Multi-Vehicle Multi-Camera Tracking with Graph-Based Tracklet Features

Tuan T. Nguyen, Hoang H. Nguyen, Mina Sartipi *Senior Member, IEEE*, Marco Fisichella*

***Abstract*—Multi-target multi-camera tracking (MTMCT) is an important application in intelligent transportation systems (ITS). The conventional works follow the tracking-by-detection scheme and use the information of the object image separately while matching the object from different cameras. As a result, the association information from the object image is lost. To utilize this information, we propose an efficient MTMCT application that builds features in the form of a graph and customizes graph similarity to match the vehicle objects from different cameras. We present algorithms for both the online scenario, where only the past images are used to match a vehicle object, and the offline scenario, where a given vehicle object is tracked with past and future images. For offline scenarios, our method achieves an IDF1-score of 0.8166 on the Cityflow dataset, which contains the actual scenes of the city from multiple street cameras. For online scenarios, our method achieves an IDF1-score of 0.75 with an FPS of 14.**

***Index Terms*—MTMCT, Vehicle Tracking, ITS, Multi-Camera Tracking.**

I. Introduction

Traffic management at the city level is becoming more efficient thanks to recent research on computer vision to predict and analyze large traffic flows. Vehicle tracking application is an essential component in intelligent traffic management. A vehicle tracking application integrates spatial, temporal, and visual information of the vehicle and creates the vehicle trajectory. It can be used to track the trajectory of vehicles in the city and determine the speed and travel time to optimize traffic flow. As shown in Figure 1, Multi-Target Multi-Camera Tracking (MTMCT) aims to extract the vehicle trajectory passing through a large area from cameras at different locations and create a complete trajectory on a global scale. One of the main paradigms in MTMCT is tracking-by-detection where MTMCT is divided into three main components: (1) *object detection*, (2) *multi-object tracking (MOT)*, and (3) *trajectory clustering*.

In *object detection*, an object detector is used to extract objects in small images called a bounding box (bbox) for each video frame. Then, *MOT* tracks the vehicle from the time it enters the camera view until it leaves the camera view. MOT uses a tracker such as DeepSORT [1] to match the detected objects from the frames of the video within a single camera. The goal of the tracker is to solve the matching problem [2], [3], where each detected object in a past frame is associated with one detected object in the current frame using pairwise object affinities. The result of MOT is tracklets containing all bboxes of given objects in that camera. The tracker in MOT use feature learnt from an object Re-identification (ReID) algorithm. Object ReID aims to find the same exact vehicle from a large gallery set extracted from multiple cameras with non-overlapping views. ReID is often based on visual features only, especially without using technologies such as license plate readers. *Trajectory clustering* is the last step of MTMCT, where tracklets from different cameras are matched to track the activity of the object on a global scale. This clustering task is also usually done using the features from an object ReID application. MTMCT can be executed offline and online, where in **offline** information from past and future frames is used to track a given object, and in **online** objects are associated only with past frames.

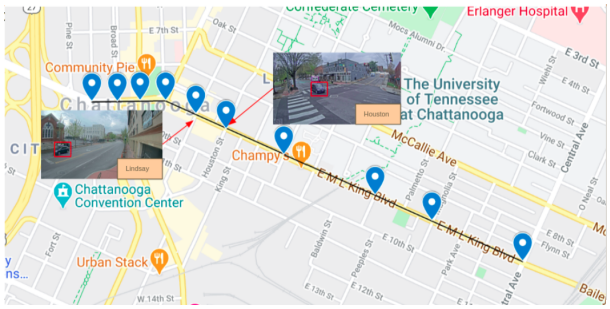

Fig. 1. **Multi-target camera tracking (MTMCT)** - The vehicle is tracked over cameras.

The rest of the paper is structured as follows: motivation is explained in Section II; the technical background is described in Section III; related work is explained in Section IV; Section V presents our proposed architecture; Section VI describes our datasets, evaluation metrics, experimental setups, and results; Section VII presents a discussion of the results; and finally, Section VIII contains our concluding remarks.

II. Motivation

There is a downside to the traditional MTMCT technique. As stated in Section I, the MTMCT framework employs the

Hoang H. Nguyen and M. Fisichella are with L3S Research Center, Leibniz University of Hannover, Appelstr. 9a, 30167 Hannover, Germany; Tuan T. Nguyen and M. Sartipi are with Center for Urban Informatics and Progress (CUIP) at the University of Tennessee at Chattanooga, 615 McCallie Ave, Chattanooga, TN, 37403, United States.

All authors contributed equally in providing critical feedback and helping shape the research, analysis, planning, and development of the evaluation and manuscript. M.F. conceived the original idea and experimental settings and directed the project. T.T.N. and H.H.N. developed the framework and performed the experiments for the evaluation. M.F. and M.S. verified the analytical methods.

*Corresponding author: mfisichella@L3S.de

Manuscript submitted in January 2023.

object ReID to extract tracklet features, despite the fact that the object ReID and MTMCT make distinct assumptions. Specifically, under the premise of object ReID, the given input is a single image of an object, and the target is finding **single** photographs of the same object captured by other cameras within a big gallery of images. In the trajectory clustering step of MTMCT, however, the input is a tracklet containing a **collection of photos** of the same vehicle captured by one camera, and the target is searching tracklets **(many photos)** of the same input vehicle captured by other cameras within other tracklets. Most MTMCT current works merely average the object ReID feature of photos within a tracklet to get the tracklet's feature. However, simple averaging may result in losing crucial associated information of images in the tracklet. Similar to the MOT problem, due to different postures of the vehicle in a trajectory, simply averaging the features may not be the best way to produce the tracklet's feature.

Our paper focuses on two challenges: solving the aforementioned problems of simply averaging image-level features in trajectory clustering tasks and proposing a single and completed framework that can be applied to both online and offline scenarios with good performance and low latency. To address these two challenges, we propose a novel technique for the trajectory clustering step consisting of 3 steps: (a) bbox feature extraction by Siamese Network, (b) graph-based tracklet features construction, and (c) trajectory matching using a graph similarity algorithm. Specifically, for solving associated information loss, we provide a straightforward but effective way for constructing the tracklet's feature in the form of a graph whose nodes and edges are bboxes and the Euclidean distance between the embedding features of the bboxes, respectively. Then the graph similarity algorithms are used to compare the graph-based tracklet features and match the vehicles from different cameras. We use different graph similarities algorithms such as MGMN [4] for offline scenarios and SimGNN [5] for online scenarios. All features of the input graphs, including graph topologies and node or edge attributes, are considered to boost the model's predictability when calculating the similarity between two tracklets using graph structure and neural network model. Particularly, the graph neural network model must be able to capture both global-level graph-graph interactions and local-level node-node interactions, as well as cross-level interactions between each node of one graph and the other whole graph, to optimally match the input tracklets graphs. In addition, the tracklet feature produced by merely averaging the bounding box image-level feature may not be robust or consistent due to the varying vehicle postures throughout a trajectory (Fig 2. On the other hand, graph structure maintains these image-level characteristics distinctly, avoiding the issue mentioned above. The proposed method is described in full in sections V-D and V-E. In addition, two ablation studies for both online and offline scenarios are undertaken in Section VI-D to demonstrate the effectiveness of our technique.

In summary, the application includes five steps as follows: (1) The objects and their features are extracted using a vehicle detection and re-identification algorithm; (2) The single camera tracklets are generated; (3a) The features for each image in these tracklets are extracted using Siamese Network; (3b) Graph-based tracklet features are constructed; (3c) The objects from different cameras are matched using the graph-based tracklet features and the graph similarity algorithms.

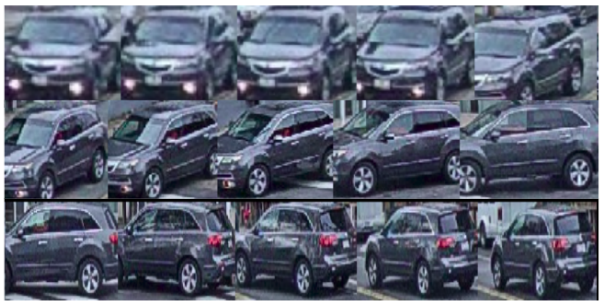

Fig. 2. Tracklet 15 - Cam 42: Different vehicle postures are captured within a tracklet by the same camera..

The contribution of our paper are highlighted below:

- We propose a method for creating graph-based tracklet features that utilize the association information between the bboxes.
- We propose a MTMCT clustering method using graph similarity algorithms such as MGMN [4] and SimGNN [5].
- Our offline MTMCT application outperforms the baselines and achieves an IDF1 of 0.8166 on the CityFlow dataset. IDF1 is the ratio of the correct identified object to the ground truth and average object. Details are explained in Section VI-B.
- Our online MTMCT application obtains an IDF1 of 0.75 with an FPS of 14.

III. Background

Object detection is introduced as the first component of this section, since the first step of the MTMCT application is to detect objects from video images and the result of this process being bbox is used as the base component for the MTMCT application. Our main contribution is proposing a strategy for creating tracklets in a graph structure using a **Siamase Network**; then using **graph similarity learning based on Graph Neural Network (GNN)** to match tracklets from different cameras. Therefore, the remainder of this section will introduce the background of these three topics.

A. Object Detection

Deep learning-based object detection has attracted much attention in recent years. Object detection consists of image classification (i.e., generating a list of object categories in an image) and object localization (i.e., creating a bbox indicating the location of each object category). Current top deep learning models for object detection are based on R-CNN [6] (Region-based CNN) or YOLO (You Only Look Once) [7]. While both models use CNN for object detection, YOLO is more popular than R-CNN due to its good performance and low latency. In 2016, the first model, known as YOLOv1, was

proposed. However, versions of R-CNN such as Fast R-CNN and Faster R-CNN have also been presented to improve speed and detection. In this paper, we use YOLO, described in section V-A.

B. Siamese Network

Many applications of machine learning revolve around checking whether two objects are similar. For example, (1) face recognition checks whether an input facial image is similar to one of the images in the database; (2) question-and-answer websites check whether a new question is similar to one of the stored questions; (3) image search engines retrieve similar images. The strategy is to obtain a vector representation (also known as embedding) for each object. An example of "images" is using the output of an intermediate layer of a pre-trained convolutional neural network (CNN), computing the similarity between the two vectors, and determining whether the two objects are similar [].

1) Architecture: A Siamese Network consists of two identical subnetworks, also called twin networks, connected at their outputs. The twin networks not only have identical architecture but also share weights. They work in parallel and are responsible for creating vector representations for the inputs. For example, we can use ResNet as twin networks if our inputs are images. One can think of Siamese Networks as wrappers for twin networks. They help create better vector representations by measuring similarities between vectors.

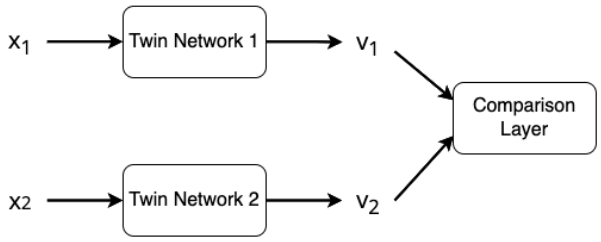

Fig. 3. The Siamese Network.

2) Network Structure: In Figure 3, x_1 and x_2 are the two objects we want to compare, and v_1 and v_2 are the vector representations of x_1 and x_2. The architecture in the comparison layer depends on the loss function and the training data labels. Since we aim to have as much information as possible in the vector representations, the comparison layer usually has a straightforward architecture. Below are some of the common options:

- Compute the cosine similarity between v_1 and v_2; the results are real numbers between -1 and +1 and the loss function is the mean square error;
- Concatenation of v_1 and v_2 and absolute difference per element $|v_1 - v_2|$, followed by fully connected layers and a softmax layer. It is useful for multiclass classification, and the loss function is the cross-entropy;
- Calculate the Euclidean distance (also other distances are accepted) between v_1 and v_2, where the loss function is the contrastive loss or triplet loss.

C. Graph Similarity Learning Based on Graph Neural Network

Similarity learning methods based on Graph Neural Networks (GNNs) aim to learn graph representations through GNNs while performing the similarity learning task in an end-to-end manner [8]. With a pair of input graphs, GNN-based graph similarity learning methods first apply multilayer GNNs with weights to learn the node or graph features in the encoding space of the input graphs. Accordingly, the learning of each graph in a pair could influence each other by the GNNs of the two graphs sharing weights and/or interacting with each other. Then, via an output matrix or vector representation for each graph by the GNN layers, a dot product layer or fully connected layers can be used to calculate or predict the similarity values between two graphs. In the final step, the similarity scores for all graph pairs and their ground truth labels are used in a loss function to train the model with weighted parameters [5], [9], [10].

In this study, we focus on two different aspects of graph similarity learning: (1) the GNN models are able to improve the accuracy compared to other state-of-the-art frameworks, and (2) the predicted results of the GNN models could be responsive to the real-time system. Below are two approaches for each of the above aspects.

(1) Accurate Approach (offline): All properties of the input graphs, including graph topologies and node or edge attributes, are considered to improve the model's predictability. The work adapts Siamese GNNs by incorporating adaptation mechanisms during learning with GNNs, and cross-graph interactions are used in the graph representation learning process. Finally, the BiLSTM [11] model aggregates the cross-level interaction features from the node-graph matching layer before learning the similarity score in the final prediction layers. Therefore, this approach is suitable for the offline scenario where there is enough computation time to derive the complex trained model.

(2) Real-Time Approach (online): Unlike the accurate approach, this approach ignores the node attributes of the input graph to reduce the computation time. The work combines Graph Neural Networks and Convolutional Neural Networks (CNNs) for graph similarity prediction. First, GNNs are used to learn graph representations, and then the learned representations are used in CNNs to predict similarity scores. The model becomes an end-to-end learning system with additional fully linked layers to predict similarity values. Accordingly, the entire process of predicting similarity values can be performed in less than five seconds while still maintaining an acceptable level of accuracy compared to the accurate approach presented above. Thus, this approach is suitable for the online scenario.

IV. Related Work

The goal of MTMCT is to track object activity across multiple cameras.

In addition, recent works use the spatio-temporal constraints and traffic rules as filters to reduce the matching space such as [12], [13]. Other works focus on the architecture such as the graph-based methods GCNNMatch [14] or GNN3DMOT [15].

In this paper, MTMCT base that has the best performance is used as baseline described in section VI-C. Below, we also provide related work of two main components of MTMCT, MOT and subcomponent of trajectory clustering, i.e, graph similarity.

A. Multi-Object Tracking (MOT)

MOT applications aim to track the movement of the object with a single camera. Tracking-by-detection [16] has been the main trend in the MOT application for many years. This paradigm includes three main stages: (1) *image object detection*, (2) *feature extraction*, and (3) *data association*.

In the image object detection stage, the objects in each video frame are detected as bboxes. The most common detection algorithms are YOLO [7] and related algorithms such as YOLO-LITE [17] for non-GPU computers, and Tinier-YOLO [18] for real-time applications.

In the second stage of feature extraction, features of appearance, motion and time are extracted from the detected bboxes and used to calculate the similarity value between pairs of objects. The bboxes that belong to the same object are combined into a tracklet. For feature appearance, much research has been done to learn image representation, including auto-encoder [19], [20], spatial attention [21], feature pyramids [22] and Siamese Network [22]–[24]. For motion features, Kalman filters [1] and LSTM [25] are usually used. A number of methods have been developed for computing similarity values using metrics such as cosine similarity, Euclidean distance, intersection over union, Siamese [26], and bilinear [27].

Object Re-identification: Object Re-identification (ReID) is a clustering task that attempts to match targets in different scenes with different factors such as viewpoint, transparency, and time of day. These targets were detected using the object detection algorithm mentioned in III-A. Object ReID is an important component for common computer vision applications including vehicle tracking, people tracking, and face recognition. In the past, most of the work focused on person ReID. In recent years, thanks to the development of smart cities and intelligent transportation systems (ITS), vehicle ReID has gained more and more attention. A number of works have been done to improve accuracy by using additional information such as license plate and vehicle model in the work of Liu et al. [28], [29]. Other methods focus on architecture such as transformer (TransReId) of He et al. [30], and rigid structure prior of Jiang et al. [31].

Data association is viewed as a clustering task, where bboxes are grouped for each object in a tracklet. Recent work implements this task using Hungarian algorithms [1], dynamic programming [32], reinforcement learning [33], and group-sensitive triplet embedding [34]. In other research [35], the traffic rule has been used to reduce the ID switch error that splits a single tracklet in a camera into many tracklets. In addition, [36] uses license plate data to enhance the tracking result.

B. Multi-target Multi-camera Tracking (MTMCT)

Similar to the MOT problem, MTMCT attempts to track the movement of objects across many cameras. After obtaining the MOT result, the standard technique calls for a subsequent step of trajectory clustering. In this step, a tracklet feature construction approach is chosen to extract the tracklet feature, and a metric is determined to compute the similarity between tracklet features.

For feature construction, the most common method is averaging features of all bboxes within a tracklet [12], [37]. Another variation method is weighted averaging, such as [35] utilizes the structure of temporal attention model [38] to calculate the weights of bounding box features based on spatial and temporal information; then weighted average these bounding box features to obtain the tracklets features. In other research, [39] suggested a method to employ batch hard triplet loss [40] to compute the clustering loss and construct the tracklet's feature, with the goal of bringing the query closer to the positive set in the same camera than in separate cameras. In another work, [41] dismisses obstacle-occluded detection bounding boxes using a background filtering algorithm and averaging the features of the remaining detection bounding boxes. Recently, the authors of [42] multiply the distance between the most similar detection pair between two tracklets and the distance of the mean of all features to generate a combined distance matrix. Concerning the similarity metric used to compare tracklet's features, the two most prevalent metrics are: cosine similarity [12], [37], [39], [42], and Euclidean distance [41], [43], [44].

Unlike existing works, our method uses the graph structure to construct the tracklet feature and a graph similarity algorithm to learn the similarity between tracklets. We assume that the above technique can utilize the associated information of bboxes in tracklets. In addition, when calculating the similarity between two tracklets using a graph-based feature and graph similarity method, all aspects of the input graphs, including graph topologies and node or edge attributes, are taken into account to increase the predictability of the model. Specifically, the graph neural network model must be able to capture both global-level graph-graph interactions and local-level node-node interactions, as well as cross-level interactions between each node of one graph and the other whole graph, to match the input tracklets graphs with the highest performance. Furthermore, due to the various vehicle postures along a trajectory, the tracklet feature obtained by simply averaging the bounding box image-level feature may not be robust or consistent. Graph structure, on the other hand, preserves these image-level features uniquely, avoiding the difficulty mentioned earlier.

C. Graph Neural Network Based Tracking

Utilizing neural networks to handle data with a graph structure was the initial application of GNNs [45]. The core idea is to design a graph with interconnected nodes and edges and to update node/edge properties based on these interconnections. In recent years, various GNNs (e.g., GraphConv [46], GCN [47], GAT [48], GGSNN [49]) have been proposed, each with a distinct feature aggregation rule that has been demonstrated to be effective on a variety of tasks, including object tracking.

For instance, in GNN3DMOT [15], the authors design an unweighted graph in which each node represents an object

feature at a particular frame, and each edge between two nodes at different frames represents the matching between detections. Specifically, they are utilizing appearance and motion data to build the node feature. Furthermore, the distance between the detection centers of the two connected nodes should be less than a predetermined threshold. Then, they employ GNNs to update node features using object relation modeling. Each node can update its feature by aggregating features from other nodes at every tier of the GNNs and define the data association task as a problem of edge classification using GNNs.

Similarly, in the paper [50], the nodes represent the feature of the bounding boxes that integrate appearance and motion information, and the edges indicate the spatial-temporal interaction between nodes. These spatial-temporal relationships serve as the basis for an adjacent matrix. The GNN is fed this adjacent matrix and matrix of features. Then, GNN propagates node features across a graph's structure and discovers the association between nodes. The nodes feature is then projected onto a space with a high dimension. The features inside the same aggregate range can be interpreted as belonging to the same object and are linked sequentially on the timeline to create a complete trajectory.

In a different work [51], the author employs the tracking-by-detection approach and formulates the association task graphically as a Maximum Weighted Bipartite Matching problem. Specifically, the similarity between current tracklets and newly detected objects must be learned at each time frame. These similarity scores are computed utilizing two models, a Siamese Convolutional Neural Network for the appearance feature and an LSTM for the motion feature. Then, a set of fully connected layers is utilized as the learnable measure to combine these two similarity scores. The assignment task was solved using a GNN trained with three loss functions, including a binary cross-entropy loss indicating whether the detection is a match or mismatch, a cross-entropy loss for multi-class classification, and a mean square error (MSE) called birth & death loss that enforces the output vector to approach negative infinity.

Another approach proposed in [52] uses two GNNs, one for learning appearance features and another for learning motion features. Then, similar to [15], the authors formulate data association as an edge classification problem using GNNs, where each node represents an object, and each edge relating to two nodes denotes the similarity between new detection and the current tracklet.

In [53], the authors address the MOT issue using Message Passing Networks (MPNs) [54], [55] in an undirected graph. The nodes are bounding boxes' appearance features extracted by CNN, and the edges are formed so that every pair of nodes from different frames are connected. Edge value is computed by passing the bounding box's relative size, position, and time distance into an MLP. According to the MPNs model, Nodes communicate appearance information with their connected edges, while edges share geometry information with their incident nodes. Additionally, each edge has a binary variable with a value of 1, indicating that two nodes belong to the same trajectory and are consecutive, and have a value of 0, denoting all other cases. Then, they approach the MOT task as a classification problem using edges, where the targeted label is specified above.

The authors in [56] assume that the MOT result is unreliable enough to input so that MTMCT could inherit the false positives and fragments problems of MOT. They then solve the problem by integrating these two models and treat data association as a global MAP problem inspired by the same MAP formulation from [57]. Then the optimal solution is the maximum posterior probability of a set of tracklets with the minimum cost. Specifically, The objective of the data association is to maximize the posterior probability of the new tracklets given the existing tracklet set under the non-overlapping constraints.

In our paper, we propose to construct the tracklet as a graph where nodes are bbox's appearance feature extracted by a Siamese Network and edges are the similarity between nodes, then use a graph similarity algorithm to compute the similarity between tracklets and perform the multi-camera matching task.

D. Graph Similarity

Previous GNN-based work on learning graph similarity can be grouped into two main categories based on how graph similarity or proximity is used in learning, including GNN-CNN mixed models for graph similarity prediction and Siamese GNNs-based graph matching networks.

Bai et al. [58] proposed GSimCNN, a model consisting of three stages for pairwise graph similarity prediction. Multi-layer graph convolutional networks (GCNs) first generate node representations, then the inner products between all possible pairs of node embeddings between two graphs from different GCN layers are computed. Finally, multiple independent CNNs and fully linked layers process the similarity matrices from different layers to predict the final similarity value.

Ktena et al. [59] propose a model that uses the siamese graph convolutional neural network to learn graph similarity. This model considers a pair of graphs as inputs and applies a spectral GCN to generate a graph embedding for each input graph. Similarly, Ma et al. [50] propose a higher-order Siamese GCN model that combines the proximity of higher-order nodes with GCNs to apply higher-order convolutions to each input graph for the graph similarity learning task.

Additionally, some recent graph matching networks [9], [10] have focused on the image matching task as an application in the fields of computer vision, where images are transferred to graph topologies. Remarkably, while the nodes of the graph converted from the input image represent the unary descriptors of extracted feature points of the image, the connections are utilized for encoding the pairwise relationships among different feature points in that image. Thus, the image feature matching problem can be reformulated as a graph matching problem based on the new graph representation.

V. Proposed Architecture

We propose an architecture based on the tracking-by-detection scheme [60], which includes three steps as illustrated in Figure 4. First, in the **(1) vehicle detection** step, vehicles are detected as bboxes by Yolo5 from video images. Then, as mentioned in section IV-A, the bboxes are associated to

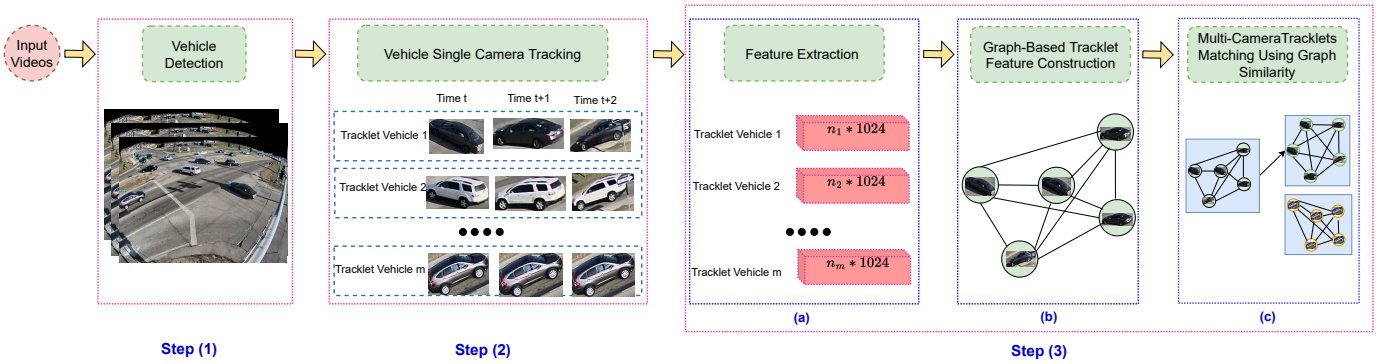

Fig. 4. **Proposed Architecture**. The architecture includes three steps. In **(1)**, the vehicle is detected in the vehicle detection step. In **(2)**, tracklets are generated based on the detection result in the vehicle tracking step using a camera. Then, a feature for each corresponding bbox is learned using the Siamese Network in **(3a)**. Next, graph-based tracklet features are created in the **(3b)** step. Finally, in **(3c)**, the multi-camera tracklets matching step, the similarity between tracklets is calculated and used for multi-camera tracklets matching.

generate corresponding single-camera tracklets in **(2) vehicle single camera tracking** model using FairMOT [61]. In step **(3)** - trajectory clustering, we present it in three sub-steps: **(3a) feature extraction** step, we learn the features using Siamese Network for each corresponding bbox. Then, we use the graph structure to construct the feature for each tracklet in the step **(3b) graph-based tracklet feature construction**. Finally, in the **(3c) multi-camera tracklets matching step**, the similarity between the tracklets is computed using the graph similarity algorithm. Then, these tracklets from different cameras are matched using these similarity scores to generate the tracklets for each vehicle in multi-cameras. Each step is described in detail below.

A. Step 1 - Vehicle Detection

In vehicle detection, the vehicle is detected from the video images as explained in section III-A. We use YOLOv5, which was published in 2020 by Glenn Jocher. There are already more than 240 research papers on this architecture. YOLOv5 is based on the PyTorch framework. It is the latest version of the YOLO object recognition model developed through the continuous work of 58 open source contributors. There are some model configuration files and different versions of the object detector. The present implementation uses the pre-trained model YOLOv5l6, which is the largest model and provides the best performance. The result is in the format of vectors:

$$[t_x, t_y, t_w, t_h, classProb],$$

where t_x, t_y are box center x and y coordinate in the image of video frame, t_w, t_h are the weight and height of the box, $classProb$ are the class probability.

B. Step 2 - Multi-Object Tracking

This task is called Multi-Object Tracking (MOT), which is explained in section IV-A. To track multiple targets within a single view, we follow the tracking-by-detection paradigm to combine frame-level detection results into tracklets. Specifically, we use FairMOT [61], the latest tracking algorithm proposed by Zhang et al., to create tracklets. It incorporates a Kalman filtering algorithm with a cascade matching algorithm to estimate the position of objects from noisy detections. One of its main advantages is the inclusion of deep visual features as association criteria, which are much more informative than simple bounding boxes. As shown in Figure 4, the corresponding vehicle images are extracted from the recognition component results and then used as input to the component to track the vehicle with a camera. Finally, the tracklets containing the triplet vectors of each object are generated:

$$T_{id} = [T_{id,i} : (t_i, b_i, f_i)],$$

where T_{id} is the tracklet with a given id, b_i is the bbox information as presented in section V-A, t_i is the time frame, and f_i is the appearance feature for the corresponding bbox.

C. Step 3a - Feature Extraction

We presented the standard architecture of the Siamese Network in section III-B. In this section, we explain the details of our implemented architecture, which consists of two symmetric CNNs with identical structures and parameters, i.e., the Twin Networks from Figure 3. Each CNN follows the ResNet-50 architecture [62]x with 50 layers (counting only the convolutional layers and the fully connected layers) and contains 3 *residual units* (RUs) outputting 256 feature maps, then 4 RUs with 512 maps, 6 RUs with $1,024$ maps, and finally 3 RUs with $2,048$ maps. The initial parameters of each ResNet-50 are the initial values of the original ResNet-50 trained on $1,000$ classes of the dataset *ImageNet* [63]. The final structure of each twin Siamese network consists of the ResNet-50 architecture augmented with two additional deeper dense layers of d neurons fully connected to the ReLU activation function, each added at the top.

The top layer is used to obtain the representation of the feature vector v_i for the input image x_i with d dimensions. Thus, this workflow generates different representations for each input image, i.e., the feature vectors. The feature vectors are then used to compute the loss used in the backpropagation for the learning process.

Network Training: For each Twin Network, we freeze the weights of all pre-trained layers of the ResNet-50 model so as not to affect the weights that the model has already learned, and add our own 2 dense layers with d neurons. We train the Siamese network by determining the Euclidean distance between the output image representations. We use the standard formulation for *triplet loss* [64]. This loss function considers triplets (x_a , x_p , x_n) as input. Here x_a is an anchor object, x_p is a positive object (i.e., x_a and x_p belong to the same class), and x_n is a negative object (i.e., x_a and x_n belong to different classes). Our goal is to have the vector representation v_a be closer to v_p than it does to v_n. The final formula is:

$$L = max(0,\ m + ||v_a - v_p|| - ||v_a - v_n||). \qquad (1)$$

During the training process, we extracted $100,000$ triplets (x_a , x_p , x_n) that serve as input to the Siamese Network. Each CNN outputs different vectors (v_a , v_p , v_n) of length d from the last layer, which are used as representations for the input images. The Siamese Network is trained simply by inputting a triplet of images and backpropagating the losses through the layers.

D. Step 3b - Graph-Based Tracklet Feature Construction

As mentioned in the III-C section, we build the tracklet features in a graph structure to use the association information in the tracklet and match the tracklets based on computing the graph similarity between them. The nodes are the feature vectors (embeddings) from the Siamese Network and the edges are the Euclidean distance between two nodes. Only the edges larger than a threshold τ remain.

The decision to set the distance threshold τ to 0.5 was determined by an empirical tuning proposed in [65], [66]. According to human judgment, the distance value of 0.5 proved to be the boundary line between the same and different images: Image pairs with a distance of more than 0.5 were mostly perceived as different, whereas those with a distance of less than 0.5 were mostly classified as identical. In other words, at a distance of 0.5 or more, a person can recognize that two images are similar but not identical. We also experimented with different thresholds.

E. Step 3c - Multi-Camera Tracklets Matching Using Graph Similarity

To increase efficiency, a spatial-temporal filter is applied before matching. First, the camera image is divided into 4 crossroad zones. Take camera 43 as an example in Figure 6: 4 colors are used to divide the different zones (white: zone 1; blue: zone 2; green: zone 3; red: zone 4). Since all cameras are located on the main road, as shown in Figure 5, zone 3 and zone 4 are connected to the main road, while zone 1 and zone 2 cross the main road. More precisely, zone 3 is connected to the next camera (camera 44), while zone 4 is connected to the previous camera (camera 42).

In the test scenario, there are six cameras with IDs 41 to 46. They are arranged in a street as shown in Figure 5. Therefore, we need to match tracklets of cameras with consecutive IDs. Before matching, we extract the information about the start/end time $[t_s, t_e]$ and start/end zone $[z_s, z_e]$ for all tracklets T_{id}. Then we use the conflict table in table I to filter all tracklet pairs from two cameras with consecutive ID. Specifically, a tracklet pair is not matched if the three conditions of intersection zone, camera, and time in the conflict table are satisfied. This procedure significantly reduces the original search space.

TABLE I
CONFLICT TABLE FROM [12] WHEN TWO TRACKLETS SASTIFY THREE CONDITIONS OF CROSSROAD ZONE, CAMERA AND TIME, THEY DO NOT MATCH.

Crossroad Zone	Camera	Time	Matching
$z_s^i = 1$ or 2	-	$t_e^j < t_s^i$	False
$z_s^i = 3$	$c^j > c^i$	$t_e^j > t_s^i$	False
$z_s^i = 4$	$c^j < c^i$	$t_e^j > t_s^i$	False
$z_s^i = 1$ or 2	-	$t_s^j > t_e^i$	False
$z_s^i = 3$	$c^j > c^i$	$t_s^j < t_e^i$	False
$z_s^i = 4$	$c^j < c^i$	$t_s^j > t_e^i$	False

To explain the idea of this filtering process with the conflict table, we use an example of matching tracklets from camera 42 and camera 43. As shown in Figure 6, we only need to filter 2 groups of vehicles from camera 42 and camera 43:

- Group 1: vehicles move from camera 42 to camera 43.
- Group 2: vehicles move from camera 43 to camera 42.

In order to filter these 2 groups. The spatial-temporal filtering is applied. As shown in Figure 7, the spatial filtering is applied as follows:

- Group 1: tracklets ending in zone 3 of camera 42 are matched with tracklets starting in zone 4 of camera 43.
- Group 2: tracklets ending in zone 4 of camera 43 are matched with tracklets starting in zone 3 of camera 42.

After the spatial filtering, we apply the temporal filter as follows:

- Group 1: the ending time of tracklets in camera 42 (ending in zone 3) must be earlier than the starting time of tracklets in camera 43 (starting in zone 4).
- Group 2: the ending time of tracklets in camera 43 (ending in zone 4) must be earlier than the starting time of tracklets in camera 42 (starting in zone 3).

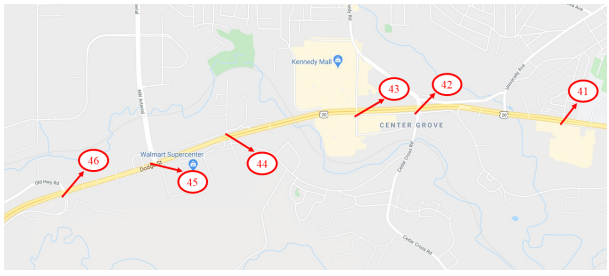

Fig. 5. **Camera Locations -** Testing scenario includes 6 camera from ID 41 to 46.

After reducing the search space by applying the above filters, the similarity scores between graph-based tracklet

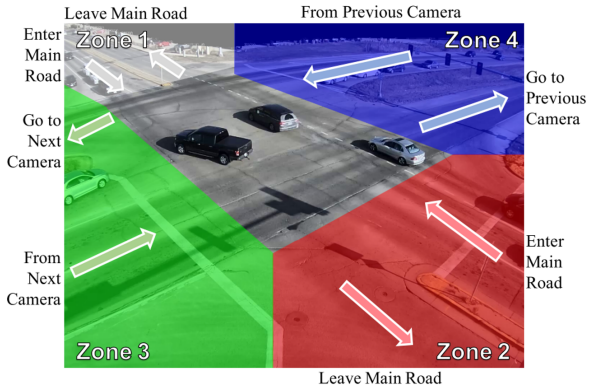

Fig. 6. **Crossroad Zone Visualization of camera 43 from [12].** The video image is divided into 4 zones to track the tracklet. Zone 4 and zone 3 are the path to the previous and next camera. Zone 1 and zone 2 is the way to leave the main road.

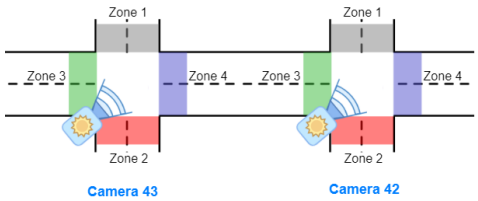

Fig. 7. **The position of camera 43, camera 42 and their zones**. There are 2 groups of vehicles moving between camera 43 and 42. Group 1 moves from zone 4 of camera 43 to zone 3 of camera 42. Group 2 moves from zone 3 of camera 42 to zone 4 of camera 43.

features is computed using one of the folliwng methods depending on being an offline or online scenario.

Accurate Approach (offline): It requires the graph neural network model to be able to capture both global-level graph-graph interactions and local-level node-node interactions while also considering cross-level interactions between each node of one graph with the other whole graph to achieve the highest performance of matching the input tracklets graphs. Therefore, we custom and apply MGMN [4], a multilevel graph matching network suitable for these criteria, to our accurate approach. In particular, we combine a GraphSAGE layer [67] with a GCN layer [47] to learn the cross-level interaction in the node-graph matching network instead of a three-layer GCN as the original MGMN. The modification based on the combination of these two different layers allows the network to aggregate more feature information from the local neighborhood of each node. We also present other variants of MGMN in Section VI for a comprehensive evaluation.

Real-Time Approach (online): Although MGMN results outperform the state-of-the-art approaches for matching multi-camera tracklets (see Section VI), this graph neural network needs hours to compute the graph similarity scores, primarily suitable for offline prediction without requiring an instant response. Thus, to adapt our framework for the real-time scenario, we replace MGMN with SimGNN [5], a simpler GNN model ignoring the node features and only focusing on the graph topologies of each pair of input graphs, to learn the graph similarity. While the accuracy of this approach is still acceptable compared to the above accurate approach, 6% less in terms of IDF1-score, one of the notable advantages of SimGNN is that it achieve an FPS of 14 which is fast enough for the real-time scenario (see Section VI-D). In addition, most current state-of-the-art approaches for the multi-target, multi-camera object tracking problem are offline scenarios, not applicable for the online scenarios. Therefore, to the best of our knowledge, we are the first to create an approach that is able to handle multi-target multi-camera tracking for vehicle objects with an FPS of 14 and high accuracy (reaching an IDF1-score of 0.75).

VI. Experiments and Results

A. Dataset

In the experiment, we use the CityFlow dataset [68], which captures the actual scenes of the city from multiple street cameras. Furthermore, CityFlow covers different types of streets such as intersections, highways, and road extensions. Both the training and validation sets contain 3.25 hours of traffic videos captured from 40 cameras at ten intersections. The test set contains 20 minutes from six cameras at six intersections.

B. Evaluation Metrics

For the evaluation metrics, we use precision, recall, and IDF1 [69] to evaluate the performance of our method. IDF1 is the ratio of correctly identified objects to ground truth and average objects. In detail:

$$IDF1 = \frac{2 * TP}{2 * TP + FP + FN},$$

where TP is true positive, FP is false positive and FN is false nagative matching.

C. Baselines

For offline scenario, our work builds on the foundation of **mcmt** baseline [12]. In general, **mcmt** includes 3 steps. First is the **vehicle detection** step, where Yolo5 is used to detect vehicles as Bboxed. Second is the **Vehicle single camera tracking** (or MOT) step, where FairMOT [61] is used to generate tracklets for each vehicle on each camera. Third is the **multi-camera tracklets matching** step, where the filter is applied to reduce the space for matching the tracklets. Then the similarity between the tracklets is calculated using the CNN features from the MOT step. Finally, these tracklets from different cameras are matched using the similarity score to create the tracklets for each vehicle with multiple cameras. In general, our method utilizes step 1 (vehicle detection) and step 2 (single camera vehicle tracking) of **mcmt** but we apply a novel method for step 3.

For the online scenario, we compare our work with other online tracking algorithms such as **TADAM** [70] and **BLSTM-MTP** [71]. **TADAM** is a model with a synergy between position prediction and embedding association. Specifically, prediction uses attentional modules to focus more on targets

and less on distractors. Such reliable embeddings can then improve identity-awareness perception through memory aggregation. At **BLSTM-MTP**, they focus on solving the problem of simultaneously considering all tracks in memory updating with a spatial overhead by using a novel multi-track pooling module.

D. Experimental Results

Several ablation studies have been performed. The experiment is divided into an online and an offline application.

1) Online Scenario: For the online scenario, we first study the result with two Siamese Network architectures: ResNet and Efficient Net. The result is shown in Table IV. For features of dimension 1024, ResNet achieves a better result, while Efficient Net has a higher FPS. Then, we keep the Siamese Network Backbone as ResNet and increase the dimension of the feature to 2048 and 4096 to determine which is best for the feature. Shown in Table IV, surprisingly, the feature with dimension 2048 obtains the best result of 0.7521 IDF1, while the feature with dimension 4096 obtains an IDF1 value of 0.7357. The FPS of our best setting is from 14.03, which is suitable for a real-time scenario.

The distance threshold τ in the construction of the graph feature is set to 0.5 by an empirical tuning proposed in [65], [66]. However, to confirm this decision, we perform an ablation study with different thresholds. Specifically, we study the final result with thresholds $0.4, 0.5$, and 0.6. These experiments use ResNet with 2048-dimension of Siamese Network backbone. From the result of Table II, a threshold of 0.5 yields the best result of 0.75 IDF1, which is consistent with previous work [65], [66].

TABLE II
ONLINE METHOD - RESULT ON DIFFERENT GRAPH CONSTRUCTION THRESHOLDS

Threshold τ	IDF1
0.4	0.7325
0.5	**0.7500**
0.6	0.7439

To demonstrate the effectiveness of our proposed method, we compare our approach (graph feature for feature creation and graph similarity for similarity computation) to existing methods. We choose some code-available techniques as mentioned in Section IV-B includes: **mcmt** [12] which calculate the mean of features of *all* bboxes within a tracklet, **UWIPL** [35] which use an attention model to calculate the weights of bounding boxes, then weighted sum these bounding box features to calculate the tracklet's features, **BUPT** [39] which utilize batch hard triplet loss as clustering loss to construct tracklet's features, and **IOSB** [41] which multiply the distance of the most similar detection pair between two tracklets with the distance of the mean of all features to yield a combined distance matrix. In the methods, cosine similarity or Euclidean distance is typically used to compute the similarity between two tracklets regarding similarity computation methods. The result presented in Table III indicates that our method outperforms all other settings by 3.04% of IDF1.

TABLE III
ONLINE METHOD - RESULT ON DIFFERENT FEATURE CONSTRUCTING AND SIMILARITY METRIC

Team Name	Feature Constructing	Similarity Metric	IDF1
UWIPL [35]	Weighted_Average	Euclidean distance	0.7135
BUPT [39]	Batch_Triple_Loss	Cosine Similarity	0.6962
IOSB [42]	All_Average*Best_Pair	Cosine Similarity	0.7204
mcmt [12]	All_Average	Cosine Similarity	0.7217
Ours	Graph-base	Graph Similarity	**0.7521**

Baseline Comparison: We also check the performance of other online tracking algorithms such as TADAM [70] and BLSTM-MTP [71]. Since these two algorithms were developed for pedestrian tracking, the results are not competitive. TADAM achieves an IDF1 of 0.53, while BLSTM-MTP achieves an IDF1 of 0.61. More importantly, the FPS of TADAM and BLSTM-MTP are 13.30 and 8.55, respectively. In summary, the results in Table IV show that our FPS is higher, while the IDF1 outperforms the existing works.

TABLE IV
ONLINE METHOD - COMPARISON TO OTHER ONLINE BASELINES.

	Method	FPS	IDF1
Baselines	TADAM [70]	13.30	0.5347
	BLSTM-MTP [71]	8.55	0.6167
Ours	SimGNN - Efficient Net (1024-dimension Feature)	**16.70**	0.6530
	SimGNN - ResNet (1024-dimension Feature)	15.63	0.7316
	SimGNN - ResNet (2048-dimension Feature)	14.03	**0.7521**
	SimGNN - ResNet (4096-dimension Feature)	12.16	0.7357

2) Offline Scenario: For the offline algorithm, we study various optimizations for the Siamese Network. In particular, we keep ResNet as the backbone with 256-dimension features and examine the result of the different optimizations: RMSprop, Adam, and AdamW. From the result of Table V, we find that Adam optimization gives the best result of 0.7832 IDF1. Therefore, we keep Adam as the default optimization for all subsequent experiments.

TABLE V
OFFLINE METHOD - RESULT ON DIFFERENT OPTIMIZATION

Optimization	IDF1
RMSprop	0.7805
Adam	**0.7832**
AdmaW	0.7626

Next, the ablation study is performed on different MGMN layers. These experiments use the Resnet backbone with 256-dimension feature and Adam optimization as the default setting. The results of the four set layers (GGNN, GIN, GraphSAGE, GCN + GraphSAGE) are shown in Table VI. GCN + GraphSAGE layers achieve the best result of 0.7866 IDF1 score. Therefore, GCN + GraphSAGE is kept as the default setting for the following experiments.

Next, we examine the effects of the different dimensions of the Siamese Network on the final result. After keeping all the

TABLE VI
OFFLINE METHOD - RESULT ON DIFFERENT MGMN LAYER

Method	IDF1
MGMN (original) - GCN layers	0.7725
MGMN - GGNN layers	0.7532
MGMN - GIN layers	0.7487
MGMN - GraphSAG layers	0.7754
MGMN - GCN + GraphSAG layers	**0.7866**

default parameters, we increase the dimensions of the Siamese Network feature from 256 to 2048. The results in Table VII show that we achieve an increasingly better IDF1 score as the feature dimension is increased. Our method achieves an IDF1 score of 0.8166 with features of dimension 2048. Therefore, we keep the 2048 dimension of the features as the default setting. In this experiment, we also calculated the precision and recall to confirm the performance of the final setting.

TABLE VII
OFFLINE METHOD - FEATURE DIMENSION ABLATION STUDY

Method	Precision	Recall	IDF1
MGMN (256-dimension Feature)	0.8549	0.7437	0.7866
MGMN (512-dimension Feature)	0.8835	0.7728	0.8076
MGMN (1024-dimension Feature)	**0.9022**	0.7514	0.8125
MGMN (2048-dimension Feature))	0.8902	**0.7964**	**0.8166**

An ablation study at different thresholds is also performed using the offline algorithm. Similar to the online ablation study, we examine the final result with thresholds τ of $0.4, 0.5$, and 0.6. In these experiments, the standard parameters mentioned above are used. From the result of Table VIII, the threshold of 0.5 yields the best result of 0.8166 IDF1, which is consistent with previous work [65], [66], [72].

TABLE VIII
OFFLINE METHOD - RESULT ON DIFFERENT GRAPH CONSTRUCTION THRESHOLD

Threshold τ	IDF1
0.4	0.7865
0.5	**0.8166**
0.6	0.7697

Similar to the online scenario in Table III, we also compare our proposed method (graph feature for feature creation and graph similarity for similarity computation) with existing methods as **mcmt** [12], **UWIPL** [35], **BUPT** [39], and **IOSB** [41] for the offline scenario. The result presented in Table IX shows that our method outperforms all other settings by 1.51% of IDF1. This proves that applying graph similarities based on GNNs can potentially improve the overall performance of MTMCT systems in both online and offline scenarios compared to the existing approaches.

Baseline comparison: While the baseline mcmt achieves an IDF1 of 0.8095 in the CityFlow dataset, our method outperforms it with an IDF1 of 0.8166 using the same data. In addition, we compare the performance of our algorithm with other leaderboards of 2021 AI City Challenge Track 3 in Table X. Our method outperforms all teams in the leaderboards. The qualitative result is visualized in Figure 8.

TABLE IX
OFFLINE METHOD - RESULT ON DIFFERENT FEATURE CONSTRUCTING AND SIMILARITY COMPUTATION METHOD

Team Name	Feature Constructing	Similarity Metric	IDF1
UWIPL [35]	Weighted_Average	Euclidean distance	0.7916
BUPT [39]	Batch_Triple_Loss	Cosine Similarity	0.7863
IOSB [42]	All_Average*Best_Pair	Cosine Similarity	0.8015
mcmt [12]	All_Average	Cosine Similarity	0.8003
Ours	Graph-base	Graph Similarity	**0.8166**

TABLE X
OFFLINE METHOD - COMPARISON TO 2021 AI CITY CHALLENGE LEADERBOARD

Team	IDF1
mcmt	0.8095
fivefive	0.7787
CyberHu	0.7651
IOSB	0.691
DAMO	0.6238
Janus Wars	0.5763
aiforward	0.5654
BUPT-MCPRL2	0.5534
oOIAMAIOo	0.5458
Dukbaegi	0.5452
Ours	**0.8166**

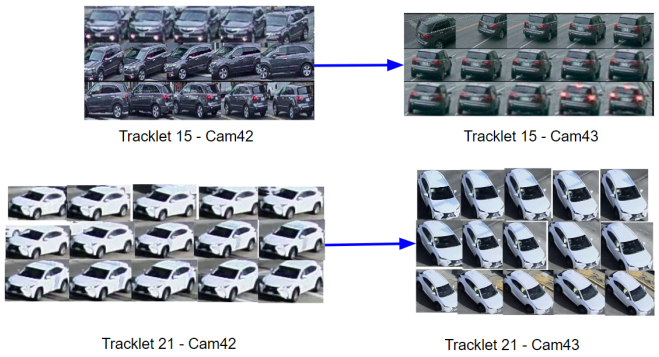

Fig. 8. Qualitative results of our proposed model: Each row is a true positive tracklet matching from camera 42 to camera 43. Each tracklet includes 15 bboxes.

VII. DISCUSSION AND FUTURE WORK

MTMCT is an essential task for an intelligent traffic management system. It can be used to track the trajectory of vehicles in the city and determine the speed and travel time to optimize traffic flow. Our proposed method uses the association information between the bboxes of a tracklet by creating a graph-based tracklet feature. Then, the graph similarity learning algorithms are used to obtain the similarity value and perform the matching. We currently focus on vehicle tracking and achieving a better performance in both online and offline scenarios. In addition, the system is also suitable for pedestrian tracking because it has the advantage of using body moment matching information. Specifically, a person's body

parts can be used as nodes to create a graph-based structure for that person. Then we can apply a similar technique to match the person from the previous to the current frame. Thus, one potential direction for future work is to expand our framework to apply to pedestrian tracking.

VIII. Conclusion

In this paper we propose a novel MCTM algorithm in MTMCT system. Our MCTM algorithm is based on building the tracklet feature as graph structure, then using graph similarity scores to match the tracklets from different cameras. We introduce the strategy for both online and offline scenarios.

In the offline scenario, information from both past and future frames is used to track a particular object. Since the offline application needs to focus on improving the matching result, the accurate approach of the graph similarity learning algorithm like MGMN is customized and applied. Our result in the offline scenario outperforms the baseline of **mcmt** and other previous works on the CityFlow dataset with an IDF1-score of 0.8166.

In the online scenario, where objects are only associated with past frames, the FPS should be fast enough to be used in a real-time application. Therefore, we customize SimGNN as a learning graph similarity for the real-time approach. Our real-time approach outperforms the baseline with an IDF1-score of 0.7521 and an FPS of 14.

Acknowledgement

This material is based upon work partially supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Award Number DE-EE0009208. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This work was also supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 833635 (project ROXANNE: Real-time network, text, and speaker analytics for combating organized crime, 2019-2022).

References

[1] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[2] M. Fisichella, "Unified approach to retrospective event detection for event- based epidemic intelligence," *Int. J. Digit. Libr.*, vol. 22, no. 4, pp. 339–364, 2021. [Online]. Available: https://doi.org/10.1007/s00799-021-00308-9

[3] A. Ceroni, U. Gadiraju, and M. Fisichella, "Justevents: A crowdsourced corpus for event validation with strict temporal constraints," in *Advances in Information Retrieval - 39th European Conference on IR Research, ECIR 2017, Aberdeen, UK, April 8-13, 2017, Proceedings*, ser. Lecture Notes in Computer Science, J. M. Jose, C. Hauff, I. S. Altingövde, D. Song, D. Albakour, S. N. K. Watt, and J. Tait, Eds., vol. 10193, 2017, pp. 484–492. [Online]. Available: https://doi.org/10.1007/978-3-319-56608-5_38

[4] X. Ling, L. Wu, S. Wang, T. Ma, F. Xu, A. X. Liu, C. Wu, and S. Ji, "Multilevel graph matching networks for deep graph similarity learning," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9516695

[5] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 384–392.

[6] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[8] A. Cuzzocrea, J. L. D. Coi, M. Fisichella, and D. Skoutas, "Graph-based matching of composite OWL-S services," in *Database Systems for Adanced Applications - 16th International Conference, DASFAA 2011, International Workshops: GDB, SIM3, FlashDB, SNSMW, DaMEN, DQIS, Hong Kong, China, April 22-25, 2011. Proceedings*, ser. Lecture Notes in Computer Science, J. Xu, G. Yu, S. Zhou, and R. Unland, Eds., vol. 6637. Springer, 2011, pp. 28–39. [Online]. Available: https://doi.org/10.1007/978-3-642-20244-5_4

[9] B. Jiang, P. Sun, J. Tang, and B. Luo, "Glmnet: Graph learning-matching networks for feature matching," *arXiv preprint arXiv:1911.07681*, 2019.

[10] R. Wang, J. Yan, and X. Yang, "Learning combinatorial embedding networks for deep graph matching," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3056–3065.

[11] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[12] C. Liu, Y. Zhang, H. Luo, J. Tang, W. Chen, X. Xu, F. Wang, H. Li, and Y.-D. Shen, "City-scale multi-camera vehicle tracking guided by crossroad zones," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4129–4137.

[13] Y.-J. Li, X. Weng, Y. Xu, and K. M. Kitani, "Visio-temporal attention for multi-camera multi-target association," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9834–9844.

[14] I. Papakis, A. Sarkar, and A. Karpatne, "Gcnnmatch: Graph convolutional neural networks for multi-object tracking via sinkhorn normalization," *arXiv preprint arXiv:2010.00067*, 2020.

[15] X. Weng, Y. Wang, Y. Man, and K. M. Kitani, "Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6499–6508.

[16] T. Gao, H. Pan, Z. Wang, and H. Gao, "A crf-based framework for tracklet inactivation in online multi-object tracking," *IEEE Transactions on Multimedia*, vol. 24, pp. 995–1007, 2022.

[17] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 2503–2510.

[18] W. Fang, L. Wang, and P. Ren, "Tinier-yolo: A real-time object detection method for constrained environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2019.

[19] H. Feng, X. Li, P. Liu, and N. Zhou, "Using stacked auto-encoder to get feature with continuity and distinguishability in multi-object tracking," in *International Conference on Image and Graphics*. Springer, 2017, pp. 351–361.

[20] K. Ho, J. Keuper, and M. Keuper, "Unsupervised multiple person tracking using autoencoder-based lifted multicuts," *arXiv preprint arXiv:2002.01192*, 2020.

[21] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 366–382.

[22] S. Lee and E. Kim, "Multiple object tracking via feature pyramid siamese networks," *IEEE access*, vol. 7, pp. 8181–8194, 2018.

[23] M. Kim, S. Alletto, and L. Rigazio, "Similarity mapping with enhanced siamese network for multi-object tracking," *arXiv preprint arXiv:1609.09156*, 2016.

[24] B. Lavi, M. F. Serj, and I. Ullah, "Survey on deep learning techniques for person re-identification task," *arXiv preprint arXiv:1807.05284*, 2018.

[25] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Thirty-First AAAI conference on artificial intelligence*, 2017.

[26] Y. Liang and Y. Zhou, "Lstm multiple object tracker combining multiple cues," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 2351–2355.

[27] C. Kim, F. Li, and J. M. Rehg, "Multi-object tracking with neural gating using bilinear lstm," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 200–215.

[28] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2167–2175.

[29] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 645–658, 2017.

[30] S. He, H. Luo, P. Wang, F. Wang, H. Li, and W. Jiang, "Transreid: Transformer-based object re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 013–15 022.

[31] M. Jiang, X. Zhang, Y. Yu, Z. Bai, Z. Zheng, Z. Wang, J. Wang, X. Tan, H. Sun, E. Ding *et al.*, "Robust vehicle re-identification via rigid structure prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4026–4033.

[32] M. Ullah and F. Alaya Cheikh, "A directed sparse graphical model for multi-target tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1816–1823.

[33] L. Ren, J. Lu, Z. Wang, Q. Tian, and J. Zhou, "Collaborative deep reinforcement learning for multi-object tracking," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 586–602.

[34] Y. Bai, Y. Lou, F. Gao, S. Wang, Y. Wu, and L.-Y. Duan, "Group-sensitive triplet embedding for vehicle reidentification," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2385–2399, 2018.

[35] H.-M. Hsu, T.-W. Huang, G. Wang, J. Cai, Z. Lei, and J.-N. Hwang, "Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models." in *CVPR Workshops*, 2019, pp. 416–424.

[36] Q. Huang, Z. Cai, and T. Lan, "A new approach for character recognition of multi-style vehicle license plates," *IEEE Transactions on Multimedia*, vol. 23, pp. 3768–3777, 2021.

[37] H. Yao, Z. Duan, Z. Xie, J. Chen, X. Wu, D. Xu, and Y. Gao, "City-scale multi-camera vehicle tracking based on space-time-appearance features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3310–3318.

[38] J. Gao and R. Nevatia, "Revisiting temporal modeling for video-based person reid," *arXiv preprint arXiv:1805.02104*, 2018.

[39] Z. He, Y. Lei, S. Bai, and W. Wu, "Multi-camera vehicle tracking with powerful visual features and spatial-temporal cue." in *CVPR Workshops*, 2019, pp. 203–212.

[40] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification. arxiv 2017," *arXiv preprint arXiv:1703.07737*, vol. 4, 2017.

[41] A. Specker, D. Stadler, L. Florin, and J. Beyerer, "An occlusion-aware multi-target multi-camera tracking system," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4173–4182.

[42] A. Specker, L. Florin, M. Cormier, and J. Beyerer, "Improving multi-target multi-camera tracking by track refinement and completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3199–3209.

[43] E. Luna, J. C. SanMiguel, J. M. Martínez, and M. Escudero-Viñolo, "Online clustering-based multi-camera vehicle tracking in scenarios with overlapping fovs," *Multimedia Tools and Applications*, vol. 81, no. 5, pp. 7063–7083, 2022.

[44] P. Li, J. Zhang, Z. Zhu, Y. Li, L. Jiang, and G. Huang, "State-aware re-identification feature for multi-target multi-camera tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[45] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE international joint conference on neural networks*, vol. 2, no. 2005, 2005, pp. 729–734.

[46] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[49] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[50] C. Ma, Y. Li, F. Yang, Z. Zhang, Y. Zhuang, H. Jia, and X. Xie, "Deep association: End-to-end graph-based learning for multiple object tracking with conv-graph neural network," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 2019, pp. 253–261.

[51] X. Jiang, P. Li, Y. Li, and X. Zhen, "Graph neural based end-to-end data association framework for online multiple-object tracking," *arXiv preprint arXiv:1907.05315*, 2019.

[52] J. Li, X. Gao, and T. Jiang, "Graph networks for multiple object tracking," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2020, pp. 719–728.

[53] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6247–6257.

[54] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[55] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2688–2697.

[56] W. Chen, L. Cao, X. Chen, and K. Huang, "An equalized global graph model-based approach for multicamera object tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 11, pp. 2367–2381, 2016.

[57] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[58] Y. Bai, H. Ding, Y. Sun, and W. Wang, "Convolutional set matching for graph similarity," *arXiv preprint arXiv:1810.10866*, 2018.

[59] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert, "Metric learning with spectral graph convolutions on brain connectivity networks," *NeuroImage*, vol. 169, pp. 431–442, 2018.

[60] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[61] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision*, vol. 129, no. 11, pp. 3069–3087, 2021.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[63] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[64] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 815–823. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298682

[65] M. Fisichella, A. Ceroni, F. Deng, and W. Nejdl, "Predicting pair similarities for near-duplicate detection in high dimensional spaces," in *Database and Expert Systems Applications - 25th International Conference, DEXA 2014, Munich, Germany, September 1-4, 2014. Proceedings, Part II*, ser. Lecture Notes in Computer Science, H. Decker, L. Lhotská, S. Link, M. Spies, and R. R. Wagner, Eds., vol. 8645. Springer, 2014, pp. 59–73. [Online]. Available: https://doi.org/10.1007/978-3-319-10085-2_5

[66] M. Fisichella, "Siamese coding network and pair similarity prediction for near-duplicate image detection," *International Journal of Multimedia Information Retrieval*, 2022. [Online]. Available: https://doi.org/10.1007/s13735-022-00233-w

[67] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[68] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8797–8806.

[69] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European conference on computer vision*. Springer, 2016, pp. 17–35.

[70] S. Guo, J. Wang, X. Wang, and D. Tao, "Online multiple object tracking with cross-task synergy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8136–8145.

[71] C. Kim, L. Fuxin, M. Alotaibi, and J. M. Rehg, "Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9553–9562.

[72] M. Fisichella, F. Deng, and W. Nejdl, "Efficient incremental near duplicate detection based on locality sensitive hashing," in *Database and Expert Systems Applications, 21st International Conference, DEXA 2010, Bilbao, Spain, August 30 - September 3, 2010, Proceedings, Part I*, ser. Lecture Notes in Computer Science, P. G. Bringas, A. Hameurlain, and G. Quirchmayr, Eds., vol. 6261. Springer, 2010, pp. 152–166. [Online]. Available: https://doi.org/10.1007/978-3-642-15364-8_11

Tuan T. Nguyen held a master's degree in applied mathematics from Ho Chi Minh City University of Science (HCMUS), Vietnam, in 2019. He joined the Center for Urban Informatics and Progress (CUIP) at the University of Tennessee at Chattanooga as a researcher and doctoral student in January 2021. He has more than 7 years of experience in applied mathematics and computer science in industry and academia. His current research interests include Computer Vision, Vehicle Tracking, ML.

Hoang H. Nguyen received his master's degree in computer science from Ho Chi Minh City University of Technology (HCMUT), Vietnam, in 2016. Since February 2020, he has joined the L3S Research Center of Leibniz University of Hannover as a researcher and then a PhD candidate. He has 10+ years of experience in Software Engineering in the industry and academia. His current research interests include Graph Mining, Network Analysis, Machine Learning, Program Analysis, and Cyber Security in Blockchain Technology.

Mina Sartipi is the Guerry Professor of the Computer Science and Engineering Department at the University of Tennessee at Chattanooga, where she is also the founding director of the Center for Urban Informatics and Progress (CUIP). Her research focuses on data-driven approaches to tackle real-world challenges in smart city applications such as transportationAt CUIP, she coordinates cross-disciplinary research and strategic visions for urbanism and smart cities advancement with a focus on people and quality of life. She has been a senior member of IEEE since 2016.

Marco Fisichella has 14+ years of experience in Artificial Intelligence both in the industry and in academia. From April 2021, he led at L3S Research Center of Leibniz University of Hannover the research group with a focus on artificial intelligence and intelligent systems. From 2015 till March 2021, he was Head of the Data at Otto Group where he was directing the department in charge of implementing algorithms against online frauds. Marco led various AI and data science projects for corporate clients (e.g. Otto, Vodafone, DriveNow, Bonprix, etc.).