Visual Tracking With Multiview Trajectory Prediction

Minye Wu, Haibin Ling, *Senior Member, IEEE*, Ning Bi, Shenghua Gao, *Member, IEEE*, Qiang Hu, *Member, IEEE*, Hao Sheng, *Member, IEEE*, and Jingyi Yu, *Senior Member, IEEE*

***Abstract*— Recent progresses in visual tracking have greatly improved the tracking performance. However, challenges such as occlusion and view change remain obstacles in real world deployment. A natural solution to these challenges is to use multiple cameras with multiview inputs, though existing systems are mostly limited to specific targets (*e.g.* human), static cameras, and/or require camera calibration. To break through these limitations, we propose a generic multiview tracking (GMT) framework that allows camera movement, while requiring neither specific object model nor camera calibration. A key innovation in our framework is a cross-camera trajectory prediction network (TPN), which implicitly and dynamically encodes camera geometric relations, and hence addresses missing target issues such as occlusion. Moreover, during tracking, we assemble information across different cameras to dynamically update a novel collaborative correlation filter (CCF), which is shared among cameras to achieve robustness against view change. The two components are integrated into a correlation filter tracking framework, where features are trained offline using existing single view tracking datasets. For evaluation, we first contribute a new generic multiview tracking dataset (GMTD) with careful annotations, and then run experiments on the GMTD and CAMPUS datasets. The proposed GMT algorithm shows clear advantages in terms of robustness over state-of-the-art ones.**

***Index Terms*— Deep learning, multiview, tracking, correlation filter, trajectory.**

I. Introduction

VISUAL object tracking is a fundamental problem in computer vision. The existing tracking approaches can be classified by the number of cameras and targets. For single-camera-single-target tracking, there are two popular main-stream branches of approaches. One branch is based on correlation filter, which trains a regressor based on the properties of circular correlation and Fourier domain operations [1]–[3]. These approaches reach a real-time tracking speed with hand-craft low-level features. References [2], [3] also exploit deep features and show more robustness in relative tasks. The other branch formulates the problem as detection/classification tasks [4]–[6]. However, their performance degenerate due to relatively poor diversity of training samples.

Moreover, object occlusion in the scene can harshly cut down the performance of single-camera trackers. Under the single camera multiple targets setting, object tracking is formulated as a detection and matching problem. Objects in each frame will be firstly detected and then re-identified in the continuous frames [7]–[9]. But their detector may be significantly affected by occlusions. Meanwhile, these trackers typically only handle certain groups of targets, e.g. pedestrians, vehicles.

Among different tracking tasks, we focus on generic (also known as model-free) visual tracking, which requests little prior information about the tracking target and has been intensively studied due to its wide range of applications. Despite the significant progresses in tracking algorithms, tracking in the real world is still challenging especially when target appearance is distorted or damaged due to view change or occlusion.

A natural way to alleviate the above issues is to use multiple cameras for tracking, which provides important multiview information for handling cross-view target appearance change and occlusion. Existing multiview tracking algorithms, however, typically focus on specific targets such as human, and often rely heavily on detection or re-identification models. Another limitation is that cameras are often assumed to be stationary, so background subtraction and/or camera calibration can be used to facilitate target localization. These limitations largely harm the generalization of multiview tracking to real world applications. Consequently, effectively using multiple cameras for generic visual tracking remains an open problem.

Manuscript received December 14, 2019; revised May 26, 2020 and July 8, 2020; accepted July 27, 2020. Date of current version August 18, 2020. This work was supported in part by the National Key Research and Development Program under Grant 2018YFB2100500; in part by the Programs of NSFC under Grant 61976138, Grant 61861166002, Grant 61872025, Grant 61635002, and Grant 61977047; in part by Science and Technology Commission of Shanghai Municipality (STCSM) under Grant 2015F0203-000-06; in part by the Science and Technology Development Fund, Macau, under Grant 0001/2018/AFJ; in part by Shanghai Municipal Education Commission (SHMEC) under Grant 2019-01-07-00-01-E00003; in part by the Fundamental Research Funds for the Central Universities; and in part by the Open Fund of the State Key Laboratory of Software Development Environment under Grant SKLSDE2019ZX-04. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ioannis Kompatsiaris. *(Corresponding authors: Qiang Hu; Jingyi Yu.)*

Minye Wu was with Yoke Intelligence, Shanghai 201210, China. He is now with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China, and also with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China.

Haibin Ling is with the Computer Science Department, Stony Brook University, Stony Brook, NY 11794 USA.

Ning Bi was with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China, and also with Yoke Intelligence, Shanghai 201210, China. She is now with the School of Computing, University of Leeds, Leeds LS29JT, U.K.

Shenghua Gao, Qiang Hu, and Jingyi Yu are with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China (e-mail: huqiang@shanghaitech.edu.cn; yujingyi@shanghaitech.edu.cn).

Hao Sheng is with the State Key Laboratory of Software Development Environment, Beijing Advanced Innovation Center for Big Data and Brain Computing, School of Computer Science and Engineering, Beihang University, Beijing 100191, China.

Digital Object Identifier 10.1109/TIP.2020.3014952

To address the above mentioned issues, we propose a novel *generic multiview tracker* (GMT) in this paper by encoding rich multiview information with learning-based strategies. A key component in GMT is a cross-camera *trajectory prediction network* (TPN), which takes tracking results from reliable views to predict those for unreliable ones. TPN effectively addresses the problem caused by occlusion or serious target view change. Another novel component in our GMT is the *collaborative correlation filter* (CCF), which assembles cross-camera information to update a correlation filter. The filter is shared among different views, and hence improves tracking robustness against view change. TPN and CCF are integrated into the correlation filter tracking framework, where the features are trained offline using existing single view tracking datasets.

For evaluation, we first construct a new *generic multiview tracking benchmark* (GMTD). Then, the proposed GMT algorithm is tested on this dataset, and demonstrates clear advantages in terms of robustness in comparison with state-of-the-art tracking algorithms.

The contributions of this paper are summarized as follows:

- We propose a *learning-based generic multiview tracking framework*, which requires little prior information about the tracking target, allows camera movement and requires no camera calibration.
- We adopt the novel *cross-view trajectory prediction network* that encodes camera geometric relations for improving tracking robustness.
- We use a *collaborative correlation filter* that learns an online cross-view model and hence achieves natural robustness against view change.
- We contribute a new generic multiview tracking dataset with manual annotations per frame, which is expected to further facilitate research in related topics.

The source code and the dataset will be released to public with the publication of this paper.

II. Related Work

A. Generic Single View Visual Tracking

Visual object tracking is one of the most fundamental tasks in computer vision and has attracted a great amount of research efforts. It is beyond the scope of this paper to review all previous work in tracking. In the following section, we choose to review some of the most relevant ones, including correlation filter-based trackers, and CNN-based trackers.

Our work is most related to correlation filter-based trackers. Based on Discriminative Correlation Filters (DCF), MOSSE is proposed in [10] to efficiently train the correlation filter by minimizing the sum of the squared error between ground-truth and output in the Fourier domain. The idea is later adopted and extended in Kernel Correlation Filter tracking (KCF) [1] and since then starts to gain great amount of attention. Among many tracking algorithms along the line, the series proposed by Danelljan *et al.* [2], [3], and [11] and Bhat *et al.* [12] provide the main backbone to our study. In these studies, DSST [13] and fDSST [11] use multiple correlation filters to estimate object translation and scale separately. C-COT [2] further enhanced predictions by learning from multi-resolution feature maps. ECO [3] and ECO+ [12] make further improvements on feature representation, sample space management and online update scheme in order to obtain more intuitive tracker and win both accuracy and efficiency.

Another line of tracking algorithms, which use deep learning for feature representation, are the CNN-based trackers. Reference [14] uses network to map an exemplar and the search region to a response map. Siamese network-based trackers [15] locates target object by matching initial object appearance. Siamese Fully Convolutional networks (SiameseFC) [16] solves the problem by training a fully convolutional Siamese architecture to evaluate the semantic similarity between proposals and target image. Utilizing light-weight CNN network with correlation filters, DCFNet [17] performs real-time tracking. Recent extensions continuously improve the performance such as in [18], [19].

Moreover, our work explicitly addresses occlusion and target view change, which have been studied explicitly as well in some single view tracking algorithms. One way to resolve the appearance variation of the target is to maintain effective sample sets [3], [20], [21], which involves balancing different aspects of the target, for correlation filters online-training. Another strategy is conducting complementary information, for example, [22] makes use of spatial information, [23] enrolls flow between frames as part of features, [24] add a semantic branch to enhance prediction. For occlusion, there are part-based correlation filter trackers like [25]–[27], which learn target appearance in parts and tend to be robuster to partial-occlusion. When targets have strong structure relation, like pedestrians in RPAC [28], KCFs are assigned to five different parts of each target for robustness. [20], [21], [29] set thresholds to evaluate the results of correlation filtering. Mei *et al.* [30] investigate sparse representation for occlusion detection. MUSTer [31] avoid drifting and stabilize tracking by aggregating image information using short- and long-term stores.

Different from all above-mentioned methods, our study focuses on multiview visual tracking that takes input streams from multiple cameras simultaneously. While borrowing some components from these single view tracking algorithms, we develop novel strategies such as cross-view trajectory prediction and a multiview collaborative correlation filter. These strategies, as demonstrated in our carefully designed experiments, clearly improve the tracking robustness.

B. Multiview Visual Tracking

Multi-camera inputs have been used for visual tracking. For examples, Khan and Shah [32] apply a planar homography constraint on calibrated cameras for tracking pedestrians on the ground, and show the power of multiview system with common overlaps; studies in [33], [34] explore spatial relations of target object in multi-camera system by analyzing entry/exit rates across pairs of cameras; [35] exploits dynamical and geometrical constraints among static cameras in a linear model; [36] models the relationship between viewpoint and 3D aspect parts of object.

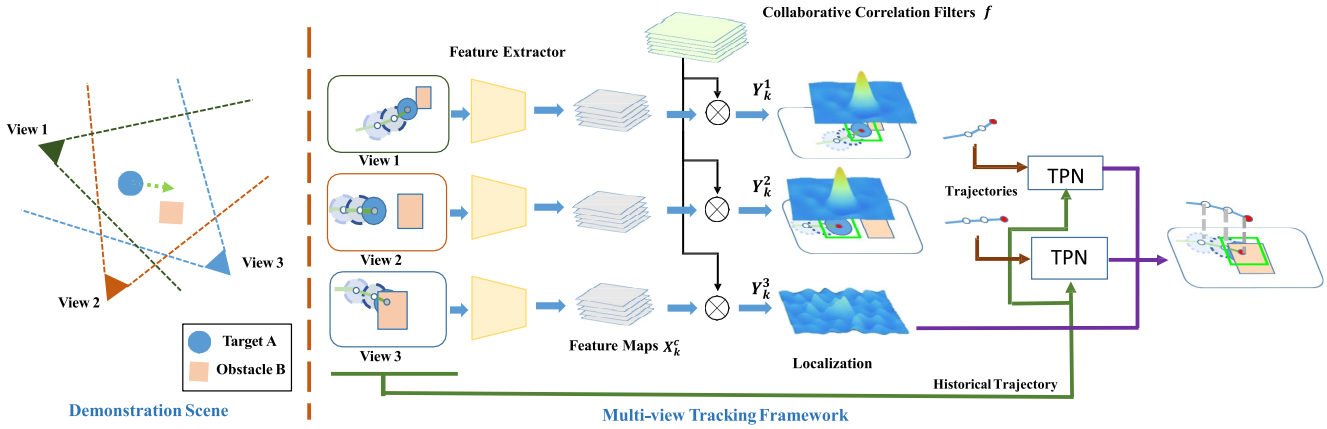

Fig. 1. Overview. **Left**: an illustrative scene including three cameras/views (view 1, view 2 and view 3), a tracking target A and an obstacle B. **Right**: frames from all three views, where occlusion happens in view 3, serve as the input for our tracking algorithm. Three major steps are applied sequentially: **1.)** Shared feature extraction layers are performed on each view to extract cross-scale spatial-aware features (§III-C). **2.)** An online updated set of collaborative correlation filters are shared by all views for tracking inference (§III-D). **3.)** For a view with low tracking confidence (e.g. due to occlusion in view 3), our framework triggers trajectory prediction network (TPN) to estimate its target location based on trajectories from other views (§III-E).

The most challenging thing in single view tracking is occlusion. The target trajectory may become discontinuous when the target is hidden by obstacles. Reference [37] tackles it on multi-view people tracking problem by using target association method between calibrated cameras based on the tracking results of each target on each view. Trajectories also can be combined with 3D and appearance cues, and form a hierarchical composition structure for multi-view people tracking in [38]. The extension [39] improves the performance by integrating semantic attributes with trajectories for cross-view people tracking.

Multi-target Multi-camera Tracking (MTMCT) problem is a special case of multi-view tracking. MTMCT tracks and identifies pedestrians across cameras. This type of methods usually combine person detection and re-identification methods in order to locate pedestrians and build matches between persons from different moment and cameras. For example, [8], [40] learn good feature for matching from detected persons by utilizing CNNs; the multiview trackers [7], [37], [41], [42] track a target in each view and match each instance between cameras that are intuitively capable of occlusion situations due to diversity of observation directions. The limitations of these methods are obvious, they need neural-based detectors, such as Faster R-CNN [43], SSD [44] or pose-based detectors [45], [46], to detect pedestrians in images. That restricts the category of target object, and cannot apply on generic objects.

III. Generic Multiview Tracking Framework

A. Problem Formulation

Given a set of synchronized video streams from different cameras/views, we aim to localize a target (initialized in the first frames) across time. More specifically, let the system input be $\mathcal{I}_t = \{I_t^c\}_{c=1}^{n_c}$ at time t for n_c cameras, and let $\mathcal{B}_1 = \{\mathbf{b}_1^c \in \mathbb{R}^4\}_{c=1}^{n_c}$ be the initial target bounding boxes for all views. Then, our multiview tracking task is to locate the target by finding $\mathcal{B}_t = \{\mathbf{b}_t^c \in \mathbb{R}^4\}_{c=1}^{n_c}$, given $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_t\}$ and $\mathcal{B}_1$,[1] where $\mathbf{b}_t^c$ is the target bounding box with four parameters in view c at time t. We also define a trajectory set $\mathcal{G}_{t_1,t_2} = \{\mathcal{G}_{t_1,t_2}^c\}_{c=1}^{n_c}$, where $\mathcal{G}_{t_1,t_2}^c = \{\mathbf{g}_t^c \in \mathbb{R}^2\}_{t=t_1}^{t_2}$ for each view c and $\mathbf{g}_t^c$ is the center of $\mathbf{b}_t^c$ in a consecutive time period from time t_1 to time t_2.

[1]The estimated $\{\mathcal{B}_2, \ldots, \mathcal{B}_{t-1}\}$ can be used as well.

B. Framework Overview

The key motivation of our *generic multiview tracker* (GMT) is to explore rich cross-view information to improve tracking robustness, especially against occlusion and target/camera view change. We adopt a correlation filter-based tracking framework as the backbone, and equip it with the novel *collaborative correlation filter* (CCF) and cross-view *trajectory prediction network* (TPN) techniques. An overview of the GMT pipeline is given in Figure 1, which we will briefly describe as follows.

During online tracking, for newly arrived multiview images $\mathcal{I}_t$, GMT locates the target (*i.e.*, calculates $\mathcal{B}_t$) in three major steps. **First**, similar to C-COT [2], we apply correlation filters on multi-scale image patch features to handle objects under variant scales indicated by k. For each view c and scale k, a 272×272 region of interest (RoI) patch U_k^c is cropped and scaled around $\mathbf{b}_{t-1}^c$. The patch is then fed into a feature extraction network $\phi(\cdot)$ (§III-C), which is shared among different views, to generate feature maps, denoted by X_k^c, for each view c.

Second, each feature map is convolved with the shared CCF f for initial target localization, producing confidence map Y_k^c. The maximum response over different scales k is then picked for the initial tracking results. For sufficiently confident results, they will be assembled cross view to update CCF. In other words, CCF is online updated to enhance its robustness against view and appearance change.

Third, for a view with low confident initial tracking results (*e.g.*, view 3 in Figure 1), TPN will be used to estimate its tracking result by implicitly taking the geometric relations among different views/cameras into account.

Algorithm 1 Generic Multiview Tracking

Input : $\mathcal{I}_t$: input images at time t;
$\mathcal{G}_{t_0,t-1}$: previous trajectories of all cameras;
f: dynamically updated collaborative filters;
$\mathcal{B}_{t-1}$: tracking results (boxes) in last frame;
$\mathcal{Z}^c$: training samples for each view c;
Output: $\mathcal{B}_t$: tracking result for time t;
$\mathcal{G}_{t_0,t}, f, \mathcal{Z}^c$: updated results;

for *each camera* c **do**
  for *each scale* k **do**
    U_k^c = CropImagePatch(I_t^c, $\mathbf{b}_{t-1}$, k);
    X_k^c = FeatureExtraction(U_k^c);
    Y_k^c = Correlation operation between X_k^c and f;
  end
  $k' = \arg\max_k\{\max(Y_k^c)\}$;
  Localize object $\mathbf{b}_t^c$ and update $\mathbf{g}^c$ according to $Y_{k'}^c$;
  $q^c = \max(Y_{k'}^c)$;
  if $(q^c \geq \tau)$ and $(t \mod 7 = 0)$ **then**
    $\mathcal{Z}^c = \mathcal{Z}^c \cup X_{k'}^c$;
  end
end
if $t \mod 7 = 0$ **then**
  Update f by training on all $\mathcal{Z}_c : c = 1, \ldots, n_c$;
end
for *each camera* c **do**
  if $q^c < \tau$ **then**
    Update $\mathbf{g}^c$ using TPN (§ III-E);
    Update $\mathbf{b}_t^c$ accordingly;
  end
end

The overall online tracking framework is summarized in algorithm 1. Ahead of this procedure, we conduct offline training for fine-tuning feature extractor (§III-C) and get TPN ready (§III-E). More details are described in the following subsections.

C. Spatial-Aware Feature Extraction Network

Compared with hand-crafted features, features generated by deep neural networks contain more semantic information, while less spatial properties are preserved, such as translation invariant. Conversely, feature maps from shallow layers contain more translation invariant property while losing semantic properties for discriminative tracking. A recent work [47] indicates that traditional ResNet backbone has difficulty to deliver translation invariance features. To address this problem, it finetunes the network for a better performance. Similarly, to enhance the translation invariance and correlation filtering adaptation, we add FFT layers to simulate correlation filtering, combine them together and train it in an end-to-end way. In this way, we exploit object's location to supervise network training. Specifically, the FFT (IFFT) layer is a differentiable Fast Fourier Transform (Inverse Fast Fourier Transform) operation.

In order to improve the feature adaptability to correlation filter-based multiview tracking, we fine tune feature extraction

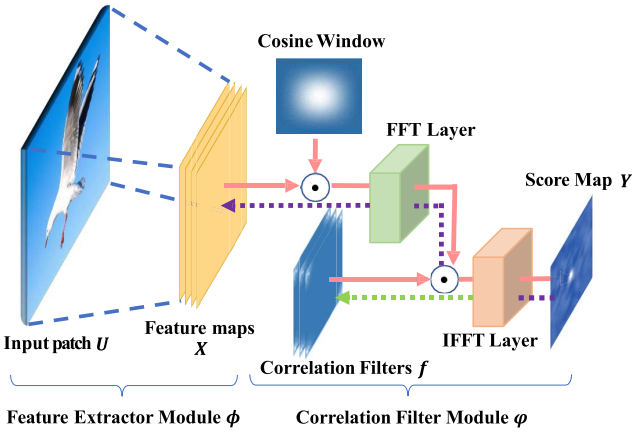

Fig. 2. Two-stage training of the feature extractor (offline). The first stage (green dotted line) is for training the correlation filters while keeping the feature extraction module fixed; while the second stage (purple dotted line) is for training the feature extraction module while keeping the correlation filters fixed. After training, the feature extractor(ResNet-50) plays an important role in the whole tracking pipeline (illustrated in figure 1).

model based on ResNet-50 that is pretrained on ImageNet. The offline fine tuning involves two components: the feature extractor module $\phi(\cdot)$ and the correlation filter module $\varphi(\cdot)$, as shown in Figure 2.

We construct the training set from existing single view tracking datasets including VOT2017 [48], OTB100 [49] and LaSOT [50]. Different frames in a sequence can be regarded as the multi-view appearances of the same object. Data augmentation is conducted by slightly disturbing object locations so that let the network has the ability to locate the same object even it has some translation in image space and improve its translation invariance.

As described above, The network needs to learn two parts of parameters during training. One is parameters from the feature extractor module $\phi(\cdot)$, the other one is correlation filters from the correlation filter module $\varphi(\cdot)$. Instead of optimizing all parameters at once, we use a two-stage training scheme. We randomly choose 16 sample pairs from the same sequence (same object) as a training batch instead of choosing from random sequences. Specifically, in the first stage of feature extractor training, we aim to obtain the optimal correlation filter for current feature extractor so that we can evaluate whether the feature extractor is good or bad in the second training stage. So Eq. 1 firstly optimizes the correlation filter to make the score map close to the ground truth while fixing the parameters of feature extractor. We also add a regular term for preventing over fitting. Here, first 10 training pairs are used to train the correlation filter f_{train} by minimizing the following objective function:

$$f_{\text{train}}^* = \arg\min_{f_{\text{train}}} \|Y^* - Y\|_2^2 + \eta \cdot \|f_{\text{train}}\|_2^2$$
$$Y = \varphi\big(\psi_{\cos}(\phi(\mathrm{U}, \theta_f)), f_{\text{train}}\big) \quad (1)$$

where Y^* is target score map; θ_f is the parameters for feature extractor; η is a penalty parameter; and $\psi_{\cos}(\cdot)$ is the Hann window function.

After the first training stage, we obtain the optimal correlation filters f^*_{train} of the current object. Then, we use the rest of the sample pairs to optimize the feature extractor $\phi(\cdot)$. We fix the correlation filter obtained from the first stage and use objective function Eq. 2 to conduct evaluation and generate gradients for back-propagation. Same as in the first stage, we make the output score map as close as possible to the ground truth. The output score map Y may have some blurred noise which is supposed to be zeros when only L2 loss is used. In other words, high score values may appear on non-peak area on output response map when using MSE loss. The value of these noises are relatively low, where the MSE loss remains in a low level. As a result, we cannot obtain gradients of sufficient strength from L2 loss. A Sobel term is added to Eq. 2 to address this issue. The gradient-like term is sensitive to this kind of noise, and can provide another direction of gradients to alleviate this phenomenon.

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \|Y^* - Y\|_2^2 + \lambda \nabla_S \\ \nabla_S &= \|\nabla_x(Y^*) - \nabla_x(Y)\|_2^2 + \|\nabla_y(Y^*) - \nabla_y(Y)\|_2^2 \\ Y &= \varphi(\psi_{\cos}(\phi(\mathrm{U}, \theta)), f^*_{\text{train}}) \end{aligned} \tag{2}$$

where $\nabla_x(\cdot)$ and $\nabla_y(\cdot)$ are horizontal and vertical Sobel operator respectively. We use Adam [51] optimizer to train our spatial-aware feature extraction networks. Note that f_{train} is only used during offline training process. After offline training, the trained feature extractor plays an important role in the whole tracking pipeline (illustrated in Figure 1).

D. Collaborative Correlation Filter

View change is a notorious issue that troubles single view trackers. Fortunately, in the multiview tracking setup, images captured from different cameras naturally provide cross-view information for building reliable tracking models. Therefore, we extend the traditional correlation filter to a *collaborative* one. Specifically, during tracking, we update the correlation filters online with information collaboratively collected from all sufficiently reliable views.

Denote the training samples dynamically collected from view c by $\mathcal{Z}^c = \{X_j^c\}_{j=1}^{m_c}$ with m_c samples. We train a shared multiview collaborative correlation filter f using all samples from different sample sets (*i.e.* $\{\mathcal{Z}^c\}_{c=1}^{n_c}$) by minimizing the following function:

$$E(f) = \sum_{c=1}^{n_c} \sum_{j=1}^{m_c} \alpha_j^c \|X_j^c * f - Y_j^c\|^2 + \|f\|^2 \tag{3}$$

where Y_j^c denotes the score map of the j-th sample in the c-th camera, and $*$ is the convolution operation. The weights $\alpha_j^c \geq 0$ represent the importance of the j-th training sample of camera c, which is positively correlated with q^c during tracking. In this formulation, training samples from all camera views contribute to filter updating, and thus enhance the robustness of the learned filters against view change. When tracking a object, we apply the same correlation filter f to all camera views. This calculation is very similar to the correlation filter module $\varphi(\cdot)$ when we train the feature extractor module $\phi(\cdot)$. But the difference is, we use the optimizer used in [3] to minimize this function during tracking.

E. Trajectory Prediction Network

Our key novelty in multiview tracking is the proposed *Trajectory Prediction Network* (TPN) for handling tracking failure using cross-view trajectory prediction. Intuitively, when the target is occluded (or damaged similarly) in a *target view b*, we can usually still reliably track the target in a different view, say, a *source view a*. Then, based on the geometric relation between the two views, we shall be able to locate the occluded object in view b from the trajectory in view a. The job, despite being nontrivial due to non-linearity and camera movement, is done by TPN.

1) Network Design: Denote the object's position at time t for view a and view b by $\mathbf{g}_t^a$ and $\mathbf{g}_t^b$, respectively. It is natural for us to find the direct mapping and prediction between them. This idea does not work in practice due to the large range of absolute coordinate of object locations. Instead, we decompose a trajectory as a sequence of between-frame movements, denoted by $\mathbf{r}_t^c = \mathbf{g}_t^c - \mathbf{g}_{t-1}^c$ as the the motion vector for camera c at time t. Then, TPN aims to map from $\mathbf{r}_t^a$ to $\mathbf{r}_t^b$ at time t.

At time t, based on 3D geometrical constrains, the object position $\mathbf{g}_t^b$ in view b can be transformed from its location $\mathbf{g}_t^a$ in a. Let d_t^a (or d_t^b) and $\mathbf{T}_t^a$ (or $\mathbf{T}_t^b$) denote respectively the depth and transformation matrix for view a (or b). We can have the following derivation

$$\begin{aligned} \lambda \begin{bmatrix} \mathbf{g}_t^b \\ 1 \end{bmatrix} &= \mathbf{T}_t^b d_t^a (\mathbf{T}_t^a)^{-1} \begin{bmatrix} \mathbf{g}_t^a \\ 1 \end{bmatrix} \\ \lambda \begin{bmatrix} \mathbf{g}_{t_0}^b + \sum_{i=t_1}^t \mathbf{r}_i^b \\ 1 \end{bmatrix} &= \mathbf{Q}_t \begin{bmatrix} \mathbf{g}_{t_0}^a + \sum_{i=t_1}^t \mathbf{r}_i^a \\ 1 \end{bmatrix} \\ \lambda \begin{bmatrix} \mathbf{r}_t^b \\ 1 \end{bmatrix} &= \mathbf{Q}_t \begin{bmatrix} \mathbf{g}_{t-1}^a + \mathbf{r}_t^a \\ 1 \end{bmatrix} - \lambda \begin{bmatrix} \mathbf{g}_{t_0}^b + \sum_{i=t_1}^{t-1} \mathbf{r}_i^b \\ 1 \end{bmatrix} \end{aligned} \tag{4}$$

where $\mathbf{Q}_t := \mathbf{T}_t^b d_t^a (\mathbf{T}_t^a)^{-1}$, λ is used for normalization, and t_0 is the begin time of a trajectory. Such relation between $\mathbf{r}_t^b$ and $\mathbf{r}_t^a$ motivates us to design the following *Recurrent Neural Network* (RNN)-based TPN model and introduce corresponding hidden parameters in the model:

$$\mathbf{r}_t^b = \Theta_{\text{pos}}(\Theta_{\text{rnn}}(\Theta_{\text{enc}}(\mathbf{r}_t^a), \mathbf{p}_t), \mathbf{h}_{\text{p}}) \tag{5}$$

In the model, $\Theta_{\text{enc}}(\cdot)$ is an encoder network to translate/convert the input; $\Theta_{\text{rnn}}(\cdot,\cdot)$ indicates stacked RNNs to simulate the non-linear transformation decided by $\mathbf{Q}_t$ and accumulate temporal information (*e.g.* $\sum_i^t \mathbf{r}_i^b$, $\mathbf{g}_t^a$ and object movement); $\mathbf{p}_t$ denotes hidden states of RNN at time t and it initially encodes camera matrices $\mathbf{T}_t^c$ and the initial position $\mathbf{g}_{t_0}^a$; $\mathbf{h}_{\text{p}}$ encodes the initial position $\mathbf{g}_{t_0}^b$; and $\Theta_{\text{pos}}(\cdot)$ decodes all features to output the results. TPN can learn these information in embedding space for each moment from object movements. It's noticed that TPN does not take any calibration data as input. As a result, our method is calibration free and allows camera movements.

The structure of TPN is shown in Figure 3. The initial hidden state $\mathbf{p}_{t_0}^k$ of k-th RNN layer consists of zero vector and

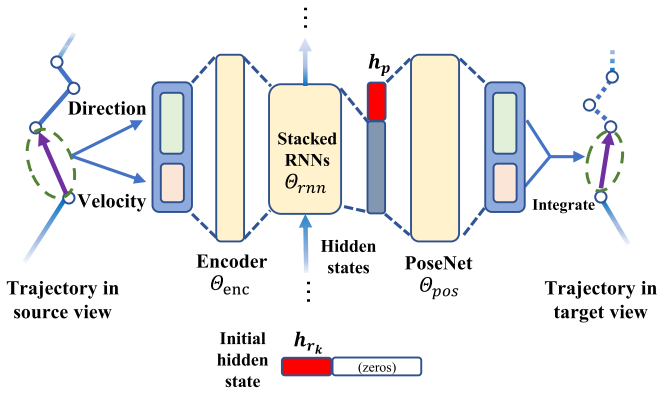

Fig. 3. Prediction by TPN. The trajectory in the source view a is decomposed into direction and velocity and arranged in a motion vector. Then the encoder Θ_{enc} maps the motion vector into a 128-dimensional representation, which then passes through Θ_{rnn} containing with 2 stacked RNN layers and 2 learnable hidden states $\mathbf{h}_{r_1}, \mathbf{h}_{r_2}$. Following that, another hidden vector $\mathbf{h}_{\text{p}}$ is concatenated with output of RNNs and sent to PoseNet Θ_{pos} that captures the between-view geometric constrain. During prediction, only $\mathbf{h}_{r_1}, \mathbf{h}_{r_2}$ and $\mathbf{h}_{\text{p}}$ need to be updated. The trajectory in the occluded target view b is corrected by integrating the predicted motion vector $\mathcal{R}^b_{t_0,t}$. Here, we only illustrate a data flow at a specified time t during prediction.

a learnable hidden parameter vector $\mathbf{h}_{r_k}$, *i.e.* , $\mathbf{p}^k_{t_0} = [\mathbf{h}_{r_k}, \mathbf{0}]$. These initial hidden states form $\mathbf{p}_{t_0} = \{\mathbf{p}^1_{t_0}, \mathbf{p}^2_{t_0}\}$ together, where the hidden states $\mathbf{p}_t$ of RNN in general encode temporal information. Moreover, we use $\mathbf{h}_{\text{rnn}} = [\mathbf{h}_{r_1}, \mathbf{h}_{r_2}]$ to represent all hidden parameters of RNN. PoseNet is a deep fully-connected network, with ReLU activation function, whose input includes both the output of RNN and a hidden parameter vector $\mathbf{h}_{\text{p}}$. Therefore, TPN can be viewed as a decoder that parses hidden parameters to a mapping function that maps motion vectors from source view into the target view. Note that $\mathbf{Q}_t$ is treated as a dynamic transformation, and thus allows camera movement.

In practice, estimated trajectories from a source view often contain noise that may cause unstable trajectory prediction for the target view. For this reason, we smooth the source trajectories before sending them to TPN. Specifically, we use three adjacent object's locations at adjacent moments to smooth motion vectors. The smoothed motion vector $\mathbf{r}^c_t$ (we abuse the notation $\mathbf{r}^c_t$ for conciseness) is estimated by

$$\mathbf{r}^c_t = \frac{1}{3}\sum_{j=0}^{2} \|\mathbf{g}^c_{t-j} - \mathbf{g}^c_{t-j-1}\| \cdot \frac{1}{3}\sum_{j=0}^{2} \frac{\mathbf{g}^c_{t-j} - \mathbf{g}^c_{t-j-1}}{\|\mathbf{g}^c_{t-j} - \mathbf{g}^c_{t-j-1}\|}. \quad (6)$$

In this way, $\mathbf{r}^{\mathbf{c}}_{\mathbf{t}}$ consists of two parts, the velocity (left) and the direction (right). The number $\frac{1}{3}$ is used for taking average.

2) Trajectory Dataset: To train and test TPN, we first prepare a trajectory dataset by collecting trajectory pairs from different kinds of camera settings and object motions. In total, 25 scenarios are used for training and 8 for testing. The data of each scenario is captured by two cameras with different relative pose constraints. The between-camera relative pose may change slightly during capturing. An object is placed in front of the cameras. We move the object or the cameras randomly in the free space so that trajectory pairs are formed in different views without occlusion. We have 30,000 frames altogether and each sequence has at least 900 frames.

3) Training TPN: As for training, we sample n_b ($n_b = 100$) trajectory pairs from the 25 scenarios per batch. The i-th trajectory pair $(\mathcal{G}^{a,i}_{t_0,t_2}, \mathcal{G}^{b,i}_{t_0,t_2})$ is chosen from 90 continuous frames, *i.e.* , $t_2 - t_0 = 89$ with t_0 randomly chosen. Using Eq.6, we get a motion vector set pair $(\mathcal{R}^{a,i}_{t_0,t_2}, \mathcal{R}^{b,i}_{t_0,t_2})$, where $\mathcal{R}^{a,i}_{t_0,t_2} = \{\mathbf{r}^{a,i}_t\}^{t_2}_{t=t_0}$. Let $\mathbf{h}^i = (\mathbf{h}^i_{\text{rnn}}, \mathbf{h}^i_{\text{p}})$ be the learn-able hidden parameters of i-th sample pair in networks and θ be other parameters of networks. Our objective is to find an optimal θ^* that:

$$\begin{aligned}\theta^* = \arg\min_{\theta} \sum_{i=1}^{n_t} &\|\Psi(\mathcal{R}^{a,i}_{t_0,t_2}, \theta, \mathbf{h}^i) - \mathcal{R}^{b,i}_{t_0,t_2}\|^2 \\ &+\lambda_2 \|\Theta_{\text{int}}(\Psi(\mathcal{R}^{a,i}_{t_0,t_2}, \theta, \mathbf{h}^i), \mathbf{g}^{b,i}_{t_0}) - \mathcal{G}^{b,i}_{t_0,t_2}\|^2 \end{aligned} \quad (7)$$

where n_t is the number of training pairs; $\Psi(\cdot,\cdot,\cdot)$ denotes TPN, which takes a set of motion vector $\mathcal{R}^a$, network parameters θ and learn-able hidden parameters $\mathbf{h}$ as inputs, and outputs $\mathcal{R}^b$ for the target view. $\Theta_{\text{int}}(\cdot,\cdot)$ integrates motion vectors into 2D absolute positions according to given initial point. We can recover a predicted trajectory by

$$\Theta_{\text{int}}(\mathcal{R}^c_{t_0,t_2}, \ \mathbf{g}^c_{t_0}) = \left\{\mathbf{g}^c_t | \mathbf{g}^c_t = \mathbf{g}^c_{t_0} + \sum_{t=t_0+1}^{t} \mathbf{r}^c_t\right\}^{t_2}_{t=t_0} \quad (8)$$

Since $\mathbf{h}^i$ is also unknown in the beginning of each batch, we divide the training process into two stages:

Stage 1: We randomly initialize $\mathbf{h}^i$ and conduct network training which only optimizes $\mathbf{h}^i$ and fixes current θ for each training batch:

$$\mathbf{h}^{i^*} = \arg\min_{\mathbf{h}^i} \|\Psi(\mathcal{R}^{a,i}_{t_0,t_1}, \theta, \mathbf{h}^i) - \mathcal{R}^{b,i}_{t_0,t_1}\|^2 + \lambda_1 \|\mathbf{h}^i\|^2 \quad (9)$$

where $(\mathcal{R}^{a,i}_{t_0,t_1}, \mathcal{R}^{b,i}_{t_0,t_1})$ is first 40 frames of $(\mathcal{R}^{a,i}_{t_0,t_2}, \mathcal{R}^{b,i}_{t_0,t_2})$, which means $t_1 - t_0 = 39$. The number of frame at the first training stage of TPN is to simulate the amount of historical data used in real scenarios. If the number is low, TPN cannot learn enough information from previous trajectory. When more historical data is used, there is a greater chance that noise will be introduced. Attacking this problem, we select first 40 frames for the first training stage in our experiments.

Stage 2: We use $\mathbf{h}^{i^*}$ as initial parameters, and train networks parameter θ by using training samples in a batch.

$$\begin{aligned}\theta^*, \mathcal{H}^{**} = \arg\min_{\theta, \mathbf{h}^{i^*}} \sum_{i=1}^{n_b} &\|\Psi(\mathcal{R}^{a,i}_{t_0,t_2}, \theta, \mathbf{h}^{i^*}) - \mathcal{R}^{b,i}_{t_0,t_2}\|^2 \\ &+\lambda_2 \|\Theta_{\text{int}}(\Psi(\mathcal{R}^{a,i}_{t_0,t_2}, \theta, \mathbf{h}^{i^*}), \mathbf{g}^{b,i}_{t_0}) \\ &-\mathcal{G}^{b,i}_{t_0,t_2}\|^2 + \lambda_1 \|\mathbf{h}^{i^*}\|^2 \end{aligned}$$

where $\mathcal{H}^{**} = \{\mathbf{h}^{i**}\}^{n_b}_{i=1}$, $\mathbf{h}^{i**}$ is optimized parameter for $\mathbf{h}^{i*}$.

We use the Rprop algorithm [52] to optimize the network parameters. After 20 epochs of training on the trajectory dataset, we obtain θ^* and finish training TPN.

4) TPN in Generic Multiview Tracking: During generic multiview tracking, the situation may be complicated. There may be more than one unreliable and reliable views. At time t, for each unreliable view b (*i.e.* $q^b < \tau$), we use TPN to estimate its trajectory. We also take the result of correlation filter, $\mathbf{g}_t^b$, into account. The corrected object's location $\mathbf{g}_t^{b'}$ for camera b is given by:

$$\mathbf{g}_t^{b'} = q^{\frac{b}{2}}\mathbf{g}_t^b + \frac{1-q^{\frac{b}{2}}}{w}\sum_{c,q^c\geq\tau} q^c \Theta_{\mathrm{TP}}(\mathcal{G}_{t_0,t}^c, \mathcal{G}_{t0,t1}^b) \qquad (10)$$

where $w = \sum_{c,q^c\geq\tau} q^c$ is a normalized coefficient and $\Theta_{\mathrm{TP}}(\cdot,\cdot)$ is a trajectory prediction function, which is TPN embedded and predicts the object location $\mathbf{g}_t^b$ in camera b.

The behavior of $\Theta_{\mathrm{TP}}(\cdot,\cdot)$ is defined in the following. For the input $(\mathcal{G}_{t_0,t}^c, \mathcal{G}_{t0,t1}^b)$, $\mathcal{G}_{t_0,t_1}^c$ and $\mathcal{G}_{t0,t1}^b$ are used to train hidden parameters $\mathbf{h}^*$ according to Eq. 9; After that, we can obtain $\mathcal{G}_{t_0,t}^b = \Theta_{\mathrm{int}}(\Psi(\mathcal{R}_{t_0,t}^c, \theta, \mathbf{h}^*), \mathbf{g}_{t_0}^b)$. Finally, we take $\mathbf{g}_t^b$ as the output result of $\Theta_{\mathrm{TP}}(\cdot,\cdot)$. t_1 is the last time when view b is reliable. We choose 40 frames to train $\mathbf{h}^*$, which means $t_0 = t_1 - 39$. This equation builds connections among multiple cameras and guides the trajectory correction when occlusion occurs. We apply TPN on a pair views. Combined with Algorithm 1 (lines 17 to 22), our method is independent from the total number of the cameras.

In reality, there can be no reliable views. In that case, our algorithm simply keep the last momentum of the target object in each view. Let $\mathbf{r}_{t_1}^c$ be the motion vector at the last reliable time t_1 in view c. $\mathbf{g}_t^c = \mathbf{g}_{t-1}^c + \mathbf{r}_{t_1}^c$.

IV. Experiments

To evaluate methods in this task, we build a multiview tracking dataset and compare our tracker with others on it. Afterwards, we analyze the performance of our *Trajectory Prediction Network*. The experiment shows that the proposed networks can find out the relationship between two trajectories effectively and improve tracking performance.

A. Multiview Tracking Datasets

There exist some multiview tracking datasets. For example, the PETS2009 dataset [53], which contains sequences taking from eight cameras, is such a dataset. PETS2009, however, by itself is insufficient for convincing experimental evaluation with its low frame rate and resolution. The target objects in most of the existing dataset are either pedestrians or vehicles and other specific type of objects. Our method can track any generic object. We need some other types of objects to evaluate trackers. For this purpose, we capture and manually annotate the *Generic Multiview Tracking Dataset* (GMTD) to facilitate relevant research and evaluation.

GMTD contains a total of 17,571 frames and consists of 10 multiview sequences with each of them captured by two or three synchronized uncalibrated cameras, under 1080p resolution and 30fps. tSpecifically, *scene1*, *scene4*, *scene5*, *scene6*, *scene7*, *scene8* and *scene10* have two camera views, the others have three camera views. During data capturing, cameras are either tripod mounted or hand held. In particular, for hand-held cameras scenario, cameras may undergo small translation and rotation. Cameras are placed with different relative angles, for example, facing opposite or same directions, to form diverse trajectories in each view.

GMTD takes into account the diversity of scenarios and targets. Several different targets, including rigid ones (*e.g.* cans, lantern and basketball) and deformable ones (*e.g.* leaves, human and cat), were captured in the indoor, outdoor, artificial or natural scenes. During the acquisition process, a target may move under multiple camera views with more than 75% overlap. These 10 sequences mainly cover six aspects of challenges in visual tracking, including scale variation, motion blur, deformation, background clusters, fast motion and occlusion.

The selected target is manually annotated in each sequence by axis-aligned rectangle bounding boxes. The annotation guarantees that a target occupies more than 60% area of the bounding box. For further analysis, we also label target state in each frame as fully-visible, partially-occluded (33.73% per sequence on average) or fully-occluded (4.78% per sequence on average). Object is occluded by 20% to 80% in partially-occluded scenario. Others are fully-visible (if below 20%) or fully-occluded (if beyond 80%). Bounding boxes are predicted by human when the object is occluded.

B. Evaluation Methodology

We evaluate our method in two ways, with (see [54]) and without re-initialization respectively.

1) With Re-Initialization: Based on widely-used tracking performance measurements, we choose two easily interpretable measurements to evaluate methods, which are accuracy and robustness. When evaluating with re-initialization, 'Accuracy' refers to the area, in percentage, of the results overlaps with the ground truth and 'Robustness' is a probability of tracker failing after S frames (see [55]).

Traditional trackers only track object in a single view, so we test them individually on each camera view of each scene. Then we can obtain average accuracy ρ^c and average robustness σ^c for camera view c in a scene. We measure the performance of a tracker in this scene by averaging them. So we can get $\overline{\rho} = \frac{1}{n_c}\sum \rho^c$ and $\overline{\sigma} = \frac{1}{n_c}\sum \sigma^c$. Although our method tracks all video sequences in a scene simultaneously, we can also calculate $\overline{\rho}$ and $\overline{\sigma}$ for our tracker.

For more details, the target bounding boxes in all views will be reset to the next nearest fully-visible frame once IoU (Intersection over Union) drops to zero (the tracking result has no overlapping with ground truth bounding box) in any view during re-initialization. The tracker will be initialized by using ground truth bounding boxes and frame images at the same time. Let $a_{i,t}^c$ denote the IoU in view c at time t in scene i, $\mathcal{V}_i = \{t|\forall c, a_{i,t}^c > 0\}$ the valid set. The per-scene accuracy ρ_i for scene i is defined as $\rho_i = \frac{1}{n_c|\mathcal{V}_i|}\sum_{t\in\mathcal{V}_i}\sum_c^{n_c} a_{i,t}^c$. We run a tracker 5 times for each scene to obtain average accuracy $\overline{\rho_i}$. Thus, the overall average accuracy $\overline{\rho}$ is obtained by the weighted average as $\overline{\rho} = \frac{1}{\sum_i |\mathcal{V}_i|}\sum_i |\mathcal{V}_i| \cdot \overline{\rho_i}$.

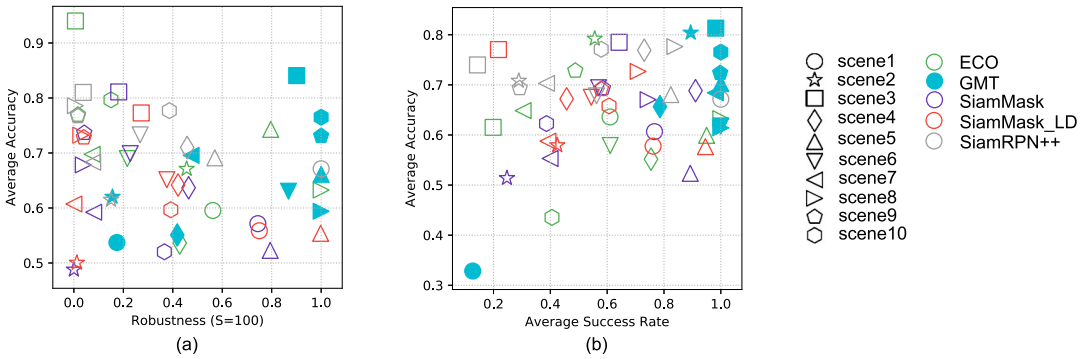

Fig. 4. Evaluation for cases **with** re-initialization (left) and **without** re-initialization (right) on GMTD. Hollow shapes in different colors represent results of different trackers, and solid ones for GMT. A tracker is better if it resides close to the top-right corner.

TABLE I
EVALUATION RESULTS ON GMTD WITH RE-INITIALIZATION

sequence	σ_{ECO}	ρ_{ECO}	σ_{SM}	ρ_{SM}	σ_{SM_LD}	ρ_{SM_LD}	$\sigma_{SiamRPN++}$	$\rho_{SiamRPN++}$	σ_{GMT}	ρ_{GMT}
scene1	0.562	0.595	0.744	0.572	0.750	0.559	**1.000**	**0.671**	0.174	0.537
scene2	**0.456**	**0.671**	0.000	0.488	0.012	0.500	0.150	0.615	0.156	0.620
scene3	0.006	**0.940**	0.181	0.811	0.272	0.773	0.038	0.810	**0.901**	0.841
scene4	0.428	0.536	**0.464**	0.637	0.421	0.642	0.458	**0.710**	0.418	0.551
scene5	0.796	**0.743**	0.794	0.523	0.998	0.554	0.570	0.691	**1.000**	0.661
scene6	0.216	0.690	0.228	0.699	0.376	0.651	0.268	**0.733**	**0.868**	0.630
scene7	0.074	**0.698**	0.080	0.592	0.000	0.607	0.076	0.683	**0.474**	0.696
scene8	**1.000**	0.633	0.038	0.678	0.028	0.732	0.006	**0.786**	**1.000**	0.594
scene9	0.016	**0.769**	0.042	0.736	0.040	0.728	0.016	0.767	**1.000**	0.731
scene10	0.150	**0.797**	0.366	0.520	0.392	0.597	0.386	0.777	**1.000**	0.765
Weighted mean	0.299	**0.754**	0.265	0.660	0.312	0.660	0.242	0.741	**0.748**	0.698

We will have n_f valid tracking fragments in a scene due to re-initialization. A tracking fragment is counted from the initial/re-initialized frame to the frame where tracking failed (IoU drops to zero in any view). The robustness is characterized as the probability of tracker failing after S frames (in this evaluation $S = 100$). Let u_i denote the length of i-th tracking fragment in this view and l represent the total length of this scene. Then the robustness is defined as:

$$\sigma = \frac{\sum_{i=0}^{n_f} \max(u_i - S, 0)}{l - S}. \tag{11}$$

The robustness will be higher if tracker can keep tracking more time. The overall average robustness is the weighted average of per-scene robustness by the length of scene.

We visualize results in accuracy-robustness(AR) plots. In AR plots, each tracker is represented as a point in terms of its overall averaged accuracy and robustness on GMTD dataset. Comparatively speaking, The tracker performs better if it is located in the top-right part of the plot and worse if it occupies the bottom-left part.

2) Without Re-Initialization: To simulate a more realistic tracking environment, we also test relevant trackers without re-initialization. Under these circumstances, the accuracy is defined as the same as re-initialization case. We denote $\delta_i = \frac{|\mathcal{V}_i|}{l_i}$ as the success rate of scene i, where l_i counts the total number of frames of the view in scene i. Similarly, tracker is visualized in a plot with respect to average accuracy and average success rate of each scene.

C. Results on the GMTD Dataset

We compare our method with ECO tracker [3], which is a typical example of correlation based tracker. Further more, we also compare our method with the latest siamese network-based trackers, such as SiamMask [56] and SiamRPN++ [47], on GMTD dataset and CAMPUS [38] dataset. Quantitative results can be seen in Figure 4 and Tables I, II and III. In our experiments, our tracker parameters are fixed and set $\tau = 0.5$.

1) With Re-Initialization: The overall averaged accuracy of GMT is 0.6984 and the overall average robustness is **0.7477**. On ECO tracker, the overall averaged accuracy is **0.7541** and the overall average robustness is 0.2985. We can see a huge improvement on robustness in most scenes. Results can be seen in Figure 4(a) and Table I. Few scenes get a worse robustness, because correlation filter tracker has already lost in the beginning. Wrong tracking trajectory history in a view will lead up to the wrong prediction of TPN. Similar phenomena can also be observed in the following experiments.

2) Without Re-Initialization: In practice, tracking algorithms in the real world cannot conduct re-initialization scheme due to the absence of ground truth. We cannot track again once the algorithm loses track of target object. Thus, we also conduct evaluation without re-initialization to evaluate the long-term performance of algorithms, which is closer to the real situation. The results are shown in Figure 4(b) and Table II. Our method can achieve the highest success rates in most of the scenes except in scene1. That is because GMT

TABLE II
EVALUATION RESULTS ON GMTD WITHOUT RE-INITIALIZATION

sequence	δ_{ECO}	ρ_{ECO}	δ_{SM}	ρ_{SM}	δ_{SM_LD}	ρ_{SM_LD}	$\delta_{SiamRPN++}$	$\rho_{SiamRPN++}$	δ_{GMT}	ρ_{GMT}
scene1	0.610	0.636	0.767	0.607	0.762	0.578	**1.000**	**0.671**	0.127	0.328
scene2	0.556	0.792	0.247	0.514	0.424	0.579	0.289	0.708	**0.894**	**0.804**
scene3	0.199	0.615	0.642	0.785	0.217	0.771	0.144	0.740	**0.982**	**0.813**
scene4	0.754	0.552	**0.911**	0.688	0.457	0.672	0.731	**0.769**	0.786	0.657
scene5	0.950	0.599	0.893	0.523	0.946	0.577	0.824	0.681	**1.000**	**0.703**
scene6	0.611	0.579	0.570	**0.694**	0.544	0.675	0.563	0.678	**1.000**	0.618
scene7	0.304	0.649	0.399	0.553	0.392	0.588	0.387	**0.703**	**0.981**	0.684
scene8	**1.000**	0.633	0.744	0.670	0.709	0.727	0.838	**0.776**	**1.000**	0.614
scene9	0.488	**0.728**	0.585	0.693	0.578	0.692	0.292	0.693	**0.998**	0.724
scene10	0.406	0.435	0.387	0.623	0.607	0.658	0.579	**0.771**	**1.000**	0.765
Weighted mean	0.513	0.624	0.611	0.661	0.506	0.673	0.481	**0.721**	**0.900**	0.697

Fig. 5. Demonstrations of trackers' results on 'scene2' of GMTD without Re-initialization. Ground truth bounding boxes are drawn in green. Tracker's results are drawn in orange. Each row shows results of close moments for each tracker.

failed initially, which will mislead TPN to output the wrong prediction. Our method also gets a competitive accuracy in most cases.

Figure 5, 6 and 7 illustrates some visualized results on video clips of GMTD. Here we compare our tracker with state-of-the-art ones. In these figures, we only show one camera view of a scene. Occlusions in these clips make the majority of trackers fail to track the target object. Our tracker (GMT) can still keep tracking with the help of TPN when serious occlusion occurs.

D. Results on the CAMPUS Dataset

We also test tracking algorithms on CAMPUS dataset. This dataset is designed for multi-view object tracking algorithms to test on dense foreground, complex scenarios, various object scales. It has around 15-25 pedestrians, frequent conjunctions and occlusions. Pedestrians in this dataset conduct diverse activities in front of dynamic background, and have interactions between objects and background. There are 4 sequences in this dataset: Garden 1, Garden 2, Auditorium and Parking Lot. Each sequence contains 3-4 cameras at 30fps 1080p configuration and have a duration about 3-4 minutes. Each camera covers both overlapping regions and non-overlapping regions with other cameras. In addition to two cameras (view-HC3 and view-HC4) of Auditorium sequence, which have not enough overlapping regions, we use all cameras in each sequence for testing.

We evaluate relevant trackers without re-initialization. The result is shown in Table III. We can see a higher

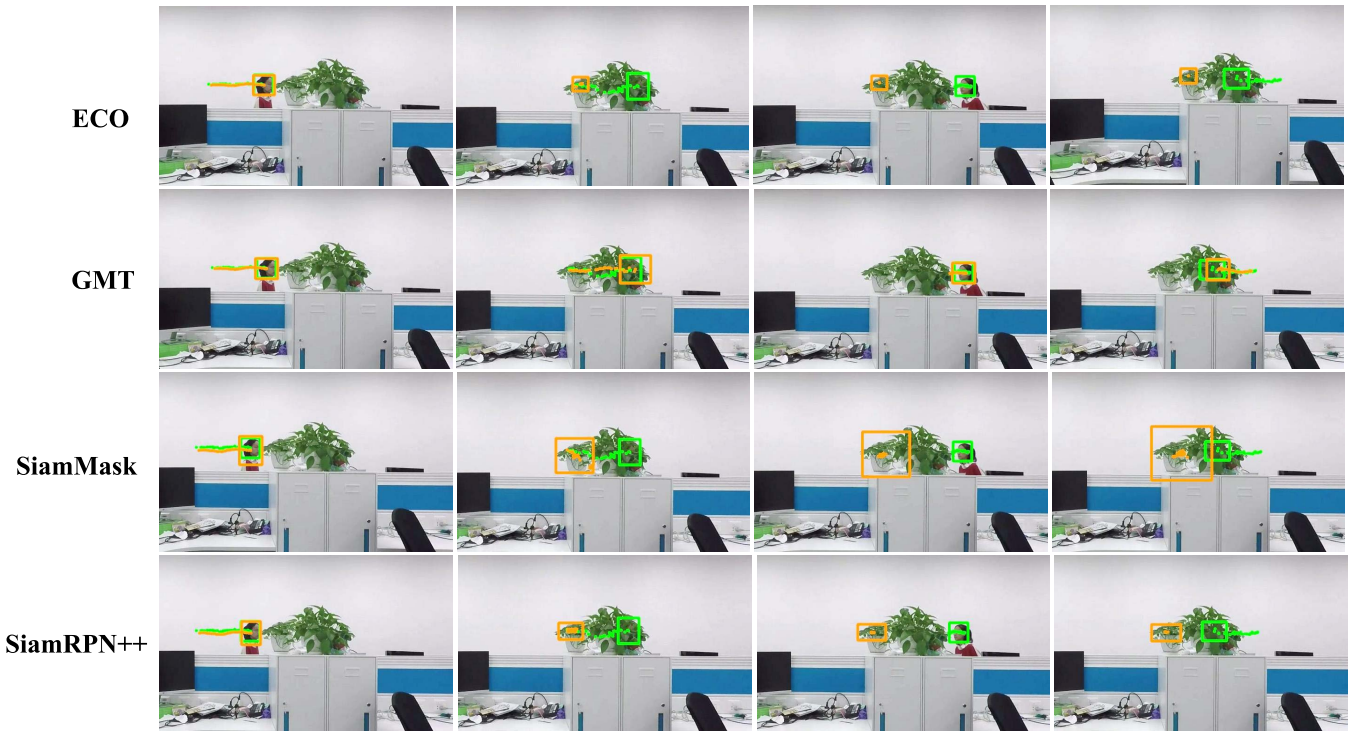

Fig. 6. Demonstrations of trackers' results on 'scene7' of GMTD without Re-initialization. Ground truth bounding boxes are drawn in green. Tracker's results are drawn in orange. Each row shows results of close moments for each tracker.

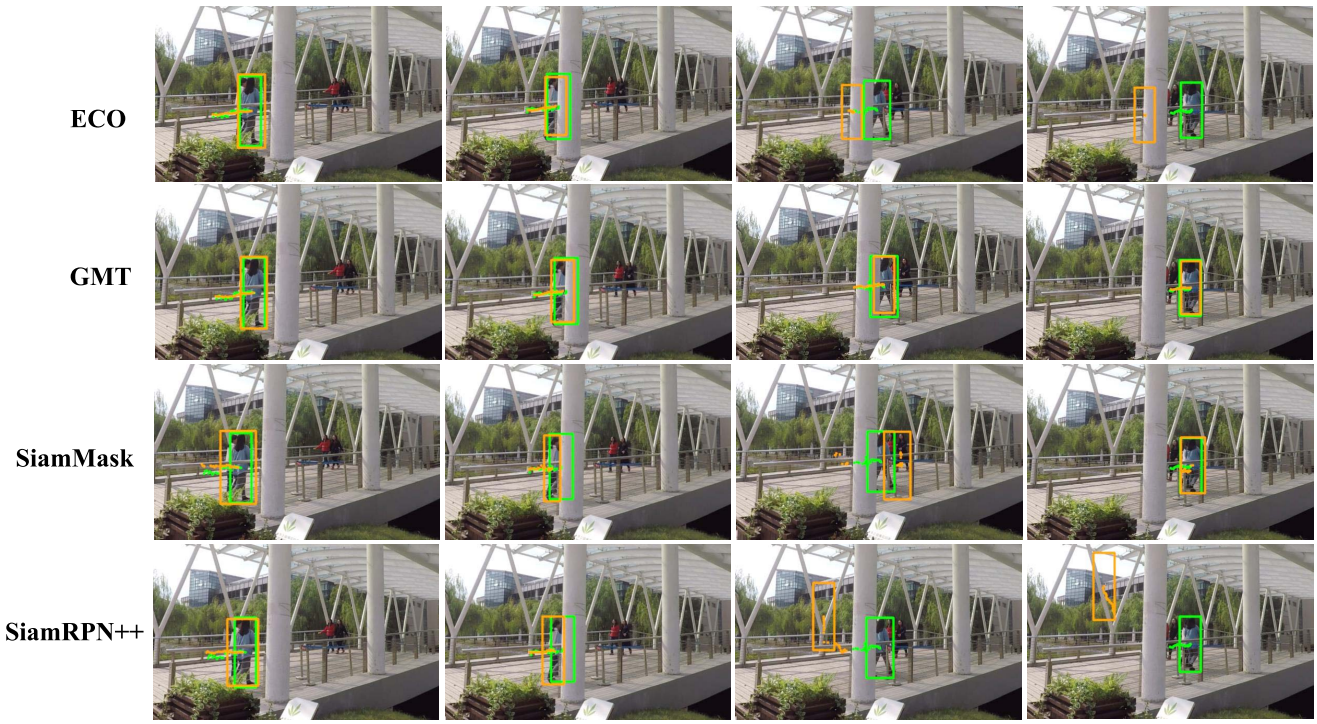

Fig. 7. Demonstrations of trackers' results on 'scene9' of GMTD without Re-initialization. Ground truth bounding boxes are drawn in green. Tracker's results are drawn in orange. Each row shows results of close moments for each tracker.

success rate in most cases. As mentioned above, TPN needs trajectory histories to predict current position. Some failure initialization, such as partial occlusion in initial target bounding box and tracking failures in the initial stage, will cause prediction errors, especially in Auditorium sequence.

TABLE III

EVALUATION RESULTS ON CAMPUS [38]. ACCURACY AND SUCCESS RATE OF TRACKERS ON EACH SEQUENCE

sequence	δ_{ECO}	ρ_{ECO}	δ_{SM}	ρ_{SM}	δ_{SM_LD}	ρ_{SM_LD}	$\delta_{SiamRPN++}$	$\rho_{SiamRPN++}$	δ_{GMT}	ρ_{GMT}
Auditorium	**0.685**	0.530	0.638	0.533	0.637	0.560	0.606	0.529	0.676	**0.572**
Garden1	0.824	0.713	0.874	0.659	0.827	0.661	0.898	0.712	**0.909**	**0.745**
Garden2	0.819	0.701	0.737	0.652	0.748	0.647	0.776	0.693	**0.875**	**0.715**
Parkinglot	0.708	0.621	0.709	0.590	0.683	0.594	**0.738**	**0.634**	0.703	0.616

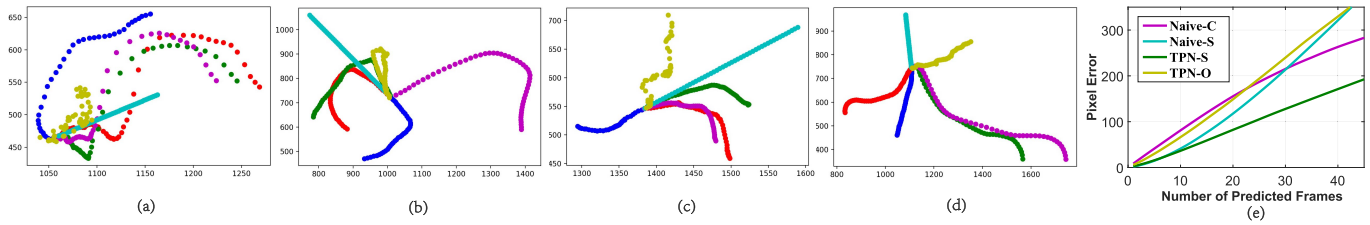

Fig. 8. Trajectory prediction evaluation. In (a-d), blue trajectories are $\mathcal{G}^b_{t_0,t_1}$ and red trajectories are $\mathcal{G}^b_{t_1+1,t_2}$. We compare predictions of different methods shown in different colors. The color correspondence can be seen in the legend. **(a)** TPN-S and naive-C predict well while others deviating from the real trajectory. **(b)** The moving directions in two views are opposite, making naive-C tend to failed. **(c)** TPN-S may have deviations from $\mathcal{G}^b_{t_1+1,t_2}$. **(d)** Methods may fail when predicting at the turning point of trajectory. **(e)** The plot of average prediction errors with the number of predicted frames.

E. Trajectory Prediction Evaluation

We evaluate our proposed TPN on trajectories' test dataset and compare TPN with naive methods. Moreover, we also test variants of TPN to show that the current structure (TPN-S) is comparatively optimal.

We provide two naive methods for trajectory prediction. One copies speeds of reference view and integrates them into the trajectory of current view (Naive-C). The other one simply repeats the last average speed of current view's trajectory (Naive-S). Variants of TPN contain the standard TPN (TPN-S) described in section III-E. Another variant's hidden parameters are removed and all network parameters are trained online during tracking (TPN-O).

We simulate trajectory predictions on test dataset in online tracking between two views, a and b, to evaluate these methods. During each simulation, we sample a trajectory pair $(\mathcal{G}^a_{t_0,t_2}, \mathcal{G}^b_{t_0,t_2})$ of two views from time t_0 to time t_2 $(t_2 - t_0 = 89)$. The trajectory pair is divided into two parts. The first part is $(\mathcal{G}^a_{t_0,t_1}, \mathcal{G}^b_{t_0,t_1})$, where $t_1 - t_0 = 39$. We train these models by using this pair. Then we use the left trajectory $\mathcal{G}^a_{t_1+1,t_2}$ of reference view a and trained model to predict the trajectory of the other view b, denoted $\mathcal{G}^{b*}_{t_1+1,t_2}$. After that, we use ground truth $\mathcal{G}^b_{t_1+1,t_2}$ to calculate the pixel distances of each frame between $\mathcal{G}^b_{t_1+1,t_2}$ and $\mathcal{G}^{b*}_{t_1+1,t_2}$. In order to prevent potential variance of performance in evaluation, we repeat the simulation 1000 times and obtain the statistic average error of pixels for predicted position in each frame.

Results are shown in Figure 8. This evaluation shows that our proposed TPN-S has the best performance in predicting trajectory from the reference view. Respectively, TPN-O, whose hidden layers are removed, suffers from over-fitting problem. Moreover, TPN-O updates all parameters online, which leads to more computational costs. There still are some failure cases for all methods, such as (d) in Figure 8. It failed because $\mathcal{G}^b_{t_0,t_1}$ lacks sufficient information to infer the relationship between cameras.

V. CONCLUSION

In this paper we propose a novel *generic multiview tracker* (GMT) for visual object tracking with multi-camera inputs. Unlike most previous multiview tracking systems, our GMT requests little prior knowledge about the tracking target object, allows camera movement, and is calibration free. Our GMT has two novel components, a cross-view *trajectory prediction network* and *collaborative correlation filter*, which are effectively integrated with a correlation filter tracking framework. For evaluation, we contribute a *generic multiview tracking dataset* to alleviate the lack of proper multiview tracking benchmarks. In our carefully designed experiments, except for initial tracking failure in some scenarios, which is the limitation, the proposed tracking algorithm demonstrates advantages over state-of-the-art tracking algorithms.

REFERENCES

[1] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.
[2] M. Danelljan, A. Robinson, F. S. Khan, and M. Felsberg, "Beyond correlation filters: Learning continuous convolution operators for visual tracking," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 472–488.
[3] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, "ECO: Efficient convolution operators for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017. vol. 1, p. 3.
[4] S. Hare *et al.*, "Struck: Structured output tracking with kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 2096–2109, Oct. 2016.
[5] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4293–4302.
[6] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8971–8980.
[7] L. Wen, Z. Lei, M.-C. Chang, H. Qi, and S. Lyu, "Multi-camera multi-target tracking with Space-Time-View hyper-graph," *Int. J. Comput. Vis.*, vol. 122, no. 2, pp. 313–333, Apr. 2017.
[8] E. Ristani and C. Tomasi, "Features for multi-target multi-camera tracking and re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6036–6046.

[9] K. Yoon, Y.-M. Song, and M. Jeon, "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views," *IET Image Process.*, vol. 12, no. 7, pp. 1175–1184, Jul. 2018.

[10] D. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2544–2550.

[11] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, "Discriminative scale space tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1561–1575, Aug. 2017.

[12] G. Bhat, J. Johnander, M. Danelljan, F. Shahbaz Khan, and M. Felsberg, "Unveiling the power of deep tracking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 483–498.

[13] M. Danelljan, G. Häger, F. Shahbaz Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *Proc. Brit. Mach. Vis. Conf.*, 2014, p. 154.

[14] K. Chen and W. Tao, "Once for all: A two-flow convolutional neural network for visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 12, pp. 3377–3386, Dec. 2018.

[15] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1420–1429.

[16] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional Siamese networks for object tracking," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 850–865.

[17] Q. Wang, J. Gao, J. Xing, M. Zhang, and W. Hu, "DCFNet: Discriminant correlation filters network for visual tracking," 2017, *arXiv:1704.04057*. [Online]. Available: http://arxiv.org/abs/1704.04057

[18] X. Wang, C. Li, B. Luo, and J. Tang, "SINT++: Robust visual tracking via adversarial positive instance generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4864–4873.

[19] H. Fan and H. Ling, "Siamese cascaded region proposal networks for real-time visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7952–7961.

[20] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5388–5396.

[21] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Adaptive correlation filters with long-term and short-term memory for object tracking," *Int. J. Comput. Vis.*, vol. 126, no. 8, pp. 771–796, Aug. 2018.

[22] C. Sun, D. Wang, H. Lu, and M.-H. Yang, "Learning spatial-aware regressions for visual tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8962–8970.

[23] Z. Zhu, W. Wu, W. Zou, and J. Yan, "End-to-End flow correlation tracking with spatial-temporal attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 548–557.

[24] A. He, C. Luo, X. Tian, and W. Zeng, "A twofold siamese network for real-time object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4834–4843.

[25] K. Chen, W. Tao, and S. Han, "Visual object tracking via enhanced structural correlation filter," *Inf. Sci.*, vols. 394–395, pp. 232–245, Jul. 2017.

[26] W. Chen, K. Zhang, and Q. Liu, "Robust visual tracking via patch based kernel correlation filters with adaptive multiple feature ensemble," *Neurocomputing*, vol. 214, pp. 607–617, Nov. 2016.

[27] Y. Li, J. Zhu, and S. C. H. Hoi, "Reliable patch trackers: Robust visual tracking by exploiting reliable patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 353–361.

[28] T. Liu, G. Wang, and Q. Yang, "Real-time part-based visual tracking via adaptive correlation filters," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4902–4912.

[29] X. Dong, J. Shen, D. Yu, W. Wang, J. Liu, and H. Huang, "Occlusion-aware real-time object tracking," *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 763–771, Apr. 2017.

[30] X. Mei, H. Ling, Y. Wu, E. Blasch, and L. Bai, "Minimum error bounded efficient ℓ_1 tracker with occlusion detection," in *Proc. CVPR*, Jun. 2011, pp. 1–7.

[31] M. Guan, C. Wen, M. Shan, C.-L. Ng, and Y. Zou, "Real-time event-triggered object tracking in the presence of model drift and occlusion," *IEEE Trans. Ind. Electron.*, vol. 66, no. 3, pp. 2054–2065, Mar. 2019.

[32] S. M. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 133–146.

[33] C. H. Kuo, C. Huang, and R. Nevatia, "Inter-camera association of multi-target tracks by on-line learned appearance affinity models," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 383–396.

[34] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2004, p. 2.

[35] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps, "Dynamic subspace-based coordinated multicamera tracking," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2462–2469.

[36] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese, "Monocular multiview object tracking with 3d aspect parts," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 220–235.

[37] Q. C. Le, D. Conte, and M. Hidane, "Online multiple view tracking: Targets association across cameras," in *Proc. 6th Workshop Activity Monitor. Multiple Distrib. Sens.*, 2018, pp. 1–9.

[38] Y. Xu, X. Liu, Y. Liu, and S.-C. Zhu, "Multi-view people tracking via hierarchical trajectory composition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4256–4265.

[39] Y. Xu, X. Liu, L. Qin, and S.-C. Zhu, "Cross-view people tracking by scene-centered spatio-temporal parsing," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 4299–4305.

[40] A. Hanif, A. B. Mansoor, and A. S. Imran, "Deep multi-view correspondence for identity-aware multi-target tracking," in *Proc. Int. Conf. Digit. Image Comput.*, 2017, pp. 1–8.

[41] X. Li, Q. Liu, Z. He, H. Wang, C. Zhang, and W.-S. Chen, "A multi-view model for visual tracking via correlation filters," *Knowl.-Based Syst.*, vol. 113, pp. 88–99, Dec. 2016.

[42] X. Mei, Z. Hong, D. Prokhorov, and D. Tao, "Robust multitask multiview tracking in videos," *IEEE Trans. neural Netw. Learn. Syst.*, vol. 26, no. 11, pp. 2874–2890, Feb. 2015.

[43] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[44] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[45] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7291–7299.

[46] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 34–50.

[47] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "SiamRPN++: Evolution of siamese visual tracking with very deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4282–4291.

[48] M. Kristan *et al.*, "The visual object tracking VOT2017 challenge results," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 1949–1972.

[49] Y. Wu, J. Lim, and M. H. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[50] H. Fan *et al.*, "LaSOT: A high-quality benchmark for large-scale single object tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5374–5383.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[52] M. Riedmiller and H. Braun, "Rprop-a fast adaptive learning algorithm," in *Proc. 7th ISCIS*. Citeseer, 1992.

[53] A. Ellis, A. Shahrokni, and J. M. Ferryman, "PETS2009 and winter-PETS 2009 results: A combined evaluation," in *Proc. 12th IEEE Int. Workshop Perform. Eval. Tracking Surveill.*, Dec. 2009, pp. 1–8.

[54] M. Kristan *et al.*, "A novel performance evaluation methodology for single-target trackers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 11, pp. 2137–2155, Nov. 2016.

[55] L. Cehovin, M. Kristan, and A. Leonardis, "Is my new tracker really better than yours?" in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2014, pp. 540–547.

[56] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1328–1338.

Minye Wu received the B.S. degree from the School of Computer Engineering and Science, Shanghai University, China, in 2015. He is currently pursuing the Ph.D. degree with ShanghaiTech University. His research interests include computer vision, deep learning, and computational photography.

Haibin Ling (Senior Member, IEEE) received the B.S. and M.S. degrees from Peking University in 1997 and 2000, respectively, and the Ph.D. degree from the University of Maryland at College Park, College Park, in 2006. From 2000 to 2001, he was an Assistant Researcher with Microsoft Research Asia. From 2006 to 2007, he worked as a Postdoctoral Scientist at the University of California at Los Angeles. In 2007, he joined Siemens Corporate Research Inc., as a Research Scientist. From 2008 to 2019, he worked as a Faculty Member at the Department of Computer Sciences, Temple University. In Fall 2019, he joined Stony Brook University as a SUNY Empire Innovation Professor with the Department of Computer Science. His research interests include computer vision, augmented reality, medical image analysis, and human–computer interaction. He received the Best Student Paper Award at ACM UIST in 2003, the NSF CAREER Award in 2014, the Yahoo Faculty Research and Engagement Program Award in 2019, and the Amazon AWS Machine Learning Research Award in 2019. He serves as an Associate Editors for several journals, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI), *Pattern Recognition* (PR), and *Computer Vision and Image Understanding* (CVIU). He has served as the Area Chair various times for CVPR and ECCV.

Ning Bi received the B.Sc. degree in computer science from ShanghaiTech University, China, in 2019. During the B.Sc. degree, she mainly studied computer vision. She is currently pursuing the Ph.D. degree with the Computational Imaging and Simulation Technologies in Biomedicine (CISTIB) Group, co-supervised by Prof. Alex Frangi and Dr. Ali Gooya. She joined the Shanghaitech Vision and Intelligent Perception (SVIP) LAB as a Research Assistant. The projects she enrolled are focused on developing deep-learning methods in object detection and object tracking. She received the Full Scholarship to carry out research on motion prediction and anomaly detection on cardiovascular disease (CVD) at CISTIB.

Shenghua Gao (Member, IEEE) received the B.Eng. degree (Hons.) from the University of Science and Technology of China in 2008 and the Ph.D. degree from Nanyang Technological University in 2012. From June 2012 to July 2014, he was a Post-Doctoral Fellow with the Advanced Digital Sciences Center, Singapore. He is currently an Assistant Professor with ShanghaiTech University, China. His research interests include computer vision and machine learning.

Qiang Hu (Member, IEEE) received the B.S. degree in electrical and information engineering from the University of Electronic Science and Technology of China in 2013 and the Ph.D. degree in information and communication engineering from Shanghai Jiao Tong University, Shanghai, China, in 2019. He is currently a Postdoctoral Researcher with the Group of Prof. Jingyi Yu at the School of Information Science and Technology (SIST), ShanghaiTech University, Shanghai. His current research interests include image and video coding and perceptual video coding and processing.

Hao Sheng (Member, IEEE) received the B.S. and Ph.D. degrees from the School of Computer Science and Engineering, Beihang University, China, in 2003 and 2009, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Beihang University. He is working on computer vision, pattern recognition, and machine learning.

Jingyi Yu (Senior Member, IEEE) received the B.S. degree from Caltech in 2000 and the Ph.D. degree from MIT in 2005. He is currently a Professor and the Executive Dean of the School of Information Science and Technology, ShanghaiTech University. He is also affiliated with the Department of Computer and Information Sciences, University of Delaware. He has published over 120 papers at highly refereed conferences and journals and holds over 10 international patents on computational imaging. His research interests include computer vision and computer graphics, especially on computational photography and nonconventional optics and camera designs. He was a recipient of the NSF CAREER Award and the AFOSR YIP Award. He has served as the Area Chair of many international conferences, including CVPR, ICCV, ECCV, ICCP, and NIPS. He is an Associate Editor of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), the IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), and CVIU (Elsevier) and will be Program Chair of ICPR 2020 and the IEEE CVPR 2021.