Single-Frame-Based Deep View Synchronization for Unsynchronized Multicamera Surveillance

Qi Zhang, *Student Member, IEEE*, and Antoni B. Chan, *Senior Member, IEEE*

***Abstract*— Multicamera surveillance has been an active research topic for understanding and modeling scenes. Compared to a single camera, multicameras provide larger field-of-view and more object cues, and the related applications are multiview counting, multiview tracking, 3-D pose estimation or 3-D reconstruction, and so on. It is usually assumed that the cameras are all temporally synchronized when designing models for these multicamera-based tasks. However, this assumption is not always valid, especially for multicamera systems with network transmission delay and low frame rates due to limited network bandwidth, resulting in desynchronization of the captured frames across cameras. To handle the issue of unsynchronized multicameras, in this article, we propose a synchronization model that works in conjunction with existing deep neural network (DNN)-based multiview models, thus avoiding the redesign of the whole model. We consider two variants of the model, based on where in the pipeline the synchronization occurs, scene-level synchronization and camera-level synchronization. The view synchronization step and the task-specific view fusion and prediction step are unified in the same framework and trained in an end-to-end fashion. Our view synchronization models are applied to different DNNs-based multicamera vision tasks under the unsynchronized setting, including multiview counting and 3-D pose estimation, and achieve good performance compared to baselines.**

***Index Terms*— Crowd counting, deep learning, image matching, pose estimation, surveillance.**

I. Introduction

COMPARED to single cameras, multicamera networks allow better understanding and modeling of the 3-D world through more dense sampling of information in a 3-D scene [1]. Multicamera based vision tasks have been a popular research field, especially deep learning-related tasks, such as 3-D pose estimation from multiple 2-D observations [2], [3], 3-D reconstruction [4], [5], multiview tracking [6]–[8], multiview crowd counting [9], and re-identification (ReID) [10]–[14]. Usually, it is assumed that the multicameras are temporally synchronized when designing deep neural networks (DNNs) models, i.e., all cameras capture images at the same time point. However, the synchronization assumption for multicamera systems may not always be valid in *practical applications* due to a variety of reasons, such as dropped camera frames due to limited network bandwidth or system resources, network transmission delays, and so on. Other examples of situations where camera synchronization is not possible include: 1) using images captured from different camera systems; 2) using images from social media to reconstruct the crowd at an event; and 3) performing 3-D reconstruction of a dynamic scene using video from a drone. Thus, handling unsynchronized multicameras is an important issue in the adoption and practical usage of multiview computer vision.

There are several possible methods to fix the problem of unsynchronized cameras. The first method is using hardware-based solutions to synchronize the capture times, such as improving network bandwidth, or by using a central clock to synchronize capture of all cameras in the multicamera network. However, this will increase the cost and overhead of the system, and is not possible when there is limited bandwidth. The second method is to capture image sequences from each camera, and then synchronize the images afterward by determining the frame offset between cameras. The fineness of the synchronization depends on the frame rate of the image sequences. However, this method is not effective when acquiring high frame-rate image sequences is not possible due to limited the bandwidth and storage space, or the frame latency between multicameras is random. The final method is to modify the multiview model to handle unsynchronized images, especially for low-frame-rate multicamera systems or random frame latency between multicameras, such as introducing new assumptions or relaxing the original constraints under the unsynchronized setting. Existing approaches for handling unsynchronized multicameras are largely based on optimization frameworks [15], [16], but are not directly applicable to DNNs-based multiview methods, which have seen recent successes in tracking [6], [7], 3-D pose estimation [2], and crowd counting [9], [17].

In this article, we propose a synchronization model that operates in conjunction with existing DNN-based multiview models by using single frames from each camera to deal with low-frame-rate unsynchronized multicamera systems or random frame latency between multicameras. Our proposed model first synchronizes other views to a reference view using a differentiable module, and then the synchronized multiviews features are fused and decoded to obtain the task-oriented output. As illustrated in Fig. 1, the synchronization can either

Manuscript received July 22, 2021; revised January 4, 2022 and March 22, 2022; accepted April 15, 2022. This work was supported by Research Grant Council of Hong Kong Special Administrative Region (SAR), China, under Grant TR32-101/15-R and Grant CityU 11212518. *(Corresponding author: Qi Zhang.)*

Qi Zhang is with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, Guangdong 518060, China (e-mail: qzhang364-c@my.cityu.edu.hk).

Antoni B. Chan is with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China (e-mail: abchan@cityu.edu.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3170642.

Digital Object Identifier 10.1109/TNNLS.2022.3170642

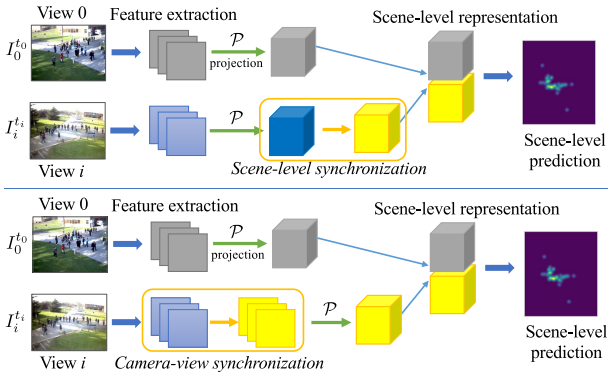

Fig. 1. Two variants of the main pipeline for unsynchronized multiview prediction tasks: (top) SLS is performed after the projection on the scene-level feature representations and (bottom) CLS is performed on the camera-view feature maps before projection.

occur after the camera-to-scene (2-D-to-3-D) projection [Fig. 1 (top)] or before the projection [Fig. 1 (bottom)]. Thus, to fully explore these options, we consider two variants of our model that perform synchronization at different stages in the pipeline (see Fig. 2): 1) scene-level synchronization (SLS) performs the synchronization after projecting the camera features to their 3-D scene representation and 2) camera-level synchronization (CLS) performs the synchronization between camera views first, and then projects the synchronized 2-D feature maps to their 3-D representations. In both cases, motion flow between the cameras' feature maps are estimated and then used to warp the feature maps to align with the reference view (either at the scene-level or the camera-level). With both variants, the view synchronization and the multiview fusion are unified in the same framework and trained in an end-to-end fashion. In this way, the original DNN-based multiview model can be adapted to work in the unsynchronized setting by adding the view synchronization module, thus avoiding the need to design a new model. Furthermore, the synchronization module only relies on content-based image matching and camera geometry, and thus is widely applicable to many DNNs-based multiview tasks, such as crowd counting, tracking, 3-D pose estimation, and 3-D reconstruction.

In summary, the contributions of this article are threefold.

1) We propose an end-to-end trainable framework to handle the issue of unsynchronized multicamera images in DNNs-based multicamera vision tasks. To the best of our knowledge, this is the first study on DNNs-based *single-frame* synchronization of multiview cameras.
2) We propose two synchronization modules, SLS and camera-view level synchronization, which are based on image-based content matching that is guided by epipolar geometry. The synchronization modules can be applied to many different DNNs-based multiview tasks.
3) We conduct experiments on multiview counting and 3-D pose estimation from unsynchronized images, demonstrating the efficacy of our approach.

The remainder of this article is organized as follows. We review related works in Section II. In Section III, we propose our single-frame multicamera synchronization methods, and in Section IV we present experiments on two applications, multiview crowd counting, and multiview 3-D human pose estimation. Finally, Section V concludes the article.

II. Related Work

In this section, we review DNN-based methods on synchronized multiview images and unsynchronized multiview video tasks, as well as traditional multiview video synchronization methods. We then review DNN-based image matching and flow estimation methods.

A. DNN-Based Synchronized Multicamera Tasks

Multicamera surveillance based on DNNs has been an active research area. By utilizing multiview cues and the strong mapping power of DNNs, many DNNs models have been proposed to solve multiview surveillance tasks, such as multiview tracking and detection [6], [7], [18], crowd counting [9], 3-D reconstruction [4], [5], [19], [20], and 3-D human pose estimation [2], [21]–[24]. Kar *et al.* [4] proposed a deep learning 3-D reconstruction framework with differentiable feature projection and unprojection steps. Ye *et al.* [10] proposed the collaboration ensemble learning for ReID with middle-level sharable two-stream network. Iskakov *et al.* [2] proposed volumetric aggregation of feature maps for 3-D pose estimation. The DNN pipelines used for these multicamera tasks can be generally divided into three stages: the single-view feature extraction stage, the multiview fusion stage to obtain a scene-level representation, and prediction stage. Furthermore, all these DNN-based methods assume that the input multiviews are synchronized, which is not always possible in real multicamera surveillance systems, or in multiview data from disparate sources (e.g., crowd-sourced images). Therefore, relaxing the synchronization assumption can allow more practical applications of multicamera vision tasks in real world.

B. Tasks on Unsynchronized Multicamera Video

Only a few works have considered computer vision tasks on unsynchronized multicamera *video*. Zheng *et al.* [15] posed the estimation of 3-D structure observed by multiple unsynchronized video cameras as the problem of dictionary learning. Zhang *et al.* [16] proposed a multicamera motion segmentation method using unsynchronized videos by combining shape and dynamical information. Takahashi *et al.* [25] proposed a method of estimating 3-D human pose from multiview videos captured by unsynchronized and uncalibrated cameras by utilizing the projections of joint as the corresponding points. Albl *et al.* [26] presented a method for simultaneously estimating camera geometry and time shift from video sequences from multiple unsynchronized cameras using minimal correspondence sets. Kuo *et al.* [27] addressed the problem of aligning unsynchronized camera views by low and/or variable frame rates using the intersections of corresponding object trajectories to match views.

Note that all these methods assume that videos or image sequences are available to perform the synchronization. In contrast, our framework, which is motivated by practical low-fps systems, is solving a harder problem, where *only a single*

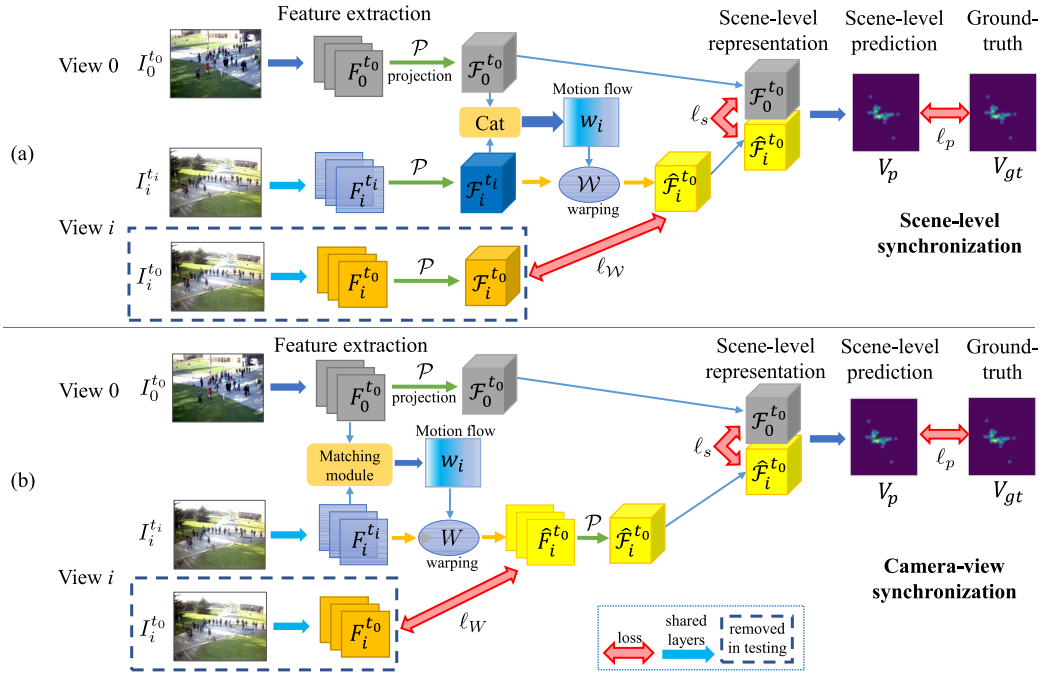

Fig. 2. General multiview pipeline consists of several stages: *camera-view feature extraction, feature projection, multiview feature fusion to obtain a scene-level representation, and prediction*. (a) **SLS** performs the synchronization *after the projection*. The unsynchronized projected features from the reference view and other views are *concatenated to predict the motion flow*, which is then used to *warp the other views' projected features to match those of the reference view*. (b) **CLS** performs the synchronization *before the projection*. The unsynchronized camera-view features from the reference view and other views are *matched together to predict the motion flow*, which is used to *warp features from other camera views to the reference view*.

image is available from each camera view, i.e., there is no temporal information available. Furthermore, these methods pose frame synchronization as optimization problems that are applicable only to the particular multiview task, and cannot be directly applied to DNN-based multiview models. In contrast, we propose a synchronization module that can be broadly applied to many DNN-based multicamera models, enabling their use with unsynchronized inputs.

C. Traditional Methods for Multiview Video Synchronization

Traditional synchronization methods usually serve as a preprocessing step for multicamera surveillance tasks. Except audio-based synchronization like [28], most traditional camera synchronization methods rely on *videos or image sequences* and hand-crafted features for camera alignment/synchronization [29]–[33]. Typical approaches recover the temporal offset by matching features extracted from the videos, e.g., space-time feature trajectories [34]–[36], image features [37], low-level temporal signals based on fundamental matrices [38], silhouette motion [39], and relative object motion [40]. The accuracy of feature matching is improved using epipolar geometry [37], [39] and rank constraints [35]. Caspi *et al.* [34] proposed to use the space-time feature trajectories matching instead of feature-points matching to reduce the search space. Dai *et al.* [29] proposed an iterative procedure to achieve the alignment in space and time with the homography assumption in spatial domain. Imre and Hilton [37] utilized image feature correspondences and epipolar geometry to find the corresponding frame indices and computes the relative frame rate and offset by fitting a 2-D line to the index correspondences. Meyer *et al.* [36] estimated the frame accurate offset by analyzing the trajectories and matching their characteristic time patterns. Pundik and Moses [38] presented a method for online synchronization that relied on the video sequences with known fundamental matrix to compute low-level temporal signals for matching. Rao *et al.* [35] proposed the rank constraint of corresponding points in two views to measure the similarity between trajectories to avoid the noise sensitivity of the fundamental matrix. Sinha and Pollefeys [39] proposed a Random sample consensus (RANSAC)-based algorithm that computed the epipolar geometry and synchronization of a pair of cameras from the motion of silhouettes in videos. Tresadern and Reid [32] estimated possible synchronization parameters via the Hough transform and refined these parameters using nonlinear optimization methods. Yan and Pollefeys [33] relied on correlating space-time interest point distribution in time between videos which represented events in video that had high variation in both space and time. Gaspar *et al.* [40] synchronized two independently moving cameras via the relative motion between objects and known camera intrinsic.

The main disadvantages for these traditional camera synchronization methods are.

1) Videos and image sequences are required, which might not be available in practical multicamera systems with limited network bandwidth and storage.
2) A fixed frame rate of the multicameras are usually assumed, which means random frame dropping cannot be handled (except [38]).
3) Feature matching is based on hand-crafted features, which lack representation ability, or known image correspondences, which requires extra manual annotations and may not always be available.

Compared with these methods, we consider a more practical and difficult setting: only single-frames and no videos (no temporal information) are available, which means that these traditional video-based methods are not suitable solutions. These traditional methods perform image content matching using hand-crafted features and traditional matching algorithms, while in contrast, our method uses DNN-based image matching. Because we also assume that only single-frames are available, our method also requires DNN-based motion estimation to estimate a frame's features after synchronization. Finally, our synchronization module is end-to-end trainable with existing multiview DNNs and thus avoids the redesign of the whole DNNs models to handle unsynchronized multicameras.

D. DNN-Based Image Matching and Flow Estimation

Image matching and optical flow estimation both involve estimating image-to-image correspondences, which is related to frame synchronization of multiviews. We mainly review the DNN-based image matching [41]–[43] or optical flow estimation methods [44]–[46], which inspire us to solve the unsynchronized multicamera based problems in a DNN-based fashion. DNN flow [47] proposed an image matching method based on a DNN feature pyramid in a coarse-to-fine optimization manner. FlowNet [48] predicted the optical flow from DNNs with feature concatenation and correlation. SpyNet [49] combined a classical spatial-pyramid formulation with deep learning and estimated large motions in a coarse-to-fine approach by warping one image to the other at each pyramid level by the current flow estimate and computing an update to the flow. Rocco *et al.* [41] addressed image correspondence problem using a convolutional neural network architecture that mimics classic image matching algorithms. PWC-Net [50] uses a feature pyramid and one image feature map is warped to the other at each scale, which is guided by the upsampled optical flow estimated from the previous scale. Lai *et al.* [51] proposed a single network to jointly learn spatiotemporal correspondence for stereo matching and flow estimation.

Our method is related to the DNN-based image matching and optical flow estimation, but the difference is still significant.

1) Typical image/geometric matching only involves *either* a camera view angle transformation (e.g., camera relative pose estimation, stereo matching) *or* a small time change in the same view (optical flow estimation), while *both* factors appear in our problem, which makes our problem harder.
2) Image/geometric matching is directly supervised by the correspondence of two images, while the multiview fusion ground-truth in the 3-D world is used as supervisory signal in our problem.
3) The 2-D-to-3-D projection causes ambiguity for multiview feature fusion, which also causes difficulties for view synchronization.

III. Single-Frame DNNs Multicamera Synchronization

In this section, we propose our single-frame synchronization model for DNN-based multiview models. The temporal offset between cameras can either be constant latency for each camera (the same offset over time), or random latency (random offsets over time). Similar to most multiview methods [2], [7], [17], [20], we assume that the cameras are static and the cameras' intrinsic and extrinsic parameters are known. The main idea of our method is to choose a camera view as the reference view, and then use the view synchronization model to warp the other camera views to be synchronized with the reference view. The synchronization model should be general enough to handle both constant and random latencies between cameras, in order to work under various conditions causing desynchronization.

DNNs models for the multicamera surveillance tasks typically consist of three stages (see in Fig. 1): **Single-view feature extraction**, which extracts single-view features of the input camera views. **Multiview feature projection and fusion**, where a fixed differentiable projection layer is first adopted to project the single-view features to the 3-D coordinate map and then the projected multiview features are fused together to form the scene-level representation. The projection layer depends the application task, and our framework can generally handle any differentiable projection layer. For example, for multiview counting [9], the projection maps the 2-D camera view to the 3-D scene plane at the average person height (assuming all camera pixels fall on the same height plane), while for 3-D pose estimation [2], the projection copies features along a view-ray in the 3-D grid, assuming an unknown height of each camera-view pixel. **Prediction**, where the decoder predicts the final result in the 3-D coordinate map, such as ground-plane density maps [9] or 3-D reconstruction [4].

In Fig. 2, we take multiview crowd counting [9] as an example to show the pipeline of the proposed single-frame-based view synchronization model. In the multiview fusion model, we denote the input multiview frames as $\{I_i^{t_0}\}_{i=0}^{n-1}$, where i denotes the camera view id and n is the input camera view number, and superscript t_0 indicates that the frames are all captured at the same time point t_0, corresponding to the synchronized multicamera setup. After being fed into the single-view feature extractor F, the extracted features are denoted as

$$F_i^{t_0} = F\left(I_i^{t_0}\right), \quad i \in \{0, 1, \ldots, n-1\}. \tag{1}$$

For multiview counting [9], the projection $\mathcal{P}$ maps the 2-D camera view to the 3-D scene plane at the average person height. After projection layer $\mathcal{P}$, the projected multiview features are

$$\mathcal{F}_i^{t_0} = \mathcal{P}\left(F_i^{t_0}\right), \quad i \in \{0, 1, \ldots, n-1\}. \tag{2}$$

We use U to denote the fusion operation (e.g., concatenation and max-pooling) of the projected multiview features, thus the fused feature is $U(\mathcal{F}_0^{t_0}, \ldots, \mathcal{F}_{n-1}^{t_0})$. Finally, the decoder D is applied to obtain the final prediction V_p

$$\begin{aligned} V_p &= D\left(U\left(\mathcal{F}_0^{t_0}, \ldots, \mathcal{F}_{n-1}^{t_0}\right)\right) \\ &= D\left(U\left(\mathcal{P}\left(F_0^{t_0}\right), \ldots, \mathcal{P}\left(F_{n-1}^{t_0}\right)\right)\right). \end{aligned} \tag{3}$$

However, when the input multicameras frames are not synchronized, denoted as $\{I_i^{t_i}\}_{i=0}^{n-1}$, the capture time for the ith

view $t_i \neq t_0$. Thus, we need to synchronize the camera views first by utilizing the view synchronization model.

The view synchronization model can be embedded into one of the first two stages, synchronizing the extracted single-view features $\{F_i^{t_i}\}$ or projected features $\{\mathcal{F}_i^{t_i}\}$, without the need to redesign a new architecture. Thus, we propose two variants of the synchronization model: 1) SLS, where the projected features $\{\mathcal{F}_i^{t_i}\}$ from different camera views are synchronized during multicamera feature fusion and 2) CLS, where the camera view features $\{F_i^{t_i}\}$ are synchronized before projection and fusion. We present the details of the two synchronization models next. Note that we first consider the case when both synchronized and unsynchronized multiview images are available for training (but not available in the testing stage). We then extend this to the case when only unsynchronized training images are available.

A. Scene-Level Synchronization

SLS works by synchronizing the multicamera features after the projection stage in the multiview pipeline. The pipeline for SLS is shown in Fig. 2(a).

1) Synchronization Module: Without loss in generality, we choose one view (denoted as view 0) as the reference view, and other views are to be synchronized to this reference view. We first assume that synchronized frame pairs are available in the training stage. The frames are $I_0^{t_0}$ from reference view 0 captured at reference time t_0, and $I_i^{t_0}$ and $I_i^{t_i}$ from view i ($i \in \{1, 2, \ldots, n-1\}$) taken at times t_0 and t_i. Note that frames $(I_0^{t_0}, I_i^{t_0})$ are synchronized, while $(I_0^{t_0}, I_i^{t_i})$ are not.

The synchronization module consists of the following stages. First, camera frame feature maps $(F_0^{t_0}, F_i^{t_0}, F_i^{t_i})$ (both synchronized and unsynchronized frames) are extracted and projected to the 3-D world space, resulting in the projected feature maps $(\mathcal{F}_0^{t_0}, \mathcal{F}_i^{t_0}, \mathcal{F}_i^{t_i})$. Second, synchronization is performed between the reference view 0 and each other view i. The projected feature map $\mathcal{F}_0^{t_0}$ from the reference view is concatenated with the projected feature map $\mathcal{F}_i^{t_i}$ from view i, and then fed into a motion flow estimation network $\mathcal{M}_s$ to predict the scene-level motion flow w_i between view i at time t_i and the reference view at time t_0

$$w_i = \mathcal{M}_s\big(\text{Cat}\big(\mathcal{F}_0^{t_0}, \mathcal{F}_i^{t_i}\big)\big), \quad i \in \{1, \ldots, n-1\} \tag{4}$$

where Cat is the concatenation operation. The $\mathcal{F}_i^{t_i}$ from view i is then synchronized with the reference view at time t_0 using a warping transformation $\mathcal{W}$ guided by w_i, $\mathcal{W}(w_i, \mathcal{F}_i^{t_i})$

$$\hat{\mathcal{F}}_i^{t_0} = \mathcal{W}\big(w_i, \mathcal{F}_i^{t_i}\big), \quad i \in \{1, \ldots, n-1\} \tag{5}$$

where $\hat{\mathcal{F}}_i^{t_0}$ are the warped projected features of the ith view synchronized to time t_0. Note that the warping $\mathcal{W}$ only applies spatial shifting to the feature map $\mathcal{F}_i^{t_i}$, i.e., it only changes the feature locations and does not change the feature values. Finally, the reference view features $\mathcal{F}_0^{t_0}$ and estimated warped features of the other views $\{\hat{\mathcal{F}}_i^{t_0}\}$ are fused and decoded to obtain the final scene-level prediction V_p

$$V_p = D\big(U\big(\mathcal{F}_0^{t_0}, \hat{\mathcal{F}}_1^{t_0}, \ldots, \hat{\mathcal{F}}_{n-1}^{t_0}\big)\big) \tag{6}$$

$$= D\big(U\big(\mathcal{F}_0^{t_0}, \mathcal{W}\big(w_1, \mathcal{F}_1^{t_1}\big), \ldots, \mathcal{W}\big(w_{n-1}, \mathcal{F}_{n-1}^{t_{n-1}}\big)\big)\big). \tag{7}$$

In the testing stage, only unsynchronized frames $(I_0^{t_0}, I_i^{t_i})$ are available and the forward operations related to frame $I_i^{t_0}$ are removed from the network.

2) Training Loss: Two losses are used in the training stage. The first loss is a task-specific prediction loss ℓ_p between the scene-level prediction V_p and the ground-truth V_{gt}. For example, for multiview crowd counting ℓ_p is the mean-square error, and V_p, V_{gt} are the predicted and ground-truth scene-level density maps. The second loss is on the multiview feature synchronization in the multiview fusion stage. Since the synced frame pairs are available during training, the feature warping loss $\ell_{\mathcal{W}}$ encourages the warped features to be similar to the features of the original synced frame of view i

$$\ell_{\mathcal{W}}\big(w_i, \mathcal{F}_i^{t_0}, \mathcal{F}_i^{t_i}\big) = \text{mse}\big(\mathcal{F}_i^{t_0}, \hat{\mathcal{F}}_i^{t_0}\big)$$

$$= \text{mse}\big(\mathcal{F}_i^{t_0}, \mathcal{W}\big(w_i, \mathcal{F}_i^{t_i}\big)\big) \tag{8}$$

where mse is the mean-square error loss. Note that the warping $\mathcal{W}$ only applies spatial shifting, and thus the minimization of the warping loss $\ell_{\mathcal{W}}$ in (8) will be based on the feature alignment via scene-level motion flow w_i and not global feature value changes (e.g., color correction). Finally, the training loss combines the task loss and the warping loss summed over all nonreference views

$$\ell = \ell_p\big(V_p, V_{gt}\big) + \gamma \sum_{i=1}^{n-1} \ell_{\mathcal{W}}\big(w_i, \mathcal{F}_i^{t_0}, \mathcal{F}_i^{t_i}\big) \tag{9}$$

where γ is a hyperparameter.

B. Camera View-Level Synchronization

Each image pixels' height in 3-D space is unknown, and thus the projection operation of multicamera DNNs models [2], [9], [17] will either project each pixel to the same assumed height level [9] (causing distortion when the true pixel height is different), or to multiple height levels [2], [17] (duplicating features along the view ray). These projection cause the features to stretch along the view ray in the 3-D scene, which makes their synchronization more difficult due to their imprecise (stretched) and ambiguous (duplicated) nature. Therefore, we also consider synchronization between camera view features before the projection. The pipeline for CLS is presented in Fig. 2(b).

1) Synchronization Model: The view synchronization model is applied to each view separately. The camera view features $(F_0^{t_0}, F_i^{t_i})$ from the unsynchronized reference view and view i are first passed through a matching module (see below) and then fed into the motion flow estimation network $\mathcal{M}_c$ to predict the camera-view motion flow w_i for view i. The warping transformation W guided by w_i then warps the camera-view features $F_i^{t_i}$ from view i to be synchronized with the reference view at time t_0

$$\hat{F}_i^{t_0} = W\big(w_i, F_i^{t_i}\big), \quad i \in \{1, \ldots, n-1\} \tag{10}$$

where $\hat{F}_i^{t_0}$ is the warped camera-view features of view i captured at time t_i, which is synchronized to reference view 0 captured at time t_0. Finally, the reference and warped camera views are projected

$$\mathcal{F}_0^{t_0} = \mathcal{P}\big(F_0^{t_0}\big), \hat{\mathcal{F}}_i^{t_0} = \mathcal{P}\big(\hat{F}_i^{t_0}\big), \quad i \in \{1, \ldots, n-1\} \tag{11}$$

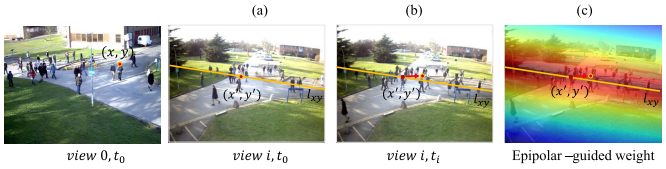

Fig. 3. Epipolar-guided weights. (a) In the synchronized setting, given the point (x, y) in view 0, the matched point (x', y') in view i must be on the epipolar line l_{xy}. (b) In the unsynchronized setting, we assume a Gaussian motion model of the matched feature location from time t_0 to t_i. (c) Epipolar-guided weight mask is use to bias the feature matching toward high-probability regions according to the motion model.

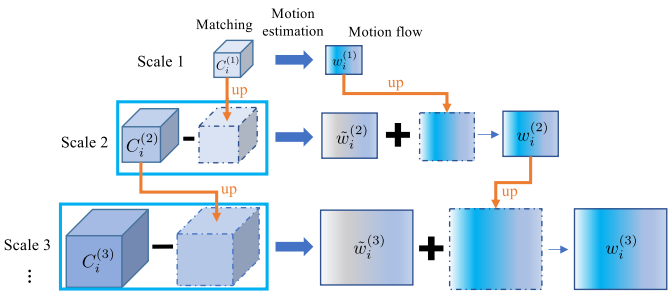

Fig. 4. Multiscale estimation of motion flow.

and then fused and decoded to obtain the scene-level prediction V_p

$$V_p = D\big(U\big(\mathcal{F}_0^{t_0}, \hat{\mathcal{F}}_1^{t_0}, \ldots, \hat{\mathcal{F}}_{n-1}^{t_0}\big)\big) \tag{12}$$

$$= D\big(U\big(\mathcal{P}\big(F_0^{t_0}\big), \mathcal{P}\big(\hat{F}_1^{t_0}\big), \ldots, \mathcal{P}\big(\hat{F}_{n-1}^{t_0}\big)\big)\big). \tag{13}$$

In the testing stage, only unsynchronized frames $(I_0^{t_0}, I_i^{t_i})$ are available and the forward operations related to frame $I_i^{t_0}$ are removed from the network.

2) Matching Module: We propose three methods to match features to predict the view-level motion flow. The first method concatenates the features $(F_0^{t_0}, F_i^{t_i})$ and then feeds them into the motion flow estimation network $\mathcal{M}_c$ to predict the motion flow w_i

$$w_i = \mathcal{M}_c\big(\mathrm{Cat}\big(F_0^{t_0}, F_i^{t_i}\big)\big), \quad i \in \{1, \ldots, n-1\}. \tag{14}$$

The second method builds a correlation map C_i between features from each pair of spatial locations in $F_0^{t_0}$ and $F_i^{t_i}$

$$C_i\big((x, y), \big(x', y'\big)\big) = F_0^{t_0}(x, y)^T F_i^{t_i}\big(x', y'\big) \tag{15}$$

which is then fed into the motion flow estimation network $\mathcal{M}_c$ to predict the motion flow w_i

$$w_i = \mathcal{M}_c(C_i), \quad i \in \{1, \ldots, n-1\}. \tag{16}$$

The third method incorporates camera geometry information into the correlation map to suppress false matches. If both cameras are synchronized at t_0, then according the multiview geometry, each spatial location in view 0 must match a location in view i on its corresponding epipolar line [Fig. 3(a)]. Thus, in the synchronized setting, detected matches that are not on the epipolar line can be rejected as false matches. For our unsynchronized setting, the matched location in view i remains on the epipolar line only when its corresponding feature/object does not move between times t_0 and t_i. To handle the case where the feature moves, we assume that a matched feature in view i moves according to a Gaussian motion model with standard deviation σ [Fig. 3(b)]. With the epipolar line and motion model, we then build a weighting mask, with high weights on locations with high probability of containing the matched features, and vice versa. Specifically, we set the mask $M_i((x, y), (x', y')) = 1$ if (x', y') is on the epipolar line induced by (x, y), and 0 otherwise, and then convolve it with a 2-D Gaussian with standard deviation σ [Fig. 3(c)]. We then apply the weight mask M_i on the correlation map $\tilde{C}_i = M_i \odot C_i$, which will suppress false matches that are not consistent with the scene and motion model. Thus, the motion flow w_i is

$$w_i = \mathcal{M}_c\big(\tilde{C}_i\big) = \mathcal{M}_c(M_i \odot C_i), \quad i \in \{1, \ldots, n-1\}. \tag{17}$$

3) Multiscale Architecture: Multiscale feature extractors are used in multicamera tasks like crowd counting [9] or to refine the final prediction via multiscale prediction fusion [50], [51]. Therefore, we next show how to incorporate multiscale feature extractors with our CLS model.[1] Instead of performing the view synchronization in each scale separately, the motion flow estimate of neighbor scales is fused to refine the current scale's estimate (see Fig. 4). In particular, let there be m scales in the multiscale architecture and j denotes one scale in the scale range $\{1, 2, \ldots, m\}$, with m the largest scale. The multiscale predicted motion flow are fused as follows.

1) When $j = 1$ (the smallest scale), the correlation map $C_i^{(1)}$ of scale 1 is fed into the motion flow estimation net to predict the motion flow $w_i^{(1)}$ for scale 1.
2) For scales $j > 1$, first the difference between the correlation map $C_i^{(j)}$ and the upsampled correlation map of the previous scale $\mathrm{up}(C_i^{(j-1)})$ is fed into the motion flow estimation net to predict the residual of the motion flow between two scales, denoted as $\tilde{w}_i^{(j)}$.
3) The refined motion flow of scale j is

$$w_i^{(j)} = \mathrm{up}\Big(w_i^{(j-1)}\Big) + \tilde{w}_i^{(j)}. \tag{18}$$

4) Training Loss: Similar to SLS, a combination of two losses (scene-level prediction and feature synchronization) is used in the training stage. The scene-level prediction loss is the same as before. The feature synchronization loss encourages the warped camera-view features at each scale to match the features of the original synchronized frame

$$\ell_W = \mathrm{mse}\Big(F_i^{t_0,(j)}, \hat{F}_i^{t_0,(j)}\Big) \tag{19}$$

$$= \mathrm{mse}\Big(F_i^{t_0,(j)}, W\Big(w_i^{(j)}, F_i^{t_i,(j)}\Big)\Big). \tag{20}$$

Similar to SLS, the warping function W only applies spatial shifting, and thus the minimization of ℓ_W in (20) will be based on feature alignment rather than feature value changes. Finally, the training loss is the combination of the prediction loss and the synchronization loss summed over all nonreference views

[1]No extra steps are needed to incorporate multiscale features with SLS because the synchronization occurs after the feature projection.

and scales

$$\ell = \ell_p\left(V_p, V_{gt}\right) + \gamma \sum_{i=1}^{n-1}\sum_{j=1}^{m} \ell_W\left(w_i^{(j)}, \mathcal{F}_i^{t_0,(j)}, \mathcal{F}_i^{t_i,(j)}\right) \quad (21)$$

where γ is a hyperparameter.

C. Training With Only Unsynchronized Frames

In the previous models, we assume that both synchronized and unsynchronized multicamera frames are available during training. For more practical applications, we also consider the case when only unsynchronized multiview frames are available for training. In this case, for the SLS, the warping feature loss $\ell_{\mathcal{W}}$ is replaced with a similarity loss ℓ_s on the projected features, to indirectly encourage synchronization of the projected multiview features

$$\ell_s = \text{mean}\left(1 - \cos\left(\mathcal{F}_0^{t_0}, \mathcal{W}\left(w_i, \mathcal{F}_i^{t_i}\right)\right)\right) \quad (22)$$

where "cos" is the cosine similarity between feature maps (along the channel dimension), and "mean" is the mean over all spatial locations. Similarly, for CLS, the warping feature loss ℓ_W is replaced by the similarity loss of the projected features ℓ_s. Note that the similarity loss ℓ_s is applied after the projection—thus the warping function only needs to predict the residual motion in the camera view, which is the object motion in time, so as to align the projected features.

IV. Experiments

We validate the effectiveness of the proposed view synchronization model on two unsynchronized multiview tasks: multiview crowd counting and multiview 3-D human pose estimation.

A. Implementation Details

The synchronization model consists of two parts: motion estimation network and feature warping layer. The input of the motion estimation network is the unsynchronized multiview features (the concatenation of the projected features) for SLS or the matching result of the 2-D camera-view features for CLS, and the output is a two-channel motion flow. The layer setting of the motion estimation network is shown in Table I. The feature warping layer warps the features from other views to align with the reference views, guided by the estimated motion flow. The feature warping layer is based on the image resampler from the spatial transformation layer in [52].

The synchronized multiview model consists of feature extraction module, projection module, and multiview prediction module. For the multiview counting model [9], Table II shows the model setting of the feature extraction and multiview prediction module. For the 3-D pose estimation model [2], the feature extraction module consists of a ResNet-152 network, a series of transposed convolution layers and a 1×1 convolution layer to predict joint heatmaps [53], and the V2V-PoseNet [54] is used for multiview prediction, which is based on hour-glass network [55].

TABLE I

Layer Settings for the Motion Estimation Net in the View Synchronization Module. The Filter Dimensions Are Output Channels, Input Channels and Filter Size $w_0 \times h_0$

Layer	Filter
conv 1	$128\times n\times 5\times 5$
conv 2	$128\times 128\times 5\times 5$
conv 3	$64\times 128\times 5\times 5$
conv 4	$64\times 64\times 5\times 5$
conv 5	$32\times 64\times 5\times 5$
conv 6	$2\times 32\times 5\times 5$

TABLE II

Model Setting of the Synchronized Multiview Counting Model [9], Consisting of Feature Extraction and Multiview Prediction. The Filter Dimensions are Output Channels, Input Channels, and Filter Size ($w \times h$)

Feature extraction	
Layer	Filter
conv 1	$16\times 1\times 5\times 5$
conv 2	$16\times 16\times 5\times 5$
pooling	2×2
conv 3	$32\times 16\times 5\times 5$
conv 4	$32\times 32\times 5\times 5$
pooling	2×2
conv 5	$64\times 32\times 5\times 5$
conv 6	$32\times 64\times 5\times 5$
conv 7	$1\times 32\times 5\times 5$

Prediction	
Layer	Filter
concat	-
conv 1	$64\times n\times 5\times 5$
conv 2	$32\times 64\times 5\times 5$
conv 3	$1\times 32\times 5\times 5$

B. Experiment Setup

We test four versions of our synchronization model: scene-level synchronization (denoted as SLS), and CLS using concatenation, correlation, or correlation with epipolar-guided weights (denoted as CLS-cat, CLS-cor, CLS-epi) for the matching module (Section III-B.2). The synchronization models are trained with the multiview DNNs introduced in each application later.

We consider two training scenarios: 1) both synchronized and unsynchronized training data is available and 2) only unsynchronized training data is available, which is the more difficult setting. For the first training scenario, we compare against two comparison methods: *BaseS* trains the DNN only on the synchronized data; *BaseSU* fine-tunes the BaseS model using the unsynchronized training data (using the full training set). For the second training scenario, *BaseU* trains the DNN directly from the unsynchronized data. Note that traditional synchronization methods [29]–[33] are based on videos (temporal information) and assume high-fps cameras with fixed frame rates, which are unavailable in our problem setting. Thus, traditional and video-based synchronization methods are not suitable for comparison.

To test the proposed method, we first create an unsynchonized multiview dataset from the existing multiview datasets (the specific datasets are introduced in each application later). In particular, suppose the frame sequence in the reference view is captured at times $t_0 + k\Delta t$, where Δt is the time offset between neighbor frames, $k \in \{0, \ldots, N-1\}$ and N is the number of frames. For view i, the unsynchronized frames are captured at times $t_0 + k\Delta t + \delta_{i,k}$, where $\delta_{i,k}$ is the desynchronization time offset between view i and the reference view. We consider two settings of the desynchronization offset. The first is a *constant latency* for each view, $\delta_{i,k} = \tau_i$, for some constant value τ_i. The second is *random*

TABLE III

UNSYNCHRONIZED MULTIVIEW COUNTING: EXPERIMENT RESULTS FOR TRAINING SET WITH BOTH SYNCHRONIZED AND UNSYNCHRONIZED FRAMES. TWO DESYNCHRONIZATION SETTINGS ARE TESTED: CONSTANT LATENCY AND RANDOM LATENCY. THE EVALUATION METRIC IS MAE AND NAE

		PETS2009				*CityStreet*			
		constant		random		constant		random	
loss	model	MAE	NAE	MAE	NAE	MAE	NAE	MAE	NAE
ℓ_p	BaseS	7.21	0.200	4.58	0.139	9.07	0.108	8.86	0.107
	BaseSU	4.36	0.137	4.30	0.140	9.02	0.106	8.82	0.108
ℓ_p, ℓ_W	SLS	4.49	0.145	4.91	0.154	8.23	0.102	8.02	0.101
	CLS-cat	4.18	0.130	4.85	0.150	8.82	0.111	8.57	0.108
	CLS-cor	4.13	0.135	**4.03**	**0.128**	**8.03**	**0.099**	7.99	0.098
	CLS-epi	**3.95**	**0.130**	4.09	0.129	8.05	0.100	**7.93**	**0.096**

TABLE IV

UNSYNCHRONIZED MULTIVIEW COUNTING: EXPERIMENT RESULTS FOR TRAINING SET WITH ONLY UNSYNCHRONIZED FRAMES UNDER CONSTANT AND RANDOM LATENCY

		PETS2009				*CityStreet*			
		constant		random		constant		random	
loss	model	MAE	NAE	MAE	NAE	MAE	NAE	MAE	NAE
ℓ_p	BaseU	6.18	0.187	6.22	0.192	10.22	0.134	9.35	0.121
ℓ_p, ℓ_s	SLS	5.37	0.178	4.82	0.150	8.50	0.105	8.33	0.100
	CLS-cat	6.00	0.186	6.08	0.189	8.48	0.102	9.17	0.110
	CLS-cor	**4.18**	0.136	**4.34**	**0.136**	**8.02**	0.098	7.77	**0.093**
	CLS-epi	4.25	**0.135**	4.77	0.144	8.04	**0.095**	**7.70**	0.094
ℓ_p	SLS	7.13	0.226	5.30	0.162	8.77	0.107	8.45	0.107
	CLS-cat	6.30	0.194	5.98	0.184	8.28	0.098	9.15	0.108
	CLS-cor	**4.25**	0.138	**4.49**	**0.141**	8.20	0.099	8.10	0.102
	CLS-epi	4.27	**0.135**	4.53	0.143	**8.16**	**0.097**	**7.86**	**0.096**

latency, where the offset for each frame and view is randomly sampled from a uniform distribution, $\delta_{i,k} \sim U(-\kappa_i, \kappa_i)$. Finally, since the synchronization is with the reference view, the ground-truth labels for the multiview task correspond to the times of the reference view, $t_0 + k\Delta t$.

C. Unsynchronized Multiview Counting

We first apply our synchronization model to unsynchronized multiview counting system, whose bandwidth is assumed to be limited and the frame latency between cameras can be fixed or random. Here, we adopt the multiview multiscale fusion model (MVMS) from [9], which is the state-of-the-art model for multiview counting DNNs. We embed the synchronization models in the MVMS model to handle the unsynchronized multiview frames for crowd counting.

1) Datasets and Metric: Two multiview counting datasets used in [9], PETS2009 [56] and CityStreet [9], are selected and desynchronized for the experiments.

PETS2009 contains three views (cameras 1, 2, and 3), and the first camera view is chosen as the reference view. The input image resolution ($w \times h$) is 384 × 288 and the ground-truth scene-level density map resolution is 152 × 177. There are 825 multiview frames for training and 514 frames for testing. The frame rate of PETS2009 is 7 fps ($\Delta t = 1/7s$). For constant frame latency, $\tau_i \in \{5 \text{ s}, -5 \text{ s}\}$ is used for cameras 2 and 3, and $\kappa_i = 5$ s for random latency.

CityStreet proposed in [9] consists of three views (cameras 1, 3, and 4), and camera 1 is chosen as the reference view. The input image resolution is 676 × 380 and the ground-truth density map resolution is 160 × 192. There are 500 multiview frames, and the first 300 are used for training and the remaining 200 for testing. The frame rate of CityStreet is 1 fps ($\Delta t = 1$ s).[2] For constant latency, $\tau_i \in \{3 \text{ s}, -3 \text{ s}\}$ for cameras 3 and 4, and $\kappa_i = 3$ s for random latency.

TABLE V

UNSYNCHRONIZED MULTIVIEW COUNTING: EXPERIMENT RESULTS FOR TRAINING SET WITH ONLY UNSYNCHRONIZED FRAMES UNDER CONSTANT AND RANDOM LATENCY AND USING GROUND-TRUTH CALCULATED FROM UNSYNCHRONIZED MULTIVIEW FRAMES

		PETS2009				*CityStreet*			
		constant		random		constant		random	
loss	model	MAE	NAE	MAE	NAE	MAE	NAE	MAE	NAE
ℓ_p	BaseU	14.89	0.458	10.95	0.484	10.96	0.146	11.30	0.149
ℓ_p, ℓ_s	SLS	6.80	0.229	6.58	0.283	9.18	0.111	9.49	0.117
	CLS-cat	7.41	0.237	6.10	0.237	9.72	0.130	9.69	0.129
	CLS-cor	5.91	0.201	5.93	0.240	8.55	0.106	8.31	0.107
	CLS-epi	**5.72**	**0.184**	**4.80**	**0.187**	**8.32**	**0.104**	**8.05**	**0.102**
ℓ_p	SLS	7.85	0.274	7.22	0.313	9.31	0.109	8.91	0.108
	CLS-cat	7.52	0.240	6.20	0.243	8.48	0.107	9.85	0.121
	CLS-cor	6.98	0.244	6.26	0.282	**8.03**	**0.099**	8.24	0.107
	CLS-epi	**6.80**	**0.229**	**5.18**	**0.200**	8.23	0.102	**8.16**	**0.103**

Following [9], the mean absolute error (MAE) and normalized absolute error (NAE) of the predicted counts on the test set are used as the evaluation metric

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{c}_i - c_i| \tag{23}$$

$$\text{NAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{c}_i - c_i|/c_i \tag{24}$$

where c_i is the ground truth count and $\hat{c}_i$ is the predicted count, and N is the number of testing images.

2) Results for Training With Synced and Unsynced Frames: The experimental results using training with synchronized and unsynchronized frames are shown in Table III. The hyperparameter $\gamma = 1$ is used for feature warping loss. On both datasets, our CLS methods, CLS-cor and CLS-epi, perform better than other methods, including the baselines, demonstrating the efficicacy of our approach. SLS performs worse than CLS methods, due to the ambiguity of the projected features from multiviews. Furthermore after projection to the ground-plane, the crowd movement between frames $I_i^{t_0}$ and $I_i^{t_i}$ on the ground-plane is less salient due to the low resolution of the ground-plane feature map. CLS-cat performs worse among the CLS methods because simple concatenation of features cannot capture the image correspondence between different views to estimate the motion flow. Finally, the two baselines (BaseS and BaseSU) perform badly on CityStreet because of the larger scene with larger crowd movement between neighboring frames (due to lower frame rate).

3) Results for Training With Only Unsynchronized Frames: The experiment results by training with only unsynchronized frames (which is a more practical real-world case) are shown in Table IV. Since the synchronized frames are not available, the MVMS model weights are trained from scratch using only unsynchronized data. Our models are trained with the similarity loss ℓ_s (with hyperparameter $\gamma = 1000$), which encourages alignment of the projected multiview features. Generally, without the synchronized frames in the training stage, the counting

[2]We obtained the higher fps version from the dataset authors.

TABLE VI

ABLATION STUDY ON THE MULTISCALE ARCHITECTURE OF THE PROPOSED METHODS FOR MULTIVIEW COUNTING ON CITYSTREET DATASET. THE TOP ROWS SHOW PERFORMANCE WHEN TRAINING WITH SYNCHRONIZED AND UNSYNCHRONIZED FRAMES AND USING FEATURE WARPING LOSS ℓ_W. THE BOTTOM IS TRAINING ONLY ON UNSYNCHRONIZED FRAMES USING FEATURE SIMILARITY LOSS ℓ_s

Loss/Training data	Method	Multi-scale		Single-scale	
		MAE	NAE	MAE	NAE
ℓ_p, ℓ_W / sync and unsync	SLS	8.02	0.101	8.31	0.100
	CLS-cat	8.57	0.108	8.77	0.102
	CLS-cor	7.99	0.098	8.25	0.099
	CLS-epi	7.93	0.096	8.12	0.098
ℓ_p, ℓ_s unsync	SLS	8.33	0.100	8.95	0.112
	CLS-cat	9.17	0.110	9.54	0.116
	CLS-cor	7.77	0.093	8.62	0.111
	CLS-epi	7.70	0.094	8.59	0.110

TABLE VII

AVERAGE FEATURE MAPS VALUE MEAN AND VARIANCE BEFORE AND AFTER THE FEATURE WARPING OF VIEWS 2 AND 3 OF CITYSTREET DATASET

Method	view 2	view 3
before warping	0.686 ± 1.006	0.777 ± 0.704
after warping	0.670 ± 0.976	0.761 ± 0.692

TABLE VIII

COMPARISON OF METHODS ON THE CITYSTREET DATASET WITH ONLY UNSYNCHRONIZED FRAMES (BOTH CONSTANT AND RANDOM UNSYNCHRONIZED FRAMES)

Method	constant	random
Color correction	8.90/0.108	8.64/0.100
SLS	8.50/0.105	8.33/0.100
CLS-cat	8.48/0.102	9.17/0.110
CLS-cor	**8.02**/0.098	7.77/**0.093**
CLS-epi	8.04/**0.095**	**7.70**/0.094

TABLE IX

MODEL PARAMETER NUMBER AND RUNNING SPEED COMPARISON OF THE BASELINE METHODS BASES/BASESU/BASEU AND THE PROPOSED SLS, CLS-CAT, CLS-COR, AND CLS-EPI FOR MULTIVIEW COUNTING ON CITYSTREET DATASET. THE INPUT RESOLUTION FOR THE CORRELATION STEP OF THE CAMERA-VIEW SYNCHRONIZATION MODULE IS 160×95

Method	Paras. Num	FPS
BaseS/BaseSU/BaseU	853.4K	21.9
SLS	3.7M	8.3
CLS-cat	3.7M	8.9
CLS-cor	37.3M	7.2
CLS-epi	37.3M	3.6

error increases for each method. Nonetheless, the proposed CLS models CLS-cor and CLS-epi perform much better than the baseline BaseU. CLS-cor and CLS-epi trained on only unsynchronized data also performs better (on CityStreet) or on par with (on PETS2009) the baseline BaseSU, which uses both synchronized and unsynchronized training data. These two results demonstrate the efficacy of our synchronization model when only unsynchronized training data are available. Finally, the error for almost all synchronization models increases on both datasets when training without the similarity loss (ℓ_p in Table IV). This demonstrates the effectiveness of using ℓ_s to align the multiview features in training.

4) Results for Using Ground-Truth From Unsynchronized Multiview Images: In the previous experiments (training with only unsynchronized frames, see Section IV-C.3), the ground-truth is corresponded (synchronized) to the frames of the reference view. We also perform experiments when the ground-truth scene-level density maps are calculated from the unsynchronized multiview images. Specifically, we project the same person's image coordinates of each unsynchronized view to the world plane and the average of the projection results is used as the ground-truth person location on the ground. Then, we use the obtained person location map to generate the scene-level density map.

The results for training with ground-truth from unsynchronized multiview images and only unsynchronized frames can be seen in Table V. From the table, we can also find that the proposed method CLS-cor/CLS-epi can achieve better performance than other methods and CLS-epi achieves the best performance, and the performance can be further improved by adding similarity loss ℓ_s.

5) Ablation Study on the Multiscale Architecture: We next present an ablation study on the multiscale architecture for the multiview counting in Table VI. Generally, the multiscale architecture performs better than single-scale architecture models, and the proposed CLS-cor/CLS-epi can perform better than SLS or CLS-cat under both single-scale or multiscale architecture, and under both training paradigms (sync and unsynced, or only unsynced).

6) Ablation Study on Color Correlation: The feature warping module only applies spatial shifting on the features of the unsynced views, i.e., it does not change the values (e.g., color) of the unsynced features [see (5) and (10)]. To demonstrate this, we calculate the average statistics (mean and variance) of the feature maps before and after feature warping of Views 2 and 3 of CityStreet, and present the results in Table VII. The statistics of the feature maps do not change much after performing feature warping, and thus the performance improvement of the feature warping module is not due to color correction (feature value changes).

We further perform an ablation study to show that image color correction by itself cannot solve the frame desynchronization problem. On the CityStreet dataset, in the baseline model (MVMS [9]), we add a learnable "color correction" layer, comprising an extra 1×1 convolution layer (32 channels) in the branches of the other camera views before the projection and fusion step. The results are denoted as "color correction" in Table VIII. The error for using "color correction" is worse than the proposed SLS, CLS-cor, and CLS-epi. The reason is that the desynchronization issue comes from the capture time difference between camera views, which is better solved by spatial shifting of features rather than color correction (changing feature values).

7) Model Size and Running Speed Comparison: We present the model size (number of parameters) and running speed of the baseline methods and the proposed SLS, CLS-cat, CLS-cor, and CLS-epi in Table IX. The input resolution for the correlation step of the camera-view synchronization module is 160×95. All models are tested on the CityStreet dataset with a NVIDIA 1080Ti GPU. The baseline methods (BaseS, BaseSU, and BaseU) do not use view synchronization modules, so their model sizes are smaller and running speeds are faster. The proposed CLS-cor and CLS-epi methods have

TABLE X

UNSYNCHRONIZED 3-D HUMAN POSE ESTIMATION: EXPERIMENT RESULTS WITH RANDOM LATENCY. FOR "CLS-COR" AND "CLS-EPI," THE CONSISTENCY LOSS HYPERPARAMETER $\gamma = 0.01$. THE EVALUATION METRIC IS MPJPE AND ABSOLUTE POSITION MPJPE (LEFT/RIGHT)

Latency	$8/50s$	$32/50s$	$64/50s$
BaseS	62.8/59.2	78.6/78.2	151.1/151.5
BaseSU	26.5/27.8	49.9/50.1	69.4/69.2
BaseU	37.3/38.9	50.9/50.6	71.0/70.7
CLS-cor(γ=0)	25.8/26.9	**36.5/36.7**	56.6/56.9
CLS-cor	25.8/27.0	38.2/38.7	46.8/47.1
CLS-epi	**25.7/26.8**	37.6/37.8	**45.7/45.6**

TABLE XI

DETAILED PERFORMANCE FOR UNSYNCHRONIZED 3-D HUMAN POSE ESTIMATION WITH RANDOM LATENCY $\kappa_i = 8/50$ s. THE EVALUATION METRIC IS MPJPE

Pose	BaseS	BaseSU	BaseU	CLS-cor(γ=0)	CLS-cor	CLS-epi
Directions	42.8	29.3	34.3	26.1	25.8	26.1
Discussion	60.7	28.4	38.8	27.3	26.7	27.0
Eating	60.7	26.4	28.8	23.9	24.0	23.4
Greeting	63.8	19.7	32.3	25.3	24.3	25.1
PhoneCall	52.2	25.7	31.0	24.7	24.5	24.4
Posing	49.7	22.0	27.6	24.1	24.0	24.0
Purchases	67.5	24.4	52.5	28.7	27.4	28.8
Sitting	33.2	22.6	36.6	23.8	24.0	24.0
SittingDown	37.4	25.7	66.6	25.9	26.8	27.2
Smoking	42.2	25.7	31.2	24.8	24.3	24.4
TakingPhoto	59.9	24.3	44.2	28.2	27.9	27.2
Waiting	44.3	19.5	35.8	23.2	23.8	24.2
Walking	161.1	31.9	32.1	27.0	30.2	27.8
WalkingDogs	91.5	34.2	54.8	30.1	30.1	29.8
WalkingTogether	126.8	33.9	31.8	25.5	26.8	25.5
Average	62.8	26.5	37.3	25.8	25.8	25.7

the correlation module, and thus have more parameters than SLS or CLS-cat. CLS-epi is slower than CLS-cor due to the extra multiplication step with the epipolar weights.

8) Visualization Results: Example results are shown in Fig. 5. Generally, the proposed synchronization methods CLS-epi and CLS-cor can predict better quality density maps, such as in the red box regions in the figure, where comparison methods tend to over-count these regions due to the same person being counted multiple times in unsynchronized frames. Furthermore, we also observe that the predicted density map is with better quality when synchronized frames are available compared to training with only unsynchronized frames. Finally, the prediction results are improved if similarity loss is enforced when training with only unsynchronized frames, such as the methods CLS-epi and CLS-cor on PETS2009.

D. Unsynchronized 3-D Human Pose Estimation

We next apply our synchronization model to the unsynchronized 3-D human pose estimation task. The DNNs model for 3-D human pose estimation is adopted from [2], which proposed two learnable triangulation methods for multiview 3-D human pose from multiple 2-D views: algebraic triangulation and volumetric aggregation. Here, we use volumetric aggregation (softmax aggregation) as the multiview fusion DNN in the experiments.

1) Datasets and Metrics: We use the Human3.6M [57] dataset, which consists of 3.6 million frames from four synchronized 50 Hz digital cameras along with the 3-D pose annotations. We follow the preprocessing step[3] recommended in [57], and sample one of every 64 frames ($\Delta t = 64/50$) for the testing set, and sample one of every four frames ($\Delta t = 4/50$) as the training set. The first camera view is always used as the reference view (if the first camera view is missing, the second one is used). We test desynchronization via random frame latency, with $\kappa_i \in \{8/50, 32/50, 64/50\}$ s, and only use unsynchronized data for training. Following [2], mean per point position error (MPJPE) and absolute position MPJPE are used as the metric for evaluation. In training, the single-view backbone uses the pretrained weights from the original 3-D pose estimation model. Baseline methods

TABLE XII

DETAILED PERFORMANCE FOR UNSYNCHRONIZED 3-D HUMAN POSE ESTIMATION WITH RANDOM LATENCY $\kappa_i = 32/50$ s. THE EVALUATION METRIC IS MPJPE

Pose	BaseS	BaseSU	BaseU	CLS-cor(γ=0)	CLS-cor	CLS-epi
Directions	46.2	48.7	65.1	42.9	42.5	43.9
Discussion	75.6	53.6	55.2	38.9	41.0	41.6
Eating	64.5	39.1	40.2	32.5	32.7	30.8
Greeting	71.5	48.5	55.4	35.7	38.1	36.8
PhoneCall	64.5	43.6	43.0	33.9	35.2	35.1
Posing	49.3	42.1	43.3	32.7	33.3	30.8
Purchases	111.5	50.9	48.7	35.9	42.4	40.4
Sitting	55.2	46.0	43.7	33.6	33.8	34.7
SittingDown	108.3	79.3	64.9	36.8	41.8	42.8
Smoking	54.5	44.3	44.0	35.5	35.9	35.7
TakingPhoto	87.9	57.0	58.6	39.3	43.0	41.3
Waiting	64.3	45.6	47.3	35.5	33.7	35.0
Walking	150.6	47.6	48.1	34.2	37.1	34.2
WalkingDogs	123.1	66.2	67.5	44.5	49.2	49.1
WalkingTogether	125.5	50.3	52.7	36.9	38.5	34.9
Average	78.6	49.9	50.9	36.5	38.2	37.6

TABLE XIII

DETAILED PERFORMANCE FOR UNSYNCHRONIZED 3-D HUMAN POSE ESTIMATION WITH RANDOM LATENCY $\kappa_i = 64/50$ s. THE EVALUATION METRIC IS MPJPE

Pose	BaseS	BaseSU	BaseU	CLS-cor(γ=0)	CLS-cor	CLS-epi
Directions	99.2	83.2	76.3	70.3	64.8	66.5
Discussion	144.1	72.0	67.5	57.3	48.2	48.4
Eating	138.2	55.3	63.2	44.7	40.4	37.9
Greeting	181.3	68.0	74.1	54.8	46.9	46.3
PhoneCall	138.1	58.8	61.1	49.2	40.7	40.5
Posing	121.7	53.5	50.2	42.1	36.6	36.3
Purchases	155.7	69.0	62.4	58.1	47.0	50.6
Sitting	74.0	64.2	67.8	55.2	41.1	39.6
SittingDown	103.8	112.3	140.7	89.8	54.6	50.6
Smoking	112.7	58.7	60.3	49.2	41.7	41.7
TakingPhoto	166.8	76.8	79.6	64.5	57.7	53.5
Waiting	120.2	62.7	61.1	51.1	42.0	42.6
Walking	301.1	66.3	69.2	49.7	44.0	41.7
WalkingDogs	219.2	95.9	91.6	77.0	62.7	55.5
WalkingTogether	302.7	67.2	68.9	54.5	43.5	42.5
Average	151.1	69.4	71.0	56.6	46.8	45.7

TABLE XIV

UNSYNCHRONIZED 3-D HUMAN POSE ESTIMATION: CLS-EPI EXPERIMENT RESULTS WITH DIFFERENT HYPERPARAMETER γ. THE EVALUATION METRIC IS MPJPE

γ	0.005	0.01	0.02
$\kappa_i = 8/50s$	25.6	25.7	26.0
$\kappa_i = 32/50s$	38.3	37.6	37.9
$\kappa_i = 64/50s$	51.7	45.7	46.8

[3]https://github.com/anibali/h36m-fetch. Accessed: October 10, 2019.

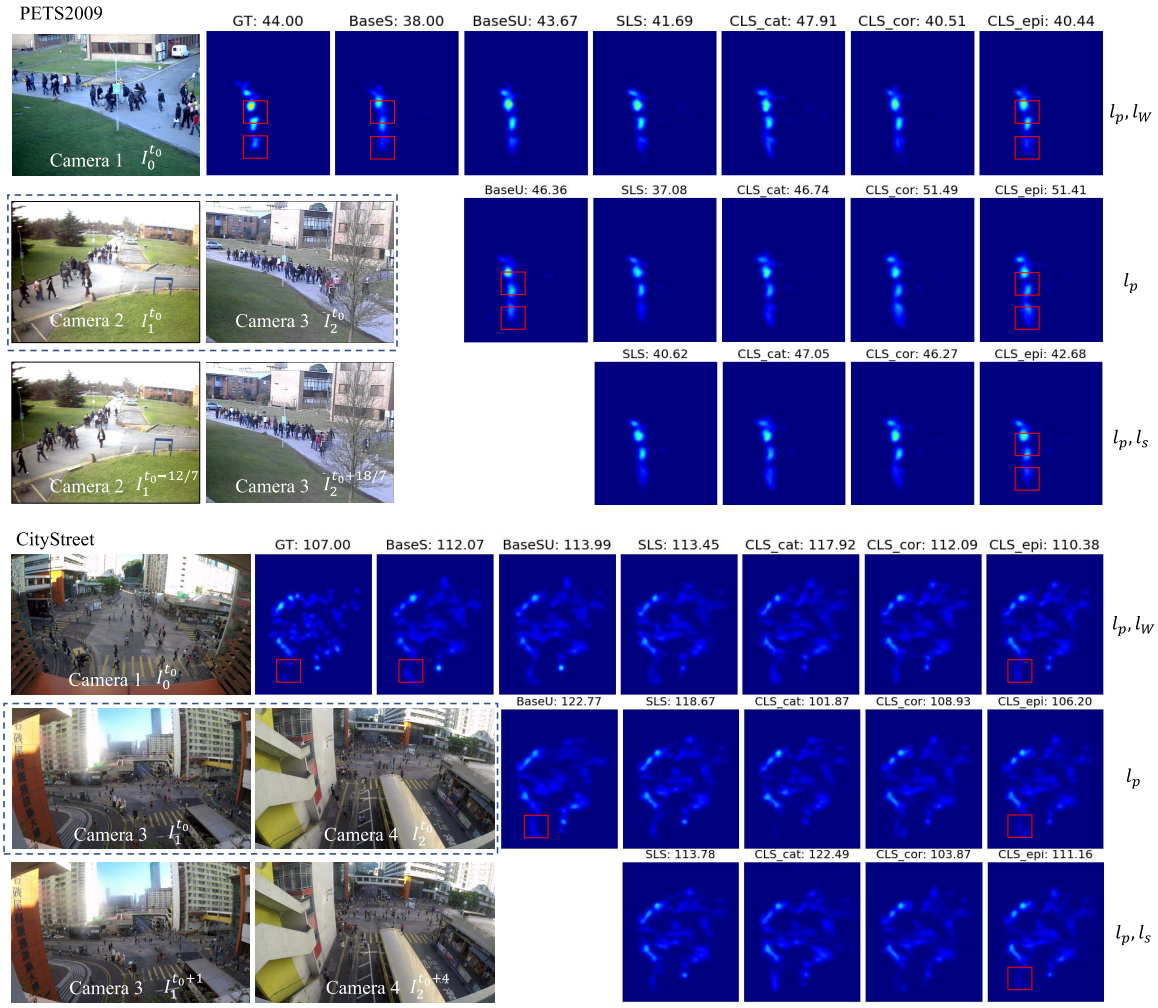

Fig. 5. Examples of unsynchronized multiview crowd counting on PETS2009 (top) and CityStreet (bottom). The left shows the input multiview frames, and note that the synchronized frames (in dotted box) are not used when training with only unsynchronized frames. The input unsynchronized frames are randomly selected around the synchronized frames. For each dataset, the result of training with synchronized and unsynchronized frames (l_p and l_W) is in row 1, the result of training only with unsynchronized frames (l_p) is shown in row 2, and the result of training with unsynchronized frames and using similarity loss between projected features (l_p and l_s) is shown in row 3. Generally, 1) the proposed synchronization methods CLS-epi and CLS-cor predict density maps with better quality compared to other comparison methods; 2) the methods achieve better performance when synchronized frames are available in training; and 3) when training only with unsynchronized frames, enforcing the similarity loss l_s can help improve the performance.

BaseS, BaseSU and BaseU are compared with our proposed camera-view synchronization models CLS-cor and CLS-epi.

2) Experiment Results: The experiments results are presented in Table X. The original 3-D pose estimation method (BaseS, BaseSU, and BaseU) cannot perform well under the unsynchronized test condition, especially under large latencies (e.g., 64/50 s). Our camera-view synchronization methods performs better than the baseline methods, with the performance gap increasing as the latency increases. Using similarity loss ℓ_s improves the performance of our models, and adding epipolar-guided weights can suppress false matches and further reduces the error. The detailed performance for each pose type under different frame latency settings is shown in Tables XI–XIII. From the tables, we can find that the proposed methods can perform especially better on the poses with larger movement between unsynchronized frames, e.g., Walking, WalkingDogs and WalkingTogether.

3) Ablation Study on γ for 3-D Pose Estimation: The ablation study on hyperparameter γ for the method CLS-epi for 3-D pose estimation is presented in Table XIV. In general, $\gamma = 0.01$ achieves better performance than other weights.

4) Model Size and Running Speed Comparison: We present the model sizes and running speed comparisons of our proposed models and the baselines for 3-D pose estimation in Table XV. The input resolution for the correlation step of the camera-view synchronization module is 48×48. As the original synchronized 3-D pose estimation model [2] is already very large, the running speed of the proposed models CLS-cor and CLS-epi is similar to the baseline methods BaseS/BaseSU/BaseU.

5) Visualization Results: Visualization results of unsynchronized 3-D pose estimation are presented in Figs. 6 and 7. In the figures, the first row shows the input unsynchronized multiview frames, and the top labels indicate the unsynchronized frame latency. Rows 2–8 show the 2-D key-joints projected from 3-D poses of Ground-truth, BaseS, BaseSU, BaseU, CLS-cor ($\gamma = 0$), CLS-cor, and CLS-epi, respectively, where *synchronized* frames are displayed for better visualization

Fig. 6. Examples of unsynchronized 3-D pose estimation (Walking Dogs). The first row shows the input unsynchronized multiview frames and the top labels indicate the unsynchronized frame latency (in seconds). The remaining rows show the ground-truth key joints and the predicted results. Blue lines are the 2-D key joints projected from 3-D poses, and the *synchronized* frames are used for better visualization. CLS-epi achieves the best performance among all methods, especially the prediction result of arms in view 0.

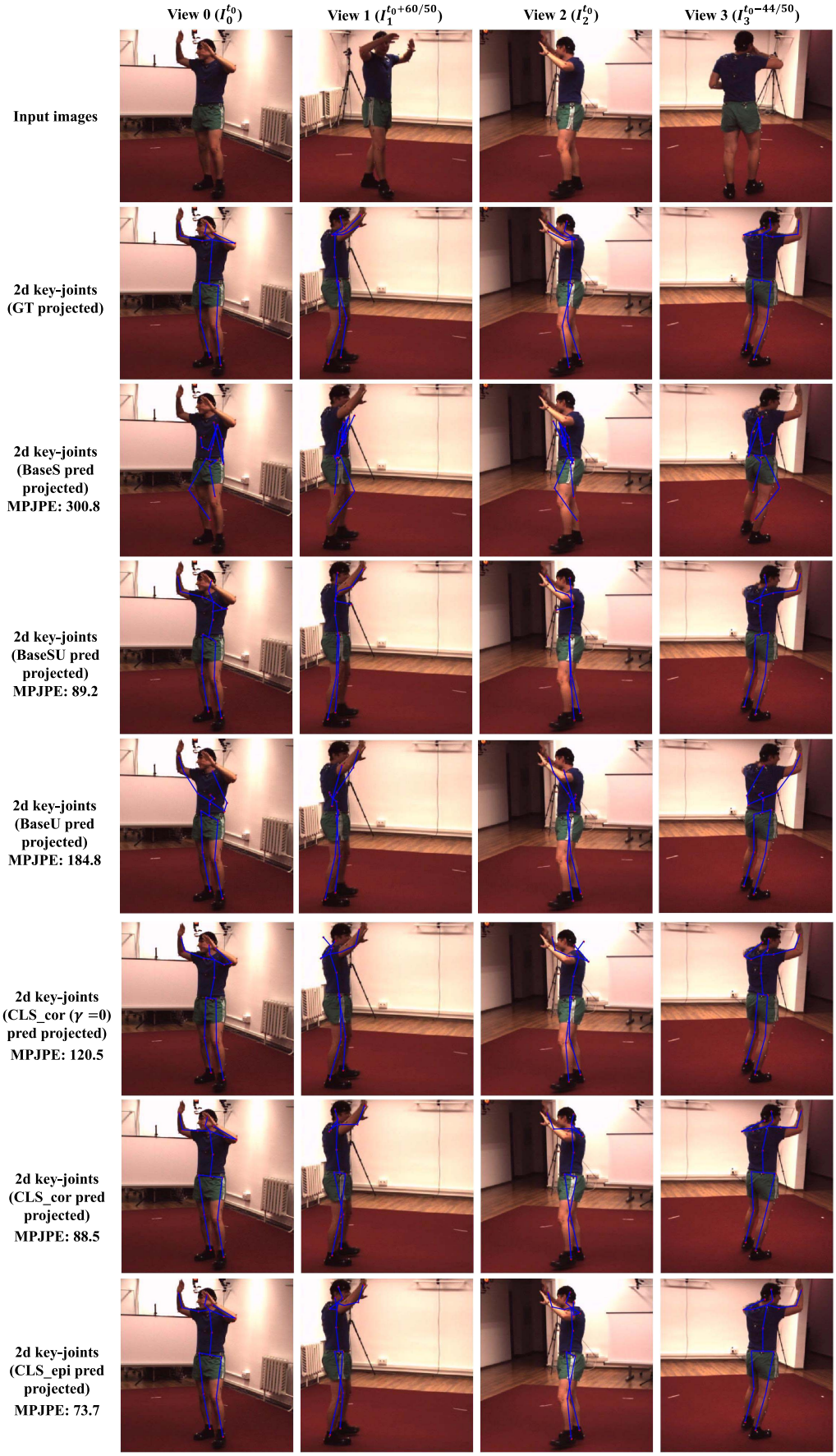

Fig. 7. Examples of unsynchronized 3-D pose estimation (Greeting). Blue lines are the 2-D key-joints projected from 3-D poses, and the synchronized frames are used for better visualization. CLS-epi achieves the best performance.

TABLE XV

MODEL PARAMETER NUMBER AND RUNNING SPEED COMPARISON OF THE BASELINE METHODS BASES/BASESU/BASEU AND THE PROPOSED CLS-COR AND CLS-EPI FOR 3-D POSE ESTIMATION ON THE HUMAN3.6M DATASET. THE INPUT RESOLUTION FOR THE CORRELATION STEP OF THE CAMERA-VIEW SYNCHRONIZATION MODULE IS 48×48

Method	Paras. Num	FPS
BaseS/BaseSU/BaseU	80.6M	3.7
CLS-cor	86.3M	3.4
CLS-epi	86.3M	3.0

effect. In Fig. 6, BaseU fails on the unsynchronized input, and CLS-epi achieves the best performance among all methods, especially the prediction of the arms in view 1. In Fig. 7, the CLS-epi also achieves the best performance among all comparison methods.

V. CONCLUSION

In this article, we focus on the issue of unsynchronized cameras in DNNs-based multiview computer vision tasks. We propose two view synchronization models based on single frames (not videos) from each view, SLS and CLS. The two models are trained and evaluated under two training settings (with or without synchronized frame pairs), and a similarity loss of the projected multiview features is proposed to boost the performance when synchronized training pairs are not available. Furthermore, to show its generality to different conditions of desynchronization, the proposed models are tested with desynchronization based on both constant and random latency. Finally, the proposed models are applied to unsynchronized multiview counting and unsynchronized 3-D human pose estimation, and achieve better performance compared to the baseline methods. Overall, CLS model using correlation and epipolar weights (CLS-epi) performs best among the proposed models.

In addition to unsynchronized multicamera crowd counting and 3-D pose estimation, the proposed method can also be applied to other multicamera vision tasks, such as multicamera detection [7], multicamera tracking [18]. In these tasks, multicameras may also be unsynchronized due to no synchronization clock or limited network bandwidth. As these DNN models [7], [18] generally follow the three-stage pipeline (single-view feature extraction, multiview projection and fusion, and prediction), our proposed synchronization modules can be inserted to adapt them to unsynchronized frames.

In our current model, image content matching is used for view synchronization, while the 2-D-to-3-D projection for multiview fusion relies on known camera parameters. The multicamera surveillance tasks themselves require known calibration for better multiview fusion. Note that our proposed view synchronization module based on correlation maps (CLS-cor) does not require camera calibrations due to the single-frame basis, and still achieves good performance. When the calibrations are provided, epipolar constraint can be utilized to achieve better results (CLS-epi). In future work, the 2-D–3-D projection in the original multiview models could be replaced with camera self-calibration modules, which can allow the model to handle unsynchronized and uncalibrated multicameras.

REFERENCES

[1] H. Aghajan and A. Cavallaro, *Multi-Camera Networks: Principles and Applications*. New York, NY, USA: Academic, 2009.

[2] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, "Learnable triangulation of human pose," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7718–7727.

[3] E. Remelli, S. Han, S. Honari, P. Fua, and R. Wang, "Lightweight multiview 3D pose estimation through camera-disentangled representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6040–6049.

[4] A. Kar, C. Háne, and J. Malik, "Learning a multi-view stereo machine," in *Proc. NIPS*, 2017, pp. 365–376.

[5] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "DeepMVS: Learning multi-view stereopsis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2821–2830.

[6] P. Baque, F. Fleuret, and P. Fua, "Deep occlusion reasoning for multicamera multi-target detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 271–279.

[7] T. Chavdarova *et al.*, "WILDTRACK: A multi-camera HD dataset for dense unscripted pedestrian detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5030–5039.

[8] L. Chen, H. Ai, R. Chen, Z. Zhuang, and S. Liu, "Cross-view tracking for multi-human 3D pose estimation at over 100 FPS," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3279–3288.

[9] Q. Zhang and A. B. Chan, "Wide-area crowd counting via ground-plane density maps and multi-view fusion CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8297–8306.

[10] M. Ye, X. Lan, Q. Leng, and J. Shen, "Cross-modality person re-identification via modality-aware collaborative ensemble learning," *IEEE Trans. Image Process.*, vol. 29, pp. 9387–9399, 2020.

[11] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3702–3712.

[12] M. Ye and P. C. Yuen, "PurifyNet: A robust person re-identification model with noisy labels," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2655–2666, 2020.

[13] M. Ye, J. Shen, X. Zhang, P. C. Yuen, and S.-F. Chang, "Augmentation invariant and instance spreading feature for softmax embedding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 2, pp. 924–939, Feb. 2022.

[14] L. Song *et al.*, "Unsupervised domain adaptive re-identification: Theory and practice," *Pattern Recognit.*, vol. 102, Jun. 2020, Art. no. 107173.

[15] E. Zheng, D. Ji, E. Dunn, and J.-M. Frahm, "Sparse dynamic 3D reconstruction from unsynchronized videos," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4435–4443.

[16] X. Zhang, B. Ozbay, M. Sznaier, and O. Camps, "Dynamics enhanced multi-camera motion segmentation from unsynchronized videos," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4668–4676.

[17] Q. Zhang and A. B. Chan, "3D crowd counting via multi-view fusion with 3D Gaussian kernels," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 12837–12844.

[18] Y. He, J. Han, W. Yu, X. Hong, X. Wei, and Y. Gong, "City-scale multi-camera vehicle tracking by semantic attribute parsing and cross-camera tracklet matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 576–577.

[19] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *Proc. ECCV*. Cham, Switzerland: Springer, 2016, pp. 628–644.

[20] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang, "Pix2 Vox: Context-aware 3D reconstruction from single and multi-view images," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2690–2698.

[21] M. Kocabas, S. Karagoz, and E. Akbas, "Self-supervised learning of 3D human pose using multi-view geometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1077–1086.

[22] C.-H. Chen *et al.*, "Unsupervised 3D pose estimation with geometric self-supervision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5714–5724.

[23] H. Joo *et al.*, "Panoptic studio: A massively multiview system for social motion capture," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3334–3342.

[24] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Harvesting multiple views for marker-less 3D human pose annotations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6988–6997.

[25] K. Takahashi, D. Mikami, M. Isogawa, and H. Kimata, "Human pose as calibration pattern: 3D human pose estimation with multiple unsynchronized and uncalibrated cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1775–1782.
[26] C. Albl, Z. Kukelova, A. Fitzgibbon, J. Heller, M. Smid, and T. Pajdla, "On the two-view geometry of unsynchronized cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4847–4856.
[27] T. Kuo, S. Sunderrajan, and B. S. Manjunath, "Camera alignment using trajectory intersections in unsynchronized videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1121–1128.
[28] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel, "Markerless motion capture with unsynchronized moving cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 224–231.
[29] C. Dai, Y. Zheng, and X. Li, "Subframe video synchronization via 3D phase correlation," in *Proc. Int. Conf. Image Process.*, Oct. 2006, pp. 501–504.
[30] F. Pádua, R. Carceroni, G. Santos, and K. Kutulakos, "Linear sequence-to-sequence alignment," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 304–320, Feb. 2010.
[31] C. Lei and Y.-H. Yang, "Tri-focal tensor-based multiple video synchronization with subframe optimization," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2473–2480, Sep. 2006.
[32] P. A. Tresadern and I. D. Reid, "Video synchronization from human motion using rank constraints," *Comput. Vis. Image Understand.*, vol. 113, no. 8, pp. 891–906, Aug. 2009.
[33] J. Yan and M. Pollefeys, "Video synchronization via space-time interest point distribution," in *Proc. Adv. Concepts Intell. Vis. Syst.*, vol. 1, 2004, pp. 12–21.
[34] Y. Caspi, D. Simakov, and M. Irani, "Feature-based sequence-to-sequence matching," *Int. J. Comput. Vis.*, vol. 68, no. 1, pp. 53–64, 2006.
[35] C. Rao, A. Gritai, M. Shah, and T. Syeda-Mahmood, "View-invariant alignment and matching of video sequences," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, Oct. 2003, pp. 939–945.
[36] B. Meyer, T. Stich, and M. Pollefeys, "Subframe temporal alignment of non-stationary cameras," in *Proc. Brit. Mach. Vis. Conf.*, 2008, pp. 1–10.
[37] E. Imre and A. Hilton, "Through-the-lens synchronisation for heterogeneous camera networks," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 1–11.
[38] D. Pundik and Y. Moses, "Video synchronization using temporal signals from epipolar lines," in *Proc. ECCV*, 2010, pp. 15–28.
[39] S. N. Sinha and M. Pollefeys, "Synchronization and calibration of camera networks from silhouettes," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2004, pp. 116–119.
[40] T. Gaspar, P. Oliveira, Favaro, and Paolo, "Synchronization two independently moving cameras without feature correspondences," in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switerland. Springer, Sep. 2014, pp. 6–12.
[41] I. Rocco, R. Arandjelovic, and J. Sivic, "Convolutional neural network architecture for geometric matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6148–6157.
[42] S. Phillips and K. Daniilidis, "All graphs lead to Rome: Learning geometric and cycle-consistent representations with graph convolutional networks," 2019, *arXiv:1901.02078*.
[43] H. Altwaijry, E. Trulls, J. Hays, P. Fua, and S. Belongie, "Learning to match aerial images with deep attentive architectures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3539–3547.
[44] T.-W. Hui, X. Tang, and C. C. Loy, "LiteFlowNet: A lightweight convolutional neural network for optical flow estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8981–8989.
[45] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2462–2470.
[46] M. Bai, W. Luo, K. Kundu, and R. Urtasun, "Exploiting semantic information and deep matching for optical flow," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2016, pp. 154–170.
[47] W. Yu, K. Yang, Y. Bai, H. Yao, and Y. Rui, "DNN flow: DNN feature pyramid based image matching," in *Proc. Brit. Mach. Vis. Conf.*, 2014, pp. 1–10.
[48] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2758–2766.
[49] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4161–4170.
[50] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8934–8943.
[51] H.-Y. Lai, Y.-H. Tsai, and W.-C. Chiu, "Bridging stereo matching and optical flow via spatiotemporal correspondence," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1890–1899.
[52] M. Jaderberg *et al.*, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.
[53] B. Xiao, H. Wu, and Y. Wei, "Simple baselines for human pose estimation and tracking," in *Proc. ECCV*, Sep. 2018, pp. 466–481.
[54] J. Y. Chang, G. Moon, and K. M. Lee, "V2 V-PoseNet: Voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5079–5088.
[55] A. Newell, K. Yang, and D. Jia, "Stacked hourglass networks for human pose estimation," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2016, pp. 483–499.
[56] J. Ferryman and A. Shahrokni, "PETS2009: Dataset and challenge," in *Proc. 12th IEEE Int. Workshop Perform. Eval. Tracking Surveill.*, Dec. 2009, pp. 1–6.
[57] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1325–1339, Dec. 2013.

Qi Zhang (Student Member, IEEE) received the B.Eng. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2014, the M.Eng. degree from the University of Chinese Academy of Sciences (UCAS), Beijing, China, in 2017, and the Ph.D. degree in computer science from City University of Hong Kong, Hong Kong, in 2021.

He was a Post-Doctoral Researcher with City University of Hong Kong, from 2021 to 2022. He currently works as the Distinguished Associate Researcher with the Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen, China. His research interests include multicamera surveillance and crowd counting.

Antoni B. Chan (Senior Member, IEEE) received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, USA, in 2000 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, CA, USA, in 2008.

He is currently a Full Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His research interests include computer vision, machine learning, pattern recognition, and music analysis.