Tracking by Animation: Unsupervised Learning of Multi-Object Attentive Trackers

Project page: https://github.com/anonymous-projects/tracking-by-animation

Abstract

Online Multi-Object Tracking (MOT) from videos is a challenging computer vision task which has been extensively studied for decades. Most of the existing MOT algorithms are based on the Tracking-by-Detection (TBD) paradigm combined with popular machine learning approaches which largely reduce the human effort to tune algorithm parameters. However, the commonly used supervised learning approaches require the labeled data (e.g., bounding boxes), which is expensive for videos. Also, the TBD framework is usually suboptimal since it is not end-to-end, i.e., it considers the task as detection and tracking, but not jointly. To achieve both label-free and end-to-end learning of MOT, we propose a *Tracking-by-Animation* framework, where a differentiable neural model first tracks objects from input frames and then animates these objects into reconstructed frames. Learning is then driven by the reconstruction error through backpropagation. We further propose a *Reprioritized Attentive Tracking* to improve the robustness of data association. Experiments conducted on both synthetic and real video datasets show the potential of the proposed model.

1 Introduction

We consider the problem of online 2D multi-object tracking from videos. Given the historical input frames, the goal is to extract a set of 2D object bounding boxes from the current input frame. Each bounding box should have an one-to-one correspondence to an object and thus should not change its identity across different frames.

MOT is a challenging task since one must deal with: (i) unknown number of objects, which requires the tracker to be correctly reinitialized/terminated when the object appears/disappears; (ii) frequent object occlusions, which require the tracker to reason about the depth relationship among objects; (iii) abrupt pose (e.g., rotation, scale, and position), shape, and appearance changes for the same object, or similar properties across different objects, both of which make data association hard; (iv) background noises (e.g., illumination changes and shadows), which can mislead tracking.

To overcome the above issues, one can seek to use expressive features, or improve the robustness of data association. E.g., in the predominant Tracking-by-Detection (TBD) paradigm (Andriluka, Roth, and Schiele 2008; Henriques et al. 2012; Breitenstein et al. 2009; Breitenstein et al. 2011), well-performed object detectors are first applied to extract object features (e.g., potential bounding boxes) from each input frame, then appropriate matching algorithms are employed to associate these candidates of different frames, forming object trajectories. To reduce the human effort to manually tune parameters for object detectors or matching algorithms, many machine learning approaches are integrated into the TBD framework and have largely improved the performance (Xiang, Alahi, and Savarese 2015; Schulter et al. 2017; Sadeghian, Alahi, and Savarese 2017; Milan et al. 2017). However, most of these approaches are based on supervised learning, while manually labeling the video data is very time-consuming. Also, the TBD framework does not consider the feature extraction and data association jointly, i.e., it is not end-to-end, thereby usually leading to suboptimal solutions.

In this paper, we propose a novel framework to achieve both label-free and end-to-end learning for MOT tasks. In summary, we make the following contributions:

- We propose a *Tracking-by-Animation (TBA)* framework, where a differentiable neural model first tracks objects from input frames and then animates these objects into reconstructed frames. Learning is then driven by the reconstruction error through backpropagation.
- We propose a *Reprioritized Attentive Tracking (RAT)* to mitigate overfitting and disrupted tracking, improving the robustness of data association.
- We evaluate our model on two synthetic datasets (MNIST-MOT and Sprites-MOT) and one real dataset (DukeMTMC (Ristani et al. 2016)), showing its potential.

2 Tracking by Animation

Our TBA framework consists of four components: (i) a *feature extractor* that extracts input features from each input frame; (ii) a *tracker array* where each tracker receives input features, updates its state, and emits outputs representing the tracked object; (iii) a *renderer* (parameterless) that renders tracker outputs into a reconstructed frame; (iv) a *loss* that uses the reconstruction error to drive the learning of Components (i) and (ii), both label-free and end-to-end.

2.1 Feature Extractor

To reduce the computation complexity when associating trackers to the current observation, we first use a neural network NN^{feat}, parameterized by $\boldsymbol{\theta}^{feat}$, as a feature extractor to compress the input frame at each timestep t:

$$\boldsymbol{C}_t = \mathrm{NN}^{feat}\left(\boldsymbol{X}_t; \boldsymbol{\theta}^{feat}\right) \tag{1}$$

where $\boldsymbol{X}_t \in [0,1]^{H\times W\times D}$ is the input frame of height H, width W, and channel size D, and $\boldsymbol{C}_t \in \mathbb{R}^{M\times N\times S}$ is the extracted input feature of height M, width N, and channel size S, containing much fewer elements than $\boldsymbol{X}_t$.

2.2 Tracker Array

The tracker array comprises I neural trackers indexed by $i \in \{1, 2, \ldots, I\}$. Let $\boldsymbol{h}_{t,i} \in \mathbb{R}^R$ be the state vector (vectors are assumed to be in row form throughout this paper) of Tracker i at time t, and $\mathcal{H}_t = \{\boldsymbol{h}_{t,1}, \boldsymbol{h}_{t,2}, \ldots, \boldsymbol{h}_{t,I}\}$ be the set of all tracker states. Tracking is performed by iterating over two stages:

(i) State Update The trackers first associate input features from $\boldsymbol{C}_t$ to update their states $\mathcal{H}_t$, through a neural network NN^{upd} parameterized by $\boldsymbol{\theta}^{upd}$:

$$\mathcal{H}_t = \mathrm{NN}^{upd}\left(\mathcal{H}_{t-1}, \boldsymbol{C}_t; \boldsymbol{\theta}^{upd}\right) \tag{2}$$

Whilst it is straightforward to set NN^{upd} as a Recurrent Neural Network (RNN) (Rumelhart, Hinton, and Williams 1986; Gers, Schmidhuber, and Cummins 2000; Cho et al. 2014) (with all variables vectorized), we introduce a novel RAT to model NN^{upd} in order to increase the robustness of data association, which will be discussed in Sec. 3.

(ii) Output Generation Then, each tracker generates its output from $\boldsymbol{h}_{t,i}$ via a neural network NN^{out} parameterized by $\boldsymbol{\theta}^{out}$:

$$\mathcal{Y}_{t,i} = \mathrm{NN}^{out}\left(\boldsymbol{h}_{t,i}; \boldsymbol{\theta}^{out}\right) \tag{3}$$

where NN^{out} is shared by all trackers, and the output $\mathcal{Y}_{t,i} = \{y^c_{t,i}, \boldsymbol{y}^l_{t,i}, \boldsymbol{y}^p_{t,i}, \boldsymbol{Y}^s_{t,i}, \boldsymbol{Y}^a_{t,i}\}$ is a *mid-level* representation of objects on 2D image planes, including:

- **Confidence** $y^c_{t,i} \in [0,1]$ Probability of having captured an object.
- **Layer** $\boldsymbol{y}^l_{t,i} \in \{0,1\}^K$ One-hot encoding of the image layer possessed by the object. We consider each image comprising K object layers and a background layer, where higher layer objects occlude lower layer objects and the background is the 0-th (lowest) layer. E.g., when $K=4$, $\boldsymbol{y}^l_{t,i} = [0,0,1,0]$ denotes the 3-rd layer.
- **Pose** $\boldsymbol{y}^p_{t,i} = [\widehat{s}^x_{t,i}, \widehat{s}^y_{t,i}, \widehat{t}^x_{t,i}, \widehat{t}^y_{t,i}] \in [-1,1]^4$ Normalized object pose for calculating the scale $[s^x_{t,i}, s^y_{t,i}] = [1 + \eta^x \widehat{s}^x_{t,i}, 1 + \eta^y \widehat{s}^y_{t,i}]$ and the translation $[t^x_{t,i}, t^y_{t,i}] = [\frac{W}{2}\widehat{t}^x_{t,i}, \frac{H}{2}\widehat{t}^y_{t,i}]$, where $\eta^x, \eta^y > 0$ are constants.
- **Shape** $\boldsymbol{Y}^s_{t,i} \in \{0,1\}^{U\times V\times 1}$ Binary object shape mask with height U, width V, and channel size 1.
- **Appearance** $\boldsymbol{Y}^a_{t,i} \in [0,1]^{U\times V\times D}$ Object appearance with height U, width V, and channel size D.

In the output layer of NN^{out}, $y^c_{t,i}$ and $\boldsymbol{Y}^a_{t,i}$ are generated by the sigmoid function, $\boldsymbol{y}^p_{t,i}$ is generated by the tanh function, and $\boldsymbol{y}^l_{t,i}$ and $\boldsymbol{Y}^s_{t,i}$ are sampled from the Categorical and Bernoulli distributions, respectively. As sampling is not differentiable, we use the Straight-Through Gumbel-Softmax estimator (Jang, Gu, and Poole 2017) to reparameterize both distributions so that backpropagation can still be applied.

The above-defined mid-level representation is not only flexible, but also can be directly used for input frame reconstruction, enforcing the output variables to be disentangled (as would be shown later). Note that through our experiments, we have found that the discreteness of $\boldsymbol{y}^l_{t,i}$ and $\boldsymbol{Y}^s_{t,i}$ is also very important for this disentanglement.

2.3 Renderer

To define a training objective with only the tracker outputs but no training labels, we first use a differentiable renderer to convert all tracker outputs into reconstructed frames, and then minimize the reconstruction error through backpropagation. Note that we make the renderer both parameterless and deterministic so that correct tracker outputs can be encouraged to get correct reconstructions, enforcing the feature extractor and tracker array to learn to generate desired outputs. The rendering process contains three stages:

(i) Spatial Transformation We first scale and shift $\boldsymbol{Y}^s_{t,i}$ and $\boldsymbol{Y}^a_{t,i}$ according to $\boldsymbol{y}^p_{t,i}$ via a Spatial Transformer Network (STN) (Jaderberg et al. 2015):

$$\boldsymbol{T}^s_{t,i} = \mathrm{STN}\left(\boldsymbol{Y}^s_{t,i}, \boldsymbol{y}^p_{t,i}\right) \tag{4}$$

$$\boldsymbol{T}^a_{t,i} = \mathrm{STN}\left(\boldsymbol{Y}^a_{t,i}, \boldsymbol{y}^p_{t,i}\right) \tag{5}$$

where $\boldsymbol{T}^s_{t,i} \in \{0,1\}^{H\times W\times 1}$ and $\boldsymbol{T}^a_{t,i} \in [0,1]^{H\times W\times D}$ are the spatially transformed shape and appearance, respectively.

(ii) Layer Compositing Then, we synthesize K image layers ($K \le I$), where each layer can contain several objects. The k-th layer is composited by:

$$\boldsymbol{L}^m_{t,k} = \min\left(1, \sum_i y^c_{t,i} y^l_{t,i,k} \boldsymbol{T}^s_{t,i}\right) \tag{6}$$

$$\boldsymbol{L}^f_{t,k} = \sum_i y^c_{t,i} y^l_{t,i,k} \boldsymbol{T}^s_{t,i} \odot \boldsymbol{T}^a_{t,i} \tag{7}$$

where $\boldsymbol{L}^m_{t,k} \in [0,1]^{H\times W\times 1}$ is the layer foreground mask, $\boldsymbol{L}^f_{t,k} \in [0,I]^{H\times W\times D}$ is the layer foreground, and $\odot$ is the element-wise multiplication which broadcasts its operands when they are in different sizes.

(iii) Frame Compositing Finally, we iteratively reconstruct the input frame layer-by-layer, i.e., for $k = 1, 2, \ldots, K$:

$$\widehat{\boldsymbol{X}}^{(k)}_t = \left(\boldsymbol{1} - \boldsymbol{L}^m_{t,k}\right) \odot \widehat{\boldsymbol{X}}^{(k-1)}_t + \boldsymbol{L}^f_{t,k} \tag{8}$$

where $\widehat{\boldsymbol{X}}^{(0)}_t$ is initialized as the background (in this paper, we assume the background is either known or easy to extract), and $\widehat{\boldsymbol{X}}^{(K)}_t$ is the final reconstruction. The rendering process is illustrated in Fig. 1, where $\eta^x = \eta^y = 1$.

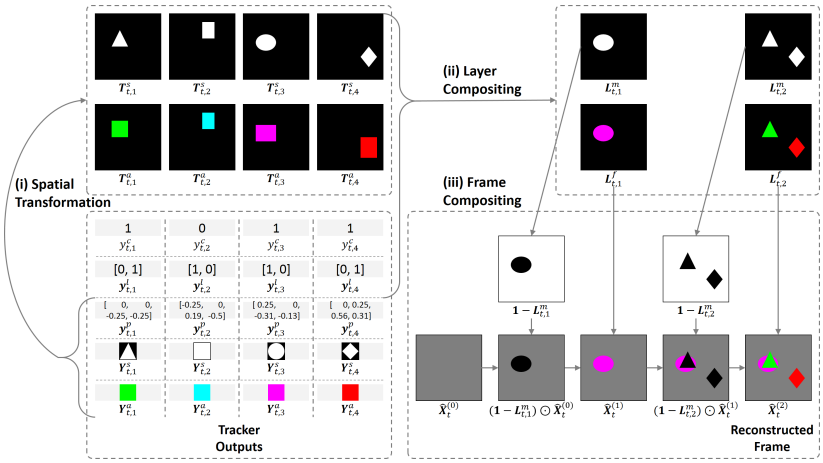

Figure 1: Illustration of the rendering process converting the tracker outputs into a reconstructed frame at time t, where the tracker number $I=4$ and the layer number $K=2$.

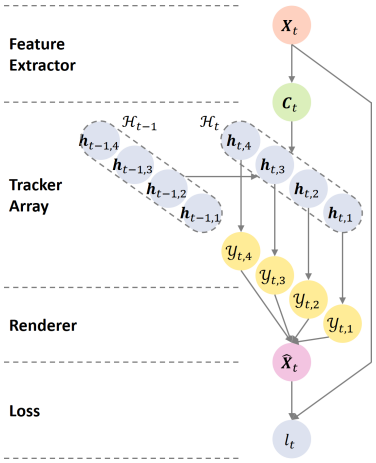

Figure 2: Overview of the TBA framework, where the tracker number $I=4$.

Whilst the layer compositing can be parallelized by matrix operations, it cannot model occlusion since pixel values in overlapped object regions are simply added; conversely, the frame compositing well-models occlusion, but the iteration process cannot be parallelized, consuming more time and memory. Thus, we combine the two to both reduce the computation complexity and maintain the ability of occlusion modeling. As occlusion usually involves only a few objects, we can set the layer number K to be small for efficiency, in which case each layer can be shared by several non-occluded objects.

2.4 Loss

To drive the learning of the feature extractor and tracker array, we define the loss l_t for each timestep:

$$l_t = \mathrm{MSE}\left(\widehat{\boldsymbol{X}}_t, \boldsymbol{X}_t\right) + \lambda \cdot \frac{1}{I}\sum_i s^x_{t,i}\, s^y_{t,i} \tag{9}$$

where, on the RHS, the first term is the reconstruction Mean Squared Error, and the second term, weighted by a constant $\lambda > 0$, is the *tightness* constraint penalizing large scales $[s^x_{t,i}, s^y_{t,i}]$ to make object bounding boxes more compact. An overview of our TBA framework is shown in Fig. 2.

3 Reprioritized Attentive Tracking

In this section, we focus on designing the tracker state update network NN^{upd} defined in (2). Although NN^{upd} can be naturally set as a single RNN as mentioned in Sec. 2.2, there can be two issues: (i) *overfitting*, since there is no mechanism to capture the data regularity that similar patterns are usually shared by different objects; (ii) *disrupted tracking*, since there is no incentive to drive each tracker to associate its relevant input features. Therefore, we propose the RAT, which tackles Issue (i) by modeling each tracker independently and sharing parameters for different trackers (this also reduces the parameter number and makes learning scalable with the tracker number), and tackles Issue (ii) by utilizing attention to achieve explicit data association (Sec. 3.1). RAT also avoids *conflicted tracking* by employing memories to allow tracker interaction (Sec. 3.2) and reprioritizing trackers to make data association more robust (Sec. 3.3), and improves efficiency by adapting the computation time w.r.t. the object number (Sec. 3.4).

3.1 Using Attention

To make Tracker i explicitly associate its relevant input features from $\boldsymbol{C}_t$ to avoid disrupted tracking, we adopt a content-based addressing. Firstly, the previous tracker state $\boldsymbol{h}_{t-1,i}$ is used to generate key variables $\boldsymbol{k}_{t,i}$ and $\beta_{t,i}$:

$$\left(\boldsymbol{k}_{t,i}, \widehat{\beta}_{t,i}\right) = \mathrm{Linear}\left(\boldsymbol{h}_{t-1,i}; \boldsymbol{\theta}^{key}\right) \tag{10}$$

$$\beta_{t,i} = 1 + \ln\left(1 + \exp\left(\widehat{\beta}_{t,i}\right)\right) \tag{11}$$

where Linear is the linear transformation parameterized by $\boldsymbol{\theta}^{key}$, $\boldsymbol{k}_{t,i} \in \mathbb{R}^S$ is the addressing key, and $\beta_{t,i} \in [1, +\infty)$ is the key strength. Then, $\boldsymbol{k}_{t,i}$ is used to match each feature vector of $\boldsymbol{C}_t$, denoted by $\boldsymbol{c}_{t,m,n} \in \mathbb{R}^S$ where $m \in \{1, 2, \ldots, M\}$ and $n \in \{1, 2, \ldots, N\}$, to get attention weights:

$$W_{t,i,m,n} = \frac{\exp\left(\beta_{t,i}\, \mathrm{K}\left(\boldsymbol{k}_{t,i}, \boldsymbol{c}_{t,m,n}\right)\right)}{\sum_{m',n'} \exp\left(\beta_{t,i}\, \mathrm{K}\left(\boldsymbol{k}_{t,i}, \boldsymbol{c}_{t,m',n'}\right)\right)} \tag{12}$$

where K is the cosine similarity defined as $\mathrm{K}(\boldsymbol{p}, \boldsymbol{q}) = \boldsymbol{p} \cdot \boldsymbol{q} / (\|\boldsymbol{p}\| \|\boldsymbol{q}\|)$, and $W_{t,i,m,n}$ is an element of the attention weight $\boldsymbol{W}_{t,i} \in [0, 1]^{M \times N}$, satisfying $\sum_{m,n} W_{t,i,m,n} = 1$. Next, a read operation is defined as a weighted combination of all feature vectors of $\boldsymbol{C}_t$:

$$\boldsymbol{r}_{t,i} = \sum_{m,n} W_{t,i,m,n}\, \boldsymbol{c}_{t,m,n} \tag{13}$$

where $\boldsymbol{r}_{t,i} \in \mathbb{R}^S$ is the read vector, representing the associated input feature for Tracker i. Finally, the tracker state is updated with an RNN, taking $\boldsymbol{r}_{t,i}$ instead of $\boldsymbol{C}_t$ as its input feature:

$$\boldsymbol{h}_{t,i} = \mathrm{RNN}\left(\boldsymbol{h}_{t-1,i}, \boldsymbol{r}_{t,i}; \boldsymbol{\theta}^{rnn}\right) \tag{14}$$

While each tracker can now attentively access $\boldsymbol{C}_t$, it still cannot attentively access $\boldsymbol{X}_t$ if the receptive field of each

feature vector $\boldsymbol{c}_{t,m,n}$ is large enough. In this case, it remains hard for the tracker to correctly associate an object from $\boldsymbol{X}_t$. Therefore, we set the feature extractor NN^{feat} as a Fully Convolutional Network (FCN) (Long, Shelhamer, and Darrell 2015; Xu et al. 2015; Wang et al. 2015) purely consisting of convolution layers. By designing the kernel size of each convolution/pooling layer, we can control the receptive field of $\boldsymbol{c}_{t,m,n}$ to be a local region on the image so that the tracker can also attentively access $\boldsymbol{X}_t$. Moreover, parameter sharing in CNN captures the spatial regularity that similar patterns are shared by objects on different image locations. As local image regions contain little information about the object translation $[t^x_{t,i}, t^y_{t,i}]$, we add this information by appending the 2D image coordinates as two additional channels to $\boldsymbol{X}_t$.

3.2 Input as Memory

To allow trackers to interact with each other to avoid conflicted tracking, at each timestep, we take the input feature $\boldsymbol{C}_t$ as an external memory through which trackers can pass messages. Let $\boldsymbol{C}^{(0)}_t = \boldsymbol{C}_t$ be the initial memory, we arrange trackers to sequentially read from and write to it, so that $\boldsymbol{C}^{(i)}_t$ records all messages written by the past i trackers. In the i-th iteration ($i = 1, 2, \ldots, I$), Tracker i first reads from $\boldsymbol{C}^{(i-1)}_t$ to update its state $\boldsymbol{h}_{t,i}$ by using (10)–(14) (where $\boldsymbol{C}_t$ is replaced by $\boldsymbol{C}^{(i-1)}_t$). Then, an erase vector $\boldsymbol{e}_{t,i} \in [0,1]^S$ and a write vector $\boldsymbol{v}_{t,i} \in \mathbb{R}^S$ are emitted by:

$$(\widehat{\boldsymbol{e}}_{t,i}, \boldsymbol{v}_{t,i}) = \mathrm{Linear}\left(\boldsymbol{h}_{t,i}; \boldsymbol{\theta}^{wrt}\right) \quad (15)$$

$$\boldsymbol{e}_{t,i} = \mathrm{sigmoid}\left(\widehat{\boldsymbol{e}}_{t,i}\right) \quad (16)$$

With the attention weight $\boldsymbol{W}_{t,i}$ produced by (12), we then define a write operation, where each feature vector in the memory is modified as:

$$\boldsymbol{c}^{(i)}_{t,m,n} = (\boldsymbol{1} - W_{t,i,m,n}\boldsymbol{e}_{t,i}) \odot \boldsymbol{c}^{(i-1)}_{t,m,n} + W_{t,i,m,n}\boldsymbol{v}_{t,i} \quad (17)$$

Our tracker state update network defined in (10)–(17) is inspired by the Neural Turing Machine (Graves, Wayne, and Danihelka 2014; Graves et al. 2016). Since trackers (controllers) interact through the external memory by using interface variables, they do not need to encode messages of other trackers into their working memories (i.e., states), making tracking more efficient.

3.3 Reprioritizing Trackers

Whilst memories are used for tracker interaction, it is hard for high-priority (small i) but low-confidence trackers to associate data correctly. E.g., when the first tracker ($i = 1$) is free ($y^c_{t-1,1} = 0$), it is very likely for it to associate or, say, 'steal' an object from succeeding trackers, since from the unmodified initial memory $\boldsymbol{C}^{(0)}_t$, all objects are equally chanced to be associated by a free tracker.

To avoid this situation, we first update high-confidence trackers so that features corresponding to the tracked objects can be firstly associated and modified. Therefore, we define the priority $p_{t,i} \in \{1, 2, \ldots, I\}$ of Tracker i as its previous (at time $t-1$) confidence ranking (in descending order) instead of its index i, and then we can update Tracker i in the $p_{t,i}$-th iteration to make data association more robust.

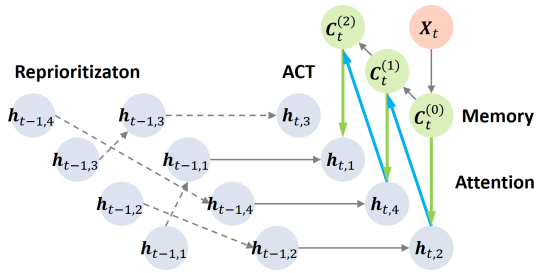

Figure 3: Illustration of the RAT with the tracker number $I = 4$. Green/Blue bold lines denote attentive read/write operations on memory. Dashed arrows denote identity transformations. At time t, the iteration is performed by 3 times and terminated at Tracker 1.

3.4 Using Adaptive Computation Time

Since the object number varies with time and is usually less than the tracker number I (assuming I is set large enough), iterating over all trackers at every timestep is inefficient. To overcome this, we adapt the idea of Adaptive Computation Time (ACT) (Graves 2016) to RAT. At each timestep t, we terminate the iteration at Tracker i (also disable the write operation) once $y^c_{t-1,i} < 0.5$ and $y^c_{t,i} < 0.5$, in which case there are unlikely to be more tracked/new objects. While for the remaining trackers, we do no use them to generate outputs. An illustration of the RAT is shown in Fig. 3.

4 Experiments

The main purposes of our experiments are: (i) investigating the importance of each component in our model, and (ii) testing whether our model is applicable to real videos. For Purpose (i), we create two synthetic datasets (MNIST-MOT and Sprites-MOT), and consider the following configurations:

TBA The full TBA model as described in Sec. 2 and Sec. 3.

TBAc TBA with constant computation time, by not using the ACT described in Sec. 3.4.

TBAc-noOcc TBAc without occlusion modeling, by setting the layer number $K = 1$.

TBAc-noAtt TBAc without attention, by reshaping the memory $\boldsymbol{C}_t$ into size $[1, 1, MNS]$, in which case the attention weight degrades to a scalar, i.e., $\boldsymbol{W}_{t,i} = W_{t,i,1,1} = 1$.

TBAc-noMem TBAc without memories, by disabling the write operation defined in (15)–(17).

TBAc-noRep TBAc without the tracker reprioritization described in Sec. 3.3.

AIR Our implementation of the 'Attend, Infer, Repeat' (AIR) (Eslami et al. 2016) for qualitative evaluation, which is a probabilistic generative model that can be used to detect objects from individual images through inference.

Note that it is hard to set a supervised counterpart of our model for online MOT, since calculating the supervised loss with ground truth data is *per se* an optimization problem which requires to access complete trajectories and thus is usually done offline (Schulter et al. 2017). For Purpose (ii), we evaluate TBA on the challenging DukeMTMC dataset (Ristani et al. 2016), and compare it to the state-of-the-art methods.

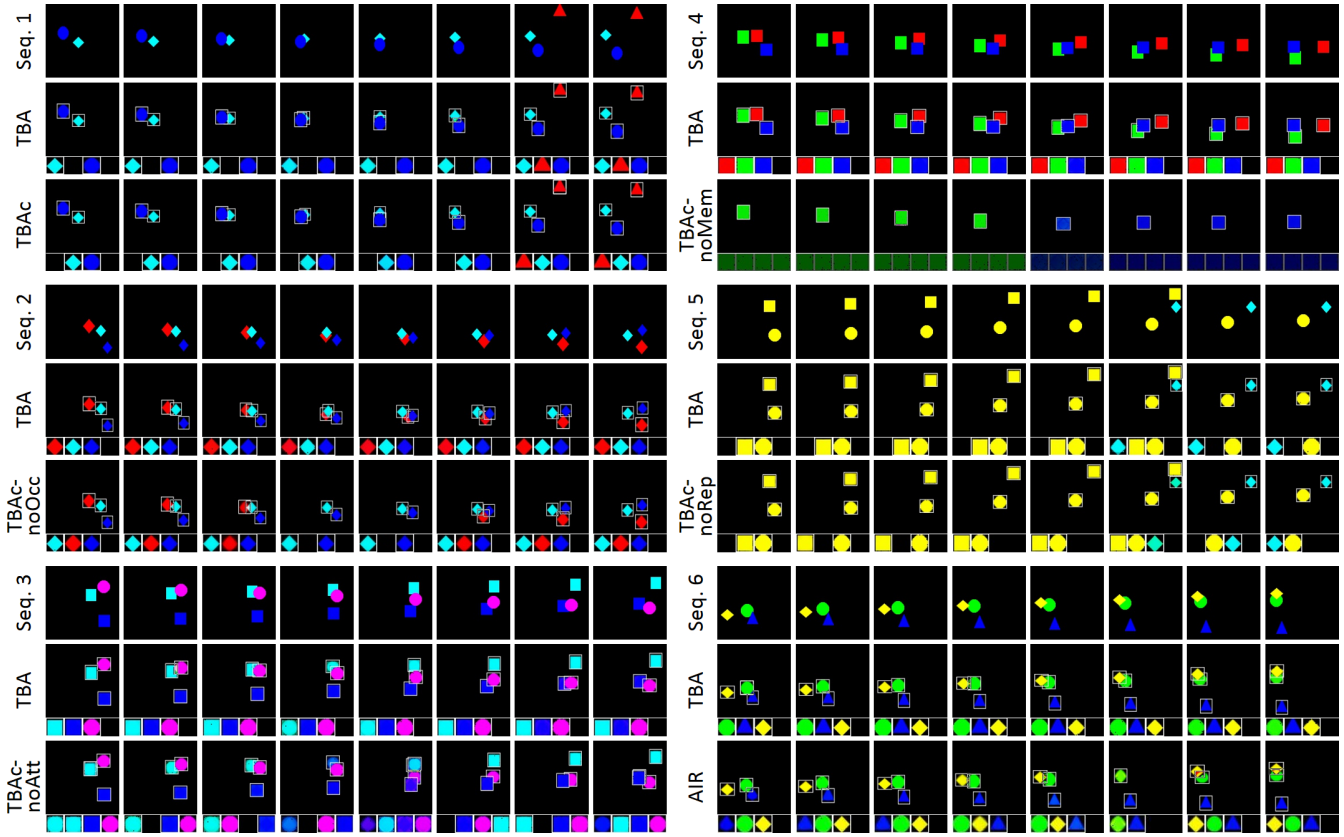

Figure 4: Qualitative results on Sprites-MOT. For each configuration, we show the reconstructed frames (top) and the tracker outputs (bottom). For each frame, tracker outputs from left to right correspond to tracker 1 to I (here $I=4$), respectively. Each tracker output $\mathcal{Y}_{t,i}$ is visualized as $\left(y^c_{t,i}\ \boldsymbol{Y}^s_{t,i} \odot \boldsymbol{Y}^a_{t,i}\right) \in [0,1]^{U\times V\times D}$.

Implementation details of our experiments are given in Appendix A.1. The MNIST-MOT experiment is reported in Appendix A.2.

4.1 Sprites-MOT

In this toy task, we aim to test whether our model can robustly handle occlusion and track the pose, shape, and appearance of the object that can appear/disappear from the scene, providing accurate and consistent bounding boxes. Thus, we create a new Sprites-MOT dataset containing 2M frames, where each frame is of size $128\times128\times3$, consisting of a black background and at most three moving sprites that can occlude each other. Each sprite is randomly scaled from a $21\times21\times3$ image patch with a random shape (circle/triangle/rectangle/diamond) and color (red/green/blue/yellow/magenta/cyan), moves towards a random direction, and appears/disappears only once. To solve this task, for TBA configurations we set the tracker number $I=4$, layer number $K=3$, and background $\widehat{\boldsymbol{X}}^{(0)}_t=\boldsymbol{0}$.

Training curves are shown in Fig. 5. TBAc-noMem has the highest validation loss, indicating that it cannot well reconstruct the input frames, while other configurations perform similarly and have significantly lower validation losses. However, TBA converges the fastest, which we conjecture benefits from the regularization effect introduced by ACT.

To check the tracking performance, we compare TBA against other configurations on several sampled sequences, as shown in Fig. 4. We can see that TBA consistently performs well on all situations, where in Seq. 1 TBAc perform as well as TBA. However, TBAc-noOcc fails to track objects from occluded patterns (in Frames 4 and 5 of Seq. 2, the red diamond is lost by Tracker 2). We conjecture the reason is that adding values of occluded pixels into a single layer can result in high reconstruction errors, and thereby the model just learns to suppress tracker outputs when occlusion occurs. Disrupted tracking frequently occurs on TBAc-noAtt which does not use attention explicitly (in Seq. 3, trackers frequently change

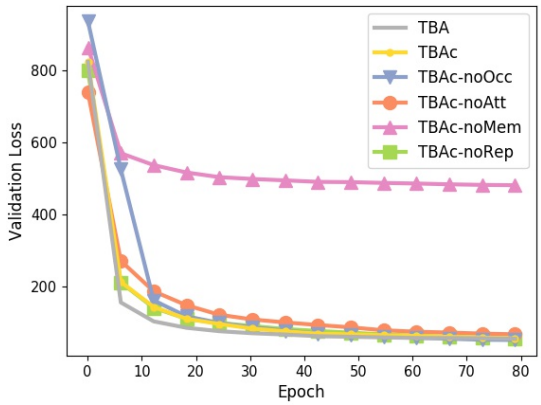

Figure 5: Training curves on Sprites-MOT.

Table 1: Tracking performances on Sprites-MOT.

Configuration	IDF1↑	IDP↑	IDR↑	MOTA↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓
TBA	97.2	97.8	96.6	97.5	81.3	0.02	980	0	224	181	142	45
TBAc	96.7	97.3	96.1	97.2	79.1	0.03	977	0	275	183	159	67
TBAc-noOcc	92.5	93.2	91.8	96.3	77.4	0.01	966	0	141	472	209	202
TBAc-noAtt	37.6	35.9	39.3	45.8	78.3	0.24	979	0	2,370	304	9,313	172
TBAc-noMem	0	–	0	0	–	0	0	987	0	22,096	0	0
TBAc-noRep	91.1	90.7	91.6	93.4	78.5	0.06	974	0	623	381	446	184

their targets). For TBAc-noMem, all trackers know nothing about each other and compete a same object, resulting in identical tracking with low confidences. For TBAc-noRep, free trackers incorrectly associate the objects tracked by the follow-up trackers. Since AIR does not consider the temporal dependency of sequence data, it fails to track objects across different timesteps.

We further quantitatively evaluate different configurations using the standard CLEAR MOT metrics (Multi-Object Tracking Accuracy (MOTA), Multi-Object Tracking Precision (MOTP), etc.) (Bernardin and Stiefelhagen 2008) that count how often the tracker makes incorrect decisions, and the recently proposed ID metrics (Identification F-measure (IDF1), Identification Precision (IDP), and Identification Recall (IDR)) (Ristani et al. 2016) that measure how long the tracker correctly tracks targets. Note that we only consider tracker outputs $\mathcal{Y}_{t,i}$ with confidences $y^c_{t,i} > 0.5$ and convert the corresponding poses $\boldsymbol{y}^p_{t,i}$ into object bounding boxes for evaluation. Table 1 reports the tracking performance. Both TBA and TBAc gain good performances and TBA performs slightly better than TBAc. For TBAc-noOcc, it has a significantly higher False Negative (FN) (472), ID Switch (IDS) (209), and Fragmentation (Frag) (202), which is consistent with our conjecture from the qualitative results that using a single layer can sometimes suppress tracker outputs. TBAc-noAtt performs poorly on most of the metrics, especially with a very high IDS of 9313 potentially caused by disrupted tracking. Note that TBAc-noMem has no valid outputs as all tracker confidences are below 0.5. Without tracker reprioritization, TBAc-noRep is less robust than TBA and TBAc, with a higher False Positive (FP) (623), FN (381), and IDS (446) that we conjecture are mainly caused by conflicted tracking.

4.2 DukeMTMC

To test whether our model can be applied to the real applications involving highly complex and time-varying data patterns, we evaluate the full TBA on the challenging DukeMTMC dataset (Ristani et al. 2016). It consists of 8 videos of resolution 1080×1920, with each split into 50/10/25 minutes long for training/test(hard)/test(easy). The videos are taken from 8 fixed cameras recording movements of people on various places of Duke university campus at 60fps. For TBA configurations, we set the tracker number $I = 10$, layer number $K = 3$, and use the IMBS algorithm (Bloisi and Iocchi 2012) to extract the background $\widehat{\boldsymbol{X}}^{(0)}_t$. Input frames are down-sampled to 10fps and resized to 108×192 to ease processing. Since the hard test set contains very different people statistics from the training set, we only evaluate our model on the easy test set.

Fig. 6 shows sampled qualitative results. TBA performs well under various situations: (i) frequent object appearing/disappearing; (ii) highly-varying object numbers, e.g., a single person (Seq. 4) or ten persons (Frame 1 in Seq. 1); (iii) frequent object occlusions, e.g., when people walk towards each other (Seq. 1); (iv) perspective scale changes, e.g., when people walk close to the camera (Seq. 3); (v) frequent shape/appearance changes; (vi) similar shapes/appearances for different objects (Seq. 6).

Quantitative performances are presented in Table 2. We can see that TBA gains an IDF1 of 80.9%, a MOTA of 76.9%, and a MOTP of 76.2%, being very competitive to the state-of-the-art methods in performance. However, unlike these methods, our model is the first one free of any training labels or extracted features.

4.3 Visualizing the RAT

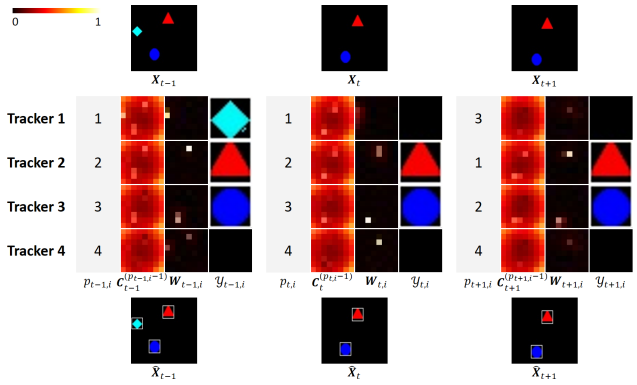

Figure 7: Visualization of the RAT on Sprites-MOT. Both the memory $\boldsymbol{C}_t$ and the attention weight $\boldsymbol{W}_{t,i}$ are visualized as $M \times N$ (8×8) matrices, where for $\boldsymbol{C}_t$ the matrix denotes its channel mean $\frac{1}{S}\sum_{s=1}^{S} \boldsymbol{C}_{t,1:M,1:N,s}$ normalized in $[0, 1]$.

To get more insights into how the model works, we visualize the process of RAT on Sprites-MOT (see Fig. 7). At time t, Tracker i is updated in the $p_{t,i}$-th iteration, using its attention weight $\boldsymbol{W}_{t,i}$ to read the memory $\boldsymbol{C}_t^{(p_{t,i}-1)}$ and then write it as $\boldsymbol{C}_t^{(p_{t,i})}$. We can see that the memory content (bright region) related to the associated object is attentively *erased* (becomes dark) by the write operation, thereby preventing the next tracker from reading it again. Note that at time $t+1$, Tracker 1 is reprioritized with a priority $p_{t+1,1} = 3$ and thus is updated at the 3-rd iteration, and the memory value has not been modified in the 3-rd iteration by Tracker 1 at which the iteration is terminated (since $y^c_{t,1} < 0.5$ and $y^c_{t+1,1} < 0.5$).

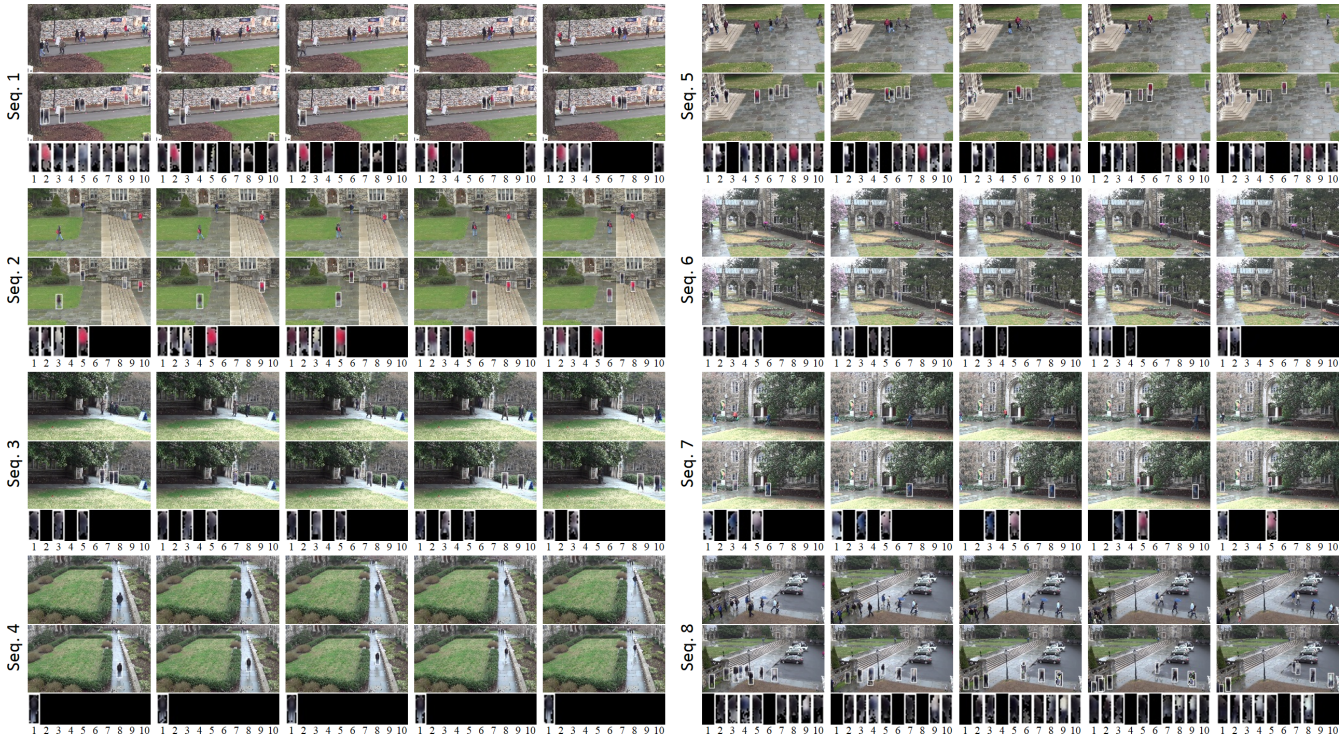

Figure 6: Qualitative results on DukeMTMC.

Table 2: Tracking performances on DukeMTMC.

Method	IDF1↑	IDP↑	IDR↑	MOTA↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓
DeepCC (Ristani and Tomasi 2018)	89.2	91.7	86.7	87.5	77.1	0.05	1,103	29	37,280	94,399	202	753
TAREIDMTMC (Jiang et al. 2018)	83.8	87.6	80.4	83.3	75.5	0.06	1,051	17	44,691	131,220	383	2,428
TBA (ours)	80.9	87.8	75.0	76.9	76.2	0.06	883	31	46,945	196,753	469	1,507
MYTRACKER (Yoon, Song, and Jeon 2018)	80.3	87.3	74.4	78.3	78.4	0.05	914	72	35,580	193,253	406	1,116
MTMC_CDSC (Tesfaye et al. 2017)	77.0	87.6	68.6	70.9	75.8	0.05	740	110	38,655	268,398	693	4,717
PT_BIPCC (Maksai et al. 2017)	71.2	84.8	61.4	59.3	78.7	0.09	666	234	68,634	361,589	290	783

5 Related Work

Unsupervised Learning for Visual Data Understanding There are many works focusing on extracting interpretable representations from visual data using unsupervised learning: some attempt to find disentangled factors ((Kulkarni et al. 2015; Chen et al. 2016; Rolfe 2017) for images and (Ondrúška and Posner 2016; Karl et al. 2017; Greff, van Steenkiste, and Schmidhuber 2017; Denton and others 2017; Fraccaro et al. 2017) for videos), some aim to extract mid-level semantics ((Le Roux et al. 2011; Moreno et al. 2016; Huang and Murphy 2016) for images and (Jojic and Frey 2001; Winn and Blake 2005; Wulff and Black 2014) for videos), while the remaining seek to discover high-level semantics ((Eslami et al. 2016; Yan et al. 2016; Rezende et al. 2016; Stewart and Ermon 2017; Wu, Tenenbaum, and Kohli 2017) for images and (Watters et al. 2017; Wu et al. 2017) for videos). However, none of these works deal with MOT tasks. To the best of our knowledge, the proposed method first achieves unsupervised end-to-end learning of MOT.

Data Association for online MOT In MOT tasks, data association can be either offline (Zhang, Li, and Nevatia 2008; Niebles, Han, and Fei-Fei 2010; Kuo, Huang, and Nevatia 2010; Berclaz et al. 2011; Pirsiavash, Ramanan, and Fowlkes 2011; Butt and Collins 2013; Milan, Roth, and Schindler 2014) or online (Turner, Bottone, and Avasarala 2014; Bae and Yoon 2014; Wu and Nevatia 2007), deterministic (Perera et al. 2006; Huang, Wu, and Nevatia 2008; Xing, Ai, and Lao 2009) or probabilistic (Schulz et al. 2003; Blackman 2004; Khan, Balch, and Dellaert 2005; Vo and Ma 2006), greedy (Breitenstein et al. 2009; Breitenstein et al. 2011; Shu et al. 2012) or global (Reilly, Idrees, and Shah 2010; Kim et al. 2012; Qin and Shelton 2012). Since the proposed RAT deals with online MOT and uses soft attention to greedily associate data based on tracker confidence ranking, it belongs to the probabilistic and greedy online methods. However, unlike these traditional methods, RAT is learnable, i.e., the tracker array can learn to generate matching features, evolve tracker states, and modify input features. Moreover, as RAT is not based on TBD and is end-to-end, the feature extractor can also learn to provide discriminative features to ease data association.

6 Conclusion

We introduced the TBA framework which achieves unsupervised end-to-end learning of MOT tasks. We also introduced the RAT to improve the robustness of data association. We validated our model on different tasks, showing its potential for real applications such as video surveillance. Our future work is to extend the model to handle videos with dynamic backgrounds.

References

[Andriluka, Roth, and Schiele 2008] Andriluka, M.; Roth, S.; and Schiele, B. 2008. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*.

[Bae and Yoon 2014] Bae, S.-H., and Yoon, K.-J. 2014. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *CVPR*.

[Berclaz et al. 2011] Berclaz, J.; Fleuret, F.; Turetken, E.; and Fua, P. 2011. Multiple object tracking using k-shortest paths optimization. *IEEE TPAMI* 33(9):1806–1819.

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

[Blackman 2004] Blackman, S. S. 2004. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine* 19(1):5–18.

[Bloisi and Iocchi 2012] Bloisi, D., and Iocchi, L. 2012. Independent multimodal background subtraction. In *CompIMAGE*.

[Breitenstein et al. 2009] Breitenstein, M. D.; Reichlin, F.; Leibe, B.; Koller-Meier, E.; and Van Gool, L. 2009. Robust tracking-by-detection using a detector confidence particle filter. In *ICCV*.

[Breitenstein et al. 2011] Breitenstein, M. D.; Reichlin, F.; Leibe, B.; Koller-Meier, E.; and Van Gool, L. 2011. Online multiperson tracking-by-detection from a single, uncalibrated camera. *IEEE TPAMI* 33(9):1820–1833.

[Butt and Collins 2013] Butt, A. A., and Collins, R. T. 2013. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *CVPR*.

[Chen et al. 2016] Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*.

[Cho et al. 2014] Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Denton and others 2017] Denton, E. L., et al. 2017. Unsupervised learning of disentangled representations from video. In *NIPS*.

[Eslami et al. 2016] Eslami, S. A.; Heess, N.; Weber, T.; Tassa, Y.; Szepesvari, D.; Hinton, G. E.; et al. 2016. Attend, infer, repeat: Fast scene understanding with generative models. In *NIPS*.

[Fraccaro et al. 2017] Fraccaro, M.; Kamronn, S.; Paquet, U.; and Winther, O. 2017. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *NIPS*.

[Gers, Schmidhuber, and Cummins 2000] Gers, F. A.; Schmidhuber, J.; and Cummins, F. 2000. Learning to forget: Continual prediction with lstm. *Neural computation* 12(10):2451–2471.

[Graves et al. 2016] Graves, A.; Wayne, G.; Reynolds, M.; Harley, T.; Danihelka, I.; Grabska-Barwińska, A.; Colmenarejo, S. G.; Grefenstette, E.; Ramalho, T.; Agapiou, J.; et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

[Graves, Wayne, and Danihelka 2014] Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

[Graves 2016] Graves, A. 2016. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*.

[Greff, van Steenkiste, and Schmidhuber 2017] Greff, K.; van Steenkiste, S.; and Schmidhuber, J. 2017. Neural expectation maximization. In *NIPS*.

[Henriques et al. 2012] Henriques, J. F.; Caseiro, R.; Martins, P.; and Batista, J. 2012. Exploiting the circulant structure of tracking-by-detection with kernels. In *ECCV*.

[Huang and Murphy 2016] Huang, J., and Murphy, K. 2016. Efficient inference in occlusion-aware generative models of images. In *ICLR Workshop*.

[Huang, Wu, and Nevatia 2008] Huang, C.; Wu, B.; and Nevatia, R. 2008. Robust object tracking by hierarchical association of detection responses. In *ECCV*.

[Jaderberg et al. 2015] Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *NIPS*.

[Jang, Gu, and Poole 2017] Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR*.

[Jiang et al. 2018] Jiang, N.; Bai, S.; Xu, Y.; Xing, C.; Zhou, Z.; and Wu, W. 2018. Online inter-camera trajectory association exploiting person re-identification and camera topology. In *MM*.

[Jojic and Frey 2001] Jojic, N., and Frey, B. J. 2001. Learning flexible sprites in video layers. In *CVPR*.

[Karl et al. 2017] Karl, M.; Soelch, M.; Bayer, J.; and van der Smagt, P. 2017. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *ICLR*.

[Khan, Balch, and Dellaert 2005] Khan, Z.; Balch, T.; and Dellaert, F. 2005. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE TPAMI* 27(11):1805–1819.

[Kim et al. 2012] Kim, S.; Kwak, S.; Feyereisl, J.; and Han, B. 2012. Online multi-target tracking by large margin structured learning. In *ACCV*.

[Kingma and Ba 2015] Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[Kulkarni et al. 2015] Kulkarni, T. D.; Whitney, W. F.; Kohli, P.; and Tenenbaum, J. 2015. Deep convolutional inverse graphics network. In *NIPS*.

[Kuo, Huang, and Nevatia 2010] Kuo, C.-H.; Huang, C.; and Nevatia, R. 2010. Multi-target tracking by on-line learned discriminative appearance models. In *CVPR*.

[Le Roux et al. 2011] Le Roux, N.; Heess, N.; Shotton, J.; and Winn, J. 2011. Learning a generative model of images by factoring appearance and shape. *Neural Computation* 23(3):593–650.

[LeCun et al. 1998] LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

[Long, Shelhamer, and Darrell 2015] Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*.

[Maksai et al. 2017] Maksai, A.; Wang, X.; Fleuret, F.; and Fua, P. 2017. Non-markovian globally consistent multi-object tracking. In *ICCV*.

[Milan et al. 2017] Milan, A.; Rezatofighi, S. H.; Dick, A. R.; Reid, I. D.; and Schindler, K. 2017. Online multi-target tracking using recurrent neural networks. In *AAAI*.

[Milan, Roth, and Schindler 2014] Milan, A.; Roth, S.; and Schindler, K. 2014. Continuous energy minimization for multi-target tracking. *IEEE TPAMI* 36(1):58–72.

[Moreno et al. 2016] Moreno, P.; Williams, C. K.; Nash, C.; and Kohli, P. 2016. Overcoming occlusion with inverse graphics. In *ECCV*.

[Niebles, Han, and Fei-Fei 2010] Niebles, J. C.; Han, B.; and Fei-Fei, L. 2010. Efficient extraction of human motion volumes by tracking. In *CVPR*.

[Ondrúška and Posner 2016] Ondrúška, P., and Posner, I. 2016. Deep tracking: seeing beyond seeing using recurrent neural networks. In *AAAI*.

[Perera et al. 2006] Perera, A. A.; Srinivas, C.; Hoogs, A.; Brooksby, G.; and Hu, W. 2006. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *CVPR*.

[Pirsiavash, Ramanan, and Fowlkes 2011] Pirsiavash, H.; Ramanan, D.; and Fowlkes, C. C. 2011. Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR*.

[Qin and Shelton 2012] Qin, Z., and Shelton, C. R. 2012. Improving multi-target tracking via social grouping. In *CVPR*.

[Reilly, Idrees, and Shah 2010] Reilly, V.; Idrees, H.; and Shah, M. 2010. Detection and tracking of large number of targets in wide area surveillance. In *ECCV*.

[Rezende et al. 2016] Rezende, D. J.; Eslami, S. A.; Mohamed, S.; Battaglia, P.; Jaderberg, M.; and Heess, N. 2016. Unsupervised learning of 3d structure from images. In *NIPS*.

[Ristani and Tomasi 2018] Ristani, E., and Tomasi, C. 2018. Features for multi-target multi-camera tracking and re-identification. In *CVPR*.

[Ristani et al. 2016] Ristani, E.; Solera, F.; Zou, R.; Cucchiara, R.; and Tomasi, C. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV*.

[Rolfe 2017] Rolfe, J. T. 2017. Discrete variational autoencoders. In *ICLR*.

[Rumelhart, Hinton, and Williams 1986] Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature* 323(6088):533–536.

[Sadeghian, Alahi, and Savarese 2017] Sadeghian, A.; Alahi, A.; and Savarese, S. 2017. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *ICCV*.

[Schulter et al. 2017] Schulter, S.; Vernaza, P.; Choi, W.; and Chandraker, M. 2017. Deep network flow for multi-object tracking. In *CVPR*.

[Schulz et al. 2003] Schulz, D.; Burgard, W.; Fox, D.; and Cremers, A. B. 2003. People tracking with mobile robots using sample-based joint probabilistic data association filters. *The International Journal of Robotics Research* 22(2):99–116.

[Shu et al. 2012] Shu, G.; Dehghan, A.; Oreifej, O.; Hand, E.; and Shah, M. 2012. Part-based multiple-person tracking with partial occlusion handling. In *CVPR*.

[Stewart and Ermon 2017] Stewart, R., and Ermon, S. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*.

[Tesfaye et al. 2017] Tesfaye, Y. T.; Zemene, E.; Prati, A.; Pelillo, M.; and Shah, M. 2017. Multi-target tracking in multiple non-overlapping cameras using constrained dominant sets. *arXiv preprint arXiv:1706.06196*.

[Turner, Bottone, and Avasarala 2014] Turner, R. D.; Bottone, S.; and Avasarala, B. 2014. A complete variational tracker. In *NIPS*.

[Vo and Ma 2006] Vo, B.-N., and Ma, W.-K. 2006. The gaussian mixture probability hypothesis density filter. *IEEE Transactions on Signal Processing* 54(11):4091–4104.

[Wang et al. 2015] Wang, L.; Ouyang, W.; Wang, X.; and Lu, H. 2015. Visual tracking with fully convolutional networks. In *ICCV*.

[Watters et al. 2017] Watters, N.; Zoran, D.; Weber, T.; Battaglia, P.; Pascanu, R.; and Tacchetti, A. 2017. Visual interaction networks: Learning a physics simulator from video. In *NIPS*.

[Winn and Blake 2005] Winn, J., and Blake, A. 2005. Generative affine localisation and tracking. In *NIPS*.

[Wu and Nevatia 2007] Wu, B., and Nevatia, R. 2007. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *IJCV* 75(2):247–266.

[Wu et al. 2017] Wu, J.; Lu, E.; Kohli, P.; Freeman, B.; and Tenenbaum, J. 2017. Learning to see physics via visual deanimation. In *NIPS*.

[Wu, Tenenbaum, and Kohli 2017] Wu, J.; Tenenbaum, J. B.; and Kohli, P. 2017. Neural scene de-rendering. In *CVPR*.

[Wulff and Black 2014] Wulff, J., and Black, M. J. 2014. Modeling blurred video with layers. In *ECCV*.

[Xiang, Alahi, and Savarese 2015] Xiang, Y.; Alahi, A.; and Savarese, S. 2015. Learning to track: Online multi-object tracking by decision making. In *ICCV*.

[Xing, Ai, and Lao 2009] Xing, J.; Ai, H.; and Lao, S. 2009. Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *CVPR*.

[Xu et al. 2015] Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

[Yan et al. 2016] Yan, X.; Yang, J.; Yumer, E.; Guo, Y.; and Lee, H. 2016. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NIPS*.

[Yoon, Song, and Jeon 2018] Yoon, K.; Song, Y.-m.; and Jeon, M. 2018. Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views. *IET Image Processing*.

[Zhang, Li, and Nevatia 2008] Zhang, L.; Li, Y.; and Nevatia, R. 2008. Global data association for multi-object tracking using network flows. In *CVPR*.

A Supplementary Materials for Experiments

A.1 Implementation Details

Model Configuration There are some common model configurations for all tasks. For the NN^{feat} defined in (1), we set it as a FCN, where each convolution layer is composed via convolution, adaptive max-pooling, and ReLU and the convolution stride is set to 1 for all layers. For the RNN defined in (14), we set it as a Gated Recurrent Unit (GRU) (Cho et al. 2014). For the NN^{out} defined in (3), we set it as a Fully-Connected network (FC), where the ReLU is chosen as the activation function for each hidden layer. For the loss defined in (9), we set $\lambda = 1$. For the model configurations specified to each task, please see in Table 3.

Training Configuration For MNIST-MOT and Sprites-MOT, we split the data into a proportion of 90/5/5 for training/validation/test; for DukeMTMC, we split the provided training data into a proportion of 95/5 for training/validation. For all tasks, the mini-batch size is set to 64 and early stopping is used to terminate training. To train the model, we minimize the averaged loss on the training set w.r.t. all model parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}^{feat}, \boldsymbol{\theta}^{upd}, \boldsymbol{\theta}^{out}\}$ using Adam (Kingma and Ba 2015) with a learning rate of 5×10^{-4}.

A.2 MNIST-MOT

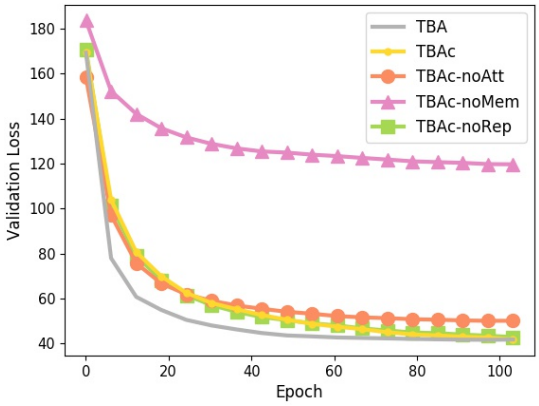

Figure 8: Training curves on MNIST-MOT.

As a pilot experiment, we focus on testing whether our model can robustly track the position and appearance of each object that can appear/disappear from the scene. Thus, we create a new MNIST-MOT dataset containing 2M frames, where each frame is of size $128 \times 128 \times 1$, consisting of a black background and at most three moving digits. Each digit is a $28 \times 28 \times 1$ image patch randomly drawn from the MNIST dataset (LeCun et al. 1998), moves towards a random direction, and appears/disappears only once. When digits overlap, pixel values are added and clamped in $[0, 1]$. To solve this task, for TBA configurations we set the tracker number $I = 4$, layer number $K = 1$, and background $\widehat{\boldsymbol{X}}_t^{(0)} = \mathbf{0}$, and fix the scale $s_{t,i}^x = s_{t,i}^y = 1$ and shape $\boldsymbol{Y}_{t,i}^s = \mathbf{1}$, thereby only compositing a single layer by adding up all transformed appearances. We also clamp the pixel values of the reconstructed frames in $[0, 1]$ for all configurations.

Training curves are shown in Fig. 8. The TBA, TBAc, and TBAc-noRep have similar validation losses which are slightly better than that of TBAc-noAtt. Similar to the results on Sprites-MOT, TBA converges the fastest, and TBAc-noMem has a significantly higher validation loss as all trackers are likely to focus on a same object, which affects the reconstruction.

Qualitative results are shown in Fig. 9. Similar phenomena are observed as in Sprites-MOT, revealing the importance of the disabled mechanisms. Specifically, as temporal dependency is not considered in AIR, overlapped objects are failed to be disambiguated (Seq. 5).

We further quantitatively evaluate different configurations. Results are reported in Table 4, which are similar to those of the Sprites-MOT.

Table 3: Model configurations specified to each task, where 'conv h×w' denotes a convolution layer with kernel size h×w, 'fc' denotes a fully-connected layer, and 'out' denotes an output layer.

Hyper-parameter	**MNIST-MOT**	**Sprites-MOT**	**DukeMTMC**
Size of $\boldsymbol{X}_t$: $[H, W, D]$	[128, 128, 1]	[128, 128, 3]	[108, 192, 3]
Size of $\boldsymbol{C}_t$: $[M, N, S]$	[8, 8, 50]	[8, 8, 20]	[9, 16, 200]
Size of $\boldsymbol{Y}^a_{t,i}$: $[U, V, D]$	[28, 28, 1]	[21, 21, 3]	[9, 23, 3]
Size of $\boldsymbol{h}_{t,i}$: R	200	80	800
Tracker number: I	4	4	10
Layer number: K	1	3	3
Coef. of $[\widehat{s}^x_{t,i}, \widehat{s}^y_{t,i}]$: $[\eta^x, \eta^y]$	[0, 0]	[0.2, 0.2]	[0.4, 0.4]
Layer sizes of NN^{feat} (FCN)	[128, 128, 1] (conv 5×5) [64, 64, 32] (conv 3×3) [32, 32, 64] (conv 1×1) [16, 16, 128] (conv 3×3) [8, 8, 256] (conv 1×1) [8, 8, 50] (out)	[128, 128, 3] (conv 5×5) [64, 64, 32] (conv 3×3) [32, 32, 64] (conv 1×1) [16, 16, 128] (conv 3×3) [8, 8, 256] (conv 1×1) [8, 8, 20] (out)	[108, 192, 3] (conv 5×5) [108, 192, 32] (conv 5×3) [36, 64, 128] (conv 5×3) [18, 32, 256] (conv 3×1) [9, 16, 512] (conv 1×1) [9, 16, 200] (out)
Layer sizes of NN^{out} (FC)	200 (fc) 397 (fc) 787 (out)	80 (fc) 377 (fc) 1772 (out)	800 (fc) 818 (fc) 836 (out)
#Parameters	1.21 M	1.02 M	5.65 M

Table 4: Tracking performances on MNIST-MOT.

Configuration	**IDF1↑**	**IDP↑**	**IDR↑**	**MOTA↑**	**MOTP↑**	**FAF↓**	**MT↑**	**ML↓**	**FP↓**	**FN↓**	**IDS↓**	**Frag↓**
TBA	98.6	97.9	99.3	97.4	82.5	0.04	978	0	360	88	124	16
TBAc	98.4	97.7	99.0	97.2	77.5	0.04	977	1	394	95	137	23
TBAc-noAtt	35.8	34.0	37.8	41.3	81.7	0.28	974	0	2779	264	10002	155
TBAc-noMem	0	–	0	0	–	0	0	983	0	22219	0	0
TBAc-noRep	93.5	92.1	95.0	94.1	77.2	0.08	979	1	790	94	424	19

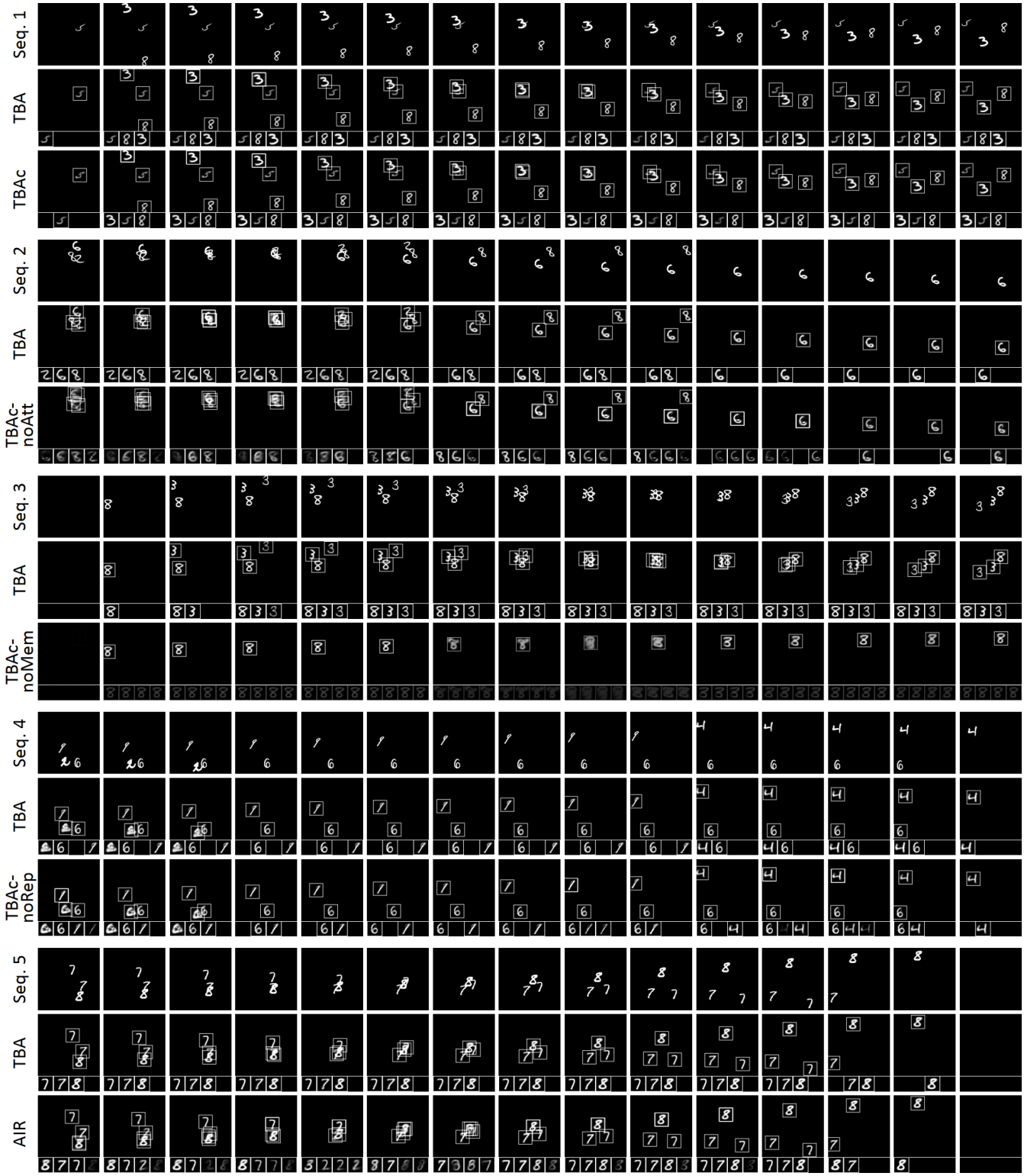

Figure 9: Qualitative results on MNIST-MOT.