Multiview Vehicle Tracking by Graph Matching Model

Minye Wu, Guli Zhang, Ning Bi, Ling Xie, Yuanquan Hu
ShanghaiTech University
{wumy, zhanggl, bining, xieling, huyq}@shanghaitech.edu.cn

Zhiru Shi
Yoke Intelligence
zhiru.shi@yo-ke.com

Abstract

Using multiple visual cameras to sensing traffic, especially tracking of vehicles, is a challenging task because of the large number of vehicle models, non-overlapping views, occlusion, view change and time-consuming algorithms. All of them remain obstacles in real world deployment. In this work, we propose a novel and flexible vehicle tracking framework, which formulates matching problem as a graph matching problem and solve it from the bottom up. In our framework, many restrictions can be added into the graph uniformly and simply. Moreover, we introduced an iterative Graph Matching Solver algorithm which can divide and reduce the graph matching problem's scale efficiently. Additionally, We also take the advantage of geographic information and make a combination with deep ReID features, motion and temporal information. The result shows that our algorithm achieves a 9th place at the AI City Challenge 2019.

1. Introduction

Nowadays, cross-view vehicle tracking not only contribute to Intelligent Transportation System (ITS) problems, but also facilitates multiple Traffic Flow Analysis tasks especially for safety purpose. As more and more cameras are deployed, these accumulated data gives great potential to data-driven methods to understand Traffic Flow Analysis. While the real scenario is complicated, cameras may suffer from poor resolutions and varying lighting conditions and vehicles are occluded occasionally.

The multiview vehicle tracking (MVT) is a typical a bottom up multiple-stage task, the performance of overall pipeline depends on the collaborative work of different parts, namely **Detection, Tracking and Matching**. There are detectors like [4, 11, 9] give fare reasonable object detection results. And quite a few trackers like [19, 1] perform stable object tracking. For matching, which usually finding the nearest neighbor with respect to specific distance metric and combining different features. We get plenty of choices on feature extractors[10, 5] while the matching strategy remains as the bottle neck.

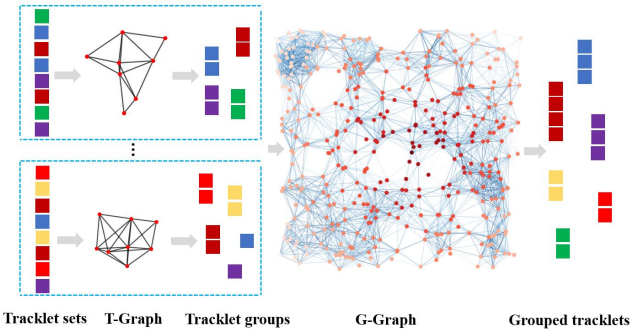

Figure 1. Matching pipeline. First, tracklets are generated from videos. Each small rectangle in the figure represents a tracklet instance. Each color represents a specific vehicle. T-Graphs are built from Tracklet sets of different intersections. Applying *graph matching solver* (GMS) on T-Graphs can obtain tracklet groups. Tracklet groups can build a G-Graph. Then the final result can be calculated by GMS.

In this paper, we propose a complete and flexible framework to settle MVT problem. In this framework, detector, tracker and feature extractors are collaborating with graph based match solver. We formulate the matching problem as graph clique finding problem. Hence, with offering features, we apply a bottom up process, matching within local nearby camera views then extending to global matching to rediscover the instances traveling between crossroads, and propose an iterative matching solver which solves the problem in an approximated way with lower running time. We test our approaches on the AI city 2019 dataset [14].

2. Related Work

2.1. Multi-object Tracking

Multi-object tracking has been extensively studied in recent years.Thanks to the considerable progress in object detection [4, 11], the tracking-by-detection pipeline [2] attracts wide-spreaded attentions and acquires impressive results. [6] formulates instances as connected component model to solve NP-hard multi-dimensional assignment

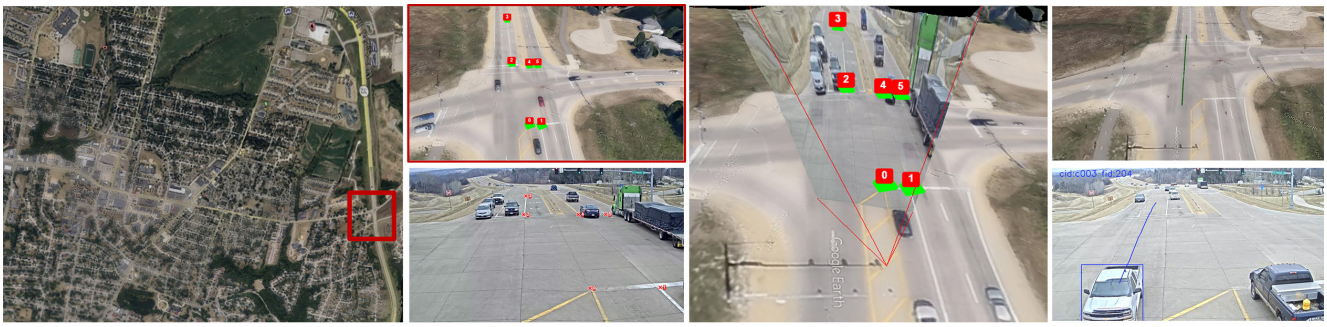

Figure 2. (a) The overview of our reconstructed city model. Red rectangle marks the position of intersection demonstrated in this figure. (b) Top is the closeup view of red rectangle in (a), bottom is a frame capture by camera 'c003'. The red-labeled 3D points in top image and red-labeled 2D pixels in bottom image are our selected 2D-3D pairs to align camera to the model. (c) The image re-projection of the camera view in city model. (d) Bottom shows the trajectory of the gray car and the top shows the corresponding 3D trajectory.

(MDA) problem. [17] utilize motion contexts from multiple objects to maintain stable trajectory sets. While extending the problem under sets of cameras, tracking multiple objects in cross-views requires re-identification of instances among views. In the meanwhile, multi-view structure offers more constrains which benefits exploring data association.

2.2. Multi-view Object Tracking

In multi-object tracking, data association has two constraints: the hard constraint that trajectories must be disjoint, and the soft constraint that the appearance and motion features of one target are stable in a small temporal span. Multi-view object tracking is usually addressed as a data association problem across cameras. [7] shows the power of multi-camera system by adding planar homography constrain on calibrated cameras for pedestrains tracking. Studies in [18, 16] take advantage of variant features from same instance under different perceptions to construct a stronger clarification of different objects. All these methods have certain strong assumptions and thus are restricted to certain specific scenarios. In this work, we are interested in a more universal multiview tracking framework that utilize both geometrical and deep feature constrains to perform a cross-view, precise, and robust vehicle tracking.

3. Methodology

Our approach can be summarized into three main steps. The first step is bounding box detection. We use the detection results from Mask RCNN [4]. and eliminate bounding boxes which are too small (smaller than 40x40 pixels) and union ones are overlapping with each other. Next step is vehicle tracklet generation. We track detected vehicles by using object tracker to compose tracklets. Thus detected bounding boxes which belong to a vehicle are strung up together. The final step is graph based vehicle matching. In this step, we take a tracklet as a unit, and match tracklets by using *Graph Matching Solver* iteratively.

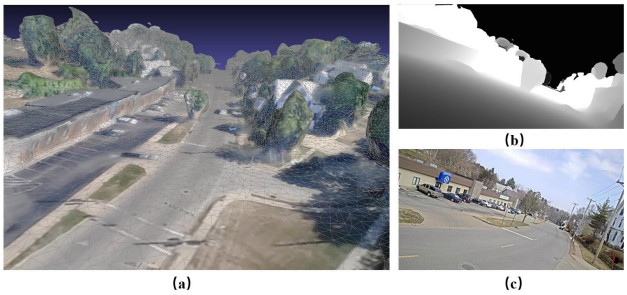

Figure 3. Demonstration of part of reconstructed 3D model. (a) Model mesh with texture. (b) Rendered depth map of camera 'c011'. (c) The original image of camera view.

A lot of factors and clues are been taken into our consideration. Such as temporal information, vehicle ReID feature and 3D geometrical constrains. All of these clues offer a lot of help to the vehicle matching. More details are described as follows.

3.1. Global Environment Set-up

Beyond the conventional 2D multi-object tracking, we propose a novel and efficient tracking framework in 3D which combines geometrical constrains and deep features. To perform vehicle tracking in 3D, we firstly construct 3D models for each intersections, then calibrate cameras and align them into the global model. With given GPS data in dataset, we manually capture screen shots from 3D Google Map then apply Structure from Motion(SFM) algorithm on these images to reconstruct global model of all intersections with textures. Then we further specify the corresponding points between each camera view image and global 3D model manually, from which we could calibrate intrinsic

K, extrinsic T and distortion parameters κ for each camera. Consequently, we can render depth maps for cameras. A example is demonstrated in figure 3.

With these information, we can recover 3D coordinate for bounding boxes in each view. We choose the center point of the bottom line of the bounding box. This point is considered as the contact point with the ground. Let x_c denotes the 2D coordinate of contact point for a bounding box in camera c. Then the 3D coordinate X of this bounding box can be obtained by following equation.

$$\begin{aligned} X &= Td(x'_c)K^{-1}x'_c \\ x'_c &= \Theta(x_c, \kappa) \end{aligned} \tag{1}$$

where $\Theta(x_c, \kappa)$ is a undistortion mapping function which transform a coordinate in distorted image to the one in undistorted image according to distortion parameters κ; $d(x_c)$ is the depth value located at x_c in the depth map of camera c.

The calibration process is demonstrated in figure 2.

3.2. ReID feature

In the image-based vehicle re-identification part, we utilize a recent released strong baseline in person re-identification[10]. We use following training tricks concluded in the origin paper: setting warm-up learning rate[3], random erasing part of image for data set augmentation[20], label smoothing[13] to prevent over-fitting, removing the last stride[12] in backbone for higher spatial resolution, adding a batch normalization[15] layer after the global features from backbone to make the distribution of features that belong to same identities more compact.

In the training stage, We choose ResNet50 and ResNet101 as our backbone, which have pre-trained on ImageNet. We firstly train this model in VehicleID [8] dataset. In the small 800 test set, the mAP reaches 87%. Then we finetune it in AICity Track 2 dataset.

The dataset track 2 also provide the tracklet information. Each tracklet contains a series of images from the same vehicle captured by one camera. Ideally, whose id belonging to the same car should be ranked as neighbours. So taking this track information should be a strong prior to our query. Specifically, we compute the average feature of each tracklet instead of feature of one image and use tracklet feature to match each other. The result with/without tracklet information can be seen in section 4.

3.3. Tracklets Generation

Before vehicle matching, we generate tracklets for each video. The reason why we take a tracklet instead of bounding box as a smallest unit for matching is to reduce the complexity of vehicle tracking problem. We can embed the temporal information of vehicle into tracklets.

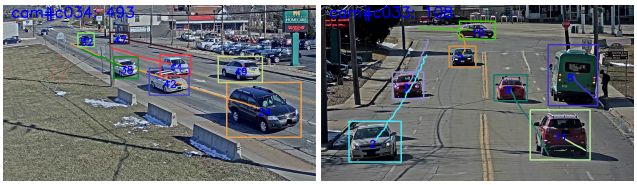

Figure 4. Tracklets example at some point. Each tracklet is marked by different color, with a bounding box which represent current position and historical trajectory.

Algorithm 1: Tracklet Generation

Input : $\mathcal{V}$: input video;
ξ_1: threshold for iou;
ξ_2: threshold for ReID feature distance;
ξ_3: threshold for tracklet confidence;
Output: Serialized tracklets.

$\mathcal{R} = \varnothing$
for *each frame* f^i *in* $\mathcal{V}$ **do**
  for *each trackerlet* r *in* $\mathcal{R}$ **do**
    UpdateTracklet(r, f^i)
  end
  B_d = ObjectionDetection(f^i)
  for *each bounding box* b_d *in* B_d **do**
    flag = True
    for *each tracklet* r *in* $\mathcal{R}$ **do**
      b_t = GetLastBoundingBox(t)
      iou = IoU between b_t and b_d
      dist = ReID feature distance between $f^i_{b_d}$ and $f^i_{b_t}$
      if $iou \geq \xi_1$ *and* $dist \leq \xi_2$ **then**
        UpdateLastBoundingBox(r, b_d)
        flag = False
      end
    end
    if *flag* = *True* **then**
      r = NewTracklet(b_d)
      $\mathcal{R} = \mathcal{R} \cap \{\mathrm{r}\}$
    end
  end
  Serialize and delete tracklet in $\mathcal{R}$ that with lower confidence than ξ_3
end
Serialize and delete tracklets in $\mathcal{R}$

Object detection methods like YoloV3 [11] and Mask R-CNN [4] show high performance on many datasets, but directly using those methods to dispose consecutive frames of a video clip would not present us consistent detection results due to object occlusion, varying exposure, and motion blur. In addition, These detectors are not able to provide

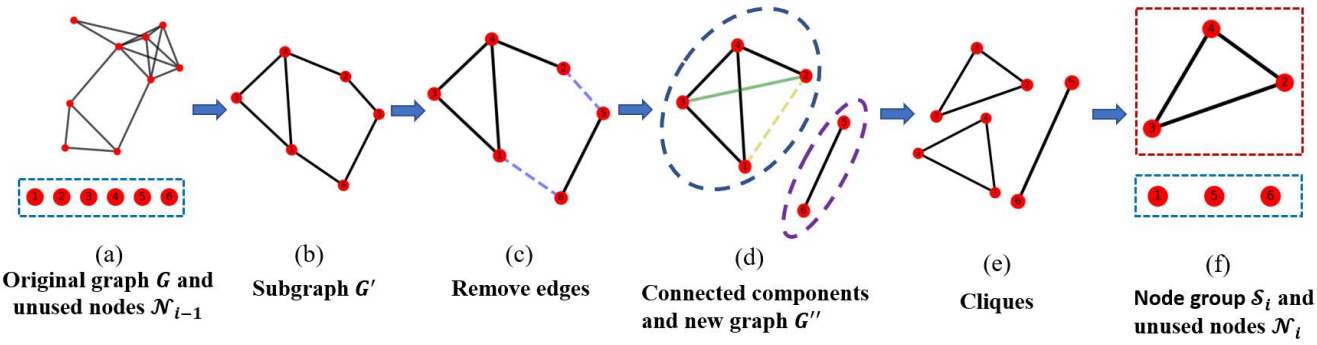

Figure 5. The demonstration of Graph Matching Solver algorithm.

temporal detection result which can build tracklets. Thus we adopt a tracking-detecting strategy to associate detected objects with currently existing tracklets.

When generating tracklets from a video, we run Mask R-CNN [4] on every frame and delete objects that are severely occluded by others. For the remaining detected objects, we determine whether a detected object belong to an existing tracklet by considering both ReID feature distance between their cropped image patch and IoU(Intersection over Union) between their bounding box. If their ReID feature distance and IoU suffice for prefixed threshold, we assume them to be the same object instance. Otherwise we take them as different object instance. In this procedure, We take ECO [2] as our tracker. The whole algorithm is shown in algorithm 1. Fig 4 shows some of our generated tracklets.

3.4. Vehicle Matching

There are thousands of vehicle tracklets in the dataset. Directly matching among these vehicle tracklets is time-consuming and results in unreliable matches. We break down the global matching into pieces to address this problem. Overlapped cameras are clustered together to form several sets, so as tracklets under each camera views. During matching, we firstly match tracklets in the same set to form a tracklet group. Then all tracklet groups are matched globally to form the final resutls. So our approach has a two-stage and bottom-up pipeline. Problem's size is reduced to accepted range in each stage. As a result, we can solver it in a efficient way.

Besides, we formulate matching problem as a graph clique finding problem and further introduce a *Graph Matching Solver* (GMS) to optimize the matching. The implementation details are discussed as follows.

3.4.1 Graph Matching Solver

More specifically, we formulate the matching process as a graph clique finding problem in an undirected weighted

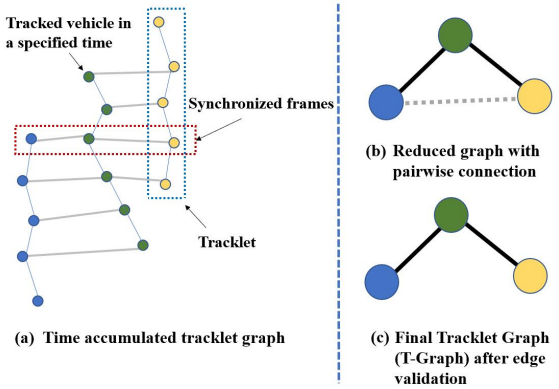

Figure 6. Visualization of matching tracklet set. Each node in (a) represents a tracked vehicle at a frame in a time. Only three tracklets with different colors are demonstrated. Gray lines indicate potential matching relationships; Each tracklet is reduced to a node in (b). The gray dotted line indicates a pairwise connection. But it violates constrains of T-Graph. (c) T-Graph prepared for matching.

graph, within which each node represents a single matching component and weighted edges represent correlation between nodes. Notice that after clustering, nodes in the same group are considered to be equivalent. A *Graph Matching Solver* (GMS) in introduced to find cliques in the graph.

We formulate the matching problem as a graph clique finding problem. There is an undirected weighted graph whose edge's weight responds to the correlation of a node pair. The goal is to cluster these nodes into groups. Nodes in a same group are considered to be equivalent. But, as we known, listing all cliques in a graph is time consuming. It is almost impossible to apply list clique algorithm directly. So we propose a *Graph Matching Solver* (GMS) to solve this problem and find the result. GMS is an iterative algorithm. In each iteration, algorithm decomposes the graph into matched node groups and unused nodes. A new graph contains only unused nodes is then formed and fed to next

iteration until all proper cliques are obtained.

In i-th iteration, the inputs of our algorithm are a original undirected graph $G = \{\mathbb{V}, \mathbb{E}\}$, and unused node set $\mathcal{N}_{i-1}$ which comes from the decomposition of $(i-1)$-th iteration. A threshold ϵ and corresponding minimum clique size m_i are given for filtering edges and select cliques. After this step, the graph's edges should be sparse because of constrains. Filtering will prune lots of edges whose nodes are less likely to be relevant in vehicle tracking problem. Then a new graph $G' = \{\mathbb{V}', \mathbb{E}'\}$, where $\mathbb{V}' = \mathcal{N}_{i-1}$ and $\mathbb{E}' = \{e_{x,y} \mid e_{x,y} \in \mathbb{E}; x, y \in \mathcal{N}_{i-1}\}$ is initialized. Then we break down G' into strong connected components by removing edges whose weight is below ϵ in G'. Next, we obtain all strongly connected components C_j^i using *Disjoint Set Union* algorithm on G'. For each C_j^i, we reconnect any two possible nodes to form a complete graph. The edge's weight is assigned as same as edge in G, or zero if corresponding edge does not exist in G. We then recheck the validity of each edge. We remove the edge from complete graph if its two nodes violate defined constrains. Modified connected components form a new graph G''. In next step, we select cliques with at least m_i nodes from G''. Nodes in same chosen clique are considered to be equivalent. The chosen node group set is denoted as $\mathcal{S}_i$. Other nodes which are not be included in any clique will be unused nodes $\mathcal{N}_i$. Algorithm detail is shown in algorithm 2 and figure 5.

The termination condition of iteration is $\mathcal{N}_i = \varnothing$. It can satisfy this condition by letting $m_i = 1$. The final group set is $\mathcal{S} = \bigcup_{i=1}^{n} \mathcal{S}_i$, where n is the total number of iteration.

3.4.2 Matching in a Tracklet Set

Our goal here is to match tracklets in the tracklet set, in other words, we group tracklets that belong to the same vehicle into same group. Each tracklet set contains tracklets under a set of nearby camera views (i.e. cameras locate in the same crossroads). For vehicles that appear in several synchronized views, their tracklets should be grouped after the matching process. We visit each synchronized frame among these cameras, and select all related tracklets from tracklet set at that time. Then we calculate score metrics of each. Score we use here is a linear combination of ReID score and 3D geometric distance score. If ReID feature is similar and 3D geometric distance at this moment is close, the score will be high. Specifically, we treat each tracklet as a node and let $w_{a,b}^t$ denotes the weight of edge between node(tracklet) r_a and node(tracklet) r_b at time t.

$$w_{a,b}^t = \frac{\lambda}{2}\left(\frac{\mathbf{f}_a^t \cdot \mathbf{f}_b^t}{\|\mathbf{f}_a^t\| \cdot \|\mathbf{f}_b^t\|} + 1\right) + (1-\lambda) \cdot \phi(r_a, r_b, t) \quad (2)$$

where $\mathbf{f}_a^t$ denotes the vehicle's ReID feature at time t in tracklet r_a; λ is the weight parameter for linear combination; $\phi(r_a, r_b, t)$ is 3D geometric distance score function

Algorithm 2: Graph Matching Solver

Input : G: $\{\mathbb{V}, \mathbb{E}\}$, original graph;
ϵ: threshold for removing edges;
$\mathcal{M}$: a set of m_i with a size n;
Output: Node group set $\mathcal{S}$.

$\mathcal{S} = \mathcal{N}_0 = \varnothing$
for *i = 1 to n* **do**
  $\mathcal{S}_i = \mathcal{N}_i = \mathcal{F} = \varnothing$
  G' = BuildSubgraph(G, $\mathcal{N}_{i-1}$)
  G' = RemoveEdges(G', ϵ)
  $\mathcal{C}^i$ = FindConnectedComponents(G')
  for *each C_j^i in $\mathcal{C}^i$* **do**
    for *each ndoe pair (a, b) in C_j^i* **do**
      if *IsValidity(a,b)* **then**
        if $e_{a,b} \notin \mathbb{E}'$ **then**
          $w = 0$
          if $e_{a,b} \in \mathbb{E}$ **then**
            $w = |e_{a,b}|$
          end
          Add an edge $e_{a,b}$ to G' with a weight of w
        end
      else
        if $e_{a,b} \in \mathbb{E}'$ **then**
          Remove $e_{a,b}$ from G'
        end
      end
    end
    $G'' = G'$
    $\mathcal{H}$ = ListAllCliques(G'')
    Sort $\mathcal{H}$ from largest to smallest according to node size of clique.
    for *each $\mathcal{H}_j$ in $\mathcal{H}$* **do**
      if $|\mathcal{H}_j| \geq m_i$ **then**
        $\mathcal{F} = \mathcal{F} \cup \{u | u \notin \mathcal{F} \wedge u \in \mathcal{H}_j\}$
        $\mathcal{S}_i = \mathcal{S}_i + \{u | u \notin \mathcal{F} \wedge u \in \mathcal{H}_j\}$
        $\mathcal{N}_i = \mathcal{N}_i \cup \{u | u \in \mathcal{F} \wedge u \in \mathcal{H}_j\}$
      else
        $\mathcal{N}_i = \mathcal{N}_i \cup \{u | u \in \mathcal{H}_j \wedge u \notin \mathcal{F}\}$
      end
    end
  end
end
$\mathcal{S} = \bigcup_{i=1}^{n} \mathcal{S}_i$

which calculates the 3D distance of vehicles at time t in tracklet r_a and r_b.

$$\phi(r_a, r_b, t) = \exp\left(-\frac{\|\omega(r_a, t) - \omega(r_b, t)\|^2}{\sigma^2}\right) \quad (3)$$

where $\omega(r,t)$ is the 3D position of tracklet r at time t. So $\phi(r_a, r_b, t)$ can scale the 3D distance to a range of 0 to 1, which represents the 3D geometric distance score. we use $\sigma = 1$ and $\lambda = 0.65$ in practical.

As a result, we can obtain an time accumulated tracklet graph as shown in figure 6 (a).

The initial tracklet graph may have too many edges between two nodes, so graph reduction is applied to eliminate the burden. We reduce nodes that belongs to a same tracklet into a new single node to represent the tracklet. Further more, edges between two tracklet are merged into a single undirected edge assigned the weight as the average weight of original edges. Average makes the result more robust in statistics. However, the quality of tracklets does not always preserve. Sometimes multiple tracklets are miss assigned to the same vehicle. So, apart from these time accumulated edges, we also take pairwise relationship of tracklets into consideration. We calculate the ReID score between each two non-overlapping compressed tracklets and add it as a new edge.

Hence, we get an undirected *Tracklet Graph*(T-Graph) demonstrated in figure 6 (c).

In order to narrow down the dimension of solution space. We add some constrains on T-Graph. The validity of a edge between two tracklets in T-Graph is defined as following:

- Two valid tracklets should not have time period overlap under the same camera.
- Two valid tracklets from the same camera should have similar moving directions.
- For the tracklets without time period overlap, the time interval between its two presents no longer than 30 frames.

After we have these constrains, we apply our *Graph Matching Solver* to cluster tracklets into groups.

3.4.3 Matching among Tracklet Groups

After clustering vehicle tracklets groups under nearby camera views, a global matching is applied to solve the matching between different sets of cameras (i.e. cameras locate at different intersections) to handle global matching in long term.

Similar to T-Graph, we build a *Group Graph*(G-Graph) for matching tracklet groups. In this graph, each node represents a tracklet group. The weight of the edge is ReID score of two tracklet groups. The edge's weight between group(node) a and group(node) b is defined as a scaled cosine distant:

$$w_{a,b} = \frac{1}{2} \cdot \left(\frac{\mathbf{f}^a \cdot \mathbf{f}^b}{\|\mathbf{f}^a\| \cdot \|\mathbf{f}^b\|} + 1\right) \tag{4}$$

where $\mathbf{f}^i$ is the feature vector of tracklet group. Specifically, for each tracklet in a tracklet group, we uniform sample maximum 10 frames. Then we use Vehicle ReID network to extract feature vectors for these frames and average them to a vector $\mathbf{f}^i_j$, where $\mathbf{f}^i_j$ donates the averaged feature for j-th tracklet in i-th tracklet group. The feature vector of tracklet group is calculated by $\mathbf{f}^i = \frac{1}{N^i}\sum_j^{N^i} \mathbf{f}^j_i$, and N^i is the number of tracklets in i-th tracklet group.

In the meantime, we apply following constrains on G-Graph.

- Two tracklet groups should not have time period overlap.
- The time interval should meet the approximated time from one cross to the other in the real world.
- Camera sets of two tracklet group should not have intersection. Other wise this scenario should be processed in T-Graph.

In tracklet groups matching, we pairwise calculate distance score for tracklet groups and check its validity. Finally we obtain a G-Graph. Our *Match Graph Solver* is applied to this G-Graph and assign final global ID to each tracklet. Tracklets with same global ID are regarded as same vehicle.

Figure 7. Results of ReID. The left part of the figure shows the queried vehicle. The right part shows the top 10 of query result.

4. Experimental Results

To evaluate our approaches, we submit results on AI City Challenge 2019 evaluation system.

4.1. ReID Result

For Car ReID part, we use a comparatively trivial network as our Reid network, as is attentively discussed in part above. Tested on the AI city 2019 dataset [14], our best network could achieve mAP at 67.84%. However, the performance drops drastically when tested on online submission site. This might probably due to disparate distribution of training and test dataset. As showed in table 1, we observed that the deeper the network goes, the better performance

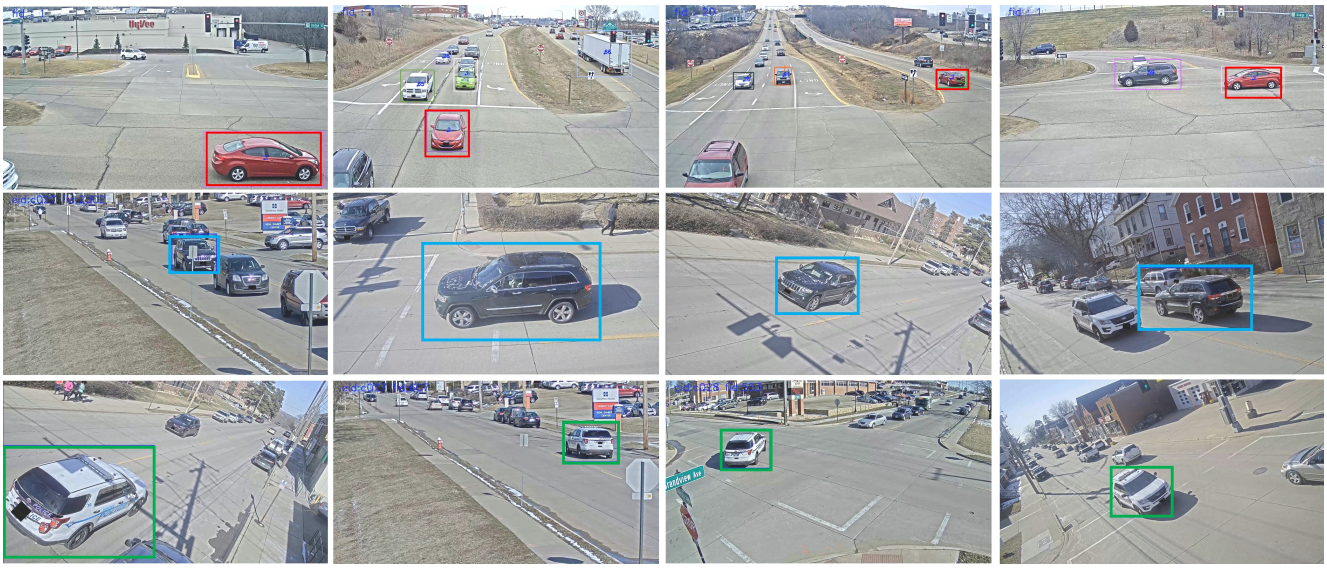

Figure 8. Visualization of some multi-camera vehicle tracking results. Each row shows a grouped tracklet.

Method	map(%)
Resnet50 Baseline	20.48
Resnet50 Track query	27.88
Resnet101 Baseline	23.16
Resnet101 Track query	30.90

Table 1. Results of each query. Baseline is queried in the unit of image. Track query is queried in track unit. The map is calculated in the submission site of track 2

could be achieved. It claims that features with deeper and more abstract information could improve the performance.

We only test two backbones as ResNet50 and ResNet102. In query part, we try to query in the unit of each track and find out the map would be much higher as showed in table1. Since images belong to same car ID should be ranked as closer to each other. Some results are visualized in figure 7

4.2. Vehicle Tracking Result

ϵ	IDF1	IDP	IDR	Precision	Recall
0.75	0.3369	0.3679	0.3107	0.6723	0.5678
0.8	0.3282	0.3964	0.2801	0.6966	0.4923

Table 2. Results on the test dataset of AICity 2019 with different ϵ.

Our approach got 9-th place in the challenge. Detail results are shown on table 2. Some visualization of grouped tracklets are shown in figure 8.

We try different threshold ϵ in *Graph Matching Solver*(GMS). Results show that we can control the number of grouped tracklets by adjusting ϵ. The number of group will be larger if we set a strict threshold (larger ϵ) and vice versa. It means that precision will be influenced in the mean time.

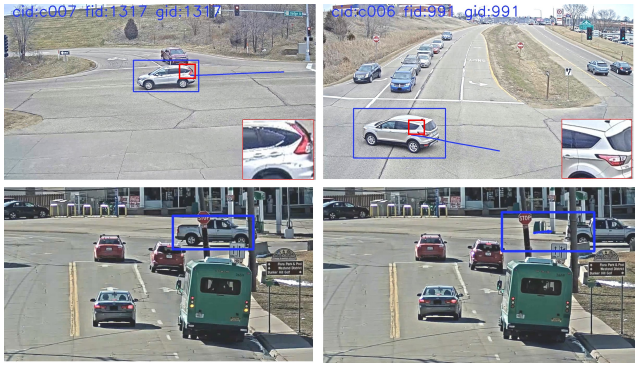

Figure 9. Failure cases of vehicle tracking. The top two images show that two different vehicles are assigned a same ID mistakenly. While the distance score of ReID feature is high. The bottom two images show the scenario that wrong tracklet which cause wrong vehicle tracking result.

We also do qualitative analysis on our result. There are two main situations that affect our results. One is wrong tracklets that not only one vehicle is in a tracklet. This situation will confuse matching because of its ReID feature. The other one is from the ReID feature's performance. It has the scenario that the distance score would be large even these two vehicles are totally different. Also, there are some scenarios that the difference is so small that it can not distin-

guish them easily. Some failure cases are shown in figure 9.

5. Conclusion

In this paper, we introduce our approaches and test result on the AICity 2019 Dataset. Our flexible graph framework can adopt different metrics and constrains into calculation. *Graph Matching Solver* can solver multiview vehicle tracking problem efficiently. Adjusting its threshold can control the performance of algorithm. However, we do not know how far it will goes because there are many limitations can be improved. Such as a better detection and generation of tracklets and better ReID networks. Our method shows great potential for Intelligent Transportation System (ITS) problems.

References

[1] Goutam Bhat, Joakim Johnander, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Unveiling the power of deep tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 483–498, 2018.

[2] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017.

[3] Xing Fan, Wei Jiang, Hao Luo, and Mengjuan Fei. Spherereid: Deep hypersphere manifold embedding for person reidentification. *CoRR*, abs/1807.00537, 2018.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Zhenyu He, Xin Li, Xinge You, Dacheng Tao, and Yuan Yan Tang. Connected component model for multi-object tracking. *IEEE transactions on image processing*, 25(8):3698–3711, 2016.

[7] Saad M Khan and Mubarak Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *European Conference on Computer Vision*, pages 133–146. Springer, 2006.

[8] Hongye Liu, Yonghong Tian, Yaowei Wang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[10] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[12] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang. Beyond part models: Person retrieval with refined part pooling. *CoRR*, abs/1711.09349, 2017.

[13] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[14] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David C. Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *CVPR 2019: IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 499–515, Cham, 2016. Springer International Publishing.

[16] Minye Wu, Haibin Ling, Ning Bi, Shenghua Gao, Hao Sheng, and Jingyi Yu. Generic multiview visual tracking. *arXiv preprint arXiv:1904.02553*, 2019.

[17] Ju Hong Yoon, Ming-Hsuan Yang, Jongwoo Lim, and Kuk-Jin Yoon. Bayesian multi-object tracking using motion context from multiple objects. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 33–40. IEEE, 2015.

[18] Shunli Zhang, Xin Yu, Yao Sui, Sicong Zhao, and Li Zhang. Object tracking with multi-view support vector machines. *IEEE Transactions on Multimedia*, 17(3):265–278, 2015.

[19] Zizhao Zhang, Yuanpu Xie, Fuyong Xing, Mason McGough, and Lin Yang. Mdnet: A semantically and visually interpretable medical image diagnosis network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6428–6436, 2017.

[20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017.