Multimedia Tools and Applications
https://doi.org/10.1007/s11042-023-17397-0

Multi-camera trajectory matching based on hierarchical clustering and constraints

Gábor Szűcs[1] · **Regő Borsodi**[1] · **Dávid Papp**[1]

Received: 15 May 2023 / Revised: 16 August 2023 / Accepted: 1 October 2023

Abstract
The fast improvement of deep learning methods resulted in breakthroughs in image classification, object detection, and object tracking. Autonomous driving and traffic monitoring systems, especially the on-premise installed fixed position multi-camera configurations, benefit greatly from recent advances. In this paper, we propose a Multi-Camera Multi-Target (MCMT) vehicle tracking system using a constrained hierarchical clustering solution, which improves trajectory matching, and thus provides a more robust tracking of objects transitioning between cameras. YOLOv5, ByteTrack, and ResNet50-IBN ReID networks are used for vehicle detection and tracking. Static attributes such as vehicle type and vehicle color are determined from ReID features with SVM. The proposed ReID feature-based attribute categorization shows better performance, than its pure CNN counterpart. Single-camera trajectories (SCTs) are combined into multi-camera trajectories (MCTs) using hierarchical agglomerative clustering (HAC) with time and space constraints (our proposed algorithm is denoted by MCT#MAC). Similarities between SCTs are measured by comparing the mean ReID features cumulated on the trajectory. The system was evaluated on more datasets, and our experiments demonstrate that constraining HAC by manipulating the proximity matrix greatly improves the multi-camera IDF1 score.

Keywords Constrained hierarchical clustering · Object re-identification · Multi-camera multi-target · Trajectory matching · Vehicle tracking

1 Introduction

The detection, identification, and tracking of vehicles and other traffic participants in video recordings is useful in many areas (e.g. in Intelligent Transportation Systems [7], in safety [41]) for modern society. In a modern city, such systems can be elements of self-driving, autonomous vehicles, and automatic traffic control systems. Vehicle tracking has become possible with the development of computer vision and machine learning tools. Among the

✉ Gábor Szűcs
szucs@tmit.bme.hu

[1] Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics, Műegyetem Rkp. 3., 1111 Budapest, Hungary

supervised machine learning methods, convolutional neural networks (CNN) are of outstanding importance in the field of image data processing, whose more modern architectures were developed, for example, ResNet [14], DenseNet [19], and EfficientNet [56]. The most basic use of convolutional networks is image classification [54], but by using the fact that feature vectors of the input image with different semantic content are produced on the layers of the network, they can be used excellently in countless areas of computer vision.

General-purpose object detectors using modified convolutional network architectures can be used to detect vehicles. R-CNN [11] and its modification, Fast R-CNN [10], use two phases for object detection. First, the selective search [60] algorithm is used to search for image parts that may contain an object, and then the feature vector of the region is extracted with a CNN and classified. Faster R-CNN [40, 48] and Mask R-CNN [13] use a separate convolutional network, the so-called RPN (region proposal network), to recommend regions. In contrast, single-phase detectors such as Yolo [47], Yolov5 [22], SSD [34], and EfficientDet [57] do not use a separate step to define the regions, but instead, define the class tags and the positions of the bounding boxes everywhere at fixed positions by running the network once.

The purpose of object re-identification is to recognize the same object in different images. This problem often arises when identifying people or animals [44, 45], but it is also used in general object-tracking methods. Vehicle re-identification is essential for multi-camera vehicle tracking. In re-identification tasks, the goal is to select the images containing the same object as a query image from a gallery as accurately as possible. State-of-the-art solutions (for both vehicle and human re-identification) use special feature extractor CNNs. The feature vectors are determined for both the gallery images and the query image with the help of CNN, then the gallery images are ordered by descending similarity to the feature vector of the query image.

Single-camera object tracking, often called MOT (multi-object tracking) or SCT (single-camera tracking), deals with the detection of objects appearing in video streams, determining their trajectory between frames. The first online algorithms did not use deep learning in the association step, for example, SORT [4] uses the Kalman filter [23] to estimate the expected location of the objects in the next frame, and then matches the objects found to the ones from the previous frame by computing a minimum-cost assignment using the Hungarian method [27], the costs being the overlaps of the detected boxes with the predicted ones (the Hungarian method combined with Kalman filter is commonly used in objects tracking [55]). The IOU tracker [5] does not use the Kalman filter but only looks at the IOU (intersection over union) value of the actual bounding boxes.

Online tracking algorithms using deep learning [2] have two major classes, the two-phase tracking algorithms run an object detector network and then use a separate re-id network to determine the feature vectors of the found objects, while the single-phase ones extract the features simultaneously with the detection. The two-phase modification of the SORT algorithm is DeepSORT [66], which uses re-id characteristics and combines them with the IOU values to calculate the cost matrix, from which it performs the association using the Hungarian method, similarly to SORT. In the case of two-phase trackers such as DeepSORT, POI [74], CNNMTT [39], or Tracktor [3], the detection and re-id extractor networks are completely separate, so they are inferior in speed to state-of-the-art single-phase MOT trackers, such as JDE [64], FairMOT [76], or ByteTrack [75].

To implement multi-camera vehicle tracking, a robust single-camera tracking algorithm, and an inter-camera association algorithm are required. Similar to single-camera tracking, Re-id features play an important role in tracking accuracy [49]. The developed solutions [17, 72] usually follow an offline approach, since in the case of multiple cameras, accurate

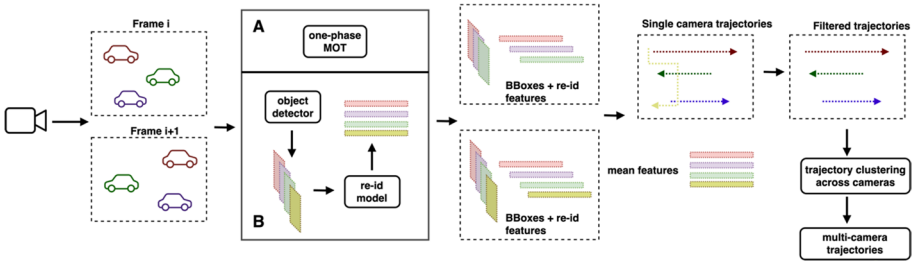

Fig. 1 Overview of the MTMC tracking process using one-phase single-camera tracking (**A**) or a two-phase one (**B**) [43]

camera synchronization is required for online operation, and when a vehicle appears, it would be necessary to check for a match with the trajectories of other cameras based on a single frame, which entails a high possibility of error [33, 53].

An overview of multi-target multi-camera (MTMC) tracking is shown in Fig. 1 [43]. A one- or two-phase MOT algorithm is performed per camera, generating the trajectories and Re-id features. After discarding the incorrect single-camera trajectories, they can be connected between cameras, taking into account the temporal and spatial constraints.

2 Related works

Re-identification is an important element of both single-camera and multi-camera tracking. The problem was most often investigated in the case of humans [38, 73], but in recent years many results have also been obtained in vehicle re-identification [25, 29, 82].

Traditional CNN architectures such as ResNet [14], ResNeXt [69], or DenseNet [19] can be used to extract the re-id features, but more recently their IBN [42] versions are more popular [79]. IBN-net increases the generalization ability of the network by combining Instance Normalization (IN) and Batch Normalization (BN), which is useful for re-identification since the network must extract characteristics of identities that it could not see during training. In practice, of course, a combination of several models gives the best results (ensemble) [37]. A modern approach is the use of a vision transformer-based backbone network, which can learn features with significantly different semantics than traditional CNN architectures [21].

Many minor or major optimizations and tricks have been developed to train convolutional networks. These include general augmentation procedures such as mirroring, rotation, contrast adjustment, or random image cropping, as well as methods for setting the learning rate, such as cosine decay or warmup learning rate, where we initially start from a lower value so that the pre-learned weights do not get destroyed.

When training Re-id neural networks, the loss function is usually the sum of a metric loss and an ID loss. For the classification (softmax) layer, each identity of the data set is considered a separate class, and the ID loss, which is typically cross-entropy, is to be measured. ID loss allows the network to distinguish between different identities (separate them), but a metric loss is commonly used too since the aim is to learn a metric space in which the embeddings of images showing yet unseen identities are close to each other if the images contain the same identity and far if they do not.

The JDE tracker [64] is a single-phase tracker based on the Yolov3 backbone, which can reach almost real-time speed. The network uses the FPN (Feature Pyramid Network) architecture [32], in which a separate prediction head is built on feature maps of several different resolutions of the backbone network, and there is backward feedback, where the smaller feature maps are iteratively added to the larger one by oversampling.

The FairMOT [76] single-phase tracker is based on the CenterNet [81] object detector. Unlike solutions using anchors, the CenterNet detector identifies each object only by its center. The name of FairMOT refers to the fact that in other single-phase trackers, when detection is performed with a network and Re-id features are also extracted, the distribution of the two tasks is not fair, because detection is more emphasized during learning, thereby extracting poor quality Re-id features. FairMOT, on the other hand, can compensate for this shortcoming.

DeepSORT [66] is the first tracking algorithm that uses features extracted with convolutional networks when assigning images to trajectories. DeepSORT determines the distance between i^{th} trajectory and the new j^{th} bounding box by the linear combination of two types of distances (see formula 1),

$$d(i,j) = \lambda \bullet d_1(i,j) + (1-\lambda) \bullet d_2(i,j) \tag{1}$$

where λ is an appropriate constant, the distance $d_1(i, j)$ is the Mahalanobis distance measured between the estimated position of the i^{th} trajectory by the Kálmán filter and the position of the j^{th} bounding box, and $d_2(i, j)$ is the minimum of the measured cosine distances of the feature of the bounding box from the last few features belonging to the i^{th} trajectory.

During the assignment, the algorithm sorts the trajectories in increasing order according to how long ago a new bounding box was assigned to them and looks for a minimal cost association – using the distance $d(i, j)$ as a cost function – between the still unpaired boxes and the trajectories of age t. It performs the assignments only in those cases where both d_1 and d_2 fall under a limit. Similar to SORT, the remaining boxes are associated with the trajectories of age $t = 1$ based on the IOU measure, and thus the new trajectories are created [66]. In DeepSORT, the object detector module can be replaced as desired since the further steps are based only on its results.

ByteTrack [75] is a tracking algorithm that introduces a new assignment step. While other detectors discard bounding boxes below the medium confidence level, ByteTrack extends this by examining the medium ones as well and discarding only those below the very low confidence level, as shown below.

- It divides the detected bounding boxes into three parts, D_{high} (e.g. boxes above 0.5 confidence), D_{low} (e.g. those between 0.2–0.5), and discards the rest.
- It executes pair matching between the bounding boxes in D_{high} with the active tracks, the remaining boxes come to D_{remain}, and the remaining trajectories go to T_1. In this phase, the matching is based on IOU or Re-id similarity metric.
- It matches the bounding boxes in D_{low} with T_1 according to the IOU metric because these are expected to contain objects that are difficult to detect, on which the Re-id similarity gives a weak result. Denote the remaining trajectories with T_2.
- The algorithm initializes new trajectories with bounding boxes from D_{remain}, and it set trajectories from T_2 as missing (similar to DeepSORT).

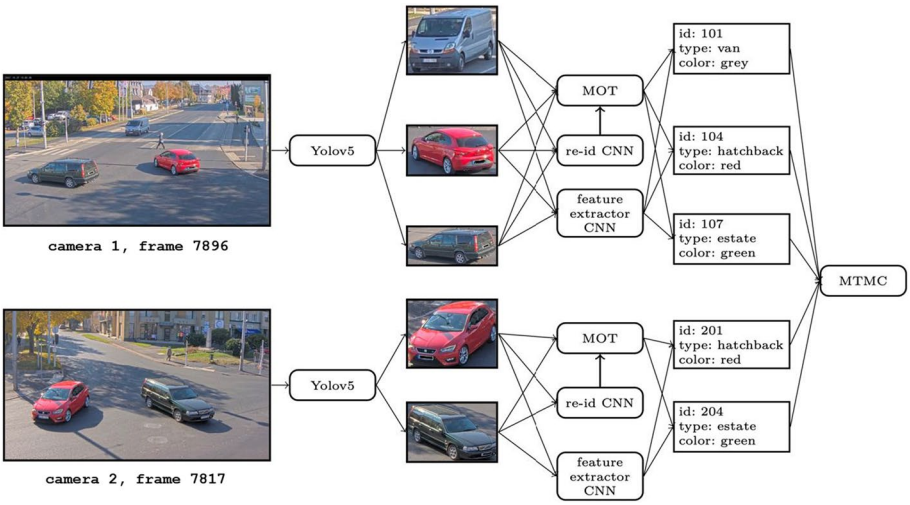

Fig. 2 The process of vehicle tracking is supplemented by the determination of static attributes

The use of D_{low} bounding boxes improves stability in cases where the objects cover each other because in this case the confidence level drops and the Re-id features do not help either.

Multi-camera tracking is commonly performed by matching the appearance features of single-camera tracks using correlation clustering [49] or bottom-up hierarchical clustering on the similarity matrix [18, 33]. Spatial and temporal constraints are often used to reduce the searching space between cameras by setting the similarity of incompatible trajectories to zero [33]. Constraints can be based on the location and distance of cameras [1, 33] the zones where the object entered or exited the view [33, 53], and the order in which objects transit between the same pair of cameras [18]. Some methods rely on overlapping views between cameras [12, 36], while others model the transition time between non-overlapping cameras [20, 59].

3 Integrated vehicle tracking and classification system

3.1 Elements of the integrated system

The implemented vehicle tracking system (Fig. 2) uses a single-phase object detector (Yolov5 [22]) and a ByteTrack or DeepSORT tracker. Yolov5 was chosen as an object detector because of its speed and accuracy. As can be seen in Table 1, Yolov5 runs with much higher FPS than Mask R-CNN or EfficientDet, while also having higher AP (average precision). Note that Yolov7 is a recently published architecture, at the time of research it was not available.

After cutting out the detected bounding boxes from the image, the Re-id networks and the convolutional networks defining the object's attributes are run on them separately. By sending the Re-id vectors, the position and confidence values of the enclosing box to the single-camera tracker, it determines the identifier (ID) for each box. IDs are unique within cameras, so collisions may occur between cameras. The single-camera trajectories are

Table 1 Speed and performance comparison between state-of-the-art object detectors

	GPU	FPS	COCO box AP
Mask R-CNN [13]	NVidia Titan V	5	41.8
EfficientDet-D7x [57]	NVidia Tesla V100	~5	54.4
Yolov5×6 [22]	NVidia Tesla P100	38	55.0
Yolov7-E6E [61]	NVidia Tesla V100	35	56.8

Table 2 Comparison of re-id results using different CNN architectures

	ID loss	Metric loss	Rank@1	Rank@5	Rank@10	mAP
ResNet50-IBN	cross-entropy	contrastive	**72.3**	**89.6**	**93.0**	**76.2**
ResNet50-IBN	cross-entropy	triplet	71.9	89.2	**93.0**	75.9
ResNet50-IBN	cross-entropy	cosface	69.1	86.7	90.7	73.1
ResNet50-IBN	cross-entropy	instance	68.6	86.8	90.8	72.7
ResNet50-IBN	cross-entropy	arcface	68.3	87.3	91.0	72.5
ResNet50-IBN	cross-entropy	sphere	68.0	86.6	90.5	72.1
ResNet50-IBN	cross-entropy	-	68.4	86.8	90.4	72.5
ResNet50	cross-entropy	-	64.6	84.8	89.6	69.2
EfficientNet-B0	cross-entropy	-	63.7	83.3	88.1	68.2
HRNet	cross-entropy	-	70.4	87.8	91.3	74.2
DenseNet121	cross-entropy	-	65.8	85.6	90.4	70.2

The bold entries are the highest scores in the last four columns presenting the best CNN architectures

formed by the detections belonging to the same identity on the given camera. Our system aggregates these at the end of the run and matches them between several cameras, which presumably contain the same vehicle. Further subsections discuss each step in more detail.

3.2 Vehicle re-identification

Numerous CNN architectures can be used for re-identification. We trained DenseNet121 [19], HRNet [62], EfficientNet-B0 [56], ResNet50 [14], ResNet50-IBN [14, 42] on a random 75% of VRIC train and tested on VRIC test [24]. A summary of the results can be seen in Table 2, more details can be seen in our GitHub repository.[1] All training used cross entropy as ID loss, while for metric loss multiple alternatives were tested. Training parameter values were enumerated below.

- Batch size = 32
- Total epoch = 20
- Warm epoch = 3
- Erasing = 0.5
- Samples per class = 4
- Learning rate = 0.05

[1] https://github.com/regob/vehicle_reid.

Based on the preliminary tests, ResNet-IBN baselines (ResNet50-IBN, and later ResNet101-IBN) were used for re-identification. The head of the model uses the BNNeck architecture [38], which consists of an average pooling layer, followed by a fully connected, batch normalization, and a leaky ReLU layer to compute the re-id embeddings, on which metric loss is measured [78]. In the inference phase, the re-id embeddings are computed for each image twice, once for the original image, and once for the vertically flipped version. The mean of the two embeddings is used [78]. The following enhancements are also used.

- Cosine learning rate scheduling and SGD optimizer (learning rate, i.e. LR = 0.05 for the head and 0.005 for the backbone) with a 3-epoch warm-up and 30 epochs in total,
- supervised contrastive loss and cross-entropy id loss,
- P x K batch sampling (P identities and K images per identity) [38],
- Random Erasing Augmentation,
- MixStyle is used in some models for better generalization [80].

3.3 Determining the static attributes of vehicles

Static attributes of vehicles are properties that do not change over time (at least during the tracking), such as color, type, number of axles, or whether the vehicle has a roof rack or a trailer. The attributes can be determined from the Re-id features or with the help of separated convolutional neural networks (CNN). In the latter case, after object detection, each network runs on the contents of all bounding boxes, while in the former, we classify the Re-id features, which is more efficient in terms of performance. Since the estimates per frame can be imprecise and the static attributes must be constant during a trajectory, at the end our system produces a prediction per trajectory, the estimates are summed and weighted by the area of the given bounding box, and the one with the highest total weight is chosen. The weighting is important because, in the case of small boxes, the estimates for low-resolution images are inaccurate, and thus the more accurate, large bounding boxes are taken into account with greater weight.

A similarity between the classification of Re-id features and the use of CNN is that in both cases a convolutional network extracts the features based on which a classifier makes a decision. In the case of dedicated CNNs, the classifier is the FC (fully connected) part of the network, while in the case of Re-id vectors, SVM (Support Vector Machine), GBM (Gradient Boosting Machine), Random forest, or an FC network includes the classifier. The difference in the two approaches, however, is that when classifying the Re-id vectors, we do not train the network that extracts the features, only the classifier; so it is questionable to what extent the given static attribute is a latent part of the Re-id feature vectors. To test the system, our model was trained to determine color and type. An example of all vehicle types recognized by the system can be seen in Fig. 3.

3.4 Single-camera tracking with post-processing

Among the tracking algorithms, DeepSORT and the version of ByteTrack built on FairMOT were integrated into our system, using only the IOU distance metric, i.e. the Re-id features were not used for single-camera tracking at ByteTrack. For a comparison between ByteTrack and DeepSORT see Table 5.

Fig. 3 The different vehicle types

The tracking result may still contain incorrect trajectories, so it is worth taking several post-processing steps even in the single-camera phase. Thus, (as a step) we solved in our system that trajectories appearing on fewer than a given number of frames were discarded.

Based on the zone list of the trajectory, we determined what routes are possible on the camera image, and thus we can divide the tracks into two categories, valid are those that comply with the defined zone rules, and invalid are all others. The zones cannot overlap, otherwise, it is non-deterministic which one the box in their intersection is classified into. Valid zone lists (rules) can be defined per camera using regular expressions.

In the first step of the post-processing, the trajectories were divided into valid and invalid categories based on whether they fit any zone rules. Since errors occur when the trajectories of two vehicles merge, it is advisable to start by splitting the invalid trajectories into half ones.

- Any invalid trajectory that can be separated into two valid ones is immediately split into two trajectories.
- For the remaining trajectories, our algorithm checked whether there were time and/or feature gaps. A time gap means any missing frame, while a feature gap can be defined as a large distance between the accumulated feature and the feature calculated on the next frame. Where this feature distance exceeded a large limit value, (or a smaller limit value, but there was also a time gap), the trajectory can be split.

After splitting, our algorithm tries to merge the invalid trajectories into valid ones matching a zoning rule. Here, the same time and feature criteria can be applied as for the splitting, but the direction is of course reversed for the features, the distance should be below a limit value, which is a smaller value than the limit value taken during the separation, otherwise, we could combine the same ones again. In addition to examining the small time gap, the overlapping of the enclosing boxes must also be observed, i.e. the smaller the time gap between the two trajectories, the larger the IOU value should be between the last bounding box of the first trajectory and the first bounding box of the second.

4 Multi-camera trajectory matching algorithm based on hierarchical clustering and constraints (MCT#MAC)

The multi-camera trajectory matching algorithm was designed for a general camera system, so it can be applied to four cameras or any case by modifying the configuration. The algorithm performs hierarchical clustering taking into account time and space

constraints. Initially, each T_i multi-camera trajectory (MCT) consists of a τ_i single-camera trajectory (SCT). The similarity between T_i and T_j can be measured by the re-id characteristics of the SCTs in them as detailed below.

The similarity between SCTs τ_i and τ_j is the cosine similarity computed between the mean features as detailed in Eqs. 2 and 3, where K is the number of the features in the aggregation (which is equal to the number of the images in which the object was detected and identified).

$$\mathrm{sim}(\tau_i, \tau_j) = \frac{\overline{h}_i^{(re\_id)} \bullet \overline{h}_j^{(re\_id)}}{\left|\overline{h}_i^{(re\_id)}\right| \bullet \left|\overline{h}_j^{(re\_id)}\right|} \tag{2}$$

$$\overline{h}_i^{(re\_id)} = \frac{\sum_{k=1}^{\mathrm{K}} h_{i,k}^{(re\_id)}}{\mathrm{K}} \tag{3}$$

We developed three variants of hierarchical clustering depending on the definition of different T_i multi-camera trajectories. The first version is the single linkage solution, where the similarity between the most similar SCTs is considered as the similarity between two MCTs as can be seen in Eq. 4.

$$\mathrm{sim}\left(T_i, T_j\right) = \max_{\tau_1 \in T_i, \tau_2 \in T_j} \mathrm{sim}(\tau_1, \tau_2) \tag{4}$$

The next variant written in Eq. 5 is the average linkage solution, where the similarity between two MCTs is defined as the average similarities between the corresponding SCTs.

$$\mathrm{sim}\left(T_i, T_j\right) = \frac{\sum_{\tau_1 \in T_i, \tau_2 \in T_j} \mathrm{sim}(\tau_1, \tau_2)}{\left|T_i\right| \bullet \left|T_j\right|} \tag{5}$$

At the complete linkage solution, the similarity between two MCTs is defined in Eq. 6 as the minimum similarity between the corresponding SCTs.

$$\mathrm{sim}\left(T_i, T_j\right) = \min_{\tau_1 \in T_i, \tau_2 \in T_j} \mathrm{sim}(\tau_1, \tau_2) \tag{6}$$

Constraints between cameras can be defined with three matrices. The first is *the C* compatibility matrix, whose element C_{ij} is 1 or 0 depending on whether two cameras are compatible with each other (or not). The rests are T^{min} and T^{max} matrices, which give the time window between camera pairs in which the transition is possible. If the trajectory τ_1 of the i^{th} camera and the trajectory τ_2 of the j^{th} camera is the investigated target that whether they can be matched, then τ_2 should include on the j^{th} camera within the interval $[(T^{min})_{ij}, (T^{max})_{ij}]$ measured from the departure time of τ_1, besides the camera compatibility.

Algorithm in Fig. 4 briefly describes our "Multi-camera Trajectory Matching Algorithm based on hierarchical clustering and Constraints" (MCT#MAC) algorithm. Its most important element is the multi-camera trajectory compatibility test function (see Fig. 5), which checks for every possible SCT pair whether the transition is possible in two directions, i.e. whether the time window taken from the end of one trajectory intersects the time window

Fig. 4 MCT#MAC algorithm

Data: $T = \{T_1, T_2, \ldots, T_n\}$, MIN_SIM
$d \leftarrow \max_{i,j} (\text{sim}(T_i, T_j) \cdot \text{Compatible}(T_i, T_j))$
while $|T| > 1$ and $d >$ MIN_SIM **do**
  $x, y \leftarrow \arg\max_{i,j} (\text{sim}(T_i, T_j) \cdot \text{Compatible}(T_i, T_j))$
  merge(T_x, T_y)
  $d \leftarrow \max_{i,j} (\text{sim}(T_i, T_j) \cdot \text{Compatible}(T_i, T_j))$
end
return T

Data: T_i, T_j
if have_cameras_in_common(T_i, T_j) **then**
  return 0
end
for $\mathcal{T}_a$ *in* T_i **do**
  for $\mathcal{T}_b$ *in* T_j **do**
    $c_a, c_b \leftarrow \mathcal{T}_a$.cam, $\mathcal{T}_b$.cam
    /* is the $\mathcal{T}_a \rightarrow \mathcal{T}_b$ transition valid? */
    if $\mathbf{C}[c_a][c_b] > 0$ and $\mathcal{T}_b$.start $\leq \mathcal{T}_a$.end $+ \mathbf{T}^{\max}[c_a][c_b]$ and $\mathcal{T}_b$.end $\geq \mathcal{T}_a$.end $+ \mathbf{T}^{\min}[c_a][c_b]$ **then**
      return 1
    end
    /* is the $\mathcal{T}_b \rightarrow \mathcal{T}_a$ transition valid? */
    if $\mathbf{C}[c_b][c_a] > 0$ and $\mathcal{T}_a$.start $\leq \mathcal{T}_b$.end $+ \mathbf{T}^{\max}[c_b][c_a]$ and $\mathcal{T}_a$.end $\geq \mathcal{T}_b$.end $+ \mathbf{T}^{\min}[c_b][c_a]$ **then**
      return 1
    end
  end
end
return 0

Fig. 5 Algorithm for compatibility examination

defined by the start and end of the other trajectory. The input data ("Data") for the algorithm is the set of trajectories denoted by T.

The pseudocode of the algorithms does not contain all details, we used some technical optimizations in the implementation. For example, (i) we stored the similarity values in a heap queue, (ii) when merging two MCTs, only the similarity of the new track to all others has to be calculated, (iii) the similarity matrix was pre-calculated from the matrix M based on Re-id row vectors by MM^{T} multiplication on the GPU. This way, the total complexity of the algorithm is $\mathcal{O}(n^2 \log n)$, assuming a constant number of cameras.

- Computing the compatibility of SCTs takes $\mathcal{O}(n^2)$ time.
- Computing the similarity matrix also takes $\mathcal{O}(n^2)$ time.
- At the start, n^2 elements are added to the heap, then $(n-1) + (n-2) + \cdots + 1 = \mathcal{O}(n^2)$ while the algorithm is running. The cost of managing the heap is $\mathcal{O}(n^2 \log n)$.
- The number of merging steps is $n-1$ at most, and during a merging step at most, the other $n-2$ trajectories need to be checked. Calculating the pairwise similarities of SCTs for two compatible MCTs is constant, and therefore the merging steps in total take $\mathcal{O}(n^2)$ time.

The algorithm differs from common hierarchical clustering methods that enforce the constraints by setting the distance value between two incompatible tracks to a very high (e.g. 10^9) number in the distance matrix, which works with complete, average, centroid, or ward linkage (i.e. not with single linkage, because of the chaining effect). This way

strict constraints can be defined, and therefore if there is at least one pair of tracks between two clusters that are incompatible, the clusters cannot be merged, because the distance is infinite.

Our method on the other hand uses weak constraints, which have to be satisfied for only one pair of SCTs between MCTs. This makes sense in cases where we do not want to (or cannot) define criteria for transition between each pair of cameras. E.g., assume we have N cameras sequentially on a highway. This way we only have to define the transition time and compatibility between neighboring cameras (N – 1), instead of providing vague transition times between N^2 cameras. The strict constraints would not be useful in defining the camera order in this case either.

Temporal constraints only work if the cameras are synchronized, i.e. a global time stamp can be assigned to each frame. Synchronization can be done with the "scale" and "offset" parameters. The former can handle the multiplicative, the latter the additive differences. In the case of a multiplicative difference, the cameras do not capture the same amount of time during the playback time of 1 s, while in the case of an additive difference, the start times of the video recordings are different. If we configure s scale and o offset for a camera, the global timestamp for the i^{th} frame of the camera is calculated as given in Eq. 7, where FPS is the video speed in frames per second.

$$t = \frac{i}{\text{FPS} \bullet s} + o \tag{7}$$

If there is no real-time data for either camera, then one can be considered as a reference and the others synchronized to its clock signal, i.e. if the same event takes place in 100 s on the recording of the reference camera, and in 105 s on the other, then the second camera its scale is 1.05.

5 Results of the evaluation

5.1 Evaluation plan

Since the tracking system can be divided into several modules, it is advisable to evaluate them separately, so that we get a more accurate picture of the overall performance of the system, and those elements that perform poorly represent bottlenecks of the tracking system. The components that can be evaluated separately are listed below [50].

- Evaluation of re-id models is possible with the mAP and Rank@K measures.
- Static models as a traditional classification task, therefore, the accuracy and macro averaged classification metrics can be used for their evaluation.
- Single-camera tracking, where MOTA, IDSW, and IDF1 metrics can be used.
- Multi-camera tracking: Single-camera tracking metrics can also be used for evaluation in such a way that the camera recordings are concatenated one after the other, and the trajectories of the combined identities between the cameras are considered to be a continuation of each other.

In the case of object tracking, True Positive (TP), True Negative (TN), and False Positive (FP) values should be calculated per frame. Denoting the set of bounding boxes found by the tracking algorithm in the t^{th} frame by Π_t, and the ground truth boxes by Γ_t,

Table 3 Comparison results of the re-id models

model	dataset	mAP		
		VW10	VW3	VRIC
ResNet50-IBN				
f = 512, cosine	VeRi-Wild	49.10	68.14	19.50
f = 2048, MixStyle, cosine	VeRi-Wild, Cityflow train*, VRIC	57.88	72.76	73.90
f = 2048, cosine	VeRi-Wild, Cityflow train*, VRIC	**58.62**	**73.40**	**76.08**
ResNet101-IBN				
f = 2048, cosine	VeRi-Wild, City	54.56	72.58	18.3

The bold entries show the largest mAP values in each dataset

*Excluding the validation set, which was used to test the classification of static attributes

these should be matched before determining the number of TPs. The IOU (intersection over union) values are used for matching, which is the ratio of the intersection and the union of two bounding boxes. On the paired graph formed by the boxes in Π_t and Γ_t as nodes, a maximum weight matching can be performed, where the edge weights are represented by the IOU values. There is usually a pre-defined minimum IOU value (0.5 generally), below which we no longer match two boxes.

A tracking-specific metric is the IDSW, where the $IDSW_t$ is the number of ground truth boxes in Γ_t in the t^{th} frame that was matched with a box in Π_t, but was matched with a box with a different ID than the last time (when it was last matched a detection to this ground truth identity if there was one already). IDSW is the sum of the $IDSW_t$ values. MOTA (Mean Object Tracking Accuracy) as an important goodness indicator can be defined using the previous metrics, multiplying this by a hundred gives a percentage value [9]. The goodness indicator introduced in Eq. 8 has focused exclusively on frame-by-frame values.

$$\text{MOTA} = 1 - \frac{FN + FP + IDSW}{\sum_t |\Gamma_t|} \tag{8}$$

Instead of on the frame level, the matching can be executed on the level of entire trajectories, and not for each frame separately, as in the case of previous metrics. A minimum-cost matching can be determined on the graph (with "virtual" FP and FN nodes added [50]) with the cost being the number of FP and FN errors generated if we match the two tracks, and then the IDTP, IDFN, IDFP values can be calculated (in a similar way as at the frame level, but prefix 'ID' shows that this belongs to the level of trajectories, i.e. to identities). The equivalents of Precision, Recall, and F_1 values IDP, IDR, and IDF_1 can be calculated from the values of IDTP, IDFN, and IDFP. MT (Mostly Tracked) is the number of tracks to whose boxes were matched a detected box in at least 80% of the cases (regardless of ID), PT (Partially Tracked) is the number of tracks found between 20 and 80%, while ML (Mostly Lost) counts those whose boxes were matched in at most 20%. In the evaluation of tracking, we used different indicators (MOTA – Mean Object Tracking Accuracy, IDP – Identity Precision, IDR – Identity Recall, IDF_1 – Identity F_1, IDSW – Identity Switches, PT – Partially Tracked, ML – Mostly Lost).

Table 4 Comparison results of the static models

model	data	Acc (type)	Acc (color)
SVM (C = 1.0 (type) | C = 1.0 (color))	VeRi-Wild, CityFlow train	**0.853**	**0.781**
GBM (n = 800, lr = 0.1, depth = 5, colsample = 0.5)	VeRi-Wild, CityFlow train	0.834	0.771
NN*	VeRi-Wild, CityFlow train	0.825	0.730
ResNet101**	VeRi-Wild, CityFlow train	0.754	0.724
SVM (C = 0.01 (type) | C = 0.1 (color))	CityFlow train	0.849	0.767
GBM (n = 800, lr = 0.1, depth = 5, colsample = 0.5)	CityFlow train	0.842	0.746
NN*	CityFlow train	0.840	0.730
ResNet50**	CityFlow train	0.790	0.660

Entries in bold indicate that the first static model (SVM) has the highest accuracy for type and color

*Neural network (FCNN) with a single hidden layer: linear(4096) – ReLu – Dropout(p = 0.5) – linear(N)

**CNN trained on the images, unlike other models, which were trained on the ReID features

5.2 Evaluation of the re-id models

The re-id models were tested only on the VRIC [24, 65] and VeRi-Wild [35] (abbreviated VW3 and VW10 for 3K and 10K sizes) test sets (Table 3.), because the CityFlow test set [58] has private labeling. We used cross-entropy ID loss and supervised contrastive metric loss during the training of each model in the table. The length of the Re-id features (f) is 2048 with one exception. The best models were trained on all datasets. The only difference between the two best ResNet50 models is that one of them used MixStyle as well during the learning. Using MixStyle slightly worsened the mAP values, presumably due to its regularizing effect.

5.3 Evaluation of the static models

The original CityFlow training set was divided into two parts, the train part (CFtrain) and the CityFlow validation part (CFval). CFtrain was set for the training of the re-id models and the static characteristics, while the static characteristics were measured on the CFval data set, so the latter one was only used for the evaluation of the static characteristics (vehicle type and color).

We paid attention that two parts (train and validation sets) should be disjoint at the level of identities, because it is much easier to classify a vehicle that the model has already seen, even in a different camera position. The static characteristics can be determined from the Re-id features or the images. We summarized the results of the trained models for both approaches in Table 4., of which the SVM, GBM, and NN classifiers were trained on Re-id features, while ResNet50 was trained on images. The re-id model was ResNet50-IBN trained with MixStyle.

5.4 Evaluation of single-camera tracking

We evaluated single-camera tracking on the S01 and S02 scenarios of the Cityflow dataset. The annotation of the dataset is not perfect, but it can be set so that finding unlabeled vehicles is not considered an error (in line with the official recommendation). The results

Table 5 Comparison results of the single-camera tracking

video	tracker	MOTA↑	IDF1↑	IDP↑	IDR↑	IDSW↓	MT↑	PT↓	ML↓
S01 c001	ByteTrack	97.9%	98.1%	98.5%	97.7%	43	75	0	2
S01 c001	DeepSORT	99.5%	99.5%	99.7%	99.3%	5	76	1	0
S01 c005	ByteTrack	99.1%	98.5%	98.9%	98.2%	30	89	5	0
S01 c005	DeepSORT	97.8%	94.8%	95.7%	93.9%	39	90	4	0
S02 c006	ByteTrack	90.6%	89.4%	91.8%	87.1%	217	111	12	1
S02 c006	DeepSORT	90.5%	90.5%	93.1%	88.1%	207	111	12	1
S02 c009	ByteTrack	93.2%	90.0%	91.7%	88.3%	258	117	20	0
S02 c009	DeepSORT	92.3%	91.1%	93.0%	89.2%	295	115	22	0

Table 6 Comparison results of the multi-camera tracking

data	linkage	time	cam	MOTA↑	IDF1↑	IDP↑	IDR↑	IDSW↓	MT↑	PT↓	ML↓
S01	average	+	+	96.0%	92.8%	94.6%	91.2%	138	90	5	0
S01	complete	+	+	95.5%	89.9%	91.8%	88.1%	177	87	7	1
S01	single	+	+	98.4%	**94.4%**	94.9%	93.9%	197	92	3	0
S01	single	-	+	96.3%	91.2%	92.7%	89.8%	177	90	5	0
S02	average	+	+	91.0%	**90.1%**	93.3%	87.2%	522	118	27	0
S02	average	-	+	89.8%	87.5%	90.7%	84.4%	694	118	26	1
S02	average	-	-	86.1%	74.7%	79.4%	70.6%	579	100	42	3
S02	complete	+	+	85.9%	85.7%	91.1%	80.9%	613	112	32	1
S02	complete	-	+	85.2%	84.3%	89.6%	79.6%	770	112	32	1
S02	complete	-	-	81.8%	76.5%	83.1%	71.0%	759	99	44	2
S02	single	+	+	91.6%	89.9%	92.6%	87.4%	582	22	22	0
S02	single	-	+	90.4%	86.0%	88.8%	83.4%	742	24	24	2
S02	single	-	-	24.9%	7.7%	4.8%	24.9%	1	9	42	94

The bold entries present the best IDF1 results in S01 and S02 scenarios, respectively

measured on 2–2 selected cameras of scenarios S01 and S02 (Table 5) are acceptable with both trackers. In the headers of the columns, the arrows present the direction of goodness at each indicator (larger or smaller is better).

There was not much difference between ByteTrack and DeepSORT results on these recordings. It varies which one achieved a better MOTA or IDF1 value. At the same time, the ByteTrack only uses IOU metrics and not Re-id features, thus it is less computationally demanding.

5.5 Evaluation of multi-camera tracking

The multi-camera tracking was evaluated on the five-camera S01 and four-camera S02 camera systems of the CityFlow dataset, as can be seen in Table 6.. In both cases, each camera sees an intersection from different directions, but there is no overlap because the central area is only partially visible. In these situations, the configuration of the algorithm is detailed below.

Table 7 SCT and MTMC runtimes on the S02 scenario

phase	info	GCloud (total)	Kaggle (total)
SCT c006	2110 frames	2 min 17 s (15.33 fps)	9 min 52 s (3.56 fps)
SCT c007	1965 frames	1 min 42 s (19.09 fps)	4 min 40 s (7.00 fps)
SCT c008	1924 frames	2 min 7 s (15.14 fps)	9 min 23 s (3.41 fps)
SCT c009	2110 frames	2 min 39 s (13.26 fps)	13 min 14 s (2.66 fps)
MTMC (total)	639 input tracks	0.606 s	1.134 s

- Each camera is compatible with all others, but not with itself.
- In both camera systems, there were additive and multiplicative time differences between the cameras. We detected this by looking for reference points at the beginning and end of the videos.
- T^{min} and T^{max} for each pair of cameras are -6 and 6, i.e. the connection between any two cameras is possible if the exiting track on one camera in the t^{th} second can be found on the other camera in the [t – 6 s, t + 6 s] interval.

The MCT#MAC algorithm can also be run without a camera configuration, in which case two multi-camera trajectories can be merged if they do not contain a track from a common camera and if they meet the similarity criterion of the hierarchical clustering procedure. In addition to the different clustering variations with different similarity calculation methods, we also tested the method without time constraints. The camera and time constraints improved the result in both camera systems, achieving an IDF1 of 94.4% on S01 and 90.1% on S02 using single and average linkage, respectively. Single and average linkage performed best, followed by complete, which achieved much lower recall values (IDR) due to the fact that it looks at the furthest trajectories when pairing them. Furthermore, the last row of Table 6. shows that the chaining property of the single linkage makes it unusable without camera constraint.

5.6 Performance evaluation

The experiments were performed using NVidia driver 510.47.03, Python 3.7, and PyTorch 1.12.0 versions on Google Cloud virtual machines and Kaggle Notebooks (see Table 10 at the end of the paper for software and hardware details). The performance of the system was measured under different hardware conditions (see Table 7 with 4 cameras from c006 to c009), and in all cases, the CPU was limited. Less than 80% utilization was measured on the GPUs with 100% CPU utilization; these experiences show that tracking is highly CPU-demanding.

The essential part of the running time was single-camera tracking (SCT) since the multi-camera trajectory matching phase ran within 1 s. Therefore, the running time of each phase of the single-camera tracking was also measured, as can be seen in Table 8. The evaluation was made using the ByteTrack tracker in the Google Cloud environment.

Extraction of re-id characteristics and detection accounted for a substantial part of the running time. The ByteTrack tracker ran at an average of 1.79 ms (see “tracker” row and “mean” column in Table 8), which is really fast, while DeepSORT’s running times were

Table 8 Runtime distribution (in milliseconds) between tasks of SCT on camera c006 of the S02 scenario

task	minimum	mean	maximum
detector	22.09	22.84	100.23
detection filter	0.12	0.26	0.74
re-id	18.62	34.25	95.95
nonmax suppression	0.18	0.41	1.01
attribute extraction	0.33	0.45	0.80
tracker	0.95	1.79	3.73
displays	0.12	0.87	2.43

between 6.5 and 17 ms on the cameras, which is quite significant. The Yolov5×6 detector ran at 22.84 ms (see "detector" row in Table 8) on average, i.e. the object detection alone is capable of over 40 FPS even with the largest model. The speed of the videos is 10 FPS, and the V100 averaged 15.33 FPS (see c006 results on GCloud in Table 7), so real-time tracking is possible on these recordings. In the case of multiple objects or higher video FPS, real-time single-camera tracking is also possible with three modifications, downsampling, hardware, and re-id optimization as detailed below.

- Downsampling: By downsampling the video, we can reduce the FPS, for example by keeping only every 3rd frame from a 30 FPS video.
- Hardware: Stronger hardware components (GPU, CPU) can significantly improve the running time. In tests, for example, the Nvidia V100 GPU ran at only 60%, showing that just a more powerful CPU can improve performance greatly.
- Re-id optimization: When using ByteTrack, re-id features are not necessary for single-camera tracking, so it is unnecessary to extract them on every frame. If it only runs every 2nd or 3rd frame, that might be enough for multi-camera tracking.

6 Discussions and limitations

The results show that the vehicle tracking system and the static feature extraction subsystem are suitable for analyzing video recordings in practice. The Re-id component of our system reached 76.1% mAP value on the VRIC test set, the accuracy of the classifier was 85.3% and 78.1% for the type and color of the vehicle, respectively. At the evaluation of the MCT#MAC algorithm, the single linkage version was the best with 98.4% MOTA and 94.4% IDF1 values in the CityFlow S01 scenario, and the average linkage version was best with 91.0% MOTA and 90.1% IDF1 values in CityFlow S02 scenario. Our experiments demonstrate that constraining hierarchical agglomerative clustering by manipulating the proximity matrix greatly improves the goodness of Multi-Camera Tracking.

Regarding re-identification, the 76.1% mAP value on the VRIC is comparable to the results of state-of-the-art methods, e.g. with SST [51] 63.93%, with TCL [51] 71.66%, with EMRN [51] 79.61%, with VSLN [63] 85.51%. On the VeRi-Wild test set, our re-id component mostly outperforms the state-of-the-art methods, e.g. the Batch Weighted method [28] gives 51.6% and 70.5%, the Transformation State Adversarial method [8] gives 58.77% and 72.77%, and the Self-supervised Attention method [26] gives 67.7% and 80.9%, while our results are 58.62% and 73.40% mAP values on the VeRi-Wild 10K and the VeRi-Wild 3K, respectively.

Table 9 Summary of comparison between our effort and state-of-the-art MTMC results on S01, S02, S05, and S06 scenarios of the CityFlow dataset

Paper	Year	Scenario	Number of cameras	IDF1
[16]	2019	S02 + S05	23	0.6519
[15]	2019	S02 + S05	23	0.6653
[18]	2019	S02 + S05	23	0.7059
[31]	2019	S02 + S05	23	0.6865
[46]	2020	S06	6	0.4585
[33]	2021	S06	6	0.8095
[72]	2021	S06	6	0.7787
[67]	2021	S06	6	0.7651
[53]	2021	S06	6	0.6910
[70]	2022	S06	6	0.8486
[30]	2022	S06	6	0.8437
[71]	2022	S06	6	0.8371
[52]	2022	S06	6	0.8348
Our effort	2023	S01 (w/o overlap)	5	**0.9120**
Our effort	2023	S02 (w/o overlap)	4	**0.8750**
Our effort	2023	S01 (w/ overlap)	5	**0.9440**
Our effort	2023	S02 (w/ overlap)	4	**0.9010**

Entries in bold indicate that our solutions have the highest IDF1 scores compared to all competitors

The MTMC results we got on the S01 and S02 scenarios are difficult to compare to published ones, because the CityFlow dataset is part of the NVidia AI City Challenge, and the test dataset is kept private. Competitors submit their results, and the organizing team evaluates them, and therefore MTMC results in the literature were tested on a slightly different dataset than ours, as can be seen in the following year-by-year breakdown.

- 2019 – S02 (4 cameras, with overlaps) + S05 (19 cameras, with overlaps)
- 2020 – S06 (6 cameras, no overlap)
- 2021 – S06 (6 cameras, no overlap)
- 2022 – S06 (6 cameras, no overlap)

The only reason we had access to S01 and S02 (even though S02 was part of a test set) is because they are part of the training and validation data from 2020, which is public. On the other hand, testing MTMC on different datasets to be able to compare the results to other solutions was not feasible, since there is not a single dataset (other than CityFlow) that satisfies every criterion (e.g. vehicle objects, spatial information and raw footage of the cameras are available) to be a proper test dataset for *vehicle* MTMC [58]. Available MTMC datasets like MARS [77], DukeMTMC [50, 68], and NLPR_MCT [6] are for person tracking, moreover, some of them only provide trajectory information but not the original video.

In Table 9. we summarize the relevant results that are published in the literature. The table includes the reference of the paper, the year, tested scenarios, the number of cameras, and the IDF1 (which is the main performance indicator for MTMC in the challenge). At the bottom of the table, we listed the results got on this paper, and we included two variants (i) one without taking the overlap between the cameras into account, denoted by "w/o

overlap", and (ii) another when overlap was used through adding constraints, denoted by "w/ overlap". Since our system is flexible, we can control how to process the data, and therefore we can make a comparison between results got on overlapping cameras (S02 + S05) and results got on non-overlapping cameras (S06). For the overlapping case, the best result in the literature is 0.7069 IDF1, while our results are 0.9440 IDF1 on S01 and 0.9010 IDF1 on S02; for the non-overlapping case, the best result in the literature is 0.8486 IDF1, while our results are 0.9120 IDF1 on S01 and 0.8750 IDF1 on S02. Despite the average ~ 23% (overlap), and ~ 5% (non-overlap) increase in IDF1, we cannot clearly state that our approach outperforms the competitor methods because of the different number of cameras.

An important distinction between our system and the ones in the literature is that they were made to be successful on the CityFlow dataset, sometimes hard coding dataset-specific details, while the solution proposed in this paper can be used for any set of cameras, as long as the necessary configuration is provided.

The limitation of our system is that, even though the SCT is capable of real-time processing, the MTMC algorithm requires the full trajectory to be processed, which makes it suitable for offline processing. However, all the listed solutions in Table 9. share this limitation.

7 Conclusions

In this paper, we described the plan and the implementation of a complex vehicle tracking system that is capable to solve more tasks in parallel. Besides the main task (tracking the moving vehicles by using a single-phase object detector), the system solves the vehicle re-identification task and classifies the vehicles. Re-id networks (which determines the identifier, briefly ID for each box) and the convolutional networks classifying the individual features can run separately. Although IDs are unique within cameras, collisions may occur between cameras. The proposed system aggregates the trajectories at the end of the run and matches them between several cameras, which presumably contain the same vehicle. For this purpose, we developed an algorithm based on hierarchical clustering for multi-camera tracking that enforces constraints among the cameras. This proposed algorithm is the MCT#MAC (Multi-Camera Trajectory Matching Algorithm based on hierarchical clustering and Constraints) taking into account different constraints. When analyzing multi-camera recordings, defining camera synchronization and transitions between cameras can cause difficulties, but our method can also be used without this.

We developed an integrated vehicle tracking and classification system that is capable of (i) re-identification, (ii) determination of the static attributes of objects, i.e. the color and the type of the detected vehicles and (iii) solving single-camera tracking with post-processing. Additionally, our contribution (iv) is a new method (MCT#MAC) that combines single-camera trajectories (SCTs) into multi-camera trajectories (MCTs) using hierarchical agglomerative clustering (HAC) with time and space constraints. We developed (v) three variants of hierarchical clustering in MCT#MAC and we have shown that (vi) the total complexity of the algorithm is $\mathcal{O}(n^2 \log n)$, assuming a constant number of cameras. The advantage of our solutions is the flexibility, because the single-camera tracking can be used with more trackers (we evaluated with ByteTrack and DeepSORT), and MCT#MAC can be used with and without constraints (time and camera constraints). The system was evaluated on more datasets, and our experiments demonstrate that constraining HAC by manipulating the proximity matrix greatly improves the multi-camera IDF1 score, our system competes with state-of-the-art methods and it can achieve better results.

Table 10 Software and hardware details of the cloud resources

	Google Cloud	Kaggle Notebook
OS version	Ubuntu 20.04.3 LTS	Ubuntu 20.04.3 LTS
Kernel	5.15.120+x86_64	5.15.120+x86_64
CPU	Intel Xeon 2.2GHz 6c/12t	Intel Xeon 2.0Ghz 2c/4t
GPU	NVidia Tesla V100	NVidia Tesla P100
RAM	32 GB RAM	16 GB RAM
Average GPU usage	60%	53%

Appendix. Software and hardware details

Preparations, exploration of algorithms, and final experiments were all conducted using cloud resources. To gain more uptime, both Google Cloud and Kaggle Notebook were employed. Details about the software and hardware can be seen in Table10.

The implementation, installation guide, software environmental details, example code, and configuration details can all be accessed in our GitHub repository.[2]

Abbreviations AP: Average precision; BN: Batch normalization; BNNeck: Batch normalization neck; CFtrain: CityFlow training set; CFval: CityFlow validation set; CNN: Convolutional Neural Networks; CNNMTT: Multi-Target Tracking using CNN; EMRN: Efficient MultiResolution Network; FC: Fully connected; FCNN: Fully connected Neural Network; FP: False positive; FPN: Feature Pyramid Network; FPS: Frames per second; GBM: Gradient Boosting Machine; HAC: Hierarchical Agglomerative Clustering; IBN: Instance and batch normalization; ID: Identity; IDF1: Identity F1 (F1 in identity level); IDFN: Identity false negative (false negative in identity level); IDFP: Identity false positive (false positive in identity level); IDP: Identity precision (precision in identity level); IDR: Identity recall (recall in identity level); IDSW: Identity switches; IDTP: Identity true positive (true positive in identity level); IN: Instance normalization; IOU: Intersection over union; JDE: Joint Detection and Embedding; LR: Learning rate; mAP: Mean Average Precision; MCMT: Multi-Camera Multi-Target; MCT: Multi-camera trajectories; MCT#MAC: Multi-Camera Trajectory Matching Algorithm based on hierarchical clustering and Constraints; ML: Mostly lost; MOT: Multi-object tracking; MOTA: Mean object tracking accuracy; MT: Mostly tracked; MTMC: Multi-Target Multi-Camera; NN classifier: Neural Network classifier; POI: Person of Interest; PT: Partially Tracked; R-CNN: Region-based Convolutional Neural Networks; Re-id: Re-identity; ReLU: Rectified Linear Unit; ResNet: Residual Neural Network; RPN: Region proposal network; SCT: Single-camera trajectories; SGD: Stochastic gradient descent; SORT: Simple Online and Realtime Tracking; SSD: Single Shot MultiBox Detector; SST: Sphere Similarity Triplet; SVM: Support Vector Machine; TCL: Triplet-Centor Loss; TN: True negative; TP: True positive; VRIC: Vehicle Re-Identification in Context; VSLN: View-aware Sphere Learning Network; YOLO: You Only Look OnceFunding Open access funding provided by Budapest University of Technology and Economics.

Data availability There are no new datasets in this paper. The used datasets (CityFlow [58] – https://www.aicitychallenge.org/, VRIC [24] – https://qmul-vric.github.io, and VeRi-Wild [35] – https://github.com/PKU-IMRE/VERI-Wild) are available from citations written in the References.

Declarations

Ethical approval Not Applicable.

[2] https://github.com/regob/vehicle_mtmc.

Competing interests The authors, Dávid Papp, Regő Borsodi, and Gábor Szűcs declare that they have no conflict of interest.

References

1. Amosa TI, Sebastian P, Izhar LI, Ibrahim O, Ayinla Bahashwan AA, Bala A, Samaila YA (2023) Multi-camera multi-object tracking: a review of current trends and future advances. Neurocomputing, Volume 552, 126558. https://doi.org/10.1016/j.neucom.2023.126558
2. Avşar E, Avşar YÖ (2022) Moving vehicle detection and tracking at roundabouts using deep learning with trajectory union. Multimed Tools Appl 81:6653–6680. https://doi.org/10.1007/s11042-021-11804-0
3. Bergmann P, Meinhardt T, Leal-Taixe L (2019) Tracking without bells and whistles. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea (South), pp. 941–951. https://doi.org/10.1109/ICCV.2019.00103
4. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B (2016) Simple online and realtime tracking. In 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, pp. 3464–3468. https://doi.org/10.1109/ICIP.2016.7533003
5. Bochinski E, Eiselein V, Sikora T (2017) High-speed tracking-by-detection without using image information. In 14th IEEE international Conference on Advanced Video and Signal Based Surveillance (AVSS) Lecce, Italy, 2017, pp. 1–6. https://doi.org/10.1109/AVSS.2017.8078516
6. Cao L, Chen W, Chen X, Zheng S, Huang K (2015) An equalised global graphical model-based approach for multi-camera object tracking. arXiv preprint arXiv:1502.03532 , 8. https://doi.org/10.48550/arXiv.1502.03532
7. Chen C, Liu B, Wan S, Qiao P, Pei Q (2021) An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. IEEE Trans Intell Transp Syst 22(3):1840–1852. https://doi.org/10.1109/TITS.2020.3025687
8. Chen Y, Ke W, Lin H, Lam CT, Lv K, Sheng H, Xiong Z (2022) Local perspective based synthesis for vehicle re-identification: A transformation state adversarial method. J Vis Commun Image Represent 83:103432. https://doi.org/10.1016/j.jvcir.2021.103432
9. Dendorfer P, Osep A, Milan A, Schindler K, Cremers D, Reid I, Roth S, Leal-Taixé L (2021) Motchallenge: A benchmark for single-camera multiple target tracking. Int J Comput Vision 129:845–881. https://doi.org/10.1007/s11263-020-01393-0
10. Girshick R (2015) Fast R-CNN, IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440–1448. https://doi.org/10.1109/ICCV.2015.169
11. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 2014, pp. p587. https://doi.org/10.1109/CVPR.2014.81
12. Gong S, Xiang T (2011) Person Re-identification. In: Visual Analysis of Behaviour. Springer, London. https://doi.org/10.1007/978-0-85729-670-2_14
13. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN, IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980–2988. https://doi.org/10.1109/ICCV.2017.322
14. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: IEEE. pp. 770–778. https://doi.org/10.1109/CVPR.2016.90
15. He Z, Lei Y, Bai S, Wu W (2019) Multi-camera vehicle tracking with powerful visual features and spatial-temporal cue. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 15–20 June 2019, pp. 203–212

16. Hou Y, Du H, Zheng L (2019) A locality aware city-scale multi-camera vehicle tracking system. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 15–20 June 2019, pp. 167–174
17. Hsu HM, Cai J, Wang Y, Hwang JN, Kim KJ (2021) Multi-target multi-camera tracking of vehicles using metadata-aided re-id and trajectory-based camera link model. IEEE Trans Image Process 30:5198–5210. https://doi.org/10.1109/TIP.2021.3078124
18. Hsu HM, Huang TW, Wang G, Cai J, Lei Z, Hwang JN (2019) Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 15–20 June 2019, pp. 416–424
19. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 2261–2269. https://doi.org/10.1109/CVPR.2017.243
20. Javed O, Shafique K, Rasheed Z, Shah M (2008) Modeling inter-camera space–time and appearance relationships for tracking across non-overlapping views. Comput Vis Image Underst 109(2):146–162
21. Jia M, Cheng X, Lu S, Zhang J (2023) Learning disentangled representation implicitly via transformer for occluded person re-identification. IEEE Trans Multimedia 25:1294–1305. https://doi.org/10.1109/TMM.2022.3141267
22. Jocher G (2020) YOLOv5 by Ultralytics. https://github.com/ultralytics/yolov5. Accessed 1 Aug 2023
23. Kalman RE (1960) A new approach to linear filtering and prediction problems. J Basic Eng 82(1):35–45. https://doi.org/10.1115/1.3662552
24. Kanacı A, Zhu X, Gong S (2019) Vehicle re-identification in context. In Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9–12, 2018, Proceedings 40 (pp. 377–390). Springer International Publishing. https://doi.org/10.48550/arXiv.1809.09409
25. Khan SD, Ullah H (2019) A survey of advances in vision-based vehicle re-identification. Comput Vis Image Underst 182:50–63. https://doi.org/10.1016/j.cviu.2019.03.001
26. Khorramshahi P, Peri N, Chen JC, Chellappa R (2020) The devil is in the details: Self-supervised Attention for Vehicle Re-identification. In: Vedaldi A, Bischof H, Brox T, Frahm JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science, vol 12359. pp. 369–386. Springer, Cham. https://doi.org/10.1007/978-3-030-58568-6_22
27. Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Res Logist Q 2:83–97. https://doi.org/10.1002/nav.3800020109
28. Kuma R, Weill E, Aghdasi F, Sriram P (2019) Vehicle Re-identification: an Efficient Baseline Using Triplet Embedding, International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1–9, IEEE. https://doi.org/10.1109/IJCNN.2019.8852059
29. Kumar R, Weill E, Aghdasi F, Sriram P (2020) A strong and efficient baseline for vehicle re-identification using deep triplet embedding. J Artif Intell Soft Comput Res 10(1):27–45. https://doi.org/10.2478/jaiscr-2020-0003
30. Li F, Wang Z, Nie D, Zhang S, Jiang X, Zhao X, Hu P (2022) Multi-camera vehicle tracking system for AI City Challenge 2022. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 2022, pp. 3264-3272. https://doi.org/10.1109/CVPRW56347.2022.00369
31. Li P, Li G, Yan Z, Li Y, Lu M, Xu P, Gu Y, Bai B, Zhang Y, Chuxing D (2019) Spatio-temporal consistency and hierarchical matching for multi-target multi-camera vehicle tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Long Beach, CA, USA, 15–20 June 2019, pp. 222–230
32. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection, IEEE conference on computer vision and pattern recognition (CVPR), Honolulu, HI, USA, 2017, pp. 936–944. https://doi.org/10.1109/CVPR.2017.106
33. Liu C, Zhang Y, Luo H, Tang J, Chen W, Xu X, Wang F, Li H, Shen YD (2021) City-scale multi-camera vehicle tracking guided by crossroad zones, In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, 2021, pp. 4124–4132. https://doi.org/10.1109/CVPRW53098.2021.00466
34. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) SSD: Single shot multibox detector. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer Vision – ECCV 2016. Lecture Notes in Computer Science, vol 9905, pp. 21–37, Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

35. Lou Y, Bai Y, Liu J, Wang S, Duan L (2019) Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 3235–3243). https://doi.org/10.1109/CVPR.2019.00335
36. Luna E, SanMiguel JC, Martínez JM, Escudero-Vinolo M (2022) Online clustering-based multi-camera vehicle tracking in scenarios with overlapping FOVs. Multimed Tools Appl 81(5):7063–7083
37. Luo H, et al (2021) An empirical study of vehicle re-identification on the AI city challenge, IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, 2021, pp. 4090–4097. https://doi.org/10.1109/CVPRW53098.2021.00462
38. Luo H, Gu Y, Liao X, Lai S, Jiang W (2019) Bag of tricks and a strong baseline for deep person re-identification, IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 2019, pp. 1487–1495. https://doi.org/10.1109/CVPRW.2019.00190
39. Mahmoudi N, Ahadi SM, Rahmati M (2019) Multi-target tracking using CNN-based features: CNNMTT. Multimed Tools Appl 78:7077–7096. https://doi.org/10.1007/s11042-018-6467-6
40. Othmani M (2022) A vehicle detection and tracking method for traffic video based on faster R-CNN. Multimed Tools Appl 81:28347–28365. https://doi.org/10.1007/s11042-022-12715-4
41. Pan H, Wang Y, Szűcs G (2022) Work-traffic crashes and aberrant driving behaviors among full-time ride-hailing and taxi drivers: a comparative study. Transportation Letters. https://doi.org/10.1080/19427867.2022.2157075
42. Pan X, Luo P, Shi J, Tang X (2018) Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision – ECCV 2018. Lecture Notes in Computer Science, vol 11208. pp. 464–479, Springer, Cham. https://doi.org/10.1007/978-3-030-01225-0_29
43. Papp D, Borsodi R (2022) Determining Hybrid re-id features of vehicles in videos for transport analysis. Infocommunications J 4(1);17–23. https://doi.org/10.36244/ICJ.2022.1.3
44. Papp D, Lovas D, Szűcs G (2016) Object detection, classification, tracking and individual recognition for sea images and videos. In CLEF (Working Notes) pp. 525–533
45. Papp D, Mogyorósi F, Szűcs G (2017) Image matching for individual recognition with SIFT, RANSAC and MCL. In CLEF (Working Notes)
46. Qian Y, Yu L, Liu W, Hauptmann A (2020) Electricity: An efficient multi-camera vehicle tracking system for intelligent city. 2020 IEEE. In CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2511–2519
47. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91.
48. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans Pattern Anal Mach Intell 39:1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
49. Ristani E, Tomasi C (2018) Features for multi-target multi-camera tracking and re-identification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 2018, pp. 6036–6046. https://doi.org/10.1109/CVPR.2018.00632
50. Ristani E, Solera F, Zou R, Cucchiara R, Tomasi C (2016) Performance measures and a data set for multi-target, multi-camera tracking. In: Hua G, Jégou H. (eds) Computer Vision – ECCV 2016 Workshops. Lecture Notes in Computer Science 9914:17–35, Springer, Cham. https://doi.org/10.1007/978-3-319-48881-3_2
51. Shen F, Zhu J, Zhu X, Huang J, Zeng H, Lei Z, Cai C (2022) An efficient multiresolution network for vehicle reidentification. IEEE Internet Things J 9(11):9049–9059. https://doi.org/10.1109/JIOT.2021.3119525
52. Specker A, Florin L, Cormier M, Beyerer J (2022) Improving multi-target multi-camera tracking by track refinement and completion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3199–3209
53. Specker A, Stadler D, Florin L, Beyerer J (2021) An occlusion-aware multi-target multi-camera tracking system, IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, 2021, pp. 4168–4177. https://doi.org/10.1109/CVPRW53098.2021.00471

54. Szűcs G, Németh M (2021) Double-view matching network for few-shot learning to classify covid-19 in X-ray images. Infocommunications Journal 13(1):26–34
55. Szűcs G, Papp D, Lovas D (2015) SVM classification of moving objects tracked by Kalman filter and Hungarian method. In Working Notes of CLEF 2015 Conference, Toulouse, France. 10 pages
56. Tan M, Le QV (2019) EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, 9–15 June 2019, 6105–6114. http://proceedings.mlr.press/v97/tan19a.html
57. Tan M, Pang R, Le QV (2020) Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10781–10790
58. Tang Z, Naphade M, Liu MY, Yang X, Birchfield S, Wang S, Kumar R, Anastasiu D, Hwang JN (2019) 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, pp. 8797–8806
59. Tang Z, Wang G, Xiao H, Zheng A, Hwang JN (2018) Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 108–115
60. Uijlings JR, Van De Sande KE, Gevers T, Smeulders AW (2013) Selective Search for Object Recognition. Int J Comput Vision 104:154–171. https://doi.org/10.1007/s11263-013-0620-5
61. Wang CY, Bochkovskiy A, Liao HYM (2023) YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7464–7475
62. Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y, Liu D, Mu Y, Tan M, Wang X, Liu W, Xiao B (2020) Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell 43(10):3349–3364
63. Wang X, Jin Y, Li C, Cen Y, Li Y (2022) VSLN: View-aware sphere learning network for cross-view vehicle re-identification. Int J Intell Syst 37(10):6631–6651. https://doi.org/10.1002/int.22857
64. Wang Z, Zheng L, Liu Y, Li Y, Wang S (2020) Towards real-time multi-object tracking. In: Vedaldi A, Bischof H, Brox T, Frahm JM. (eds) Computer Vision – ECCV 2020. Lecture Notes in Computer Science, vol 12356. pp. 107–122, Springer, Cham. https://doi.org/10.1007/978-3-030-58621-8_7
65. Wen L, Du D, Cai Z, Lei Z, Chang MC, Qi H, Lim J, Yang MH, Lyu S (2020) UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. Comput Vis Image Underst 193:102907. https://doi.org/10.1016/j.cviu.2020.102907
66. Wojke N, Bewley A, Paulus D (2017) Simple online and realtime tracking with a deep association metric. In IEEE International Conference on Image Processing (ICIP) Beijing, China, 2017, pp. 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962
67. Wu M, Qian Y, Wang C, Yang M (2021) A multi-camera vehicle tracking system based on city-scale vehicle re-id and spatial-temporal information. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 2021, pp. 4072-4081. https://doi.org/10.1109/CVPRW53098.2021.00460
68. Wu Y, Lin Y, Dong X, Yan Y, Ouyang W, Yang Y (2018) Exploit the unknown gradually: One-shot video-based person re-identification by stepwise learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, UT, USA, pp. 5177–5186. https://doi.org/10.1109/CVPR.2018.00543
69. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 5987–5995. https://doi.org/10.1109/CVPR.2017.634
70. Yang X, Ye J, Lu J, Gong C, Jiang M, Lin X, Zhang W, Tan X, Li Y, Ye X, Ding E (2022) Box-grained reranking matching for multi-camera multi-target tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 2022, pp. 3095-3105. https://doi.org/10.1109/CVPRW56347.2022.00349
71. Yao H, Duan Z, Xie Z, Chen J, Wu X, Xu D, Gao Y (2022) 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, New Orleans, LA, USA, 2022, pp. 3309-3317. https://doi.org/10.1109/CVPRW56347.2022.00374
72. Ye J, et al (2021) A robust MTMC tracking system for AI-City challenge 2021. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Nashville, TN, USA, 2021, pp. 4039–4048. https://doi.org/10.1109/CVPRW53098.2021.00456
73. Ye M, Shen J, Lin G, Xiang T, Shao L, Hoi SC (2022) Deep learning for person re-identification: a survey and outlook. IEEE Trans Pattern Anal Mach Intell 44(6):2872–2893. https://doi.org/10.1109/TPAMI.2021.3054775

74. Yu F, Li W, Li Q, Liu Y, Shi X, Yan J (2016) POI: Multiple object tracking with high performance detection and appearance feature. In: Hua, G., Jégou, H. (eds) Computer Vision – ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science, vol 9914. pp. 36–42, Springer, Cham. https://doi.org/10.1007/978-3-319-48881-3_3
75. Zhang Y, Sun P, Jiang Y, Yu D, Weng F, Yuan Z, Luo P, Liu W, Wang X (2022) ByteTrack: Multi-object Tracking by Associating Every Detection Box. In: Avidan S, Brostow G, Cissé M, Farinella GM, Hassner T (eds) Computer Vision – ECCV 2022. ECCV 2022. Lecture Notes in Computer Science, vol 13682. pp. 1–21, Springer, Cham. https://doi.org/10.1007/978-3-031-20047-2_1
76. Zhang Y, Wang C, Wang X, Zeng W, Liu W (2021) FairMOT: On the fairness of detection and re-identification in multiple object tracking. Int J Comput Vision 129:3069–3087. https://doi.org/10.1007/s11263-021-01513-4
77. Zheng L, Bie Z, Sun Y, Wang J, Su C, Wang S, Tian Q (2016) MARS: A video benchmark for large-scale person re-identification. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VI 14, pp. 868–884. Springer International Publishing. https://doi.org/10.1007/978-3-319-46466-4_52
78. Zheng Z, Yang X, Yu Z, Zheng L, Yang Y, Kautz J (2019) Joint discriminative and generative learning for person re-identification. In proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, USA, 2019, pp. 2133-2142. https://doi.org/10.1109/CVPR.2019.00224
79. Zhou K, Yang Y, Cavallaro A, Xiang T (2021) Learning generalisable omni-scale representations for person re-identification. IEEE Trans Pattern Anal Mach Intell 44(9):5056–5069. https://doi.org/10.1109/TPAMI.2021.3069237
80. Zhou K, Yang Y, Qiao Y, Xiang T (2021) Domain generalization with mixstyle. ICLR (International Conference on Learning Representations). https://doi.org/10.48550/arXiv.2104.02008
81. Zhou X, Koltun V, Krähenbühl P (2020) Tracking objects as points. In: Vedaldi A, Bischof H, Brox T, Frahm JM. (eds) Computer Vision – ECCV 2020. Lecture Notes in Computer Science, vol 12349. Springer, Cham. https://doi.org/10.1007/978-3-030-58548-8_28
82. Zhu X, Luo Z, Fu P, Ji X (2020) VOC-ReID: Vehicle re-identification based on vehicle-orientation-camera. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 2020, pp. 2566-2573. https://doi.org/10.1109/CVPRW50498.2020.00309