuracy (MOTA)[9]. Our system has been compared with two other approaches. The first one represents the ideal scenario in which the detector could be called in every frame of a video maintaining real-time performance,

thus the low level tracking would not be necessary, simply a data association based on IOU would be enough to match the ground truth in consecutive frames (tracking-by-detection approach), named *Ideal_case*. It is worth mentioning here that in certain scenarios an assignment of ground truth bounding boxes by IOU in successive frames does not necessarily have to provide the perfect result (100% MOTP and 100% MOTA). However, it is always a good point to take as a reference to measure a tracking system. The second approach is the real case in which we could only run detection two times per second on average to guarantee real-time processing in the application, as an average call takes 0.39 seconds in a FPN with ImageNet and ResNeXt101. The assignment is made by IOU as in the previous case. This approach is named *Real_case*, and represents the baseline approach and it is the proposal that our system aims to significantly improve. To guarantee a fair comparison, in those cases in which a new object appears in the scene, this is added to the trackers list even though there is no detection in that particular frame. This affects our proposal and the *Real_case* approach since they do not use detection at all the frames, unlike the *Ideal_case*.

The results obtained in our dataset processing more than 1,000 objects are shown in Table 2. In view of the results we can see a clear improvement of our system with respect to the baseline, both in precision and accuracy, being in some cases close to the ideal case.

Table 3 shows a breakdown of the accuracy of the system in misses, false positives and mismatches. Misses are those ground truth objects that do not have an associated tracker, false positives are trackers that do not follow any real object and mismatches are identity switches. In view of the measures, we can observe an increase up to 46% in precision and 74% in accuracy, highlighting in the latter the maximum 99% of decrease in the number of mismatches thanks to data association.

## 5 Conclusions

We have presented a stand-alone multiple object visual tracking system that combines a DCF with a Kalman filter to handle occlusions, and solves the data association with deep features. The system runs in real-time —including the detector runtime— on a distributed architecture and, on a single CPU and GPU, is able to track tens of simultaneous objects in HD 1080 video. The proposal has been validated with several videos with more than 1,000 vehicles showing very good MOTP and MOTA metrics for up to 60 concurrent objects. Moreover, a fast version of the tracking system allows to track up to 180 simultaneous objects with a performance that is suitable for real-life applications.

## References


1. Caffe2. https://caffe2.ai/, accessed: 2019-04-15
2. DGT: Dirección General de Tráfico. http://www.dgt.es/es/, accessed: 2019-04-15
3. Docker. https://www.docker.com/, accessed: 2019-04-15
4. Github: DaSiamRPN. https://github.com/foolwood/DaSiamRPN, accessed: 2019-04-15
5. MOTChallenge: The Multiple Object Tracking Benchmark. https://motchallenge.net/, accessed: 2019-04-15
6. OpenCV: Open Source Computer Vision Library. https://opencv.org/, accessed: 2019-04-15
7. Pytorch. https://pytorch.org/, accessed: 2019-04-15
8. ROS: The Robot Operating System. http://www.ros.org/, accessed: 2019-04-13
9. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. Journal on Image and Video Processing p. 1 (2008)
10. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: European Conference on Computer Vision Workshops (2016)
11. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: IEEE International Conference on Image Processing (ICIP) (2016)
12. Danelljan, M., Bhat, G., Khan, F.S., Felsberg, M.: ECO: Efficient convolution operators for tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
13. Danelljan, M., Häger, G., Khan, F.S., Felsberg, M.: Discriminative scale space tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(8), 1561–1575 (2017)
14. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(3), 583–596 (2015)
15. Kristan, M., Leonardis, A., Matas, J., Felsberg, M., Pflugfelder, R., ehovin, L., Vojir, T., Bhat, G., Lukei, A., Eldesokey, A., Fernandez Dominguez, G., Garcia-Martin, A., Iglesias-Arias, ., Alatan, A., Gonzalez-Garcia, A., Petrosino, A., Memarmoghadam, A., Vedaldi, A., Muhi, A.: The Sixth Visual Object Tracking VOT2018 Challenge Results, pp. 3–53 (01 2019)
16. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
17. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision (ECCV) (2014)
18. Reid, D., et al.: An algorithm for tracking multiple targets. IEEE Transactions on Automatic Control **24**(6), 843–854 (1979)
19. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 3645–3649. IEEE (2017)
20. Zhu, Z., Wang, Q., Bo, L., Wu, W., Yan, J., Hu, W.: Distractor-aware siamese networks for visual object tracking. In: European Conference on Computer Vision (2018)