Appearances can be deceiving: Learning visual tracking from few trajectory annotations

Santiago Manen[1], Junseok Kwon[1], Matthieu Guillaumin[1], and Luc Van Gool[1,2]

[1] Computer Vision Laboratory, ETH Zurich
[2] ESAT - PSI / IBBT, K.U. Leuven
{smanenfr, kwonj, guillaumin, vangool}@vision.ee.ethz.ch

Abstract. Visual tracking is the task of estimating the trajectory of an object in a video given its initial location. This is usually done by combining at each step an appearance and a motion model. In this work, we learn from a small set of training trajectory annotations how the objects in the scene typically move. We learn the relative weight between the appearance and the motion model. We call this weight: *visual deceptiveness*. At test time, we transfer the deceptiveness and the displacement from the closest trajectory annotation to infer the next location of the object. Further, we condition the transference on an event model. On a set of 161 manually annotated test trajectories, we show in our experiments that learning from just 10 trajectory annotations *halves* the center location error and improves the success rate by about 10%.

Keywords: Visual tracking, Motion learning, Event modelling

1 Introduction

Visual object tracking from static cameras is an important topic of computer vision with applications in video surveillance, traffic monitoring, and augmented reality [27]. Typically, visual tracking starts with the initial location of the object in an initial frame. Then, an appearance model which is continuously updated attempts to localize the object at each subsequent frame [20, 18, 16, 19, 8].

An aspect of tracking in surveillance video that is often disregarded is the possibility to generalize from a set of annotated trajectories of other objects, *i.e.* supervised learning of visual tracking. Using trajectories as training data poses two main challenges which have discouraged the tracking community. First, trajectories are sequential data and objects can adopt a wide variety of trajectories. Hence, predicting trajectories is very challenging and its functional form is much more complex than for recognition and detection [4]. Second, such annotations are more costly to obtain than bounding boxes or class labels. Despite these problems, we believe that trajectories are very informative for a given scene, such that a few may suffice to robustly learn visual tracking. As any learning scenario, our framework has two phases, as shown in Fig. 1. The first is a *training phase*, where annotated data is collected for a scene and a model for tracking is learnt. The second is to use this model to improve tracking in a *test* sequence

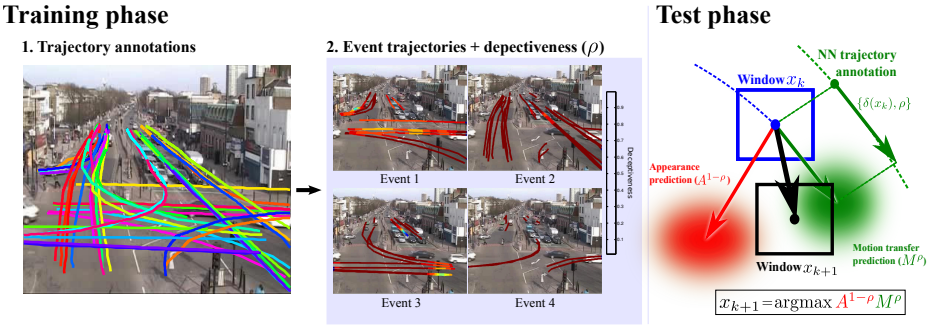


Fig. 1: At training time, we obtain trajectory annotations (1) we associate them to the events in the scene and we learn their *visual deceptiveness* ρ (2). At test time (right), we transfer displacement δ and deceptiveness ρ from the closest annotations in the same event and combine it with the appearance model to predict the next location of the object. (All our figures are best viewed in color)

of the same scene, where previously unseen objects are tracked from their initial bounding boxes. We run experiments on traffic surveillance videos and show that even a few trajectory annotations considerably improve tracking performance.

Similar to other visual tracking methods, we use a mixture of *appearance* (A) and *motion* (M) models [27]. Our motion model is based on transferring displacements observed in the training set and the mixture weight $\rho \in [0, 1]$ between A and M is determined at every step of tracking, also using transfer from annotated trajectories (*c.f.* Fig. 1). In essence, this mixture weight captures the situations in which the appearance of an object is not reliable for tracking. For this reason, we refer to ρ as *visual deceptiveness*. Importantly, in our framework, the user does not provide ρ for the ground-truth trajectories, but instead we propose an automatic way to learn it. As we show in this paper, the regions with high deceptiveness typically align with the failure cases of the appearance model: occlusions with static objects, tracking of small objects, bad lighting conditions.

Furthermore, we propose to associate the trajectories and their deceptiveness to the events discovered by [13]. In essence, [13] segments the video in consistent optical flow prototypes. This association improves the robustness of our deceptiveness estimation and removes ambiguities when transferring displacements. We show in our experiments that combining events and deceptiveness produces the best results in our datasets and improves over the state of the art [6–8].

In summary, our main contributions are: i) A framework to learn the visual deceptiveness of annotated trajectories, which are automatically selected; ii) A motion model based on transferring displacement and deceptiveness from trajectories; iii) The conditioning of this transfer on an event model; iv) The thorough evaluation of performance on two fully annotated test videos (with 161 trajectories in total, publicly available online). In the following, we first discuss related work in Sec. 2. We then present our tracking model in Sec. 3 and, in Sec. 4, the training procedure, *i.e.*, how visual deceptiveness is learnt and tracks selected for annotation. The conditioning on the event model is presented in Sec. 5 and our experimental validation in Sec. 6. We draw conclusions in Sec. 7.

2 Related work

There are many works on single and multiple target tracking [3, 10, 24, 15, 23]. The most relevant ones are those based on motion prior and trajectory patterns.

Spatial motion prior. Several authors [1, 30] have proposed tracking methods that uses static, space-dependent fields as a motion prior. These works simply encode the dominant direction of motion for each spatial region, limiting their use to simple videos. Rodriguez *et al.* [22] extended this idea with a set of motion directions, able to capture time-varying patterns. Further spatio-temporal dependencies between nearby motion patterns have been recently explored by Liu *et al.* [17] for tracking sport players. There, the authors exploit correlations between players and hand-designed motion rules for sport games.

We go beyond these works by conditioning our motion model spatially and temporally, without any further assumptions on the type of scene. We rely on a state-of-the-art unsupervised event discovery model [13] to capture, via optical flow prototypes, the dynamics of motion in a scene.

Analyzing trajectory patterns. In our work, we automatically select which trajectories should be annotated by a user. This relates to works, mostly on human tracking, that find typical trajectories patterns in videos. Zhou *et al.* [31] analyze the collective behavior of pedestrians in crowded scenes. They identify trajectory clusters and derive a generative model from which trajectories can be simulated. Similarly, when tracking in a simple physical environment, Kitani *et al.* [11] are able to predict plausible paths and destinations of people. These methods learn trajectory patterns in an unsupervised manner, hence assuming that automatic tracking succeeds. They are thus limited to scenarios where failures are relatively rare. In difficult tracking situations, [29] and [17] resort to a large amount of manually annotated trajectories. As this is very time-consuming, they have focused on a domain where tracks can be easily re-used: sport fields.

In our traffic scenes, automatic tracks fail rather often, so manual annotations are also necessary to correct them. To reduce the annotation cost as much as possible, our method automatically selects which trajectories shall be annotated.

Combining motion and appearance. Combining appearance and motion models has already been explored by various works. Yang and Nevatia [26] combined motion patterns and appearance models for multiple object tracking, but without assessing which component is more critical to the success of tracking. Cifuentes *et al.* [2] studies tracking with moving cameras and automatically chooses at each frame the most appropriate among six simple camera motion models (*e.g.*, "travelling", "forward", etc.). For single-object tracking, Kwon and Lee [14] proposed an MCMC sampling method to select, at each frame, the best tracker out of a pool of independent motion and appearance trackers.

In these works, the trackers are independent or naïvely combined. In contrast, we integrate motion and appearance in a single mixture model tracker, and we learn at training time how to adapt this weight dynamically to minimize tracking errors at test time. Notably, this weight (the *visual deceptiveness*) is not a fixed value: it depends on the time and the location of the object to track.

3 Tracking with motion and deceptiveness transfer

In this section, we describe our tracking model. For now, we assume that a set $\mathcal{T}$ of trajectory annotations with their respective local deceptiveness is available. As explained in the introduction, the *visual deceptiveness* ρ is a mixture weight which has a high value when the appearance model is less reliable. Thus, it identifies image regions where tracking based on appearance is more difficult.

More formally, let x_k be the current estimated position of the tracked object at frame k and $y_{1:k}$ be the set of all previous observations until this step. Then, like most single object tracking methods [7, 14], we compute the posterior distribution of the location at frame k based on the following Bayesian filtering:

$$p(x_{k+1}|y_{1:k+1}) \propto \underbrace{p(y_{k+1}|x_{k+1})}_{A(y_{k+1}|x_{k+1})^{1-\rho}} \underbrace{\int p(x_{k+1}|x_k)p(x_k|y_{1:k})\mathrm{d}x_k}_{M(x_{k+1}|y_{1:k})^{\rho}}, \tag{1}$$

where we have highlighted the definitions of A, M and ρ. In the equation above, we have abstracted the *appearance* model A, a *motion* model M and a weight ρ between the two. The value of ρ depends on x_k and determines the relative weight between the motion model and the appearance model.

As appearance model, we use the tracker of [7], which was shown to be the state of the art [25]. This model predicts the most likely location for the object in frame $k+1$ based on the appearance model trained up to frame k.

Displacement transfer motion model. Our motion model, illustrated in Fig. 1, is based on *transferring* a displacement probability distribution from ground-truth trajectories, which we model as a Gaussian distribution:

$$M(x_{k+1}|y_{1:k}) = \mathcal{N}(x_k + \delta(x_k), \sigma_M), \tag{2}$$

where $\delta(x_k)$ is the displacement at x_k as estimated from $\mathcal{T}$, and σ_M is a fixed variance. Note that x_k is uniquely determined by $y_{1:k}$. From $\mathcal{T}$, we also estimate $\rho(x_k)$, which leads to the following expression for the probability of x_{k+1}:

$$p(x_{k+1}|y_{1:k+1}) \propto A(y_{k+1}|x_{k+1})^{1-\rho(x_k)}\mathcal{N}(x_k + \delta(x_k), \sigma_M)^{\rho(x_k)}. \tag{3}$$

Similarly to [7], the scale and aspect ratio of the windows are sampled.

Nearest neighbor displacement and deceptiveness estimation from $\mathcal{T}$. Let $\mathcal{T} = \left\{\bar{x}^i_{k'}, i \in [1, \cdots, N]\right\}$ be the set of annotated trajectories, where $\bar{x}^i_{k'}$ is the bounding box of trajectory $\bar{x}^i$ at time k'. Since $\mathcal{T}$ contains few annotations, we estimate the displacement $\delta(x_k)$ and deceptiveness $\rho(x_k)$ using the nearest neighbour bounding-box $\bar{x}^i_{k'}$ of x_k in $\mathcal{T}$ and use its deceptiveness $\rho(\bar{x}^i_{k'})$, but only if $\bar{x}^i_{k'}$ is close enough. Formally, we use $\bar{x}^i_{k'} = \mathrm{argmin}_{b\in\mathcal{T}}\, d(x_k, b) = \|c(x_k) - c(b)\| \cdot (1 - IoU(x_k, b))$, where $c(\cdot)$ is the center of a window, $\|\cdot\|$ is the Euclidean norm and $IoU(\cdot,\cdot)$ is the intersection over union of two windows, and:

$$\delta(x_k) = \begin{cases} \bar{x}^i_{k'+1} - \bar{x}^i_{k'} & \text{if } d(x_k, \bar{x}^i_{k'}) \le \kappa \\ 0 & \text{otherwise} \end{cases} \quad \text{and } \rho(x_k) = \begin{cases} \rho(\bar{x}^i_{k'}) & \text{if } d(x_k, \bar{x}^i_{k'}) \le \kappa \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

That is, if $\bar{x}^i_{k'}$ is further than κ, our model falls back to the appearance model alone. This κ threshold avoids transferring from too distant neighbours. In our experiments, we use a σ_M of 2 pixels and a κ of 25.

4 Training phase: Learning to track

An important contribution of our work is to learn, from a set of trajectory annotations, the visual deceptiveness of the scene, *c.f.* Sec. 4.1. Annotating trajectories is a tedious task, so we want to limit their number. We present in Sec. 4.2 a method to automatically select which tracks the user should annotate.

4.1 Learning the visual deceptiveness of tracks

The crux of our system is the visual deceptiveness $\rho \in [0, 1]$ of trajectories, *i.e.*, the relative weight between motion and appearance models to be used when tracking (*c.f.* Eq. (3)). We propose to learn it in 2 steps from a set of annotated trajectories $\mathcal{T}$. We first learn a *raw deceptiveness* $\widetilde{\boldsymbol{\rho}}$ independently for each ground-truth trajectory. As $\widetilde{\boldsymbol{\rho}}$ tends to overfit to its own track, in the second step we diffuse it across neighboring trajectories to obtain a more consistent deceptiveness ρ that generalizes better. We detail these steps below.

Learning raw deceptiveness from one trajectory. In this section we want to learn the raw deceptiveness $\widetilde{\boldsymbol{\rho}}$ of a single ground-truth trajectory. This is a vector of values $\widetilde{\rho}_k$ for each frame k in the trajectory. We learn it by tracking the object automatically and seeing where and how much ground-truth motion information is needed to reproduce the ground-truth object trajectory.

Problem formulation. We start with the important observation that $\widetilde{\boldsymbol{\rho}}$ should be *sparse*, *i.e.*, rely on appearance as much as possible instead of motion. This prevents overfitting and ensures that unobserved or unexpected behavior of the objects will still be captured. In other words, we want $\widetilde{\boldsymbol{\rho}} > 0$ only when strictly necessary. We encode this objective with the following optimization problem:

$$\begin{aligned} \underset{\widetilde{\boldsymbol{\rho}}}{\text{minimize}} \quad & \sum_{k=1}^{T} \widetilde{\rho}_k \\ \text{subject to} \quad & \forall k \in [1, T], \quad 0 \leq \widetilde{\rho}_k \leq 1, \\ & \text{trackingError}(\widetilde{\boldsymbol{\rho}}) \leq \theta, \end{aligned} \tag{5}$$

where trackingError is a function that measures the overall error made by the tracker over the full trajectory and $\theta \geq 0$ is the threshold we use to decide whether the tracking was successful. In our experiments, we use the common *Center Location Error* (CLE) as trackingError. When tracking an object during T frames, trackingError $= \sum_{k=1}^{T} \xi_k \,/T$, where ξ_k is the distance at frame k between the center location of the automatic and the ground-truth trajectories.

Note that a solution always exists: setting $\widetilde{\boldsymbol{\rho}}$ to $\mathbf{1}$, the tracker is simply reproducing the ground-truth and the trackingError is 0. In practice, this never happens with the relatively loose error threshold θ of 15 pixels that we use.

Deceptiveness as a set of local Gaussian updates. The optimization problem in Eq. (5) is very general and complex to solve because the tracking error depends on the full trajectory and the full sequence of $\widetilde{\boldsymbol{\rho}}$. Fortunately, we can exploit the cumulative nature of trackingError and the sequential properties of tracking (as

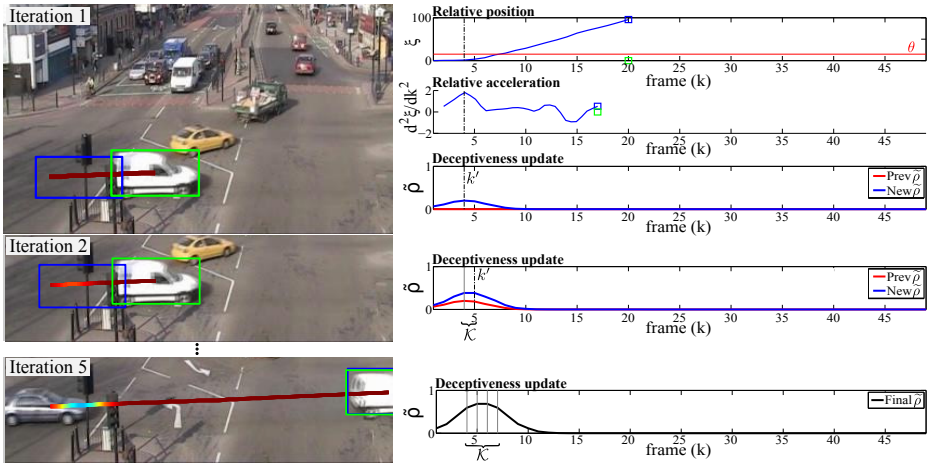


Fig. 2: We learn how to make the tracker reproduce a trajectory that is closer than trackingError to the ground-truth. We start by completely trusting appearance ($\widetilde{\boldsymbol{\rho}}=0$) and we track with it. If trackingError goes over θ then an update is applied to $\widetilde{\boldsymbol{\rho}}$ and we restart tracking. The process is repeated until $\widetilde{\boldsymbol{\rho}}$ is high enough to fulfill the trackingError constraint. (NB: The original images have been flipped for clarity.)

detailed below and in the supplementary material) to derive an efficient algorithm that starts from $\widetilde{\boldsymbol{\rho}} = \mathbf{0}$ and iteratively increments it until the trackError constraint is fulfilled. To do so, we propose to model $\widetilde{\boldsymbol{\rho}}$ as the truncated sum of a set of *local spatial Gaussian updates* $\mathcal{G}=\{g(\cdot|k'_i)\}_{i=1\ldots N_c}$, where each Gaussian update g has a fixed standard deviation σ_{up} and is centered around a frame k'_i:

$$\widetilde{\rho}_k = \min\left(1, \sum_{i=1}^{N_c} g(k|k'_i)\right), \qquad g(k|k'_i) = \widetilde{\mathcal{N}}(s(k)|s(k'_i), \sigma_{\text{up}}), \tag{6}$$

where $s(k)$ is the cumulative distance from the frame 1 to frame k. Since the values of $\widetilde{\boldsymbol{\rho}}$ are now sums over local Gaussian updates, learning $\widetilde{\boldsymbol{\rho}}$ becomes finding the (small) set $\mathcal{G}$ of frames k'_i where to apply those updates. Notably, multiple updates will accumulate near the most deceptive regions of a track.

In practice, we set σ_{up} to the size of the bounding box and g has a maximum response of 0.2 at the center of the Gaussian, hence the notation $\widetilde{\mathcal{N}}$.

Optimization process. Using this new definition of $\widetilde{\boldsymbol{\rho}}$, we now describe our variation of backjumping [21] to optimize of Eq. (5), as illustrated in Fig. 2. We initialize $\mathcal{G} = \varnothing$, $\widetilde{\boldsymbol{\rho}} = \mathbf{0}$ and $k = 1$, and repeat the following steps:

1. We track the object using $\widetilde{\boldsymbol{\rho}}$ from frame k until failure or end of track. If we reached the end of the track, we return $\widetilde{\boldsymbol{\rho}}$ as the solution. Otherwise we continue with the following:
2. We find the frame k' where the relative acceleration between the track and the ground-truth is maximum, *i.e.*, we use the 2nd-order derivative of ξ_k. We add $g(\cdot|k')$ to the set of local updates $\mathcal{G}$ and remove from $\mathcal{G}$ all the updates on later frames, since they are now potentially irrelevant.

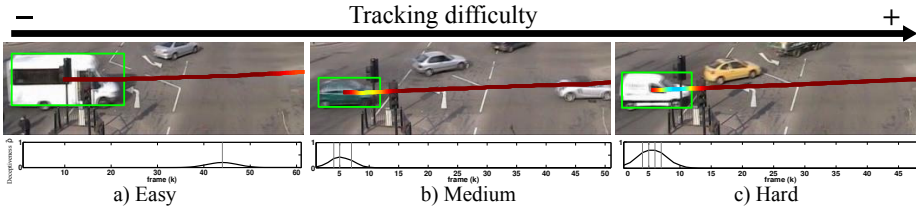


Fig. 3: Learning $\widetilde{\rho}$ independently for each object can give different results. Compared to b), a) is easier to track because it is bigger and c) is harder to track because it contrasts more with the occluding traffic lights.

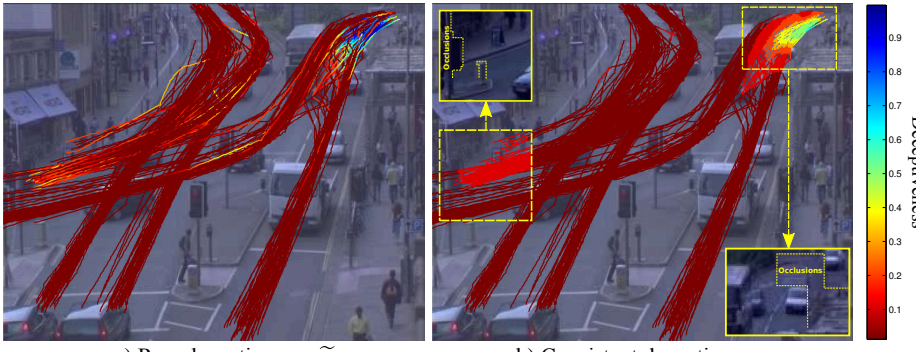


Fig. 4: Comparison of raw and consistent deceptiveness. Note the variability of $\widetilde{\rho}$ and how ρ manages to detect the underlying difficult zones, cropped occlusions in b). The intensity of ρ translates to the difficulty of the occlusions.

3. We recompute $\widetilde{\boldsymbol{\rho}}$ as for the new set of local Gaussians $\mathcal{G}$. Let $k'' < k'$ be the first frame for which the deceptiveness was impacted by this update.
4. We set $k = k'' - 1$ and go back to step 1.

In this algorithm, we have exploited many observations. First, trackingError is cumulative, hence we know that the tracking has failed as soon as the partial error reaches θ. Second, a failure at frame k can only be recovered with updates on earlier frames. Finally, an update at frame k invalidates all the later updates but does not affect earlier tracking.

To further prevent overfitting, we post-process as described below the deceptiveness of ground-truth tracks $\widetilde{\boldsymbol{\rho}}$ to ensure spatial consistency.

Spatial diffusion of raw deceptiveness. Some objects, due to size and appearance, are harder to track than others, *c.f.* Fig. 3. Therefore, the learnt $\widetilde{\boldsymbol{\rho}}$ are not always good estimates of the deceptiveness for all tracks. We aim to learn an *underlying* deceptiveness ρ that is *consistent* for neighboring annotations, hence finding regions in the scene where appearance is deceiving, *c.f.* Fig. 4.

To this end, we propose diffusing the $\widetilde{\rho}$ across the image pixels with [12], employing a grid-graph with 8-connected neighbourhoods. The graph has pairwise potentials encouraging smoothness via l_2-norm between neighbouring ρ values and unaries trying to preserve the original $\widetilde{\rho}$. Specifically, we define N_{ij} as the number of trajectories that include pixel (i, j). For each pixel (i, j) with at least

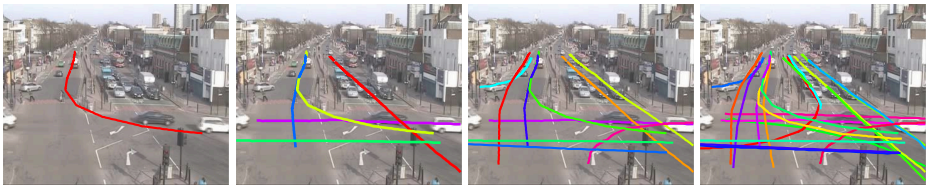


Fig. 5: Trajectory annotations we obtain with the IAP annotation selection method.

one observation ($N_{ij} > 0$), we assign it a representative observation $\widetilde{\rho}_{ij}$, which is the average of the observations in a spatial neighbourhood of radius of 25% of the average size of the windows. We then use $\widetilde{\rho}_{ij}$ to define unary potentials:

$$U_{ij}(\rho) = \begin{cases} \mathcal{N}(\rho|\widetilde{\rho}_{ij}, \sigma_D) & \text{if } N_{ij} > 0 \\ \text{Unif}(0,1) & \text{otherwise,} \end{cases} \tag{7}$$

which we discretize uniformly in 11 values: $0, 0.1, \ldots, 1$. That is, for pixels with observations ($Nij > 0$) our unary is a Gaussian centered around $\widetilde{\rho}_{ij}$ with a fixed standard deviation $\sigma_D = 0.2$. Otherwise ($N_{ij} = 0$), the unary is uniform.

Our experiments show that this diffusion improves tracking at test time.

4.2 Automatic annotation selection

In our work, we aim at improving tracking performance by exploiting trajectory annotations. As already argued, we want to limit the number of annotations that the user has to provide. Thus, we propose to first automatically select the most *representative* and *diverse* trajectories using a variant of Affinity Propagation (AP, [5]), which we denote as *Incremental Affinity Propagation* (IAP).

Using AP is beneficial in our scenario. First, it is a clustering algorithm which is based on *data similarities*, which are more natural for sequential data such as tracks than data points. Second, AP is an *exemplar-based* clustering method that selects the data sample that is the most representative for each cluster.

However, AP has the drawback that the exemplars for K clusters need not be a superset of the exemplars for $K' < K$ clusters. Ideally, if the user has already annotated K' tracks, observes that the performance can still be improved and believes that annotating $K > K'$ could help, then only $K - K'$ new tracks should be needed, and not K. Thus, whether the user chooses to annotate tracks one-by-one or by batches has no influence on the final tracking performance.

To achieve this property, we use the so-called *data preference* value P_i to ensure that, when AP returns a set ψ_N of N exemplars, it is a superset of ψ_{N-1}. This is done in a incremental fashion by modifying P_i dynamically, using P_i^k at step k. We start with $P_i^1 = \lambda_1$, a constant value determined so that only one exemplar is chosen: $\psi_1 = \{t_{i_1}\}$. Then, at each subsequent step $k \geq 2$, we assign an infinite preference to the already selected trajectories ψ_{k-1}, *i.e.*, $P_i^k = \infty$ if $t_i \in \psi_{k-1}$, and λ_k otherwise. λ_k is found by bisection such as AP returns a set of k exemplars, ψ_k. In practice, the sequence of λ_k is progressively increasing. This process is repeated until N trajectories have been incrementally clustered.

To define track similarities, we first represent a track t_i by a set of $Q = 50$ trajectory centers interpolated uniformly in space: $t_i = \{c_j^i, j = 1 \dots Q\}$. Interpolation in space is a very effective way to account for variation in time and speed, while avoiding to resort to more complex methods such as dynamic time warping. Then, for a pair of trajectories (t_i, t_j), we define their asymmetric similarity $\hat{s}(t_i, t_j)$ as $\hat{s}(t_i, t_j) = -\sum_{k=1}^{Q} \min_l \|c_k^i - c_l^j\|^2$. Finally, we symmetrize the similarities using $S(t_i, t_j) = \frac{1}{2}(\hat{s}(t_i, t_j) + \hat{s}(t_j, t_i))$.

Fig. 5 shows the trajectory annotations we obtain with this procedure. We show in the experiments that IAP yields more useful annotations than a random trajectory selection, especially for a small number of annotations.

5 Conditioning transfer on an event model

The flow of objects in structured scenes typically presents patterns and spatio-temporal dependencies [13, 1]. We build upon the model and code of [13] to exploit this idea and improve tracking performance. Put simply, [13] learns a Hidden Markov Model (HMM) temporal segmentation of the video by finding prototypical optical flows, (*c.f.* Fig. 6). The states of the HMM correspond to *events* and the transition probabilities between such events are also learnt. We can use this HMM to assign a global event to each frame in a video of the same scene. In our videos, it successfully detects high-level sequences in the scene, such as traffic light cycles. Among the benefits of [13], it is completely unsupervised, so we train it with sequences of 30 minutes without requiring any user interaction, and it automatically finds the optimal number of events. To exploit the events, we adapt both the training and test phases as described below.

Training phase. We use the HMM model to infer the event s_k of each frame k in the training set. Then, the bounding-boxes $\bar{x}_k^i$ of trajectories i in frame k are augmented with the event information. That is, a trajectory annotation i is now composed of a sequence of location, deceptiveness and event triplets $\{(\bar{x}_k^i, \rho_k^i, s_k^i)\}_k$. The event model does not impact how we learn the raw deceptiveness $\widetilde{\rho}$, since it is learned individually for each trajectory, but we apply the spatial smoothing separately for each event and its associated (sub-)trajectories.

Test phase. We also infer the most probable event segmentation of the test sequences using the HMM model. Thereby also obtaining an event s_k for each frame k. Then, we condition the nearest neighbour search described in Sec. 3 on s_k. We do this by simply restricting the search on the subset of the trajectory annotations that are also assigned to s_k: $\bar{x}_{k'}^i = \text{argmin}_{(b,\cdot,s_k)\in\mathcal{T}} d(x_k, b)$. After $\bar{x}_{k'}^i$ is found, its displacement and deceptiveness are transferred as in Sec. 3.

Fig. 6 shows how associating the trajectory annotations to events prevents ambiguities that can appear when searching the nearest neighbor. This conditioning corresponds to grouping the trajectory annotations by event, so we refer to these groups as *event trajectories*. Note from Fig. 6 how the trajectory annotations are much more powerful than optical flow in regions where objects are small or occluded. We show in the experiments that this association has a considerable impact on the tracking performance at test time.

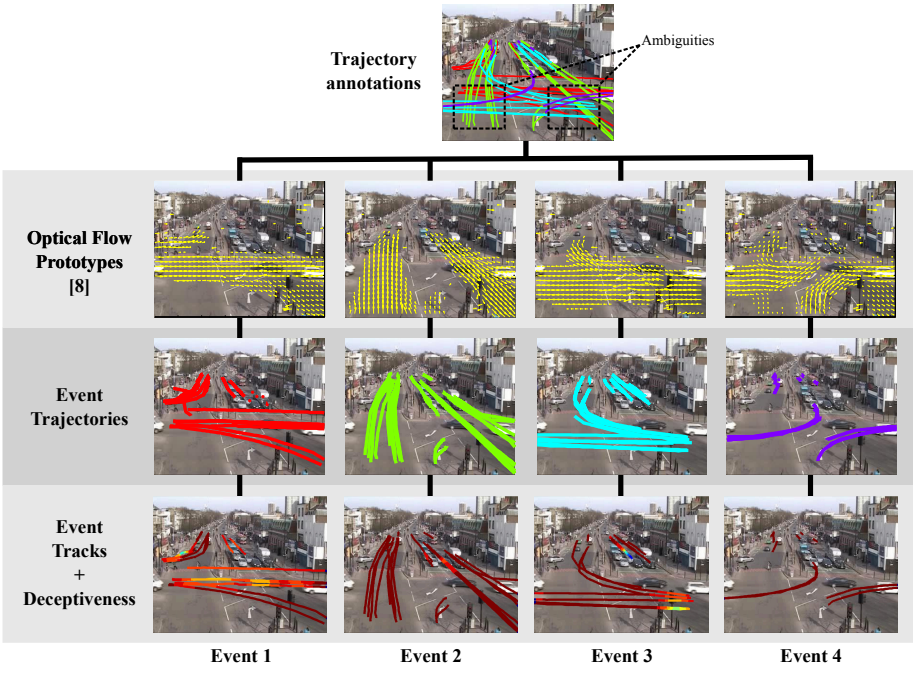


Fig. 6: Illustration of our full learning phase with 40 annotations for *Hospedales1*, with trajectories and deceptiveness conditioned on events. Ambiguous regions present in the original annotations (top) do not occur in event trajectories.

6 Experiments

We present in this section our experimental setup and results. In Sec. 6.1 we first describe our datasets and evaluation metrics. The remainder of the section is split in two parts: we first show in Sec. 6.2 a comparison with the state of the art and then, in Sec. 6.3, we evaluate the impact of the different components of our framework on the final results. We use the same parameters for all the experiments, as defined in the previous chapters. We also provide an analysis of the speed of our method in Sec. 6.4.

6.1 Dataset and experimental protocol

For our experiments, we have used two different scenarios from [13], denoted as *Hospedales1*, (*c.f.* Fig. 6), and *Hospedales3*, (*c.f.* Fig. 4). These two scenarios show two different crossings with heavy vehicle traffic and traffic light cycles. They are challenging due to occlusions, appearance changes (vehicles turning), low resolution and entry/exit points at the horizon.

We have used [28] to create 61 and 100 test track annotations, respectively, for a total length of 112,345 frames (74.5 minutes). For training, we collected 517 and 341 additional trajectory annotations, respectively. We have created them on a completely separate span of time of 10 minutes in each scene. In

Table 1: Comparison with the state of the art on 61 tracks in *Hospedales1* and 100 tracks in *Hospedales3*. The improvement of our approach with respect to the state of the art is shown in blue in parenthesis. In bold we show the best method between [7], [8], [6] and using 10 annotations in our method.

(a) Hospedales1 (61 tracks)

Metrics	[6]	[8][4]	[7]	[7] with Kalman	Proposed method with # annotations				
					0	10	20	30	40
CLE ↓	31.8	39.7	42.5	40.4	42.5	**9.6** (-70%)	10.6 (-67%)	9.6 (-70%)	10.4 (-67%)
SP ↑	0.35	0.41	0.48	0.49	0.48	**0.59** (+20%)	0.57 (+16%)	0.58 (+18%)	0.57 (+16%)

(b) Hospedales3 (100 tracks)

Metrics	[6]	[8][4]	[7]	[7] with Kalman	Proposed method with # annotations				
					0	10	20	30	40
CLE ↓	57.6	68.7	33.2	44.3	33.2	**23.9** (-28%)	28.4 (-14%)	17.7 (-47%)	16.0 (-52%)
SP ↑	0.15	0.21	0.43	0.39	0.43	**0.44** (+2%)	0.45 (+5%)	0.46 (+7%)	0.46 (+7%)

total we have 858 available training tracks.[3] However, our method only needs a small subset of them. This larger number of training tracks will help us evaluate important aspects of our method: (i) How does the tracking performance evolve as we use more trajectory annotations? (ii) Are our automatically selected tracks better than randomly selected tracks? We provide answers in Sec. 6.3.

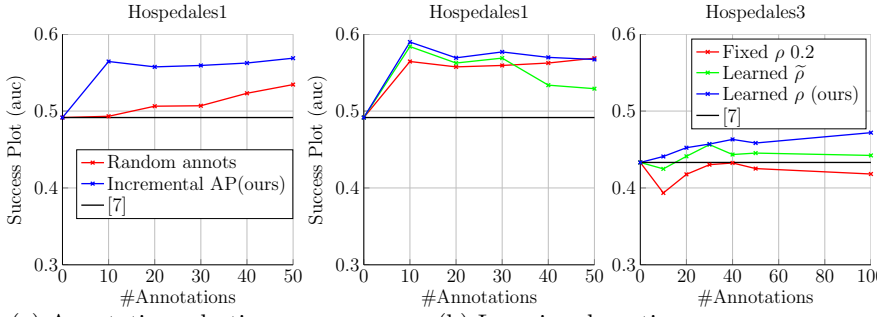
Concerning the event discovery model of [13], since it is unsupervised, we trained it on a 30-minute clip for each sequence, without the need for human intervention or annotation. These training clips were chosen such that they do not contain any of the objects of the test sequences.

To measure performance, we use two popular tracking performance metrics [25]: the *Center Location Error* (CLE) and the *Success Plot* (SP). CLE averages for each target object and each frame the distance between the center of the estimated window and the center of the ground-truth window. The Success Plot measures the percentage of frames where the target object is successfully detected for a certain intersection-over-union (IoU) threshold [25], *i.e.* the detection rate. For one target object, detection rate is plotted as a function of IoU, as it is varied between 0 and 1. To summarize the plots for all target objects, we compute their areas-under-the-curve (AuC) and we refer to the average over tracks as SP. A high-performing tracker will have a low CLE and a high SP.

6.2 Comparison with the state of the art

Tabs. 1a and 1b show the performance of the state of the art and our framework for *Hospedales1* and *Hospedales3*, respectively. Note that [6–8] rank within the 5 best single object trackers according to the recent exhaustive benchmark [25][5]. We show the performance of our framework for different number of annotations, a common theme that we adopt in the experimental section of this work. We obviously obtain the same result as [7] if we do not use any annotations, because in that case our model precisely falls back to the appearance

[3] This data is available on `www.vision.ee.ethz.ch/~smanenfr/deceptiveness`.
[4] We only evaluated [8] for the frames for which it provided a bounding box.
[5] We used the code of the authors publicly available in their webpage.



(a) Annotation selection (b) Learning deceptiveness

Fig. 7: In a) we compare our IAP annotation selection method with a random ranking baseline using a fixed ρ of 0.2 and all events (over 10 different runs). b) compares the results of the proposed framework using different variants of deceptiveness (see text).

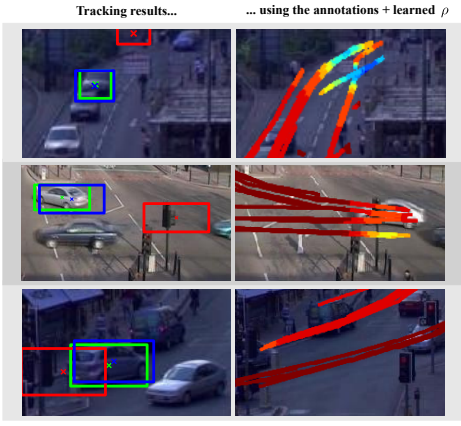
model. As we are provided with trajectory annotations we obtain a considerable improvement with respect to the state of the art. For example, with as few as 10 trajectory annotations, we obtain a relative improvement of 70% and 28% CLE for *Hospedales1* and *Hospedales3* respectively and 20% and 2% SP for these same scenarios. With 30 trajectory annotations, the improvement *Hospedales3* becomes 47% for CLE and 7% for SP. For comparison, we also extended [7] with a Kalman filter [9] (using a hand-tuned $\rho = 0.2$) and show that it brings a much lower improvement than our motion model. This occurs despite the use of the exact same fixed parameters for our motion model and the Kalman filter. The classical Kalman filter is simply not as powerful as our motion model since it does not learn from trajectory annotations. These results highlight the difficulty of tracking in our videos, and the huge benefits that our method is able to gain from a few manual annotations.

Note that our framework is not restricted to the use of the appearance model of [7]. It can easily accommodate future, better performing appearance models.

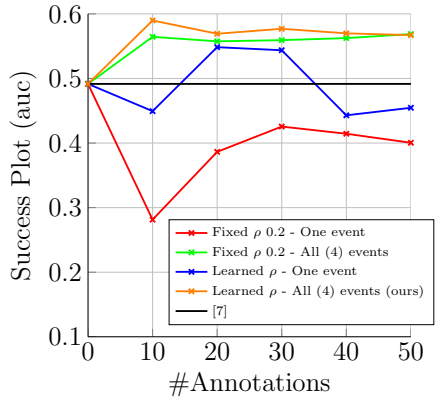
6.3 Baseline comparison

In this section we perform some baseline comparisons for the three main components of our proposed framework: (i) Our selection annotation approach, (ii) learning the visual deceptiveness and (iii) conditioning the transfer on the event model of [13]. For space reasons, we only show the SP performance metric. We show all the results in the supplementary material.

Automatic annotation selection. We compare the IAP annotation selection method described in Sec. 4.2 with an incremental random selection baseline. Fig. 7a shows the performance of the framework with these two methods and a fixed ρ of 0.2. For the random sampling of annotations, we show the average of the curves obtained with 10 different random sets. Due to space constraints, Fig. 7a only shows the results for the scene *Hospedales1*. The results of *Hospedales3*, which have similar conclusions, can be found in the supplementary material. Our annotation selection approach shows a considerable improvement



(a) Success cases



(b) Event modeling (*Hospedales1*)

Fig. 8: a) shows some cases in which our framework improves the tracking results. Ground truth in green, [7] in red and ours (40 annotations) in blue. Our method can track through deceiving zones by following previously learned trajectories in the regions where the original tracker failed at training time. b) shows a comparison of conditioning the transfer on one or all events with fixed and learned deceptiveness for *Hospedales1*.

for few annotations, since we pick representative and diverse trajectories. This is especially important for practical applications, where obtaining trajectory annotations is costly. As expected, this difference between the two methods decreases as more annotations become available and vanishes when using all of them.

Learning deceptiveness. Learning ρ is a major component of our framework and our contributions. We show in Fig. 7b the final tracking performance if we use a fixed deceptiveness of 0.2, if we learn it independently for each trajectory ($\widetilde{\rho}$) and if we additionally diffuse it spatially to obtain a consistent deceptiveness ρ. We draw two conclusions: i) Diffusing the deceptiveness across trajectories improves the performance of our approach and considerably impacts its stability with respect to the annotation sets. As expected, the diffusion aggregates the independently learned deceptiveness, extracting an underlying and consistent one, *c.f.* Fig. 4. ii) Learning deceptiveness with the proposed algorithm improves the results with respect to fixing the deceptiveness to a hand-tuned value of 0.2, and consistently outperforms the baseline tracker on both datasets. The improvement is especially noticeable in *Hospedales3*, since this dataset has zones with different degrees of difficulty to track. Fig. 4 shows how our framework is able to learn larger ρ values for regions with more difficult occlusions.

Fig. 8a shows some examples of especially difficult tracking regions with the corresponding deceptiveness that we have learned and how it helped to successfully track the objects at test time.

Conditioning transfer on an event model. The last component of our framework, and one of our contributions, is the conditioning of the transfer on the events in the scene. We show in Fig. 8b how much this can improve tracking performance in *Hospedales1*. Again, the results of *Hospedales3* can be found

in the supplementary material. Generally, using conditioning on events in the scene improves results, whether we learn the visual deceptiveness or not. Indeed, transferring from the event trajectories helps to solve ambiguities in the scene, *c.f.* Fig. 6. Importantly, even when using just one event the method we propose to learn ρ also gives better results than a fixed ρ. Overall, the best results are obtained with our full framework: learning deceptiveness and conditioning the transfer on the event model of the scene.

6.4 Speed

The proposed tracking framework is very efficient at test time. Computing the appearance model of [7] takes 175ms per frame on average. In contrast, computing our motion model comes at a cost of only 1.5ms with unoptimized code, excluding the real-time optical flow computation. This occurs because at test time the algorithm just has to do a nearest neighbor search and transfer the displacement and deceptiveness that was learned at training time.

7 Conclusion

In this paper, we have proposed a new visual tracking framework for a surveillance setup where we combine appearance and motion models in a single tracker. In this framework, we automatically learn from a small set of trajectory annotations how deceiving the appearance model is in different spatial and temporal regions of the scene. This leads to the concept of *visual deceptiveness*. At test time, we transfer the displacement and deceptiveness from the trajectory annotations. The framework is extended by conditioning the learning and transfer on an event model, which helps resolving ambiguities when searching for the closest trajectory from which to transfer. Moreover, we propose an incremental clustering approach to automatically select from the training sequence a small number of tracks which should be annotated by the user. We show in our experiments that those contributions are complementary and lead to state-of-the-art tracking results on 161 tracks that we publicly release.

One of the limitations of our current framework is that it disregards context, *i.e.* the position of the other vehicles, to learn deceptiveness. We plan to include such context modelling in future work. We also want to cast our framework in an active learning scenario, where the learned deceptiveness on the first few annotated tracks will guide the selection of new trajectory annotations so as to further improve tracking. Ideally, the system would also find the number of annotated tracks that are needed.

Acknowledgements This work was supported by the European Research Council (ERC) under the project VarCity (#273940). The authors gratefully acknowledge support by Toyota.

References

1. Ali, S., Shah, M.: Floor fields for tracking in high density crowd scenes. In: ECCV (2008)
2. Cifuentes, C.G., Sturzel, M., Jurie, F., Brostow., G.J.: Motion models that only work sometimes. In: BMVC (2012)
3. Collins, R.T., Liu, Y., Leordeanu, M.: Online selection of discriminative tracking features. PAMI 27(10), 1631–1643 (2005)
4. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html
5. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315, 972–976 (2007)
6. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: ICCV (2011)
7. Jia, X., Lu, H., Yang, M.H.: Visual tracking via adaptive structural local sparse appearance model. In: CVPR (2012)
8. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. PAMI 34(7), 1409–1422 (2012)
9. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering (1960)
10. Khan, Z., Balch, T., Dellaert, F.: MCMC-based particle filtering for tracking a variable number of interacting targets. PAMI 27(11), 1805–1918 (2005)
11. Kitani, K.M., Ziebart, B.D., Bagnell, J.A., Hebert, M.: Activity forecasting. In: ECCV (2012)
12. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. PAMI (2006)
13. Kuettel, D., Breitenstein, M.D., Van Gool, L., Ferrari, V.: What's going on? Discovering spatio-temporal dependencies in dynamic scenes. In: CVPR (2010)
14. Kwon, J., Lee, K.M.: Tracking by sampling trackers. In: ICCV (2011)
15. Leibe, B., Schindler, K., Cornelis, N., Van Gool, L.: Coupled object detection and tracking from static cameras and moving vehicles. PAMI 30(10), 1683–1698 (2008)
16. Li, X., Dick, A., Wang, H., Shen, C., van den Hengel, A.: Graph mode-based contextual kernels for robust svm tracking. ICCV (2011)
17. Liu, J., Carr, P., Collins, R.T., Liu, Y.: Tracking sports players with context-conditioned motion models. In: CVPR (2013)
18. Mei, X., Ling, H.: Robust visual tracking using l1 minimization. ICCV (2009)
19. Oron, S., Bar-Hillel, A., Levi, D., Avidan, S.: Locally orderless tracking. CVPR (2012)
20. Perez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. ECCV (2002)
21. Prosser, P.: Hybrid algorithms for the constraint satisfaction problem. Computational intelligence (1993)
22. Rodriguez, M., Ali, S., Kanade, T.: Tracking in unstructured crowded scenes. In: ICCV (2009)
23. Segal, A.V., Reid, I.D.: Latent data association: Bayesian model selection for multi-target tracking. ICCV (2013)
24. Smith, K., Carleton, A., Lepetit, V.: General constraints for batch multiple-target tracking applied to large-scale videomicroscopy. CVPR (2008)
25. Wu, Y., Lim, J., Yang, M.H.: Online object tracking: A benchmark. CVPR (2013)

26. Yang, B., Nevatia, R.: Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In: CVPR (2012)
27. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. In: ICASSP (2006)
28. Yuen, J., Russell, B.C., Liu, C., Torralba, A.: Labelme video: Building a video database with human annotations. In: ICCV (2009)
29. Zhang, T., Ghanem, B., Ahuja, N.: Robust multi-object tracking via cross-domain contextual information for sports video analysis. In: ICASSP (2012)
30. Zhao, X., Medioni, G.: Robust unsupervised motion pattern inference from video and applications. In: ICCV (2011)
31. Zhou, B., Wang, X., Tang, X.: Understanding collective crowd behaviors:learning a mixture model of dynamic pedestrian-agents. In: CVPR (2012)