Self-Supervised Human Pose based Multi-Camera Video Synchronization

Liqiang Yin*
College of Intelligence and Computing, Tianjin University
yinliqiang@tju.edu.cn

Ruize Han*
College of Intelligence and Computing, Tianjin University
han_ruize@tju.edu.cn

Wei Feng
College of Intelligence and Computing, Tianjin University
wfeng@tju.edu.cn

Song Wang
University of South Carolina, Columbia, USA
songwang@cec.sc.edu

ABSTRACT

Multi-view video collaborative analysis is an important task and has many applications in multimedia community. However, it always requires the given multiple videos to be temporally synchronized. Existing methods commonly synchronize the videos by the wired communication, which may hinder the practical application in real world, especially for moving cameras. In this paper, we focus on the human-centric video analysis and propose a self-supervised framework for the automatic multi-camera video synchronization. Specifically, we develop SeSyn-Net with the 2D human pose as input for feature embedding and design a series of self-supervised losses to effectively extract the view-invariant but time-discriminative representation for video synchronization. We also build two new datasets for the performance evaluation. Extensive experimental results verify the effectiveness of our method, which achieves the superior performance compared to both the classical and state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; **Matching**.

KEYWORDS

video synchronization, multi-view videos, human pose, self supervision.

ACM Reference Format:
Liqiang Yin, Ruize Han, Wei Feng, and Song Wang. 2022. Self-Supervised Human Pose based Multi-Camera Video Synchronization. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3503161.3547766

*Equal contribution. Corresponding authors: W. Feng and S. Wang.

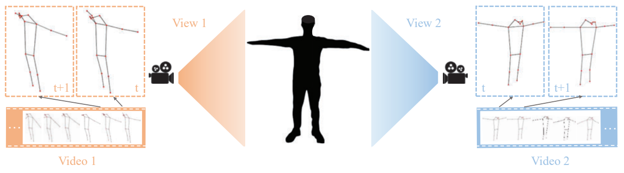


Figure 1: An illustration of human pose from two different perspectives, indicated by orange and blue boxes, respectively.

1 INTRODUCTION

Multi-view video collaborative analysis has attracted a lot of attention in recent years, given its importance in many multimedia applications, especially in various tasks of human-centric video-based analysis, e.g., multi-view based 3D human pose estimation [6, 8, 26], human association and tracking [11, 15–17, 36], human action recognition [12, 14, 18, 37], 3D reconstruction [4], etc. Most existing methods for these tasks require that the multi-view videos are strictly synchronized along the time axis, i.e., the corresponding frames in different videos are taken at the same time. The previous works usually assume that the given multi-view videos have been synchronized by the wired communication [21] or the manual annotation [47]. However, this assumption is not practical in many real-world applications, which may use the cameras without wired connections, e.g., the wearable mobile cameras. Under these conditions, it is very difficult to accurately synchronize the multi-view videos using the telecommunications, e.g., Wi-Fi, Bluetooth, or the built-in clock [43]. Therefore, exploring *automatic accurate video synchronization based on the video content* is fundamental and significant for the multi-view video collaborative analysis.

In this paper, we focus on the multi-view video synchronization for the human-centric video analysis. In particular, as shown in Figure 1, we first consider the videos covering the same person that are taken by two cameras with totally different views. Although the scene does not seem complex, the task of accurate (frame-level) video synchronization is very challenging. For the target frame in one view, the purpose of synchronization is to identify its corresponding frame in the other view. However, given the *subtle inter-frame visual difference in each view* (at adjacent frames) and the *significant cross-view difference at the same time* (in different views) as shown in Figure 1, the intra-view difference within a short time interval is very small, and much smaller than the cross-view difference of the same scene at the same time. This way, the challenge of this problem is to explore and model the discriminative features to capture both the intra-view difference and the cross-view invariance.

Previous works aim to handle this problem using some priors, e.g., the camera extrinsic parameters calibration [1, 10, 41, 46], or certain auxiliary conditions, e.g., the flickering of the flash [20, 31, 32]. These may hinder the practical application of such methods in the real world. In [43], the problem is modeled as a deep learning based classification problem using the human pose as feature. Each of the $\pm M$ class labels – represents an offset of the definite number of frames between the two videos. Limited number of labels, i.e., possible offsets, lead to limited accuracy of the resulting synchronization. As a supervised learning method, it also requires the ground-truth time offset between the videos as supervision for training. Another task similar to this work is video alignment, which aligns the key actions performed by a person in different videos according to the rhythm and procedure of the actions. Differently, it produces the action-level alignment results but not the frame-level synchronization results, making it not directly applicable to address our problem.

In this work, we aim to develop a self-supervised deep framework for the human pose based multi-camera video synchronization. Specifically, we propose to leverage the correspondence of the 2D human pose in different views for accurate (frame-level) video synchronization. We develop a **se**lf-supervised pose based **syn**chronization network (SeSyn-Net) for our task, in which we first obtain the frame-level feature based on the human pose estimation results using a spatial-temporal representation embedding network. We then design a series of self-supervised losses, including the self-identity loss, self-deviation loss, and interval-consistency loss as the self supervision, to train the feature embedding network. Among them, the self-identity loss implies the cyclic consistency between the videos, which is a classical property used in many self-supervised learning tasks. Besides, we also integrate the self-deviation loss and interval-consistency loss to model the discriminability and stationarity of the synchronization result, respectively. With the SeSyn-Net, we can get the video synchronization result by applying an easy inference strategy during testing stage. Furthermore, we build two datasets for performance evaluation and achieve the state-of-the-art performance (with the synchronization error less than 4 frames) on both datasets. The main contributions of this work are:

- This is the first work to develop a self-supervised framework for the accurate (frame-level) video synchronization.
- The proposed SeSyn-Net with a series of self-supervised losses can effectively extract the feature embeddings for video synchronization, which overcomes the challenge of subtle intra-view visual difference and significant cross-view difference.
- We build two benchmarks, i.e., NTU-SYN and CMU-SYN based on two public video datasets, for studying the task of video synchronization. Extensive experimental results verify the effectiveness of our method, which achieves the superior performance compared to both the classical and state-of-the-art methods. We released the benchmarks and code to the public at https://github.com/Yliqiang/SeSyn-Net.

2 RELATED WORK

Video temporal synchronization has been studied in the computer vision community for many years. Early methods based on the trajectory features usually use the principle of epipolar geometry [1, 41, 46], which require a clear geometric estimation as prior. The appearance feature based methods [10, 49] usually apply the Histogram of Oriented Gradients (HoG) feature [7] or SIFT feature [24] from the image. Some of such methods [10, 27] also need the given of basic matrix between cameras as input. In addition, in [22], a self-similarity based action descriptor is proposed, which can be used for the video synchronization and action recognition tasks. In [23], the motion state transformation of the feature points in the video is defined as an event, and the corresponding relationship of each event is used for video synchronization. In [35], the synchronization is performed according to the rhythm of the joints between move and stop. Other methods rely on the flickering of the flash in the video for synchronization [20, 31, 32]. Obviously, the above methods have limitations of requiring some auxiliary priors, e.g., the camera pose, which may not be available in practice. In [43], this task is modeled as a deep learning based classification problem using the human pose as feature, which, however, requires the annotated time offset between the videos, and two videos with the same length as input. This cannot meet the requirements of video synchronization in real-world applications. Different from the above methods, we solve this problem by developing a self-supervised learning framework, which does not depend on the priors or auxiliaries, e.g., the camera pose, flash flickering, and greatly improves the practicability of the method.

Key action based video alignment shows certain similarity to our problem. The purpose of the former is to align the key actions performed by a target (usually a human) in different videos according to the rhythm and procedure of the actions. For example, the videos of two people playing golf in two scenes can be aligned according to the consistency of a series of actions, such as swinging and hitting the ball. This task has received widespread attention in recent years. Previous methods first extract the appearance feature [38, 48] or the trajectory feature [2, 25, 29], and then calculate the distance matrix between the two videos according to the characteristics of each video, and finally a time warping algorithm, e.g., Dynamic Time Warping (DTW) [3], is applied to achieve the action alignment. Recently, many deep learning methods use the appearance features [9, 19, 28], and a few use the pose features [33] to handle this problem.

However, the action alignment task is fundamentally different from our problem in real-application scenarios – the former focuses on the key procedure and action alignment from the similar activities occurring at different times, while the latter aims to achieve the accurate clock synchronization of different videos covering the same scene. Also, the methods for the former can not be directly applied to the latter (frame-level synchronization), since they usually use the feature extracted from the multi-frame clip that is not distinguishable for each frame.

Specific scene based video alignment is proposed to align the videos taken by the moving cameras for the static scenes. For example, the street view videos of the same road taken at several months apart can be aligned according to some specific content (i.e., the same street signs, houses, trees, etc.). To handle this problem, several methods [39, 40, 42] are developed by first computing the distance matrix through the appearance features, and then applying the DTW to carry out the alignment, which is similar to many action-alignment methods discussed above. Note that, in this problem, the scene in the video is mostly static. Even if a dynamic object, e.g., a moving car, appears in one video, it does not appear in the other video with a large temporal gap of the shooting time. With this, we can see that the scene alignment problem is also different from our work.

3 PROPOSED METHOD

3.1 Overview

Given two non-synchronized videos containing the same moving subject (person) taken from different perspectives, assuming the frame rate of the two videos is consistent, we aim to find the optimal time offset so that they can be synchronized in time after edit. In this work, we model the above task as a frame association problem. Specifically, we associate each pair of frames across the two videos according to their visual contents. With the optimized matching pairs, each frame pair provides a time offset, and we finally combine all the offsets to determine the offset between these two videos.

Figure 2 shows the overall framework of our method. The input is the human pose sequences from the two views (referred to as views u and v), which can be denoted as $\boldsymbol{P}^u = \{p_1^u, p_2^u, \cdots, p_M^u\} \in \mathbb{R}^{M\times K\times D}$, $\boldsymbol{P}^v = \{p_1^v, p_2^v, \cdots, p_N^v\} \in \mathbb{R}^{N\times K\times D}$, where M and N are the video lengths, K represents the number of human joints and D represents the feature dimension of each pose joint. Here the pose features can be obtained by an existing human pose estimation method, e.g., HRNet [34], which produces the 2D coordinates and a 1D confidence scores for each joint. Next, the input sequences are fed into a two-branch feature embedding network with shared weights to get the representations. Finally, we design three self-supervised loss functions to train the feature embedding network for the synchronization task.

3.2 Feature Embedding Network

Since our method aims to leverage the human pose self similarity in different views at the same time as a clue for video synchronization, a feature embedding network that can effectively model the spatio-temporal information of human pose is required. For this, we choose the graph convolutional network (GCN), which has been widely used in the skeleton-based action recognition problem [5, 13, 45]. It builds the graph structure on the skeleton sequence to better learn the spatio-temporal information of the human pose. Specifically, we use ST-GCN [44] alike structure as the backbone network. Original ST-GCN generates the feature of the whole video. In order to make it suitable for our work, we make several modifications on the original network structure. First, in order to obtain the frame-level feature representation, we change the stride of the temporal convolution layer from 2 to 1, and use the features before the final pooling operation. Second, to reduce the complexity, we remove the last convolution layer of ST-GCN and change the number of the final output channel from 256 to 128.

The input of the feature embedding network are two skeleton sequences without time stamp annotation, and the output is the frame-based embedding of the two sequences, denoted as $\boldsymbol{F}^u \in \mathbb{R}^{M\times C}$ and $\boldsymbol{F}^v \in \mathbb{R}^{N\times C}$, where M and N are the sequence lengths, and C is the dimension of embedded features. Here $C = K \times 128$ that embeds a 128-dimensional vector for each joint.

3.3 Self-Supervised Learning Loss

As discussed above, we aim to design a self-supervised framework in this work. Self-supervised learning methods commonly use the pretext tasks to excavate the supervision information from unlabeled data for training. In this work, we use the self-identity property of the human pose in different views at the same time as a pretext task, to construct the self-supervised learning losses for training the feature embedding network.

Self-identity loss. In this work, we aim to learn the embeddings of the human pose taken at the same time in different perspectives to have the self-identity, i.e., the *view-independent and time-discriminative feature embeddings*. As shown in Figure 2, given two embeddings from the frames taken at the same time (absolute time), i.e., $\boldsymbol{F}_m^u \in \mathbb{R}^C$ in video V^u at frame m and $\boldsymbol{F}_n^v \in \mathbb{R}^C$ in video V^v at frame n, we hope that $\boldsymbol{F}_n^v$ should be the most similar frame to $\boldsymbol{F}_m^u$ among all frames in $\boldsymbol{F}^v$ of video V^v, and vice versa.

To formulate the above concepts, we can measure the similarities between all frames in the two embedding sequences by $\boldsymbol{S}^{u,v} \in \mathbb{R}^{M\times N}$ as

$$\boldsymbol{S}^{u,v}(m,n) = -\|\boldsymbol{F}_m^u - \boldsymbol{F}_n^v\|^2, \tag{1}$$

where m, n denote the index of row and column in $\boldsymbol{S}^{u,v}$ and $\boldsymbol{S}^{u,v}(m,n)$ represents the similarity score between the embeddings from the frame m in the video V^u and the frame n in video V^v. Then we get the frame-to-frame matching matrix based on similarity by performing row softmax on $\boldsymbol{S}^{u,v}$

$$\boldsymbol{X}^{u,v}(m,n) = \frac{\exp(\boldsymbol{S}^{u,v}(m,n))}{\sum_{k=1}^{N}\exp(\boldsymbol{S}^{u,v}(m,k))}, \tag{2}$$

and we have $\boldsymbol{X}^{u,v}(m,n) \in [0,1]$.

Clearly, the one-on-one frame matching from video V^u to video V^v can be obtained by performing the *argmax* operation on each row of $\boldsymbol{X}^{u,v}$. Since the *argmax* operation is not differentiable, as in [9], we directly take each row in $\boldsymbol{X}^{u,v}$ as weights to obtain the best matched frame in video V^v for each frame in video V^u,

$$\boldsymbol{F}^{\widetilde{v}} = \boldsymbol{X}^{u,v} \cdot \boldsymbol{F}^v \in \mathbb{R}^{M\times C}, \tag{3}$$

where $\boldsymbol{F}^{\widetilde{v}}$ represent the embeddings of the most similar frames in video V^v to those in video V^u. Note that, the $\boldsymbol{F}^{\widetilde{v}}$ are actually the *reconstructed* embeddings from the frames in video V^v with the matching weights in $\boldsymbol{X}^{u,v}$.

After obtaining $\boldsymbol{F}^{\widetilde{v}}$, we can then measure its similarity with $\boldsymbol{F}^u$ to verify whether the most similar frame found in video V^v to each frame in video V^u satisfy the self-identity condition. The similarity matrix between $\boldsymbol{F}^{\widetilde{v}}$ and $\boldsymbol{F}^u$, referred to as $\boldsymbol{S}^{\widetilde{v},u} \in \mathbb{R}^{M\times M}$, can be calculated as

$$\boldsymbol{S}^{\widetilde{v},u}(m',m) = -\|\boldsymbol{F}_{m'}^{\widetilde{v}} - \boldsymbol{F}_m^u\|^2, \tag{4}$$

whose elements $\boldsymbol{S}^{\widetilde{v},u}(m',m)$ represents the similarity between the m'-th embedding feature in $\boldsymbol{F}^{\widetilde{v}}$ to the m-th embedding feature in $\boldsymbol{F}^u$.

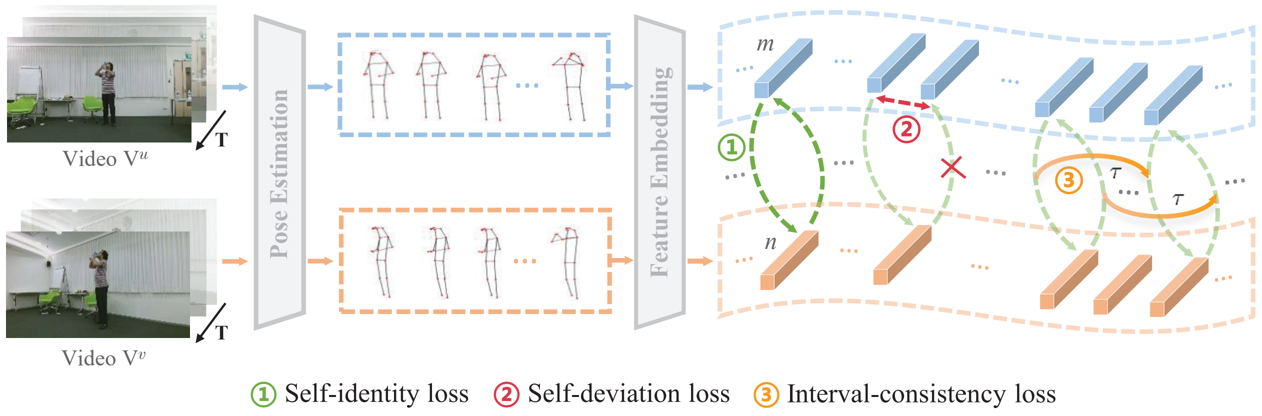

Video V^u
T
Video V^v
T
Pose Estimation
Feature Embedding
m
n
① ② ③
τ
τ
① Self-identity loss ② Self-deviation loss ③ Interval-consistency loss

Figure 2: Overall framework of the proposed method. Given two videos from different cameras, we first obtain the poses of the person in the videos through a pose estimation network, and then obtain the features of each frame through the feature embedding network. Finally, three self-supervised losses are constructed based on the embedded features to train the synchronization network, where m and n represent the features from the m-th and n-th frames in two videos, respectively, and τ represents the cross-frame interval.

For $\forall k \in \{1, 2, ..., M\}$, since $\boldsymbol{F}_k^{\tilde{v}}$ represents the most similar frame in the reconstructed embedding sequence to the k-th frame embedding $\boldsymbol{F}_k^u$ in video V^u, ideally, the diagonal elements in $\boldsymbol{S}^{\tilde{v},u}$ have the largest similarity in each row, that is, satisfying the self-identity condition. To compel the embedding features to satisfy the self-identity as much as possible, we define the following loss function

$$\mathcal{L}_{\text{iden}} = -\sum_{m=1}^{M} \boldsymbol{y}_m \log \hat{\boldsymbol{y}}_m, \tag{5}$$

where $\hat{\boldsymbol{y}}_m = \boldsymbol{S}^{\tilde{v},u}(m, 1:M) \in \mathbb{R}^{1\times M}$ represents the m-th row in $\boldsymbol{S}^{\tilde{v},u}$, ideally its m-th element has the largest value. We denote a one-hot vector $\boldsymbol{y}_m \in \mathbb{R}^{1\times M}$, whose m-th element is 1, and the other elements are all 0. The self-identity loss is defined as the cross entropy loss between $\boldsymbol{y}_m$ and $\hat{\boldsymbol{y}}_m$.

Self-deviation loss. Self-identity loss restricts the elements on the diagonal of $\boldsymbol{S}^{\tilde{v},u}$ to have the largest value in their respective rows in a classified manner, but does not penalize the situation of not meeting this condition. For this reason, we design the following self-deviation loss to help the network learn the feature embedding better.

Specifically, as shown in Figure 2, we first take the frame indices from the video V^u, i.e.,

$$\boldsymbol{g} = [1, 2, 3, \cdots, M-1, M]^T \in \mathbb{Z}^{M\times 1}. \tag{6}$$

Given the frames with the above indices, we identify the corresponding frames in another view v orderly, from which we go back to video V^u to match the corresponding frames again. Ideally, the backtracked frame indices are still the vector $\boldsymbol{g}$.

To model the above situation, as before, we perform the row softmax on $\boldsymbol{S}^{\tilde{v},u}$ to obtain the matching matrix $\boldsymbol{X}^{\tilde{v},u} \in \mathbb{R}^{M\times M}$ from the reconstructed embedding sequence $\boldsymbol{F}^{\tilde{v}}$ to $\boldsymbol{F}^u$,

$$\boldsymbol{X}^{\tilde{v},u}(m', m) = \frac{\exp(\boldsymbol{S}^{\tilde{v},u}(m', m))}{\sum_{k=1}^{M} \exp(\boldsymbol{S}^{\tilde{v},u}(m', k))}, \tag{7}$$

where m', m denote the index of row and column in $\boldsymbol{X}^{\tilde{v},u}$, respectively. With the original index vector $\boldsymbol{g}$ and the matching matrix $\boldsymbol{X}^{\tilde{v},u}$, we perform the permutation operator on the original index vector $\boldsymbol{g}$ as

$$\boldsymbol{p} = \boldsymbol{X}^{\tilde{v},u} \cdot \boldsymbol{g} \in \mathbb{R}^{M\times 1}. \tag{8}$$

where $\boldsymbol{p}$ is regarded as the backtracked frame indices of the original $\boldsymbol{g}$. Ideally, $\boldsymbol{X}^{\tilde{v},u}$ is an identity matrix, so $\boldsymbol{p}$ should be equal to $\boldsymbol{g}$. But, in practice $\boldsymbol{X}^{\tilde{v},u}$ is not an ideal identity matrix, so the result $\boldsymbol{p}$ is a real-value vector. This way, we define the self-deviation loss to measure the deviation between the original $\boldsymbol{g}$ and the predicted $\boldsymbol{p}$ as

$$\mathcal{L}_{\text{devi}} = \frac{\sum_{m=1}^{M} |p_m - g_m|}{M}, \tag{9}$$

where p_m, g_m represent the m-th element in $\boldsymbol{p}$ and $\boldsymbol{g}$, respectively.

Interval-consistency loss. Let's further consider an analogy, between the self-supervision training and a shooting game. The constraint of self-identity loss is analogue to whether hitting the target, and the self-deviation loss is analogue to the distance error between each shot result and the target. Both of them are measured by the results against the target. We further consider the interval-consistency loss, which is analogue to the relative shooting stability over time.

As discussed above, $\boldsymbol{p}$ is the backtracked index vector of the original index vector $\boldsymbol{g}$. We consider imposing a consistency loss on $\boldsymbol{p}$ and $\boldsymbol{g}$ as

$$\mathcal{L}_{\text{intv}} = \frac{\sum_{m=\tau+1}^{M} |(p_m - p_{m-\tau}) - (g_m - g_{m-\tau})|}{M - \tau}, \tag{10}$$

where τ is the interval of the frame index. This constraint requires that the interval of the backtracked indices should be approximated to the corresponding one of the original indices.

We empirically set $\tau = 1$ in this work and the above loss is reformulated as

$$\mathcal{L}_{\text{intv}} = \frac{\sum_{m=2}^{M} |p_m - p_{m-1} - 1|}{M - 1}, \tag{11}$$

which denotes that we focus on the continuous two frames, i.e., the adjacent frames in the original sequence are desired to be backtracked also as the adjacent frame indices. Note that, this loss is used to constrain the consistency between the intervals in $\boldsymbol{p}$ and $\boldsymbol{g}$. Therefore, given the continuity of the original sequence frames arranged by $\boldsymbol{g}$, this loss can also maintain the continuity and stability of the backtracked adjacent frame indices p_m and p_{m-1}.

Total Loss. Finally, we define the self-supervised loss as

$$\mathcal{L} = \mathcal{L}_{\text{iden}} + \mu_1 \mathcal{L}_{\text{devi}} + \mu_2 \mathcal{L}_{\text{intv}}, \tag{12}$$

where μ_1, μ_2 are the weighting parameters for balancing the three loss terms.

3.4 The Framework

Inference. We train the feature embedding network described in Section 3.2 using the self-supervised loss proposed in Section 3.3. In the inference stage, we directly use the trained network to perform the task of video synchronization. The specific process for optimal offset determination is as follows:

❶ *Feature embedding*. Given two videos V^u and V^v, we first use HRNet and the proposed feature embedding network to get the representation of all frames, denoted as $\boldsymbol{F}^u$ and $\boldsymbol{F}^v$.

❷ *Frame matching*. According to Eqs. (1) and (2), we calculate the matching matrix $\boldsymbol{X}^{u,v}$ between $\boldsymbol{F}^u$ and $\boldsymbol{F}^v$, and then perform the *argmax* operation on each row of $\boldsymbol{X}^{u,v}$ to obtain the frame matching vector between V^u and V^v, which is denoted as

$$\boldsymbol{o} = [o_1, o_2, \cdots, o_L] \in \mathbb{Z}^{1 \times L}, \tag{13}$$

where L is equal to the length of V^u , the index of each element represents the corresponding frame index in V^u, and the value of each element in $\boldsymbol{o}$ represents the matched frame index in V^v.

❸ *Offset determination.* We determine the final time offset between V^u and V^v by simply calculating the median of the offsets of all matched frame pairs, i.e.,

$$o = \lfloor \text{Med}(o_i - i) \rfloor, \quad i = 1, 2, \cdots, L \tag{14}$$

where the positive/negative value of o means that V^u is delayed or advanced in time axis against V^v.

Implement details. The proposed SeSyn-Net was implemented using PyTorch. We train SeSyn-Net using SGD for 400 epochs for both datasets on an NVIDIA RTX 3090 GPU. The learning rate of the SGD optimizer is 0.005, the Nesterov momentum is 0.9, and the weight decay is 0.0001. In Eq. (12) the weight parameters μ_1, μ_2 are set to 0.5, 0.035 for NTU-SYN and 0.7, 0.015 for CMU-SYN.

4 EXPERIMENTS

4.1 Datasets

We can not find applicable benchmarks for the proposed human centric multi-view video synchronization problem. The most related dataset in [43] is not applicable, since the length of the videos in the dataset is short (32 frames), which is also not released to the public. For performance evaluation, we built two datasets, NTU-SYN and CMU-SYN for the multi-view video synchronization task.

NTU-SYN Dataset is built based on the NTU RGB+D dataset [30], which is used for multi-view action recognition. Each pair of videos is taken from a different perspective, which contains a person performing various actions. The shooting frame rate is 30 fps, and the average duration is 120 frames. According to the cross-subject division principle of the original dataset, we selected a total of 5,560 pairs of synchronized videos, including 3,762 pairs of videos as the training set and 1,798 pairs of videos as the testing set. For initially synchronized videos, we randomly set a time offset in the range of [-30,30] frames for constructing each video pair in both the training set and the testing set.

CMU-SYN Dataset is constructed from CMU Panoptic Studio dataset [21]. Based on the original long video pairs taken from different perspectives, 106 pairs of videos are trimmed with the duration of 480 frames. We randomly select 74 pairs of videos to construct the training set, and the remaining 32 pairs of videos to construct the testing set. By setting 6 different time offsets for each pair of videos, we finally get 74×6 pairs of videos as the training set, and 32×6 pairs of videos as the testing set, and the range of time offset for each video pair is also [-30,30] frames.

4.2 Evaluation Metrics

In order to comprehensively evaluate the performance of our method on the multi-view video synchronization task, we define several evaluation metrics as below.

Metric I: Frame error. Frame error is calculated as $Frm.err. = |P_i - R_i|$, where P_i and R_i refer to the predicted and the ground-truth time offset (using frame as unit), respectively. For the whole testing set, we use the average value of the frame error of all video pairs. In addition, we also compute the ratio of the video pairs with the frame error less than 1, 5, and 10 frames among all testing sample, which are denoted as Acc@1, Acc@5 and Acc@10, respectively.

Metric II: Absolute error. The absolute error is a measure of the frame error in time, which represents the error between the predicted time offset and the ground-truth time offset, and is calculated as $Abs.err. = Frm.err./fps$, where fps represents the frame rate of the input videos.

Metric III: Relative error. The relative error is used to measure the relative improvement between the predicted time offset and the initial offset, which is defined as $Rel.err. = \frac{\sum_{i=1}^{N} Frm.err.}{\sum_{i=1}^{N} |R_i|}$, where N is the total number of video pairs for testing. We also evaluate $Rel.err.$ on each testing video pair, and then compute the ratio of the testing samples with the relative errors less than 10%, 30% and 50% among the total number of testing samples, which are denoted as rAcc@10, rAcc@30 and rAcc@50, respectively. Note that, the testing samples with $R_i = 0$ do not participate in the calculation of rAcc@10, rAcc@30, and rAcc@50, because 0 cannot be the denominator.

4.3 Experimental Results

Comparison Methods. According to Section 2, there are not many existing methods that can be directly applied to synchronization problems. Therefore, we include several related methods by modifying them for a fair comparison.

• ***Simple***: Firstly, we construct a straightforward method *Simple*, which does not go through the feature embedding network, but directly applies the inference strategy proposed in our paper to perform video synchronization based on the pose estimation results.

Table 1: Comparative results of different methods on the NTU-SYN and CMU-SYN using the metrics of Frm.err. (frame), Acc@1 (%), Acc@5 (%), Acc@10 (%), Abs.err. (second) and Rel.err. , rAcc@10 (%), rAcc@30 (%), rAcc@50 (%).

Dataset	Method	Frm. err. ↓	Acc@1 ↑	Acc@5 ↑	Acc@10 ↑	Abs. err. ↓	Rel. err. ↓	rAcc@10 ↑	rAcc@30 ↑	rAcc@50 ↑
NTU-SYN	Simple	19.55	9.62	27.25	42.27	0.6516	1.28	9.16	21.59	32.17
	SSM [22]	7.52	22.80	60.23	78.36	0.2507	0.49	15.77	55.57	72.19
	SynNet [43]	5.98	44.88	75.08	83.82	0.1993	0.41	43.64	64.10	77.91
	TCC [9]	5.48	36.60	61.96	81.20	0.1827	0.36	28.15	51.78	70.66
	LAV [19]	7.16	34.93	60.46	75.64	0.2388	0.47	31.66	51.84	63.71
	Ours	**3.39**	**62.29**	**80.65**	**90.21**	**0.1128**	**0.22**	**54.83**	**73.26**	**82.08**
CMU-SYN	Simple	39.12	4.69	22.40	33.33	1.3040	2.48	3.70	22.22	29.10
	SSM [22]	8.15	30.21	71.35	75.52	0.2715	0.52	28.04	64.02	71.43
	SynNet [43]	18.57	8.85	23.44	33.85	0.6188	1.22	9.60	16.38	37.85
	TCC [9]	15.57	18.23	41.14	59.38	0.5191	0.99	17.99	41.27	53.44
	LAV [19]	25.06	13.02	23.96	36.46	0.8352	1.59	10.58	21.69	32.28
	Ours	**2.27**	**69.27**	**91.67**	**93.75**	**0.0755**	**0.14**	**62.43**	**78.84**	**85.71**

Table 2: Ablation study of the proposed method using different self-supervised losses.

Dataset	Method	Frm. err. ↓	Acc@1 ↑	Acc@5 ↑	Acc@10 ↑	Abs. err. ↓	Rel. err. ↓	rAcc@10 ↑	rAcc@30 ↑	rAcc@50 ↑
NTU-SYN	$\mathcal{L}_{\text{iden}}$	4.17	47.78	71.91	87.43	0.1390	0.27	39.46	63.43	78.01
	$\mathcal{L}_{\text{iden+devi}}$	3.67	52.73	75.25	90.10	0.1225	0.24	44.38	67.33	82.25
	$\mathcal{L}_{\text{iden+devi+intv}}$	3.39	62.29	80.65	90.21	0.1128	0.22	54.83	73.26	82.08
CMU-SYN	$\mathcal{L}_{\text{iden}}$	16.82	10.94	36.46	49.48	0.5608	1.07	11.64	26.98	33.86
	$\mathcal{L}_{\text{iden+devi}}$	12.30	36.46	55.21	67.71	0.4099	0.78	31.22	48.15	58.20
	$\mathcal{L}_{\text{iden+devi+intv}}$	2.27	69.27	91.67	93.75	0.0755	0.14	62.43	78.84	85.71

• ***SSM***: We also include a non-deep learning method *SSM* [22], which performs video synchronization by computing self-similarity-based action descriptors. To keep the input features consistent for fair comparison, we compute self-similarity based on trajectory features for video synchronization.

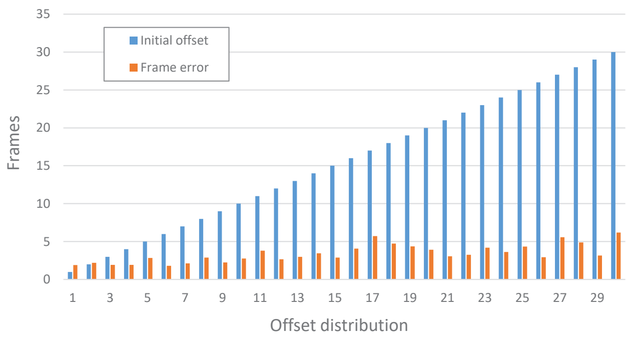
• ***SynNet***: Besides, we also use *SynNet* [43] as a comparison method, which is a fully-supervised video synchronization method, and we downsample the collected datasets in the same way as the original article for a fair comparison.

• ***TCC*** and ***LAV***: Finally, we include two related methods for comparison, *TCC* [9] and *LAV* [19], which are two key action-based video alignment methods. Note that, both methods use the appearance features, in which the feature extraction network merges the multi-frame sequence as a unified representation, thus cannot perform the frame-level video synchronization task. Therefore, we replace the feature and embedding network with the settings adopted in our paper, while retaining the original training strategy and loss function for a fair comparison.

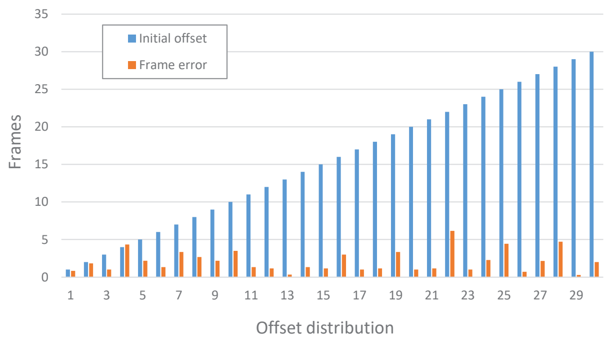
Result comparisons. Table 1 shows the comparison results on the two datasets. We can see that, on the NTU-SYN dataset, most methods provide the frame synchronization errors (Frm. err.) within 10 frames, and our method achieves the most accurate synchronization errors of 3.39 frames. For other metrics, our method also achieves the best performance. On the CMU-SYN dataset, we can see that the results of many comparison methods, e.g., Simple, SynNet, TCC and LAV, have deteriorated a lot, which may be caused by the larger cross-view difference in this dataset. On CMU-SYN, only our method and SSM still provide frame synchronization error within 10 frames, in which our method significantly decreases the frame error to 2.27. This indicates that the self-supervised losses proposed by us can well constrain the matching between videos and make use of its rich information. For the absolute error (Abs. err.), we can see that the proposed method produces the results with the error of about 0.1 second, which is accurate enough for many real applications. Also, for the relative error (Rel. err.), we can see that some methods provide the results with a relative error exceeding 1, which means the synchronization results are larger than the initial offset between the given videos. The proposed method obtains promising results with the relative errors of 0.22 and 0.14 on two datasets, respectively, which significantly reduces asynchrony between videos. To sum up, the proposed method performs better than other methods by a large margin in all metrics on both datasets.

4.4 Ablation Study

Effectiveness of self-supervised loss. We verify the effectiveness of the three self-supervised losses, *i.e.*, $\mathcal{L}_{\text{iden}}$, $\mathcal{L}_{\text{devi}}$ and $\mathcal{L}_{\text{intv}}$, as shown in Table 2. By adding each loss item by item, we can see that the equipment of each loss will lead to an improvement for video synchronization performance on both datasets. In particular, on the CMU-SYN dataset, the equipment of $\mathcal{L}_{\text{devi}}$ and $\mathcal{L}_{\text{intv}}$ leads to a significant performance improvement, which demonstrates that the proposed self-supervised losses are very effective, especially under the challenge scenarios. Specifically, as mentioned in Section 3.3, $\mathcal{L}_{\text{iden}}$ constrains the video sequences to satisfy the self-identity property, but does not penalize those who do not satisfy this property with different punishment strength – the further between the backtracked position and the origin position, the greater penalty. This is inadequate to handle the complex scenes. The designed



(a) Results on NTU-SYN



(b) Results on CMU-SYN

Figure 3: Analysis of influence of different initial offsets on video synchronization results.

$\mathcal{L}_{\text{devi}}$ discriminatively penalizes the frames not satisfying the self-identity constraint, and $\mathcal{L}_{\text{intv}}$ considers the internal relationship of multiple adjacent frames. That is, the proposed self-deviation loss $\mathcal{L}_{\text{devi}}$ can penalize the false matches with similar pose feature but with large temporal interval and the interval-consistency loss constrains the continuity and stability of matching results to avoid the accidental errors. The effectiveness of two losses is especially verified by the performance improvement on the CMU-SYN.

Influence of input features. We also explore the impact of using different features as the input for our method. For this purpose, motion features and appearance features are applied in our method. For the motion features, we use the flow of the joint between adjacent frames and use the same feature embedding network in our method to extract features for synchronization. For the appearance features, we use the ResNet followed by an LSTM layer as extractor. Table 3 shows the synchronization performance on both datasets of our method using different input features. Note that, we do not include the results of the appearance feature based method on CMU-SYN since the training of it takes much GPU memory, which can not be implemented on the GTX 3090. We can see that using motion features can also provide acceptable synchronization performance, but frame-level video synchronization using appearance features is very difficult.

Table 3: Ablation study of different input features.

Dataset	Feature	Frm. err.	Acc@1	Rel. err.	rAcc@10
NTU-SYN	w motion	3.72	53.67	0.24	44.26
	w appearance	27.51	4.00	1.80	3.96
	w pose (Ours)	3.39	62.29	0.22	54.83
CMU-SYN	w motion	4.83	46.35	0.31	47.62
	w pose (Ours)	2.27	69.27	0.14	62.43

Influence of the number of joints. We also explore the impact of using different numbers of joints as input. We select 5 (nose, shoulders and hips), 9 (nose, shoulders, wrists, hips, ankles) and 13 (nose, shoulders, elbows, wrists, hips, knees, ankles) joints as input from the 17 joints used in our method. Furthermore, We reconstruct the skeleton graph of ST-GCN according to the number of joints and the neighboring relationship to ensure the reliability of the experimental results. Table 4 shows the synchronization performance of our method using the different number of joints on both datasets. We can see that using more joints usually leads to better performance, and use 17 joints can provide the best performance. However, there are exceptions that the performance decreases with the input of 13 joints on NTU-SYN. This maybe because of the experimental uncertainty, e.g., the suboptimal training model. Finally, our method uses 17 joints as input for the best synchronization performance.

Table 4: Ablation study of the proposed method with different number of human joints.

Dataset	Joints	Frm. err.	Acc@1	Rel. err.	rAcc@10
NTU-SYN	w 5 joints	6.78	40.66	0.44	33.52
	w 9 joints	4.67	58.51	0.31	50.42
	w 13 joints	5.80	39.93	0.38	33.52
	w 17 joints (Ours)	3.39	62.29	0.22	54.83
CMU-SYN	w 5 joints	11.24	7.81	0.71	6.88
	w 9 joints	10.21	9.9	0.65	6.88
	w 13 joints	6.95	16.67	0.44	10.05
	w 17 joints (Ours)	2.27	69.27	0.14	62.43

4.5 Experimental Analysis

We conduct more experiments for the in-depth analysis of the proposed method under different conditions.

Analysis of different initial offsets. Figure 3 shows the frame errors (Frm. err.) of the proposed method on NTU-SYN and CMU-SYN under different initial offsets. The blue column representing the initial offset is gradually getting higher, but the orange column representing the average frame error under each initial offset shows no significant correlation. This demonstrates the stability of our method for synchronizing with different initial offsets. Furthermore, for each initial offset, the height ratio of two columns reflects that the asynchrony between the videos is significantly reduced after applying our method, which is more remarkable on CMU-SYN.

Analysis of video with dropped frames. We also explored the synchronization performance when there is a small amount of dropped frames in the videos. This way, during the testing stage, we randomly drop out a few frames on each video and perform the synchronization of them. The results are shown in Table 5, where 'Drop rate' represents the percentage of dropped frames. It can be

Table 5: Synchronization results under different frame drop rate.

Dataset	Drop rate	Frm. err.	Acc@1	Abs. err.
NTU-SYN	0%	3.39	62.29	0.1128
	1%	3.37	59.40	0.1136
	3%	3.37	58.62	0.1158
	5%	3.47	54.73	0.1217
	10%	3.85	34.98	0.1428
CMU-SYN	0%	2.27	69.27	0.0755
	1%	2.51	57.29	0.0845
	3%	2.97	51.04	0.1022
	5%	3.19	41.67	0.1120
	10%	3.66	28.65	0.1354

seen that, in general, the synchronization performance decreases with the increase of the frame drop rate. We can also see from the results on NTU-SYN that when the frame drop rate is 1%, the frame error changes very little, and the proportion of test samples with an error within 1 frame (Acc@1) is still approximate to 60%. When the frame drop rate is 10%, the synchronization result obtained by the proposed method is still acceptable, where the frame errors on both datasets are still within 4 frames. Note that, given the frame dropping, the frame rate of the video is not strictly with 30 fps like the original one. Here we provide the absolute error, i.e., Abs. err., using the second as unified measurement unit in the last column. We find that, even with 10% frame drop, which is actually severe in many cases, the proposed method can produce a synchronization accuracy within 0.15 second that is satisfied the requirement of many applications. In the above experiments, the frame drop out is only performed during testing, and the network training is still conducted on the original videos.

Analysis of synchronization performance against pose estimation errors. Although the estimation of human 2D pose has made great progress in recent years, and the estimation results have been very accurate in simple scenes, we still want to explore the performance of our method under pose estimation results with errors. To this end, we add a series of Gaussian noises to each joint of the pose estimation results, which is only implemented on the testing dataset but not on the training data for the model training. The results are shown in Table 6, where 'Noise' represents the standard deviation (in pixels) of different Gaussian distribution, and the expectation is set to 0 for all noises. The results show that, on NTU-SYN, its performance is almost unaffected. On CMU-SYN, although its performance degrades as the increase of the noise, the proportion of synchronization error no more than 1 frame (Acc@1) is still approximately 60%. Note that, the added noises on pose estimation results, i.e., 9 pixels, are relatively large errors for the existing pose estimation methods.

Cross-dataset training and testing analysis. Finally, we show the synchronization performance of the proposed method across two datasets for training and testing. The results are shown in Table 7, where 'Dataset' indicates the dataset for testing, and 'Model' indicates which dataset the model was trained from. We can see that the model trained on the CMU-SYN and tested on NTU-SYN achieve the synchronization performance of 6.16 frames, outperforming all comparison methods as shown in Table 1. Specifically,

Table 6: Synchronization results under the various degrees of noises on the human pose.

Dataset	Noise	Frm. err.	Acc@1	Rel. err.	rAcc@10
NTU-SYN	0	3.39	62.29	0.22	54.83
	3	3.40	62.35	0.22	54.89
	6	3.40	62.85	0.22	54.95
	9	3.42	62.63	0.22	54.32
CMU-SYN	0	2.27	69.27	0.14	62.43
	3	2.56	64.06	0.16	60.85
	6	3.54	59.90	0.22	55.56
	9	5.05	58.85	0.32	58.20

the results in Table 1 are trained on the original dataset, where only *TCC* and *SynNet* outperform our method. We retain the TCC model under the cross-dataset setting and obtain the synchronization performance in Table 7, which is much worse than ours. In addition, since the model structure of *SynNet* is depended on the length of the input videos, which are inconsistent on the two datasets, so this experiment cannot be implemented. The model trained on NTU-SYN and tested on CMU-SYN can still achieve promising results, which outperforms all other comparison methods except slightly lower than *SSM*. In general, we find this cross-dataset results of our proposed method is acceptable, because the two datasets are quite different in the number of samples, scenes, and perspectives.

Table 7: Synchronization performance of cross-dataset training and testing on NTU-SYN and CMU-SYN.

Dataset	Model	Frm. err.	Acc@1	Rel. err.	rAcc@10
NTU-SYN	NTU-SYN (Ours)	3.39	62.29	0.22	54.83
	CMU-SYN (Ours)	6.16	46.89	0.40	41.27
	CMU-SYN (TCC)	19.66	12.57	1.29	10.86
CMU-SYN	CMU-SYN (Ours)	2.27	69.27	0.14	62.43
	NTU-SYN (Ours)	9.58	47.40	0.61	48.15

5 CONCLUSION

In this paper, we have studied a self-supervised framework for the automatic video synchronization, which is critical for multi-video collaborative analysis. Using the human pose as input, we developed SeSyn-Net with three self-supervised losses for training the feature embedding network. In the inference stage, with the embedded features, we get the video synchronization result by applying a simple offset estimation strategy. We have also built two datasets for evaluating the methods. Extensive experimental results verified the effectiveness, robustness and applicability of the proposed method. In addition, although the scenarios of the above experiments contain only one person, our method is easily extended to multi-person scenarios, which we show in the supplementary material. In the future, we plan to develop a more delicate algorithm that can be applied to the multi-view video synchronization in the multi-person scenarios.

Acknowledgement. This work was supported in part by the NSFC under Grants U1803264, 62072334, Tianjin Research Innovation Project for Postgraduate Students under Grant 2021YJSB174.

REFERENCES

[1] Cenek Albl, Zuzana Kukelova, Andrew Fitzgibbon, Jan Heller, Matej Smid, and Tomas Pajdla. 2017. On the two-view geometry of unsynchronized cameras. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4847–4856.

[2] Jean-Charles Bazin and Alexander Sorkine-Hornung. 2016. Actionsnapping: Motion-based video synchronization. In *European Conference on Computer Vision*. 155–169.

[3] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series.. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining Workshop*. 359–370.

[4] Akin Caliskan, Armin Mustafa, Evren Imre, and Adrian Hilton. 2020. Multi-View Consistency Loss for Improved Single-Image 3D Reconstruction of Clothed People. In *Asian Conference on Computer Vision*.

[5] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. 2020. Skeleton-based action recognition with shift graph convolutional network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 183–192.

[6] Hau Chu, Jia-Hong Lee, Yao-Chih Lee, Ching-Hsien Hsu, Jia-Da Li, and Chu-Song Chen. 2021. Part-aware measurement for robust multi-view multi-human 3d pose estimation and tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1472–1481.

[7] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 886–893.

[8] Zijian Dong, Jie Song, Xu Chen, Chen Guo, and Otmar Hilliges. 2021. Shape-aware Multi-Person Pose Estimation from Multi-View Images. In *IEEE/CVF International Conference on Computer Vision*. 11158–11168.

[9] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2019. Temporal cycle-consistency learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1801–1810.

[10] Ahmed Elhayek, Carsten Stoll, Kwang In Kim, H-P Seidel, and Christian Theobalt. 2012. Feature-based multi-video synchronization with subframe accuracy. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*. 266–275.

[11] Yiyang Gan, Ruize Han, Liqiang Yin, Wei Feng, and Song Wang. 2021. Self-supervised Multi-view Multi-Human Association and Tracking. In *ACM International Conference on Multimedia*. 282–290.

[12] Lingling Gao, Yanli Ji, Gedamu Alemu Kumie, Xing Xu, Xiaofeng Zhu, and Heng Tao Shen. 2021. View-invariant Human Action Recognition via View Transformation Network. *IEEE Transactions on Multimedia* (2021).

[13] Xiang Gao, Wei Hu, Jiaxiang Tang, Jiaying Liu, and Zongming Guo. 2019. Optimized skeleton-based action recognition via sparsified graph regression. In *ACM International Conference on Multimedia*. 601–610.

[14] Xuehao Gao, Yang Yang, Yimeng Zhang, Maosen Li, Jin-Gang Yu, and Shaoyi Du. 2021. Efficient Spatio-Temporal Contrastive Learning for Skeleton-Based 3D Action Recognition. *IEEE Transactions on Multimedia* (2021).

[15] Ruize Han, Wei Feng, Yujun Zhang, Jiewen Zhao, and Song Wang. 2021. Multiple human association and tracking from egocentric and complementary top views. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[16] Ruize Han, Yiyang Gan, Jiacheng Li, Feifan Wang, Wei Feng, and Song Wang. 2022. Connecting the Complementary-View Videos: Joint Camera Identification and Subject Association. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2416–2425.

[17] Ruize Han, Yun Wang, Haomin Yan, Wei Feng, and Song Wang. 2022. Multi-View Multi-Human Association With Deep Assignment Network. *IEEE Transactions on Image Processing* 31 (2022), 1830–1840.

[18] Ruize Han, Jiewen Zhao, Wei Feng, Yiyang Gan, Liang Wan, and Song Wang. 2020. Complementary-view co-interest person detection. In *ACM International Conference on Multimedia*. 2746–2754.

[19] Sanjay Haresh, Sateesh Kumar, Huseyin Coskun, Shahram N Syed, Andrey Konin, Zeeshan Zia, and Quoc-Huy Tran. 2021. Learning by Aligning Videos in Time. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5548–5558.

[20] Bo-Song Huang, Day-Fann Shen, Guo-Shiang Lin, and Sin-Kuo Daniel Chai. 2019. Multi-Camera Video Synchronization Based on Feature Point Matching and Refinement. In *IEEE/ACIS International Conference on Computer and Information Science*. 136–139.

[21] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, et al. 2017. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 1 (2017), 190–204.

[22] Imran N Junejo, Emilie Dexter, Ivan Laptev, and Patrick Perez. 2010. View-independent action recognition from temporal self-similarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2010), 172–185.

[23] Yiguang Liu, Menglong Yang, and Zhisheng You. 2012. Video synchronization based on events alignment. *Pattern Recognition Letters* 33, 10 (2012), 1338–1348.

[24] David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (2004), 91–110.

[25] Cheng Lu and Mrinal Mandal. 2012. A robust technique for motion-based video sequences temporal alignment. *IEEE Transactions on Multimedia* 15, 1 (2012), 70–82.

[26] Zehai Niu, Ke Lu, Jian Xue, Haifeng Ma, and Runchen Wei. 2021. Multi-view 3D Smooth Human Pose Estimation based on Heatmap Filtering and Spatio-temporal Information. In *ACM International Conference on Multimedia*. 442–450.

[27] Dmitry Pundik and Yael Moses. 2010. Video synchronization using temporal signals from epipolar lines. In *European Conference on Computer Vision*. 15–28.

[28] Senthil Purushwalkam, Tian Ye, Saurabh Gupta, and Abhinav Gupta. 2020. Aligning videos in space and time. In *European Conference on Computer Vision*. 262–278.

[29] Cen Rao, Alexei Gritai, Mubarak Shah, and Tanveer Syeda-Mahmood. 2003. View-invariant alignment and matching of video sequences. In *IEEE International Conference on Computer Vision*. 939–939.

[30] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. 2016. NTU RGB+D: A large scale dataset for 3d human activity analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1010–1019.

[31] Prarthana Shrestha, Hans Weda, Mauro Barbieri, and Dragan Sekulovski. 2006. Synchronization of multiple video recordings based on still camera flashes. In *ACM International Conference on Multimedia*. 137–140.

[32] Matej Šmıd and Jirı Matas. 2017. Rolling shutter camera synchronization with sub-millisecond accuracy. In *International Conference on Computer Vision Theory and Applications*. 8.

[33] Jennifer J Sun, Jiaping Zhao, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, and Ting Liu. 2020. View-invariant probabilistic embedding for human pose. In *European Conference on Computer Vision*. 53–70.

[34] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5693–5703.

[35] Siqi Sun, Kosuke Takahashi, Dan Mikami, Mariko Isogawa, and Yoshinori Kusachi. 2020. Multi-view video synchronization using motion rhythms of human joints. *ITE Transactions on Media Technology and Applications* 8, 2 (2020), 100–110.

[36] Zheng Tang, Renshu Gu, and Jenq-Neng Hwang. 2018. Joint multi-view people tracking and pose estimation for 3D scene reconstruction. In *IEEE International Conference on Multimedia and Expo*.

[37] Hoang Nhat Tran, Hong Quan Nguyen, Huong Giang Doan, Thanh Hai Trana, and Hai Vu. 2021. Pairwise-Covariance Multi-view Discriminant Analysis for Robust Cross-view Human Action Recognition. *IEEE Access* PP, 99 (2021).

[38] Ankit Tripathi, Benu Changmai, Shrukul Habib, Nagaratna B Chittaragi, and Shashidhar G Koolagudi. 2017. Normalized videosnapping: A non-linear video synchronization approach. In *International Conference on Contemporary Computing*.

[39] Tinne Tuytelaars and Luc Van Gool. 2004. Synchronizing video sequences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[40] Oliver Wang, Christopher Schroers, Henning Zimmer, Markus Gross, and Alexander Sorkine-Hornung. 2014. Videosnapping: Interactive synchronization of multiple videos. *ACM Transactions on Graphics* 33, 4 (2014), 1–10.

[41] Xue Wang and Qing Wang. 2016. Video synchronization with trajectory pulse. In *Chinese Conference on Intelligent Visual Surveillance*. 12–19.

[42] Patrick Wieschollek, Ido Freeman, and Hendrik PA Lensch. 2017. Learning robust video synchronization without annotations. In *IEEE International Conference on Machine Learning and Applications*. 92–100.

[43] Xinyi Wu, Zhenyao Wu, Yujun Zhang, Lili Ju, and Song Wang. 2019. Multi-Video Temporal Synchronization by Matching Pose Features of Shared Moving Subjects. In *IEEE/CVF International Conference on Computer Vision Workshops*.

[44] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI Conference on Artificial Intelligence*.

[45] Fanfan Ye, Shiliang Pu, Qiaoyong Zhong, Chao Li, Di Xie, and Huiming Tang. 2020. Dynamic gcn: Context-enriched topology learning for skeleton-based action recognition. In *ACM International Conference on Multimedia*. 55–63.

[46] Qiang Zhang, Lin Yao, Yajun Li, and Jungong Han. 2019. Video Synchronization Based on Projective-Invariant Descriptor. *Neural Processing Letters* 49, 3 (2019), 1093–1110.

[47] Jiewen Zhao, Ruize Han, Yiyang Gan, Liang Wan, Wei Feng, and Song Wang. 2020. Human identification and interaction detection in cross-view multi-person videos with wearable cameras. In *ACM International Conference on Multimedia*. 2608–2616.

[48] Feng Zhou and Fernando De la Torre. 2015. Generalized canonical time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 2 (2015), 279–294.

[49] Luca Zini, Andrea Cavallaro, and Francesca Odone. 2013. Action-based multi-camera synchronization. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 3, 2 (2013), 165–174.

A APPLICABILITY OF THE PROPOSED METHOD IN MORE PRACTICAL SCENARIOS

In this section, we show that the proposed method is not limited to the scenes containing one person using the static cameras for recording. In the experiments of main body, we use the datasets collected from CMU Panoptic Studio [21] and NTU RGB+D [30] datasets (for one person) since they include sufficient video pairs and accurate synchronization calibration, for training and evaluation.

A.1 Datasets

To illustrate the generalization of the proposed method for more practical scenes, we collected a new dataset based on the dataset used in CVID [47]. Specifically, the original dataset contains the videos for the multi-human interaction scenes (5-10 persons) captured by multiple GoPro cameras in different views. The videos from different views are manually synchronized. Based on the original videos, we select 66 pairs of synchronized videos with the length of 120 frames for each. As in this work, we set 5 types of offsets between every two videos in the range of [-30,30] frames. Finally, we get 66*5=330 video pairs, denoted as CVID-SYN dataset, for testing and evaluation.

A.2 Experimental Results

During the testing stage, we directly apply the proposed model trained on CMU-SYN/NTU-SYN datasets to the CVID-SYN dataset. Note that, CVID-SYN dataset contains multiple persons in each video. We select five shared persons, on each of them we apply the proposed method in this work to estimate the offset of the two videos. We finally use a simple strategy, i.e., the mean of the offsets from the selected five persons, to get the final offset.

The evaluation results are shown in the following table, where 'Model' indicates which dataset the model was trained from. We can see that, although we use the pre-trained model directly testing on the moving camera dataset CVID-SYN, and with a very simple multi-person fusion strategy, the results are very promising (both less than 10 frames). We also find that the results of the model trained on NTU-SYN are better than CMU-SYN. Maybe it's because the training set is much larger.

Table 8: Synchronization results of multi-human scenarios.

Model	Frm. err.	Acc@1	Rel. err.	rAcc@10
CMU-SYN	9.07	5.45	0.59	3.69
NTU-SYN	4.41	19.39	0.29	15.38