A General Recurrent Tracking Framework without Real Data

Shuai Wang[1,2], Hao Sheng[1,2,*], Yang Zhang[3], Yubin Wu[1,2], Zhang Xiong[1]

[1]State Key Laboratory of Software Development Environment, School of Computer Science and Engineering, Beihang University, Beijing 100191, P.R.China
[2]Beihang Hangzhou Innovation Institute Yuhang, Beihang University, Xixi Octagon City, Yuhang District, Hangzhou 310023, P.R.China
[3]College of Information Science & Technology, Beijing University of Chemical Technology, Beijing 100029, P.R.China
{shuaiwang, shenghao, yubin.wu, xiongz}@buaa.edu.cn, yang_zh@mail.buct.edu.cn

Abstract

Recent progress in multi-object tracking (MOT) has shown great significance of a robust scoring mechanism for potential tracks. However, the lack of available data in MOT makes it difficult to learn a general scoring mechanism. Multiple cues including appearance, motion and etc., are limitedly utilized in current manual scoring functions. In this paper, we propose a Multiple Nodes Tracking (MNT) framework that adapts to most trackers. Based on this framework, a Recurrent Tracking Unit (RTU) is designed to score potential tracks through long-term information. In addition, we present a method of generating simulated tracking data without real data to overcome the defect of limited available data in MOT. The experiments demonstrate that our simulated tracking data is effective for training RTU and achieves state-of-the-art performance on both MOT17 and MOT16 benchmarks. Meanwhile, RTU can be flexibly plugged into classic trackers such as DeepSORT and MHT, and makes remarkable improvements as well.

1. Introduction

Multi-object tracking (MOT) has attracted a lot of attention for its wide application such as surveillance, cell analysis and autonomous driving. Tracking-by-detection has emerged as the preferred paradigm to solve MOT for its simple pipeline: (i) detecting object locations, (ii) form tracks by linking corresponding detections across time [2]. In particular, the performance of the linking step, or the data association highly depends on a robust scoring mechanism.

Currently, the scoring mechanisms in most approaches only take account of the appearance feature extracted from the cropped object patches [29]. The tracking accuracy of these methods is limited due to the following reasons. Firstly, objects across frames suffer from frequent occlusions and pose variation, which leads to much noise in the appearance features. Secondly, much long-term information is lost since they can not consider the temporal relationship. Even some works [10, 18, 32, 27] utilize multiple cues such as appearance and motion, the manual scoring function is unsatisfactory when the objects are missing or occluded. Moreover, the parameters of these methods should be adjusted for new scenes. A general scoring mechanism is urgently needed to adapt to various scenarios and tracking frameworks.

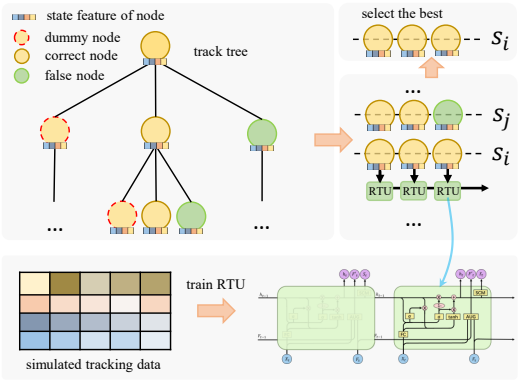


Figure 1. The processing procedure of our methods. RTU is trained on simulated data first. After building the track tree, each possible association is evaluated by RTU to get the score. Then the tracker selects the best one.

To solve this, neural network based scoring methods have attracted many researchers. For example, Kim *et .al* [11] use a specialized gating mechanism in RNN to aggre-

gate information and score tracks over time. Sheng *et al.* [19] present the STCCNet based on spatial-temporal attention to restrain the occlusion in tracklets. They indicate that their method is able to get more accurate scores. However, these methods suffer from the lack of available data. Because of the diversity of the datasets (MOT15˜20), these methods should be retrained for each dataset. Even when using another feature extractor, they have to train the model again for adapting new types of features.

To solve the problems mentioned above, a general tracking framework is presented by introducing three types of basic nodes to reconstruct the data association, as shown in Fig. 1. Based on this, a transplantable Recurrent Tracking Unit (RTU) is proposed to score tracks by utilizing multiple long-term cues. In addition, we explore the distribution of real data and generate simulated tracking data to train RTU avoiding the lack of real data. Experimental results indicate that our method achieves state-of-the-art performance on both MOT17 and MOT16 benchmarks, and our RTU trained by simulated tracking data improves the other trackers significantly.

This paper presents three main contributions:

- Construct a general Multiple Nodes Tracking framework, which can be extended to most trackers and adapts to various scenarios.
- Propose RTU to solve the scoring problem of tracks, which integrates multiple cues in a simple but efficient way. RTU is also flexibly plugged into other trackers to bring improvement.
- Present a method for generating simulated tracking data to replace real data. Based on MNT and RTU, it is easy to fuse kinds of cues in this simulated data.

2. Related Work

Tracking-by-Detection. Most famous MOT methods [9, 5, 30, 17, 24] belong to the tracking-by-detection paradigm, which heavily depends on the performance of the underlying detector. The progress of detector have greatly boosted researches in MOT, but it still remains a challenging task, especially in crowded scenarios. Generally, previous works are classified into online methods and batch methods.

Online methods only utilize past information in data association, and usually reach real-time performance. For example, SORT [4] creates Kalman filter for each past track to estimate their state, and then associations new detections with tracks by motion metric. Further, DeepSORT [27] introduces a ReID network for measuring appearance similarity, while motion information is used to gate. Both these two methods solve the data association by the bipartite graph. Currently, new online methods such as JDE [26] and Fair [31] achieve SOTA performance on the benchmark, but they still adopt a similar association scheme as DeepSORT. Their improvement benefits from their powerful detectors more. Our method also follows tracking-by-detection but mainly focuses on the data association scheme.

Batch methods can refer both past and future information to associate detections. Because they take account of more cues, their permformance is better than online methods. Milan *et al.* [15] combine multiple cues by a conditional random field method. Kim *et al.* [10] revise MHT in object tracking with a powerful appearance model. Sheng *et al.* [18] extend MHT to an iterative paradigm by tracklet-level association. In order to utilize the long-term information, Zhang *et al.* [34] design an LSTM-based appearance model and indicate that the training data is important.

Scoring Mechanism. It is common to formulate tracking as a graph, where each detection corresponds to a node and each edge indicates a possible link [2]. The data association can then be formulated as maximum flow [1], maximum weight independent set, or bipartite graph with either fixed costs (scores) based on similarity or combinational costs (scores) including appearance, motion, and etc.

Online methods prefer the bipartite graph because of its efficiency. Typically, the appearance similarity is regarded as the score of each edge in the bipartite graph based on methods [4, 27, 26]. Then this graph is solved by the Hungarian algorithm with motion gating. Because the exploitation of appearance and motion cues is too simple, these methods perform worse in long-term tracking. Batch methods consider various cues more richly. For instance, MHT_DAM [10] takes a manual scoring function to combine appearance and motion information. eHAF [20] introduces the interaction cues to solve the identity switch problem in crowded scenes.

Nerual network based methods [23, 37, 8] have shown their competitive results recently. TLMHT [34] trains an LSTM to score possible tracks and the result indicates that their model performs better in long-term tracking. Xv *et al.* [29] design a new feature representation to score the association. Kim *et al.* [11] investigate what is stored in RNN during scoring and introduce a specialized gating mechanism in RNN. Although these methods have shown their potential, they are bound by the available data, i.e., the training data is limited in current datasets. And it is inconvenient to apply them in other methods, because they are trained on the data with different distribution.

Our approach is motivated by these works. In order to design a general scoring mechanism that can utilize multiple long-term cues, we present a general tracking framework that can be extended to most trackers. Then we propose the transplantable RTU to integrate kinds of information for scoring tracks. Finally, we present a method to generate simulated tracking data, which can be used to train RTU or similar networks.

3. Method

3.1. The Multiple Nodes Tracking Framework

Popular tracking methods such as MHT_DAM [10], DeepSORT [27] and Fair [31] follow the tracking-by-detection paradigm, which regards the tracking problem as the association between detections. Different detections are linked to build track proposals and the trackers finally select the best proposals to represent objects.

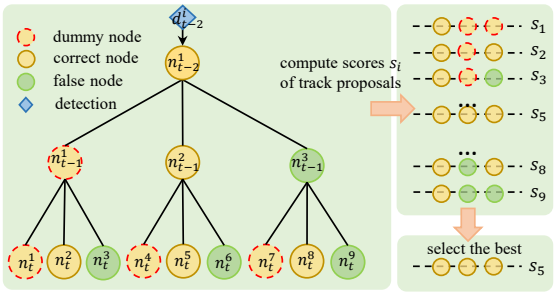


Figure 2. The processing steps of the Multiple Nodes Tracking framework (MNT).

In this paper, we propose the Multiple Nodes Tracking framework (MNT), which follows the tracking-by-detecting paradigm as well. Different from the aforementioned works, MNT links three types of basic nodes to build trajectories instead of detections. The correct nodes and false nodes represent the correctly and falsely linked detections respectively. We also design the dummy nodes to represent the missing detections in case of heavy occlusion.

As shown in Fig. 2, each object is represented by a track tree that contains various nodes. Let F_t denotes the t^{th} frame in a video. Assuming that there is one non-associated detection d^i_{t-2} in frame F_{t-2}, it will be initialized as the root node n^1_{t-2} for target O_i. For the new frame, the track tree is extended by linking new basic nodes to existing nodes to build new branches. Each branch represents a potential track proposal. The nodes are classified by RTU (discussed in Section.3.2). Specifically, the correct node corresponds to the correct association between detection and target O_i while the false nodes correspond to the false association (only one false node is drawn in Fig. 2). In addition, a dummy noded is created to denote missing detection. The target is regarded as lost once it has $N_{max} = 10$ consecutive dummy nodes. Repeating this progress, a track tree is constructed. In order to solve the exponential growth of the track tree, we adopt the pruning strategy in [10]. Firstly, each branch is scored by RTU. Then the tree can be solved by graph theory. The tree is pruned when the optimal solution is found.

This is a universal tracking framework that adapts to most tracking methods such as MHT_DAM and DeepSORT.

3.2. Recurrent Tracking Unit

In MNT, each branch represents a potential track proposal. It is a core for tracking to score the proposals and select the best one. Previous works [10, 18, 6, 20, 34, 15] design manual scoring function via appearance and motion information, or directly use the appearance similarity as the score. These manual functions can only use short-term information, but trajectories are long-term sequences containing rich temporal information. Motivated by the progress of the Recurrent Neural Network in sequence data, we propose the Recurrent Tracking Unit (RTU) in order to solve such problems.

The track proposal is represented by a node sequence in unfixed size. Each track proposal is updated recurrently in each time step. So the procedure of track proposal generation can be defined as follows:

$$\begin{aligned} node &\Rightarrow track \\ track \cup node &\Rightarrow new\ track \\ &\Rightarrow track \end{aligned} \tag{1}$$

As shown in Eq. 1, one node can be defined as a track, because sometimes a target only appears once. The union of an old track and a new node is defined as a new track. So the node can be the basic recurrent unit of a track.

To record information of each node, we introduce the state feature into MNT. According to previous works [10, 11], appearance and motion information is essential for scoring track proposal. Instead of the appearance feature, this paper sets the appearance similarity between parent node and new detection as the first dimension of state feature. The influence of different feature distributions can be decreased by utilizing similarity, since different types of appearance features still ensure the high similarity of the same target. In addition, it can decrease the potential distribution of the state feature due to its concise representation. The second dimension is the motion consistency which reflects the space distance or motion relationship between the parent node and new node. Both appearance similarity and motion consistency are normalized to [0, 1]. Considering that detection missing is another important factor, a dummy indicator is introduced to the third dimension of the state feature. It is a boolean variable, and is set as 1 when the node is dummy otherwise 0. We find that some worse detections may result in tracking failures such as false alarm or identity switches. Thus, the confidence of detections is integrated into the fourth dimension of the state feature, which is normalized to [0, 1]. For a track proposal, it can be represented by a sequence of state features. So the state feature, as the record of tracking information, is input to RTU.

Fig. 3 shows the graphical depiction of RTU. For time steps t, RTU takes four inputs: the old appearance feature template F_{t-1}, the appearance feature F_t of the current

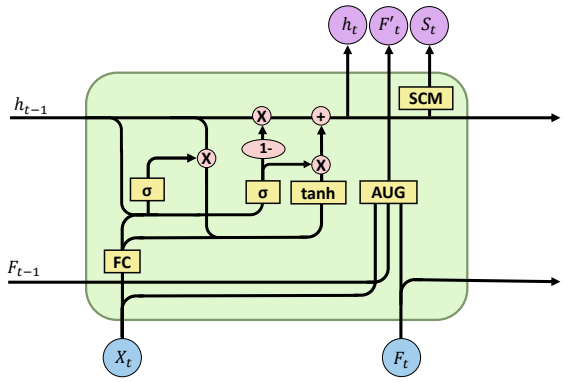


Figure 3. The structure of RTU.

node, the old hidden state h_{t-1} and the state feature X_t of the current node. In the first time step, the appearance feature template is initialized the same as appearance feature F_0 while the hidden state is initialized to zero.

First, RTU projects the state feature to the hidden space. Then the $reset$ gate r is computed by:

$$\begin{aligned} X'_t &= W_x X_t \\ r_t &= \sigma(W_r[h_{t-1}, X'_t] + b_r) \end{aligned} \tag{2}$$

where σ is the sigmoid function, W_r and W_x are weight matrices which are learned, b_r represents the bias.

Similarly, the $update$ gate z is computed by:

$$z_t = \sigma(W_z[h_{t-1}, X'_t] + b_z) \tag{3}$$

where W_z is a learned weight matrix, b_z is the bias.

The update of the hidden state is computed by:

$$\begin{aligned} \hat{h}_t &= tanh(W_{\hat{h}}[r_t * h_{t-1}, X'_t]) \\ h_t &= (1 - z_t)h_{t-1} + z_t\hat{h}_t \end{aligned} \tag{4}$$

In this formulation, the $reset$ gate chooses to ignore the previous hidden state and reset with the current input only.

In practice, it is common that some regions of an object suffer from occlusion, blur and varied postures. In the occluded regions, the raw appearance feature is easily polluted. However, the appearance information remembered from regions of the same position in previous frames can help to recover the polluted information [13]. This paper design the Appearance Update Gate (AUG) in RTU to refine the appearance feature, which is defined as follow:

$$F'_t = F_{t-1} * X^1_t + (1 - X^1_t) * F_t \tag{5}$$

where X^1_t represents the first dimeneion of X_t.

Finally, the hidden state is projected to the score S_t, after the Score Compute Module (SCM). It is formulated as:

$$S_t = W^2_s(relu(W^1_s h_t + b^1_s)) + b^2_s \tag{6}$$

where W^1_s and W^2_s are learned weight matrices, b^1_s and b^2_s are bias.

3.3. Simulated Tracking Data Generation

Although many deep learning-based methods have been proposed to resolve the problem of scoring potential tracks, they may perform worse when facing a new scenario. In other words, they can not generalize satisfactorily in new dataset since they are influenced by the style of the training dataset. In addition, their training data is limited by the scale of the dataset used.

In our tracking framework, each potential track is a node sequence. It is simple to generate artificial track proposals by randomly arranging different nodes. Although this way ensures that most possible track proposals are covered, the scale of sample space grows exponentially as the track length increases. In order to repress this, this paper proposes a conditional sample strategy.

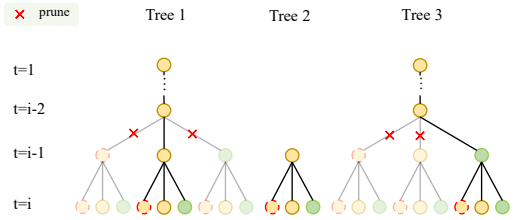


Figure 4. A pruning process.

First, we take account of the distribution of different types of nodes during tracking. Pruning is an essential step for most trackers due to the exponential increase in the number of potential track proposals. The pruning has a great influence on the distribution of track proposals. As shown in Fig. 4, an N-scan (N=2) pruning approach [10] is adopted, the tracker traces back to the node at frame $t-N$ and prunes the other branches that diverge from the selected branch at that node. So only one of the nodes born in time $t = i - 1$ survives after pruning. This procedure is looping until stop tracking. The pruned branches (drawn transparent in Fig. 4) are nonexistent for the future nodes (at time $t > i$) after pruning at $t = i$. Because once remove other branches, the track proposal from frames 1 to $i - N$ is confirmed. In other words, this confirmed part T_{cof} nearly fits a conditional probability distribution:

$$\begin{aligned} p(T_{cof}) &= p(n_l \cup n_{l-1} \cup n_{l-2} \cup ... \cup n_1) \\ &= \prod_{i=1}^{l} p(n_i | n_{i-1}, ..., n_1) \end{aligned} \tag{7}$$

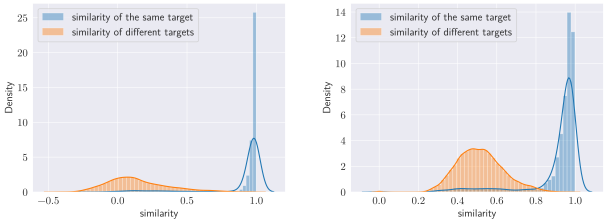
where $p(.)$ reprents the probability, i means the current time step, l is the length of the confirmed part of the track.

However, the pruning is not always correct, sometimes it keeps a false node like tree 3 in Fig. 4. Hence we suppose that the accuracy of pruning is α, and the probability of a

track composed of entire correct nodes is:

$$p(T) = p(c_l \cup c_{l-1} \cup c_{l-2} \cup ... \cup c_1) = \prod_{i=1}^{l} \alpha \quad (8)$$

Based on this distribution, the sample space is repressed to generate simulated tracking data more efficiently. For each node, we introduce a state feature to record information in Section.3.2. Hence, we can generate different nodes by adjusting their state features.



(a) Fair feature. (b) PCB feature.

Figure 5. The similarity distribution of two kinds of features.

As described in Section.3.2, the state features contain four types of information. The most important information is appearance similarity, which reflects the trend of object appearance to some extend. The appearance feature from different extractors usually is of great variety in distribution, which may lead to the different distribution of appearance similarity. We explore the common behind them and finds some inspiring phenomena. As shown in Fig. 5, the blue line represents the appearance similarity of the same target while the orange line denotes the similarity of different targets. In Fig. 5(a), with Fair [31] feature, we find that the appearance similarity of the same target follows normal distribution approximately. Meanwhile, the similarity of different targets follows another normal distribution with a different mean value. When using PCB [21] feature, the similarity of the same targets is still in an approximately normal distribution. Although there is a huge gap between the similarity distribution of different features, the statistical results show that the same target maintains a high appearance similarity (bigger than 0.8) with different features. Similarly, we observe the distribution of normalized Euclidean distance and find that it is a normal distribution as well. As for the confidence in state feature, it lies more randomly within the range of [0, 1]. According to these observations, it is easy to generate different nodes. For example, a simulated correct node can be built with a high appearance similarity and high Euclidean distance.

Further, the corresponding label of simulated data is another core for training RTU. In order to solve this, we introduce a reward mechanism to ensure the order of data. We assign different rewards to those nodes and calculate the total reward of each track. Obviously, the correct nodes should deserve the biggest reward r_c, while the dummy and the false get less reward r_d and r_f. Exactly, the reward for the false nodes is called "penalty" instead. The total reward of a track can be regarded as its score. Hence the training label, that is, the ground truth score of each simulated track proposal is computed by :

$$\begin{aligned} label =& r_c \sum_{t=1}^{L} \mathbb{1}_c[X_t] * (X_t^{(1)} + X_t^{(2)}) \\ &+ r_d \sum_{t=1}^{L} \mathbb{1}_d[X_t] * (X_t^{(1)} + X_t^{(2)}) \\ &- r_f \sum_{t=1}^{L} \mathbb{1}_f[X_t] * (2 - X_t^{(1)} - X_t^{(2)}) \end{aligned} \quad (9)$$

where * represents the element-wise product, $\mathbb{1}_c[.]$ is a boolean variable that is 1 when X_t belongs to a correct node otherwise is 0. $\mathbb{1}_d[.]$ and $\mathbb{1}_f[.]$ are the corresponding boolean variable for the dummy node and false node. $X_t^{(i)}$ denotes the i^{th} dimension of X_t.

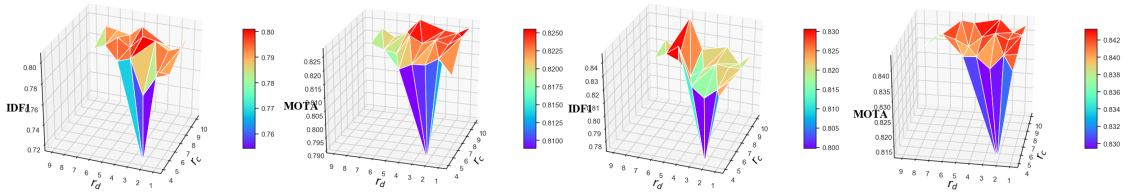
This design ensures that the track proposals are strictly ordered by the total reward, which is of greate benefit to select the best track. With this label, RTU can be trained by mean squared error (MSE) loss in an end-to-end way.

4. Experiments

In this section, the implementation details are described first and then the hyperparameter searching experiments are shown. Next, the ablation study experiments are conducted to demonstrate the effectiveness of our methods. In addition, we plug RTU into other trackers to valid its transportability in effect analysis experiments. Finally, we report the results of the proposed approach compared to other state-of-the-art tracking methods based on MOT17 and MOT16 datasets.

4.1. Implementation Details

Network Architecture and Training. It is discussed by Kim *et al.* [11] that the hidden state dimension $d_h = 512$ has a good performance when processing appearance and motion information. So in RTU, the hidden state is of dimension 512. Two fully connected layers with dimension 1024 are applied after updating the hidden state, which are used to compute the final score (see Section.3.2). During training, we use the standard Adam optimizer with an initial learning rate of 1×10^{-4}. The pruning accuracy α is set as 0.8 according to the statistic result from trackers. RTU is trained on the simulated tracking data without any other dataset. In each epoch, we randomly generate different length training data from min length $L_{min} = 2$ to max



(a) IDF1 surface (PCB) (b) MOTA surface (PCB) (c) IDF1 surface (Fair) (d) MOTA surface (Fair)

Figure 6. The surfaces of IDF1 and MOTA in searching space. In (a) and (b), the tracker utilizes PCB [21] as the appearance feature extractor while use Fair [31] in (c) and (d).

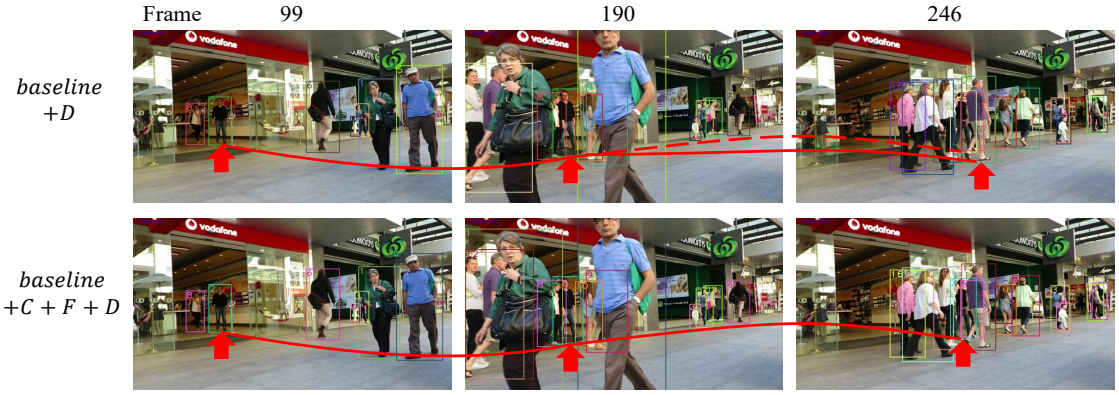


Figure 7. The tracking result of MOT17-09. The red arrows point to the linked detections and the red line represents the association while the red dotted line denotes the correct association.

length $L_{max} = 355$. To accelerate the training process, all the training data is padding to the max length of 355.

Datasets and Evaluation Metrics. Two MOT Benchmarks [14] are utilized for evaluation in our experiments. These benchmarks are widely used in multi-object tracking to evaluate different trackers for the large difference in illumination camera motion and etc. MOT17 has 7 training sequences and 7 testing sequences. These sequences have different frame rates, resolutions, viewpoints and etc. For quantitive evaluation, the CLEAR MOT [3] metrics are adapted in this paper, including MOTA↑ (multiple object tracking accuracy), IDF1↑ (IDF1 score), FP↓ (false positive), FN↓ (false negatives) and IDs↓ (identity switches). MOTA mainly reflects the most tracks are tracked or not, which is easily affected by the total number of detections. IDF1 concerns whether a target is labeled with a correct ID.

4.2. Hyperparameter Search

There are three important hyperparameters in our methods, so we search the parameters space to maximize IDF1 and MOTA. Accourding to the formulation in Section.3.3, it is obvious that the reward r_c of correct nodes should be bigger than the reward r_d of the dummy. If we divide both sides of Eq. 9 by r_f ($r_f > 0$), then the order of label is stable and r_f is reduced. Based on this rule and some experience, the hyperparameter searching space can be determined: $r_c \in [2, 10], r_d \in [1, r_c)$.

In order to acquire more robust parameters, we repeat the hyperparameters searching experiments with different appearance features. First, the tracker obtains features from PCB. As shown in Fig. 6 (a) and (b), when $r_c = 2$ and $r_d = 1$, both IDF1 and MOTA are lowest. As r_c increases, MOTA and IDF1 climb rapidly. When adjusting r_d, the results show that MOTA is more stable since it relies on detection quality more. But IDF1 first increase and then drops.

Setting	IDF1↑	MOTA↑	IDs↓	FP↓	FN↓
baseline	63.1	78.4	809	1330	22034
+$\mathcal{D}$	78.3	82.2	503	6044	13359
+$\mathcal{C}$+$\mathcal{F}$+$\mathcal{D}$	80.6	83.2	494	5323	13706

Table 1. Ablation study of various settings of node types and scoring mechnism.

Similar phenomenon can be observed with features from Fair [31]. As shown in Fig. 6 (c) and (d), IDF1 and MOTA are at the lowest point when r_c and r_d are too small. When tuning r_c, we find that MOTA improves significantly at begin. But the IDF1 surface presents slightly differently with Fig. 6 (a). It reaches the highest with $r_d = 5$, but in Fig.

	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT%↑	ML%↓	Frag↓	MOTP↑
MHT_DAM [10]	76.2	83.1	4566	13667	659	66.8	13.5	886	81.6
+ RTU	79.4	83.5	4344	13714	457	66.0	12.7	767	81.7
MLT [33]	77.8	81.8	5245	14621	480	61.5	16.7	822	81.7
+ RTU	78.7	83.9	3928	13653	522	63.7	13.7	907	81.7
DeepSORT [27]	72.5	83.6	2841	14811	691	61.7	5.9	2011	81.1
+ RTU	76.3	83.7	2814	14826	681	62.2	6.1	2010	81.1

Table 2. The effect analysis experiments on MOT17 training dataset. All trackers acquire appearance features from PCB[21].

Method	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT%↑	ML%↓	Frag↓	MOTP↑
MHT_DAM [10]	79.4	84.0	5757	11530	626	71.1	10.6	1018	81.6
+ RTU	79.6	84.1	5651	11398	760	71.3	10.4	1143	81.6
MLT [33]	80.7	83.9	5745	11750	519	66.3	13.9	960	81.5
+ RTU	82.2	85.6	4003	11703	430	68.7	12.2	946	81.7
DeepSORT [27]	75.6	80.8	7117	13690	673	61.1	10.4	848	81.6
+ RTU	81.4	83.5	2564	15130	768	60.8	12.5	1964	82.0

Table 3. The effect analysis experiments on MOT17 training dataset. All trackers acquire appearance features from Fair [31].

6 (a) it reaches the highest with $r_d = 4$. Hence we adopt $r_c = 10, r_d = 4.5$ in other experiments.

4.3. Ablation Study

As shown in Tab. 1, we conduct the ablation study experiment to validate the effectiveness of each component. In the ablation study, all experiments use the same detection provided by [31]. The baseline is a naive MHT framework, which is modified from [18]. It only implements data association, pruning and optimization. The division of three types of nodes is not included. In addition, the baseline uses the same manual scoring function as [18]. We add dummy nodes ($\mathcal{D}$) to the baseline and still use the manual scoring function. The result shows that IDF1 increases from 63.1 to 78.3, because dummy nodes can represent the missing detections and the fragmented tracks are linked to a complete trajectory. A significant reduction in FN is observed, indicating that most false negatives are suitably estimated by dummy nodes. Trajectories are more completed and smoother than baseline, so MOTA improves to 82.2.

Further, the manual score function is replaced with RTU to classified correct ($\mathcal{C}$) and false nodes($\mathcal{F}$). As shown in Fig. 7, the tracker (baseline+$\mathcal{D}$) associates the man with ID 2 falsely due to the heavy occlusion at 246^{th} frame. But after adding RTU (baseline +$\mathcal{C}$+$\mathcal{F}$+$\mathcal{D}$), the tracker refinds the correct association. The results in Tab. 1 show that it is obvious that IDF1 and MOTA raise by 2.3 and 1 respectively. Because RTU can classify the correct nodes and false nodes according to historical information, the track proposals containing false nodes get lower scores. The sum of FP and FN decrease 374, which ensures the unity of targets' identity. Hence MOTA reaches 83.2 and IDF1 improves to 80.6.

4.4. Effect Analysis

To demonstrate the effectiveness and university of our method, we choose three famous open-sourced trackers to perform a series of experiments. The scoring procedures of these trackers are replaced with RTU. In addition, two kinds of appearance feature are adopted in these experiments to prove the robustness of our approach. The first is proposed by Zheng *et al.* [21] (denoted by PCB), and is only trained on Market [35]. The second is designed by Zhan *et al.* [31], which is trained on many datasets such as CityPersons [32], Caltech Pedestrian[7], and MOT17 [14]. All the corresponding results are listed in Tab. 2 and 3.

MHT_DAM is a classic tracker based on a manual scoring mechanism. It is obvious that adding RTU into MHT_DAM substantially improves the IDF1 and MOTA by 3.2 and 0.4 respectively, as shown in Tab. 2. Even when using another appearance feature, MOTA and IDF1 improve as well.

MLT conducts the association among different length tracklets, which is of great difference with frame-level methods. We replace its score module with RTU accordingly and the results improve as well. As shown in Tab. 2, when adding RTU to MLT, the IDF1 indicator increases by 0.9 and MOTA reaches 83.9. This result indicates that RTU is still adaptable to tracklet-based methods, since the association among tracklets still can be described by MNT.

DeepSORT is a famous real-time tracker, which is the basis of many new trackers such as Fair [31]. It solves the association by the bipartitegraph so performs worse than others in IDF1. Here, dummy nodes are introduced into DeepSORT and the weight (score) in bipartite is computed

Method	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT%↑	ML%↓	Frag↓	MOTP↑
TubeTK [16]	58.6	63.0	27060	177483	5137	31.2	19.9	5727	81.1
GSDT [25]	68.7	66.2	43368	144261	3318	40.8	18.3	8046	79.9
CenterTrack [36]	64.7	67.8	**18498**	160332	3039	34.6	24.6	6102	78.4
TraDes [28]	63.9	69.1	20892	150060	3555	36.4	21.5	4833	78.9
MLT [33]	70.1	69.3	35844	135888	**1347**	38.5	39.0	2028	81.3
CSTrack [12]	72.9	74.9	23847	114303	3567	41.5	17.5	7668	80.9
Fair [31]	72.3	73.7	27507	117477	3303	43.2	**17.3**	8073	81.3
ours	**75.0**	**74.9**	32007	**107616**	1812	**49.7**	18.9	**1824**	**81.7**

Table 4. Tracking performance on MOT17 benchmark dataset.

Method	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT%↑	ML%↓	Frag↓	MOTP↑
TubeTk [16]	62.2	66.9	11544	47502	1236	39.0	16.1	1444	78.5
TraDes [28]	64.7	70.1	8091	45210	1144	37.3	20.0	1575	78.8
LMP [22]	70.0	71.0	**7880**	44564	**434**	46.9	21.9	**587**	80.2
Fair [31]	72.8	74.9	10163	34484	1074	44.7	**15.9**	2567	81.2
CSTrack [12]	73.3	75.6	9646	33777	1121	42.8	16.5	2450	80.8
ours	**76.0**	**76.5**	11438	**30866**	584	**51.5**	17.0	597	**81.6**

Table 5. Tracking performance on MOT16 benchmark dataset.

by RTU. As shown in Tab. 2, RTU boosts the performance of DeepSORT in almost all indicators. DeepSORT prunes at each frame so that loses much available information. But with RTU, historical information is able to be recurrently stored in the node. So even it is a bipartitegraph-based method, it still can utilize long-term information via RTU.

As shown in Tab. 3, when we take the feature design in [31], all the trackers obtain remarkable improvements by RTU as well. This indicates that not only RTU is effective to other trackers, but also the proposed simulated tracking data is robust and efficient, so it is easy to embed RTU into other trackers.

4.5. State-of-the-art Comparison

In Tab. 4 and Tab. 5, we compare our tracking framework with other latest published trackers on the MOT17 and MOT16 benchmarks. We use the same model and setting as Section.4.2 as our final method for evaluating on the benchmarks. The input detection is provided by [31]. The results show that our methods conduct state-of-the-art performance among other methods.

As shown in Tab. 4, we achieve the highest IDF1 by 75.0, 2.7 higher than the second results given by Fair, because our RTU restrains the identity switches and fragmentation. Since RTU can classify false nodes and correct nodes according to long-term information, our tracker reaches the best IDF1 by 74.9. Meanwhile, the dummy node in MNT is effective for detection missing under occlusion. Although some unprecise dummy nodes are regarded as FPs, those precise ones can complement the missing detections suitably. So our method achieves the lowest FN.

Tab. 5 shows the results compared on MOT16. We still acquire the best performance on IDF1 and MOTA. The IDs indicator of ours is half of CSTrack and Fair. Because their association scheme is easy to make tracking failure over a long time, especially in crowded scenes. So our result is smoother and decreases Frag about 2000 compared to them.

5. Conclusions

This paper presents a general tracking framework composed of multiple nodes. To efficiently combine various information, an extendable state feature is designed for each node. Based on this, we introduce RTU to combine information over a long time and score possible tracks. In addition, we propose a method for generating simulated tracking data to solve the lack of data in MOT. The experimental results prove that RTU can be flexibly plugged into other tracks and bring obvious improvements. Meanwhile, our methods conduct state-of-the-art performance on both MOT17 and MOT16 datasets.

Acknowledge

This study is partially supported by the National Key R&D Program of China (No.2018YFB2100800), the National Natural Science Foundation of China (No.61861166002) , and the Science and Technology Development Fund, Macau SAR (File no.0001/2018/AFJ) and the Open Fund of the State Key Laboratory of Software Development Environment (No. SKLSDE-2021ZX-03), and the Fundamental Research Funds for the Central Universities (buctrc202123). Thank you for the support from HAWKEYE Group.

References

[1] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1806–1819, 2011.

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951, 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 *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 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*, pages 6247–6257, 2020.

[6] Jiahui Chen, Hao Sheng, Yang Zhang, and Zhang Xiong. Enhancing detection model for multiple hypothesis tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–27, 2017.

[7] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009.

[8] Kuan Fang, Yu Xiang, Xiaocheng Li, and Silvio Savarese. Recurrent autoregressive networks for online multi-object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 466–475. IEEE, 2018.

[9] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *International Conference on Machine Learning*, pages 4364–4375. PMLR, 2020.

[10] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision*, pages 4696–4704, 2015.

[11] Chanho Kim, Fuxin Li, and James M Rehg. Multi-object tracking with neural gating using bilinear lstm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 200–215, 2018.

[12] Chao Liang, Zhipeng Zhang, Yi Lu, Xue Zhou, Bing Li, Xiyong Ye, and Jianxiao Zou. Rethinking the competition between detection and reid in multi-object tracking. *arXiv preprint arXiv:2010.12138*, 2020.

[13] Yiheng Liu, Zhenxun Yuan, Wengang Zhou, and Houqiang Li. Spatial and temporal mutual promotion for video-based person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8786–8793, 2019.

[14] 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.

[15] Anton Milan, Konrad Schindler, and Stefan Roth. Multi-target tracking by discrete-continuous energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2054–2068, 2015.

[16] 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*, pages 6308–6318, 2020.

[17] Jinlong Peng, Tao Wang, Weiyao Lin, Jian Wang, John See, Shilei Wen, and Erui Ding. Tpm: Multiple object tracking with tracklet-plane matching. *Pattern Recognition*, 107:107480, 2020.

[18] Hao Sheng, Jiahui Chen, Yang Zhang, Wei Ke, Zhang Xiong, and Jingyi Yu. Iterative multiple hypothesis tracking with tracklet-level association. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(12):3660–3672, 2018.

[19] Hao Sheng, Shuai Wang, Yang Zhang, Dongxiao Yu, Xiuzhen Cheng, Weifeng Lyu, and Zhang Xiong. Near-online tracking with co-occurrence constraints in blockchain-based edge computing. *IEEE Internet of Things Journal*, 2020.

[20] Hao Sheng, Yang Zhang, Yubin Wu, Shuai Wang, Weifeng Lyu, Wei Ke, and Zhang Xiong. Hypothesis testing based tracking with spatio-temporal joint interaction modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(9):2971–2983, 2020.

[21] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In *Proceedings of the European conference on computer vision (ECCV)*, pages 480–496, 2018.

[22] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3548, 2017.

[23] Fan Wang, Lei Luo, and En Zhu. Two-stage real-time multi-object tracking with candidate selection. In *International Conference on Multimedia Modeling*, pages 49–61. Springer, 2021.

[24] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490, 2019.

[25] Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint object detection and multi-object tracking with graph neural networks. *arXiv preprint arXiv:2006.13164*, 5, 2020.

[26] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2(3):4, 2019.

[27] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[28] 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*, 2021.

[29] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. Spatial-temporal relation networks for multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3988–3998, 2019.

[30] Yihong Xu, Aljosa Osep, Yutong Ban, Radu Horaud, Laura Leal-Taixé, and Xavier Alameda-Pineda. How to train your deep multi-object tracker. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6787–6796, 2020.

[31] Yifu Zhan, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A simple baseline for multi-object tracking. *arXiv preprint arXiv:2004.01888*, 2020.

[32] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3221, 2017.

[33] Yang Zhang, Hao Sheng, Yubin Wu, Shuai Wang, Wei Ke, and Zhang Xiong. Multiplex labeling graph for near-online tracking in crowded scenes. *IEEE Internet of Things Journal*, 7(9):7892–7902, 2020.

[34] Yang Zhang, Hao Sheng, Yubin Wu, Shuai Wang, Weifeng Lyu, Wei Ke, and Zhang Xiong. Long-term tracking with deep tracklet association. *IEEE Transactions on Image Processing*, 29:6694–6706, 2020.

[35] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.

[36] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

[37] 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)*, pages 366–382, 2018.