**Graph Convolution Neural Network-Based Data Association for Online Multi-Object Tracking**

**JIMI LEE, MIRA JEONG, AND BYOUNG CHUL KO**, (Member, IEEE)

Department of Computer Engineering, Keimyung University, Daegu 42601, South Korea

Corresponding author: Byoung Chul Ko (niceko@kmu.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), Ministry of Education, under Grant 2019R1I1A3A01042506.

**ABSTRACT** In this paper, a graph convolutional network (GCN)-based multi-object tracking (MOT) algorithm, consisting of a module for extracting the initial features and a module for updating the features, that estimates the affinity between nodes is proposed. The feature extraction module utilizes the pose feature of the object such that the tracking is correct even when the object is partially occluded. Unlike previous graph neural network (GNN)-based MOT methods, this study is based on a GCN and includes a new feature update mechanism, which is updated by combining the output of the neural network and the node similarity between the tracker and detection nodes for each layer. The node feature is updated by aggregating the updated edge feature and the connection strength between the tracker and detection. In each GCN layer, the three networks for the node, edge update, and edge classification were designed to minimize the network parameters to enable faster MOT compared to other GCN-based MOTs. The entire GCN network was designed to learn end-to-end through an affinity loss. The experimental results for the MOT16 and 17 challenge datasets show that the proposed method achieves a superior or similar performance in terms of tracking accuracy and speed compared to state-of-the-art methods, including GCN-based MOT.

**INDEX TERMS** Multiple object tracking, graph neural network, graph convolution neural network, data association.

**VOLUME 9, 2021 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/ 

Received August 5, 2021, accepted August 12, 2021, date of publication August 16, 2021, date of current version August 24, 2021. Digital Object Identifier 10.1109/ACCESS.2021.3105118

**I. INTRODUCTION**

Object tracking can be primarily divided into single-object tracking (SOT) and multi-object tracking (MOT). In terms of practical applications, including video surveillance, autonomous vehicles, and robot navigation, the MOT, which can track multiple objects simultaneously, is receiving more attention than the SOT, which tracks only one object. The tracking-by-detection paradigm, the most common approach in MOT, largely depends on two performances. The first is the object detection performance. The object must be accurately detected in every frame such that the tracking avoids breaking or being incorrectly connected during subsequent tracking operations. Various object detectors [1]–[4] with a high performance based on a CNN have recently been introduced, and the degradation of the tracker from an erroneous object detection has been resolved to a certain extent. However, object detectors can still detect incorrect objects or miss objects owing to object occlusions or camera shaking. The second is the data association performance. To compensate for the inaccuracy of the MOT caused by a false object detection, the data association can link previously obtained trajectories and new detection responses. The importance of real-time data association in online MOT is emphasized more than in offline MOT based on a global association. The key issue in real-time data association is determining the optimal association between detections and trackers. The most representative data association methods are bipartite assignment [5], [6] and Hungarian-based approaches [7]–[10]. These methods model weights as an affinity matrix between graph sets consisting of existing trajectory nodes and new detection nodes [11]. Matching between nodes in the graph set is determined according to the weight of the affinity matrix. Dynamic programming [12]–[14] is used to find and match the shortest path between the detection and tracklet. A min-cost flow [15]–[17] and conditional random field [18]–[20] deal with data association as a graph. This approach expresses the detections or tracklets as nodes of the graph, and a flow model or label predicts the edge strength, and nodes with high strength are linked.

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.
Deep learning technologies applied in a data association for MOT have recently achieved a state-of-the-art performance. The Siamese network [10], [21], [22] and its modified triplet [23], [24] and quadruplet network [25] applied the same network to the detection and tracker and calculated the similarity based on the difference in the output characteristics. The more complex the structure of the shared network is, the higher the matching performance; however, real-time online tracking becomes more difficult. A Siamese-network-based data association can encode reliable pairwise interactions between objects but is unsuitable when accounting for high-order information in which multiple objects and states exist in a scene [26].

Therefore, graph network (GNN) and graph convolutional network (GCN)-based MOT approaches have recently been attempted to solve the data association of MOT in terms of the appearance of objects and their geometric information [26]–[33]. GNN-based MOT methods commonly generate a graph structure using object nodes and edges based on the geometric relations of the nodes. A CNN is used to extract node features from appearance features, and temporal or geometric information is used to extract the edge features. The information embedded with the GNN is used to calculate the affinities of the detections and tracklets while considering the interactions of the nodes in the network through the GNN. In the next section, we will describe related studies focusing on a GNN or GCN-based MOT, as a recent issue.

A. CONTRIBUTIONS OF THIS STUDY

Several recent studies have proposed GCN-based MOT approaches and have demonstrated outstanding results. However, most GCN-based MOT methods [26], [27], [32] use geometric information considering only limited values, such as the position and scale of the bounding box of an object as edge features for updating the processing of nodes and edges. These methods cannot accurately reflect the local features of objects and make real-time processing difficult owing to the connection of a large number of nodes. Therefore, in this paper, we propose a GCN-based MOT that reflects the local characteristics of objects and enables faster online tracking compared to other GCN-based MOTs.

A summary of the contributions of this paper is as follows.

- The data association time was reduced by minimizing the number of networks for feature aggregation and edge classification.
- The proposed method is lighter and achieves significantly improved tracking results compared with other GCN-based state-of-the-art methods based on experiments conducted using various MOT benchmark datasets.

The remainder of this paper is structured as follows. In Section II, MOT-related studies focusing on GNN-based tracking are reviewed. In Section III and IV, we present the details of our proposed method in terms of feature and node updating in proposed GCN MOT architecture. A comprehensive evaluation of the proposed method is provided in Section V. Finally, conclusion and future works are remarks given in Section VI.

II. RELATED STUDIES

GNN has been consistently applied not only in MOT but also in SOT. As representative methods of SOT, Gao et al. [34] proposed a new graph convolutional tracking for high-performance visual tracking in order to comprehensively utilize the spatial-temporal structure of an object and to take advantage of context information. In addition, Guo et al. [35] proposed a Siamese graph attention network to establish a partial correspondence between an object and a search region as a complete bipartite graph. SOT methods using GNN show superior performance compared to existing SOT approaches. However, because this paper focuses on MOT research, this session will focus on state-of-the-art studies related to MOT.

A. STATE-OF-THE ART MOT APPROACHES

CenterTrack [36] additionally applies the CenterNet model to detection of pairs of images and previous frames. This method coordinates the detector with two successive frames and a heatmap of the previous tracklet represented by points. For object association, object tracking speed is greatly improved through greedy matching based on the distance between the predicted offset and the center point detected in the previous frame.

Kim et al. [37] proposed a multitrack pooling module to solve the problem of considering all tracks simultaneously during tracker memory update with small space overhead. This method also proposed a training strategy suitable for multi-track pooling to generate hard-tracking episodes online.

Yang et al. [38] presented a network structure that aggregates inter-frame object regression and appearance feature extraction into one model to improve the harmony between the various feature modules of MOT. This method also proposed an end-to-end training method that expands the training data by combining the target’s past position and the motion prediction model during training.

Saleh et al. [39] proposed a probabilistic autoregressive motion model to score MOT tracklet proposals. This method
not only assigns new detections to existing tracklets, but also reconnects tracklets by sampling the tracklets even when objects are occluded.

B. GNN-BASED MOT
Li et al. [28] proposed an MOT method consisting of two graph networks, an appearance graph network, and a motion graph network. The two GNNs compute two similarities between objects and trackers with a four-step updating module in order of edge I, node, global, and edge II. The affinity matrix is obtained from the weighted result of combining the two similarity values.

Jiang et al. [32] proposed an end-to-end data association model as output affinity scores using appearance and motion affinity learning modules based on a Siamese network. With this method, the GNN is used as an optimization module to solve the configured maximum weighted binary matching.

Weng et al. [33] proposed a GNN-based 3D MOT with 2D-3D feature learning. This method uses a feature extractor that applies motion and appearance features in a 2D and 3D space and feeds them into a GNN for feature interaction. These node features were iteratively updated using the feature aggregation technique, and the affinity matrix was computed using the edge regression module.

Braso and Leal-Taxie [26] introduced an MOT solver based on a message passing network with a time-aware update step that can handle the temporal structure of MOT graphs. As the main idea of this method, the node and edge updating process considers the interaction between observations in past and future frames. The embeddings in the past and future frames were separately aggregated. Moreover, this method directly learns to apply a data association using an edge classification. However, because this method is an offline MOT approach, it is unsuitable for real-time online MOT.

The use of MOT with GCNs, which is a modified version of a GNN, has also been attempted. A GCN is the application of the convolution concept of a CNN to a graph network. Papakis et al. [27] proposed a graph convolutional neural network (GCNN)-based MOT. The GCNN produces interactive features through a feature aggregation by updating the shared appearance information of the node features with their linking edge. Edge features share geometric information with their incident nodes for pairs of object tracklets. Embedded interaction features are composed of an affinity matrix after calculating the similarity score between pairs.

Dai et al. [40] proposed an iterative GCN clustering method to reduce computational cost and rank according to the estimated quality score while maintaining the quality of the generated proposals. This method adopted a simple de-overlapping strategy to generate the trace output.

He et al. [41] proposed a GCN-based MOT that models the relationship between tracklets and intra-frame detections as a general undirected graph. Similar to CenterTrack [36], this method uses public detection refinement to improve tracking performance and speed. Therefore, direct comparison with MOT methods using only public detection including the proposed method cannot be performed.

III. METHOD OF THE PROPOSED SYSTEM
In this section, we describe the overall MOT procedure based on the proposed GCN. Section 3.1 introduces the overview of the proposed system and Section 3.2 explains the first feature embedding module consists of node feature and edge feature. Section 3.3 explains the second feature updating module and affinity loss is described in Section 3.4.

A. OVERVIEW OF THE PROPOSED SYSTEM
Let us suppose that the tracker set T of the t-1 frame is composed of N trackers $T^{t-1} = \{t_{r_1}^{t-1}, t_{r_2}^{t-1}, \ldots, t_{r_N}^{t-1}\}$ and the detection set D of the t frame composed of M objects $D^t = \{d_1, d_2, \ldots, d_M\}$. Each object is set as $\{x, y, w, h\}$, where $x$ and $y$ are the center positions, and $w$ and $h$ are the width and height, respectively, of an object bounding box. When $t = 0$, the elements of tracker set $T^0$ are initialized to the elements of $D^0$. The process of matching the tracker and detection starts from frame $t > 1$.

Our GCN structure consists of two modules. In the first module, the initial node features for the tracker and detection are extracted using the backbone network ResNet-18 of a mask-RCNN, and the initial edge features are extracted using the keypoint distance between two objects, as described in Section 3.2. In the second module, we update the edge and node features using a GCN consisting of $L$ layers. The updated edge feature is input to the edge classifier network to measure the similarity between the two nodes. The final affinity matrix is formed by the similarities between the node outputs through the edge classifier.

In GCN-based MOT, for data association between nodes (objects) at times $t$ and $t-1$, a node feature representing the characteristics of a node and an edge feature representing the affinity between nodes are generally used. The goal of our GCN is to estimate the N × M affinity matrix $A = \{a_{ij}\}_{|T|×|D|}$ based on the pairwise similarity of the features extracted from N trackers and M detection objects in online MOT. Here, $|T|$ and $|D|$ are the number of trackers and detections, respectively. As an element of $A$, $a_{ij} = 1$ when the detection $d_{j}$ is associated with tracker $t_{r_i}^{t-1}$ and $a_{ij} = 0$ when two objects are not associated. A is generated from the output layer of the GCN, and the Hungarian algorithm is used to link detections and trackers. Figure 1 shows an overview of the proposed MOT system. In the next section, we describe each module in Fig. 1 in detail.

B. FEATURE EMBEDDING
1) NODE FEATURE
To extract the node features, we first normalize the size of the bounding boxes of the detection and tracker and feed them into the CNN structure. As the backbone network, we used the ResNet-18 of the Mask-RCNN structure [2]. In this way, the appearance features of a high dimensionality
are converted into feature vectors of low dimensionality with the abstract information of an object. The converted features of each detection and tracker object are called the initial node features. The dimensions of each node feature vector are 512. For every object, the initial node feature of the tracker is $n^0_i$ and a detection is $n^0_j$.

### 2) EDGE FEATURE

For the initial edge feature, most GNN-based MOT approaches use motion features [28] or a combination of the appearance and geometric features \{x, y, w, h\} [26], [27]. Kim et al. [29] even initialized edge features using edge labels depending on the node connections. Instead of initializing the edge feature by concatenating the node and geometric features, we combine the pose feature of the objects inferred from joint keypoints and the appearance feature inferred from the output of the Mask-RCNN to consider the local pose variation of the node objects and global appearance together. The pose feature supplements object information that is partially occluded by other objects. First, each normalized object bounding box is fed to the same Mask-RCNN, and the positions of 17 key points and score pairs \{x, y, score\} are obtained. The $17 \times 3$ output matrix is flattened into a 51-dimensional pose vector. In contrast to GCNNMatch [27], we compute the element-wise distance vector. The first distance vector is for the pose vector of the i-th tracker at time $t-1$, $p^0_i$, and the j-th detection object at time $t$, $p^0_j$. The second distance vector is for the appearance feature (initial node) vectors $n^0_i$ and $n^0_j$. The output distance vector $e^0_{ij}$ served as the initial edge feature.

\[
e^0_{ij} = \sigma \left( \| p^0_i \circ p^0_j \| \right) \oplus \sigma \left( \| n^0_i \circ n^0_j \| \right), \tag{1}
\]

where $\sigma$ is the sigmoid function used to normalize the value into [0, 1]. The dimensions of the initial edge feature were 563 ($51 + 512$).

For the real-time edge connection (link) between each pair of nodes, we simply obtain a Euclidean distance based on the location information between the $n_i$ and $n_j$ nodes. To make a sparse edge connection, the edges are set at the K-nearest neighbor nodes of $n_i$ based on $n_j$.

The initial node and edge features are updated by reflecting similar node characteristics while passing through the GCN layer in the learning process.

### C. FEATURE UPDATING

#### 1) EDGE UPDATING

Unlike existing methods for updating the output of the MLP with a new node feature [26]–[29], the proposed method applies the similarity between the previous edge feature and the node feature to the edge feature update such that the new edge feature reflects the intra-feature variation. In addition, the edge feature value is prevented from rapidly drifting by considering the initial edge features.

Given $n^{\ell-1}_i$ and $e^{\ell-1}_{ij}$ from layer $\ell-1$, the edge feature $e^\ell_{ij}$ at layer $\ell$ is updated by combining the initial edge feature $e^0_{ij}$, the Euclidean similarity $\sigma$ between a node $n^{\ell-1}_j$ and a connected node $n^{\ell-1}_i$, and the previous edge feature $e^{\ell-1}_{ij}$. The role of $e^0_{ij}$ is to prevent the edge feature from changing in the wrong direction owing to a mismatch during the edge update process. The new edge features are normalized by integrating the transformed edge features using a single-layer network $f^\ell_e$ with the edge features of neighboring nodes.
K-similar nodes. The edge feature is updated using the following proposed formula:

$$
\mathbf{e}_{ij}^\ell = \frac{\mathbf{e}_{ij}^{\ell-1} \cdot s(\mathbf{n}_i^{\ell-1}, \mathbf{n}_j^{\ell-1}) - \mathbf{e}_{ij}^{\ell-1}}{\sum_{k \in K} \mathbf{e}_{ik}^{\ell-1} \cdot \mathbf{f}_j^\ell(\mathbf{e}_{ik}^{\ell-1} \cdot s(\mathbf{n}_i^{\ell-1}, \mathbf{n}_k^{\ell-1}) ; \theta_e) / \sum_{k \in K} \mathbf{e}_{ik}^{\ell-1}}
$$

(2)

where $f_j^\ell$ represents a learnable network consisting of a single-layer network using parameters $\theta_e$ and the ReLU activation function. This process is shared across the entire graph.

2) NODE UPDATING

Unlike a node-feature-based node similarity classification [27], [29], [32], [33], we apply an edge feature-based node similarity classification. After updating the edge feature $\mathbf{e}_{ij}^\ell$, the updated edge feature is reflected in the update of the node feature again. The node feature $\mathbf{n}_j^\ell$ at layer $\ell$ is updated by aggregating the previous node feature and the transformed edge feature using another network, $f_n^\ell$, when considering the node affinity similarity of K neighborhood nodes. The node feature is updated using the following proposed formula:

$$
\mathbf{n}_j^\ell = \mathbf{n}_j^{\ell-1} + \sum_{k \in K} f_n^\ell(\mathbf{e}_{ij}^\ell ; \mathbf{A}_{jk} ; \theta_e),
$$

(3)

where $f_n^\ell$ is a learnable two-layer MLP network using the parameters $\theta_e$ and the ReLU activation function. The updated node feature is used as the new node feature in the next layer.

The detailed structure of the edge and node feature update processes is shown in Fig. 2.

3) EDGE PAIRWISE SIMILARITY

The updated edge features $\mathbf{e}_{ij}^\ell$ output from the $\ell$-layer GCN are fed to the two-layer MLP edge classification network $f_s$ for prediction of edge affinity $a_{ij}$.

$$
a_{ij} = f_s \left( \mathbf{e}_{ij}^\ell ; \theta_s \right) = \text{Sigmoid} \left( \text{ReLU} \left( \mathbf{e}_{ij}^\ell W^0 \right) W^1 \right).
$$

(4)

where $W^0$ and $W^1$ of $\theta_s$ represent two different learnable parameters of the MLP $f_s$.

The output of the edge pairwise similarity $a_{ij} \in [0, 1]$ between the i-th and j-th detection nodes has a scalar value between 0 and 1, and the higher the value, the higher the similarity between node i and node j.

To make the $N \times M$ affinity matrix $\mathbf{A}$, we predict pairwise similarities between $N$ trackers at time $t-1$ and $M$ detections at time $t$. The affinity between nodes is calculated with the updated edge features at each layer. The affinity matrix $\mathbf{A}$ output from the last GCN layer $L$ is used for a data association of the online MOT.

D. AFFINITY LOSS

Among the two modules of the proposed system, the mask-RCNN of the first module uses a pre-trained model, and the remaining two modules need to be newly trained for online MOT. To learn the parameters of the MLPs used in the GCN layer, we used binary cross-entropy in an end-to-end manner. First, we create a ground-truth affinity matrix $\mathbf{A}^{gt}$ that has a value of 0 or 1 according to the connection state of the tracker and detection nodes of the training data. The affinity matrix $\mathbf{A}$ is created by collecting the pairwise edge similarity $a_{ij}$ predicted by feeding the updated edge features to the edge classification network for all edge $(i, j)$ pairs. Therefore, the binary cross-entropy loss for learning two modules calculates the difference between each entry $(a_{ij}^{gt}, a_{ij})$ in the predicted affinity matrix $\mathbf{A}$ and the ground-truth matrix $\mathbf{A}^{gt}$.

$$
L = -\frac{1}{N \times M} \sum_i \sum_j a_{ij}^{gt} \log (a_{ij}) + (1-a_{ij}^{gt}) \log (1-a_{ij}),
$$

(5)

The detailed procedures for training the GCN-based affinity matrix are described in Algorithm 1.
Algorithm 1 Training of GCN for Affinity Matrix

| Input: Tracker set $T^{t-1} = \{t_1, t_{r1}, \ldots, t_N\}$  
| Detection set $D^t = \{d_1, d_{r1}, \ldots, d_M\}$  
| Parameters: $\theta_f$ for $f_a$, $\theta_f$ for $f_b$, and $\theta_c$ for edge classifier $f_c$
| Affinity matrix: $A_{ij}^{\theta_f}, i \leq |T|, j \leq |D|$  
| Initialize: $A_{ij}, e_{ij}^{\theta_f}, n_i^0, f_a(x_i; \theta_f), f_b(x_j; \theta_f), \forall i, j$

**Step 1:**
for $i = 1, \ldots, N$ do  
// Initial node feature
$n_i^0, P_i \leftarrow \text{Mask} - \text{RCNN} (t_i)$
end
for $i = 1, \ldots, M$ do  
$n_j^0, P_j \leftarrow \text{Mask} - \text{RCNN} (d_j)$
end
for $i = 1, \ldots, N, j = 1, \ldots, M$ do  
// Initial edge feature
$e_{ij}^{\theta_f}$ using Eq. (1)
end

**Step 2:**
for $\ell = 1, \ldots, L$ do  
// Layers of GCN
for $i = 1, \ldots, N, j = 1, \ldots, M$ do  
// Edge feature update
$e_{ij}^{\ell} \leftarrow \text{EdgeUpdate}(e_{ij}^{\ell-1}, n_i^{\ell-1}, n_j^{\ell-1}, e_{ij}^{\ell-1}; \theta_c)$
end
for $i = 1, \ldots, M, j = 1, \ldots, N$ do  
// Node feature update
$n_i^\ell \leftarrow \text{NodeUpdate}(e_{ij}^{\ell}, n_i^{\ell-1}, n_j^{\ell-1}; \theta_n)$
end
$A_{ij} \leftarrow \text{EdgeClassification}(e_{ij}^{\ell}; \theta_c)$
end

**Step 3:** // Affinity Loss
$L \leftarrow \text{ComputeLoss}(A_{ij}^{\theta_f}, A_{ij})$
// Repeat steps 2-3 to reach the minimum loss

**Output:** Trained GCN ($\theta_f$; $\theta_c$; $\theta_n$)

### IV. ONLINE MOT MANAGEMENT

For every frame, we detect an object using a mask-RCNN and apply a data association for MOT using the affinity matrix of a GCN. This study applies step-by-step affinity measures for data association. As a first step, we obtain the affinity matrix $A$ output from the last GCN layer between the trackers and detections. Then, only detections within a certain radius of the i-th tracker are defined as valid matching pairs of affinity $A$. In addition, even if the affinity score is the smallest pair, if the i-th tracker are defined as valid matching pairs of affinity $A$. This study applies step-by-step affinity measures for data association.

Algorithm 1: Training of GCN for Affinity Matrix

**Input:** Tracker set $T^{t-1} = \{t_1, t_{r1}, \ldots, t_N\}$  
Detection set $D^t = \{d_1, d_{r1}, \ldots, d_M\}$  
Parameters: $\theta_f$ for $f_a$, $\theta_f$ for $f_b$, and $\theta_c$ for edge classifier $f_c$

**Affinity matrix:** $A_{ij}^{\theta_f}, i \leq |T|, j \leq |D|$

**Initialize:** $A_{ij}, e_{ij}^{\theta_f}, n_i^0, f_a(x_i; \theta_f), f_b(x_j; \theta_f), \forall i, j$

**Step 1:**
for $i = 1, \ldots, N$ do  
// Initial node feature
$n_i^0, P_i \leftarrow \text{Mask} - \text{RCNN} (t_i)$
end
for $i = 1, \ldots, M$ do  
$n_j^0, P_j \leftarrow \text{Mask} - \text{RCNN} (d_j)$
end
for $i = 1, \ldots, N, j = 1, \ldots, M$ do  
// Initial edge feature
$e_{ij}^{\theta_f}$ using Eq. (1)
end

**Step 2:**
for $\ell = 1, \ldots, L$ do  
// Layers of GCN
for $i = 1, \ldots, N, j = 1, \ldots, M$ do  
// Edge feature update
$e_{ij}^{\ell} \leftarrow \text{EdgeUpdate}(e_{ij}^{\ell-1}, n_i^{\ell-1}, n_j^{\ell-1}, e_{ij}^{\ell-1}; \theta_c)$
end
for $i = 1, \ldots, M, j = 1, \ldots, N$ do  
// Node feature update
$n_i^\ell \leftarrow \text{NodeUpdate}(e_{ij}^{\ell}, n_i^{\ell-1}, n_j^{\ell-1}; \theta_n)$
end
$A_{ij} \leftarrow \text{EdgeClassification}(e_{ij}^{\ell}; \theta_c)$
end

**Step 3:** // Affinity Loss
$L \leftarrow \text{ComputeLoss}(A_{ij}^{\theta_f}, A_{ij})$
// Repeat steps 2-3 to reach the minimum loss

**Output:** Trained GCN ($\theta_f$; $\theta_c$; $\theta_n$)

### V. EXPERIMENTS

In this section, the MOT benchmark dataset (MOT-BD) [42] is used to evaluate the tracking performance. To prove the validity of the proposed method, we discuss various experiments conducted, including ablation studies and comparisons with state-of-the-art methods, and elaborate on these results.

#### A. DATASETS AND EVALUATION METRICS

**1) DATASETS**

Most MOT studies use MOT-BD to verify the performance of the proposed method. MOT-BD provides a framework that includes datasets and evaluation tools that ensure a fair evaluation for comparing the MOT performance. We also used the MOT16 and MOT17 datasets provided by MOT-BD for all experiments. The MOT16 data set consists of a set of 14 sequences and contains more complex scenarios, different perspectives, camera movements, and weather conditions. All sequences are annotated to strict standards by experts. The MOT16 annotates not only pedestrians, but also vehicles, sitting persons, occluded objects, etc. MOT17 dataset is an extended version of the MOT16 dataset. This dataset contains 14 testing sequences of an urban environment with different viewpoints, the size and number of pedestrians, camera movements, and frame rates. It also provides three types of detection results: DPM [43], Faster-RCNN [44], and SDP [45]. Most MOT methods using MOT-BD apply these detection results simply to measure the tracking performance regardless of the detection accuracy.

**2) EVALUATION METRICS**

For a quantitative evaluation of the proposed method, we considered CLEAR MOT metrics [46], including the multiple object tracking accuracy (MOTA), identity F1 score (IDF1), number of false positives (FP), number of false negatives (FN), and number of ID switches (IDsw). Detailed quantitative evaluation results were uploaded onto the MOT-BD website, and detailed evaluation results can be obtained.

#### B. IMPLEMENTATION DETAILS

We trained and tested our proposed model on an Intel Core i9-9900k CPU cluster with Nvidia GeForce RTX 2080Ti GPUs. Our model was implemented using Pytorch and Pytorch-Geometric. The network configuration of the proposed method and learning setup are as follows.

- **With our GCN architecture, the dimensionality of the initial appearance feature extracted based on ResNet-18 was 512.**
- **The dimensionality of the initial edge feature combined based on the pose and appearance features is 563.**
- **The edge and node features updated in each GCN layer have dimensions of 64.**
The threshold value τ₁ is set to 0.5 for matching the score in data association checking. The number of potential tracker-matches for the new tracker assignment, τ₂, is set to two in MOT17.

During the training process, the entire network is trained with the Adam optimizer with a learning rate of $1 \times 10^{-4}$ for the proposed GCN. The batch size was set to 16, and the weight decay was $1 \times 10^{-4}$.

C. ABLATION STUDY

In this section, we describe a series of ablation studies conducted on the proposed GCN-based MOT to understand the importance of individual components. Because an evaluation can only be conducted through an online evaluation server, and the performance of the test video is limited to four attempts, we measured the ablation performance using the video sequences of the MOT17 training set instead of the test set as standard practice in the MOT literature [27], [28]. For the ablation studies, we divided the MOT17 training set into training sets (MOT17-02, 05, 10, and 11) and test sets (MOT17-04, 09 and 13). To accurately measure the performance of the GCN components, the test set consists of pedestrian street scenes filmed from a static and moving camera.

1) EFFECTIVENESS OF THE NUMBER OF GCN LAYERS

Feature aggregation is the process of improving the initial features into high-order features by updating the node and edge features while going through one or more layers in a GCN. To determine the optimal number of GCN layers, we evaluated the tracking accuracy according to the number of layers. MOTA and IDF1 were measured when the number of layers was changed from one to four in the proposed GCN. The results are shown in Fig. 3.

![Performance comparison of MOTA and IDF1 according to the change in the number of GCN layers.](image)

As can be seen in Fig. 3, when only two GCN layers were used, the performance was better than the other number of layers in terms of MOTA, and the difference was even more severe in terms of IDF1. From the results, we can confirm that a GCN does not require many layers, unlike a CNN. This is because as the GCN layer increases, many graph nodes are considered, resulting in a decrease in the strength of the aggregated features and some loss of feature information. In addition, too many layers slow down the processing, and thus two layers are a good number for a GCN.

2) EFFECTIVENESS OF USING POSE FEATURES

In this study, to construct the strong edge characteristics of a GCN, the local characteristics of the objects were reflected in the graph by using the pose characteristics instead of the geometric characteristics mainly used in a GNN-based MOT approach [26], [27], [32], [47]. To prove the effectiveness of the pose feature, comparison experiments were conducted on three cases that constitute the edge features: 1) geometric, 2) pose, and 3) appearance + pose features. In this experiment, the pose features use the coordinates and reliability scores of 17 key points extracted from a Mask-RNN, and the geometric feature uses the coordinates, width, and height of the object bounding box, as in other studies [26], [27], [32], [47]. Table 1 lists the MOT evaluation results according to the edge feature configuration.

| Edge feature | MOTA (%) | IDF1 (%) | FP (%) | FN (%) | IDsw |
|--------------|----------|----------|--------|--------|------|
| Geometry     | 61.6%    | 39.3%    | 3,118  | 16,402 | 741  |
| Pose         | 62.9%    | 48.0%    | 2,971  | 16,267 | 330  |
| Appearance+  | 64.4%    | 64.2%    | 2,496  | 16,137 | 171  |

As shown in Table 1, the performance of MOT showed that the use of the pose feature was 1.3% higher for MOTA and 8.7% higher for IDF1 than the use of the geometry feature as an edge feature. Moreover, the IDsw of the pose feature was reduced by 411 compared to the geometry feature because the position of the hidden keypoints can be roughly predicted even if the object is occluded. However, when the appearance feature used for the node feature was combined with the pose feature, the MOTA increased by 1.5% and IDF1 was 16.2% higher than the pose feature alone. In terms of IDsw, the third case was reduced by 159 compared to the case using only the pose feature.

From the results, we found that the pose feature can supplement the object information through an occlusion and that the performance of the edge feature can be further improved by supplementing the appearance feature. Therefore, in the subsequent experiment, the pose and appearance features were combined and used for the edge feature.

3) EFFECTIVENESS OF EDGE CLASSIFICATION

We obtained the affinity matrix between the detection node and the tracker node for a data association using the learned edge classification module. In the third ablation study, we conducted comparative experiments on the construction of an optimal pairwise object affinity matrix using the
The proposed GCN features. We compared and evaluated the MOT performance of the proposed edge classification module with the method using the cosine similarity and L2 distance, which are conventional metrics used in the MOT study. The experimental results are presented in Table 2.

TABLE 2. Comparison of MOT performance according to methods for calculating the affinity matrix.

| Method                | MOTA↑ | IDF1↑ | FP↓  | FN↓  | IDsw↓ |
|-----------------------|-------|-------|------|------|-------|
| L2 distance           | 62.9% | 54.2% | 3,038| 16,234| 307   |
| Cosine similarity     | 62.2% | 43.3% | 3,194| 16,302| 456   |
| Edge classification   | 64.4% | 64.2% | 2,496| 16,137| 171   |

As can be seen from the results, the proposed edge classification has a higher accuracy of 1.5% and 2.2% for MOTA and 10% and 20.9% for IDF1, compared to the conventional methods using the L2 distance and cosine similarity, respectively. In particular, the proposed edge classification is 698 for FP and 285 for IDsw lower than that of the cosine similarity. This means that, although the edge classification module uses a simple network, it maintains the object trajectory by reflecting the edge similarity between objects.

4) EFFECTIVENESS OF COMPONENTS FOR EDGE FEATURE UPDATING

For edge feature updating, the existing GNN-based MOT methods only consider the previous edge and node features, whereas our method uses the initial edge features along with the previously used features to prevent the edge feature values from rapidly drifting. Therefore, for this experiment, we investigated the necessity of the initial edge function to update the edge function in a GCN. The results are listed in Table 3. In Table 3, the first row is the proposed method using a combination of node features (Node f.) and previous edge features (Prev. edge f.) for the edge feature updating. The second row is the proposed approach combining node features, previous edge features, and initial edge features (Initial edge f.). As shown in the five metrics used in the experiment, the proposed GCN was able to improve by 0.8% for MOTA and 4% for IDF1 when combining the initial edge features. These results show that our approach prevents edge features from changing rapidly and eventually maintains affinity between nodes.

D. EVALUATION ON MOT BENCHMARK

To verify the MOT performance of the proposed method, we conducted comparative experiments using existing state-of-the-art MOT methods on the MOT16 and MOT17 dataset. The MOT performance on MOT16 dataset was measured using 12 state-of-the-art approaches as follows. The offline graph-based MOTs are (7) CenterTrack [36], (8) BLSTM_MTP_O [41], (9) MFI_TST [47], and (10) ArTIST-T [48]. The online graph-based MOTs are (1) MPNTrack [26], (2) eHAF [48], (3) NOTA [49], (4) FWT [50], and (5) LPC_MOT [40].

The online graph-based MOTs are (9) GCNNMatch [27], (10) GSM-Tracker [47], (11) GNMOT [28], and (12) our proposed method. In addition, the performance was compared with general CNN-based MOT methods without using graphs. The non-graph-based MOTs are (6) BLSTM_MTP_O [41], (7) MFI_TST [47], and (8) ArTIST-T [48].

As shown in Table 4, the proposed method achieved the highest MT and ML, and FN score in all online methods. In terms of MOTA, the proposed method was slightly inferior to MFI_TST [47], but among the online graph-based methods from (9) to (12), the proposed method showed the best performance with 58.8%. In offline graph-based methods from (1) to (5), LPC_MOT [49] showed the best overall performance including MOTA. However, in terms of MOTA, this score is the same as the proposed method, and it can be seen that although our method is online tracking, the performance does not decrease significantly.

The MOT performance on MOT17 dataset was measured using 15 state-of-the-art approaches as follows. The offline graph-based MOTs are (1) MPNTrack [26], (2) JBNOT [51], (3) eHAF [48], (4) NOTA [49], (5) FWT [50], and (6) LPC_MOT [40]. The online graph-based MOTs are (11) GCNNMatch [27], (12) GSM-Tracker [47], (13) GNMOT [28], (14) EDA-GNN [32] and (15) our proposed method. The non-graph-based MOTs are (7) CenterTrack [36], (8) BLSTM_MTP_O [37], (9) MFI_TST [38], and (10) ArTIST-T [39].

As shown in Figure 5, CenterTrack [36] showed higher performance for MOT17 compared to comparative methods. Because CenterTrack [36] used CenterNet to refine public detections, high object detection performance had a significant impact on tracking. Therefore, as pointed out by [37], CenterTrack [36] using a different method to refine public detections is not directly comparable to other methods using only MOT public detections.

In offline graph-based methods from (1) to (6), LPC_MOT showed the best performance in MOTA, IDF1, MT, and FN metrics. However, in terms of MOTA, it is 0.5% lower than the proposed method. In all online methods from (7) to (15), except for CenterTrack [36], MFI_TST [38] shows even good performance in most evaluation metrics. However, this performance is not significantly different from the proposed method, and rather, the proposed method has a processing speed of 2.7 times faster than that of

1In MOT challenge website, proposed method is labeled as 'GCNoMOT'.

| Components                  | MOTA↑ | IDF1↑ | FP↓  | FN↓  | IDsw↓ |
|-----------------------------|-------|-------|------|------|-------|
| Node f.                     | 64.4% | 64.2% | 2,496| 16,137| 171   |
| + Prev. edge f.             | 63.6% | 60.2% | 2,850| 16,184| 212   |
| Initial edge f. (ours)      | 64.4% | 64.2% | 2,496| 16,137| 171   |
TABLE 4. Comparison of our proposed method with state-of-the-art MOT methods on the MOT16 dataset.

| Mode   | Trackers            | MOTA (%) | IDF1 (%) | MT (%) | ML (%) | FP (%) | FN (%) | IDsw (%) |
|--------|---------------------|----------|----------|--------|--------|--------|--------|----------|
| Online |                     |          |          |        |        |        |        |          |
| 1)     | MPNTrack [26]       | 58.6     | 61.7     | 27.3   | 34.0   | 4,949  | 70,252 | 354      |
| 2)     | eHAF [48]           | 47.2     | 52.4     | 18.6   | 42.8   | 12,586 | 83,107 | 542      |
| 3)     | NOTA [49]           | 49.8     | 55.3     | 17.9   | 37.7   | 7,248  | 83,614 | 614      |
| 4)     | FWT [50]            | 47.8     | 44.3     | 19.1   | 38.2   | 9,253  | 85,431 | 852      |
| 5)     | LPC_MOT [40]        | 58.8     | 67.6     | 27.3   | 27.3   | 6,167  | 68,432 | 435      |
| 6)     | BLSTM_MTP_O[37]†    | 48.3     | 53.5     | 17.0   | 38.7   | 9,792  | 83,707 | 735      |
| 7)     | MFI_TST [38]†       | 59.9     | 58.7     | 24.1   | 30.8   | 3,660  | 68,923 | 616      |
| 8)     | ArTIST-T [39]†      | 56.6     | 57.8     | 22.4   | 37.5   | 3,532  | 75,031 | -        |
| 9)     | GCNMMatch [27]      | 57.2     | 55.0     | 22.9   | 34.0   | 3,905  | 73,493 | 559      |
| 10)    | GSM-Tracker [47]    | 57.0     | 58.2     | 22.0   | 34.5   | 4,332  | 73,573 | 475      |
| 11)    | GNMONOT [28]        | 47.7     | 43.2     | 16.1   | 34.3   | 9,518  | 83,875 | 1,907    |
| 12)    | Proposed Method     | 58.8     | 43.1     | 34.9   | 18.4   | 14,127 | 58,528 | 2,402    |

† Non-graph-based MOT methods

TABLE 5. Comparison of our proposed method with state-of-the-art MOT methods on the MOT17 dataset.

| Mode   | Trackers        | MOTA (%) | IDF1 (%) | MT (%) | ML (%) | FP (%) | FN (%) | IDsw (%) |
|--------|-----------------|----------|----------|--------|--------|--------|--------|----------|
| Online |                 |          |          |        |        |        |        |          |
| 1)     | MPNTrack [26]   | 58.8     | 61.7     | 28.8   | 33.5   | 17,413 | 213,594| 1,185    |
| 2)     | JBNNOT [51]     | 52.6     | 50.8     | 19.7   | 35.8   | 31,576 | 232,659| 3,050    |
| 3)     | eHAF [48]       | 51.8     | 54.7     | 23.4   | 37.9   | 33,212 | 236,772| 1,834    |
| 4)     | NOTA [49]       | 51.3     | 54.5     | 17.1   | 35.4   | 20,148 | 252,531| 2,285    |
| 5)     | FWT [50]        | 51.3     | 47.6     | 21.4   | 35.2   | 24,101 | 247,921| 2,648    |
| 6)     | LPC_MOT [40]    | 59.0     | 66.8     | 29.9   | 33.9   | 23,102 | 206,948| 1,122    |
| 7)     | CenterTrack [36]† | 61.5     | 59.6     | 26.4   | 31.9   | 14,076 | 200,672| 2,583    |
| 8)     | BLSTM_MTP_O[37]† | 51.5     | 54.9     | 20.4   | 35.5   | 29,616 | 241,619| 2,566    |
| 9)     | MFI_TST [38]†   | 60.1     | 58.8     | 26.0   | 29.7   | 13,503 | 209,475| 2,065    |
| 10)    | ArTIST-T [39]†  | 56.7     | 57.5     | 22.7   | 37.2   | 12,353 | 230,437| -        |
| 11)    | GCNMMatch [27]  | 57.0     | 56.1     | 23.3   | 34.6   | 12,283 | 228,242| 1,957    |
| 12)    | GSM-Tracker [47]| 56.4     | 57.8     | 22.2   | 34.5   | 14,379 | 230,174| 1,485    |
| 13)    | GNMONOT [28]    | 50.2     | 47.0     | 19.3   | 32.7   | 29,316 | 246,200| 5,273    |
| 14)    | EDA-GNN [32]    | 45.5     | 40.5     | 15.6   | 40.6   | 25,685 | 277,663| 4,091    |
| 15)    | Proposed Method † | 59.5     | 55.3     | 27.9   | 32.3   | 22,310 | 203,167| 3,304    |

† Non-graph-based MOT methods

MFI_TST [38] in a similar system environment. Among the online graph-based methods from (11) to (15), the proposed method is the highest at 59.5% in MOTA, 27.9% in MT, and lowest at 32.3% in ML, 22,310 in FN for the online methods.

Therefore, we can confirm that the proposed method performs well in state-of-the-art offline and online MOT approaches. Although the proposed tracking method has good MOTA, MT, and ML performance, the ID switch caused by the long-term ghost tracker in a camera with severe motion are problems that need to be solved.

E. EVALUATION ON TRACKING SPEED

Although GCN-based MOT can improve matching performance through feature aggregation, it has a disadvantage that tracking speed can be slowed down when the graph structure becomes complex. To evaluate the online MOT tracking speed, we compared the tracking speed between
three GCN-based MOT methods and three CNN-based MOT methods. For objective comparison, we compared the performance of the algorithms used in Table 4 and Table 5 only for experiments using similar CPU and GPU environments.

As shown in Table 6, among the comparison methods, CenterTrack [36] showed the fastest tracking speed at 17Hz for the MOT17. This speed is 13 times faster than GCN-Match [27] and 2.8 times faster than the proposed method. However, as pointed out by [37], since this method uses different methods to refine public detections to reduce tracking time, a direct comparison cannot be made.

Among GCN-based MOT methods, GCNNMatch was tested in a good system environment, but the tracking speed is very low with an average of 0.8Hz due to the complex graph structure. The proposed method showed the fastest speed for MOT16, but has a slightly slower speed of 1.7Hz compared to GSM-Tracktor [47] for MOT17.

From the results, we can confirm that the proposed method achieves a good performance in terms of tracking and speed among state-of-the-art online and GNN-based MOT approaches. However, the proposed tracking method is still slower than the CenterTrack [36], and especially when the crowd object moves in a camera with high motion, the tracking speed decreases. Therefore, like the recent trend of the MOT methods, the proposed method can also improve the tracking performance and speed by using a CenterTrackor-type redefine detector [41].

Figure 4 shows qualitative examples of how well the proposed method maintains the tracking ID for occlusion. As shown in Fig. 4 (a)~(c), it can be seen that for partially hidden or temporarily hidden pedestrians, the tracker ID remains accurate until the end of occlusion (red arrow).

From the results, we can confirm that the proposed method achieves a good performance in terms of tracking and speed among state-of-the-art online and GNN-based MOT approaches. However, the proposed tracking method is still slower than the CenterTrack [36], and especially when the crowd object moves in a camera with high motion, the tracking speed decreases. Therefore, like the recent trend of the MOT methods, the proposed method can also improve the tracking performance and speed by using a CenterTrackor-type redefine detector [41].

Figure 4 shows qualitative examples of how well the proposed method maintains the tracking ID for occlusion. As shown in Fig. 4 (a)~(c), it can be seen that for partially hidden or temporarily hidden pedestrians, the tracker ID remains accurate until the end of occlusion (red arrow).
Despite the complexity of the graph; therefore, in future studies, we plan to improve the GCN such that it is not significantly affected by the number of nodes.

However, as shown in Figure 4 (d), if the camera movement is large and full occlusion occurs for a long time, the tracker is missing or a new tracker ID is assigned (yellow arrow). Therefore, in the future, we need to study additional methods to solve the problem of tracker ID losing or switching even for long-term occlusion without losing the advantages of real-time online tracking.

VI. CONCLUSION

In this paper, we introduced a new MOT method using a GCN to solve the data association problem, which is one of the biggest obstacles to online MOT. In the previous GNN-based MOT, node and edge updates were used as simple MLPs, whereas in the proposed method, a new formula combining the neural network’s output and node similarity was proposed for edge feature updates, and node feature updates were conducted using MLP output and updated edge features. Through experiments on various data of the MOT challenge, it was confirmed that the proposed method shows a superior or similar performance compared to the latest GNN- and DNN-based MOT methods. In particular, to shorten the slow tracking time, which was a problem of the existing GNN-based MOT, we made a near-online MOT possible by simplifying the GCN structure. However, as the number of detections and trackers increases, the tracking speed may decrease because of the complexity of the graph; therefore, in future studies, we plan to improve the GCN such that it is not significantly affected by the number of nodes.

REFERENCES

[1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2999–3007.
[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2980–2988.
[3] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” 2018, arXiv:1804.02767. [Online]. Available: https://arxiv.org/abs/1804.02767
[4] Q. Zhao, T. Sheng, Y. Wang, Z. Tang, Y. Chen, L. Cai, and H. Ling, “M2Det: A single-shot object detector based on multi-level feature pyramid network,” in Proc. AAAI Conf. Artif. Intell., Jul. 2019, pp. 9259–9266.
[5] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, “Robust tracking-by-detection using a detector confidence particle filter,” in Proc. IEEE 12th Int. Conf. Comput. Vis., Sep. 2009, pp. 1515–1522.
[6] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, “Part-based multi-person tracking with partial occlusion handling,” in Proc. IEEE Comput. Vis. Pattern Recognit., Jun. 2012, pp. 1815–1821.
[7] J. Xing, H. Ai, and S. Lao, “Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses,” in Proc. IEEE Comput. Vis. Pattern Recognit., Jun. 2009, pp. 1200–1207.
[8] A. G. A. Perera, C. Srinivas, A. Hoogs, G. Brooksby, and W. Hu, “Multi-object tracking through simultaneous long occlusions and split-merge conditions,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2006, pp. 666–673.
[9] S. J. Kim, J.-Y. Nam, and B. C. Ko, “Online tracker optimization for multi-pedestrian tracking using a moving vehicle camera,” IEEE Access, vol. 6, pp. 48675–48687, 2018.
[10] J. Lee, S. Kim, and B. C. Ko, “Online multiple object tracking using rule distilled Siamese random forest,” IEEE Access, vol. 8, pp. 182828–182841, Oct. 2020.
[11] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T. Kim, “Multiple object tracking: A literature review,” Artif. Intell., vol. 293, pp. 1–18, Apr. 2021.
[12] J. K. Wolf, A. M. Viterbi, and G. S. Dixon, “Finding the best set of K paths through a trellis with application to multitarget tracking,” IEEE Trans. Aerosp. Electron. Syst., vol. 25, no. 2, pp. 287–296, Mar. 1989.
[13] W. Choi and S. Savarese, “A unified framework for multi-target tracking and collective activity recognition,” in Proc. Eur. Conf. Comput. Vis. (ECCV), vol. 7575, May 2012, pp. 215–230.
[14] J. Berclaz, F. Fleuret, E. Türeken, and P. Fua, “Multiple object tracking using K-shortest paths optimization,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 9, pp. 1806–1819, Sep. 2011.
[15] L. Zhang, Y. Li, and R. Nevatia, “Global data association for multi-object tracking using a CRF model,” in Proc. IEEE Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–8.
[16] A. A. Butt and R. T. Collins, “Multi-target tracking by Lagrangian relaxation to min-cost network flow,” in Proc. IEEE Comput. Vis. Pattern Recognit., Jun. 2013, pp. 1846–1853.
[17] P. Lenz, A. Geiger, and R. Urtasun, “FollowMe: Efficient online min-cost flow tracking with bounded memory and computation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 4364–4372.
[18] B. Yang, C. Huang, and R. Nevatia, “Learning affinities and dependencies for multi-target tracking using a CRF model,” in Proc. CVPR, Jun. 2011, pp. 1233–1240.
[19] B. Yang and R. Nevatia, “An online learned CRF model for multi-target tracking,” in Proc. IEEE Comput. Vis. Pattern Recognit., Jun. 2012, pp. 2034–2041.
[20] N. Le, A. Heili, and J. M. Odobez, “Long-term time-sensitive costs for crf-based tracking by detection,” in Proc. Eur. Conf. Comput. Vis. Workshops (ECCVW), vol. 9914, Oct. 2016, pp. 43–51.
[21] L. Leal-Taixe, C. Canton-Ferrer, and K. Schindler, “Learning by tracking: Siamese CNN for robust target association,” in Proc. Proc. EUR. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2016, pp. 418–425.
[22] M. Kim, S. Alletto, and L. Rigazio, “Similarity mapping with enhanced Siamese network for multi-object tracking,” 2016, arXiv:1609.09156. [Online]. Available: https://arxiv.org/abs/1609.09156
[23] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and X. Wu, “Learning fine-grained image similarity with deep ranking,” in Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR), Jun. 2014, pp. 1386–1393.
[24] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” in Proc. Int. Workshop Similarity-Based Pattern Recognit., Oct. 2015, pp. 84–92.
[25] J. Son, M. Baek, M. Cho, and B. Han, “Multi-object tracking with quadruplet convolutional neural networks,” in Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR), Jun. 2017, pp. 3786–3795.

TABLE 6. Comparison results of tracking speed on MOT 16 and 17 sequences.

| Method               | Tracking Speed (Hz) | System Environment       |
|----------------------|---------------------|--------------------------|
| CenterTrack [36]     | -                   | Intel Core i7-8086K CPU / TITAN Xp GPU |
| MFI, TST [38]        | 0.7                 | Intel Core I9-9900 3.6GHz CPU / GTX 2080Ti GPU |
| ArtTIST-T [39]       | 4.5                 | - / GTX 2080Ti GPU       |
| GCNNMatch [27]†      | 0.3                 | Intel 2.6 GHz CPU / TITAN RTX GPUs |
| GSM-Tracktor [47]†   | 7.6                 | Workstation 2.6 GHz CPU / TITAN Xp GPU |
| Proposed Method†     | 7.8                 | Intel Core I9-9900k CPU / RTX 2080Ti GPU |

†: GCN-based MOT methods. Tracking speed was referred to the MOT Challenge site.
[26] G. Braso and L. Leal-Taixe, “Learning a neural solver for multiple object tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 6247–6257.

[27] I. Papakis, A. Sarkar, and A. Karpatsky, “GCNNMatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization,” 2020, arXiv:2010.00067. [Online]. Available: https://arxiv.org/abs/2010.00067

[28] J. Li, X. Gao, and T. Jiang, “Graph networks for multiple object tracking,” in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), Mar. 2020, pp. 719–728.

[29] J. Kim, T. Kim, S. Kim, and C. D. Yoo, “Edge-labeling graph neural network for few-shot learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 11–20.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” IEEE Trans. Neural Netw., vol. 20, no. 1, pp. 61–80, Jan. 2009.

[31] T. Liang, L. Lan, and Z. Luo, “Enhancing the association in multi-object tracking via neighbor graph,” 2020, arXiv:2007.00265. [Online]. Available: https://arxiv.org/abs/2007.00265

[32] X. Jiang, P. Li, Y. Li, and X. Zhen, “Graph neural based end-to-end data association framework for online multiple-object tracking,” 2019, arXiv:1907.05315. [Online]. Available: https://arxiv.org/abs/arXiv:1907.05315

[33] X. Weng, Y. Wang, Y. Man, and K. M. Kitani, “GN3DMOT: Graph neural network for 3D multi-object tracking with 2D-3D multi-feature learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 6498–6507.

[34] J. Gao, T. Zhang, and C. Xu, “Graph Convolutional Tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4649–4659.

[35] D. Guo, Y. Shao, Y. Cui, Z. Wang, L. Zhang, and C. Shen, “Graph attention network for multi-object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 9543–9552.

[36] X. Zhou, V. Koltun, and P. Krähenbühl, “Tracking objects as points,” in Proc. Eur. Conf. Comput. Vis. (ECCV), Oct. 2020, pp. 474–490.

[37] C. Kim, F. Li, M. Alostaibi, and J. Rehg, “Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 9553–9562.

[38] J. Yang, H. Ge, J. Yang, Y. Tong, and S. Su, “Online multi-object tracking using multi-function integration and tracking simulation training,” Int. J. Speech Technol., vol. 146, pp. 1–21, May 2021.

[39] F. Saleh, S. Aliakbarian, H. Rezatofighi, M. Salzmann, and S. Gould, “Probabilistic tracklet scoring and inpainting for multi-object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 14329–14339.

[40] P. Dai, R. Weng, W. Choi, C. Zhang, Z. He, and W. Ding, “Learning a proposal classifier for multi-object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 2443–2452.

[41] J. He, Z. Huang, N. Wang, and Z. Zhang, “Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 5299–5309.

[42] MOT Benchmark. Accessed: Jun. 3, 2021. [Online]. Available: https://motchallenge.net/data/MOT17/

[43] P. P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[44] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[45] F. Yang, W. Choi, and Y. Lin, “Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 2129–2137.

[46] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: The CLEAR MOT metrics,” EURASIP J. Image Video Process., vol. 2008, pp. 1–10, May 2008.

[47] Q. Liu, Q. Chu, B. Liu, and N. Yu, “GSM: Graph similarity model for multi-object tracking,” in Proc. 29th Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 530–536.

[48] 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.

[49] L. Chen, H. Ai, R. Chen, and Z. Zhuang, “Aggregate tracklet appearance features for multi-object tracking,” IEEE Signal Process. Lett., vol. 26, no. 11, pp. 1613–1617, Nov. 2019.

[50] R. Henschel, L. Leal-Taixe, D. Cremers, and B. Rosenhahn, “Fusion of head and full-body detectors for multi-object tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2018, pp. 1509–15090.

[51] R. Henschel, Y. Zou, and B. Rosenhahn, “Multiple people tracking using body and joint detections,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2019, pp. 770–779.

**JIMI LEE** received the B.S. degree in computer engineering from Keimyung University, Daegu, South Korea, in 2019, where she is currently pursuing the master’s degree with the Computer Vision and Pattern Recognition Laboratory. Her current research interests include multiple object tracking, deep learning, and depth estimation from monocular image. She received the Best Paper Award from the IEIE Conference, in 2020.

**MIRA JEONG** received the M.S. and Ph.D degrees in computer engineering from Keimyung University, Daegu, South Korea, in 2009 and 2018, respectively. From 2009 to 2012, she was a Researcher with JVM, and SMEC Cooperation, as a Senior Researcher. She is currently a Post-doctoral Researcher with Keimyung University. Her current research interests include intelligent vehicle applications, deep model compression, and facial expression recognition.

**BYOUNG CHUL KO** (Member, IEEE) received the B.S. degree from Kyonggi University, Suwon, South Korea, in 1998, and the M.S. and Ph.D. degrees in computer science from Yonsei University, Seoul, South Korea, in 2000 and 2004, respectively. From 2004 to 2005, he was a Senior Researcher at Samsung Electronics, Suwon, where he worked on the Ubiquitous Robot Companion Project. He is currently a Professor with the Department of Computer Engineering and the Vice Dean of the College of Engineering, Keimyung University, Daegu, South Korea. He is also the Chief of the KMU AI Fusion Research Center. He was selected as the Best Researcher and a Lecturer at Keimyung University, in 2013, 2014, 2015, and (2018–2021). His current research interests include interpretable machine learning, deep model compression, advance driver assistance systems, and biomedical image processing. He has been a Committee Member of ACRP. He has received excellent paper awards from several conferences. He has been served for various international journals and conferences, including a Special Issue Editor for the Sensors.

* * *