

Eliminating Exposure Bias and Metric Mismatch in Multiple Object Tracking

Andrii Maksai Pascal Fua

Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL)

{andrii.maksai, pascal.fua}@epfl.ch

Abstract

Identity Switching remains one of the main difficulties Multiple Object Tracking (MOT) algorithms have to deal with. Many state-of-the-art approaches now use sequence models to solve this problem but their training can be affected by biases that decrease their efficiency. In this paper, we introduce a new training procedure that confronts the algorithm to its own mistakes while explicitly attempting to minimize the number of switches, which results in better training.

We propose an iterative scheme of building a rich training set and using it to learn a scoring function that is an explicit proxy for the target tracking metric. Whether using only simple geometric features or more sophisticated ones that also take appearance into account, our approach outperforms the state-of-the-art on several MOT benchmarks.

1. Introduction

A common concern in Multi Object Tracking (MOT) approaches is to prevent identity switching, the erroneous merging of trajectories corresponding to different targets into a single one. This is difficult in crowded scenes, especially when the appearances of the individual target objects

are not distinctive enough. Many recent approaches rely on tracklets—short trajectory segments—rather than individual detections, to keep track of the target objects. Tracklets can be merged into longer trajectories, which can be split again when an identity switch occurs.

Most state-of-the-art approaches [33, 18, 24, 56, 26] operate on sequences or clusters of detections, often with the help of deep, recurrent neural networks. This requires training the sequence models and is subject to one or both of two well-known problems, which our approach overcomes:

- **Metric mismatch.** It occurs when training by optimizing a metric poorly aligned with the actual desired performance during inference. In MOT, one example is the use of a classification loss to create trajectories optimal for a tracking-specific metric, such as **MOTA** [7] or **IDF** [45]. To eliminate this mismatch, we introduce an original way to score tracklets that is an explicit proxy for the **IDF** metric and can be computed without the ground truth. We use it to identify how confidently the person is tracked, predict tighter bounding box locations, and estimate whether the real trajectory extends beyond the observed tracklet.
- **Exposure bias.** It stems from the model not being exposed to its own errors during training and results in very

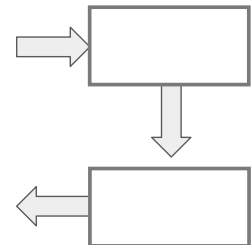


Figure 1. Keeping track in a difficult situation. **Top row:** Because of the occlusion created by the passing car, a tracker can easily return a trajectory that includes several identity switches. The corresponding bounding boxes inside camera’s field of view are shown on the right. **Bottom row:** Our algorithm not only eliminates identity switches but also regresses to a set of much tighter bounding boxes. In this example our algorithm did it solely on the basis of simple geometric features without requiring the use of appearance information.

different data distribution observed during training and inference/tracking. We remove this bias by introducing a much more exhaustive, yet computationally feasible, approach to exploiting the data while training the model. To this end, during training, we do not limit ourselves to only using tracklets made of detections of one or two people as in [39, 34, 48]. Instead, we consider any grouping of tracklets produced by the tracking algorithm to be a potential trajectory but prevent a combinatorial explosion by controlling the number of tracklets that share many common detections. This yields a much richer training dataset, solves the exposure bias problem, and enables our algorithm to handle confusing situations in which a tracking algorithm may easily switch from one person to the next or miss someone altogether. Fig. 1 depicts one such case. Note that this can be done even when appearance information is *not* available.

Our contribution is therefore a solution to these two problems. By integrating it into an algorithm that only uses very simple features—bounding boxes, detector confidence—we outperform other approaches that do not use appearance features either. By also exploiting appearance-based features, we similarly outperform state-of-the-art approaches that do. Taken together, these results demonstrate the effectiveness of our training procedure.

In the remainder of this paper, we first briefly review related work and current approaches to mitigating metric mismatch and exposure bias. We then introduce our approach to tracking; it is a variation of multiple hypothesis tracking designed for learning to efficiently score the tracklets. Next, we describe the exact form of our scoring function and its ability to reduce both mismatch and bias. Finally, we present our results.

2. Related work

Multiple Object Tracking (MOT) has a long tradition, going back many years for applications such as radar tracking [9]. With the recent improvements of object detectors, the tracking-by-detection paradigm [2] has become a *de facto* standard and has proven effective for many applications such as surveillance or sports player tracking. It involves first detecting the target objects in individual frames, associating these detections into short but reliable trajectories known as tracklets, and then concatenating these tracklets into longer trajectories. They can then be used to solve tasks such as social scene understanding [1, 36, 3], future location prediction [30], or human dynamic modeling [16].

While grouping individual detections into trajectories it is difficult to guarantee that each resulting trajectory represents a *whole* track of a *single* individual, that is, that there are no identity switches.

Many approaches rely on appearance [6, 17, 28, 57, 58, 12, 32, 46], motion [15], or social cues [20, 43]. They

are mostly used to associate pairs of detections, and only account for very short-term correlations. However, since people trajectories are often predictable over many frames once a few have been seen, superior performance could be obtained by modeling behavior over longer time periods [22, 27, 36]. Increasing availability of annotated training data and benchmarks, such as MOT15-17 [29, 38], DukeMTMC [45], PathTrack [37], and Wildtrack [10] now makes it possible to learn the data association models required to leverage this knowledge. Since this is what our method does, we briefly review here a few state-of-the-art approaches to achieving this goal.

2.1. Modeling Longer Sequences

The work of [42, 41] is one of the first recent approaches to modeling long trajectories using a recurrent neural network. The algorithm estimates ground-plane occupancy, but does not perform explicit data association. [39] presented an approach to performing data association without using appearance features by predicting the future location of the target. Several MOT approaches have followed, using sequence models to make data association more robust for the purpose of people re-identification [48, 34], learning better social models [1], forecasting future locations [30, 54] or joint detection, tracking, and activity recognition [3].

These models are usually trained on sample trajectories that perfectly match a single person’s trajectory or only marginally deviate from that, making them vulnerable to exposure bias. Furthermore, the loss function is usually designed primarily for localization or identification rather than fidelity to a ground truth trajectory. This introduces a mismatch with the metric, usually **IDF** [45] or **MOTA** [7], which reflects more reliably the desirable behavior of the algorithm.

Most state-of-the-art approaches that use sequence models rely on one of two optimization techniques, hierarchical clustering for data association [50, 58, 45, 33, 18, 24] or multiple hypothesis tracking [56, 26, 11]. The former involves valid groups of observations without shared hypotheses while the latter allows for conflicting sets of hypotheses to be present until the final solution is found. The approach most similar to our is that of [26]. It also uses a combination of multiple hypothesis tracker and a sequence model for scoring. However, the training procedure mostly relies on ground truth information and is therefore more subject to exposure bias. Another closely related method is that of [39] that trains a sequence model for data association solely from geometric features and is therefore well-suited for comparison with our approach when also using only geometric cues. These methods are all recent and collectively represent the current state-of-the-art. In Section 5, we will therefore treat them as baselines against which we can com-

pare our approach.

2.2. Reducing Bias and Metric Mismatch

Since exposure bias and metric mismatch (also called loss-evaluation mismatch [49]) are also problems in Natural Language Processing (NLP) [51] and in particular in machine translation [53], several methods have been proposed in these fields to reduce it [44, 4]. Most of them, however, operate under the assumption that output sequences can comprise any character from a predefined set. As a result, they typically rely on a beam-search procedure, which itself frequently uses a language model to produce a diverse set of candidates that contains the correct one. More generally, techniques that allow training models without differentiable relation between inputs and outputs such as policy gradient [52], straight-through estimation [5], and Gumbel-Softmax [23] can be seen as methods reducing exposure bias. In this domain, our approach is similar to DAgger [47] and SEARN [14], which iterate between learning the policy and obtaining data, to address exposure bias problem.

Unfortunately, in the case of MOT, the detections form a spatio-temporal graph in which many nearly identical trajectories can be built. This can easily overwhelm standard beam-search techniques: when limiting oneself to only the top scoring candidates to prevent a combinatorial explosion, it can easily happen that only a set of very similar but spurious trajectories will be considered and the real one ignored. This failure mode has been addressed in the context of single-object tracking and future location prediction in [21, 35] with a tracking policy learned by reinforcement learning and in [13] by introducing a spatio-temporal attention mechanism over a batch of images, thus ensuring that within the batch there is no exposure bias. Instead, the algorithm relies on historical positive samples from already obtained tracks, thus re-introducing it. For MOT, a reinforcement learning-based approach has been proposed [55] to decide whether to create new tracklets or terminate old ones. This is also addressed in [48] but the learning of sequence models is done independently and is still subject to exposure bias. Approach of [36] attempts to explicitly optimize for the IDF metric. It does so by refining the output of other tracking methods. This reduces the metric mismatch but the sequence scoring model is hard-coded rather than learned and we will show that learning it yields better results.

3. Tracklet-Based Tracking

Our approach to tracking relies on creating and merging tracklets to build high-scoring trajectories as in multiple hypothesis tracking [25]. In this section, we formalize it and describe its components, assuming that the scoring function is given. The scoring function and how it is learned will be discussed in the following section.

3.1. Formalization

Let us consider a video sequence made of N frames, on which we run a people detection algorithm on each frame individually. This yields a set D of people detections d_i

⁴, where the four elements of d_i are the coordinates of the corresponding bounding box in the image. We represent a tracklet T as a $4 \times N$ matrix of the form $[d_1, d_2, \dots, d_N]$. In practice, tracklets only rarely span the whole sequence. We handle this by setting d_n to zero for frames in which the person's location is unknown. The first non-zero column of a tracklet is therefore its start and the last its end. Two tracklets T_1 and T_2 can be merged into a single one if there is no single frame in which they contain different detections.

Let $f : 4 \times N \rightarrow F \times N$ be a *feature* function that assigns a feature vector of dimension F to each column of a tracklet. In practice, these features can be bounding box coordinates, confidence level, and shift from the nearest detection in a previous frame. They can also be image-based features associated to the detection and we list them all in Section 5.3. Let us further assume that we can compute from these features a score $S(T)$ that is high when the tracklet truly represents a single person's trajectory and low otherwise. Tracking can then be understood as building the set of non-overlapping tracklets T_j that maximizes the objective function

$$S(\bigcup_j T_j). \quad (1)$$

In the remainder of this section, we will assume that S is given and assigns low scores to the wide range of bad candidate trajectories that can be generated, and high scores to the real ones.

3.2. Creating and Merging Tracklets

We iteratively merge tracklets to create ever longer candidate trajectories that include the real ones while suppressing many candidates to prevent a computationally infeasible combinatorial explosion. We then select an optimal subset greedily. We consider two trajectories to be overlapping when they have a large intersection over union. More specifically, if the total number of pixels shared by bounding boxes of the two tracklets, normalized by the minimum of the sum of areas of bounding boxes in each of them, is above a threshold C_{IoU} . We also eliminate tracklets that are either shorter than N - the length of the batch, or whose score is below another threshold C_{score} . C_{IoU} and C_{score} are hyper-parameters that we estimate on a validation set. Outlined procedure involves the two main steps described below.

3.2.1 Generating Candidate Trajectories

The set of candidate trajectories must contain the real ones but its size must be kept small enough to prevent a com-

binatorial explosion. To this end, given the initial set of detections D , which we take to be the initial tracklet set.

We then iterate the following two steps for $n = 2, \dots, N$.

1. **Growing:** Merge pairs of tracklets that can be merged and result would be bigger than the biggest of two by 1. Tracklets with k_1 and k_2 non-zero detections yields a tracklet of $\max(k_1, k_2) + 1$ non-zero detections, that includes non-zero detections from both of them.
2. **Pruning:** Given tracklet T_1 , for all T_2 that were merged with it during growing phase, only retain the merger that maximizes the score $S(\cdot)$.

This process keeps the number of hypotheses linear with respect to the number of detections. Yet, it retains a candidate for every possible detection. This prevents the algorithm from losing people and terminating trajectories too early even if mistakes are made early in the pruning process. We give an example in Fig. 2, (b). In supplementary material, we compare this heuristic to several others and show that it is effective at preventing combinatorial explosion without losing valid hypotheses.

3.2.2 Selecting Candidates

Given the resulting set of tracklets, we want to select a compatible subset that maximizes our objective function. To this end we select a subset of hypotheses with the best possible sum of scores, subject to a non-overlapping constraint. We do this greedily, starting with the highest scoring trajectories. As discussed in the supplementary material, we also tried a more sophisticated approach that casts it as an integer program solved optimally, and the results are similar.

4. Learning to Score

The scoring function $S(\cdot)$ of Eq. 1 is the heart of the tracking procedure of Section 3. When we create and merge tracklets, we want it to favor those that can be associated to a single person without identity switch, that is, those that score well in terms of the **IDF** metric. We choose **IDF** over other popular alternative such as **MOTA** because it has been shown to be more sensitive to identity switches [45].

In the remainder of this section, we first define S , which we implement using the deep network depicted by Fig. 2(a). We then describe how we train it.

4.1. Defining the Scoring Function

Ideally, we should have $S(T) = \text{IDF}(T, G)$ for every tracklet T and the corresponding ground truth trajectory G . Unfortunately, at inference time, G is unknown by definition. To overcome this difficulty, recall from [45] that **IDF** for tracklet $T = [d_1, \dots, d_n]$ and ground truth trajectory $G = [g_1, \dots, g_n]$ is defined as twice the number

of detections matched by ground truth, over sum of total lengths of the two:

$$\text{IDF}(T, G) = \frac{2 \cdot \sum_{n: d_n = \emptyset, g_n = \emptyset} (\text{IoU}(d_n, g_n) > 0.5)}{|\{n : d_n = \emptyset\}| + |\{n : g_n = \emptyset\}|}, \quad (2)$$

where **IoU** is the intersection over union of the bounding boxes. To approximate it without knowing G , we write

$$S(T) = \frac{2 \cdot \sum_{n: d_n = \emptyset, \text{lab}_n > 0.5} \text{iou}_n}{|\{n : d_n = \emptyset\}| + |\{n : \text{lab}_n > 0.5\}|}, \quad (3)$$

assuming that the deep network of Fig. 2, (a) has been trained to regress from T to

- iou_n : the prediction of intersection over union of the d_n and g_n boxes;
- lab_n : the prediction of whether the ground truth trajectory exists in frame n .

We also train our network to predict the necessary change to bounding box d_n to produce the ground truth bounding box g_n , which we denote sft_n . It is not used to compute S , but can be used during inference to better align the observed bounding boxes with the ground truth.

To train the network to predict the lab_n , iou_n , and sft_n values introduced above, we define a loss function that is the sum of errors between predictions and ground truth. We write it as

$$L(T, G) = \sum_{n=1}^N L_{\text{lab}}(d_n, g_n) + \sum_{n: d_n = \emptyset} L_{\text{iou}}(d_n, g_n) + \sum_{n: d_n = \emptyset} L_{\text{sft}}(d_n, g_n), \quad (4)$$

$$L_{\text{lab}}(d_n, g_n) = \|\text{lab}_n - (g_n = \emptyset)\|_2,$$

$$L_{\text{iou}}(d_n, g_n) = \|\text{iou}_n - \text{IoU}(d_n, g_n)\|_2,$$

$$L_{\text{sft}}(d_n, g_n) = 1 - \text{IoU}(d_n + \text{sft}_n, g_n),$$

where $d_n + \text{sft}_n$ denotes the shifting the bounding box d_n by sft_n .

Arguably, we could have trained the network to directly regress to **IDF** instead of first estimating iou_n , lab_n , and sft_n and then using the approximation of Eq. 3 to compute it. However, our experiments have shown that asking more detailed feedback for every time step, as we do, forces the network to better understand motion, while a good estimation of **IDF** can be often produced by an average prediction.

We chose not to apply any weight factors to the components of the loss function because its components could be seen as identifying the false positive (when lab should be zero) and false negative (when $\text{IoU} < 0.5$) errors, and since we wanted to weigh the two equally, we did not use any weight factors to L_{lab} , L_{sft} , L_{iou} .

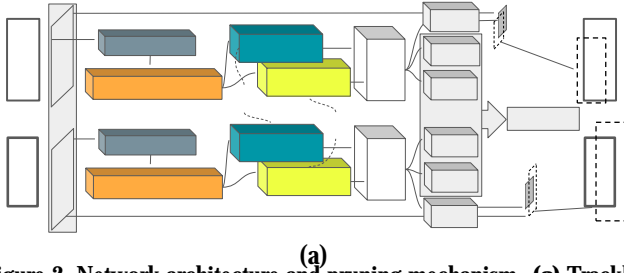


Figure 2. Network architecture and pruning mechanism. (a) Tracklet features are passed through an embedding layer and then processed using a bi-directional LSTM. Its outputs are used to predict the IoU with ground truth bounding boxes (iou), presence of a person in a scene (lab), and regress bounding box shift to obtain ground truth bounding boxes (sft). (b) Candidate tracklets starting from two different bounding boxes in blue and ending with bounding boxes in white. In this case, during pruning phase the best tracklets, shown in green, are assigned the highest score and retained, and all others are eliminated.

4.2. Training Procedure

The key to avoiding exposure bias while training the network is to supply a rich training set. To this end, we alternate between the following two steps:

1. Run the hypothesis generation algorithm of Section 3.2 using current network weights when evaluating S ;
2. Add the newly created tracklets to the training set and perform a single epoch of training.

In addition to learning the network weights, this procedure helps refine the final tracking result: The tracking procedure of Section 3 makes discrete choices about which hypotheses to pick or discard, which is non-differentiable. We nevertheless help it make the best choices by training the model on all candidates, both good and bad, encountered during tracking. In other words, our approach makes discrete choices during training and then updates the parameters based on all hypotheses that *could* have been selected, which is similar in spirit to using a straight-through estimator [5].

While simple in principle, this training procedure must be carefully designed for optimal performance. We list here the most important details of our implementation and study their impact in the ablation study.

Stopping criterion. We start the process with random network weights and stop it when the training set size increases by less than 5% after iterating the process 10 times. We then fully train the model on the whole resulting training set. This process can be understood as a slow traverse of the search space. It starts with an untrained model that selects random hypotheses. Then, as the training progresses, new hypotheses are added and help the network both to differentiate between good and bad alternatives and to pick the best ones with increasing confidence.

Randomized merging During inference, we grow each tracklet by merging it with one that yields the highest possible score. By contrast, during training, we make the training set more diverse by randomizing the merging process. To do

that, when selecting between two candidates, we can select the one with the lower score with probability proportional to the exponent of the score difference, divided by the temperature. We initially set the coefficient so that the optimal pair is almost always chosen and we then progressively reduce it to increase the randomness.

Balancing the dataset. One potential difficulty is that this procedure may result in an unbalanced training set in terms of the **IDF** values to which we want to regress. We solve this by splitting the dataset into 10 groups by **IDF** value $([0.0; 0.1], [0.1; 0.2], \dots, [0.9, 1.0])$, selecting all samples from the smallest group, and then the same number from each other group. This enables us to perform h -hard-mining by selecting h/K samples at random and retaining the K that contribute most to the loss.

5. Results

We now present the datasets we use, baselines we compare against, our results, and finally a qualitative analysis.

5.1. Datasets

We used the following publicly available datasets to benchmark our approach.

DukeMTMC [45]. It contains 8 sequences, with 50 minutes of training data, and testing sequences of 10 ("Hard", dense crowd traversing several camera views) and 25 minutes ("Easy") with hidden ground truth, at 60fps.

MOT17 [38]. It contains 7 training-testing sequence pairs with similar statistics and hidden ground truth for the test sequences, spanning 785 trajectories and both static and moving cameras. For each, there are 3 different sets of detections using different algorithms, which makes it possible to evaluate the quality of the tracking without overfitting to a specific detector.

MOT15 [29]. It contains 11 training and 11 testing sequences, with moving and stationary cameras in various settings. The ground truth for testing is hidden, and for each

testing sequence there is a sequence with roughly similar statistics in the training data.

5.2. Baselines

We compare against a number of recent algorithms that collectively represent the state-of-the-art. We distinguish below between those that do not use appearance cues and those that do, and provide expanded description in supplementary materials. Baselines without appearance are **LP2D** [29], **RNN** [39], **PTRACK** [36], and **SORT** [8, 40]. Baselines with appearance are **MHT** [56], **CDSC** [50], **REID** [58], **BIPCC** [45], **DMAN** [59], **JCC** [24], **MOTDT17** [33], **MHTBLSTM** [26], **EDMT17** [11], and **FWT** [18]. Most similar to our approach are **RNN** and **MHTBLSTM**, which both use sequence model to score the tracks, but use a different loss function and training data. **MHTBLSTM** also relies on multiple hypothesis tracking.

5.3. Experimental Protocol

In this section, we describe the features we use in practice along with our approach to batch processing, training, and choosing hyperparameters.

Features For a fair comparison against the two classes of baselines described above, we use either features in which appearance plays no part or features that encode actual image information. We describe them below.

Appearance-less features. We use the following simple features that can be computed from the detections without further reference to the images:

- Bounding box coordinates and confidence ($\times 5$).
- Bounding box shift with respect to previous and next detection in the tracklet ($\times 8$).
- Social feature - a description of the detections in the vicinity, $\times 3 \times M$. It comprises offsets to the M nearest detections and their confidence values. All values are expressed relative to image size for better generalization.

Appearance-based features. As a basis for appearance, we used the 128-dimensional vector produced from a bounding box by the re-identification model of [19]. Distance in euclidian space between such vectors indicate similarity between people appearances and likelihood that they are the same person. To this end, we provide following additional features in our appearance-based model:

- Appearance vector for each bounding box ($\times 128$).
- Euclidian distance from appearance in the bounding box to the appearance that best represents trajectory so far before the current batch, if one is available ($\times 1$). To pick the appearance that best represents trajectory so far,

Method	App.	IDF	MOTA	IDs	IDF	MOTA	IDs
Sequence		Easy			Hard		
OURS	+	84.0	79.2	169	76.8	65.4	267
MHT	+	80.3	78.3	406	63.5	59.6	1468
REID	+	79.2	68.8	449	71.6	60.9	572
CDSC	+	77.0	70.9	693	65.5	59.6	1637
OURS-geom	-	76.5	69.3	426	65.5	59.1	972
PTRACK	-	71.2	59.3	290	65.0	54.4	661
BIPCC	+	70.1	59.4	300	64.5	54.6	652

Table 1. Results on the **DukeMTMC** dataset. The second column indicates whether or not the method uses appearance information.

we computed euclidian distances between each pair of appearances in the trajectory, and picked one with the smallest sum of distances to all others.

- Crowd density feature - distance from the center of current bounding box to the center of nearest 1st, 5th, and 20th detection in the current frame ($\times 3$). As we discuss in the ablation study, that feature made impact on the behavior of our model with appearance in very dense crowd scenarios.

Batch processing. In Section 3, we focused on processing a batch of N images. In practice, we process longer sequence by splitting them into overlapping batches, shifting each one by $\frac{N}{3}$ frames. While pruning hypotheses, we never suppress all those that can be merged with trajectories from the previous batch. This ensures that we can incorporate all tracks from the previous batch. We used 3-second long batches for training as in [48]. During inference, we observed that our model is able to generalize beyond 3s, and having longer batches can be beneficial in cases of long occlusions. Inference used 6-second long batches.

Training and Hyperparameters For all datasets and sequences, cross-validation revealed that thresholds C_{IoU} and C_{score} of Sec. 3.2 equal to 0.6 and the hard-mining parameter h of Sec. 4.2 equal to 3 to be near-optimal choices. For **DukeMTMC**, we selected a validation set of 15'000 frames for each camera, pre-trained the model on data from all cameras simultaneously, and performed a final training on the training data for each individual sequence. We used only **DukeMTMC** training data to train the appearance model of [19]. For each **MOT15** pair of training and testing sequence pair, we used the training sequence for validation purposes and the remaining training sequences to learn the network weights. For **MOT17**, we pre-trained our model on **PathTrack**, the appearance model of [19] on on **CUHK03** [31] dataset, and used the **MOT17** training sequences for validation purposes. More details are in supplementary materials.

5.4. Comparative Performance

We compared on **DukeMTMC** and **MOT15** against methods that ignore appearance features because their re-

Method	Use appearance	IDF	MOTA	IDs
OURS-geom	-	27.1	22.2	700
SORT	-	26.8	21.7	1231
RNN	-	17.1	19.0	1490
LP2D	-	—	19.8	1649

Table 2. Results on the **MOT15** dataset. Appearance is never used.

Method	Use appearance	IDF	MOTA	IDs
OURS	+	57.2	44.2	1529
DMAN	+	55.7	48.2	2194
JCC	+	54.5	51.2	1802
MOTDT17	+	52.7	50.9	2474
MHTBLSTM	+	51.9	47.5	2069
EDMT17	+	51.3	50.0	2264
FWT	+	47.6	51.3	2648

Table 3. Results on the **MOT17** dataset. Appearance always used.

sults are reported on these two datasets. For the same reason, we used **DukeMTMC** and **MOT17** to compare against those that exploit appearance. We summarize the results below, reporting **IDF** and **MOTA** tracking metrics, and a number of identity switches (IDs), and provide a much more detailed breakdown in the supplementary material. We present some tracking results in Fig. 3 and Fig. 4 and videos can be found in supplementary material.

Comparing to Algorithms that exploit Appearance. We report our results on **MOT17** in Tab. 3 and on **DukeMTMC** in Tab. 1.

On **DukeMTMC**, our approach performs best both for the Easy and Hard sequences in terms of **IDF**, **MOTA**, and the raw number of identity switches. Furthermore, unlike other top scoring methods that use re-identification networks pre-trained on additional datasets, ours was trained using only the **DukeMTMC** training data.

On **MOT17**, our approach is best both in terms of **IDF** metric, and the number of identity switches. However, it does poorly on **MOTA**. Strikingly, **FWT** does the exact opposite: it yields best **MOTA** and the worst **IDF** on this dataset. We did experiments to investigate this, in Sec. 5.5.

Comparing to Algorithms that ignore Appearance. We report our results on **MOT15** in Tab. 2 and on **DukeMTMC** in Tab. 1. On **MOT15** dataset, method most similar to ours is **RNN**, which also uses an RNN to perform data association. Despite the fact that **RNN** uses external data to pre-train their model, and we use only the **MOT15** training data, our approach is able to outperform it with a large margin. Another interesting comparison is with **SORT**, which performs nearly as good as our approach. However, it can not leverage training data effectively, and to show that we additionally ran this approach on the validation data we used for **DukeMTMC**, where there is much more training data than in **MOT15**. This resulted in a **MOTA** score of 49.9 and **IDF** one of 24.9, whereas our method reaches 70.0 and 74.6 on the same data.

Figure 3. Bounding boxes and the last 6 seconds of tracking, denoted by lines, in dense crowd on **DukeMTMC** dataset.

Figure 4. Bounding boxes and last 6 seconds of tracking, denoted by lines, in two sequences of the **MOT17** dataset.

Remarkably, on **DukeMTMC** dataset, even though we ignored appearance for the purpose of this comparison, our approach also outperforms or rivals some the methods that exploit it [45, 50]. This shows that our training procedure is powerful enough to overcome this serious handicap.

5.5. Analysis

We now analyze results and components of our method.

IDF-MOTA metric disagreement Careful examination of the trajectories on **MOT17** shows that metric disagreement comes from producing many short trajectories that increase the overall number of tracked detections, and therefore **MOTA**, at the cost of assigning many spurious identities, increasing fragmentation, and decreasing **IDF**. This example illustrates why we believe **IDF** to be the more meaningful metric and why we have designed our tracklet scoring function to be a proxy for it.

To further strengthen this claim, we investigated the following toy example. Consider one ground truth trajectory of 100 frames, with detector firing randomly 97% of the time. Combining consecutive detections yields tracklets of varying lengths. We sort them by length and take several longest ones as our tracking result. In Fig. 5, we plot the resulting **MOTA** and **IDF** scores as a function of percentage of taken tracklets. **MOTA** monotonically increases whereas **IDF** monotonically decreases. In other words, adding the very short tracklets that our algorithm rejects improves the **MOTA** score and focusing on the long ones that

Figure 5. Mean and variance of **IDF** and **MOTA** score as a function of the number of tracklets in 100 runs of the toy experiment.

IDF favors degrades MOTA.

We also looked at the results of the top methods on **MOT15** on 'AVG-Towncentre' sequence (those where raw tracking data is available) and found that methods with some of the highest **IDF** scores feature little or no tracks shorter than 5 frames (i.e. 'TDAM'-0, 'RAR15Pub'-0, 'JointMC'-0, 'QuadMOT'-1), while those with the worst **IDF** featuring more than 50 (i.e. 'NOMT'-82, 'DCCRF'-138). This points in the same direction as our toy example, namely that there is **MOTA**/**IDF** trade-off that depends on the length of the retained tracklets.

Computational Complexity. We performed training on a single 2.5Hz CPU, and all other actions (computing **IDF** values for dataset balancing, generating training data, etc.) in parallel on 20 such CPUs. Training data contained at most 1.5×10^7 tracklets (**DukeMTMC** dataset, camera 6), resulting in at most 1.35×10^6 training data points after balancing the dataset. Generating training data took under 6 hours, and training on it achieved best validation scores within 30 epochs, taking under 10 minutes each. Inference runs at about 2 frames per second. However, adding a cutoff on sequence scores in the pruning step of Sec. 3.2.1 speeds up our python implementation to 30fps, at the cost of a very small performance decrease (**IDF** of 71 instead of 74.6).

Ablation study. The last 15'000 frames of training sequences of **DukeMTMC** were used for an ablation study. We varied the three main components of our solution to show their effect on the tracking accuracy: data composition, scoring function, and training procedure. We report the drop in **IDF** when applying such changes. Creating a fixed training set by considering tracklets with at most one identity switch as in [48, 34] decreased performance (-3.9). Pruning hypotheses based on their scores or total count like [56] resulted in either a computational explosion or reduced performance (-20). Computing loss on the pre-

diction of $S(T)$, regressing **IDF** value directly, not regressing bounding box shifts, or using a standard classification loss as in [48] were equally counter-productive (-5.1, -13.2, -2.2, -32.8). Not balancing the training set or not using hard-mining also adversely affected the results (-4.7, -2.5). Selecting the final solution using an Integer Program instead of a greedy algorithm, pre-training model with each type of features separately, or training a deeper network had no significant effect.

Feature groups. We also performed an evaluation of how different features affect the quality of the solution. *Appearance features* improved overall **IDF** from 74.6 to 82.5, with appearance distance feature having the biggest effect. Crowd density feature mostly affected crowded scenarios, where our merging procedure preferred to merge detections that are further apart in time, but more visually similar, compared to less crowded scenarios, where it preferred to merge detections based more on the spatial vicinity. *Social feature* mostly affected appearance-less model, helping to preserve identities by ensuring that detections of the surrounding people are consistent throughout trajectory, improving **IDF** from 67.5 to 74.6. *Probabilistic merging* from Sec. 4.2 was vital to fuse appearance-based and geometry-based features together. Without it, picking only the best candidate resulted in a model that performed merges mostly either based on the appearance information (largely ignoring spatial vicinity), or based on the spatial and motion information (largely ignoring appearance information).

6. Conclusion

We have introduced a training procedure that significantly boosts the performance of sequence models by iteratively building a rich training set. We have also developed a sophisticated model that can regress from tracklets to the **IDF** multiple target tracking metric. We have shown that our approach outperforms state-of-the-art ones on several challenging benchmarks both in scenarios where appearance is used and where it is not. In the second case, we can even come close to what appearance-based method can do without using it. This could prove extremely useful to solve problems in which appearance is hard to use, such as cell or animal tracking [39].

In future work, we will extend our data association procedure to account for more advanced appearance features, such as 2D and 3D pose. We will also look into further reducing the loss-evaluation mismatch by using the actual **IDF**, instead of our proposed **IDF** regressor, which would require the use of reinforcement learning.

Acknowledgement

This work was supported in part by the Swiss National Science Foundation.

References

- [1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *Conference on Computer Vision and Pattern Recognition*, 2014.
- [2] M. Andriluka, S. Roth, and B. Schiele. People-Tracking-By-Detection and People-Detection-By-Tracking. In *Conference on Computer Vision and Pattern Recognition*, June 2008.
- [3] T. Bagautdinov, A. Alahi, F. Fleuret, P. Fua, and S. Savarese. Social Scene Understanding: End-To-End Multi-Person Action Localization and Collective Activity Recognition. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, 2015.
- [5] Y. Bengio, N. Léonard, and A. Courville. Estimating or Propagating Gradients through Stochastic Neurons for Conditional Computation. *ARXIV*, 2013.
- [6] H. BenShitrit, J. Berclaz, F. Fleuret, and P. Fua. Multi-Commodity Network Flow for Tracking Multiple People. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1614–1627, 2014.
- [7] K. Bernardin and R. Stiefelhagen. Evaluating Multiple Object Tracking Performance: the Clear Mot Metrics. *EURASIP Journal on Image and Video Processing*, 2008, 2008.
- [8] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple Online and Realtime Tracking. In *International Conference on Image Processing*, 2016.
- [9] S.S. Blackman. *Multiple-Target Tracking with Radar Applications*. Artech House, 1986.
- [10] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret. The Wildtrack Multi-Camera Person Dataset. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [11] J. Chen, H. Sheng, Y. Zhang, and Z. Xiong. Enhancing Detection Model for Multiple Hypothesis Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [12] L. Chen, H. Ai, C. Shang, Z. Zhuang, and B. Bai. Online Multi-Object Tracking with Convolutional Neural Networks. In *International Conference on Image Processing*, 2017.
- [13] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu. Online Multi-Object Tracking Using Cnn-Based Single Object Tracker with Spatial-Temporal Attention Mechanism. In *International Conference on Computer Vision*, 2017.
- [14] H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 2009.
- [15] C. Dicle, O. I Camps, and M. Sznai. The Way They Move: Tracking Multiple Targets with Similar Appearance. In *International Conference on Computer Vision*, 2013.
- [16] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent Network Models for Human Dynamics. In *International Conference on Computer Vision*, 2015.
- [17] D. Held, S. Thrun, and S. Savarese. Learning to Track at 100 Fps with Deep Regression Networks. In *European Conference on Computer Vision*, 2016.
- [18] R. Henschel, L. Leal-Taixé, D. Cremers, and B. Rosenhahn. Fusion of Head and Full-Body Detectors for Multi-Object Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [19] A. Hermans, L. Beyer, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737*, 2017.
- [20] M. Hu, S. Ali, and M. Shah. Detecting Global Motion Patterns in Complex Videos. In *ICPR*, 2008.
- [21] J. Supancic III and D. Ramanan. Tracking as Online Decision-Making: Learning a Policy from Streaming Videos with Reinforcement Learning. In *International Conference on Computer Vision*, 2017.
- [22] U. Iqbal, A. Milan, and J. Gall. Posetrack: Joint Multi-Person Pose Estimation and Tracking. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [23] E. Jang, S. Gu, and B. Poole. Categorical Reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.
- [24] M. Keuper, S. Tang, B. Andres, T. Brox, and B. Schiele. Motion Segmentation and Multiple Object Tracking by Correlation Co-Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.
- [25] C. Kim, F. Li, A. Ciptadi, and J. Rehg. Multiple Hypothesis Tracking Revisited. In *International Conference on Computer Vision*, 2015.
- [26] C. Kim, F. Li, and J. M. Rehg. Multi-Object Tracking with Neural Gating Using Bilinear LSTM. In *European Conference on Computer Vision*, 2018.
- [27] Y. J. Koh and C.-S. Kim. CDTs: Collaborative Detection, Tracking, and Segmentation for Online Multiple Object Segmentation in Videos. In *International Conference on Computer Vision*, 2017.
- [28] L. Leal-taixé, C. Canton-ferrer, and K. Schindler. Learning by Tracking: Siamese CNN for Robust Target Association. In *Conference on Computer Vision and Pattern Recognition*, 2016.
- [29] L. Leal-taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a Benchmark for Multi-Target Tracking. In *ARXIV*, 2015.
- [30] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. HS Torr, and M. Chandraker. Desire: Distant Future Prediction in Dynamic Scenes with Interacting Agents. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [31] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep Filter Pairing Neural Network for Person Re-Identification. In *Conference on Computer Vision and Pattern Recognition*, pages 152–159, 2014.
- [32] Z. Lin, H. Zheng, B. Ke, and L. Chen. Online Multi-Object Tracking Based on Hierarchical Association and Sparse Representation. In *International Conference on Image Processing*, 2017.
- [33] C. Long, A. Haizhou, Z. Zijie, and S. Chong. Real-Time Multiple People Tracking with Deeply Learned Candidate

- Selection and Person Re-Identification. In *International Conference on Multimedia and Expo*, 2018.
- [34] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie. Trajectory Factory: Tracklet Cleaving and Re-Connection by Deep Siamese Bi-GRU for Multiple Object Tracking. In *ICME*, 2018.
- [35] W.-C. Ma, D.-A. Huang, N. Lee, and K. M. Kitani. Forecasting Interactive Dynamics of Pedestrians with Fictitious Play. In *Conference on Computer Vision and Pattern Recognition*, 2017.
- [36] A. Maksai, X. Wang, F. Fleuret, and P. Fua. Globally Consistent Multi-People Tracking Using Motion Patterns. In *International Conference on Computer Vision*, 2017.
- [37] S. Manen, M. Gygli, D. Dai, and L. Van Gool. Pathtrack: Fast Trajectory Annotation with Path Supervision. In *International Conference on Computer Vision*, 2017.
- [38] A. Milan, L. Leal-taixe, I. Reid, S. Roth, and K. Schindler. Mot16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*, 2016.
- [39] A. Milan, S.H. Rezatofighi, A. Dick, I. Reid, and K. Schindler. Online Multi-Target Tracking Using Recurrent Neural Networks. In *AAAI*, 2017.
- [40] S. Murray. Real-Time Multiple Object Tracking-A Study on the Importance of Speed. *arXiv preprint arXiv:1709.03572*, 2017.
- [41] P. Ondruska, J. Dequaire, D.Z. Wang, and I. Posner. End-To-End Tracking and Semantic Segmentation using Recurrent Neural Networks. In *Workshop on limits and potentials of Deep Learning in Robotics*, 2016.
- [42] P. Ondruska and I. Posner. Deep Tracking: Seeing Beyond Seeing Using Recurrent Neural Networks. In *AAAI*, 2016.
- [43] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking. In *International Conference on Computer Vision*, 2009.
- [44] M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence Level Training with Recurrent Neural Networks. In *ICLR*, 2015.
- [45] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. In *European Conference on Computer Vision*, 2016.
- [46] E. Ristani and C. Tomasi. Features for Multi-Target Multi-Camera Tracking and Re-Identification. In *Conference on Computer Vision and Pattern Recognition*, 2018.
- [47] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.
- [48] A. Sadeghian, A. Alahi, and S. Savarese. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. In *International Conference on Computer Vision*, 2017.
- [49] S. Wiseman and A. M. Rush. Sequence-To-Sequence Learning as Beam-Search Optimization. In *Conference on Empirical Methods in Natural Language Processing*, 2016.
- [50] Y. T. Tesfaye, E. Zemene, A. Prati, M. Pelillo, and M. Shah. Multi-Target Tracking in Multiple Non-Overlapping Cameras Using Constrained Dominant Sets. *arXiv preprint arXiv:1706.06196*, 2017.
- [51] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and Tell: A Neural Image Caption Generator. In *Conference on Computer Vision and Pattern Recognition*, 2015.
- [52] L.R. Williams and D.W. Jacobs. Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Saliency. *Neural Computation*, 9(4):837–858, 1997.
- [53] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation. *arXiv Preprint*, 2016.
- [54] J. Xiang, G. Zhang, J. Hou, N. Sang, and R. Huang. Multiple Target Tracking by Learning Feature Representation and Distance Metric Jointly. *arXiv Preprint*, 2018.
- [55] Y. Xiang, A. Alahi, and S. Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. In *Conference on Computer Vision and Pattern Recognition*, 2015.
- [56] K. Yoon, Y. Song, and M. Jeon. Multiple Hypothesis Tracking Algorithm for Multi-Target Multi-Camera Tracking with Disjoint Views. *IET Image Processing*, 2018.
- [57] M. Zhai, M. J. Roshtkhari, and G. Mori. Deep Learning of Appearance Models for Online Object Tracking. *arXiv preprint arXiv:1607.02568*, 2016.
- [58] Z. Zhang, J. Wu, Ix. Zhang, and C. Zhang. Multi-Target, Multi-Camera Tracking by Hierarchical Clustering: Recent Progress on DukeMTMC Project. *arXiv preprint arXiv:1712.09531*, 2017.
- [59] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M. Yang. Online Multi-Object Tracking with Dual Matching Attention Networks. In *European Conference on Computer Vision*, 2018.