Unsupervised Multi-Object Tracking with Dual-Tracker Consistency

Favyen Bastani, Songtao He, Sam Madden

Abstract

In this paper, we develop an unsupervised approach for multi-object tracking under the detect-to-track framework: we assume that an object detector trained on image-level bounding box annotations is available, but do not use any expensive video-level bounding box and track annotations. In our approach, we propose training an RNN tracking model on unlabeled video using self-supervised learning. However, learning signals proposed in prior work on self-supervised single-object tracking, such as color propagation and cycle-consistency, are not effective for training RNN trackers: they yield degenerate models that, for instance, always match a new detection to the track with the closest initial detection. We propose a novel self-supervised learning method that we call dual-tracker consistency: we construct two distinct inputs for one video segment, where each input is a variation of the video segment where different information has been hidden; we then apply two instances of a tracker independently on each input, and back-propagate a loss that enforces consistency between the two outputs. We evaluate our method on the MOT17 benchmark, and find that, when training only on a corpus of unlabeled video, our fully unsupervised method is competitive with four recent supervised state-of-the-art trackers that train on expensive video-level annotations.

1 Introduction

Multi-object trackers identify all instances of a particular object type in video, and track each instance through the segment of video in which it is visible in the camera frame. Annotating training data for multi-object tracking is tedious and costly; for example, annotation of pedestrian tracks in just six minutes of video in the training set of the MOT15 Challenge [Leal-Taixé *et al.*, 2015] requires an estimated 22 hours [Manen *et al.*, 2017] of human labeling time using LabelMe [Yuen *et al.*, 2009]. While unsupervised, heuristic detect-to-track methods [Bewley *et al.*, 2016; Bochinski *et al.*, 2018] have been proposed that group detections into tracks by estimating motion using a combination of spatial and visual cues, these methods suffer low-accuracy in scenarios with frequent occlusion where heuristics are insufficient.

Recent work has proposed applying self-supervised learning for training *single*-object tracking models on unlabeled video [Vondrick *et al.*, 2018; Wang *et al.*, 2019]. These approaches train a model to propagate instance labels from a reference frame through the rest of a video segment. In contrast to work on self-supervised representation learning from video, these fully unsupervised approaches do not require fine-tuning to apply the model for single-object tracking.

However, a significant limitation in prior work is that the model independently compares pairs of frames at a time. In multi-object tracking, a key challenge is robustly re-localizing tracks across potentially long occlusions, especially when an object instance is occluded by other instances of the same object type. Pairwise frame comparisons are thus insufficient for high-accuracy multi-object tracking; instead, learning recurrent features that encode the history of a track is crucial for enabling robust re-localization. However, extending prior work to learn RNN parameters is challenging. For example, Wang et al. [Wang *et al.*, 2019] propose training using forward-backward consistency: from a patch in an initial frame, after tracking forwards through video and then backwards to return to the initial frame, the final patch should align with the original patch. Training an RNN in this way would be ineffective as the RNN could simply memorize the features of the original patch.

To address this challenge, we propose a novel self-supervised learning method, *dual-tracker consistency*. At a high-level, we derive a learning signal from unlabeled video by applying two instances of a tracker model (where the instances share parameters) through two distinct input variations extracted from the same video segment, and training the tracker to produce consistent tracking outputs across both inputs. We propose two alternative input-hiding schemes for computing the input variations: occlusion-based hiding and visual-spatial hiding. Occlusion-based hiding eliminates detections in random intermediate subsequences of frames to simulate occlusion incidents; it constructs two inputs by eliminating different subsequences. Visual-spatial hiding applies the tracker once when only observing spatial inputs (bounding box coordinates), and once when only observing visual inputs (pixel values inside detection boxes). Then, on each training step, we sample a segment of video, compute two in-

put variations under the chosen input-hiding scheme, apply a tracker instance on each input, and back-propagate a learning signal that measures the consistency between tracks computed across the two inputs. To attain high consistency, the model must accurately group detections that correspond to the same object: if the model were to instead arbitrarily group detections into tracks, then variations in the inputs would lead the two tracker instances to produce inconsistent outputs.

We use a detect-to-track framework, i.e., we first compute object detections in each frame, and then group detections into tracks across frames. We assume that a robust detector is available—image-level bounding boxes are much cheaper to annotate than tracks, and are required even in unsupervised detect-to-track methods like SORT [Bewley *et al.*, 2016]. Then, under the detect-to-track framework, the challenge is to automatically learn to determine which detections correspond to the same object across different frames, and to identify when an object enters or leaves the camera frame.

To implement dual-tracker consistency, we adapt a now standard RNN model and tracker architecture from prior work [Kim *et al.*, 2018]: the tracker processes each frame in sequence by matching detections in the current frame with tracks computed up to the previous frame. In prior work, this model is trained under a supervised procedure: they sample a video segment $\langle I_0, \ldots, I_n \rangle$ and a track t in that segment, and apply the tracker on t over the video segment. On each frame I_j, the RNN outputs a probability distribution indicating the likelihood that the prefix of t up to I_j matches with each detection in I_j. Prior work back-propagates the label (i.e., the correct detection of t in I_j) under cross entropy loss.

In contrast, under our method, on each training iteration, we propose to sample a segment $\langle I_0, \ldots, I_n \rangle$ from a corpus of unlabeled video, and apply the RNN model to compute a transition matrix that specifies the probability that each detection in I_0 (rows) matches with each detection in I_n (columns). Then, when applying two instances of the tracker on two input variations extracted from the segment, we obtain two transition matrices (one from each instance). We compute the dot-product similarity to measure the consistency between these matrices, and back-propagate this as a loss function.

We evaluate our approach on the MOT17 benchmark against 8 state-of-the-art methods, including both unsupervised and supervised methods. We train our tracker model using dual-tracker consistency over a corpus of unlabeled video, which can be cheaply obtained. Like other unsupervised methods, we use an object detector trained on image-level bounding box annotations in MOT17Det, but do not use any expensive video-level annotations. We find that our approach improves both IDF1 and MOTA accuracy over the unsupervised baselines by 4% to 10%. Moreover, remarkably, our fully unsupervised approach is competitive with four of the five supervised methods we compared, even though these methods train on expensive video-level bounding box and track annotations.

2 Related Work

Self-supervised learning over video has been studied extensively in many contexts. Most work focuses on learning

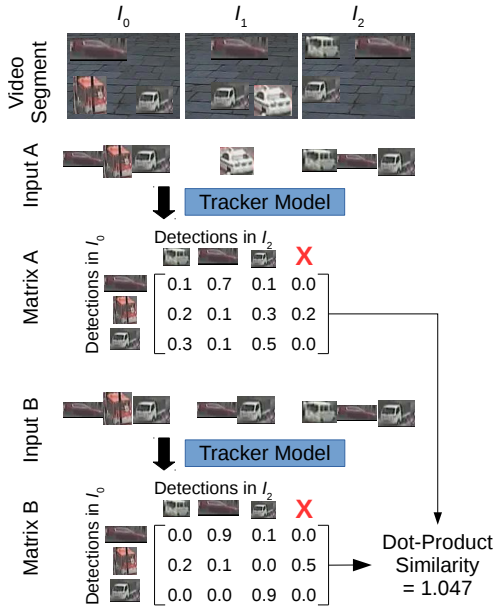

Figure 1: Overview of dual-tracker consistency. An input-hiding scheme produces two input variations, A and B. We apply the tracker model on each input to derive two transition matrices, and back-propagate a similarity score between the matrices that encourages the model to produce consistent outputs across both inputs.

representations of video that can be applied through fine-tuning for tasks such as activity recognition, image classification, and object detection [Doersch *et al.*, 2015; Doersch and Zisserman, 2017; Srivastava *et al.*, 2015]. More closely related to our work, several recent approaches have proposed leveraging widely available unlabeled video to directly train *single*-object tracking models, without needing fine-tuning. Vondrick et al. [Vondrick *et al.*, 2018] train a model to colorize gray-scale video by propagating colors from a colored reference frame. The model is then applied to track objects at inference time by propagating instance IDs instead of colors. Wang et al. [Wang *et al.*, 2019] train a model to capture correspondence by applying a cycle-consistent loss: from a patch in an initial frame, after tracking forwards through video and then backwards to return to the initial frame, the final patch should align with the original patch.

Our work is also related to unsupervised, heuristic detect-to-track multi-object tracking methods. SORT [Bewley *et al.*, 2016] proposes modeling the motion of a track based only on spatial cues using a Kalman filter. To assign detections in the next frame to tracks, SORT formulates a bipartite matching problem where the cost is computed from the intersection-over-union score between the detection bounding box and the estimated track position. V-IOU [Bochinski *et al.*, 2018] incorporates visual cues into this framework, leveraging optical flows computed by comparing pixel values between pairs of consecutive frames to update the position of a track through segments where the detector fails to localize the track.

Several fully supervised multi-object tracking methods have recently been proposed [Liu *et al.*, 2020; Zhou *et al.*, 2020; Feng *et al.*, 2019]. Although these methods yield high-

accuracy, they require video-level bounding box and track annotations that are expensive to hand-label, and thus are costly to extend to new types of video. Tracktor++ [Bergmann *et al.*, 2019] applies the regression network in an object detector to localize a track in the next frame. MHT-BLSTM [Kim *et al.*, 2018] applies a deep multiple hypothesis tracking approach, incorporating a bilinear LSTM to store long-term visual and spatial features and gate the track hypotheses. FAMNet [Chu and Ling, 2019] jointly learns to extract features, estimate affinity, and assign track IDs in one network.

3 Dual-Tracker Consistency

In our noval dual-tracker consistency method, we derive an learning signal for training an RNN tracker model through a three-step process. We assume that a corpus of unlabeled video is available, along with detections of the object category of interest computed automatically by an object detector in each video frame. First, during training, we repeatedly randomly sample a video segment $\langle I_0, \ldots, I_n \rangle$, where each I_k is a frame. Let D_k by the detections automatically computed in I_k, and let $d_i^k = (im, x, y, w, h)$ be a detection in D_k, where (x, y, w, h) are the 4D spatial coordinates (center point and lengths) of the detection bounding box, and im is the window of I_k corresponding to that box. We apply an input-hiding scheme to select two input variations $A(D), B(D)$ for the video segment, where each variation is a modified sequence of detections in the frames, $A(D) = \langle D_0^A, \ldots, D_n^A \rangle, B(D) = \langle D_0^B, \ldots, D_n^B \rangle$. For example, some detections may be removed entirely, while others may be partially hidden. Second, we apply two instances of the tracker model through each input variation to derive two probabilistic tracking outputs, represented as transition matrices. Third, we compare the transition matrices with dot-product similarity to update the RNN parameters.

Figure 1 summarizes our approach.

Below, we first introduce the model architecture that we adapt from prior work in Section 3.1. We then detail our novel training procedure, including the computation of the transition matrices and dot-product loss, in Section 3.2. Finally, we propose two input-hiding schemes for selecting the input variations required by our approach in Section 4.

3.1 Tracker Model

We adopt a tracker model that is similar to prior work [Kim *et al.*, 2018]. We summarize the architecture in Figure 2. Given a video segment $\langle I_0, \ldots, I_n \rangle$, and sets of detections $D_k = \{d_1^k, \ldots, d_{m_k}^k\}$ detected in each frame I_k, to initialize the tracking process, we create a track $t_i = \langle d_i^0 \rangle$ for each detection d_i^0 in the first video frame I_0. On each subsequent frame I_k, the model outputs a probability $p_{i,j}$ that each track t_i corresponds to each detection $d_j^k \in D_k$. At inference time, we formulate the problem of matching tracks with detections in I_k as a bipartite matching problem, where the cost of matching t_i with d_j^k is $1 - p_{i,j}$. We solve this problem and compute a minimum-cost matching using the Hungarian algorithm; for each pair (t_i, d_j^k) in the matching, we append d_j^k to t_i. For each detection in I_k that no track matches to, we create a new track for that detection.

The model consists of a CNN, RNN, and matcher network. Together, these components score the likelihood that the ith track, $t_i = \langle d_1, \ldots, d_m \rangle$, matches with the jth detection in I_k, d_j^k. We first apply the CNN to derive detection-level features. Given a detection $d = (im, x, y, w, h)$, the CNN inputs im resized to 64×64, and consists of 6 strided convolutional layers, with ReLU activation in the first 5 layers and linear activation in the last layer. It outputs a 64-vector, which we concatenate with the 4D spatial coordinates to derive a 68-vector detection representation $f(d)$. Then, we compute track-level features $f(t_i)$ by applying the RNN (an LSTM with 64 hidden states) over the sequence of detection-level features of detections in the track, $\langle f(d_1), \ldots, f(d_m) \rangle$. We use the output of the RNN on the last timestep as the track-level features $f(t_i)$. Finally, we apply a matching network to score the likelihood that t_i matches d_j^k. The matching network inputs the concatenation of $f(t_i)$ and $f(d_j^k)$, applies four fully-connected layers, and outputs a match score.

3.2 Training Procedure

We develop a novel self-supervised learning method for training the model parameters on unlabeled video. During training, we repeatedly sample segments of up to 16 consecutive video frames $\langle I_0, \ldots, I_n \rangle$. We apply one of two input-hiding schemes, which we will detail in the following section, to extract two distinct input variations $A(D)$ and $B(D)$ from a sampled video segment, where each input is a sequence of detections. We then apply instances of the tracker model on each variation, where the instances share the same model parameters. Dual-tracker consistency trains the model by enforcing it to produce similar outputs on both inputs. To represent tracker outputs, we compute a transition matrix $M^{(0,n)}$, where $M_{i,j}^{(0,n)}$ is the probability that the track t_i matches d_j^n. When applying the model over video segments during training, we update tracks with new detections based on the scores output by the model on intermediate frames, but do not create additional tracks on frames after I_0; thus, each track t_i corresponds directly to a detection d_i^0 in I_0 (i.e., $t_i = \langle d_i^0, \ldots \rangle$). Then, applying two instances of the tracker yields two transition matrices $A^{(0,n)}$ and $B^{(0,n)}$, where both matrices have $|D_0|$ rows and $|D_n| + 1$ columns (the extra column represents tracks that are no longer visible). We train the model (CNN, RNN, and matching network) end-to-end to maximize the dot-product similarity between these matrices.

Below, we detail our method to compute transition matrices, and discuss dot-product similarity loss.

Transition Matrix. We propose computing a transition matrix $M^{(0,k)}$ on each frame I_k to represent the tracker outputs, where $M_{i,j}^{(0,k)}$ is the probability that the track t_i matches the detection d_j^k. On intermediate frames, we apply the Hungarian method on $M^{(0,k)}$ to match detections in D_k with tracks, updating each track with the matched detection (if any). On the last frame I_n, we use the $M^{(0,n)}$ matrix produced under different inputs (denoted $A^{(0,n)}$ and $B^{(0,n)}$) to compute and back-propagate a consistency score. Because we do not create new tracks after I_0 during training, $M_{i,j}^{(0,n)}$ is the like-

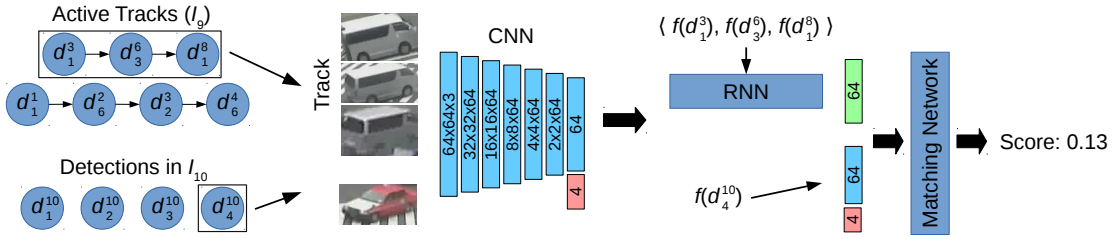

Figure 2: Tracker model architecture. The model scores the likelihood that each detection in the current frame matches with each track.

lihood that d_i^0 and d_j^n match (since t_i begins with d_i^0).

We first construct a score matrix $S^{(0,k)}$, by computing $S_{i,j}^{(0,k)}$ as the score (any real number) output by the tracker model given the track t_i and detection d_j^k. We then transform the score matrix into a probability matrix to derive $M^{(0,k)}$. We could simply compute $M^{(0,k)}$ by taking softmax along rows in $S^{(0,k)}$. However, computing the transition matrix in this way would allow the tracker to cheat and maximize similarity between $A^{(0,n)}$ and $B^{(0,n)}$ by simply matching all detections in I_0 to a single detection $d_j^n \in D_n$. Indeed, we find that in practice this yields degenerate models.

Thus, instead, we compute $M^{(0,k),\text{row}}$ and $M^{(0,k),\text{col}}$ by applying softmax along rows and columns, respectively, and compute $M^{(0,k)} = \min(M^{(0,k),\text{row}}, M^{(0,k),\text{col}})$:

$$M_{i,j}^{\text{row}} = \frac{\exp(S_{i,j})}{\sum_k \exp(S_{i,k})} \qquad M_{i,j}^{\text{col}} = \frac{\exp(S_{i,j})}{\sum_k \exp(S_{k,j})} \tag{1}$$

$$M_{i,j} = \min(M_{i,j}^{\text{row}}, M_{i,j}^{\text{col}}) \tag{2}$$

This produces a transition matrix $M^{(0,k)}$ that is almost doubly stochastic: rows and columns sum to at most 1, but not necessarily exactly 1. The operation ensures that the model must match each detection in I_0 to unique detections in I_n to maximize the consistency score between $A^{(0,n)}$ and $B^{(0,n)}$: if two detections in I_0 are matched to the same detection in I_n, then the columnar softmax would reduce those probabilities in the corresponding matrix to at most 0.5, thereby reducing any dot-products involving those rows.

So far, $M^{(0,k)}$ does not indicate the likelihood that a track fails to match to any detection. Determining when a track is not visible in a frame (it may be transiently occluded, or may have left the camera frame) is a key challenge in detect-to-track approaches. To represent this case in our transition matrix, we append an additional column to the score matrix $S^{(0,k)}$ for the "absent" case, where a track does not appear as a detection in I_k. The absent column is populated with a constant score $S_{i,\text{absent}} = s_{\text{absent}}$; the parameter s_{absent} is learned during training. Thus, if the ith track does not appear in a frame, the matching model should output low scores in $S^{(0,k)}$ when matching that track to detections in I_k, which will yield a large probability $M_{i,\text{absent}}^{(0,k)}$.

Dot-Product Similarity. We train the RNN tracker by computing two transition matrices $A^{(0,n)}$ and $B^{(0,n)}$ over different input variations, and then back-propagating a loss that measures the inconsistency between the matrices. In particular, we use the dot-product to measure the similarity of corresponding rows in the matrices. We define the loss as:

$$L = -\sum_i \log \sum_j A_{i,j}^{(0,n)} B_{i,j}^{(0,n)}$$

Here, L is computed by taking the logarithm of the dot product of corresponding rows in $A^{(0,n)}$ and $B^{(0,n)}$, averaged across rows. Note that this is equivalent to the cross-entropy loss between the diagonal matrix and the matrix product of $A^{(0,n)}$ and the transpose of $B^{(0,n)}$.

This loss function has several desirable properties. First, the dot-product is maximized when the matrices computed based on different inputs are most similar. This pushes the matching model to learn reasonable visual and spatial tracking constraints, because arbitrarily matching tracks to detections will lead to dissimilarity. Second, the dot-product pushes each matrix to be almost doubly stochastic rather than leaving some rows and columns summing to much less than 1. The model can only produce doubly stochastic $A^{(0,n)}$ and $B^{(0,n)}$ matrices by finding unique detections in I_n for each detection in I_0. Because the matching model is constrained to observing one track and one detection at a time, it must learn to match similar detections.

4 Input-Hiding Schemes

In this section, we detail two alternative input-hiding schemes for selecting the two input variations, denoted $A(D) = \langle D_0^A, \ldots, D_n^A \rangle$ and $B(D) = \langle D_0^B, \ldots, D_n^B \rangle$.

4.1 Occlusion-based Hiding

At a high level, occlusion-based hiding produces the variations $A(D)$ and $B(D)$ by simulating random occlusion incidents where all detections in occluded frames are eliminated from the input, i.e., if I_k is occluded for $A(D)$, then D_k^A is empty. We only occlude intermediate frames I_k, $0 < k < n$, so that the transition matrices still compare detections in I_0 with those in I_n. When processing an occluded I_k, the tracker is forced to match all tracks to the absent column in that frame, and re-localize the tracks after the occlusion.

We first introduce two schemes that do not work in isolation, and then show that we can combine these schemes to produce input variations that result in effective training.

Only-Occlusion. For each training sequence $\langle I_0, \ldots, I_n \rangle$, Only-Occlusion randomly selects four indexes $0 < k_1 \leq$

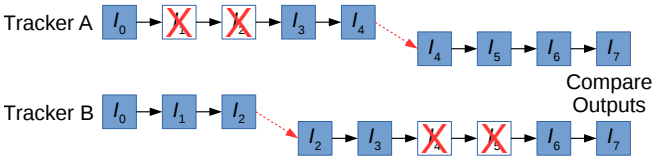

Figure 3: Occlusion-based hiding produces input variations with different subsequences of occluded frames where all detections are hidden from the tracker. It also independently applies the tracker before and after a hand-off frame (I_4 and I_2), and merges the outputs through the matrix product.

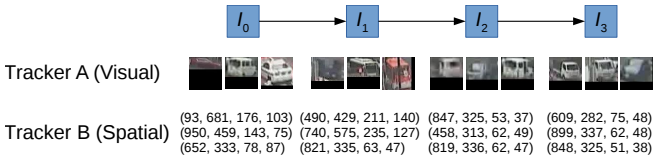

Figure 4: Visual-spatial hiding. One variation includes only visual inputs, and the other includes only spatial inputs.

$k_2 < k_3 \leq k_4 < n$ to construct two disjoint frame subsequences $\langle I_{k_1}, \ldots, I_{k_2} \rangle$ and $\langle I_{k_3}, \ldots, I_{k_4} \rangle$. In $A(D)$, we occlude each frame I_k such that $k_1 \leq k \leq k_2$, and in $B(D)$, we occlude I_k if $k_3 \leq k \leq k_4$.

When training under Only-Occlusion, the tracker is forced to leverage track features computed through the RNN to relocalize tracks after each simulated occlusion ends. Learning to merely compare detection features across consecutive frames would yield low accuracy since features in occluded frames are not observed. Furthermore, because one tracker observes the detections and the other tracker does not, the model must make similar tracking decisions when relocalizing across occluded frames as it does when observing detections in each frame.

However, in practice, Only-Occlusion produces a model that simply memorizes the detections in I_0, and computes $A^{(0,n)}$ and $B^{(0,n)}$ by comparing the detections in I_n against memorized detections. This strategy yields high similarity because it is unaffected by simulated occlusions in intermediate frames. Thus, we must prevent the propagation of features directly from I_0 to I_n to make this scheme effective.

RNN Hand-off. RNN Hand-off prevents simple memorization by cutting off the propagation of RNN features through the application of two separate RNN executions. We select two indexes $0 < k_5, k_6 < n$. Instead of computing $A^{(0,n)}$ directly, we first apply the tracker on the frame sequence $\langle I_0, \ldots, I_{k_5} \rangle$ to derive a transition matrix $A^{(0,k_5)}$ that matches detections in I_0 with detections in I_{k_5}. We then independently apply the tracker on $\langle I_{k_5}, \ldots, I_n \rangle$ to derive another matrix $A^{(k_5,n)}$ that matches detections in I_{k_5} with detections in I_n. We combine these matrices through the matrix product to compute $A^{(0,n)}$: we compute $A^{(0,n)} = A^{(0,k_5)} A^{(k_5,n)}$. Similarly, we compute $B^{(0,n)} = B^{(0,k_6)} B^{(k_6,n)}$.

This scheme forces the tracker to find the same unique detection in I_{k_5} (and I_{k_6}) for two detections of the same object in I_0 and I_n in order to maximize similarity between the matrix products. On the other hand, though, a tracker that learns to match tracks to detections by comparing only the detection features in consecutive frames will exhibit high similarity between $A^{(0,n)}$ and $B^{(0,n)}$ under this scheme.

Combined Scheme. Only-Occlusion and RNN Hand-off have opposite advantages and drawbacks. Thus, we combine these in our occlusion-based hiding scheme. We first select the two sequences for simulated occlusion, $\langle I_{k_1}, \ldots, I_{k_2} \rangle$ and $\langle I_{k_3}, \ldots, I_{k_4} \rangle$. Then, we randomly pick k_5 and k_6 such that $k_3 \leq k_5 \leq k_4$ and $k_1 \leq k_6 \leq k_2$, i.e., the hand-off for one tracker occurs when the other tracker observes a simulated occlusion. We summarize the scheme in Figure 3.

Under this scheme, neither memorizing features in I_0 nor comparing detections solely in a pairwise frame-by-frame manner is an effective tracking strategy. Instead, the tracker must learn to leverage RNN features for re-localizing across simulated occlusion, while still ensuring the tracking decisions reflect intermediate outputs.

4.2 Visual-Spatial Hiding

Under visual-spatial hiding, we apply one tracker instance that observes only visual features and one tracker instance that observes only spatial features: in $A(D)$, we set $x = 0, y = 0, w = 0, h = 0$ for all detections, and in $B(D)$, we set $im = 0$. By encouraging similarity in the tracking outputs, each tracker learns to leverage its input features and track objects as robustly as possible.

To prevent the visual tracker from examining the background to extract an approximation of the spatial features, we do not use a recurrent unit for the visual tracker; instead, its matching network inputs visual features for detections in I_0 and for detections in I_n, and scores each pair of detections in $A^{(0,n)}$ without observing intermediate frames. We compute $B^{(0,n)}$ through the spatial tracker, which inputs 4D spatial coordinates for each detection, and processes frames with the matching network and recurrent unit. Figure 4 illustrates the training procedure.

Since the training process eliminates recurrent features for the visual tracker instance, we make corresponding adjustments to the inference process. First, during inference, we compute $M^{(0,k)}$ for an input video sequence $\langle I_0, \ldots, I_k \rangle$ as the minimum of the visual tracker output $A^{(0,k)}$ and the spatial tracker output $B^{(0,k)}$. Additionally, to compute scores in $A^{(0,k)}$, for each track, we compute match scores between 5 randomly selected images associated with the track and each detection in I_k, and average these scores.

5 Evaluation

We compare our method and six baselines on the MOT17 challenge [Milan *et al.*, 2016].

Baselines. We compare with the two unsupervised methods (SORT [Bewley *et al.*, 2016] and V-IOU [Bochinski *et al.*, 2018]), one semi-supervised method (Tracktor++ [Bergmann *et al.*, 2019]), and five fully supervised methods introduced in Section 2. Like our approach, SORT and V-IOU require image-level bounding box annotations (for training an object detector), but do not train on video-level bounding box

Method	IDF1	MOTA
Occlusion (ours)	52.4	56.7
Visual-Spatial (ours)	**57.3**	**60.2**
Spatial-Only (ours)	56.5	57.8
SORT	53.4	56.0
V-IOU	47.4	56.3

Table 1: Ablation study and comparison with unsupervised baselines on the MOT17 training set.

	Method	IDF1	MOTA
Unsupervised Methods	Visual-Spatial (ours)	**58.3**	**56.8**
	SORT [Bewley *et al.*, 2016]	39.8	43.1
	IOU [Bochinski *et al.*, 2017]	39.4	45.5
Semi-Supervised	Tracktor [Bergmann *et al.*, 2019]	52.3	53.5
Supervised Methods	MHT-BLSTM [Kim *et al.*, 2018]	51.9	47.5
	FAMNet [Chu and Ling, 2019]	48.7	52.0
	LSST [Feng *et al.*, 2019]	62.3	54.7
	GSM [Liu *et al.*, 2020]	57.8	56.4
	CenterTrack [Zhou *et al.*, 2020]	59.6	61.5

Table 2: Performance on the MOT17 test set.

and track annotations. The fully supervised methods train on video-level annotations; Tracktor++ also requires video-level annotations for training a re-identification network.

Dataset. The MOT17 dataset [Milan *et al.*, 2016] consists of 14 video sequences of pedestrians in a wide range of contexts, including a moving camera inside a shopping mall and a fixed, elevated view of an outdoor plaza. The dataset is split into 7 training sequences and 7 test sequences; each split includes approximately 11 minutes of video.

Training. The semi-supervised and fully supervised baselines train on video-level bounding box and track annotations provided by MOT17. In contrast, our method trains only on a corpus of unlabeled video, and detections automatically inferred in the video by an object detector. Because video-level annotations are expensive to label, our method requires substantially less annotation time, and thus can greatly reduce the effort needed to apply multi-object tracking on new datasets.

We collect unlabeled video from two sources: we use five hours of video from seven YouTube walking tours, and all train and test sequences from the PathTrack dataset [Manen *et al.*, 2017] (we do not use the PathTrack ground truth annotations). We train our tracker model on an NVIDIA Tesla V100 GPU; training time varies between 24 and 48 hours depending on the input-hiding scheme. During training, we select sequence lengths n between 4 and 16, and apply stochastic gradient descent one sequence at a time. We apply the Adam optimizer with learning rate 0.0001.

Metrics. We use the Multi-Object Tracking Accuracy (MOTA) [Milan *et al.*, 2016] and ID F1 Score (IDF1) [Ristani *et al.*, 2016] metrics. Broadly, MOTA and IDF1 measure the accuracy of inferred tracks against ground truth tracks, and penalize both when an inferred track contains a detection that doesn't match to some ground truth detection (or vice versa), and when a ground truth track is split into two or more inferred tracks (or vice versa).

Ablation Study and Unsupervised Baselines. We first compare occlusion-based hiding, visual-spatial hiding, and the unsupervised baselines on the MOT17 training set in Table 1. None of these methods train on MOT17 training annotations. Visual-spatial hiding yields high performance on IDF1 and MOTA, improving over SORT and V-IOU by 4% on both metrics. On the other hand, occlusion-based hiding performs poorly on MOT17; objects are often visible in the video for only a short duration, making it challenging to learn to relocalize objects over simulated occlusion since the simulated occlusion must then also be short.

Under Spatial-Only, we show results for visual-spatial hiding when inputting only the spatial bounding box coordinates of detections during inference (no image features). Even when inputting only spatial features, our method improves accuracy over SORT and V-IOU, suggesting that the model learns spatial patterns that heuristics used in SORT and V-IOU do not effectively capture.

Quantitative Results. Table 2 shows results on the MOT17 test set[1]; metrics are automatically computed by the challenge website. Per the challenge policy, we only submit the best method, and thus show Visual-Spatial performance.

Our approach again outperforms the unsupervised and semi-supervised baselines. Moreover, its performance is competitive with four recently proposed supervised tracking methods: MHT-BLSTM [Kim *et al.*, 2018], FAMNet [Chu and Ling, 2019], LSST [Feng *et al.*, 2019], and GSM [Liu *et al.*, 2020]: comparing against LSST, our approach improves in MOTA but yields lower IDF1. Thus, we have shown that by leveraging unlabeled video, our unsupervised approach performs competitively with four recent fully supervised MOT methods that train on expensive video-level annotations. Nevertheless, CenterTrack [Zhou *et al.*, 2020] yields slightly higher accuracy on both metrics.

Qualitative Results. We show qualitative results in the supplementary video.

6 Conclusion

In this paper, we have shown that a robust, fully unsupervised multi-object tracker can be trained through a novel self-supervised learning signal, dual-tracker consistency, that enforces consistency in the tracking outputs across different input variations of one video sequence. Despite training only on unlabeled video, our approach is competitive on MOT17 with four recent supervised trackers, which train on expensive video-level bounding box and track annotations.

[1]These results are taken from https://motchallenge.net/results/MOT17/, where our method is denoted UNS20regress. Baselines are denoted SORT17, IOU17, Tracktor++, MHT_bLSTM, FAMNet, LSST17, GSM_Tracktor, and CTTrackPub.

References

[Bergmann *et al.*, 2019] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[Bewley *et al.*, 2016] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple Online and Realtime Tracking. In *IEEE International Conference on Image Processing (ICIP)*, 2016.

[Bochinski *et al.*, 2017] Erik Bochinski, Volker Eiselein, and Thomas Sikora. High-Speed Tracking-by-Detection Without Using Image Information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[Bochinski *et al.*, 2018] Erik Bochinski, Tobias Senst, and Thomas Sikora. Extending IOU Based Multi-Object Tracking by Visual Information. In *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018.

[Chu and Ling, 2019] Peng Chu and Haibin Ling. FAMNet: Joint Learning of Feature, Affinity and Multi-dimensional Assignment for Online Multiple Object Tracking. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[Doersch and Zisserman, 2017] Carl Doersch and Andrew Zisserman. Multi-task Self-Supervised Visual Learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[Doersch *et al.*, 2015] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised Visual Representation Learning by Context Prediction. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[Feng *et al.*, 2019] Weitao Feng, Zhihao Hu, Wei Wu, Junjie Yan, and Wanli Ouyang. Multi-object tracking with multiple cues and switcher-aware classification. *arXiv preprint arXiv:1901.06129*, 2019.

[Kim *et al.*, 2018] Chanho Kim, Fuxin Li, and James M. Rehg. Multi-Object Tracking with Neural Gating using Bilinear LSTM. In *European Conference on Computer Vision (ECCV)*, 2018.

[Leal-Taixé *et al.*, 2015] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[Liu *et al.*, 2020] Qiankun Liu, Qi Chu, Bin Liu, and Nenghai Yu. GSM: Graph Similarity Model for Multi-Object Tracking. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[Manen *et al.*, 2017] Santiago Manen, Michael Gygli, Dengxin Dai, and Luc Van Gool. PathTrack: Fast Trajectory Annotation with Path Supervision. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[Milan *et al.*, 2016] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[Ristani *et al.*, 2016] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. In *European Conference on Computer Vision (ECCV)*, 2016.

[Srivastava *et al.*, 2015] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised Learning of Video Representations using LSTMs. In *International Conference on Machine Learning (ICML)*, 2015.

[Vondrick *et al.*, 2018] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking Emerges by Colorizing Videos. In *European Conference on Computer Vision (ECCV)*, 2018.

[Wang *et al.*, 2019] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning Correspondence from the Cycle-Consistency of Time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[Yuen *et al.*, 2009] Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba. LabelMe video: Building a Video Database with Human Annotations. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[Zhou *et al.*, 2020] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking Objects as Points. In *European Conference on Computer Vision (ECCV)*, 2020.