DRT: Detection Refinement for Multiple Object Tracking

Bisheng Wang[1, 3]
316106002478@njust.edu.cn
Christian Fruhwirth-Reisinger[2, 3]
christian.reisinger@icg.tugraz.at
Horst Possegger[3]
possegger@icg.tugraz.at
Horst Bischof[2, 3]
bischof@icg.tugraz.at
Guo Cao[1]
caoguo@njust.edu.cn

[1] School of Computer Science and Engineering
Nanjing University of Science and Technology

[2] Christian Doppler Laboratory for Embedded Machine Learning

[3] Institute of Computer Graphics and Vision
Graz University of Technology

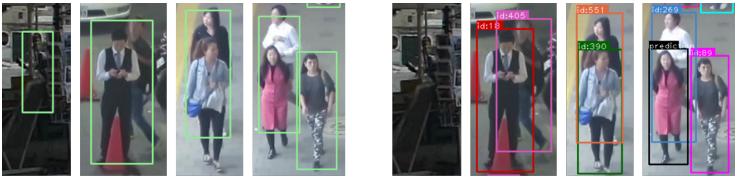
Abstract

Deep learning methods have led to remarkable progress in multiple object tracking (MOT). However, when tracking in crowded scenes, existing methods still suffer from both inaccurate and missing detections. This paper proposes *Detection Refinement for Tracking* (DRT) to address these two issues for people tracking. First, we construct an encoder-decoder backbone network with a novel semi-supervised heatmap training procedure, which leverages human heatmaps to obtain a more precise localization of the targets. Second, we integrate a "one patch, multiple predictions" mechanism into DRT which refines the detection results and recovers occluded pedestrians at the same time. Additionally, we leverage a data-driven LSTM-based motion model which can recover lost targets at a negligible computational cost. Compared with strong baseline methods, our DRT achieves significant improvements on publicly available MOT datasets. In addition, DRT generalizes well, *i.e.* it can be applied to any detector to improve their performance.

1 Introduction

Robust and well-performing MOT algorithms are vital for applications ranging from autonomous vehicles to automated video analysis and surveillance. However, in crowded scenes, we still suffer from various challenges, such as occlusion, complex and cluttered backgrounds, pose variations, *etc*.

Most state-of-the-art trackers, *e.g.* [4, 11, 41, 44], adopt the tracking-by-detection principle, which consists of two major steps: (1) Detecting all objects frame by frame; (2) Associating targets in different frames. Therefore, the performance of MOT relies greatly on the detection results. Despite the encouraging progress from better detectors [23, 29], there are still two main problems with MOT in complex scenarios and high-density crowds, as



(a) (b)

Figure 1: Detection failures of the state-of-the-art tracker JDE [37] (a) and our corresponding refined results (b) when using the same input detections as JDE on the MOT16 dataset [25].

illustrated in Figure 1. First, complex backgrounds may be incorrectly detected due to appearance ambiguities. Second, in crowded scenes with frequent occlusions, occluded objects are easy to miss and adjacent objects are often imprecisely detected as one, resulting in many false negative trajectories and ID switches.

The easiest way to handle these problems is to collect larger training datasets [24, 46] or to design better detectors [14, 28, 46]. In this paper, we take another route: Post-processing the detected patches to improve performance. We propose Detection Refinement for Tracking (DRT), which can be used in any MOT method to refine detections, no matter which detector is originally used. Our DRT consists of an encoder-decoder convolutional network with two side branches, called DRT-net, and an LSTM based motion model.

The DRT-net has three components: an encoder-decoder backbone network and two side branches, each branch containing a "Feature Integration" (FI) module. The backbone network is trained to generate human heatmaps which contain segmentation information and indicate the specific location of every pedestrian. With the generated heatmaps, the branches can output more precise coordinates of targets. In order to recover the occluded targets, a "one patch, multiple predictions" mechanism is introduced. Rather than only predicting one location for each detected patch, we carefully construct two side branches, one of which is responsible for attaining more precise detections while the other is for recovering potentially occluded people. We show that with a well-designed loss function and an efficient semi-supervised training procedure, difficult targets can be recovered more accurately. Moreover, we demonstrate the importance of an FI module in effectively reducing false positives by inserting it into side branches.

In recent years, trajectory forecasting [30] has been tackled by some well-designed RNN-based models [1, 34]. If transferred to MOT, such robust trajectory forecasting methods can help recover missed targets during tracking. Consequently, we replace the widely used Kalman Filter [19] with an LSTM-based motion model in our tracking framework. With the long term information, the simple motion model can help recover missed targets and thus, reduce false negatives during online tracking. Moreover, because of the low dimensional (8D) input vector and simple network structure, prediction time is negligible.

In summary, we make the following contributions:

(1) A novel post-processing method, DRT, is designed for better detection and tracking.

(2) Instance segmentation information as attention and a "one patch, multiple predictions" mechanism are introduced into DRT to refine detection and recover occluded targets.

(3) An LSTM-based motion model with a carefully designed training schema supports the tracking framework with precise predictions to recover missed targets.

(4) Even with a simplified association framework, our methods can achieve state-of-the-art performance on standard tracking benchmarks.

2 Related work

We briefly summarize related work, focusing on top-performing online trackers. To this end, we resort to the leaderboard of the MOT Challenge[1], which aims to collect datasets and to create a standardized evaluation environment. In the following, we classify the related online trackers by whether they use publicly available detections or private detections.

Tracking with public detections. To track with public detections, existing methods [7, 8, 50] achieve better tracking results by integrating predictors [16, 19] to obtain more candidates. MOTDT [7] generated candidates from Kalman Filter [19] and used R-FCN [11] to filter out reduplicated candidates. Some other methods [8, 50] adopted a similar idea with new predictors [12, 16]. These methods are simple but may result in many false positives. Tracktor [2] leveraged the tracked targets in the current frame as proposals to do a second detection for the next frame. A second line of works use different association frameworks for robust tracking. Xiang *et al*. [41] formulated MOT as a Markov Decision Process and designed a detailed tracking framework. Some methods [13, 31, 42] integrated multiple types of information and constructed complex RNN models for association. Moreover, Brasó and Leal-Taixé [5] formulated tracking as a graph problem based on message passing networks.

Tracking with private detections. To achieve better tracking performance, many methods, *e.g*. [4, 38, 44], have used their own detectors. SORT [4] adopted Faster RCNN [29] for accurate detections. POI [44] collected larger amounts of datasets to train the detection model and person re-identification (ReID) model. Moreover, MAT [14] took advantage of Cascade RCNN [6] to ensure a well-performing detection. There have also been some one-stage methods which achieve a balance between performance and efficiency. JDE [37] inserted an identity classification branch into the FPN [23] detector. FAIR [46] improved the performance by employing an anchor-free detection network. CTracker [28] relied on two adjacent frames as input to the detector to obtain detections and associate targets at the same time. These methods get competitive results by introducing well-performing detectors, while in our work, we improve the tracking performance by a post-processing method, DRT, which can improve all existing detectors, no matter how good they initially are. In addition, ReMOT [43] also put forward the concept of 'refinement'. However, they concentrate on how to refine the tracklets rather than detections.

3 Detection Refinement for Tracking

In this section, we will first introduce our tracking process in Section 3.1. Then the two specific parts: DRT-net, which post-processes the inaccurate detections, and the new LSTM-based motion model will be carefully explained in Section 3.2 and Section 3.3, respectively.

3.1 The overall tracking framework

Data preparation. Let the trajectories in the previous frames be $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_n\}$ and their corresponding predicted locations in the current frame t as $P_t = \{p_1, p_2, ..., p_n\}$. The refined detections output from DRT-net in the current frame are $D_t = \{d_1, d_2, ..., d_m\}$ and the appearance features of D_t are $F_t = \{f_1, f_2, ..., f_m\}$. n and m are the number of trajectories in previous frames and the number of refined detections in the current frame.

[1]The official web page of MOT Challenge can be found at https://motchallenge.net.

Matching. First, the appearance cost matrix $C_{n\times m}$ between $\mathcal{T}$ and D_t is computed. For this, we store the R_i most recent feature representations for each $\tau_i = \{r_1, r_2, ..., r_{R_i}\}$ and compute the cost $c_{i,j}$ of matching it with detection d_j as:

$$c_{i,j} = \min_{k\in\{1,...,R_i\}} (1 - r_k^T f_j) \tag{1}$$

We apply the Hungarian algorithm on $C_{n\times m}$ to get the appearance matched pairs M_{app}, as well as unmatched trajectories and detections, $\mathcal{T}^{app}_{unmatched}$ and $D^{app}_{unmatched}$. Pairs with cost larger than the threshold δ will not be matched. Second, we find the potential pairs between $\mathcal{T}^{app}_{unmatched}$ and $D^{app}_{unmatched}$ by their IoU (Intersection over Union) cost $B_{n'\times m'}$, where n' and m' are the number of $\mathcal{T}^{app}_{unmatched}$ and $D^{app}_{unmatched}$. For each unmatched τ_i and d_j, the IoU cost is:

$$b_{i,j} = 1 - IoU(p_i, d_j) \tag{2}$$

We apply the Hungarian algorithm again on $B_{n'\times m'}$ to find location matched pairs M_b. If any pair has $b_{i,j}$ larger than 0.5, it will not be matched. After the two steps, we can get the final matched pairs $M = M_{app} \cup M_b$, unmatched tracks $\mathcal{T}_{unmatched}$ and detections $D_{unmatched}$.

Update. After association, for each matched pair (τ_i, d_j), we assign the feature f_j and the location of d_j to τ_i and drop the oldest feature in τ_i if τ_i has kept more than R_i features. For any unmatched detection, we generate a new trajectory if its detection confidence is higher than the pre-set threshold (0.7) or if it can be matched in the next frame. For any unmatched trajectory, if it has been continuously tracked for a sufficiently long time (4 frames), we check its predicted location with DRT-net to decide whether to output the predicted location. The trajectory will be terminated if it is not associated for more than pre-defined A_{max} frames.

Comparison of DRT, JDE and DeepSORT. Our association framework is simplified from DeepSORT without cascade strategy. Similar to most trackers, DeepSORT, JDE and DRT employ the same widely-used strategy, *i.e.* using a motion model to smooth the trajectories and the Hungarian algorithm for data association. DeepSORT adds a deep appearance descriptor and a cascaded matching step into the framework. JDE is also based on DeepSORT. They simultaneously obtain detections and appearance features in one network, but also need the tracking-by-detection pipeline for association. Our contributions are to refine the detections via DRT-net (Section 3.2) and the data-driven motion model (Section 3.3).

3.2 DRT-net

In this section, we introduce the architecture of DRT-net, which refines the outputs of standard object detectors before associating them with trajectories. As illustrated in Figure 2, the network consists of a backbone network to generate human-shaped heatmaps and two side branches. Each branch has a Feature Integration (FI) module to learn discriminative features.

3.2.1 Backbone network

The backbone of DRT-net is an encoder-decoder network consisting of the convolution layers of ResNet50 [15] and three up-projection modules from [18, 21]. During training, the pre-trained ResNet50 is utilized for fine-tuning. Outputs of the backbone are heatmaps which can indicate the location of people and will be leveraged as attention to help the side branches predict more precise locations. Because binary segmentation masks do not convey enough boundary information (especially in the case of an occlusion), inspired by [48], we adopt the Gaussian function for generating human heatmaps (Figure 2 (a)).

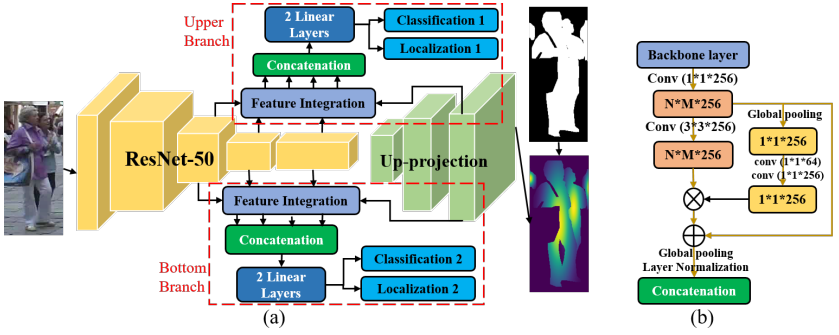


Figure 2: (a) DRT-net. It consists of one backbone network and two side branches. The right side are the segmentation mask and heatmap of the patch. (b) Feature Integration module.

Generating heatmaps. Given the patch of a target in an image and its corresponding segmentation mask M, we first obtain the coordinate of every pixel inside the target $(x_i, y_i), i \in \{1, 2, ..., N\}$. N is the amount of pixels inside a target. Then, we use these coordinates to compute the center location $(\hat{x}, \hat{y})$ as well as the variance (σ_x, σ_y):

$$\hat{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \hat{y} = \frac{1}{N}\sum_{i=1}^{N} y_i, \quad \sigma_x = \frac{\sum_{i=1}^{N} x_i - \hat{x}}{N-1}, \quad \sigma_y = \frac{\sum_{i=1}^{N} y_i - \hat{y}}{N-1}. \tag{3}$$

Finally, the heatmap at location (x, y) is computed using Gaussian function:

$$H(x,y) = \begin{cases} 0 & M(x,y) = 0, \\ \frac{1}{2\pi\sqrt{\sigma_x \sigma_y}} e^{-\left(\frac{(x-\hat{x})^2}{2\sigma_x} + \frac{(y-\hat{y})^2}{2\sigma_y}\right)} & M(x,y) = 1. \end{cases} \tag{4}$$

For any partially occluded target, its heatmap is computed only on the visible areas.

Collecting data. Because there is limited annotation for instance segmentation in MOT, we obtain training data in the following way: First, we utilize the MOTS dataset [35], which is publicly available and shares the same videos as MOT [25]. In MOTS, some but not all, people are labelled. Second, we use Detectron2 [40] to detect other unlabelled data. If a detected box has an IoU larger than 0.7 with any ground truth, its corresponding segmentation will be kept. We also do a manual check to ensure precise segmentation (imprecise ones will be dropped directly). The whole data preparation step is needed only once per dataset and can be done with rather few effort. Third, targets neither labelled nor detected will be ignored for the backbone network loss and only utilized to train the branches. Therefore, the backbone network training is actually a semi-supervised task.

Finally, we leverage the smooth l_1 loss [27] to train the backbone. Let h and H represent the predicted and ground truth heatmap, then the loss is computed as:

$$l_h = F_{smooth\_l1}(h, H). \tag{5}$$

3.2.2 Feature Integration

In order to learn more abundant features, we fuse features from different backbone layers for the branch network. Four backbone layers in Figure 2 (a) go through the FI module (Figure 2 (b)) [36] and are then concatenated together to provide low-level texture information, high-level semantic information and segmentation information. Each backbone layer will pass three paths (Figure 2 (b)): A convolution layer, a global pooling layer and an identity

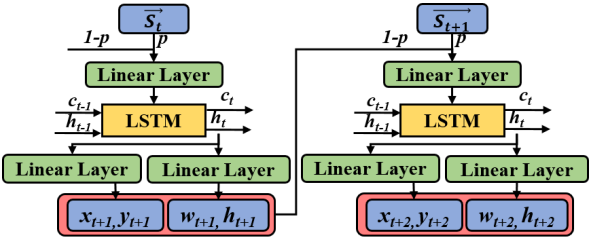


Figure 3: The LSTM-based motion model.

mapping, which is a combination of SENet [17] and ResNet [15]. In Section 4.2, we will also demonstrate that FI can help filter out more false positives.

3.2.3 One patch, multiple predictions

Recently, pedestrian detection in crowded scenes received more attention from our research community, *e.g.* [9, 32, 33]. To recover occluded people, we adapt the "one proposal, multiple prediction" approach from [9] for multi object tracking. First, we need to adapt the image based network, which relies on features from a larger receptive field, into a patch based network. To this end, we design a new two-step loss function and training mode:

$$\text{step}_1 : L = l_h + l_{c_1}(c_1, g_1) + l_{l_1}(l_1, g_1), \tag{6}$$

$$\text{step}_2 : L = l_{c_2}(c_2, g_2) + l_{l_2}(l_2, g_2). \tag{7}$$

Here, l_h is from Equation (5). c_1, l_1, c_2 and l_2 are the classification and localization predictions output from the upper and bottom branches in Figure 2, while g_1 and g_2 are the top-two ground truth annotations, *i.e.* the two best matches for the corresponding patch. $l_{c_1}, l_{c_2}, l_{l_1}$, and l_{l_2} are the classification and localization losses for the two branches, which share the same loss function as SSD [24].

Different from the loss function in [9], which randomly arranges different matchings between two outputs and their corresponding ground truth targets, we assign a fixed permutation for the loss function: The "Upper Branch" is assigned to find the best matched target and refine its location, while the "Bottom Branch" is responsible for predicting if the patch has another person and recovering its location. This is because our refinement network, which has limited receptive fields, learns information mainly on the best matched target. During training, if the loss functions in Equation (6) and (7) are trained together, convergence is suboptimal due to optimizing five separate loss terms simultaneously. To this end, we divide the training into two steps. In step 1, we only train the backbone and upper branch. Then in step 2, the second branch is trained with parameters of the backbone and the upper branch fixed.

For classification, if the input patch has an IoU larger than 0.5 with its best matched or second matched ground truth, it will be regarded as a positive example by the corresponding branch. If the IoU is lower than 0.4, it will be used as a negative example. Otherwise it will be ignored. For localization loss, only patches with IoU larger than 0.4 are used.

3.3 LSTM-based motion prediction model

To follow a temporarily occluded target, we need a motion model forecasting its trajectory. Recent MOT approaches often replace the traditional Kalman filter by various LSTM based motion models, especially for evaluations on trajectory forecasting benchmarks [30]. However, these typically predict only the location of the bounding box center and are not robust to

noisy detections, which we often encounter in real tracking applications. Thus, we propose the following data augmentations and training scheme to obtain a both efficient and robust LSTM-based motion model.

First, instead of predicting only the center locations, we construct an 8D input vector to predict location and size of the bounding boxes. More formally, let the state $\vec{s}_t$ of a target in frame t be $\vec{s}_t = (x_t, y_t, w_t, h_t, \Delta x_t, \Delta y_t, \Delta w_t, \Delta h_t)$. Here, x_t, y_t, w_t and h_t are the center coordinates, width and height of the target, respectively, while $\Delta i_t = i_t - i_{t-1} (i \in \{x, y, w, h\})$ are the velocities (initialized with 0 at the first frame). The network inputs the 8D state vector through a linear layer, then an LSTM unit, and finally two linear output layers to predict the location and bounding box size in the next frame (see Figure 3).

Second, we discovered that motion models trained with ground truth annotations are not stable in real scenarios, where only noisy detection results are available. We overcome this issue by introducing two augmentations to train a more robust model: (1) We replace ground truth annotations by high quality detections if available, *i.e.* if there is a corresponding detection with IoU > 0.7. (2) Additionally, we add noise to the training data by randomly shifting the boxes.

In practical situations, we need to predict the location of lost targets very frequently. Training with labelled data, however, can not ensure good long-term forecasting performance. To bridge this gap between training and inference, we employ a sampling mechanism known from language processing [45], which randomly selects a training sample from either ground truth or the predicted bounding box. Let $p(i)$ denote the probability that we sample the input from ground truth data in epoch i. Then, we will sample the input from the prediction instead with a probability of $1 - p(i)$. At the beginning, $p(i)$ should be high so that the model can converge fast, while in later epochs, more prediction data should be used. Therefore, we introduce a decay function to generate $p_{(i)}$ for different training epoch i:

$$p_{(i)} = \frac{\mu}{\mu + \exp^{(i/\mu)}}, \tag{8}$$

where μ is a pre-defined parameter. This yields a robust and efficient motion model, as we will demonstrate in the following evaluation.

4 Experiments

4.1 Training settings

Experiments are mainly conducted on two public datasets[2]: MOT16 [25] and MOT17, which provide the same 7 training videos and 7 test videos but different public detections and ground truth labels. These videos are challenging because they contain frequent occlusions, varied camera perspectives, human poses as well as cluttered backgrounds.

Unless stated otherwise, we use the same detections as JDE [37] for tracking. DRT-net and the motion model only take advantage of the MOT dataset for training (since the segmentation labels from MOTS [35] are provided for exactly the same train sequences). All our experiments are implemented in PyTorch [27] with a single Nvidia GTX TITAN Xp GPU. For DRT-net, the training lasts for 70 epochs with an initial learning rate of 0.0001, which will be divided by 10 every 30 epochs. Batch size is 40 for each iteration. To extract features, we use the same Re-Identification (ReID) model trained in [5]. For our motion

[2] Additional results are included in the supplemental material.

Method	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓
JDE	64.4	55.8	35.4	20.0	-	-	1544
JDE + ResNet50	65.2	59.2	36.1	17.6	10642	51321	1421
JDE + DRT-net1	66.8	60.0	35.2	18.1	8155	51150	1324
JDE + DRT-net1 + FI ⋆	68.7	60.8	36.6	17.9	6824	48966	1320
JDE + DRT-net1 + FI	69.1	**61.2**	35.2	17.5	**5610**	49417	1267
JDE + DRT-net2 + FI	69.5	60.3	36.4	17.5	5716	48549	1302
JDE + DRT-net2 + FI + KF	70.2	60.6	39.5	16.9	7075	46106	1185
JDE + DRT-net2 + FI + LSTM	**71.1**	60.9	**44.0**	**15.9**	8244	**43247**	**1173**

Table 1: Ablation study of different components in our method on MOT16. See Section 4.2 for details. ⋆ only uses masks generated from Detectron2.

	DRT-net1	DRT-net1 + FI	DRT-net2 + FI	DRT-net2 + FI + LSTM
ResNet50	46.4	52.1	55.6	69.1
ResNet34	29.8	35.2	39.1	58.0

Table 2: Ablation study on the average time (ms) consumed per frame.

model, the training process also takes 70 epochs with an initial learning rate of 0.001 and batch size 8. All trainable parameters in DRT-net and the motion model are trained using the Adam optimizer [20] with β_1 and β_2 set to 0.5 and 0.999, respectively.

Evaluation Metrics. We follow the common evaluation protocol and use the widely accepted CLEAR MOT metrics [4], including Multiple Object Tracking Accuracy (MOTA), ID F1 Score (IDF1), Multiple Object Tracking Precision (MOTP), along with the trajectory quality measures [22], Mostly Tracked (MT), Mostly Lost (ML), False Positives (FP), False Negatives (FN) and Identity Switches (IDS).

4.2 Ablation study

We conduct ablation experiments on the MOT16 test set to evaluate the performance of different components. The results have been obtained by the official MOT Challenge submission server and are summarized in Table 1. We start with (1) the baseline method JDE [37] which uses a framework similar to DeepSORT [38], but uses a private detector. Using the JDE detections as input, we gradually add our components: (2) ResNet50 backbone without any other modules is adopted as a reference for fair comparison. (3) DRT-net1 has only the upper branch of Figure 2(a) without the FI module. (4) DRT-net1 + FI adds the FI module to the top branch to train a more discriminative model. We also evaluate this network configuration trained only on Detectron2-generated masks (*i.e.* replacing the MOTS annotations by Detectron2 estimates). (5) DRT-net2 + FI adds the bottom branch in Figure 2(a) to DRT-net. Finally, (6) DRT-net2 + FI + LSTM replaces the Kalman Filter with our LSTM-based motion model.

As can be seen in Table 1, both our DRT-net and the LSTM-based motion model perform well. For the single branch DRT-net1 (without FI module), we achieve 2.4% MOTA improvement compared with JDE. Adding the FI module yields another 2.1% improvement by reducing much more false positives. These improvements can be attributed to two reasons. On the one hand, our refined detections become more accurate (as can also be seen by the DetPr score in Table 4), indicating that previous false negatives (true detections with IoU < 0.5) now become true positives. On the other hand, the classification branch filters out many background boxes. Some examples can be seen in Figure 1. In addition, because detected boxes are more precise after going through DRT-net, ID switches also decrease slightly compared to JDE. The two-branch DRT-net2 obtains another 0.4% MOTA improve-

	Predicted Frames	0	1	2	3	4	5
Kalman Filter	MOTA	79.2	80.4	80.8	80.9	80.9	80.7
	FP	880	1673	2311	2981	3682	4458
	FN	21242	19157	18053	17307	16655	16084
LSTM	MOTA	79.2	80.8	81.4	81.7	82.0	82.1
	FP	863	1417	1930	2459	3003	3583
	FN	21234	18976	17866	16985	16155	15474

Table 3: Trajectory forecasting results of Kalman filter and our LSTM model.

Datasets	Method	MOTA↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	DetPr↑	Hz↑	GPU
MOT16	DeepSORT [53]	61.4	62.2	32.8	18.2	12852	56668	781	74.7	17.4	Quadro M 6000
	JDE [51] ⋆	64.4	55.8	35.4	20.0	-	-	1544	-	18.8	RTX2080Ti
	POI [49]	66.1	65.1	34.0	20.8	**5061**	55914	1073	-	9.9	GTX970
	Tub_TK_POI [42]	66.9	62.2	39.0	16.1	11544	47502	1236	75.0	1.0	TITAN XP
	CTTracker [43]	67.6	57.2	32.9	23.1	8934	48305	1897	75.7	**34.4**	Tesla P40
	FAIR [46] ♢	69.3	72.3	40.3	16.7	13501	41653	815	76.3	25.9	RTX2080Ti
	TraDeS [53] •	70.1	64.7	37.3	20.0	8091	45210	1144	76.2	22	RTX2080Ti
	FAIRv2 [47] ♣	74.9	72.8	44.7	15.9	-	-	1074	-	25.9	RTX2080Ti
	DRT_ResNet34_JDE (Ours) ⋆	70.2	60.6	40.6	17.0	7568	45595	1142	75.1	8.0	TITAN XP
	DRT_JDE (Ours) ⋆	71.1	60.9	44.0	15.9	8244	43247	1173	76.9	6.8	TITAN XP
	DRT_FAIR (Ours) ♢	74.4	69.3	44.5	16.1	8006	37710	871	**77.0**	6.5	TITAN XP
	DRT_FAIRv2 (Ours) ♣	**76.3**	**72.9**	**48.0**	**14.4**	14416	**28023**	**731**	75.0	6.4	TITAN XP
MOT17	DeepSORT [53]	60.3	61.2	31.5	20.3	36111	185301	2442	-	17.4	Quadro M 6000
	Tube_TK [42]	63.0	58.6	31.2	19.9	27060	177483	4137	75.7	1.0	TITAN XP
	CTTracker [43]	66.6	57.4	32.2	24.2	22284	160491	5529	76.3	**34.4**	Tesla P40
	FAIR [46] ♢	67.5	69.8	37.7	20.8	-	-	2868	-	25.9	RTX2080Ti
	CenterTrack[43] •	67.8	64.7	34.6	24.6	**18498**	160332	3039	76.9	22	TITAN XP
	TraDeS [53] •	69.1	63.9	36.4	21.5	20892	150060	3555	76.8	22	RTX2080Ti
	FAIRv2 [47] ♣	73.7	**72.3**	43.2	17.3	-	-	3303	-	25.9	RTX2080Ti
	DRT_JDE (Ours) ⋆	70.0	60.2	42.5	18.0	21798	144027	3606	76.8	6.8	TITAN XP
	DRT_FAIR (Ours) ♢	73.7	68.5	42.8	16.8	19608	126144	2742	**77.8**	6.5	TITAN XP
	DRT_FAIRv2 (Ours) ♣	**76.4**	**72.3**	**46.8**	**15.2**	35934	**94788**	**2334**	76.2	6.4	TITAN XP

Table 4: Comparison with other state-of-the-arts on MOT16 and MOT17. All methods are under the "private detector" protocol. ↑/↓ denote that higher/lower results are better. Methods marked with the same sign use the same detector.

ment by recovering occluded targets.

Adding the motion model predictions further improves the performance due to recovering more missing detections. In particular, our LSTM model achieves another 1.6% MOTA improvement and 129 less ID switches, while the Kalman filter only obtains 0.7% MOTA improvement and 117 less ID switches. To carefully analyze this improvement, Table 3 analyses how FP, FN and MOTA change when using the motion models to predict a different amount of frames for lost targets. For our LSTM model, we use 6 of the 7 available training videos to train the LSTM and use the remaining one for testing. This process is repeated 7 times to get the final results of all 7 videos. For the Kalman filter, we report the best results after a parameter sweep. We can see that the MOTA improvement saturates after 3 and 5 frames for the Kalman filter and our LSTM model, respectively. The results show that our LSTM is more robust for forecasting trajectories and has a better performance at long term occlusions. Because we rank variants by their MOTA score in Table 1, we let the Kalman filter predict at most 3 frames for each lost target, whereas the LSTM model predicts up to 5 frames. This also explains why FP increases in Table 1 after applying the LSTM.

4.3 Comparison with the State-of-the-Art

We also compare our method with other state-of-the-art approaches using private detections. Three different detectors from JDE [51], FAIR [46] and FAIRv2 [47] are used in our method to prove that our DRT can be easily applied to any detector. The results (see Table 4) are compared on both MOT16 and MOT 17 test datasets. Additional results on MOT20 and a

separate detection evaluation on MOT17Det are included in the supplemental material.

Comparison with baseline methods. By refining the detections from three strong detectors, our DRT significantly improves the baseline results. With MOTA 71.1% and IDF1 60.8%, we improve over JDE by 6.7% and 5.0% on MOT16. Compared with one of the currently strongest detectors, FAIR [46], our DRT still improves MOTA by 5.1% (MOT16) and 6.2% (MOT17) and performs on par in terms of IDF1 and ID switches. Applied on the recent FAIRv2 [47], we achieve another MOTA improvement of 1.4% (MOT16) and 2.7% (MOT17).

Comparison with other methods. Our DRT achieves better results than recent state-of-the-art approaches published at major conferences. Even based on a worse detector, MOTA of DRT_JDE is between 1.0% and 4.2% higher than Tub_TK_POI [26], CTTracker [28] and TraDes [39]. Our results confirm that we can refine existing detectors with a deliberately simple approach based on using heat maps and motion context to achieve state-of-the-art tracking-by-detection performance. The heat map structure is a new idea which uses mask information to achieve more precise detections, thus we obtain the highest detection precision (DetPr) of all methods. The motion model, on the other hand, can effectively recover previously missed targets. This results in much less missed trajectories and overall, favorable MT and ML scores.

Runtime analysis. As DRT is a post-processing approach applicable to any detector, it is inevitable that it is slower than the baselines. Without any speed optimizations, DRT achieves 6.8 fps (ResNet50 backbone) and 8.0 fps (ResNet34). From the runtime ablation in Table 2 we see that the computation time is dominated by DRT-net, whereas the motion model imposes no bottleneck. Also considering the feature extraction backbone results in the reported frame rates shown in Table 4. To further improve the speed, the proposed ideas could be integrated into a detector, turning it into a one-stage refinement process. Since our focus is on detector-agnostic improvements, we will consider this in our future research.

5 Conclusion

In this paper, we propose a new tracking method, DRT, to handle inaccurate and missing detections in MOT. Our DRT consists of DRT-net and an LSTM-based motion model. DRT-net refines the location of detected boxes and recovers occluded objects. The motion model is used to recover lost targets. With well-organized training data and a carefully designed training scheme, our new tracking method achieves state-of-the-art performance. Extensive experiments on publicly available MOT datasets demonstrate the benefits of DRT and that it can improve the results of even the best MOT methods.

Acknowledgement

This work was supported in part by China Scholarship Council (NO. 201906840048), the National Science Foundation of China (61801222) and the Nature Science Foundation of Jiangsu Province (BK20191284). This work was also partially funded by the Austrian Research Promotion Agency (FFG) under the project INTERACT (879648) and we gratefully acknowledge the financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association.

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social Lstm: Human Trajectory Prediction in Crowded Spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without Bells and Whistles. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[3] Keni Bernardin and Rainer Stiefelhagen. Evaluating Multiple Object Tracking Performance: The Clear Mot Metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

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

[5] Guillem Brasó and Laura Leal-Taixé. Learning a Neural Solver for Multiple Object Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] Zhaowei Cai and Nuno Vasconcelos. Cascade R-cnn: Delving into High Quality Object Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] Long Chen, Haizhou Ai, Zijie Zhuang, and Chong Shang. Real-Time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[8] Peng Chu, Heng Fan, Chiu C Tan, and Haibin Ling. Online Multi-Object Tracking with Instance-Aware Tracker and Dynamic Model Refreshment. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[9] Xuangeng Chu, Anlin Zheng, Xiangyu Zhang, and Jian Sun. Detection in Crowded Scenes: One Proposal, Multiple Predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[10] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep Learning in Video Multi-Object Tracking: A Survey. *Neurocomputing*, 381:61–88, 2020.

[11] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object Detection via Region-based Fully Convolutional Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[12] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient Convolution Operators for Tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent Autoregressive Networks for Online Multi-Object Tracking. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[14] Shoudong Han, Piao Huang, Hongwei Wang, En Yu, Donghaisheng Liu, Xiaofeng Pan, and Jun Zhao. Mat: Motion-Aware Multi-Object Tracking. *arXiv preprint arXiv:2009.04794*, 2020.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[16] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(3):583–596, 2014.

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Rivisiting Single Image Depth Estimation: Toward Higher Resolution Maps with Accurate Object Boundaries. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[19] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82:35–45, 1960.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2016.

[22] Yuan Li, Chang Huang, and Ram Nevatia. Learning to Associate: Hybridboosted Multi-Target Tracker for Crowded Scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single Shot Multibox Detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[25] 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.

[26] Bo Pang, Yizhuo Li, Yifan Zhang, Muchen Li, and Cewu Lu. Tubetk: Adopting Tubes to Track Multi-Object in a One-Step Training Model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[28] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Chained-Tracker: Chaining Paired Attentive Regression Results for End-to-End Joint Multiple-Object Detection and Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-cnn: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[30] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning Social Etiquette: Human Trajectory Understanding in Crowded Scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[31] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[32] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-End People Detection in Crowded Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] Siyu Tang, Mykhaylo Andriluka, and Bernt Schiele. Detection and Tracking of Occluded People. *International Journal of Computer Vision (IJCV)*, 110(1):58–69, 2014.

[34] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social Attention: Modeling Attention in Human Crowds. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[35] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-Object Tracking and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] Bisheng Wang, Guo Cao, Licun Zhou, Youqiang Zhang, and Yanfeng Shang. Task Differentiation: Constructing Robust Branches for Precise Object Detection. *Computer Vision and Image Understanding (CVIU)*, 199:103030, 2020.

[37] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards Real-Time Multi-Object Tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[38] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2017.

[39] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to Detect and Segment: An Online Multi-Object Tracker. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[40] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[41] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to Track: Online Multi-Object Tracking by Decision Making. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[42] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-Temporal Relation Networks for Multi-Object Tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[43] Fan Yang, Xin Chang, Sakriani Sakti, Yang Wu, and Satoshi Nakamura. Remot: A Model-Agnostic Refinement for Multiple Object Tracking. *Image and Vision Computing*, 106:104091, 2021.

[44] Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, and Junjie Yan. Poi: Multiple Object Tracking with High Performance Detection and Appearance Feature. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[45] Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the Gap between Training and Inference for Neural Machine Translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2019.

[46] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[47] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the Fairness of Detection and Re-identification in Multiple Object Tracking. *International Journal of Computer Vision*, pages 1–19, 2021.

[48] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[49] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[50] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online Multi-Object Tracking with Dual Matching Attention Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.