IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 22, 2019, accepted June 3, 2019, date of publication June 10, 2019, date of current version June 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2921975

Real-Time Online Multi-Object Tracking in Compressed Domain

QIANKUN LIU[1,2], BIN LIU[1,2], YUE WU[3], WEIHAI LI[1,2], AND NENGHAI YU[1,2]
[1]School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China
[2]Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei 100864, China
[3]Alibaba Group, Hangzhou 311121, China

Corresponding author: Bin Liu (flowice@ustc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61371192, in part by the Key Laboratory Foundation of the Chinese Academy of Sciences under Grant CXJJ-17S044, and in part by the Fundamental Research Funds for the Central Universities under Grant WK2100330002.

ABSTRACT Recent online multi-object tracking (MOT) methods have achieved desirable tracking performance. However, the tracking speed of most existing methods is rather slow. Inspired from the fact that the adjacent frames are highly relevant and redundant, we divide the frames into key and non-key frames and track objects in the compressed domain. For the key frames, the RGB images are restored for detection and data association. To make data association more reliable, an appearance convolutional neural network (CNN) which can be jointly trained with the detector is proposed. For the non-key frames, the objects are directly propagated by a tracking CNN based on the motion information provided in the compressed domain. Compared with the state-of-the-art online MOT methods, our tracker is about 6× faster while maintaining a comparable tracking performance.

INDEX TERMS Compressed domain, multi-object tracking, online, real-time.

I. INTRODUCTION

Multi-Object Tracking (MOT) is an important computer vision task which aims to estimate the trajectories of interested objects and maintain their identities across frames. It has various applications that with real-time and online requirements, such as autonomous driving and robot navigation. However, real-time online MOT still remains a challenging task.

Driven by advances in object detection, tracking-by-detection has become a popular strategy for MOT. Most existing methods focus on designing a complicate approach to tackle MOT in a data association manner. These methods can be divided into two categories: offline and online methods. The offline methods [1]–[3] usually use future frames to track objects, which makes them impractical for casual applications. On the contrary, the online methods [4]–[14] track objects based on the past and current frames and have achieved desirable performance, as shown in Figure 1. However, detection and data association are performed frame-by-frame in these online methods to ensure a good tracking performance, which is time-consuming and makes them unable to be applied in real-time applications. In order to re-recognize objects when occlusion happens, appearance affinity is commonly used in data association process [4]–[7], [13], [15]. However, the appearance models utilized in these methods are independent from the detector and are therefore potentially sub-optimal.

In practical applications, videos are usually captured, stored and transmitted in the compressed domain. Implementations for some video tasks in the compressed domain are necessary and more suitable. Firstly, implementations in the compressed domain can take lower computational cost since not all frames need to be restored into RGB images (the RGB images in this paper denotes the regular colorful or gray images, which is used to distinguish them from the frames in the compressed domain). Secondly, the motion information is readily provided in the compressed domain which is helpful for video tasks. A few works, such as tracking [8], [17], video object detection [18] and action recognition [19], have been done in the compressed domain. Generally, there are two goals of the implementations in the compressed domain: (1) Feature propagation [18]. Features are only extracted from the key frames which are restored into RGB images, and then

The associate editor coordinating the review of this manuscript and approving it for publication was Yongqiang Zhao.

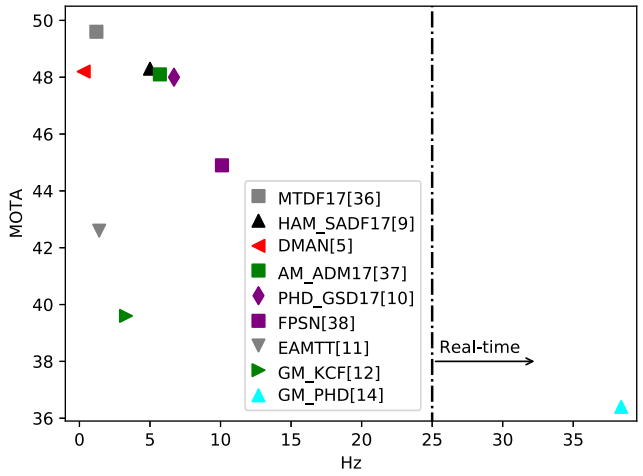

FIGURE 1. Performance-speed of some online trackers on MOT17 test split. Vertical axis: Multi-Object Tracking Accuracy (MOTA) [16]. Horizontal axis: The number of frames that the tacker can process in one second, i.e., the frequency. Better trackers lie in top-right of the figure.

propagated to the non-key frames. Computational cost can be saved since the frequency of feature extraction is reduced and the feature propagation is much more efficient than feature extraction. (2) Motion cues extraction [8], [17], [19]. The motion cues of objects are directly extracted from the motion information (motion vectors and residuals, more specifically) without access to the RGB images, which are further used to handle the task.

Existing compressed domain based tracking algorithms focus on the motion cues extraction from the pixel level [17] or the bounding-box level [8]. For the pixel level, each pixel that locates in the bounding-box is shifted separately based on the motion vectors (MVs), then the smallest axis-aligned rectangle that includes all the shifted pixels is selected as the new bounding-box. For the bounding-box level, MVs that locate in the bounding-box are averaged to get the displacement, then the bounding-box is shifted. However, the scale variation can not be handled.

In this paper, we focus on real-time online multi-object tracking. To this end, we propose an Online MOT Tracker in *Compressed Domain* (OTCD). Since the adjacent frames are highly relevant, it is redundant to perform detection and data association in all frames. Motivated by this, we divide the frames into key and non-key frames respectively. For the key frames, detection and data association are performed. An *Appearance CNN* (A-CNN) which shares features with the detector is designed to assist data association, and it can be jointly trained with the detector. Note that RGB images are restored for the key frames since both detection and appearance feature extraction need to be performed on the RGB images. For the non-key frames, objects are directly propagated based on the motion cues which are extracted from the MVs and residuals by a *Tracking CNN* (T-CNN). Owning to the sparsity of key frames and the share of features between A-CNN and detector, our tracker achieves a great boost in tracking speed with little performance degradation.

To sum up, our contributions are as follows:

- We develop an online unified MOT tracker to track objects in compressed domain for real-time applications.
- We propose an appearance CNN to assist data association. The joint training of appearance CNN and detector helps to further promote the performance of our method.
- We propose a tracking CNN to propagate objects through non-key frames while maintaining their identities without detection and data association, which accelerates our tracker greatly with little performance degradation.

The rest of this paper is organized as follows. Section II reviews the related work. Section III introduces the proposed tracker in detail and section IV represents experimental results. Finally, section V makes a conclusion on our work in this paper.

II. RELATED WORK

In this section, we provide a brief overview about the usage of appearance features in MOT and the works implemented in the compressed domain.

A. APPEARANCE FEATURES IN MOT

Appearance features can be used to improve tracking performance in crowded scenario where occlusion often happens. And various appearance features have been used in MOT, such as histogram of gradients [20], [21], color histogram [20] and integral channel features [15]. Recently, the powerful deep features extracted by CNN have been introduced to MOT [4]–[7], [13], [15]. Bae *et al.* [4] proposed a deep appearance learning method to learn a discriminative appearance model in an online manner. Kieritz *et al.* [15] designed an appearance model which was incrementally trained online for each object. Both [4] and [15] need to collect the training samples online, which is time-consuming. Chu *et al.* [7] utilized single object tracker for MOT based on appearance features, but the online learned target-specific CNN layers need to be preserved for each target. Instead of learning the appearance model online, some researchers trained an appearance model offline [5], [6], [9], [13], which can be used as a function to measure the affinity between different features while tracking online. Zhu *et al.* [5] trained a spatial attention CNN which can focus on matching patterns of input image patches. The works in [6], [9], [13] trained appearance models using person re-identification dataset and achieved great improvements.

The aforementioned methods separated data association from detection, and the utilized appearance models were isolated from the detector. Our work focuses on designing a compact appearance model, which shares features and can be jointly trained with the detector.

B. WORKS IN THE COMPRESSED DOMAIN

In order to use the motion information provided in the compressed domain freely, a few works [8], [17]–[19] have been done in the compressed domain. Ujiie *et al.* [8] interpolated

the bounding-boxes of objects in the bounding-box level for some frames to avoid detection and data association. Alvar *et al.* [17] constructed approximate bounding-box of the target object in the pixel level based on the bounding-box in previous frame for single object tracking, but each frame needs to be restored into RGB image for detection. The work in [18] fed MVs and residuals to a network to propagate the features across frames for object detection. However, the frames are processed in a batch manner which is inapplicable for online tasks. Wu *et al.* [19] recognized different actions in the compressed domain and achieved great performance. Nevertheless, the MVs and residuals are traced back to the reference frame and accumulated on the way, which augments the computational cost.

Among these methods, the work in [8] is the most related to our work. However, the work in [8] cannot handle the scale variations of bounding-boxes since MVs are simply used (by averaging MVs that locates in the bounding-boxes) to predict the displacements of objects. While our method utilizes a tracking CNN to predict the velocities of objects based on MVs and residuals, in which the scale variation of bounding-boxes are considered.

III. METHOD

Frames in a compressed video are divided into Group of Pictures (GOP), and there are three types of frame generally: I-frame (intra-coded frame), P-frame (predictive frame) and B-frame (bi-directional frame). Among these three types of frame, I-frame can be treated as a regular RGB image, while P-frame and B-frame are encoded with MVs and residuals. The main difference between P-frame and B-frame is that P-frame is encoded in a predictive manner, while B-frame is encoded in a bi-directional manner. We share the same assumption with [8], [17]–[19] that the compressed video only contains I-frame and P-frame for simplicity. The reason is that the B-frames, which are encoded in a bi-directional manner, require special handling since we focus on online tracking, and we leave this for the future work. In this paper, we track objects in raw MPEG-4 videos which have an I-frame before every 11 P-frames. However, the proposed method is applicable for different compression techniques, such as MPEG-2 [22] and H.264 [23]. The reason is that different compression techniques usually use motion vectors and residuals for frame compression, thus the frames in the videos can be easily divided into key and non-key frames.

A. OVERVIEW

The overview of the proposed tracker OTCD is shown in Figure 2. The frames are divided into key and non-key frames in OTCD. Suppose there is a key frame every K consecutive frames. Since MVs and residuals are required for the non-key frames and there are no MVs or residuals for I-frames in the compressed domain, I-frames are always regarded as the key-frames, which means K should be a factor of GOP size.

Let $O^t = \{o_i^t\}_{i=1}^{I_t}$ and $D^t = \{d_j^t\}_{j=1}^{J_t}$ denote the sets of objects and detections in frame t respectively. Note that D^t is defined for the key frames only. For a key frame at time t, the RGB image is restored and fed into a detector, which produces a set of detections D^t. The objects in O^{t-1} from last frame are associated with the detections in D^t. The data association is solved by Hungarian algorithm based on the Intersection-over-Union (IoU) between bounding-boxes and the appearance affinity obtained by A-CNN. After then the birth and death of objects are managed. For each non-key frame, the corresponding MVs and residuals are fed into T-CNN to propagate objects from the previous frame to current frame.

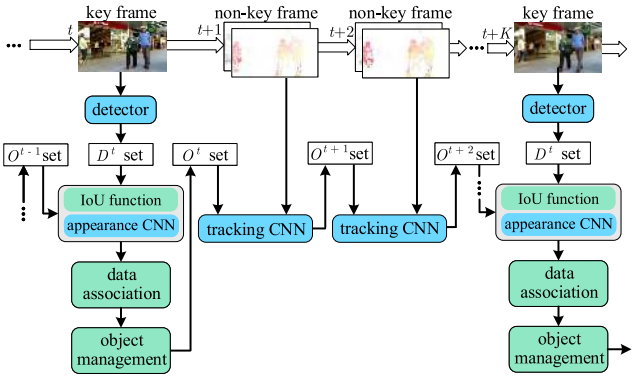

FIGURE 2. Overview of the proposed tracker. MVs are plotted in HSV color space. O^t is the set of objects in frame t, D^t is the set of detections in key frame t. Detection, data association and object management are only performed in key frames. Objects are propagated through non-key frames by tracking CNN while maintaining their identities.

Before introducing the method in detail, we first introduce the representations of objects and detections. The j-th detection in D^t is denoted by a tuple $d_j^t = (b_j^t, f_j^t)$, where $b_j^t = (x_j^t, y_j^t, w_j^t, h_j^t)$ is the bounding-box represented by the center coordinate, width and height. $f_j^t \in \mathbb{R}^{m\times m\times c}$ is the appearance feature cropped by RoIAlign [24] from the feature map provided by detector, where m and c are the spatial size and the number of channels respectively. As for objects, three states $\{Tentative, Confirmed, Deleted\}$ are defined to handle the birth and death of objects, which are denoted by $\{s_T, s_C, s_D\}$ for simplicity. The i-th object in O^t is denoted as $o_i^t = (b_i^t, s_i^t, F_i)$, where $b_i^t = (x_i^t, y_i^t, w_i^t, h_i^t)$ is the bounding-box, $s_i^t \in \{s_T, s_C, s_D\}$ is the state. And $F_i = \{f_i^{t-\tau}\}_{\tau=0}^{l_f-1}$ is the set of appearance features collected in the history, where l_f is the maximum number of appearance features.

B. TRACKING IN KEY FRAMES

The tracking in key frames follows the footprint of per-frame approaches, including detection, data association and object management.

1) DETECTOR

The detector is responsible for the detection of interested objects (pedestrian, particularly) and feature extraction. The R-FCN [25] with ResNet-101 [26] is used in our work directly since detection is beyond the scope of this paper.

We take pedestrian as the foreground and others as the background for detection. The appearance features for detections are cropped from the feature map provided by the last convolutional layer on the conv4 stage of the backbone of detector.

2) APPEARANCE CNN

Given the appearance feature f_j^t of the detection d_j^t and one appearance feature $f_i^{t-\tau} \in F_i$ of the object o_i^{t-1}, the probability $p_{i,j}^{\tau}$ of these two features belonging to the same object is used as the affinity between these two features:

$$p_{i,j}^{\tau} = \mathcal{N}_A(f_i^{t-\tau}, f_j^t), \tag{1}$$

where $\mathcal{N}_A(\cdot,\cdot)$ denotes A-CNN. As shown in Figure 3, A-CNN is a binary classifier in fact. Note that the appearance features f_j^t and $f_i^{t-\tau}$ are cropped from the feature map provided by the conv4 stage of the backbone of detector, and no feature extraction (except the three convolutional layers) is performed within A-CNN, which means A-CNN shares features with the detector. The superscripts are omitted for simplicity in the following.

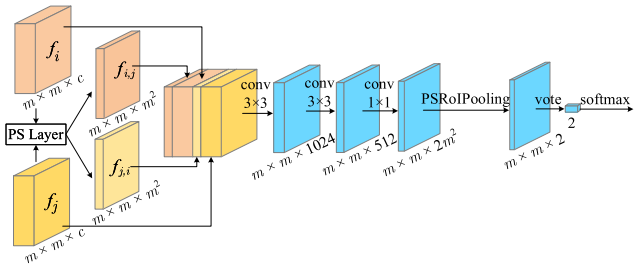

FIGURE 3. Architecture of A-CNN. Each convolution layer is followed by a ReLU function. A position sensitive (PS) layer is introduced to exploit the position information between input features.

a: POSITION-SENSITIVE LAYER

Given two features $f_i, f_j \in \mathbb{R}^{m\times m\times c}$, we try to obtain the affinity between these two features. The L^2-normalization is first applied to f_i and f_j along the channel dimension, which produces the corresponding normalized features f_i' and f_j'. Then two position-sensitive feature maps $f_{i,j}, f_{j,i} \in \mathbb{R}^{m\times m\times m^2}$ are produced respectively:

$$f_{i,j} = f_i' \otimes \mathcal{T}(f_j'), \quad f_{j,i} = f_j' \otimes \mathcal{T}(f_i'), \tag{2}$$

where $\otimes$ denotes the matrix multiplication, and $\mathcal{T}(\cdot)$ is the function that reshapes and transposes a 3-D feature map in $\mathbb{R}^{m\times m\times c}$ to a 2-D feature map in $\mathbb{R}^{c\times m^2}$. Each column in $\mathcal{T}(f_i')$ corresponds to a feature vector that locates in one spatial position in f_i'. Finally, the four feature maps $f_{i,j}$, f_i, $f_{j,i}$ and f_j are concatenated together. Despite the values in $f_{i,j}$ and $f_{j,i}$ are the same, the distributions of them are different. We keep both of them to preserve more position information.

The intuition of the Position-Sensitive (PS) layer is that we assume the features extracted from the same patches in different RGB images should be the same, but the patches may not well aligned due to the inaccurate detection, occlusion and pose change. The corresponding features in f_i' and f_j' may locate in different spatial positions. Hence, it is necessary to compare the feature vector from one spatial position in f_i' with the feature vectors from all spatial positions in f_j', which produces a single channel feature map in $f_{j,i}$.

b: TRAINING OF A-CNN

During the training process, each training sample contains two appearance features cropped from the feature map provided by the detector. The corresponding label is set to 0 (these two appearance features belong to different objects) or 1 (these two appearance features belong to the same object).

A-CNN is trained by the cross-entropy loss. Let L_A and L_D be the loss of A-CNN and detector respectively. Then A-CNN and the detector can be jointly trained via a multi-task loss $L = L_D + \lambda L_A$, where λ is the weight to balance the loss.

3) DATA ASSOCIATION

Given the set D^t of detections in key frame t, and the set O^{t-1} of objects in the previous non-key frame $t-1$, the data association process is divided into two steps.

Step 1: assign the detections in D^t to confirmed objects based on the IoU cost between bounding-boxes. Let $c_{i,j}^{iou}$ be the IoU cost between object o_i^{t-1} and detection d_j^t

$$c_{i,j}^{iou} = 1 - IoU(b_i^{t-1}, b_j^t), \tag{3}$$

and they will not be associated with each other if $c_{i,j}^{iou}$ is greater than a threshold τ_{iou}.

Step 2: assign the unmatched detections to objects in tentative state as well as those unmatched objects in step 1 based on the appearance cost. Let $c_{i,j}^{app}$ be the appearance cost between object o_i^{t-1} and detection d_j^t

$$c_{i,j}^{app} = 1 - \max_{\tau\in\{1,2,\ldots,|F_i|\}} p_{i,j}^{\tau}, \tag{4}$$

where $p_{i,j}^{\tau}$ is the appearance affinity obtained by A-CNN. And they will not be associated if $c_{i,j}^{app}$ is greater than a threshold τ_{app}.

Suppose the detection d_j^t is assigned to the object o_i^{t-1}, the bounding-box of the i-th object in frame t is succeeded from the bounding-box of d_j^t. The detection's appearance feature f_j^t is added to the feature set F_i. The oldest feature will be abandoned if there are more than l_f features in F_i.

4) OBJECT MANAGEMENT

The objects are managed by transforming their states between the pre-defined three states $\{s_T, s_C, s_D\}$. Particularly:

1) An unmatched detection is initialized as a tentative object, and it will be confirmed if its detection confidence is larger than a threshold $c_{s_T\to s_C}$.
2) A confirmed object is transformed to tentative state if it has not been associated with any detections for more than $l_{s_C\to s_T}$ consecutive key frames.

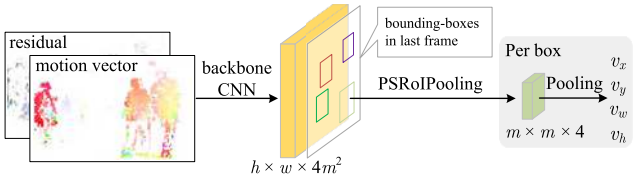

FIGURE 4. The Velocity prediction. The velocity of each un-deleted object is estimated by PSRoIPooling based on the bounding-box in last frame. The number of bounding-boxes in current frame is the same with that in last frame.

3) A tentative object will be confirmed if it has been associated with a detection for more than $l_{s_T \to s_C}$ consecutive key frames.
4) A tentative object will be deleted if it has not been associated with any detections for more than $l_{s_T \to s_D}$ consecutive key frames.
5) A deleted object remains at s_D forever.

C. TRACKING IN NON-KEY FRAMES

The tracking in non-key frames is much more straightforward. The objects are directly propagated by T-CNN while maintaining their identities. The appearance features and state of each object are not changed during propagation since the RGB images are not restored for detection and data association. Let $\hat{v}_i^t = (\hat{v}_{i,x}^t, \hat{v}_{i,y}^t, \hat{v}_{i,w}^t, \hat{v}_{i,h}^t)$ be the predicted velocity of the i-th object in frame t based on the bounding-box in frame $t-1$:

$$\hat{v}_i^t = \mathcal{V}(b_i^{t-1}, \mathcal{N}_T(\mathfrak{f}^t)), \tag{5}$$

where $\mathcal{N}_T(\cdot)$ is the backbone CNN of T-CNN, and $\mathfrak{f}^t$ represents the input data for the non-key frame t. $\mathcal{V}(\cdot,\cdot)$ is the velocity prediction function. As shown in Figure 4, $\mathcal{V}(\cdot,\cdot)$ is implemented by position-sensitive region-of-interest pooling layer (PSRoIPooling) proposed in R-FCN [25]. Then the bounding-boxes in frame t can be predicted easily by $b_i^t = \mathcal{B}(\hat{v}_i^t, b_i^{t-1})$, where $\mathcal{B}(\cdot,\cdot)$ is the bounding-box prediction function that defined as

$$\begin{cases} x_i^t = w_i^{t-1}\hat{v}_{i,x}^t + x_i^{t-1}, \\ y_i^t = h_i^{t-1}\hat{v}_{i,y}^t + y_i^{t-1}, \\ w_i^t = w_i^{t-1}\exp(\hat{v}_{i,w}^{t-1}), \\ h_i^t = h_i^{t-1}\exp(\hat{v}_{i,h}^{t-1}). \end{cases} \tag{6}$$

Obviously, the identity is maintained for each object during the propagation process.

a: BACKBONE CNN

The network used in T-CNN is much smaller than the network in detector, since the MVs and residuals only store the *changes* between two frames. Besides, a smaller network can reduce the computational cost. The backbone CNN is modified from ResNet-18 [26]. Particularly, the last average pooling layer and fully connection layer are removed. As a common practice [25], the effective stride of ResNet-18 is reduced from 32 pixels to 16 pixels, which increases the resolution of feature maps. Then we can get three types of the modified ResNet-18 by changing the number of input channels in the first convolutional layer to 2, 3 and 5, which are denoted as ResNet$_2$-18, ResNet$_3$-18 and ResNet$_5$-18, respectively.

In order to explore the tracking ability of T-CNN, four T-CNNs are designed, as shown in Figure 5:

- T-CNN$_{mv}$: only MVs are used to predict the velocities.
- T-CNN$_{res}$: only residuals are used to predicted the velocities.
- T-CNN$_{mv|res}$: MVs and residuals are both used, but they are concatenated together firstly to be fed into T-CNN.
- T-CNN$_{mv||res}$: MVs and residuals are both used, and they are fed into their corresponding branches. Then the outputs of these two branches are concatenated together.

For all prototypes of T-CNN, the 1×1 convolutional layer (followed by a ReLU function) is used to produce a feature map with $4m^2$ channels.

b: TRAINING OF T-CNN

The training of T-CNN is independent of the training of A-CNN and detector. The reason is that detection and appearance feature extraction need to be performed on RGB images, while T-CNN needs motion vectors and residuals to predict the velocities of objects. Given the ground-truth bounding-boxes of the i-th object in frame $t-1$ and t, the ground-truth velocity $v_i^t = (v_{i,x}^t, v_{i,y}^t, v_{i,w}^t, v_{i,h}^t)$ can be computed by $v_i^t = \mathcal{B}^{-1}(b_i^{t-1}, b_i^t)$, where $\mathcal{B}^{-1}(\cdot,\cdot)$ is the inverse function of $\mathcal{B}(\cdot,\cdot)$. The loss of T-CNN can be computed by

$$L_T = \frac{1}{I}\sum_{i=1}^{I}\sum_{u\in\{x,y,w,h\}} smooth_{L_1}(v_{i,u}^t - \hat{v}_{i,u}^t), \tag{7}$$

in which

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise, \end{cases} \tag{8}$$

is the function defined in [27], and I is the number of objects that appear in frames t and frame $t-1$. The objects that only appear in one frame will not contribute to the training process.

D. TIME CONSUMPTION ANALYSIS

Let T_{det}, T_{ass}, T_{man} and T_{pro} be the time consumption of detection, data association, object management and object propagation in each frame. Compared to the per-frame approaches, the speedup factor s of our tracker depends on the sparsity of key frames:

$$s = \frac{K(T_{det} + T_{ass} + T_{man})}{T_{det} + T_{ass} + T_{man} + (K-1)T_{pro}}. \tag{9}$$

Generally, $T_{man} \ll T_{det}$ and $T_{man} \ll T_{ass}$. In our implementation, $T_{pro} \approx \frac{T_{det}+T_{ass}}{10}$. Then s is about

$$s \approx \frac{10K}{K+9}. \tag{10}$$

For example, our tracker is $2.5\times$ faster approximately when $K = 3$. The tracking method is shown in Algorithm 1.

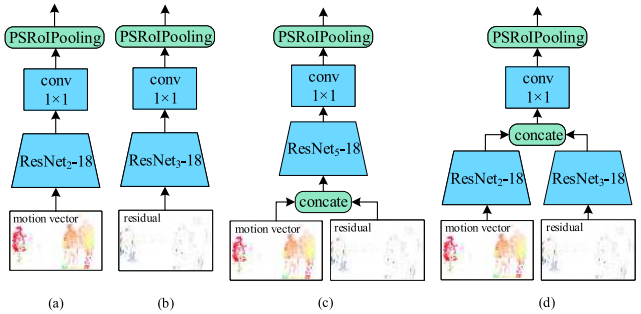

FIGURE 5. Four prototypes of T-CNN. (a): T-CNN$_{mv}$, only MVs are fed into T-CNN. (b): T-CNN$_{res}$, only residuals are fed into T-CNN. (C): T-CNN$_{mv|res}$, MVs and residuals are concatenated together to be fed into T-CNN. (d): T-CNN$_{mv||res}$, MVs and residuals are fed into ResNet$_2$-18 and resNet$_3$-18 respectively, then the outputs of these two CNNs are concatenated together. For all designs of T-CNN, the corresponding 1 × 1 convolutional layer (followed by a ReLU function) is used to produce a feature map with $4m^2$ channels.

IV. EXPERIMENTS

The proposed MOT tracker OTCD is implemented based on PyTorch library without optimization. Evaluation is on a workstation with 2.6 GHz CPU and Nvidia TITAN Xp GPU.

A. DATASETS

We evaluate the proposed tracker on Citypersons [28], 2DMOT2015 [29], MOT16 [30] and MOT17 [30]. The sequences in MOT16 are the same with those in MOT17 but are provided with a less accurate ground-truth. Each sequence in 2DMOT2015 and MOT17 is compressed into a MPEG-4 video, and all images in Citypersons are compressed into a MPEG-4 video. All data used for training and testing is loaded from the compressed domain with the tool provided by [19]. The loaded MVs and residuals can be treated as special *images* (2 and 3 channels respectively) which have the same resolution with the video.

Sequences in 2DMOT2015 and MOT17 are divided into three sets. *Testing set*: the sequences in MOT17 test split. *Validation set*: MOT17-09 and MOT17-10. *Training set*: the rest sequences in MOT17 as well as those sequences in 2DMOT2015 train split but not included in MOT17.

B. SETTINGS

The variable m used in A-CNN and T-CNN is set to 7. The detections with confidence less than 0.95 are abandoned. The detection confidence threshold $c_{s_T \to s_C}$ is set to 0.99. The number of consecutive key frames $l_{s_T \to s_C}$, $l_{s_C \to s_T}$, $l_{s_T \to s_D}$ are set to 3, 2, 10 respectively. And the number of historical appearance features l_f is set to 24. The thresholds τ_{iou} and τ_{app} are set to 0.3 and 0.25 respectively.

Citypersons and training set are used to train R-FCN and A-CNN. The training samples for A-CNN are generated during training process. Particularly, for each ground-truth box in one image, two positive and one negative samples are collected with ≥ 0.7 and ≤ 0.3 IoU overlap ratios with this ground-truth box. We also randomly choose a ground-truth box belonging to other objects as another negative sample. The separate training of R-FCN and A-CNN are conducted with initial learning rate 10^{-3} and 10^{-4} respectively. The joint training of R-FCN and A-CNN is divided into 3 phases. Phase 1: train R-FCN for 15 epochs with initial learning rate 10^{-3}. Phase 2: train A-CNN for 5 epochs with initial learning rate 10^{-4}. Phase 3: train A-CNN and R-FCN jointly with initial learning rate 10^{-4}. Here, we set λ to 1. T-CNN is trained on training set with initial learning rate 10^{-4}. During all training process, the batch size is set to 2 and the learning rate decades every 8 epoches with exponential decay rate 0.1. All models are optimized by stochastic gradient descent until they converge.

Algorithm 1 OTCD Tracker

Input: a MPEG-4 video v with T frames, key frame scheduler K

Output: Trajectories of objects $B = \{\{b_i^t, id_i\}_{i=1}^{I^t}\}_{t=1}^{T}$

Initialization: $B \leftarrow \emptyset$, $O^0 \leftarrow \emptyset$, $t \leftarrow 1$
while $t \leq T$ **do**
  /⋆ track objects online ⋆/
  $F^t \leftarrow$ Load(v, t) // load a frame
  if $(t-1) \bmod K = 0$ **then** // key frame
    $F^t \leftarrow$ RestoreToRGB(F^t)
    $D^t \leftarrow$ R-fcn(F^t)
    Association(D^t, O^{t-1})
    $O^t \leftarrow$ Management(D^t, O^{t-1})
  else // non-key frame
    $O^t \leftarrow$ Propagate(O^{t-1}, F^t)
  end if
  /⋆ store the trajectories ⋆/
  $B^t \leftarrow \emptyset$
  for each $o_i^t \in O^t$ **do**
    if $s_i^t = s_C$ **then**
      $B^t \leftarrow B^t \cup \{b_i^t, id_i\}$
    end if
  end for
  $B \leftarrow B \cup B^t$
end while
return B

C. METRICS

1) TRACKERS

We choose the following metrics to evaluate different trackers: Multi-Object Tracking Accuracy (MOTA) [16], Multi-Object Tracking Precision (MOTP) [16], how often an object is identified by the same ID (IDF1) [31], Mostly Tracked objects (MT), Mostly Lost objects (ML), number of False Positives (FP), number of False Negatives (FN), number of Identity Switches (IDS) [32], number of Fragments (Frag), and running speed (Hz).

TABLE 1. Tracking performance on validation set with different settings for data association. Values in bold highlight the best results.

	OTCD				
Settings / Metrics	A-CNN$^{-}_{PS}$	A-CNN	IoU	IoU+A-CNN	IoU → A-CNN
IDS↓	555	348	195	167	**141**
MOTA↑	32.0%	33.2%	35.0%	35.4%	**35.5%**
Hz↑	3.2	2.3	**16.5**	2.1	15.7

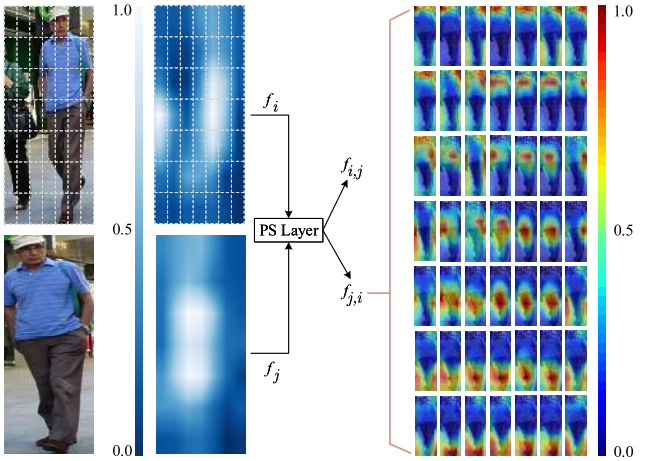

FIGURE 6. Qualitative results on PS layer. The feature maps are resized to the size of image patches. Left: Two image patches (and their appearance features) need to be compared with. The pixel values plotted in the feature map are proportional to the L_2-norm of corresponding feature vectors along channel dimension. Right: two (m^2-channel) feature maps $f_{i,j}$ and $f_{j,i}$ that obtained by PS layer. Only $f_{j,i}$ is presented in detail. The top image patch is divided into $m \times m$ bins ($m = 7$ in our experiments), corresponding to the spatial size of the features that input to A-CNN. Each bin is compared with the bottom image patch, producing a single channel feature map in $f_{j,i}$. The similar two patterns are with a high similarity score.

2) DETECTOR

Except FP and FN, we choose additional metrics to evaluate different detectors: Recall (Rcll), Precision (Prcn), Multiple Object Detection Accuracy (MODA) [33].

All metrics are evaluated by the toolkit provided by MOTChallenge benchmark [29], [30].

D. COMPONENTS ANALYSES

1) APPEARANCE CNN

We first present the features produced by the PS layer in Figure 6. The PS layer actually divides one image patch into $m \times m$ bins, and compares each bin with another image patch. A high similarity score will be produced if the compared image patterns are similar.

In order to demonstrate the effectiveness of the two steps data association procedure introduced in section III-B.3 (denoted as IoU→A-CNN), an one step data association procedure that simultaneously take IoU cost and appearance cost into consideration are also conducted (denoted as IoU+A-CNN). For the one step data association procedure, we use $c_{i,j} = \alpha c^{iou}_{i,j} + (1-\alpha)c^{app}_{i,j}$ as the cost between object o^{t-1}_i and detection d^t_j, and they will not be associated with each other if $c_{i,j}$ is greater than the threshold $\tau = \alpha\tau^{iou} + (1-\alpha)\tau^{app}$. Several experiments are conducted by varying α from 0.1 to 0.9 with the interval 0.1, and the best MOTA is achieved on validation set when $\alpha = 0.5$, which is our default setting for the one step data association procedure. Note that $\alpha = 1$ and $\alpha = 0$ are two special cases of IoU+A-CNN which are denoted as IoU and A-CNN, respectively. An A-CNN without PS layer (A-CNN$^{-}_{PS}$) is also trained to further demonstrate the effectiveness of the PS layer in A-CNN. The results are shown in Table 1.

When the appearance cost is only used, MOTA is improved by 1.2% and IDS is greatly reduced by 37.3% with the help of the PS layer at the cost of tracking speed dropped from 3.2 Hz to 2.3 Hz. Favorable MOTA and IDS can be obtained when the IoU cost is only used. This is due to the fact that the bounding boxes of one object in the adjacent frames may be much closer, and the IoU cost is sufficient enough to associate them with each other. The best tracking speed is also possessed by the case where the IoU cost is only used. This is reasonable since appearance cost is no need to be computed, which is more computational expensive than IoU cost. However, MOTA is still improved by 0.5% and IDS is further reduced by 27.7% when A-CNN is used in IoU→A-CNN with the little price of Hz dropped from 16.5 to 15.7.

Though MOTA is almost the same in IoU+A-CNN and IoU→A-CNN, a better IDS and Hz are achieved by IoU→A-CNN. Here is the explanation: (1) Hz. Both IoU cost and appearance cost need to be computed for each object-detection pair in IoU+A-CNN. While in IoU→A-CNN, appearance cost (which is computational expensive) only needs to be computed for a small proportion of object-detection pairs. (2) IDS and MOTA. IoU cost and appearance cost may not be always both reliable. Simultaneously consideration of IoU cost and appearance cost will make them influence each other.

A case of occlusion is also shown in Figure 7. When data association procedure is conducted based on the IoU cost only, the tracker fails to re-recognize the occluded pedestrian. However, when the data association procedure is conducted based on both IoU and appearance cost as described in section III-B.3, the tracker can re-recognize the occluded pedestrian successfully, which demonstrates the effectiveness of the proposed A-CNN and the data association method.

2) JOINT TRAINING OF APPEARANCE CNN AND DETECTOR

The effectiveness of joint training of A-CNN and detector is shown in Table 2. In terms of detection, all metrics are

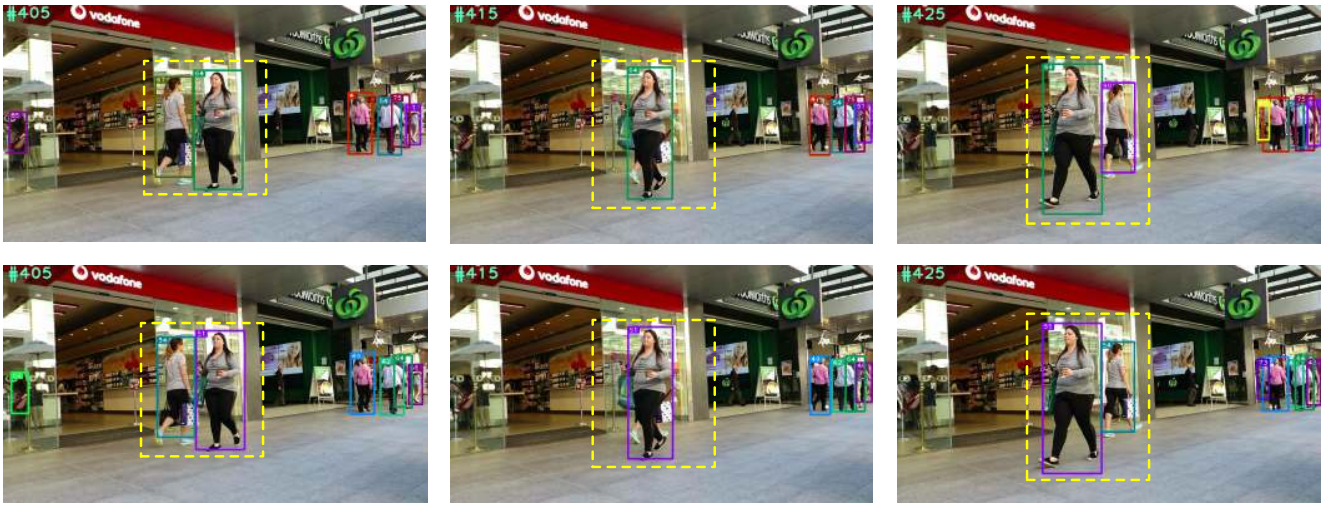

FIGURE 7. A case of occlusion in MOT17-09. Please pay attention to the two pedestrians that locates in yellow dashed boxes. The color of bounding-boxes and the numbers locate in top-left corner of the bounding-boxes denote the identities of corresponding pedestrian. From left to right: Frame 405 (before occlusion occurs), frame 415 (when occlusion occurs), frame 425 (after occlusion occurs). Top row: Data association is conducted based on the IoU cost only. The tracker fails to re-recognize the occluded pedestrian. Bottom row: Data association is conducted based on the IoU cost and the appearance cost as described in section III-B.3. The tracker re-recognizes the occluded pedestrian successfully.

TABLE 2. The effect of joint training of A-CNN and detector on detection and tracking performance. Tested on validation set.

	joint training	Rcll↑	Prcn↑	FP↓	FN↓	MODA↑
R-FCN		55.1%	85.4%	1165	5537	45.6%
	✓	**55.8%**	**88.0%**	**934**	**5448**	**48.2%**

	joint training	MOTA↑	IDF1↑	FP↓	FN↓	Frag↓
OTCD		35.5	22.9%	1437	**9759**	362
	✓	**38.4%**	**25.1%**	**911**	9770	**321**

improved. Particularly, FP is greatly reduced about 19.8%, Prcn and MODA are both improved by more than 2%. In terms of tracking, FP is greatly reduced about 36.6%, which leads to a higher overall tracking performance MOTA. Furthermore, IDF1 is improved by 2.2%, which means the tracker can recognize an object with the same ID more often.

3) TRACKING CNN

We use T-CNN to speedup OTCD by varying $K \in \{1, 2, 3, 4, 6\}$. The four prototypes of T-CNN are firstly evaluated, as shown in Figure 8 (a). The subscript of OTCD denotes the corresponding prototype of T-CNN. Among our four trackers, OTCD_{res} performs the worst since no motion information is provided. Compared with $\text{OTCD}_{mv|res}$ and $\text{OTCD}_{mv||res}$, OTCD_{mv} is less stable, we argue this to the absence of residuals. Based on the above analysis, we can find that MVs and residuals complement each other. Compared with $\text{OTCD}_{mv||res}$, $\text{OTCD}_{mv|res}$ has a superior tracking capability. What's more, the amount of parameters in $\text{T-CNN}_{mv|res}$ is much smaller than that in $\text{T-CNN}_{mv||res}$. We choose $\text{OTCD}_{mv|res}$ as our default tracker.

To demonstrate the effectiveness of the proposed T-CNN, we compare our method with some other trackers, including DeepSORT [6], SORT [34] and IOU [39]. The results are summarized in Figure 8 (b). The bounding-boxes of objects in the non-key frames are predicted by Kalman filter in DeepSORT and SORT, while they are copied from previous frame in IOU. For all trackers, the detection time (about 60 ms) is considered and only detections in the key frames are provided for a fair comparison. However, the time consumption in the non-key frames are not considered for DeepSORT, SORT and IOU. As we can see, the tracking speed is limited due to the detection time consumption when $K = 1$. But all trackers achieve significant speedup with the descent tracking accuracy drop when $K > 1$. Thanks to the powerful tracking capability of T-CNN, our tracker possesses the slowest performance decline when compared with other trackers. For example, $\text{OTCD}_{mv|res}$ is accelerated from 15.8 Hz to 46.0 Hz at the cost of accuracy drop from 38.4% to 37.8%, while DeepSORT is accelerated from 8.5 Hz to 34.0 Hz at the cost of accuracy drop from 38.4% to 33.4%.

The effective performance gains brought by T-CNN does not hold when K increases. Reasons are as follows: (1) Tracking accuracy. The birth and death of objects are not handled in the non-key frames, but objects may disappear or reappear during these frames, which leads to a poorer tracking performance when K increases. (2) Tracking speed. Since we track objects without detection or data association on the non-key frames, the bounding-boxes of tracked objects may be imperfect, which results in more imperfect bounding-boxes in the following non-key frames. So more confirmed objects will not be associated with detections based on the IoU cost when the next one key frame arrives. Hence, more detections need to be assigned to objects based on the appearance cost, which

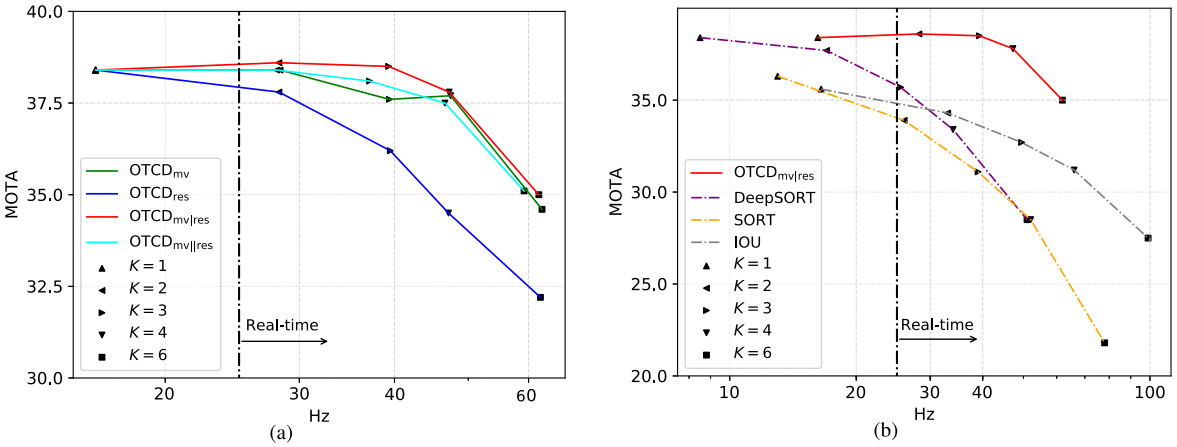

FIGURE 8. Performance-speed tradeoff with different K. The subscript of OTCD denotes the corresponding prototype of T-CNN. (a) Tracking results of OTCD with different prototypes of T-CNN. (b) Tracking results of our default tracker and other trackers. Tested on validation set.

TABLE 3. Results of different trackers on MOTChallenge benchmark. The subscript of OTCD is the value of K. Values in bold highlight the best results and Hz in blue highlight the real-time trackers. Hz marked by § means detection time is considered. Trackers marked by ⋆ and † use the detections provided by [13] and ourselves, respectively. While other trackers use the public detections.

benchmark	trackers	MOTA↑	MOTP↑	IDF1↑	MT↑	ML↓	FP↓	FN↓	IDS↓	Frag↓	Hz↑
MOT16	DeepSORT* [6]	**61.4%**	79.1%	**62.2%**	32.8%	18.2%	12852	56668	**781**	2008	17.4
	SORT* [34]	59.8%	**79.6%**	53.8%	25.4%	22.7%	**8698**	63245	1423	1835	**59.5**
	MVint(LinearK)* [8]	55.0%	76.7%	-	20.4%	24.5%	15766	65297	1024	**1594**	16.9
	OTCD$^{\star}_1$	60.2%	79.1%	55.1%	**35.8%**	**16.1%**	14724	55244	2670	2310	12.6
	OTCD$^{\star}_3$	59.7%	78.5%	56.8%	35.4%	16.9%	16882	**55156**	1456	1971	28.6
	OTCD$^{\dagger}_1$	46.8%	73.6%	45.0%	15.9%	43.0%	12914	82777	1236	2666	13.6§
	OTCD$^{\dagger}_3$	46.6%	74.2%	46.4%	15.5%	45.2%	12359	84155	783	1984	30.3§
	AMIR [35]	**47.2%**	75.8%	46.3%	14.0%	41.6%	**2681**	92856	774	1675	1.0
	DMMOT [5]	46.1%	73.8%	**54.8%**	**17.4%**	42.7%	7909	89874	532	1616	0.3
	STAM16 [7]	46.0%	74.9%	50.0%	14.6%	43.6%	6895	91117	**473**	1422	0.2
	MTDF [36]	45.7%	72.6%	40.1%	14.1%	**36.4%**	12018	**84970**	1987	3377	1.5
	PHD_GSDL16 [10]	41.0%	**75.9%**	43.1%	11.3%	41.5%	6498	99257	1810	3650	8.3
	AM_ADM [37]	40.1%	75.4%	43.8%	7.1%	46.2%	8503	99891	789	1736	5.8
	OTCD$_1$	44.4%	75.4%	45.6%	11.6%	47.6%	5759	97927	759	1787	17.6
	OTCD$_3$	42.4%	75.7%	46.4%	11.1%	50.1%	8318	96177	570	**1299**	**31.7**
MOT17	MTDF17 [36]	**49.6%**	75.5%	45.2%	18.9%	**33.1%**	37124	**241768**	5567	9260	1.2
	HAM_SADF17 [9]	48.3%	**77.2%**	51.1%	17.1%	41.7%	20967	269038	**1871**	**3020**	5.0
	DMAN [5]	48.2%	75.7%	**55.7%**	**19.3%**	38.3%	26218	263608	2194	5378	0.3
	AM_ADM17 [37]	48.1%	76.7%	52.1%	13.4%	39.7%	25061	265495	2214	5027	5.7
	PHD_GSDL17 [10]	48.0%	**77.2%**	49.6%	17.1%	35.6%	23199	265954	3998	8886	6.7
	FPSN [38]	44.9%	76.6%	48.4%	16.5%	35.8%	33757	269952	7136	14491	10.1
	EAMTT [11]	42.6%	76.0%	41.8%	12.7%	42.7%	30711	288474	4488	5720	1.4
	GM_KCF [12]	39.6%	74.5%	36.6%	8.8%	43.3%	50903	284228	5811	7414	3.3
	GM_PHD [14]	36.4%	76.2%	33.9%	4.1%	57.3%	23723	330767	4607	11317	**38.4**
	OTCD$_1$	48.6%	76.9%	47.9%	16.2%	41.2%	**18499**	268204	3502	5588	15.5
	OTCD$_3$	46.9%	76.8%	49.0%	14.9%	44.0%	25204	272153	2154	4072	33.4

is more time-consuming than the IoU cost. We choose $K = 3$ for the balance of performance and speed.

E. MOTCHALLENGE BENCHMARK

We compare the proposed tracker OTCD with other online state-of-the-art trackers in Table 3 on MOT16 and MOT17 test splits. During this evaluation, the detector is used for feature extraction, which means except for the backbone network of the detector, other parts of the detector are not used. The metrics of MVint(LinearK) are accessed from [8], while others are accessed from MOTChallenge leaderboards.

1) MOT16

We first compare OTCD with some trackers that track objects based on the detections provided by [13]. Note that the tracking accuracy MOTA and speed Hz of DeepSORT is superior to those of OTCD$^{\star}_1$, which is not the result reflected in Figure 8 (b), the reasons are as follows: (1) MOTA.

The detections used in Figure 8 (b) and Table 3 are different. The quality of our own detections is inferior to the detections provided by [13]. And a poor quality of detections has a negative impact on trackers. (2) Hz. Detection time consumption is considered for DeepSORT, SORT and OTCD in Figure 8 (b), while it is not considered in Table 3.

Both MVint(LinearK) and $\text{OTCD}_3^\star$ are in the compressed domain, but $\text{OTCD}_3^\star$ achieves a better performance than MVint(LinearK) in all metrics except FP, IDS and Frag. Particularly, the tracking speed of $\text{OTCD}_3^\star$ is about 1.7× faster than MVint(LinearK), while possessing a 4.7% higher MOTA. The tracking results of OTCD based on our own detections are also provided. Note that there is a big gap of MOTA between $\text{OTCD}_1^\star$ and $\text{OTCD}_1^\dagger$, which means the quality of detections has a significant impact on the tracking performance.

As for the public detections, OTCD achieves the fastest tracking speed among all trackers while maintaining a satisfying tracking accuracy. For example, the tracking speed of $\text{OTCD}_1^\star$ is 17× faster than AMIR, which is the state-of-the-art method. Furthermore, $\text{OTCD}_3^\star$ achieves the best performance in Frag.

Interestingly, when K is increased from 1 to 3, both IDS and Frag are greatly reduced. This is reasonable since objects need to be recognized with a lower frequency when data association is only performed in sparse key frames.

2) MOT17

Overall, the tracking performance of OTCD is comparable with other trackers. Particularly, OTCD_1 performs the best in FP, and OTCD_3 takes the second place in Hz. Compared with MTDF17, which achieves the best in MOTA, our trackers run more than 10× (OTCD_1) and 25× (OTCD_3) faster but only with 1.0% (OTCD_1) and 2.7% (OTCD_3) performance degradation in MOTA. Compared with GM_PHD, which possesses the best tracking speed (38.4 Hz), OTCD_3 tracks object at a lower speed (33.4 Hz), but with a 10.5% higher MOTA.

V. CONCLUSION

In this paper, we propose an online MOT tracker OTCD in compressed domain. The RGB images are restored in the key frames for detection and data association, while the MVs and residuals are directly fed into a tracking CNN to propagate objects through non-key frames, which can accelerate our tracker significantly. Furthermore, an appearance CNN which shares features with detector is introduced to assist data association, and it can be trained with detector jointly. Experimental results on MOTChallenge benchmark demonstrate the effectiveness of appearance CNN and tracking CNN, as well as the joint training of appearance CNN and detector.

REFERENCES

[1] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[2] S. Tang, M. Andriluka, B. Andres, and B. Schiele, "Multiple people tracking by lifted multicut and person re-identification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3701–3710.
[3] B. Yang and R. Nevatia, "An online learned CRF model for multi-target tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2034–2041.
[4] S.-H. Bae and K.-J. Yoon, "Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 595–610, Mar. 2018.
[5] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 379–396.
[6] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3645–3649.
[7] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4846–4855.
[8] T. Ujiie, M. Hiromoto, and T. Sato, "Interpolation-based object detection using motion vectors for embedded real-time tracking systems," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 729–7298.
[9] Y.-C. Yoon, A. Boragule, Y.-M. Song, K. Yoon, and M. Jeon, "Online multi-object tracking with historical appearance matching and scene adaptive detection filtering," in *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Nov. 2018, pp. 1–6.
[10] Z. Fu, P. Feng, F. Angelini, J. Chambers, and S. M. Naqvi, "Particle phd filter based multiple human tracking using online group-structured dictionary learning," *IEEE Access*, vol. 6, pp. 14764–14778, 2018.
[11] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro, "Online multi-target tracking with strong and weak detections," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 84–99.
[12] T. Kutschbach, E. Bochinski, V. Eiselein, and T. Sikora, "Sequential sensor fusion combining probability hypothesis density and kernelized correlation filters for multi-object tracking in video data," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug./Sep. 2017, pp. 1–5.
[13] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan, "Poi: Multiple object tracking with high performance detection and appearance feature," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 36–42.
[14] V. Eiselein, D. Arp, M. Pätzold, and T. Sikora, "Real-time multi-human tracking using a probability hypothesis density filter and multiple detectors," in *Proc. IEEE 9th Int. Conf. Adv. Video Signal-Based Surveill.*, Sep. 2012, pp. 325–330.
[15] H. Kieritz, S. Becker, W. Hübner, and M. Arens, "Online multi-person tracking using integral channel features," in *Proc. 13th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2016, pp. 122–130.
[16] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The CLEAR MOT metrics," *EURASIP J. Image Video Process.*, vol. 2008, pp. 1-1–1-10, Dec. 2008.
[17] S. R. Alvar and I. V. Bajić, "MV-YOLO: Motion vector-aided tracking by semantic object detection," in *Proc. IEEE 20th Int. Workshop Multimedia Signal Process. (MMSP)*, Aug. 2018, pp. 1–5.
[18] S. Wang, H. Lu, and Z. Deng, "Fast object detection in compressed video," 2019, *arXiv:1811.11057*. [Online]. Available: https://arxiv.org/abs/1811.11057
[19] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6026–6035.
[20] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 685–692.
[21] B. Wang, G. Wang, K. L. Chan, and L. Wang, "Tracklet association with online target-specific metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1234–1241.
[22] *Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video*, Standard ITU (ISO/IEC) JTC1, Rec.H.262 ISO/IEC 13 818-2 (MPEG-2 Video), 1994.
[23] T. Wiegand, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec.H.264|ISO/IEC 14496-10 AVC)*, document JVT-G050, 2003.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.
[25] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.
[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[27] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[28] S. Zhang, R. Benenson, and B. Schiele, "Citypersons: A diverse dataset for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, vol. 1, no. 2, pp. 4457–4465.
[29] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," 2015, *arXiv:1504.01942*. [Online]. Available: https://arxiv.org/abs/1504.01942
[30] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," 2016, *arXiv:1603.00831*. [Online]. Available: https://arxiv.org/abs/1603.00831
[31] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 17–35.
[32] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybridboosted multi-target tracker for crowded scene," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 2953–2960.
[33] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The CLEAR 2006 evaluation," in *Proc. Int. Eval. Workshop Classification Events, Activities Relationships*. Berlin, Germany: Springer, 2006, pp. 1–44.
[34] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3464–3468.
[35] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, vol. 4, no. 5, pp. 300–311.
[36] Z. Fu, F. Angelini, J. Chambers, and S. M. Naqvi, "Multi-level cooperative fusion of GM-PHD filters for online multiple human tracking," *IEEE Trans. Multimedia*, to be published.
[37] S.-H. Lee, M.-Y. Kim, and S.-H. Bae, "Learning discriminative appearance models for online multi-object tracking with appearance discriminability measures," *IEEE Access*, vol. 6, pp. 67316–67328, 2018.
[38] S. Lee and E. Kim, "Multiple object tracking via feature pyramid Siamese networks," *IEEE Access*, vol. 7, pp. 8181–8194, 2019.
[39] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug./Sep. 2017, pp. 1–6.

QIANKUN LIU received the B.S. degree from Xidian University, in 2017. He is currently pursuing the M.S. degree in electrical engineering with the University of Science and Technology of China. His research interests include detection and object tracking.

BIN LIU received the B.S. and M.S. degrees, both in electrical engineering, from the University of Science and Technology of China, Hefei, China, in 1998 and 2001, respectively, and the Ph.D. degree in electrical engineering from Syracuse University, Syracuse, NY, USA, in 2006. He is currently an Associate Professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include computer vision and the Internet of Things.

YUE WU received the B.E. degree in electronic engineering and the Ph.D. degree in information and communication engineering from the University of Science and Technology of China (USTC), Hefei, China, in 2012 and 2017, respectively. He is a Research Engineer with Alibaba Group, Hangzhou, China. His research interests include multimedia, computer vision, machine learning, and data mining.

WEIHAI LI was born in Dalian, China, in 1975. He received the Ph.D. degree in electromagnetic field and microwave technology from the University of Science and Technology of China (USTC), in 2003. He is currently an Associate Professor with the School of Information Science and Technology of USTC. He leads a research team of Digital Media Analysis in the Group of Information Processing Center. His team is investigating technologies of media forensics, media content protection and analysis, remote sensing image processing, etc.

NENEGHAI YU received the B.S. degree from Nanjing University of Posts and Telecommunications, in 1987, the M.E. degree from Tsinghua University, in 1992, and the Ph.D. degree from the University of Science and Technology of China, in 2004, where he is currently a Professor. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.