Extendable Multiple Nodes Recurrent Tracking Framework With RTU++

Shuai Wang, Hao Sheng, *Member, IEEE*, Da Yang, Yang Zhang, Yubin Wu, and Sizhe Wang

***Abstract*— Recently, tracking-by-detection has become a popular paradigm in Multiple-object tracking (MOT) for its concise pipeline. Many current works first associate the detections to form track proposals and then score proposalns by manual functions to select the best. However, long-term tracking information is lost in this way due to detection failure or heavy occlusion. In this paper, the Extendable Multiple Nodes Tracking framework (EMNT) is introduced to model the association. Instead of detections, EMNT creates four basic types of nodes including correct, false, dummy and termination to generally model the tracking procedure. Further, we propose a General Recurrent Tracking Unit (RTU++) to score track proposals by capturing long-term information. In addition, we present an efficient generation method of simulated tracking data to overcome the dilemma of limited available data in MOT. The experiments show that our methods achieve state-of-the-art performance on MOT17, MOT20 and HiEve benchmarks. Meanwhile, RTU++ can be flexibly plugged into other trackers such as MHT, and bring significant improvements. The additional experiments on MOTS20 and CTMC-v1 also demonstrate the generalization ability of RTU++ trained by simulated data in various scenarios.**

***Index Terms*— Multi-object tracking, data association, scoring mechanism, recurrent network, simulated data.**

I. Introduction

MULTI-OBJECT tracking (MOT) has always been a crucial challenge in computer vision applications such as urban surveillance, human-computer interaction, and autonomous driving. Recently, with the progress in object detection, tracking-by-detection (TBD) has become a popular paradigm for its clear pipeline: (i) detect objects from images, (ii) link detections to form track proposals, (iii) score track proposals and select the best. However, despite such progress, MOT remains a very challenging task due to many factors like occlusions, especially in the scoring step. Many works [1]–[3] take account of various information to score tracks, but they often fail in crowded scenes due to heavy occlusion and complex object interaction.

Many online approaches such as DeepSORT [4] and Fair [5] filter track proposals by motion distance and regard appearance similarity as the score of track proposal. This scoring mechanism is limited because the appearance feature is easy to be polluted by occlusion. Some other trackers like MHT [6] and network flow [7] design manual scoring functions to combine appearance and motion information. Whereas, the manual scoring function is unsatisfactory under long-term detection missing. In addition, although manual scoring functions combine multiple clues, they can not explore historical information because of the rigid functions. Thus, a general scoring mechanism is still urgently needed for long-term tracking.

Since neural network has shown its great potential in many fields, some researchers begin to explore the network-based scoring mechanism. Kim *et al.* [6] design multiple kinds of LSTM [8] to combine motion and appearance to score each track proposal. Their experiments show that LSTM is effective for temporal tracking data. Zhang *et al.* [9] propose a long-term tracking framework, including two LSTM-based classification networks. Sheng *et al.* [10] introduce the STCC-Net based on spatial-temporal attention to extract appearance features. They indicate that the tracker is able to obtain more accurate scores via this feature representation. Most of the works aforementioned have indicated that the training data is very important for their methods. Nevertheless, the available data for training such a network is very limited. First, the labeled MOT tracking data is expensive to collect, which limits the exploration of deep learning-based methods. Secondly, there is an unignorable difference in the dataset styles, which results in the trained network is difficult to be directly applied to another dataset.

In order to solve the problems mentioned above, a multi-scene general Extendable Mutiple Nodes Tracking (EMNT) framework is proposed in this paper, which introduces four types of nodes to model the data association during tracking. We implement an extendable global tracks optimization strategy in EMNT, balancing the speed and accuracy. Furthermore, a transplantable General Recurrent Tracking Unit (RTU++)

Manuscript received 31 October 2021; revised 2 June 2022 and 30 June 2022; accepted 10 July 2022. Date of publication 26 July 2022; date of current version 8 August 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102200, in part by the National Natural Science Foundation of China under Grant 61872025, in part by the Science and Technology Development Fund, Macau SAR under Grant 0001/2018/AFJ, in part by the Open Fund of the State Key Laboratory of Software Development Environment under Grant SKLSDE-2021ZX-03, and in part by HAWKEYE Group. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Clinton Fookes. *(Corresponding author: Hao Sheng.)*

Shuai Wang, Da Yang, Yubin Wu, and Sizhe Wang are with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, and also with the Beihang Hangzhou Innovation Institute Yuhang, Xixi Octagon City, Hangzhou, Yuhang 310023, China (e-mail: shuaiwang@buaa.edu.cn; da.yang@buaa.edu.cn; yubin.wu@buaa.edu.cn; sizhewang@buaa.edu.cn).

Hao Sheng is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, also with the Beihang Hangzhou Innovation Institute Yuhang, Xixi Octagon City, Hangzhou, Yuhang 310023, China, and also with Faculty of Applied Sciences, Macao Polytechnic University, Macau, SAR, China (e-mail: shenghao@buaa.edu.cn).

Yang Zhang is with the College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China (e-mail: yang_zh@mail.buct.edu.cn).

Digital Object Identifier 10.1109/TIP.2022.3192706

is presented to accurately score track proposals. To cope with the lack of available data in MOT, we introduce a Node-based General Tracking Data Generation algorithm (NGTDG) to generate sufficient simulated artificial tracks. Then RTU++ is trained on pure simulated tracking data with the guidance from a reward mechanism. Experimental results show that our methods achieve state-of-the-art performance on MOT17, MOT20, and HiEve benchmarks. Our methods can handle whether low or high density pedestrian tracking scenes. Even on a cell tracking dataset, our methods still achieve stable performance without any retraining. Moreover, we expand our method to Multi-object Tracking and Segmentation (MOTS) to validate its generalization. The results of EMNT on MOTS20 benchmark are comparable with other state-of-the-art methods in MOTS.

The main contribution of this paper is in three folds:

- Construct a general Extendable Multiple Nodes Tracking framework, which models the tracking by four types of basic nodes: correct, false, dummy and termination. This framework well illuminates the actual association under detection failures and can be further extended to MOTS.
- Propose RTU++ to score track proposals, which captures long-term information of nodes types to distinguish proposals. It also can be flexibly plugged into other trackers based on EMNT.
- Present a Node-based General Tracking Data Generation method to generate simulated tracking data, which can replace real data to train RTU++. The experiments show that RTU++ trained by this data is able to be directly applied to many different scenes. It provides a new solution to solve the lack of available data in MOT.

This paper is an extension of our previous conference version [11]. The current work presents abundant technology in great detail and introduces significant novelties. First, we integrate more complete association strategy based on new types of nodes and an extendable global tracks optimization strategy into EMNT. The termination node is introduced to control the exit of objects during tracking. Second, RTU++ computes the increment of the score and introduces a temporal classification loss to enforce the memory of historical nodes' information. Third, inspired by simulated annealing, we propose NGTDG to approximate the distribution of real data. Moreover, we conduct considerable new experimental results including ablation study, effectiveness analysis, state-of-the-art comparison and generalization validation to fully validate our methodology. Specially, we extend our method from MOT to MOTS and evaluate it on the MOTS20 benchmark. The comparable results prove the effectiveness of our framework.

II. Related Work

In this section, we briefly reviewed the recent tracking-by-detection paradigm and the related work in its core scoring mechanism.

A. Tracking-by-Detection

Most early MOT methods derive from filter algorithms such as Kalman filter [12], [13] and particle filtering [14]. Although these approaches achieve real-time performance, they require manual initialization of a fixed number of objects in the first frame [15]. However, recent tracking-by-detection can discover new objects and terminate disappearing ones automatically. Thus, this paradigm is more popular since filter-based objects can not deal with the case that new objects appear. In the tracking-by-detection paradigm, objects are first detected and then associated to form trajectories. Thanks to object detection techniques [16]–[18], MOT has seen great improvement in the past few years. Commonly, the tracking-by-detection paradigm is classified into online and batch methods.

Online methods [4], [19], [20] only use the past information while batch method [21]–[23] can utilize both past and future information to associate detections. Online methods are usually applied on real-time and lightweight applications for their high efficiency and low consumption [10]. SORT [19] implements an efficient association scheme based on filtering algorithm and Euclidean distance. Further, Wojke *et al.* [4] combine a neural network-based feature extractor with SORT, and obtain more outstanding results. Due to the limited past information, online methods can hardly handle the identity switches problem.

Batch methods often achieve better performance than online methods. For example, Yang *et al.* [24] propose CRF to combine various kinds of tracking clues. Afshin *et al.* [25] formulate the data association during tracking as maximum multiple cliques problem. Jiang *et al.* [26] design a linear programming algorithm to finds the best trajectories. Kim *et al.* [6] reconstruct the multi-object tracking by maximum weight independent set and introduce an online updated appearance model in their trackers. To deal with the detection errors, Zhang *et al.* [22] propose a heterogeneous association graph fusion mechanism, in which both high-level detection and low-level super-pixel are considered to build tracks. The methods mentioned above usually require a large number of computing resources, which limits their application in practice.

Recently, most online methods such as JDE [27] and Fair [5] pay more attention to jointly detecting and embedding model, while batch methods tend to find a new formulation to solve the global optimization problem. The improvement of JDE-based methods commonly benefit from their sufficiently trained detectors, which decreases the detection error significantly. However, the aforementioned approaches still directly link the detection to describe the association. In this paper, we define four types of basic nodes including correct, false, dummy and termination. And the basic nodes are associated to form track proposals instead of detections, so each proposal gets a more accurate representation.

B. Scoring Mechanism

It is common to formulate data association as a graph, in which each vertex represents a detection and each edge indicates a possible link [28]. The weight of each edge, i.e., the score, is usually calculated based on multiple types of tracking information.

Online methods prefer to construct a bipartite graph for its efficient optimization. For example, Bochinski *et al.* [29] develop the IoU tracker and regard the interaction over union of two detections as the score. Although their methods run at 100K fps, they fail to handle detection errors. Bewley *et al.* [19] introduce a filtering algorithm to refine the object location, and score track proposals by the Mahalanobis distance. Further, Wojke *et al.* [4] integrate an appearance model to SORT and regard appearance similarity as the score, combing both appearance and motion information. Fair [5] combine the detecting step and feature extraction into one stage, but it still considers appearance similarity as score.

Batch methods design more complex scoring mechanisms, which take account of multiple kinds of information. Kim *et al.* [6] design a least square-based appearance model, then implement a manual scoring function that balances appearance and motion information. Chari *et al.* [30] model tracking by network flow, and propose an energy cost function to score track proposals. To describe the frequent interactions among objects, Zhang *et al.* [23] propose the HTBT tracker by constructing and testing hypotheses. They score track hypotheses by a spatio-temporal interaction graph and then formulate tracking as a multi-dimensional assignment problem. However, such manual scoring functions are limited since they can not consider hidden long-term information.

With the development of Recurrent Neural Network (RNN), many researchers begin to explore its potential on MOT. In order to handle long-term tracking, TLMHT [9] design the Motion Evaluation Network and Appearance Evaluation Network to calculate the score of track proposals. Kim *et al.* [31] implement different LSTM-based scoring networks to study the best combination of appearance and motion features. Amir *et al.* [32] use an LSTM with a linear layer to fuse three types of features extracted from other LSTMs, and then directly outputs the final score. Ran *et al.* [33] propose a Pose-based Triple Stream Network, in which the score is computed by combining appearance, motion and pose features. Although these methods achieve outstanding results in many tasks, they are still bounded by the limited training data. First, MOT dataset is still scarce since the labeled MOT data is expensive to collect. Second, it is difficult to directly apply them to new scenarios with different styles, since there is a nonnegligible gap between the distribution of data.

In this paper, we also follow the tracking-by-detection paradigm but define four types of basic nodes. Then we build a general tracking framework to model tracking. Based on this, we proposed RTU++ to score track proposal. As for the problem of limited data, we design a simulated tracking data generation algorithm to generate artificial tracks.

III. Multiple Nodes Tracking

In this section, we introduce the Extendable Multiple Nodes Tracking framework (EMNT). Our work mainly concentrates on building the association between targets and organizing heterogeneous clues during tracking. Four types of basic nodes including correct, false, dummy and termination are introduced to describe the tracking procedure and the fusion state feature is designed to integrate multiple clues regularly. In addition, we design an extendable global track optimization strategy, which extends EMNT to many other methods.

A. Overview

It is a core for tracking to build the association between detections. The tracker is required to construct complete track proposals and select trajectories correctly. There exist amount of complex interactions among objects, which usually result in association errors. Apart from the complex interactions, it is also very significant to handle the status of objects, i.e., judge whether the objects enter into or leave the scenes.

From this point of view, we propose EMNT to describe the relationship among detections. To construct the trajectory, EMNT builds a track tree for each target and updates it with new detections in each frame. Different from previous work [4], [5], [19], EMNT links four types of basic nodes to build trajectories instead of detections as shown in Fig. 1. Specifically, correct nodes represent those correctly associated detections while false nodes denote false associations. We adopt the dummy nodes to handle the missing detections in case of occlusion or detection failure. In addition, it is obvious that those objects departed should not participate in the association again. The false associations of the objects that leave from the scene usually result in a large number of false positives. Most of the current methods recognize those departed objects by counting the number of consecutive missing associations. However, such a mechanism greatly relies on the hyperparameter, which restrains the performance in practical scenes. For the sake of handling the object's status, we introduce the termination node into EMNT to indicate that the object has left the tracking scenes. Once an object is linked with a termination node, it no longer join into the future association.

Following the tracking-by-detection paradigm, all detections are obtained from an object detector. Let F_t represents the t^{th} frame in a video sequence, d_t^j and n_t^i denote the j^{th} detection and the i^{th} node respectively in F_t.

For frame F_t, existing trees are updated with new arrival detections. As shown in Fig. 1, the track tree is extended by appending new basic nodes as its children to construct new branches. This paper designs the state feature (discussed in Sec. III-B) for each node and leverage it to classify new nodes via RTU++ (discussed in Sec. IV-A). The dummy node is appended to each branch to describe missing detection. Existing branch is linked with a termination node once the object is predicted left. The termination node represents that the object is lost and will not be extended.

In order to control the scale of the tree, a gating mechanism including both appearance and motion information is applied to filter detections. Let $T = \{n_{t-L}^{i_{t-L}}, \ldots, n_{t-1}^{i_{t-1}}, n_t^{i_t}\}$ denotes a certain track proposal (branch), where $n_t^{i_t}$ represents the ${i_t}^{th}$ node in the t^{th} frame. In this paper, the appearance feature of the node is the same with coresponding detection. The appearance distance between detection d_t^j and branch T is

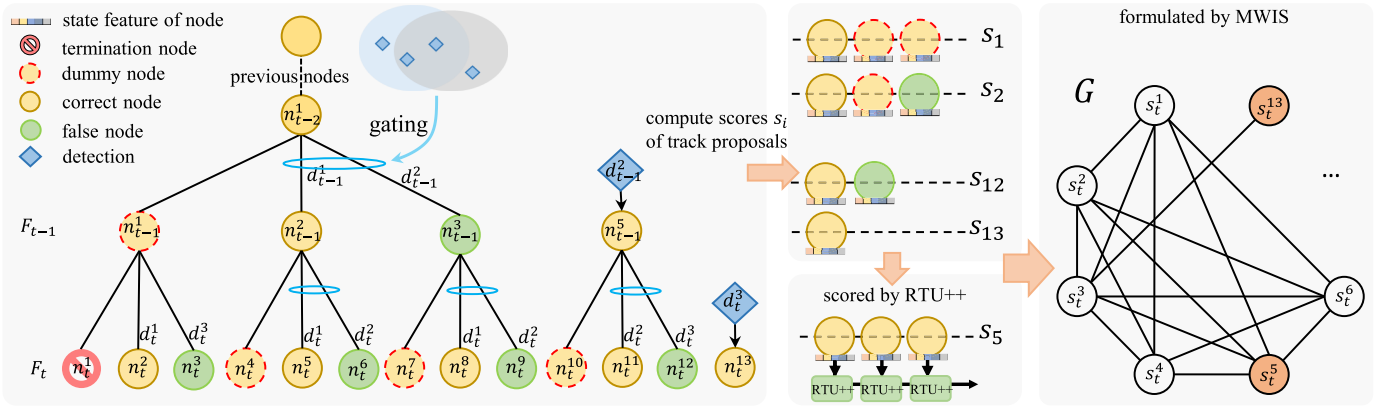


Fig. 1. The illustration of the track tree transformation. Each branch represents a potential track proposal and will be scored by RTU++. All track trees are formulated to an undirected graph, in which each branch is a vertex and an edge connects two conflicting branches. An example of the MWIS is highlighted in orange. Only the selected node is reserved in the tree after finding the optimal solution.

defined as follow:

$$dis_a = \max_{n_t^{i_t} \in T} \{ \frac{a_{n_t^{i_t}} \cdot a_{d_t^j}}{\|a_{d_t^j}\| \|a_{n_t^{i_t}}\|} \} \tag{1}$$

where $a_{n_t^{i_t}}$ and $a_{d_t^j}$ represent the appearance features of node $n_t^{i_t}$ and detection d_t^j respectively.

It is necessary to take account of motion information as well. In our framework, the location of a target is defined as the center point of the bounding box. Meanwhile, EMNT holds a Kalman filter for each target to predict the location in the next frame. Thus, the motion gating distance dis_m between detection d_t^j and branch T is defined as follow:

$$dis_m = (\tau_{d_t^j} - \tau_T^t)^{\mathsf{T}} (\tau_{d_t^j} - \tau_T^t) \tag{2}$$

where $\tau_{d_t^j}$ represent the location of detection d_t^j, τ_T^t denotes the location predicted by Kalman filter of branch T in the t^{th} frame.

Detections far from the gating region ($dis_a > \theta_a$ & $dis_m > \theta_m$, θ_m and θ_a are gating thresholds) are not used for updating existing trees but are initialized as the root node (classified as correct) of a new tree.

As shown in Fig. 1, new nodes are created when detections locate in the gating region. Then RTU++ classifies these nodes and calculates the score of each branch based on the fusion state feature.

B. Multi-Clue Fusion in Extendable State Feature

Although there are many clues underlying tracking, most methods only utilize appearance or motion information. Many works such as [31] and [34] tend to concatenate appearance and motion features. However, it is difficult to balance two types of features due to the huge gap in the dimension. The concatenation leads to low computation efficiency in training or testing. To solve these problems, this paper designs the fusion state feature.

Let X_n denotes the state feature of node n. Since appearance information makes contributions to distinguish one target from each other, it is introduced in state feature first. Some previous works [5], [19], [27] store the appearance feature and take account of manual or deep learning methods to model the trend of appearance feature. Different from them, we store the appearance similarity in the state feature instead of appearance feature. Firstly, the experimental results show that the appearance similarity between a linked node and its parent drops greatly under identity switches. This phenomenon indicates that the sequence of appearance similarity can reflect some tracking failures such as identity switches in some way. Secondly, storing appearance similarity contributes to decreasing the negative influence of the difference in various original features distribution. Thus, the high dimensional appearance feature is compressed to a concise representation, which occupies less memory in the real application. Meanwhile, state features are still efficient with new types of appearance features, since appearance features have already guaranteed the high appearance similarity of the same target.

The second dimension of the state feature takes account of motion information, which is usually more effective under long-term occlusion. Some existing methods calculate the euclidean distance between the predicted location and the real location, but the range of this distance is not definite. We estimate the motion consistency by composing the Kalman filter and multivariate normal distribution. The Kalman filter is used to predict the future location, and then the motion consistency between node $n_t^{i_t}$ and detection d_t^j is obtained as follows:

$$dis_{mc} = \max_{n_t^{i_t} \in T} \{ exp(-(P_{off} - \mathcal{Z})^{\mathsf{T}} cov^{-1} (P_{off} - \mathcal{Z})/2) \} \tag{3}$$

where $\mathcal{Z}$ is a 4-order zero matrix, cov is the covariance matrix of the Kalman filter, P_{off} is the location offset between node $n_t^{i_t}$ and detection d_t^j.

It is proved by Sheng *et al.* [10] that detection response reflects occlusion in some way. Considering that the detection confidence is provided by most detectors, we set it as the third dimension of the state feature. As discussed in Sec. III-B, EMNT describes missing detections by the dummy node. To distinguish the true nodes and the dummy nodes, a dummy indicator is initialized in the fourth dimension. It is set as 1 when the node is dummy otherwise 0. The first three dimensions of a dummy node state feature are set the same as its

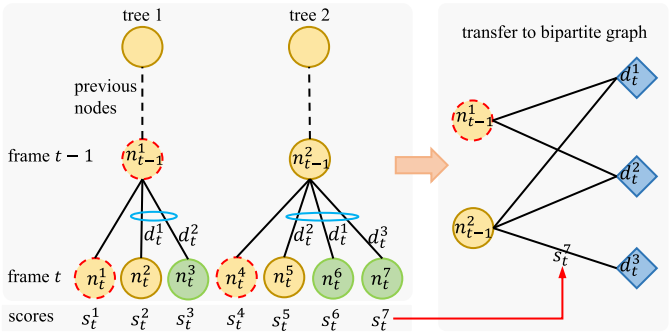


Fig. 2. The illustration of the track tree transformation by the bipartite graph. Each branch represents a track proposal and will be scored by RTU++. All track proposals are formulated to a bipartite graph, in which each edge corresponds to a track proposal.

parent node. In addition, the length of a track is also important, because it reflects the life period of a target. Usually, the longer track is more reliable than the short tracks, which is not mentioned in previous conference version. So we defined the fifth dimension of state feature as the current length of this track.

This fusion of various clues is efficient for its concise feature representation. Meanwhile, it can be extended by appending a normalized feature as the new dimension. In fact, the node as a basic structure exists in many tracking frameworks. Our proposed state feature is designed for the basic node, so it can be applied to other methods as well.

C. Extendable Global Tracks Optimization

Each branch in the tree represents a potential track proposal. It is very important for tracking to calculate the score of each track and select the highest one. The detail of obtaining scores is discussed in Sec. IV-A. Due to the failure in detection or association, different trees may represent the same trajectory. Thus it is necessary to handle the conflict between trees. The global track selection can be described as a global optimization problem. Our target is to find the optimal solution of the graph formed by all track trees. After obtaining the optimal solution, EMNT takes the pruning strategy similar to [6]. The other branches are pruned except the selected one, so only one node survives at the end of a frame. The efficiency of the tracker has critical importance in its practical application. In order to balance the speed and accuracy, the extendable global tracks optimization strategy is designed to find the optimal solution.

When the pruning depth P_d is more than 2, the global optimization problem is NP-hard. We follow the idea in [6] to formulate this problem as a maximum weighted independent set (MWIS) to find the most likely set of trajectories according to their scores obtained by RTU++. As shown in Fig. 1, new nodes are built after gating and form new branches. RTU++ takes the state features of the nodes in the branch as input and outputs the final score s. An undirected graph $G = (V, E)$ is constructed by assigning each branch T_i to a graph vertex $x_i \in V$ with the weight s_i. An edge $(i, j) \in E$ connects two vertexes x_i and x_j if two tracks have conflicts due to shared detections or having the same ancestor.

However, the MWIS-based optimization strategy is limited by the number of tracks. Since the real-time response is more important in some practice scenarios, we reconstruct the track trees as the bipartite graph when the pruning depth is 1. As shown in Fig. 2, the track trees are transformed to a bipartite graph $G = (V, E)$, in which each edge is assigned the weight calculated by RTU++ as well. In the bipartite graph, the left vertexes represent the parent nodes while the right denote the detections in the current frame. An edge $(i, j) \in E$ connects two vertexes n_{t-1}^i and d_t^j if the detection d_t^j corresponds to a branch in track trees. Then Hungrainan algorithm can be used to find an optimal solution in this bipartite graph.

On the one hand, we integrate two optimization methods in EMNT to balance the speed and accuracy. On the other hand, such integration makes that it is easy to reconstruct some other trackers by EMNT, which lets RTU++ can be flexibly plugged into them.

D. Generalize to MOTS

Although EMNT is designed for MOT at the beginning, it is also convenient to be extended to MOTS. In EMNT, all the trajectories are represented by the basic nodes. For MOT, the basic nodes actually correspond to detections as described in Sec. III-A. But as to MOTS task, EMNT takes the masks as input instead of detections. In other words, the basic nodes of EMNT represent the input mask regions in MOTS task.

Considering the mask regions are usually in irregular shapes, we leverage the enclosing rectangles to describe the boundaries of the masks. These rectangles regions are cropped from the original image, and the pixels except the mask regions are pad as zero. The appearance features of masks can be extracted by CNNs like PCB [35]. Then EMNT takes the same scheme as MOT to process the mask input. In conclusion, the core of data association and object status handling is the same for both MOT and MOTS. The reasonable adjustments lay in the types of input and some adaption to the mask such as the appearance feature. As mentioned above, EMNT implements a general tracking strategy for MOT and MOTS.

IV. General Recurrent Tracking Unit With Node-Based Data Generation

Based on EMNT, each trajectory is represented by a sequence of nodes. The various clues of the targets are stored in the temporal sequence of state features in nodes. In this section, we first introduce the structure of RTU++ in great detail to process the temporal sequence. Then we present a method of generating tracking data to overcome the defect of limited available data in MOT.

A. General Recurrent Tracking Unit

In recent multi-object tracking approaches, it has been a difficulty to incorporate long-term information to efficiently score object tracks under severe occlusion and detection missing. Batch approaches such as MHT, network flow utilize manual scoring function, while online methods tend to compare

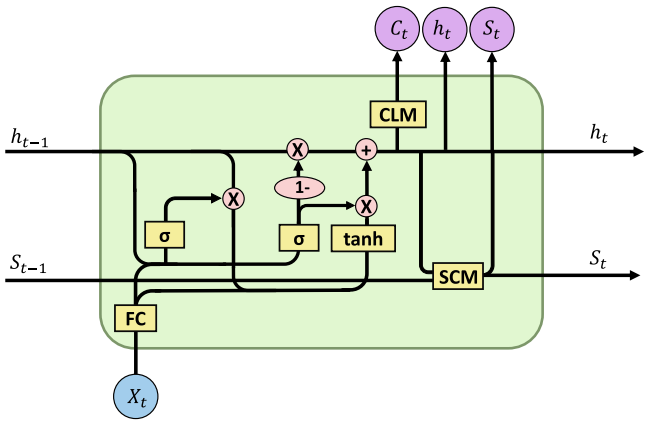


Fig. 3. The structure of RTU++. It takes three inputs (previous hidden feature h_{t-1}, previous score S_{t-1} and the state feaure X_t) at each step and outputs the hidden feature h_t, class C_t and score S_t.

the information between a pair of detections. However, these methods are limited because they only model the information in the short term. Recurrent Neural Network (RNN) is applied on Single Object Tracking (SOT) [36] firstly, and nowadays some researchers [31], [37], [38] utilize it in MOT to learn appearance or motion model. Although MOT is a more challenging problem due to the amount of occlusion and identity switches problem, RNN still shows its great potentiality, especially in processing sequence data.

The common usage of RNN includes two types, regression and classification. Our previous conference version [11] RTU follows the regression paradigm to score potential tracks. But in further experiments, we find that this paradigm results in the unstable regression for those very long tracks. Because in the previous version, we leverage RTU to directly compute the total score of a potential track. That means the range of the network output is $[-\infty, +\infty]$ as a regression paradigm. For some extremely long tracks, the corresponding scores grow greatly large, which goes against the training of the network. Meanwhile, RTU only utilizes the node types to generate the training labels, whereas does not learn to classify nodes explicitly. For these reasons, we combine both the regression and classification into one network. By learning historical information such as the node types, RTU++ computes the increment of the track score instead of the total score. The output range is reduced so as to train RTU++ more stably.

In EMNT, each track proposal is represented by a node sequence in unfixed size. So the sequential state features in the track proposal compose time data, which can be input to RTU++.

Fig. 3 shows the graphical depiction of RTU++. For time steps t, RTU++ takes three inputs: the old hidden state h_{t-1}, the old score S_{t-1} and the state feature X_t of the current node. In the first time t_0, the hidden state and score is initialized to zero and 0.01 respectively.

First, RTU++ projects the state feature to the hidden space $\mathbb{R}^{1\times D_h}$. Then the $reset$ gate r is computed by:

$$\begin{aligned} X'_t &= W_x X_t \\ r_t &= \sigma(W_r[h_{t-1}, X'_t] + b_r) \end{aligned} \tag{4}$$

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

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

$$z_t = \sigma(W_z[h_{t-1}, X'_t] + b_z) \tag{5}$$

The update of the hidden state is computed by:

$$\begin{aligned} \hat{h}_t &= tanh(W_{\hat{h}}[r_t * h_{t-1}, X'_t]) \\ h_t &= (1 - z_t)h_{t-1} + z_t\hat{h}_t \end{aligned} \tag{6}$$

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

Different from RTU, we take account of the node types in RTU++. Concretely, RTU measures the changing trend of appearance and motion and then estimates the scores of track proposals. It assigns lower scores to those track proposals with worse temporal consistency. Although it can learn to distinguish what is a good track or bad, sometimes it can not select the best one in good tracks. Obviously, the best track proposal should have the maximum correct nodes and the minimum false nodes. In order to mimic this principle, the Classification Module (CLM) is introduced into RTU++, which is calculated as follow:

$$C_t = softmax(W_c^2(relu(W_c^1 h_t + b_c^1)) + b_c^2) \tag{7}$$

where W_c^1 and W_c^2 are learned weight matrices, b_c^1 and b_c^2 are biases, $softmax$ is the activation function.

In each time step, RTU++ updates the hidden state h_{t-1} to h_t, then h_t is transferred to classificiation space $\mathbb{R}^{1*D_h}$. Finally, the hidden state is projected to the score increment $\hat{S}_t$, after two linear layers. Then the score S_t is computed by adding the score increment $\hat{S}_t$ and the score S_{t-1} at previous time. It is formulated as:

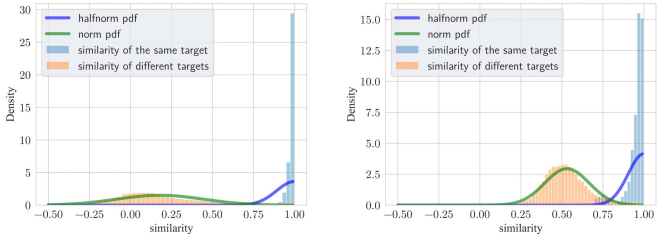
$$S_t = \hat{S}_t + S_{t-1} = W_s^2(relu(W_s^1 h_t + b_s^1)) + b_s^2 + S_{t-1} \tag{8}$$

where W_s^1 and W_s^2 are learned weight matrices, b_s^1 and b_s^2 are biases.

In RTU++, the hidden state encodes the long-term clues of one track proposal, which helps to exploit historical information to score robustly. In order to differentiate proposals more accurately, it remembers the information of node types in hidden states simultaneously. All track proposals are scored by RTU++ during tracking, then the scores are assigned to the weights of vertexes or edges as discussed in Sec. III-C.

B. Node-Based General Tracking Data Generation

It is a common challenge for deep learning-based methods to generate the training data. Previous works such as [31] and [9] generate the training sequence from the public MOT datasets. Each training sequence includes detection from the same target and one detection from a different object at the end. However, these approaches are limited by the original dataset like MOT17 and may poorly represent actual track proposals in actual scenes. First, the training data generated from ground truth is still insufficient due to the scale of the original dataset. Secondly, the generated training data is easy to be influenced by the style of the original dataset, i.e., they



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

Fig. 4. The similarity of two kinds of features. We calculate the similarity distribution of the same targets and different targets respectively. The fitted probability density function (pdf) is also showed in each sub-figure.

can not generalize satisfactorily in new datasets. We design a node-based tracking data generation methodology in order to solve these problems.

The tracking procedure is illuminated by the association between different types of nodes in EMNT. Actually, all the trajectories are represented by the node sequence. In turn, it means that we can combine different types of nodes to generate artificial trajectories. Meanwhile, multiple clues in tracking are encoded in the state feature for each node. Thus we can obtain various nodes by adjusting the state feature.

EMNT introduces four types of nodes, correct, false, dummy and termination nodes. Generally, the correct node keeps a high correlation in terms of appearance and motion, which reflects in the high appearance similarity and motion consistency. On the contrary, the false node usually has a worse correlation. With this assumption, we pick some ground truth track annotation to analyze the distribution of types of nodes.

The trajectories from ground truth compose of sequential detections. In order to robustly analyze the distribution of appearance similarity, we extract two kinds of Re-ID appearance features for each detection including PCB [35] and Fair [5]. We count the appearance similarity of detection pairs picked from the same ground truth trajectory to emulate the correct nodes. As for the false nodes, we randomly pick the detection pairs from the different trajectories. As shown in Fig. 4, the statistic results indicate that the appearance similarity of the correct nodes is approximately a half-normal distribution $\mathcal{N}(\mu_{ca1}, \sigma^2_{ca1})$. Although the distributions of similarity within different features present a little difference, they still show some important commons. For example, even the domain has been stretched by the standard deviation factor σ, the mean values of two distributions are close to 1. In fact, the standard deviation reflects the distinguishing ability of different features to some extends. And we can inference that about 99.7% data are within three standard deviations according to the 3-sigma rule. Thus the lower bound of the similarity of the same target in various features can be defined as follows:

$$\beta_{app} = \min\{\mu_{ca1} - 3\sigma_{ca1}, \mu_{ca2} - 3\sigma_{ca2}, \ldots\} \tag{9}$$

where σ_{cai} and μ_{cai} represents the standard deviation and mean value of the appearance similarity distribution in i^{th} feature representation respectively.

Similarly, we analyze the distribution of motion consistency in the correct node or false node. The static results show that the motion consistency is nearly a half-normal distribution but

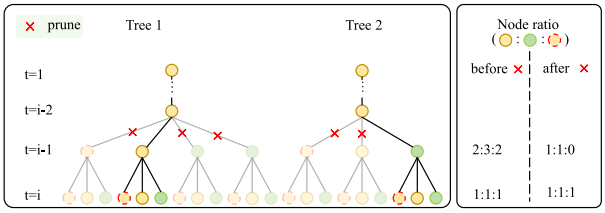


Fig. 5. The illustration of a pruning process and the change of node ratio during this process. The pruned branches are drawn as transparent. The distribution of nodes is changing along with the pruning process.

it has a bigger standard deviation than appearance similarity. So the the lower bound of motion consistency of the same target can be defined as follows:

$$\beta_{mot} = \mu_{cm} - 3\sigma_{cm} \tag{10}$$

where σ_{cm} and μ_{cm} represents the standard deviation and mean value of the motion consistency distribution respectively.

As for the confidence in each node, it lies more randomly within the range of [0, 1]. Here we donote the distribution of confidence as $D(0, 1)$. In addition, considering that each correct or false node corresponds to a true detection in fact, the dummy indicators of them are assigned to 0 directly. In terms of the dummy node, we assume that its first to third dimensions of state feature fit the same distributions as correct node, but its dummy indicator is assigned to 1. Based on these observations, the correct node and false node should satisfy Eq. 11 and Eq. 12 during the random generation.

$$\beta_{app} \le X^1_c \le 1 \ and \ \beta_{mot} \le X^2_c \le 1 \tag{11}$$

$$0 \le X^1_f < \beta_{app} \ or \ 0 \le X^2_f < \beta_{mot} \tag{12}$$

$$\begin{aligned} s.t. \quad X^1_c &\sim \mathcal{N}(\frac{\mu_{ca1} + \mu_{ca2}}{2}, \frac{\sigma^2_{ca1} + \sigma^2_{ca2}}{4}) \\ X^2_c &\sim \mathcal{N}(\mu_{cm}, \sigma^2_{cm}) \\ X^1_f &\sim \mathcal{N}(\mu_{f1}, \sigma^2_{f1}) \\ X^2_f &\sim \mathcal{N}(\mu_{f2}, \sigma^2_{f2}) \\ X^3_{c|f} &\sim \mathcal{D}(0, 1) \end{aligned} \tag{13}$$

where X^i_c and X^i_f represents the i^{th} dimension of the state feature in correct node and false node respectively.

It is simple to directly build artificial track proposals by generating four types of nodes in complete random. In this way, the generated track proposals $\mathcal{T}$ nearly fit a conditional probability distribution:

$$\begin{aligned} p(\mathcal{T}) &= p(n_l \cup n_{l-1} \cup n_{l-2} \cup \ldots \cup n_1) \\ &= \prod_{i=1}^{l} p(n_i | n_{i-1}, \ldots, n_1) \end{aligned} \tag{14}$$

where l represents the length of the generated track proposals $\mathcal{T}$, n_i represents the i^{th} node in $\mathcal{T}$.

Although this simple method can generate numerous potential track proposals, it is not efficient due to the exponential growth of proposal space. We find that lots of track proposals exist only in the theory but are pruned in fact. As shown in Fig. 5, we draw two possible track trees during tracking and analyze the change of node ratio when pruning. At time $i - 1$, the node ratio is 2:3:2 before pruning.

Algorithm 1 Node-Based General Tracking Data Generation Algorithm

Input: the sampled length L and the initial ratios $\gamma_c, \gamma_f, \gamma_d$
Output: the generated track proposal $\mathcal{T}$

1: **for** i = 1 to L **do**
2: Simple a probability p
3: Update the ratios γ_c, γ_f using Eq. 15
4: **if** $0 \leq p < \gamma_c$ **then**
5: Simple a correct node n_i using Eq. 11
6: **else if** $\gamma_c \leq p < \gamma_c + \gamma_f$ **then**
7: Simple a false node n_i using Eq. 12
8: **else**
9: Simple a dummy node n_i using Eq. 11
10: **end if**
11: $\mathcal{T} \leftarrow \mathcal{T} \cup n_i$
12: **end for**
13: $\mathcal{T} \leftarrow \mathcal{T} \cup n_T$
14: $\mathcal{T} \leftarrow reverse(\mathcal{T})$

After pruning, abundant dummy and false nodes are pruned (drawn transparent), which results in the change in node ratio. Therefore, the node distribution is under dynamic changing during generation. Considering that the pruning is correct in most cases, most of the previous nodes in the final track tree are correct, while the latest nodes (like nodes at $t = i$ in Fig. 5) present another distribution. Therefore, we design a Node-based General Tracking Data Generation algorithm to mimic this phenomenon.

Let γ_c and γ_f denotes the ratio of correct nodes and false nodes in each time step respectively. In Eq. 15, we introduce a simulated annealing mechanism to decrease the ratio of false nodes in the case of the aforementioned pruning.

$$\begin{aligned} \gamma_f &= e^{\frac{i}{\tau}} \\ \gamma_c &= 1 - \gamma_d - \gamma_f \end{aligned} \tag{15}$$

Furthermore, we can generate different nodes according to their state feature distribution. The complete algorithm is shown in Alg. 1.

Alg. 1 takes the length L of track proposal as input and output an artificial proposal. It samples a probability p to decide to generate which types of node. In each loop, the ratio of correct node is updated according to Eq. 15 as shown in line 3. Then the algorithm chooses one node based on its distribution. In the last step, a termination node n_T is appended to the track proposal $\mathcal{T}$. With this generation algorithm, we can generate sufficient data to train the proposed RTU++.

C. Training RTU++ With Generated Data

Nowadays, substantial dataset is a key for many deep learning-based methods. Such methods [9], [31] take appearance features as input and are trained on the public datasets. However, they need to be retrained when facing new datasets or combing with new feature extractors. Current data-driven methods perform worse on testing data when the training data has a great difference from testing data. Moreover, since labeled multi-object tracking datasets are expensive to collect, it is challenging to train a multi-scene robust model. But our proposed state feature can decrease the influence from the dataset style. Meanwhile, the aforementioned tracking data generation algorithm is able to generate substantial training data. Thus, we train RTU++ on pure generated data.

Fig. 6. Three potential track proposals with the same number of correct/false/dummy nodes.

In the last subsection, the data is generated by combining various artificial nodes. In order to guide RTU++ to learn a scoring model, all generated training data are assigned a label. As shown in Fig. 6, all track proposals have the same number of correct, false and dummy nodes. Obviously, the label should assure that all track proposals can be sorted in strict order since only one track proposal represents the best. Thus we introduce a reward mechanism to ensure the order of data. Different types of nodes are assigned with different rewards. The correct nodes deserve the biggest reward r_c, the dummy node is paid a lower reward r_d, but the false node obtains a 'penalty'' r_f instead. Finally, considering that the length is another key property of a track, we give each track a length reward r_l. Different from our previous version, RTU++ is designed to calculate the score increment, so the training label is a score increment as well. Hence the training label, that is, the ground truth score of each simulated track proposal is computed by:

$$\begin{aligned} S = r_c * \mathbb{1}_c[X_t] * \sum_{i=0}^{2} X_t^{(i)} \\ + r_d * \mathbb{1}_d[X_t] * \sum_{i=0}^{2} X_t^{(i)} \\ - r_f * \mathbb{1}_f[X_t] * \sum_{i=0}^{2} X_t^{(i)} \\ + r_l * X_t^{(4)} \end{aligned} \tag{16}$$

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 great benefit to select the best track. Furthermore, in order to guide the training of RTU++, we firstly define the score loss function $\mathcal{L}_s$ as follow:

$$\mathcal{L}_s = \|s_T - S\|_2 \tag{17}$$

where $\|.\|_2$ is the L_2 norm, S is the objective score given by Eq. 16, s_T denotes the score increment computed by RTU++.

To learn the hisrotical node types information, the temporal classification loss $\mathcal{L}_c$ is introduced:

$$\mathcal{L}_c = \sum_{t=1}^{T}(-log\frac{e^{c_{t,C}}}{\sum_{j=1}^{4} e^{c_{t,j}}}) \tag{18}$$

where C represents the target class, c_t is the output of RTU++ at time t. Finally, the total loss of our approach can be expressed as:

$$\mathcal{L}_G = \lambda\mathcal{L}_s + \mathcal{L}_c \tag{19}$$

where λ is the weight to balance two objectives.

Based on this, we can utilize aforementioned generation algorithm to generate sufficient data and train RTU++ in an end-to-end way.

V. Experiments

In this section, the network details and metrics are described first. Then, we design a group of ablation study experiments to demonstrate the effectiveness of each component. In order to validate the effectiveness of our method, a series of comparison experiments are conducted to prove the generalization of RTU++. Finally, we report the results of the proposed approach compared to other state-of-the-art tracking methods on MOT17 [39], MOT20 [40] and HiEve [41] benchmarks. In addition, we extend our methods to cell tracking and MOTS scenarios to validate its generalization.

A. Implementation Details

1) Datasets and Evaluation Metrics: Five datasets are utilized for evaluation in this paper including MOT17, MOT20, HiEve, MOTS20 and CTMC-v1 [42]. These datasets are of a great difference in style, motion pattern, crowd density and etc. MOT17 contains 7 training sequences and 7 testing sequences. These sequences are of different resolutions and cameras types. MOT20 contains 8 more challenging video sequences in extremely crowded scenes. All sequences are taken by fixed cameras. HiEve includes one of the largest number of trajectories in complex events. CTMC-v1 dataset is different from the previous two since it pays attention to cell tracking. This dataset consists of 86 live-cell imaging videos that represent 14 different cell lines of various shapes and sizes. And MOTS20 extends MOT benchmark to a new benchmark defined on a pixel-level with precise segmentation masks.

For quantitive evaluation, we adopt the CLEAR MOT metrics [43] and MOTS metric [44]. The most important indicators are MOTA↑, IDF1↑ [45] and HOTA↑ [46]. MOTA denotes multi-object tracking accuracy that combines FP↓ (false positives), FN↓ (missed targets) and IDs↓ (identity switches), while IDF1 is the ratio of correctly identified detections over the average number of ground-truth and computed detections. HOTA is the higher order tracking accuracy, which better aligns with human visual evaluation of tracking performance. sMOTSA↑ denotes mask-based soft multi-object tracking accuracy, which accumulates the mask overlaps of true positives.

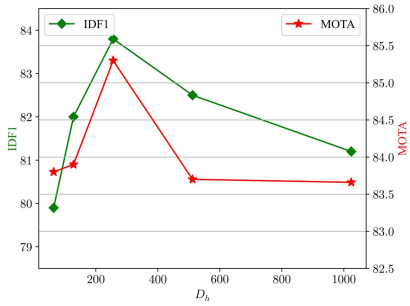


Fig. 7. MOTA and IDF1 curves when adjusting the dimension of hidden state. Both IDF1 and MOTA become higher first and then drop as the dimension of the hidden state increases.

2) Network Architecture and Training: All the experiments are conducted on the following specifications: Intel i9-10900K CPU, 128 GB RAM, and RTX 3090 GPU using PyTorch 1.8. We adopt the Adam optimizer to train RTU++ for 200 epochs. The learning rate is set as 1e-5. According our previous experiments, We take $\beta_{app} = 0.8, \beta_{mot} = 0.6, \mu_{ca1} = 0.9, \sigma_{ca1} = 0.03, \mu_{cm} = 0.7, \sigma_{cm} = 0.05, \mu_{f1} = 0.25, \sigma_{f1} = 0.1, \mu_{f2} = 0.1, \sigma_{f2} = 0.1, \lambda = 1$. In RTU++, the hyperparameters r_c, r_d and r_f has a great influence on its performance. Thus we first take the similar searching strategy as [11] and adopt $r_c = 5$, $r_d = 4.5$, $r_f = 10$ and $r_l = 1$. Then, considering that the temporal information is stored in the hidden state, we confirm the dimension of hidden state by the experiments shown in Fig. 7. The experimental results show that both MOTA and IDF1 climb first as the hidden state dimension increase, because the bigger dimension brings a more powerful representation ability. However, an over-high dimension actually contains redundant information, so the performance drops when the dimension is bigger than 512. Finally, we take a 256 dimension hidden state in all other experiments.

B. Effectiveness Analysis

To demonstrate the effectiveness of our method, we first compare our method with RTU and TT17 on some typical tracking scenarios within different lengths. These video sequences are in different density and occlusion degrees, reflecting different challenges in tracking. In this group of experiments, all the trackers take the same input detections and appearance features provided by [5]. Experimental results are listed in Tab. I.

We pick two pairs of long-short video sequences from MOT17 and MOT20 datasets. The results show that in longer or more crowded sequences, TT17 [10] conducts worse performance, especially on the IDF1 indicator. Different from RTU++, it regards the scoring step as a binary classification problem, which limits the gap between false and correct tracks. In addition, we find that TT17 can not generalize well on new datasets such as MOT20 since it is trained on ground truth. But in MOT20, there is an obvious difference between the ground truth and the testing dataset. Thus, it suffers a more serious identity switches problem.

Different from TT17, both RTU and RTU++ are trained on simulated tracking data instead of ground truth. So they achieve more stable performance on MOT20. The results show

TABLE I

THE EFFECT ANALYSIS EXPERIMENTS ON MOT17 AND MOT20 DATASETS. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD. ↑ / ↓ INDICATES THE HIGHER/LOWER VALUES DENOTE THE BETTER PERFORMANCE

Sequence	Length	Tracks	Types	Methods	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT↑	ML↓	Frag↓
MOT17-04	1050	83	Fixed	TT17	90.4	**95.0**	**1072**	**1234**	41	**77**	**2**	71
				RTU	90.8	94.1	1252	1496	56	75	3	69
				RTU++	**93.4**	95.0	1097	1252	**40**	75	**2**	**50**
MOT17-10	654	133	Moving	TT17	67.8	72.9	727	2674	80	34	**1**	186
				RTU	67.0	74.5	**638**	2983	104	27	3	144
				RTU++	**72.0**	**79.4**	672	**1899**	**69**	**38**	3	**118**
MOT20-01	429	90	Fixed	TT17	85.9	88.5	826	**1405**	54	66	**0**	55
				RTU	87.2	88.6	**431**	1788	45	67	**0**	61
				RTU++	**87.6**	**89.7**	460	1571	**43**	**68**	**0**	**39**
MOT20-05	3315	1211	Fixed	TT17	83.9	89.8	**21964**	43388	989	952	55	1094
				RTU	87.0	89.9	27840	36283	879	**994**	**54**	879
				RTU++	**87.4**	**90.4**	27039	**35106**	**710**	**994**	**54**	**696**

TABLE II

THE EFFECT ANALYSIS EXPERIMENTS ON MOT17 TRAINING DATASET. ALL METHODS TAKE THE SAME INITIAL INPUTS PROVIDED BY [5]

Method	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT↑	ML↓	Frag↓	MOTP↑
TT17 [6]	76.2	83.1	4566	**13667**	659	**364**	74	886	81.6
TT17 + RTU++	**80.9**	**83.7**	**3931**	13928	**401**	361	**69**	**767**	**81.7**
DeepSORT [4]	72.5	83.6	2841	**14811**	691	336	**323**	2011	**81.1**
DeepSORT + RTU++	**79.1**	**83.7**	**2814**	14826	**681**	**340**	333	**2010**	**81.1**

that RTU++ performs better than RTU. We attribute this improvement to two aspects. First, RTU++ no longer learns the total score of a track but learns the increment of the score. It means that the output range of the network shrinks so that the projection of scoring is easier to be learned. Second, RTU only considers historical information and does not remember the node types. But RTU++ leverages the temporal classification loss to remember the node types in memory explicitly. Thus in those tracking scenarios with very complex associations, RTU++ achieves significant improvement compared to RTU. Fig. 8 shows the tracking results of RTU++ in different scenarios. Even MOT20-05 is an extremely crowded scenario, RTU++ still acquires the lowest IDs at 710. In other low or high-density tracking scenarios, RTU++ also achieves stable as well as outstanding results, which proves the effectiveness of our methodology.

As discussed in Sec. III-C, EMNT can be extended to many famous trackers, then RTU++ is able to be flexibly plugged into them. Here we reconstruct two classic trackers including TT17 and DeepSORT. TT17 is a tracklet-based batch method while DeepSORT is a real-time online algorithm. Their scoring modules are replaced with RTU++ but other components are reserved. We provide the same input detections and appearance feature to them and the corresponding results are shown in Tab. II.

It can be seen that both TT17 and DeepSORT benefit from RTU++, especially on the IDF1 indicator. Compared to the original DeepSORT, RTU++ utilizes five types of features to compute the score, while DeepSORT regards appearance similarities as scores. So RTU++ can still distinguish good track proposals, but DeepSORT is usually influenced by the heavy occlusion. The results show that RTU++ can improve IDF1 to 79.1 for DeepSORT. This proves that our RTU++ trained by simulated data can be applied on some bipartite-graph-based methods.

TT17 trains two LSTM-based networks on MOT17 ground truth to classify track proposals, combing appearance and motion information. But the available groud truth is limited due to the dataset scale. And the networks trained by ground truth can not generalize well on the test dataset. Thus when we add RTU++ into TT17, IDs drops significantly by 258. Their LSTM-based networks regard scoring as binary classification, which decreases the gap between scores of tracks. After adding RTU++, we can see that IDF1 and MOTA reach 80.9 and 83.7 respectively, which shows that RTU++ is still effective for MHT-based methods.

C. Ablation Study

We conduct the ablation study experiment on the MOT17 training dataset. In this experiment, we train RTU++ on the pure simulated tracking data, the MOT17 training dataset is only used for estimation since it has the labeled ground truth. All the detection and appearace featuers in ablition study are provided by [5].

Our baseline (denoted by $\mathcal{B}$) is a naive MHT [6] framework, which only contains data association, pruning and a manual scoring function. As shown in Eq. 20, the manual scoring function only considers two kinds of information. The distinguish between multiple nodes is not implemented in the baseline.

$$S = W_a S_a + W_m S_m \tag{20}$$

where W_s and W_m are the weights.

First, we add the dummy node $\mathcal{D}$ to the baseline (denoted as $\mathcal{B}+\mathcal{D}$) and still utilize the manual scoring function. An object is regarded as lost when it has 30 dummy nodes. As shown in

Fig. 8. The tracking result on different tracking scenarios. The lines behind the bounding boxes represent the historical trajectories of objects. Our tracker shows outstanding results on both low-density and high-density. Even on the MOTS scene, it still tracks targets stably. No matter on the outdoor street or in the market, the masks are associated with targets accurately.

Tab. III, after adding dummy nodes, the IDs and FN indicators decrease greatly by 305 and 6477 respectively, which proves that dummy nodes can handle the detection missing. They link the fragmented tracks to complete trajectories, so IDF1 improves by 16.4 as well. But FP becomes worse, since part of dummy nodes can not estimate the status of occluded target correctly. These false dummy nodes result in the increase of FP. Simutenously, the dummy nodes bring extra computation, leading to the reduce in FPS (from 19.5 to 19.1).

Secondly, RTU++ is added to replace the original manual scoring function. But in this experiment, RTU++ is only trained with the score loss $\mathcal{L}_S$ and the termination node is froze (denoted by $\mathcal{B}+\mathcal{D}+\mathcal{R}^*+\mathcal{L}_S$). The results show that FN is lower than $\mathcal{B}+\mathcal{D}$ by 2519. The IDs indicator drops as well, because RTU++ is capable to give correct tracks higher scores compared to the manual scoring function. The correct and false track proposals are well distinguished by RTU++, since it stretches the score gap between correct and false proposals. Thus IDF1 and MOTA improves to 81.7 and 83.9 respectively. Because all the potential track is organized as batch data, RTU++ is able to score them in one inference. The runtime speed decreases slightly from 19.1 to 18.4.

Then we add RTU++ trained with total loss $\mathcal{L}_G$ including score and classification loss. This experiment is denoted by $\mathcal{B}+\mathcal{D}+\mathcal{R}^*+\mathcal{L}_G$. We can see that IDs decrease again by 48. Meanwhile, both FP and FN drop, especially FN is 659 lower than before. This is because the classification loss contributes to remembering historical information of node types in the hidden state of RTU++.

TABLE III

ABLATION STUDY OF VARIOUS SETTINGS OF NODE TYPES AND SCORE MECHNISM

Setting	IDF1↑	MOTA↑	IDs↓	FP↓	FN↓	FPS
$\mathcal{B}$	63.1	78.4	809	**1330**	22034	19.5
$\mathcal{B}$+$\mathcal{D}$	79.5	82.4	504	3814	15557	19.1
$\mathcal{B}$+$\mathcal{D}$+$\mathcal{R}^*$+$\mathcal{L}_S$	81.7	83.9	475	4540	13038	18.4
$\mathcal{B}$+$\mathcal{D}$+$\mathcal{R}^*$+$\mathcal{L}_G$	83.4	84.7	427	4226	**12379**	18.4
$\mathcal{B}$+$\mathcal{D}$+$\mathcal{R}^*$+$\mathcal{L}_G$+$\mathcal{E}$	83.7	**85.3**	**403**	3453	12639	18.5
$\mathcal{B}$+$\mathcal{D}$+$\mathcal{R}^*$+$\mathcal{L}_G$+$\mathcal{E}$+$\mathcal{O}$	**83.8**	**85.3**	410	3458	12636	**22.9**

Furthermore, we integrate the termination node $\mathcal{E}$ to mark the terminated objects ($\mathcal{B}+\mathcal{D}+\mathcal{R}^*+\mathcal{L}_G+\mathcal{E}$). Since the object location is constantly updated for each objects, EMNT can predict whether the object leaves the tracking scenario. Compared with directly counting the number of dummy nodes, the termination nodes make the termination of tracking more flexibly. Those missing objects but acutually existing in the scenario is still represented by dummy nodes. But the missing objects that are predicted leaving will be terminated. Most reduncant dummy nodes are reduced. Thus, we can see that FP decreases by 773 and MOTA increase to 85.3. Since the inference of RTU++ is not changed in this experiment, the speed is almost the same as before.

TABLE IV
TRACKING PERFORMANCE ON MOT17 BENCHMARK

Method	IDF1↑	MOTA↑	HOTA↑	FP↓	FN↓	IDs↓	MT↑	ML↓	Frag↓	FPS↑
TubeTK [47]	58.6	63.0	48.0	27060	177483	5137	735	468	5727	3.0
GSDT [48]	68.7	66.2	55.5	43368	144261	3318	960	432	8046	4.9
CenterTrack [49]	64.7	67.8	52.2	**18498**	160332	3039	816	579	6102	3.8
TraDes [50]	63.9	69.1	52.7	20892	150060	3555	858	507	4833	**66.9**
MLT [3]	70.1	69.3	-	35844	135888	1347	906	918	2028	19.3
CSTrack [51]	72.9	74.9	59.3	23847	114303	3567	978	411	7668	15.8
Fair [5]	72.3	73.7	59.3	27507	117477	3303	1017	408	8073	25.9
RTU [11]	75.0	74.9	62.0	32007	107616	1812	1170	444	**1824**	17
ByteTrack [52]	77.3	**80.3**	63.1	25491	**83721**	2196	1254	342	2277	29.6
RTU++ (ours)	**79.1**	79.5	**63.9**	29508	84618	**1302**	**1302**	**318**	2046	24.7

TABLE V
TRACKING PERFORMANCE ON MOT20 BENCHMARK

Method	IDF1↑	MOTA↑	HOTA↑	FP↓	FN↓	IDs↓	MT↑	ML↓	Frag↓	FPS↑
Fair [5]	67.3	61.8	54.6	103440	88901	5243	855	**94**	7874	13.2
GSDT [48]	67.5	67.1	53.6	31507	135395	3230	660	164	9878	1.5
RTU [11]	65.0	65.6	53.4	54928	123067	3238	680	143	5921	11.8
CSTrack [51]	68.6	66.6	54.0	**25404**	144358	3196	626	192	7632	4.5
SGT	70.5	72.8	56.9	25165	112897	2649	798	157	5486	17.2
MAA	71.2	73.9	57.3	24942	108744	1331	741	153	1450	14.7
ReMOT [53]	73.1	77.4	61.2	28351	**86659**	2121	846	123	2121	0.4
ByteTrack [52]	75.2	**77.8**	61.3	26249	87594	1223	**859**	118	1460	**17.5**
RTU++ (ours)	**76.8**	76.5	**62.8**	19247	101290	**971**	811	132	**1190**	15.7

Finally, we validate the efficiency of the proposed extendable global tracks optimization (denoted by $\mathcal{O}$). As shown in the last row, the extendable optimization strategy brings remarkable improvement to FPS. Since many complex track trees are transformed to bipartite graph, the computation is simplified when the tracker finds the optimal solution. Meanwhile, there exists some inherent difference in the solutions from two types of optimization strategies, which results in the change in some indicators. But such changes are acceptable with the improvement on speed.

These results demonstrate that RTU++ is stronger than RTU and the total loss is more effective to train RTU++. The extendable optimization strategy is also efficient to reduce the computation.

D. State-of-the-Art Comparison

We report the quantitative results obtained by our method on MOT17 and MOT20 benchmarks in Tab. IV and Tab. V respectively. The results show that our method obtains state-of-the-art results on both benchmarks. In these experiments, we use the detection extracted from [57].

As shown in Tab. IV, our RTU++ achieves outstanding performance on most indicators. It achieves the highest IDF1 and HOTA by 79.1 and 63.9 respectively. Meanwhile, it holds a real time performance. In our methods, the track is represented by four types of nodes, which is suitable to describe the association under occlusion. Based on this, many fragmented tracks are linked to a complete track. So we achieve the lowest Frag. Our RTU++ trained by the reward mechanism gives differentiated scores to potential tracks to select the correct tracks. Thus compared to other methods, RTU++ maintains the lowest IDs by 1302. The results show that RTU++ has a lower FP than RTU. Since RTU also implements dummy nodes, but those false dummy nodes are regarded as FPs. RTU++ encodes the nodes types information in hidden state, then many false associations are pruned. In this way, the dummy nodes are estimated more accurately, which decreases the FP indicator. Even on the latest evaluation indicator HOTA, it still achieves the best performance at 63.9, which is 0.8 higher than the second tracker.

As for the latest benchmark MOT20, the results listed in Tab.V show that our methods outperform other trackers obviously on IDF1, HOTA and IDs. As shown in Fig. 8, MOT20 is of great difference from MOT17 for its very crowded scenes. Many trackers [3], [5] have to retrain or finetune their networks on MOT20 or external datasets. Our RTU++ is only trained on pure simulated data and not finetuned anymore on MOT20. The results show that RTU++ achieves the highest IDF1 at 76.8. Even ByteTrack acquires higher MOTA, it performs worse on IDs, because it can not handle the heavy occlusion. The association in crowded scenes is more complex than in others. For a long-term association in crowded scenes, RTU may select a false track due to the great complex interaction. As discussed in Sec. IV-A, RTU can not give correct scores for those extremely long tracks. But RTU++ not only leverages the node types information but also adjusts to learning the score increment. That is very suitable for long tracks in crowded. Thus it achieves the lowest IDs at 971, and the highest HOTA at 62.8. The entire results demonstrate that our methods are general and robust in extremely crowded scenarios.

Besides, we evaluate our methods on the HiEve benchmark, which contains large-scale video data in complex events. As shown in Tab. VI, our method achieves the highest IDF1

TABLE VI
TRACKING PERFORMANCE ON HIEVE BENCHMARK UNDER THE PRIVATE PROTOCOL

Method	IDF1↑	MOTA↑	FP↓	FN↓	IDs↓	MT↑	ML↓	Frag↓	FPS↑
SiamTRCNN [54]	43.2	46.5	4667	30489	2133	26.3	30.8	1981	8
Fair [5]	46.7	35.0	6523	37750	995	16.3	44.2	2312	25
CSTrack [51]	51.5	48.7	2366	31933	1475	20.5	33.6	2341	30
CrowdTracker [55]	57.2	55.5	3439	25783	1772	32.9	22.9	2351	4
Adapt [56]	60.0	60.2	4530	**21292**	1889	**41.2**	**20.3**	1808	5
ByteTrack [52]	63.1	**61.7**	2622	22852	1031	38.3	21.6	1266	**30.5**
RTU++ (ours)	**67.0**	60.1	**1591**	25586	**598**	32.0	28.2	**693**	23

TABLE VII
TRACKING PERFORMANCE ON MOTS BENCHMARK

Method	IDF1↑	sMOTSA↑	FP↓	FN↓	IDs↓	MT↑	ML↓	FPS↑
GMPHD [58]	66.4	69.4	935	3985	484	249	11	2.6
MAF_HDA [59]	67.0	69.9	863	**3958**	401	245	12	4.6
UniTrack [60]	67.2	68.9	801	4185	622	**253**	**9**	7.6
COSTA [61]	70.3	69.5	806	8721	421	**253**	**9**	2.1
ReMOTS [62]	75.8	**70.4**	819	3999	**229**	248	12	0.3
RTU++ (ours)	**77.0**	70.0	**666**	4326	261	234	13	**11.7**

TABLE VIII
TRACKING PERFORMANCE ON CTMC-V1 DATASET. ALL METHODS TAKE THE SAME INITIAL INPUTS

Method	IDF1↑	MOTA↑	IDs↓	FP↓	FN↓
IOU [29]	87.7	**95.7**	754	29877	**16376**
DeepSORT [19]	80.3	89.9	1170	31123	82854
RTU++ (ours)	**89.3**	95.4	**200**	**28013**	23035

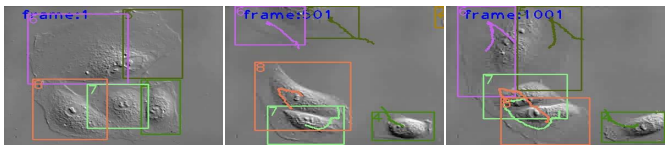


Fig. 9. The tracking result of CV-1-run01 sequence. Even though the shapes of cells vary greatly, RTU++ still obtains accurate trajectories.

by 67.0 and the lowest IDs by 598. Since our method utilizes different types of nodes to model the association, it is suitable to describe the object interaction in complex scenes. The results show that RTU++ get the lowest Frag simultaneously. Compared to other methods, IDs and Frag indicators drop about 50%. This proves that RTU++ has advantage on maintaining the identities of objects. Naturally, although MOTA indicator is slightly lower than ByteTrack, RTU++ acquires a comparable performance on IDF1 by 67.0. This shows the stable performance of RTU++ in large-scale complex scenes.

E. Generalization in MOTS and Cell Tracking

In order to validate the generalization ability of our method, we conduct the comparison on the MOTS20 and CTMC-v1 benchmarks.

For MOTS20, our method takes the same initial masks as MAF_HDA [59]. Tab. VII shows the entire results on MOTS20 benchmark. As discussed in Sec. III-D, RTU++ is extended to MOTS with some necessary changes. Although MOTS is more challenging than MOT, RTU++ still achieves the best performance on IDF1 by 77.0. We attribute this to the robustness of our data association mechanism. Compared with other methods, RTU++ conducts the fastest processing speed, since it does not have much complex computation for object masks. Especially, with the same mask inputs as MAF_HDA, our method obtains 10% inprovement on IDF1 and nearly 35% reduction on IDs, which proves the efficiency of RTU++. Fig. 8 shows the tracking results on the MOTS20 benchmark. We find that RTU++ still accurately tracks targets even in different scenes.

As shown in Tab. VIII, we compare our methods with two other famous open-sourced trackers. All the trackers are provided the same detection from YOLOV4 [63]. The IoU tracker only uses motion information but DeepSORT and ours use appearance information. For a fair comparison, all appearance features are extracted by a ResNet50 pre-trained on ImageNet.

The results show that RTU++ achieves the best performance on IDF1 and IDs indicators. We can see that DeepSORT obtains a worse performance than IoU tracker. That is the appearance model trained on ImageNet is not suitable for cells. However, our RTU++ also utilize the appearance information, but still obtains outstanding result. Even the motion and appearance pattern of cells is of great difference from pedestrian, our methods still correctly track most cells. We attribute this to our general tracking and score methodology. First, the EMNT is still suitable for cells. Second, as discussed in Sec. III-B and Sec. IV-B, the state feature and simulated tracking data decrease the influence of dataset style. Fig. 9 shows a visualization result of our tracker, we can see it tracks most cells even there is a great size change in some cells. This demonstrates that our method can generalize well to wilder datasets.

VI. CONCLUSION AND DISCUSSION

In this paper, we propose an Extendable Multiple Nodes Tracking framework to illuminate tracking. We define four types of basic nodes to build the essential association, in which

various clues are integrated into state features. Furthermore, we design RTU++ to solve the scoring problems in tracking. It utilizes the historical tracking clues in state feature and remembers node types information into memory. Then tracks can be scored by RTU++ recurrently. In addition, we present a Node-based General Tracking Data Generation method to overcome the lack of available data in MOT. By analyzing the experimental results, our EMNT combined with RTU++ shows convincing performance in MOT17, MOT20, HiEve, MOTS20 and CTMC-v1. We also conduct more detailed experiments to demonstrate the generalization of our methodology. In these experiments, we find that our methods often obtain larger FP. This is caused by the imprecise estimation of track state under illumination changes or deformation, which are deeply studied in single object tracking. These advanced technologies [64]–[66] provide a new direction to further enhance our method. Moreover, it is also a good choice to extend RTU++ to wider fields such as CATER [67], so as to perform more tasks of video analysis.

REFERENCES

[1] W. Ren, X. Wang, J. Tian, Y. Tang, and A. B. Chan, "Tracking-by-counting: Using network flows on crowd density maps for tracking multiple targets," *IEEE Trans. Image Process.*, vol. 30, pp. 1439–1452, 2021.

[2] P. Dai, R. Weng, W. Choi, C. Zhang, Z. He, and W. Ding, "Learning a proposal classifier for multiple object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2443–2452.

[3] Y. Zhang, H. Sheng, Y. Wu, S. Wang, W. Ke, and Z. Xiong, "Multiplex labeling graph for near-online tracking in crowded scenes," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7892–7902, Sep. 2020.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3645–3649.

[5] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "FairMOT: On the fairness of detection and re-identification in multiple object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 11, pp. 3069–3087, 2021.

[6] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4696–4704.

[7] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *Proc. CVPR*, Jun. 2011, pp. 1201–1208.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Y. Zhang *et al.*, "Long-term tracking with deep tracklet association," *IEEE Trans. Image Process.*, vol. 29, pp. 6694–6706, 2020.

[10] H. Sheng *et al.*, "Near-online tracking with co-occurrence constraints in blockchain-based edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2193–2207, Feb. 2021.

[11] S. Wang, H. Sheng, Y. Zhang, Y. Wu, and Z. Xiong, "A general recurrent tracking framework without real data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 1–8.

[12] G. Bishop *et al.*, "An introduction to the Kalman filter," in *Proc. SIGGRAPH, Course*, vol. 8, 2001, p. 41.

[13] M. Hoshiya and E. Saito, "Structural identification by extended Kalman filter," *J. Eng. Mech.*, vol. 110, no. 12, pp. 1757–1770, Dec. 1984.

[14] P. M. Djuric *et al.*, "Particle filtering," *IEEE Signal Process. Mag.*, vol. 20, no. 5, pp. 19–38, Sep. 2003.

[15] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artif. Intell.*, vol. 293, Apr. 2021, Art. no. 103448.

[16] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 91–99.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2961–2969.

[19] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3464–3468.

[20] W. Choi, "Near-online multi-target tracking with aggregated local flow descriptor," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3029–3037.

[21] H. Sheng, J. Chen, Y. Zhang, W. Ke, Z. Xiong, and J. Yu, "Iterative multiple hypothesis tracking with tracklet-level association," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3660–3672, Dec. 2019.

[22] H. Sheng, Y. Zhang, J. Chen, Z. Xiong, and J. Zhang, "Heterogeneous association graph fusion for target association in multiple object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 11, pp. 3269–3280, Nov. 2019.

[23] H. Sheng *et al.*, "Hypothesis testing based tracking with spatio-temporal joint interaction modeling," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 2971–2983, Sep. 2020.

[24] B. Yang and R. Nevatia, "An online learned CRF model for multi-target tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2034–2041.

[25] A. Dehghan, S. M. Assari, and M. Shah, "GMMCP tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4091–4099.

[26] Z. Jiang *et al.*, "Detecting and tracking of multiple mice using part proposal networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 29, 2022, doi: 10.1109/TNNLS.2022.3160800.

[27] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 107–122.

[28] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 941–951.

[29] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2017, pp. 1–6.

[30] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic, "On pairwise costs for network flow multi-object tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5537–5545.

[31] C. Kim, F. Li, and J. M. Rehg, "Multi-object tracking with neural gating using bilinear LSTM," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 200–215.

[32] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 300–311.

[33] N. Ran, L. Kong, Y. Wang, and Q. Liu, "A robust multi-athlete tracking algorithm by exploiting discriminant features and long-term dependencies," in *Proc. Int. Conf. Multimedia Model. (ICME)*. Cham, Switzerland: Springer, 2019, pp. 411–423.

[34] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust multi-modality multi-object tracking," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2365–2374.

[35] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 480–496.

[36] S. E. Kahou, V. Michalski, R. Memisevic, C. Pal, and P. Vincent, "RATM: Recurrent attentive tracking model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1613–1622.

[37] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 366–382.

[38] K. Yoon, D. Y. Kim, M. Jeon, and Y. C. Yoon, "Data association for multi-object tracking via deep neural networks," *Sensors*, vol. 19, no. 3, p. 559, 2019.

[39] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," 2016, *arXiv:1603.00831*.

[40] P. Dendorfer *et al.*, "MOT20: A benchmark for multi object tracking in crowded scenes," 2020, *arXiv:2003.09003*.

[41] W. Lin *et al.*, "Human in events: A large-scale benchmark for human-centric video analysis in complex events," 2020, *arXiv:2005.04490*.

[42] S. Anjum and D. Gurari, "CTMC: Cell tracking with mitosis detection dataset challenge," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 982–983.

[43] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear MOT metrics," *EURASIP J. Image Video Process.*, vol. 2008, pp. 1–10, Dec. 2008.

[44] P. Voigtlaender *et al.*, "MOTS: Multi-object tracking and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7942–7951.

[45] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2016, pp. 17–35.

[46] J. Luiten *et al.*, "HOTA: A higher order metric for evaluating multi-object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 548–578, Feb. 2021.

[47] B. Pang, Y. Li, Y. Zhang, M. Li, and C. Lu, "TubeTK: Adopting tubes to track multi-object in a one-step training model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6308–6318.

[48] Y. Wang, K. Kitani, and X. Weng, "Joint object detection and multi-object tracking with graph neural networks," 2020, *arXiv:2006.13164*.

[49] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 474–490.

[50] J. Wu, J. Cao, L. Song, Y. Wang, M. Yang, and J. Yuan, "Track to detect and segment: An online multi-object tracker," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 12352–12361.

[51] C. Liang, Z. Zhang, X. Zhou, B. Li, S. Zhu, and W. Hu, "Rethinking the competition between detection and Reid in multi-object tracking," 2020, *arXiv:2010.12138*.

[52] Y. Zhang *et al.*, "ByteTrack: Multi-object tracking by associating every detection box," 2021, *arXiv:2110.06864*.

[53] F. Yang, X. Chang, S. Sakti, Y. Wu, and S. Nakamura, "ReMOT: A model-agnostic refinement for multiple object tracking," *Image Vis. Comput.*, vol. 106, Feb. 2021, Art. no. 104091.

[54] B. Shuai *et al.*, *Application of Multi-Object Tracking With Siamese Track-RCNN to the Human in Events Dataset*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 4625–4629, doi: 10.1145/3394171.3416297.

[55] Y. Jing, B. Guo, Y. Liu, Z. Wang, Z. Yu, and X. Zhou, "CrowdTracker: Object tracking using mobile crowd sensing," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. Proc. ACM Int. Symp. Wearable Comput.*, 2017, pp. 85–88.

[56] A. Wu, C. Lin, B. Chen, W. Huang, Z. Huang, and W.-S. Zheng, "Transductive multi-object tracking in complex events by interactive self-training," in *Proc. 28th ACM Int. Conf. Multimedia*, Oct. 2020, pp. 4620–4624.

[57] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," 2021, *arXiv:2107.08430*.

[58] Y.-m. Song, Y.-c. Yoon, K. Yoon, M. Jeon, S.-W. Lee, and W. Pedrycz, "Online multi-object tracking and segmentation with GMPHD filter and mask-based affinity fusion," 2020, *arXiv:2009.00100*.

[59] Y.-M. Song, Y.-C. Yoon, K. Yoon, H. Jang, N. Ha, and M. Jeon, "Multiobject tracking and segmentation with embedding mask-based affinity fusion in hierarchical data association," *IEEE Access*, vol. 10, pp. 60643–60657, 2022.

[60] Z. Wang, H. Zhao, Y.-L. Li, S. Wang, P. Torr, and L. Bertinetto, "Do different tracking tasks require different appearance models?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 726–738.

[61] (2021). *Costa Tracker*. [Online]. Available: https://motchallenge.net/method/MOTS=87&chl=17

[62] F. Yang *et al.*, "ReMOTS: Self-supervised refining multi-object tracking and segmentation," *CoRR*, vol. abs/2007.03200, 2020. [Online]. Available: https://arxiv.org/abs/2007.03200

[63] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[64] J. Shen, Y. Liu, X. Dong, X. Lu, F. S. Khan, and S. C. H. Hoi, "Distilled Siamese networks for visual tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Nov. 11, 2021, doi: 10.1109/TPAMI.2021.3127492.

[65] W. Han, X. Dong, F. S. Khan, L. Shao, and J. Shen, "Learning to fuse asymmetric feature maps in Siamese trackers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 16570–16580.

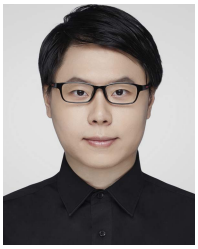
[66] X. Dong, J. Shen, L. Shao, and F. Porikli, "CLNet: A compact latent network for fast adjusting Siamese trackers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 378–395.

[67] R. Girdhar and D. Ramanan, "CATER: A diagnostic dataset for compositional actions and TEmporal reasoning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–16.

Shuai Wang received the B.S. degree from the School of Computer Science and Engineering, Beihang University, China, in 2019, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering. His research interests include computer vision and multiple object tracking.

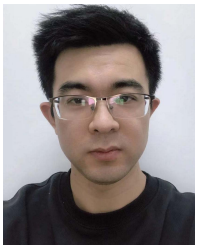
Hao Sheng (Member, IEEE) received the B.S. and Ph.D. degrees from the School of Computer Science and Engineering, Beihang University, in 2003 and 2009, respectively. He is currently a Professor and the Ph.D. Supervisor with the School of Computer Science and Engineering, Beihang University, China. His research interests include computer vision, pattern recognition, and machine learning.

Da Yang received the B.S. degree from the School of Computer Science and Engineering, Beihang University, Beijing, China, in 2012, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering.



Yang Zhang received the B.S. and Ph.D. degrees from the School of Computer Science and Engineering, Beihang University, in 2014 and 2020, respectively. He is an Associate Professor with the College of Information Science and Technology, Beijing University of Chemical Technology, China. He is working on computer vision and machine learning.

Yubin Wu received the B.S. degree from the School of Computer Science and Engineering, Beihang University, China, in 2016, where he is currently pursuing the Ph.D. degree. His research interests include computer vision and multiple object tracking.



Sizhe Wang received the B.S. degree from the Xi'an University of Technology in 2012 and the M.S. degree from Xi'an Jiaotong University in 2014. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Beihang University, China. His research interest includes computer vision.