Meetor: A Human-centered Automatic Video Editing System for Meeting Recordings

HAIHAN DUAN, Shenzhen MSU-BIT University, Shenzhen, China, The Chinese University of Hong Kong, Shenzhen, China, and Mohamed bin Zayed University of Artificial Intelligence, Masdar City, United Arab Emirates
JUNHUA LIAO, Sichuan University, Chengdu, China
LEHAO LIN, The Chinese University of Hong Kong, Shenzhen, China
ABDULMOTALEB EL SADDIK, Mohamed bin Zayed University of Artificial Intelligence, Masdar City, United Arab Emirates and University of Ottawa, Ottawa, Canada
WEI CAI, The Chinese University of Hong Kong, Shenzhen, China

Widely adopted digital cameras and smartphones have generated a large number of videos, which have brought a tremendous workload to video editors. Recently, a variety of automatic/semi-automatic video editing methods have been proposed to tackle these issues in some specific areas. However, for the production of meeting recordings, the existing studies highly depend on extra equipment in the conference venues, such as the infrared camera or special microphone, which are not practical. In this article, we design and implement Meetor, a human-centered automatic video editing system for meeting recordings. The Meetor mainly contains three parts: an audio-based video synchronization algorithm, human-centered video content flaw detection algorithms, and an automatic video editing algorithm. Two main experiments are conducted from both objective and subjective aspects to evaluate the performance of the Meetor. The experimental results on a testbed illustrate that the proposed algorithms could achieve state-of-the-art (SOTA) performance in video content flaw detection. However, the conducted user study demonstrates that Meetor could generate meeting recordings with a satisfactory quality compared with professional video editors. Moreover, we also present a practical application of the Meetor in a university campus prototype, in which the Meetor is applied in the automatic editing of lecture recordings. All in all, the proposed Meetor can be utilized in practical applications to release the workload of professional video editors.

CCS Concepts: • **Human-centered computing** → *Visual analytics*; • **Computing methodologies** → **Visual content-based indexing and retrieval;**

This work is supported in part by Shenzhen Science and Technology Program (Grant No. JCYJ20210324124205016); in part by Guangdong-Hong Kong-Macao Joint Laboratory for Emotional Intelligence and Pervasive Computing, Artificial Intelligence Research Institute, Shenzhen MSU-BIT University; in part by Project 61902333 by National Natural Science Foundation of China.
Authors' addresses: H. Duan, Shenzhen MSU-BIT University, Shenzhen, Guangdong, China and The Chinese University of Hong Kong, Shenzhen, Guangdong, China and Mohamed bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, United Arab Emirates; e-mail: duanhaihan@smbu.edu.cn; J. Liao, Sichuan University, Chengdu, China; e-mail: liaojunhua@stu.scu.edu.cn; L. Lin and W. Cai (Corresponding author), The Chinese University of Hong Kong, Shenzhen, Guangdong, China; e-mails: lehaolin@link.cuhk.edu.cn, caiwei@cuhk.edu.cn; A. El Saddik, Mohamed bin Zayed University of Artificial Intelligence, Masdar City, Abu Dhabi, United Arab Emirates and University of Ottawa, Ottawa, Ontario, Canada; e-mail: elsaddik@uottawa.ca.

ACM 1551-6865/2024/08-ART265
https://doi.org/10.1145/3648681

Additional Key Words and Phrases: Automatic video editing, meeting recording, human-centered computing

ACM Reference Format:
Haihan Duan, Junhua Liao, Lehao Lin, Abdulmotaleb El Saddik, and Wei Cai. 2024. Meetor: A Human-centered Automatic Video Editing System for Meeting Recordings. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 9, Article 265 (August 2024), 23 pages. https://doi.org/10.1145/3648681

1 INTRODUCTION

Various meetings are held around the world every day, including academic conferences, sports press conferences, annual enterprise conferences, and so on. With the popularity of digital cameras and smartphones, the generation of videos has become increasingly convenient with lower cost, so most meetings currently adopt video to record the process of the meetings. For example, the organizer of conferences will settle multiple cameras to record the meetings from different perspectives and hire professional video editors to finish the cutting, composing, and editing of the meeting recordings.

From the perspective of quality requirements, an effective meeting recording must satisfy three criteria [33]: (1) It must capture enough visual information to allow viewers to understand what took place; (2) It must be compelling to watch; (3) It must not require substantial human effort. According to the above principle, the core requirement of meeting recording is sufficient information that viewers could utilize to infer the events, while after-effects are not important. Therefore, the editing process of meeting recording is relatively simple, but the tediously long video content and the repetitive work also bring a huge workload and boring experience for professional video editors, which is a waste of professional resources. To this end, it is imperative to create an automatic video editing system to help video editors in editing meeting recordings.

In recent years, the development of **multimedia (MM)** and **computer vision (CV)** provides promising technologies to relieve the burden of professional video editors. Some video editing–related tools/applications have gradually been known to the public in many specific areas, e.g., multiparty conversation [38], social gatherings [54], school concerts [19], instructional videos for physical demonstrations [4], dialogue-driven scenes [20], dance videos [44], social cameras (cameras that are carried or worn by people in activities) [2], narrated videos [43], home video [10], and video montage from themed text[46]. These video editing methods have achieved surprising performance in their targeted areas. However, there are also some studies aiming at the generation of meeting recordings [21, 23, 33], but these works have limited practical scenarios due to the dependency on extra equipment, such as the infrared camera or special microphone.

Therefore, we intend to propose an automatic video editing system that could facilitate the editing process of meeting recordings. According to the quality requirement mentioned above, the core challenge of achieving automatic video editing is to reasonably composite recording segments from different tracks to filter out the video content flaws that influence the understanding of the events. Our interviews with professional video editors highlighted three most common video content flaws when editing meeting recordings: **(1) blurriness:** the focus point of the camera usually changes from one speaker to another, causing blurriness in the video during the procedure; **(2) jitter:** the camera usually needs to be moved by the photographer, which might result in jitters in the video; **(3) occlusion:** the cameras are inevitably occluded by some objects or persons that pass through the cameras. Figure 1 shows the schematic diagram of the three common flaws in real shooting scenes of meeting recordings. In this article, we extend the human-centered video content flaw detection system in our prior conference publication [6] and propose Meetor, which could synchronize the videos of different tracks, provide flaw detection modules to accurately

Fig. 1. Schematic diagram of three common flaws in meeting recordings.

detect the video content flaws, and automatically composite the meeting recording segments from different tracks based on the detected results. The major contributions can be concluded as follows:

— We design a video synchronization algorithm based on the cross-correlation of audio in the recording materials, which can achieve the precision of 1ms in synchronization.
— We introduce human-centered video content flaw detection algorithms focusing on blurriness, jitter, and occlusion. The experiments illustrate the proposed algorithms can achieve **state-of-the-art (SOTA)** performance.
— We design an automatic video editing algorithm, which can automatically select qualified video segments from materials and composite a complete meeting recording.
— We integrate the audio-based video synchronization algorithm, human-centered video content flaw detection algorithms, and automatic video editing algorithm to implement an automatic video editing system for meeting recordings, named Meetor. The user study demonstrates the Meetor could effectively generate satisfied meeting recordings.
— We apply the Meetor system in our university campus metaverse prototype [5] as a practical application to automatically edit lecture recordings.

Compared with the conference version of the human-centered video content flaw detection system [6], this work mainly encompasses the extension of the following aspects:

— This version adds a video synchronization algorithm based on the cross-correlation of audio in meeting recordings, which is the preliminary of automatic video editing.
— This version proposes an automatic video editing algorithm to build a complete loop of the qualified video segment selection and composition.
— This version conducts a user study to evaluate the automatic video editing system Meetor.
— This version implements a practical application to automatically edit lecture recordings in our university campus metaverse prototype [5].

The remainder of this article is organized as follows: We review related works in Section 2 and present a system overview of the Meetor in Section 3. Then, we introduce our algorithm design in Section 4, including an audio-based video synchronization algorithm, human-centered video content flaw detection algorithms, and an automatic video editing algorithm. In Section 5, we evaluate the proposed flaw detection algorithms, and a user study is conducted to assess the performance of the Meetor system from non-expert users' aspects in Section 6. To better clarify the study, Figure 2 provides the article structure regarding algorithm design and corresponding performance evaluation, where the green arrow indicates the algorithms are evaluated by objective metrics, and the blue arrow denotes the algorithms are integrated as Meetor and evaluated by user study. Moreover, our practical application in a university campus metaverse is illustrated in Section 7. Section 8 concludes this article.

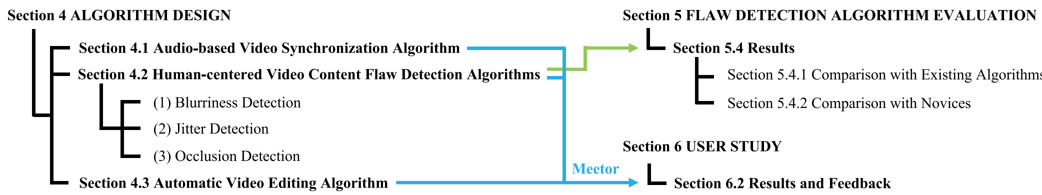

Fig. 2. Article structure regarding algorithm design and corresponding performance evaluation.

2 RELATED WORK

2.1 Video Editing Systems in Different Areas

Currently, some video editing systems are proposed to facilitate video editing in different areas. Takemae et al. [38] built a system based on the gaze of participants to extract and convey the flow of multiparty conversation. Zsombori et al. [54] introduced an evaluated approach to the automatic generation of video narratives from user-generated content gathered in a shared repository of recordings of social events. Based on the above work, Laiola et al. [19] realized a prototype software that enables community-based users to navigate through a large common content space and to generate personalized video compilations of targeted interest within a social circle. Chi et al. [4] built a semi-automatic video editing system that improves the quality of amateur instructional videos for physical tasks, e.g., to segment a single-shot demonstration recording and apply video editing effects based on user markers. Leake et al. [20] presented a system for editing video of dialogue-driven scenes to speed up the editing process by letting editors specify the set of film-editing idioms they would like to enforce and then automatically generating the corresponding edit. Tsuchida et al. [44] presented a system that automatically edits dance-performance videos taken from multiple viewpoints. Arev et al. [2] proposed an approach that takes multiple videos captured by social cameras (cameras that are carried or worn by people in activities) and produces a coherent cut video of the activity. Truong et al. [43] created an interactive video editing tool to help authors efficiently edit narrated videos. Girgensohn et al. [10] built a system that can analyze videos and allow users to easily create custom home videos from raw video shots. Wang et al. [46] presented a tool that allows novice users to generate a video montage given an input themed text and a related video repository either from online websites or personal albums. The above-mentioned studies have achieved significant performance in their targeted areas, so we can draw some experience from these pioneers. Moreover, some studies also provide video content flaw detection modules [43, 46] that would be utilized as comparison methods to evaluate our proposed algorithms.

2.2 Video Editing Systems for Meeting

Besides the video editing systems in different areas, there are some studies that focus on video editing for meetings. Lefevre et al. [21] proposed an automatic video stream selection method based on the detection of the visual state change of the microphones to select the camera that is recording the speaker. This method can be utilized in a part of meetings, but, intuitively, it cannot work if the microphones used in the meeting do not have any indicator light. Ranjan et al. [33] implemented an automated meeting capture system to capture and present videos of small group meetings. This system introduced a lot of sensors, such as infrared cameras and the Vicon motion tracking system,[1] which can be expensive to deploy in ordinary meeting venues. Liu et al. [23] set up three cameras in the lecture room (speaker-tracking camera, audience-tracking camera, and overview camera), and proposed a virtual video director based on a **finite state machine**

[1]https://www.vicon.com/

(FSM) to automatically manage the cameras. This system also depends on the pre-defined layout and principles, so it is not applicable in practical meeting recordings. Therefore, according to the above-mentioned works, the existing video editing methods are hard to deal with the practical meeting scene due to the dependency on extra equipment.

2.3 Video Flaw Detection Algorithms

In recent years, lots of researchers have contributed works about video quality assessment. But most of them focus on the objective quality loss after compression, encoding, or transmission, such as References [27, 34, 36, 51], while few pay attention to the subjective sense of viewers from a human-centered perspective, e.g., the flaws caused by the shooting of videos. Specifically, for the video flaws highlighted in this article, there are some studies that have related contributions: (1) Blurriness detection: There are two main kinds of blurriness detection methods that have been known to the public. The first kind of method focuses on defocus blur detection on images and generates defocused regions as semantic segmentation [1, 39, 50, 52], which is not suitable for the frame-level evaluation of this article. And another kind of method pays attention to the blur degree estimation of video frames [1, 3, 31], which is applied in this article with specific improvement. (2) jitter detection: Currently, there are some existing video editing systems that embed the jitter (or shake) detection algorithms to filter the video materials [43, 46], while other jitter detection algorithms mainly fall into some special areas, e.g., videos from the satellite [24, 40, 47]. But the performance of the existing jitter detection methods is not satisfactory in practice, so a new jitter detection algorithm will be proposed in this article. (3) Occlusion detection: The occlusion detection algorithms play an important role in many CV applications, e.g., video tracking [12, 17, 18, 49], optical flow estimation [15, 16, 37, 48], and pedestrian detection [26, 28, 29, 53], but the occlusion detection in these studies only serves their core task while not from a human-center perspective. The most relevant study is proposed by Liao et al. [22], which builds a large-scale database for occlusion detection and proposes a SOTA deep learning model as the benchmark. In this article, we will utilize the occlusion detection database [22] to train our neural network model and evaluate the performance.

3 SYSTEM OVERVIEW

To achieve the automatic video editing of meeting recordings, we design and implement Meetor, which has a concise video editing workflow as shown in Figure 3, containing five main steps:

(1) Import Materials. At first, the users need to import video materials that are shot from different perspectives to the Meetor. Moreover, many meetings would use a voice recorder to clearly record the presentations and conversations, which is also supported as the audio track in Meetor.

(2) Video Synchronization. Before video editing, the first step is to synchronize the videos captured from different cameras. The procedure is time-consuming if the users synchronize the videos through their hearing, and the synchronization results are usually not accurate, which highly influences the viewers' experience. In Meetor, we provide an audio-based synchronization algorithm to facilitate this procedure, which will be discussed in Section 4.1.

(3) Video Flaw Detection. After the synchronization of videos, Meetor needs to analyze the video materials for searching the video flaws from a human-centered perspective, which applies three video flaw detection modules regarding blurriness, shake, and occlusion, respectively. The video flaw detection modules will use a sliding window to extract some frames and send them to the backend for calculation. Moreover, intuitive visualization reports corresponding to each meeting recording will be generated for video editors to fast locate and check the detected results. The details of video flaw detection will be discussed in Section 4.2.

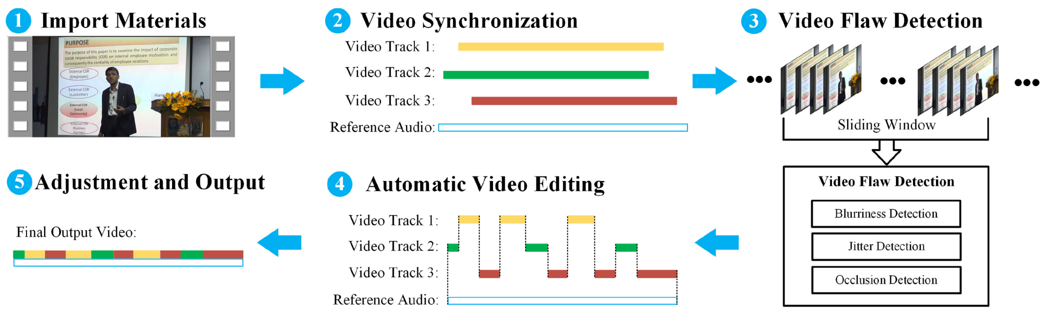

Fig. 3. The flowchart of the Meetor system.

(4) Automatic Video Compositing. For the video editing process, the aim of the user is to select the meeting segments in different video tracks to compose a complete output video. In Meetor, we propose an automatic video editing algorithm, which will be discussed in Section 4.3.

(5) Adjustment and Output. After the automatic video editing, the Meetor can generate a frame indexes script and audio file of the final output video, which can be composed as the final output video. Moreover, if the users want to adjust the editing decision, then the frame indexes script can also be converted to an **Edit Decision List (EDL)**,[2] a kind of text script that records video editing decisions. After that, the users can preview the result of the Meetor and manually adjust the editing decision as their preference in commercial video editing software, e.g., Adobe Premiere.[3]

4 ALGORITHM DESIGN

According to the automatic video editing workflow discussed in Section 3, in this section, we will present the applied algorithms of the Meetor in each step, including audio-based video synchronization algorithm, video flaw detection algorithms, and automatic video editing algorithm.

4.1 Audio-based Video Synchronization Algorithm

In practical video shooting with multiple cameras, the recording cannot start accurately at the same time, which always has time offsets between each video. Therefore, the first step is to synchronize the video tracks captured from different cameras. In Meetor, we utilize the audio of each video material to synchronize the videos.

First, the users need to select an audio from the input materials as the audio of the final output recording after automatic video editing, which is named reference audio in this article. Note that, as discussed in Section 3, the users are allowed to upload the meeting audio recorded by voice recorder, which generally has a higher quality compared with audio recorded by the camera. Then, Meetor will calculate the offsets between the reference audio and all video materials. For each video material, its audio will be extracted to calculate the cross-correlation with the reference audio, and the value of the cross-correlation function could be regarded as the similarity between two audio at a specific moment [8]. The similarity will calculate many times in a sliding manner from start to end of the reference audio, and the sampling rate is set as 1,000 Hz, which means the value of the cross-correlation will be calculated every 1 ms. Note that the **International Telecommunication Union (ITU)** recommended the threshold for detectability of audio-to-video synchronization as −125 ms to +45 ms [32], so the precision of 1 ms could satisfy the requirement. Finally, the Meetor

[2] https://xmil.biz/EDL-X/CMX3600.pdf
[3] https://www.adobe.com/products/premiere.html

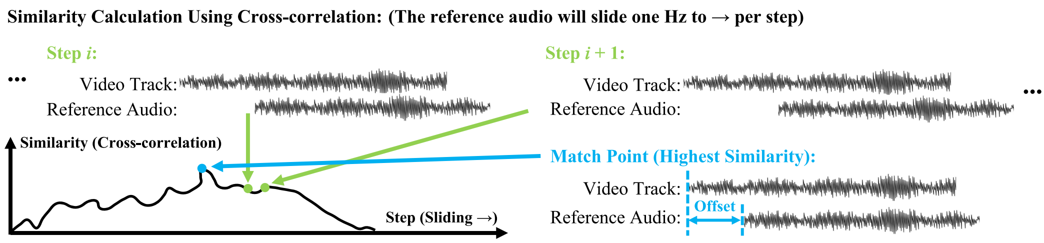

Fig. 4. Schematic diagram of audio-based video synchronization algorithm.

will select the moment with the highest similarity as the match point for each audio extracted from videos and calculate the offset between the video and reference audio based on the match point. The schematic diagram of the audio-based video synchronization algorithm is shown in Figure 4, in which we illustrate two intermediate steps of the similarity calculation (cross-correlation) between a video track and the reference audio. Moreover, the moment with the highest similarity after calculation would be selected as the match point and obtain the offset between the two videos.

In fact, the strict and brute-force synchronization method is indeed accurate, but it also would cost a lot of time due to the huge computational overload. Therefore, in practice, there are some tradeoff methods that could improve the efficiency of the proposed algorithm. On the one hand, the users can specify the offset range (e.g., the time offsets of the cameras are usually less than 5 minutes in practical shooting) so the Meetor does not need to compare the audios from start to end. On the other hand, the sampling rate can be set as 100 Hz, which could save 10× computational cost but maintain a sufficient accuracy of the audio-to-video synchronization.

4.2 Human-centered Video Content Flaw Detection Algorithms

As shown in the third step of the proposed Meetor (Figure 3), a sliding window would extract some frames and send them to the video flaw detection system to detect three common flaws in real shooting scenes (blurriness, jitter, and occlusion), where we set the size of the sliding window as 8 frames. The following subsections will discuss each flaw detection algorithm in detail. Note that we use $V = \{f_1, f_2, \ldots, f_n\}$ to denote an input video with totally n frames, and f_i represents the ith frame of the video. And other notations for specific concepts will be introduced in each subsection.

(1) Blurriness Detection. The representation of out-of-focus is blurriness in videos, and a typical feature is that there are few edges in these frames, which inspires many significant methods [1, 31]. However, the existing blurriness detection method only sets a constant value as the threshold, which cannot perform well for videos that are recorded by different shooting equipment. For example, all frames of the video with poor quality shooting equipment might be misrecognized by the method with an unsuitable constant threshold.

In the Meetor system, we modified a simple but sound blurriness detection method based on the Laplacian operator [31], as shown in Algorithm 1. To better explain the algorithm, we also illustrate a schematic diagram as shown in Figure 5, including the sample frames of clear and blur cases. We first calculate the second derivative of an image to represent the number of its edges, so the Laplacian operator is applied on each frame f_i and outputs their Laplacian map as $L = \{L_1, L_2, ..., L_n\}$. Then, we calculate the variance of each Laplacian map as $v = \{v_1, v_2, ..., v_n\}$, where v_i could represent the number of edges in frame f_i. Intuitively, if the frame f_i is a blur frame, then the v_i would get a lower value. Then, we will calculate the standard deviation of the v, represented by $SD(v)$ to evaluate the average degree of blurriness. If the standard deviation $SD(v)$ is larger than $\theta = 1,000$, then we consider the video has an unbalanced clarity and then we will search out the blurriness. The frame would be regarded as a blur frame if it satisfies the following

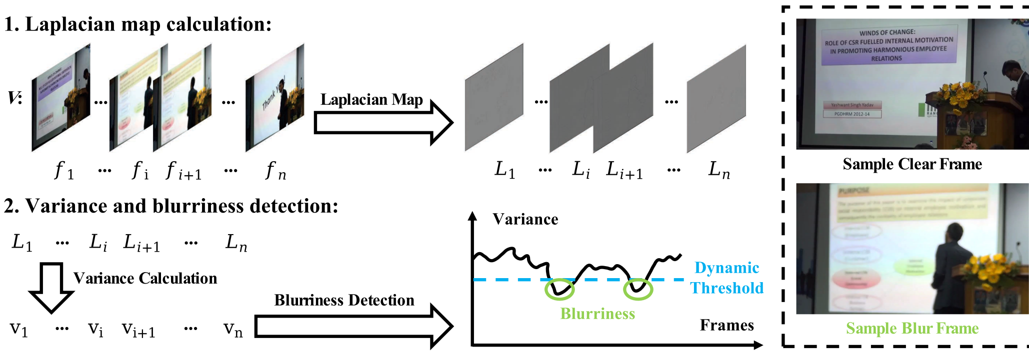

Fig. 5. Schematic diagram of blurriness detection algorithm.

equation:

$$v_i < \overline{v} - \frac{\overline{v} - v_{min}}{k}, \quad i = 1, \ldots, n, \tag{1}$$

where $\overline{v}$ denotes the mean of variance array v, v_{min} is the minimum of v, and k is a coefficient and set as 3. The motivation of this algorithm is to utilize the global degree of blurriness for judgment instead of a fixed threshold.

ALGORITHM 1: Blurriness Detection Algorithm

Input: Video $V = \{f_1, f_2, ..., f_n\}$, parameter θ, k
Output: Frames with blurriness B

Init: *Detected blurriness frame array B, Laplacian map array L, variance array of Laplacian map v*
for $i = 1$ *to* $i = n$ **do**
    L_i = Laplacian_map(f_i);
    v_i = Variance(L_i);
end
$SD(v)$ = Standard_deviation(v);
if $SD(v) > \theta$ **then**
    t = mean(v) - (mean(v) - min(v)) / k;
    for $i = 1$ *to* $i = n$ **do**
        if $v_i < t$ **then**
            B.append(f_i);
        end
    end
end
return B;

(2) Jitter Detection. In complicated shooting environment, the camera might be shaken and result in jitters in final recordings. However, the jitter is hard to be defined, since there are many normal camera moves that might confuse the detection algorithms. Our preliminary experiments show that the existing methods [43, 46] cannot effectively detect jitter, since they would falsely regard all camera movements as jitters. In this article, we design a novel jitter detection algorithm based on an intuitive observation that normal consecutive frames should not move toward different directions in a short time slot (e.g., left then right), while moving in a single direction is a normal movement. Algorithm 2 shows the pseudocode.

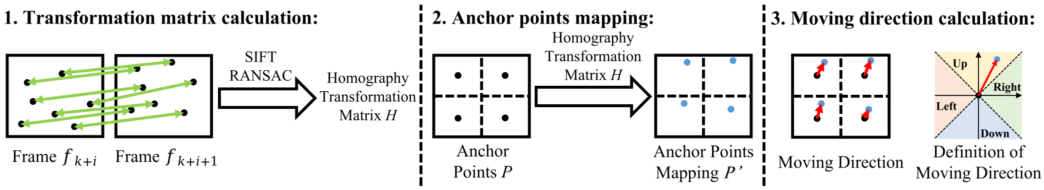

Fig. 6. Schematic diagram of moving direction calculation.

ALGORITHM 2: Jitter Detection Algorithm

Input: Frames $f = \{f_k, f_{k+1}, ..., f_{k+7}\}$
Output: Whether the input frames f are jitters

Init: *Direction array* $D[7]$, *anchor points* $P[4]$,$P'[4]$
Resize all frames of f to (480×270);
for $i = 0$ *to* $i = 7$ **do**
    Calculate homography transformation matrix H using SIFT features of f_{k+i} and f_{k+i+1};
    Calculate the mapping P' of P using H;
    if *More than 1/4 Manhattan_distance*$(P, P') > dis$ **then**
        Calculate the moving direction as D_{k+i};
    end
    else
        $D_{k+i} = 0$;
    end
end
for $i = 0$ *to* $i = 7$ **do**
    if $D_{k+i} \neq D_{k+i+1}$ *and* $D_{k+i}, D_{k+i+1} \neq 0$ **then**
        return True;
    end
end
return False;

As shown in the flowchart of Figure 3, the sliding window would extract frames and send them for flaw detection. Our algorithm first calculates the moving direction of extracted frames. As shown in Figure 6, we calculate the homography transformation matrix H of consecutive frames in the sliding window using SIFT features [25] and RANSAC regression [9], where H contains the movement information of feature points. After obtaining H, each video frame is split into 4 equal parts from the middle, and the 4 pivots of which are selected as anchor points P. The 4 anchor points are used to measure whether the camera is moving according to the principle of Algorithm 2. For example, if only 1/4 of anchor points have a shift, then it might be a person who is walking while the camera does not move. Using the matrix H, we can calculate the mapping of anchor points as P' to estimate the Manhattan distance between P and P'. If more than 1 anchor points have a shift distance larger than a predefined distance *dis* (set as 0.7), then we believe the camera is moving. We define 4 directions in axis directions, in which the definition of the moving direction is shown in Figure 6, including up, down, left, right according to the included angle between the moving direction and the axis (the example moving direction in Figure 6 is up). Then, we calculate the moving direction D_{k+i} if the 4 anchor points have the same moving direction, otherwise, we set $D_{k+i} = 0$. Then the algorithm will search for jitters from the array D. If 2 consecutive frames move toward different directions, then we believe the input frames have jitters.

Table 1. Occlusion Detection Neural Network Model

Stage	Parameters	
Conv1	$32, 3 \times 3 \times 3, (1, 1, 1)$	
	Max Pool 3D $1 \times 2 \times 2$ with Stride (1,2,2)	
Conv2	$32, 1 \times 1 \times 1, (0, 0, 0)$	$32, 1 \times 1 \times 1, (0, 0, 0)$
	$64, 1 \times 3 \times 3, (0, 1, 1)$	$64, 3 \times 1 \times 1, (1, 0, 0)$
	$64, 1 \times 1 \times 1, (0, 0, 0)$	$64, 1 \times 1 \times 1, (0, 0, 0)$
	Max Pool 3D $1 \times 2 \times 2$ with Stride (1,2,2)	
Conv3	$64, 1 \times 1 \times 1, (0, 0, 0)$	$64, 1 \times 1 \times 1, (0, 0, 0)$
	$128, 1 \times 3 \times 3, (0, 1, 1)$	$128, 3 \times 1 \times 1, (1, 0, 0)$
	$128, 1 \times 1 \times 1, (0, 0, 0)$	$128, 1 \times 1 \times 1, (0, 0, 0)$
	Max Pool 3D $1 \times 2 \times 2$ with Stride (1,2,2)	
Conv4	$128, 1 \times 1 \times 1, (0, 0, 0)$	$128, 1 \times 1 \times 1, (0, 0, 0)$
	$256, 1 \times 3 \times 3, (0, 1, 1)$	$256, 3 \times 1 \times 1, (1, 0, 0)$
	$256, 1 \times 1 \times 1, (0, 0, 0)$	$256, 1 \times 1 \times 1, (0, 0, 0)$
	Max Pool 3D $1 \times 2 \times 2$ with Stride (1,2,2)	
Conv5	$256, 1 \times 1 \times 1, (0, 0, 0)$	$256, 1 \times 1 \times 1, (0, 0, 0)$
	$512, 1 \times 3 \times 3, (0, 1, 1)$	$512, 3 \times 1 \times 1, (1, 0, 0)$
	$512, 1 \times 1 \times 1, (0, 0, 0)$	$512, 1 \times 1 \times 1, (0, 0, 0)$
	Max Pool 3D $1 \times 2 \times 2$ with Stride (1,2,2)	
Conv6	$512, 1 \times 1 \times 1, (0, 0, 0)$	$512, 1 \times 1 \times 1, (0, 0, 0)$
	$1,024, 1 \times 3 \times 3, (0, 1, 1)$	$1,024, 3 \times 1 \times 1, (1, 0, 0)$
	$1,024, 1 \times 1 \times 1, (0, 0, 0)$	$1,024, 1 \times 1 \times 1, (0, 0, 0)$
	Max Pool 3D $1 \times 2 \times 2$ with Stride (1,2,2)	
Conv7	$1,024, 1 \times 1 \times 1, (0, 0, 0)$	$1,024, 1 \times 1 \times 1, (0, 0, 0)$
	$2,048, 1 \times 3 \times 3, (0, 1, 1)$	$2,048, 3 \times 1 \times 1, (1, 0, 0)$
	$2,048, 1 \times 1 \times 1, (0, 0, 0)$	$2,048, 1 \times 1 \times 1, (0, 0, 0)$
	Global Max Pool 2D	
FC	Fully Connected $8 \times 2,048$	
	Fully Connected $8 \times 1,024$, Dropout 0.5	
	Fully Connected 8×512, Dropout 0.5	
	Fully Connected 8×2 and Softmax	

(3) Occlusion Detection. The occlusion detection is relatively complex, since the definition of occlusion does not have a consensus in different research areas. In video editing of meeting recordings, we regard the objects that appear at the improper time as occlusions, e.g., the objects that occlude the speakers. This task is highly suitable for the deep learning–based algorithm because it requires a semantic understanding of content.

In the Meetor system, we design a deep neural network model as shown in Table 1, which is a binary classification network (to classify whether the frame has occlusion). In this table, except for the special notes, all layers are convolutional layers with zero padding. For the input layer, 8 consecutive frames from the sliding window are resized as $8 \times 171 \times 128 \times 3$ as the input tensor. In each convolutional layer block, there are two branches of the network that divide the 3D convolution as a 1D+2D paradigm, and the features would be added before the max-pooling layer to combine the information. The loss function, which is demonstrated to be effective in occlusion detection [22], is applied in the training process of the model, which is defined as:

$$L_{occ} = -e^{\frac{occ\_ratio}{\lambda}} (l_j^i \log(o_j^i) + (1 - l_j^i) \log(1 - o_j^i)), \tag{2}$$

where l_j^i and o_j^i represent the label and prediction result of the jth frame in the ith video, respectively. The occ_ratio means the ratio of occlusion by the frame, which is defined as:

$$occ\_ratio = \frac{area\_of\_occlusion}{area\_of\_frame}, \tag{3}$$

where $area\_of\_occlusion$ represents the area of occlusion, and $area\_of\_frame$ denotes the original area of the frame. And λ is a hyper-parameter to control the degree of penalty.

(4) Detection Results Visualization. After the processing of the flaw detection, a Web-based report with an intuitive user interface would be generated to visualize the detection results, as shown in Figure 7, which is named FLAD in our prior conference publication [6]. The report provides three timelines with different colors to display detected flaws corresponding to blurriness, jitter, and occlusion, respectively. And each timeline shows the average prediction confidence of detected flaws measured in seconds, e.g., the deep learning–based occlusion detection algorithm will predict whether a frame has occlusion with a confidence value ranging from 0 to 1. For example, in the screenshot of Figure 7, the dialogue of the front two persons occluded the presentation of the major speaker, so this series of frames is detected as occlusion by the Meetor system, and there is a ridge in the blue timeline. At the same time, since the camera is focusing on the major

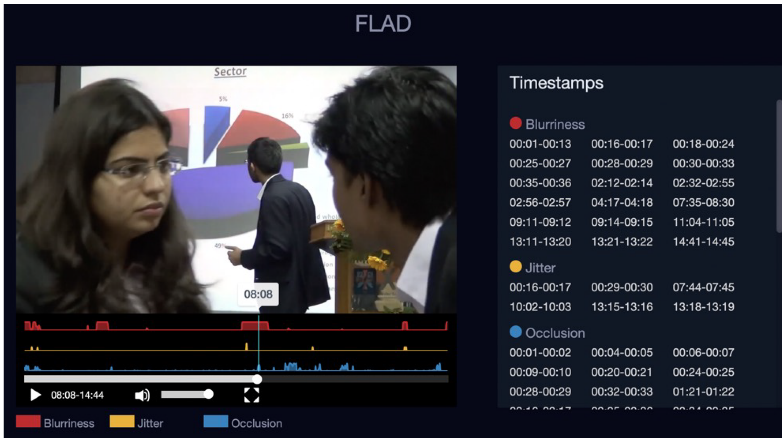

Fig. 7. A screenshot of the FLAD visualization report.

speaker, the front two persons show a significant blurriness, and the Meetor system can also accurately locate the blurriness as shown in the red timeline. On the other side, there is a panel named "Timestamps" that allow video editors to jump the video and check the specific flaws, and this panel can also illustrate the precise duration of flaws. With this visualization module, the Meetor system could efficiently help users in video editing.

4.3 Automatic Video Editing Algorithm

After the video flaw detection process, the last step is to design an algorithm to automatically edit meeting recordings. Specifically, the video editing method can be regarded as a selection algorithm for the input video tracks' segments. According to the quality requirement mentioned in Section 1, we conclude the following principles for the design of the automatic video editing algorithm: (1) The final output videos should retain the meetings' procedures as completely as possible; (2) The final output videos should have few video flaws unless the flaws cannot be avoided (e.g., all tracks have flaws at the same time); (3) The final output videos should have natural track switches rather than frequent and trivial track switches, which would influence the viewing experience of the audience.

Therefore, we design an automatic video editing algorithm according to the above-mentioned principles. Figure 8 shows an example illustration of the proposed algorithm, which is an example expansion of the second step of the flowchart (Figure 3). In this example, we assume that there are three video tracks annotated by yellow, green, and red, and a reference audio denoted by blue. Note that, for most cases, the reference audio from the voice recorder will cover the whole procedure of the meeting (start before and end after all video recording tracks) because the storage cost of audio recordings is significantly lower than video recordings.

The automatic video editing algorithm is constructed by three main parts: audio-based video synchronization, human-centered video content flaw detection, and automatic video segment selection. The details of video synchronization and flaw detection are discussed in Section 4.1 and Section 4.2, respectively, so here we only raise some notable points in algorithm implementation that are not mentioned in the previous subsections: (1) After the video synchronization, we can calculate the start time and end time of the input videos, denoted by T_start and T_end. Then, we can use the T_start and T_end to extract the final audio. Also, the total frame number of the final output video can be calculated using the FPS of the input video recordings. Thus, the input

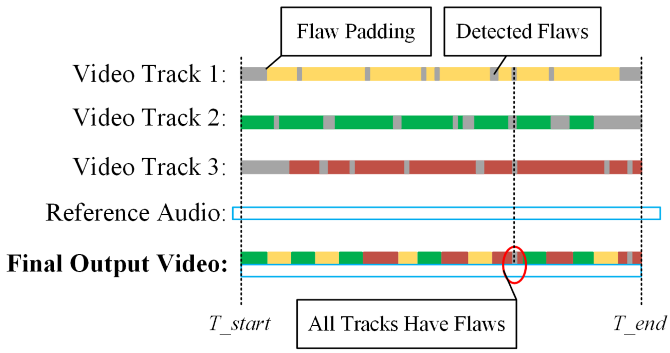

Fig. 8. An example illustration of the automatic video editing algorithm.

videos can be transformed as an array of $N \times frame\_count$, named $video\_frame\_list$, where N is the number of video tracks, and each element of this array is a frame. (2) According to Section 4.2, the flaw detection process will focus on three kinds of flaws, denoted by $flaw\_results\_list$ with $N \times frame\_count \times 3$. If a specific frame has any flaws, then the frame would be regarded as a flawed frame. Thus, the $flaw\_results\_list$ can be converted as $available\_frame\_list$ with $N \times frame\_count$. Note that the 2 seconds before and after a flawed frame would be regarded as flaw frames to avoid omission of the flaws. (3) The example in Figure 8 shows a sample result after video synchronization and flaw detection, where the time duration with detected flaws is annotated by gray boxes. In algorithm implementation, we also regard the blank head and tail of each video track between T_start and T_end as flaws, named Flaw Padding in this article, which could unify the video segment selection procedure. (4) After the previous steps, the synchronized video tracks would be converted as a numerical array $available\_frame\_list$ (0: the frame has no flaw; 1: the frame has flaws; a large number: null frame after Flaw Padding), which is the input of the automatic video segment selection.

As shown in Algorithm 3, the proposed automatic video segment selection algorithm is a greedy algorithm due to the two advantages: (1) The video segment selection algorithm does not have a globally optimal, so the greedy algorithm is the fastest method with $O(n)$ time complexity; (2) Using the greedy algorithm, the implementation of the Meetor system could complete the stream processing based on a queue manner, which could decrease the consumption of memory when processing tediously long video recordings.

First, the algorithm initializes a variable to record the track indexes of each frame, named $final\_frame\_indexes$. Then the algorithm goes through the time axis from T_start to T_end to select available video segments. The users need to input a specific range of the video segments' length, denoted as shortest duration SD and longest duration LD (8 and 15 seconds, respectively, in our experiments). Thus, for each while loop, a random temporal duration of a video segment will be generated. Then, the algorithm will check all video tracks and search out a list of available tracks without video flaws in the temporal duration, named $available\_track\_list$. After that, the following procedure is to select an available track for the temporal duration, which contains three conditions: (1) If there is no available track, which means all video tracks have content flaws in the temporal duration (refer to "All Tracks Have Flaws" in Figure 8), then the algorithm will select a track that has the shortest duration of flaws; (2) If there is only one track in the $available\_track\_list$, then the track will be selected in the final output video; (3) If there are many available tracks, then the algorithm will preferentially select the track that is different from the last video segment. Finally, the $final\_audio$ and $final\_frame\_indexes$ would be returned

ALGORITHM 3: Automatic Video Editing Algorithm

```
Input: Video Track List V[N], Reference Audio A, Frame Per Second FPS, Shortest Duration SD,
       Longest Duration LD
Output: Final Frame Indexes Script final_frame_indexes, Final Audio File final_audio
# Step 1: Audio-based Video Synchronization
offset_list[N] = Synchronization_Algorithm(V, A);
T_start, T_end = Calculate_Final_Video_Duration(offset_list, V);
final_audio = A[T_start, T_end];
frame_count = length(final_audio) × FPS;
video_frame_list[N][frame_count] = Video_To_Frame(frame_count, offset_list, V);
# Step 2: Human-centered Video Content Flaw Detection
for i = 0 to i = N do
    flaw_results_list[i][frame_count][3] = Flaw_Detection(video_frame_list[i]);
end
available_frame_list[N][frame_count] = Convert_To_Available_Frames(flaw_results_list);
available_frame_list = Head_Tail_Flaw_Padding(available_frame_list, V);
# Step 3: Automatic Video Segment Selection
final_frame_indexes = int[frame_count];
current_time = T_start, current_track = None;
while current_time > T_end do
    temp_duration = Random_Int(SD, LD);
    if current_time + temp_duration > T_end then
        temp_duration = T_end - current_time;
    end
    available_track_list = Check_Availability(current_time, temp_duration, available_frame_list);
    if len(available_track_list) == 0 then
        current_track = Find_Shortest_Flaws_Track(current_time, temp_duration,
          available_frame_list);
    end
    else if len(available_track_list) == 1 then
        current_track = available_track_list[0];
    end
    else
        available_track_list.remove(current_track);
        current_track = Randomly_Select_A_Track(available_track_list);
    end
    final_frame_indexes[current_time: current_time + temp_duration] = current_track;
    current_time += temp_duration;
end
return final_frame_indexes, final_audio;
```

to compose and render the final output video. As a result, the Meetor can also output EDL for manual adjustment by the users.

5 FLAW DETECTION ALGORITHM EVALUATION

In this section, we conduct experiments to validate the performance of the proposed human-centered video content flaw detection algorithms. This section will discuss the construction of the experimental testbed, the implementation details of the neural network training, and the applied

Fig. 9. Sample frames of the testbed.

Table 2. Details of Video Content Flaw Testbed

Video No.	Scenario	Length	Blurriness	Jitter	Occlusion
Video 1	Academic Report	14:44	4 (1.0s ± 0.0s)	4 (2.5s ± 1.7s)	12 (5.3s ± 7.0s)
Video 2	Tutorial	08:54	2 (1.0s ± 0.0s)	6 (3.5s ± 3.5s)	4 (3.5s ± 2.1s)
Video 3	Project Meeting	09:01	0 (0.0s ± 0.0s)	1 (6.0s ± 0.0s)	7 (48.6s ± 90.7s)
Video 4	Award Ceremony	14:44	1 (2.0s ± 0.0s)	8 (12.5s ± 7.9s)	0 (0.0s ± 0.0s)
Video 5	Annual Meeting	13:36	0 (0.0s ± 0.0s)	2 (26.5s ± 24.5s)	3 (50.7s ± 44.0s)
Video 6	Academic Report	05:33	0 (0.0s ± 0.0s)	1 (6.0s ± 0.0s)	2 (6.0s ± 2.0s)
Video 7	Academic Conference	13:37	0 (0.0s ± 0.0s)	7 (1.7s ± 1.2s)	8 (3.4s ± 3.4s)
Video 8	Press Conference	05:13	0 (0.0s ± 0.0s)	6 (9.3s ± 6.2s)	0 (0.0s ± 0.0s)

human-centered evaluation metrics. More importantly, the experimental results are illustrated in the last subsection to demonstrate the efficiency of the proposed algorithms.

5.1 Testbed

For the evaluation of the proposed algorithms, we totally collected 8 meeting recordings with different scenarios and quality (resolution of $1,280 \times 720$) from YouTube. Some sample frames of the testbed are shown in Figure 9. Specifically, in video 3, video 7, and video 8, there are some obvious occlusions that could be easily recognized, where the speakers are occluded by the audience or passers-by. Then, we invited a professional video editor with eight years of experience to carefully check the video materials. All the content flaws (blurriness, jitter, and occlusion) of the collected videos were annotated by the professional video editor second-by- second, which could be regarded as the baseline for further experiments. Detailed information about the dataset can be found in Table 2, which presents the counts and length statistics (mean value ± standard deviation) of blurriness, jitter, and occlusion annotated by the professional video editor. For comparison, we also recruited seven novices who are not familiar with video editing to point out the video content flaws second-by-second where they feel discomfort. This testbed will be open-sourced at *Kaggle*[4] to support the related studies.

5.2 Implementation Details

The Meetor is deployed on a server with six processors (Intel Core i7-7700 @ 3.60 GHz) and 16 GB RAM, which is equipped with an NVIDIA GTX 1080Ti GPU (11 GB). The deep neural network model of occlusion detection is implemented using PyTorch [30]. A large dataset for occlusion detection [22] is utilized in the training of the neural network model, which contains 1,000 video segments where the appeared occlusions are annotated frame-by-frame. In the training process, the **Stochastic Gradient Descent (SGD)** is applied with 0.9 momentum and 0.0005 weight decay. The learning rate is initialized as 0.0001 with a learning rate decay of 0.5 every 10 epochs. And the degree of penalty λ in the loss function is set as 10. The whole training process contains 50 epochs.

5.3 Human-centered Evaluation Metrics

For the evaluation, we first evaluate how many annotated flaws are detected, which is the true positive rate commonly used in statistics, denoted by TPR. The classic action detection or recognition tasks apply frame-level evaluation, but, from a human-centered perspective, the event-level evaluation is more reasonable, since the human sense cannot be accurate to frame-level detection. Therefore, we consider that the flaw is detected if there is an overlap between the annotation from the professional video editor and the experimental detection result from algorithms or novices.

[4] https://www.kaggle.com/datasets/seaxiaod/flad-video-flaw-detection

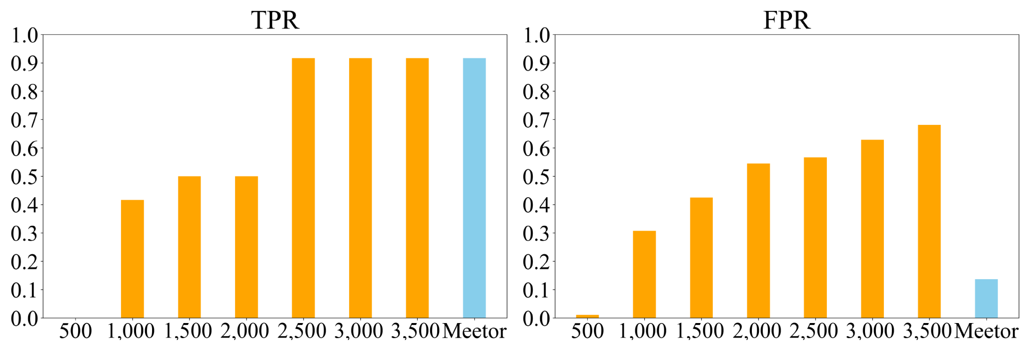

Fig. 10. Comparison of blurriness detection algorithms.

However, the higher *TPR* usually brings higher false positive results, so we use the duration of misrecognized flaws detected by each algorithm and novices to divide the total length of the video as the false positive ratio, represented as *FPR*. The *FPR* could be regarded as the additional workload that the users of the Meetor system need to check whether there is a flaw.

5.4 Results

5.4.1 Comparison with Existing Algorithms. In this section, we compare the proposed video content flaw detection algorithm with state-of-the-art methods.

(1) Blurriness Detection. Existing blurriness detection methods [1, 31] would set a fixed threshold, while the proposed algorithm can dynamically adjust the threshold using global information, so the evaluation mainly focuses on its effectiveness. We implemented the most applied blurriness detection method based on the Laplacian operator as the baseline [31], in which we changed its predefined threshold from 500 to 3,500 with an interval of 500. Note that more clear images will show a higher variance of the Laplacian map, so the higher threshold is more sensitive to blurriness. The comparative results are illustrated in Figure 10. As shown in this figure, if the threshold is set as 500, then the baseline method cannot detect any blurriness. With the growth of the threshold, the *TPR* shows a significant increase, while the *FPR* also grows fast accordingly. Specifically, when the threshold is set as 2,500, the baseline method and the Meetor system share the same *TPR*, which is larger than 0.9, but the *FPR* of the Meetor system is obviously lower than the baseline method. The experimental results demonstrate that the proposed blurriness detection algorithm can achieve better performance in detection and effectively maintain a low *FPR*.

(2) Jitter Detection. To evaluate the performance of the proposed jitter detection algorithm, we introduce two SOTA jitter detection methods for comparison, including QuickCut [43] and Write-A-Video [46], which mainly evaluate the acceleration of video content to determine whether the camera is shaking. The comparison results of different jitter detection algorithms are shown in Figure 11. We can find that although the two comparative methods may detect more jitters in some specific videos, the proposed Meetor system can reach a higher average *TPR*. It is worth noting that, for video 3, there is a jitter annotated by the professional video editor, but none of the three algorithms could recognize it. However, the Meetor system also has better control over the *FPR*. Therefore, the proposed jitter detection method could achieve SOTA performance.

(3) Occlusion Detection. To evaluate the occlusion detection model, we apply frame-level binary classification accuracy, **receiver operating characteristic (ROC)** curve with **area under the curve (AUC)**, and **frames per second (FPS)** as evaluation metrics. The experiments are conducted in the occlusion detection dataset built by Liao et al. [22] with the same settings as the benchmark. Since only minor studies focus on video occlusion detection, regardless of four SOTA

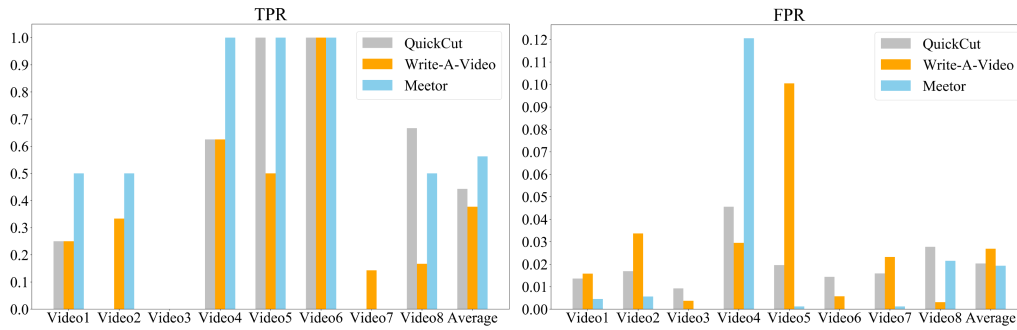

Fig. 11. Comparison of jitter detection algorithms.

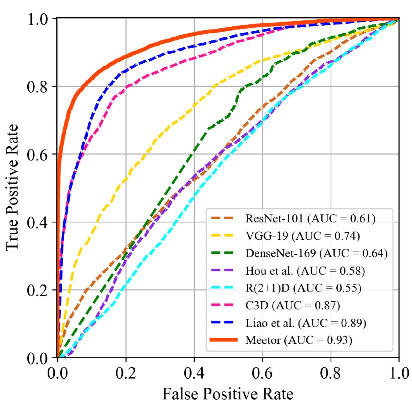

Fig. 12. Receiver operating characteristic curves of occlusion detection.

Table 3. Experimental Results of Occlusion Detection

Method	Parameters	Accuracy	FPS
ResNet-101 [11]	42.5M	0.6106	83
VGG-19 [35]	139.59M	0.6885	70
DenseNet-169 [14]	12.49M	0.6556	95
Hou et al. [13]	23.51M	0.4266	60
R(2+1)D [42]	33.18M	0.5910	99
C3D [41]	107.36M	0.7839	109
Liao et al. [22]	59.64M	0.8270	106
Meetor	50.17M	**0.8557**	**126**

occlusion detection methods (Liao et al. [22], Hou et al. [13], R(2+1)D [42], C3D [41]), three most representative deep neural network models (ResNet-101 [11], VGG-19 [35], DenseNet-169 [14]) are also included as comparative methods.

The numbers of parameters and classification accuracy of different methods are shown in Table 3. As shown in this table, the model of the Meetor system could obtain the highest classification accuracy and FPS on this dataset, while the number of parameters is only 50.17M which is less than the model of the SOTA method (Liao et al. [22]). Moreover, the ROC curves with AUC values are illustrated in Figure 12. We can find that the model of the Meetor system has better AUC values (0.93) compared with other models. The experimental results illustrate the proposed Meetor system could achieve the SOTA performance in video occlusion detection. Figure 13 shows some examples of the occlusion detection. For the false positive cases, the video editor and novices would not regard the audience's head as occlusion, since they exist in the whole video. However, for the deep neural network model, a fitting of the occlusion detection dataset, the audience exactly occluded the speaker, so the occlusion detection model actually did not make obvious mistakes. To this end, utilizing global information to improve human-centered flaw detection is worthy of further study.

5.4.2 Comparison with Novices. Compared with the existing methods, the Meetor system could achieve SOTA in the detection of the three video content flaws, and then we evaluate whether the Meetor could achieve higher detection ability compared with novices in the proposed testbed. The detailed experimental results are illustrated in Table 4, where the notation "-" denotes the videos

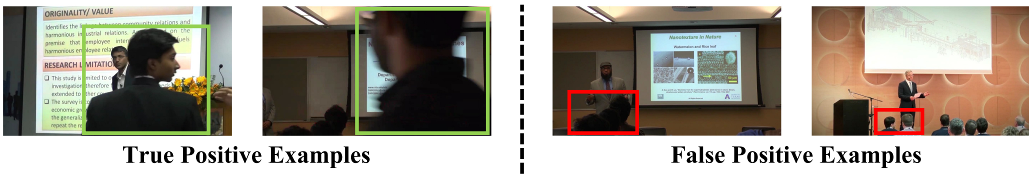

Fig. 13. Examples of occlusion detection.

Table 4. Results of Flaw Detection Compared with Novices

Video No.	Blurriness				Jitter				Occlusion			
	TPR		*FPR*		*TPR*		*FPR*		*TPR*		*FPR*	
	Novices	Meetor	Novices	Meetor	Novices	Meetor	Novices	Meetor	Novices	Meetor	Novices	Meetor
Video 1	0.2143	0.7500	0.0008	0.1379	0.2500	0.5000	0.0034	0.0045	0.5714	0.5833	0.0111	0.0701
Video 2	0.1429	1.0000	0.0027	0.4682	0.2381	0.5000	0.0067	0.0056	0.3571	1.0000	0.0088	0.4270
Video 3	–	–	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4694	1.0000	0.0783	0.1070
Video 4	0.7143	1.0000	0.0034	0.0590	0.4286	1.0000	0.0352	0.1206	–	–	0.0000	0.1314
Video 5	–	–	0.0019	0.4301	0.1429	1.0000	0.0004	0.0012	0.5714	0.6667	0.0033	0.1801
Video 6	–	–	0.0000	0.0000	0.8571	1.0000	0.0016	0.0000	0.8571	1.0000	0.0053	0.9626
Video 7	–	–	0.0005	0.0000	0.0816	0.0000	0.0014	0.0012	0.6429	1.0000	0.0051	0.0623
Video 8	–	–	0.0009	0.0000	0.2857	0.5000	0.0409	0.0215	–	–	0.0000	0.0585
Average	0.3572	**0.9167**	**0.0013**	0.1369	0.2855	**0.5625**	**0.0112**	0.0193	0.5782	**0.8750**	**0.0140**	0.2499

that are not annotated with blurriness or occlusion by the professional video editor. Note that, for novices, the average values of their experimental results are utilized in comparison.

(1) Blurriness detection. The Meetor has a significantly higher average *TPR* compared with novices, which is higher than 0.9. On the contrary, the novices have better control over the *FPR*, especially in video 2 and video 5.

(2) Jitter detection. As shown in Table 4, the Meetor could achieve a *TPR* of 0.5625, while the novices only have 0.2855, which is about half of the Meetor system. However, although the novices have better control over *FPR*, the *FPR* of the Meetor system is very close to the novices. In fact, the *TPR* of 0.5625 also shows there are still possibilities for improvement.

(3) Occlusion detection. The Meetor also shows a higher *TPR* compared with novices, which is close to 0.9, but the *FPR* of Meetor is also higher than novices. In fact, the criteria or user acceptance are highly different for different people, but the experimental results could illustrate that, from the criteria of the professional editor, Meetor is more sensitive to video flaws.

In fact, compared with the novices, the Meetor shows a lower performance in controlling *FPR*, which can be a potential research direction to improve the system. However, the average *FPR* of the novices is too low, because the novices only annotated a small number of flaws. Thus, the experimental results with novices also present that the normal audiences of meeting recordings usually have a higher tolerance and acceptance of video content flaws, so professional video editors might finish unnecessary or over-qualified work and waste a lot of the professional workforce. To provide an intuitive sense, we illustrate the detailed results of different subjects, Meetor, and the professional video editor in Figure 14, using the colors of Figure 7 to denote the flaws: red (blurriness), yellow (jitter), and blue (occlusion). In this example, although there are some false positive cases, Meetor can detect most flaws annotated by the video editor, which might be neglected by the novices. To this end, it is reasonable to apply the automatic video editing systems to finish this job.

6 USER STUDY

Section 5 illustrates that the proposed human-centered video content flaw detection algorithms can achieve SOTA performance. In this section, we conduct a user study to investigate whether the proposed Meetor system could generate acceptable meeting recordings for normal viewers.

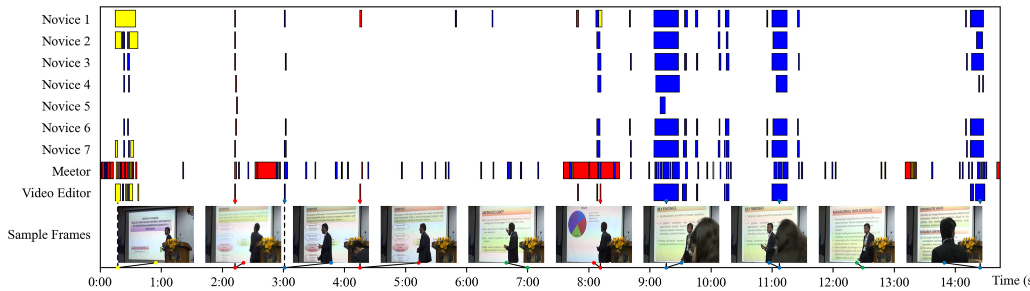

Fig. 14. An illustration of sample comparison results based on Video 1.

6.1 Experimental Settings

In fact, since the viewing experience is a relatively subjective sensation for different people, it is difficult to build an objective quantification method to evaluate the quality of generated meeting recordings, so we conducted a user study to investigate the performance of Meetor. Similar evaluation approaches can be found in related works [19, 20, 43, 44, 46]. Thereinto, the two studies of semi-automatic video editing systems [19, 20] mainly investigated the interaction efficiency of their system, while they did not focus on the quality of generated videos. For automatic video editing of dance videos [44], the authors invited 10 people to watch 14 videos and asked seven questions based on a seven-point Likert scale. QuickCut [43] prepared six videos to ask 3 experts and 10 normal viewers a binary question whether they think the generated video can be acceptable for publishing as a class project video or social media video. The most relevant work is Write-A-Video [46], which conducted a comparison between the generated results from the proposed system and a professional editor using two video tasks, containing 10 non-expert subjects who were invited to watch the video results and fill a seven-point Likert scale to evaluate the quality of produced videos.

Therefore, in this article, we adopt the method of Write-A-Video [46] to evaluate the performance of Meetor, as a similar form of the Turing Test [45]. We invite a professional video editor with eight years of video editing experience to edit the meeting recordings using commercial frame-based editing software. Then, we recruited 13 non-expert subjects to watch the final output videos of the editor and Meetor (7 males, 6 females, age: 37.2 ± 12.9). The subjects are asked to rate each video based on a 7-point Likert scale (1: very dissatisfied; 2: dissatisfied; 3: slightly dissatisfied; 4: neutral; 5: slightly satisfied; 6: slightly satisfied; and 7: very satisfied), considering three perspectives: synchronization of audio and video, general visual quality, and consistency of the meeting procedure. During the experiments, the videos will be displayed in full-screen mode in a random order within each pair, where the two videos are also in a random order, and the creators of each video are not exposed to the subjects. The subjects can freely jump, turn back, speed up the videos, or watch them multiple times. After finishing the scoring of a pair, the subjects will watch the next pair.

For the experimental materials, we collected four groups of the real meeting recording tracks in different scenarios, including academic conference, speech contest, annual meeting, and course lecture. Thereinto, the previous three groups are collected from a co-operated commercial video editing studio, and the course lecture is recorded by the authors in our classroom. The length information of the materials are shown in Table 5, and all videos have a resolution of $1,920 \times 1,080$ and FPS of 25. In fact, the original meeting is tediously long (e.g., the audio of the academic conference is longer than three hours), so it is hard to ask the recruited subjects to watch hours of videos. Therefore, we selected some short video segments as the experimental materials for this user study, which could ensure the time cost is lower than two hours to prevent fatigue of the

Table 5. Experimental Materials of Meeting Recording

Scenario	Video Track 1	Video Track 2	Video Track 3	Reference Audio
Speech Contest	0:13:31	0:13:27	–	Video Track 1
Course Lecture	0:12:19	0:12:23	0:12:06	0:12:30
Annual Meeting	0:13:36	0:13:32	–	0:14:25
Academic Conference	0:17:13	0:14:01	–	3:22:08

Table 6. Statistical Results of the User Study (Student's t-test with a Confidence Level of 95%)

Perspectives	Speech Contest			Course Lecture			Annual Meeting			Academic Conference		
	Mean	CI	p-value	Mean	CI	p-value	Mean	CI	p-value	Mean	CI	p-value
Editor: Synchronization	6.46	(6.15, 6.78)	0.45	6.46	(6.06, 6.86)	0.51	6.46	(6.15, 6.78)	1.00	6.38	(6.08, 6.69)	0.45
Meetor: Synchronization	6.62	(6.31, 6.92)		6.62	(6.31, 6.92)		6.46	(6.15, 6.78)		6.38	(6.22, 6.85)	
Editor: Visual Quality	5.85	(5.62, 6.07)	0.43	6.00	(5.57, 6.43)	0.34	5.85	(5.07, 6.62)	0.41	4.08	(2.96, 5.19)	0.37
Meetor: Visual Quality	5.54	(4.73, 6.34)		5.62	(4.85, 6.38)		5.46	(4.83, 6.10)		4.69	(3.76, 5.63)	
Editor: Synchronization	6.00	(5.57, 6.43)	0.19	5.85	(5.30, 6.39)	1.00	6.00	(5.57, 6.43)	0.81	4.92	(3.92, 5.92)	0.17
Meetor: Synchronization	5.46	(4.70, 6.23)		5.85	(5.36, 6.33)		5.92	(5.40, 6.44)		5.69	(5.07, 6.32)	

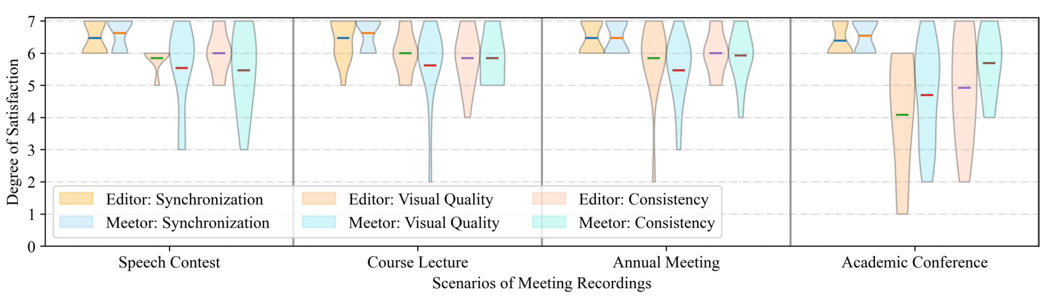

Fig. 15. Experimental results of the user study.

experimental subjects. Note that the notation "-" represents that the group does not have Video Track 3, and not all meetings have an audio recorded by an independent voice recorder, e.g., we utilize the audio of Video Track 1 as the reference audio for the scenario of the speech contest.

6.2 Results and Feedback

Figure 15 illustrates the experimental results of the user study, using a violin plot to show the score distribution and mean value of the 13 subjects regarding the four scenarios. We also made a statistical test based on Student's t-test with a confidence level of 95%, as shown in Table 6. According to the experimental results, the Meetor could provide a better experience in the synchronization of video and audio, which demonstrates the proposed audio-based video synchronization algorithm could achieve effective performance. For the general visual quality, the results of the video editor exceed the Meetor in 3/4 scenarios, while the results of the Meetor are in scope beyond "slightly satisfied." According to the feedback of some subjects, Meetor indeed captured more jitters and occlusions, but the editors can better control the general quality, including illumination change, presenters' action, and so on, which are not covered in Meetor. For consistency of the meeting procedure, the results cannot show a significant difference between the editor and Meetor, with ups and downs on both sides in the four scenarios, and the editor shows a better average value by a narrow margin compared with the Meetor. The feedback of the subjects provides some potential research directions, especially semantic understanding of the meeting procedure, For example, if the presenter is showing and switching the slides, the correct choice is to select the camera with slides. However, the Meetor might select a camera with only the presenter, which means the viewer

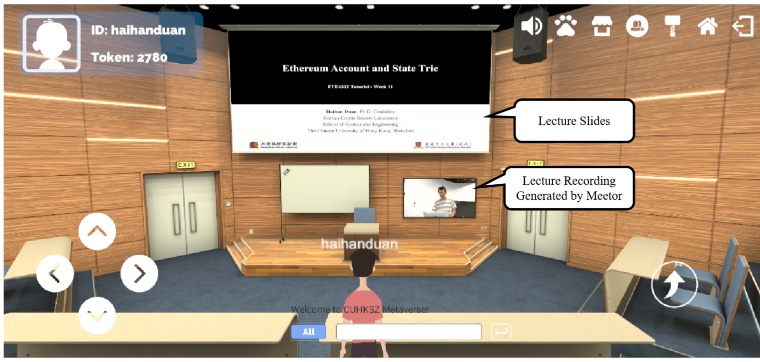

Fig. 16. Practical application of Meetor in university campus metaverse prototype.

would miss some slides. According to the statistical results, the p-value shows that there is no significant difference between the experimental results of the professional video editor and Meetor system. Generally speaking, the user study demonstrates that the proposed Meetor could achieve a satisfactory performance from the perspective of non-expert viewers, which can be utilized in practical applications to release the workload of professional video editors.

7 PRACTICAL APPLICATION IN A UNIVERSITY CAMPUS METAVERSE

In this article, we build a practical application demo in our university campus metaverse prototype, **The Chinese University of Hong Kong, Shenzhen Metaverse (CUHKSZ Metaverse)** [5]. Since a core function of a university is course teaching, we intend to build a virtual classroom to support the online learning of our students. Therefore, we import a virtual classroom scene *University Lecture Theater 03*[5] based on Unity.[6] More importantly, we apply the proposed Meetor system to help the teachers automatically edit lecture recordings. The teachers only need to input their lecture information (e.g., course ID, course number, date) and upload their recording materials and lecture slides, and then a digital twin [7] of this lecture can be created in our metaverse prototype. As shown in Figure 16, the main projector curtain displays the slides of the lecture, and the screen under the projector curtain plays the lecture recording generated by the Meetor system. In this virtual classroom, the students can review or self-study the course lectures multiple times as they wish, and the students learning the same course can also discuss with each other.

8 CONCLUSIONS

In this article, we propose a human-centered automatic video editing system for meeting recordings, named Meetor. Specifically, three core functions are implemented to build the Meetor, including video synchronization, video flaw detection, and automatic video compositing: (1) the audio-based video synchronization algorithm applies a sliding window searching method to synchronize the video recordings and reference audio, which could achieve the synchronization precision of 1 ms; (2) the human-centered video content flaw detection algorithms build a bridge between subjective sense and objective measures to assess the video quality, and the experimental results demonstrate that the proposed three algorithms can achieve SOTA performance compared with the existing approaches; (3) the automatic video editing system applies a greedy algorithm to compose the video segments and avoid the video content flaws as much as possible, and the user study

[5]https://assetstore.unity.com/packages/3d/environments/university-lecture-theater-03-224651
[6]https://unity.com/

illustrates the composed video recordings could achieve an approximate quality compared with recordings of a professional video editor. Moreover, we also show a practical application demo in a university campus metaverse prototype, where the proposed Meetor is utilized in facilitating automatic lecture recording editing of online learning. In the future, we will keep improving the performance of proposed algorithms, adding more semantic understanding modules from a human-centered perspective, and enlarging the scalability and robustness of the Meetor system.

REFERENCES

[1] Usman Ali and Muhammad Tariq Mahmood. 2018. Analysis of blur measure operators for single image blur segmentation. *Appl. Sci.* 8, 5 (2018), 807.

[2] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. 2014. Automatic editing of footage from multiple social cameras. *ACM Trans. Graph.* 33, 4 (2014), 1–11.

[3] Raghav Bansal, Gaurav Raj, and Tanupriya Choudhury. 2016. Blur image detection using Laplacian operator and Open-CV. In *International Conference System Modeling & Advancement in Research Trends (SMART'16)*. IEEE, 63–67.

[4] Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Bjoern Hartmann. 2013. DemoCut: Generating concise instructional videos for physical demonstrations. In *26th Annual ACM Symposium on User Interface Software and Technology*. 141–150.

[5] Haihan Duan, Jiaye Li, Sizheng Fan, Zhonghao Lin, Xiao Wu, and Wei Cai. 2021. Metaverse for social good: A university campus prototype. In *29th ACM International Conference on Multimedia*. 153–161.

[6] Haihan Duan, Junhua Liao, Lehao Lin, and Wei Cai. 2022. FLAD: A human-centered video content flaw detection system for meeting recordings. In *32nd Workshop on Network and Operating Systems Support for Digital Audio and Video*. 43–49.

[7] Abdulmotaleb El Saddik. 2018. Digital twins: The convergence of multimedia technologies. *IEEE Multim.* 25, 2 (2018), 87–92.

[8] Daniel P. W. Ellis, Courtenay V. Cotton, and Michael I. Mandel. 2008. Cross-correlation of beat-synchronous representations for music similarity. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 57–60.

[9] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

[10] Andreas Girgensohn, John Boreczky, Patrick Chiu, John Doherty, Jonathan Foote, Gene Golovchinsky, Shingo Uchihashi, and Lynn Wilcox. 2000. A semi-automatic approach to home video editing. In *13th Annual ACM Symposium on User Interface Software and Technology*. 81–89.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[12] Zhibin Hong, Chaohui Wang, Xue Mei, Danil Prokhorov, and Dacheng Tao. 2014. Tracking using multilevel quantizations. In *European Conference on Computer Vision*. Springer, 155–171.

[13] Ruibing Hou, Bingpeng Ma, Hong Chang, Xinqian Gu, Shiguang Shan, and Xilin Chen. 2019. VRSTC: Occlusion-free video person re-identification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7183–7192.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4700–4708.

[15] Junhwa Hur and Stefan Roth. 2017. MirrorFlow: Exploiting symmetries in joint optical flow and occlusion estimation. In *IEEE International Conference on Computer Vision*. 312–321.

[16] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. 2018. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *European Conference on Computer Vision (ECCV'18)*. 614–630.

[17] Saad M. Khan and Mubarak Shah. 2008. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 3 (2008), 505–519.

[18] Dieter Koller, Joseph Weber, and Jitendra Malik. 1994. Robust multiple car tracking with occlusion reasoning. In *European Conference on Computer Vision*. Springer, 189–196.

[19] Rodrigo Laiola Guimaraes, Pablo Cesar, Dick C. A. Bulterman, Vilmos Zsombori, and Ian Kegel. 2011. Creating personalized memories from social events: Community-based support for multi-camera recordings of school concerts. In *19th ACM International Conference on Multimedia*. 303–312.

[20] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. 2017. Computational video editing for dialogue-driven scenes. *ACM Trans. Graph.* 36, 4 (2017), 130–1.

[21] Florent Lefevre, Vincent Bombardier, Nicolas Krommenacker, Patrick Charpentier, and Bertrand Petat. 2018. Automatic video stream selection method by on-air microphone detection. *International Conference on Pattern Recognition and Artificial Intelligence (ICPRAI'18)*.

[22] Junhua Liao, Haihan Duan, Xin Li, Haoran Xu, Yanbing Yang, Wei Cai, Yanru Chen, and Liangyin Chen. 2020. Occlusion detection for automatic video editing. In *28th ACM International Conference on Multimedia*. 2255–2263.

[23] Qiong Liu, Yong Rui, Anoop Gupta, and Jonathan J. Cadiz. 2001. Automating camera management for lecture room environments. In *SIGCHI Conference on Human Factors in Computing Systems*. 442–449.

[24] Shijie Liu, Xiaohua Tong, Fengxiang Wang, Wenzheng Sun, Chengcheng Guo, Zhen Ye, Yanmin Jin, Huan Xie, and Peng Chen. 2016. Attitude jitter detection based on remotely sensed images and dense ground controls: A case study for Chinese ZY-3 satellite. *IEEE J. Select. Topics Appl. Earth Observ. Rem. Sens.* 9, 12 (2016), 5760–5766.

[25] David G. Lowe. 1999. Object recognition from local scale-invariant features. In *7th IEEE International Conference on Computer Vision*, Vol. 2. IEEE, 1150–1157.

[26] Markus Mathias, Rodrigo Benenson, Radu Timofte, and Luc Van Gool. 2013. Handling occlusions with Franken-classifiers. In *IEEE International Conference on Computer Vision*. 1505–1512.

[27] Xiongkuo Min, Guangtao Zhai, Jiantao Zhou, Mylene C. Q. Farias, and Alan Conrad Bovik. 2020. Study of subjective and objective quality assessment of audio-visual signals. *IEEE Trans. Image Process.* 29 (2020), 6054–6068.

[28] Wanli Ouyang and Xiaogang Wang. 2012. A discriminative deep model for pedestrian detection with occlusion handling. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 3258–3265.

[29] Wanli Ouyang and Xiaogang Wang. 2013. Joint deep learning for pedestrian detection. In *IEEE International Conference on Computer Vision*. 2056–2063.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[31] José Luis Pech-Pacheco, Gabriel Cristóbal, Jesús Chamorro-Martinez, and Joaquín Fernández-Valdivia. 2000. Diatom autofocusing in brightfield microscopy: A comparative study. In *15th International Conference on Pattern Recognition*, Vol. 3. IEEE, 314–317.

[32] International Telecommunication Union. 1998. Recommendation ITU-R BT.1359-1: Relative Timing of Sound and Vision for Broadcasting. https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.1359-1-199811-I!!PDF-E.pdf. Accessed: 03-02-2024.

[33] Abhishek Ranjan, Jeremy Birnholtz, and Ravin Balakrishnan. 2008. Improving meeting capture by applying television production principles with audio and motion detection. In *SIGCHI Conference on Human Factors in Computing Systems*. 227–236.

[34] Reza Rassool. 2017. VMAF reproducibility: Validating a perceptual practical video quality metric. In *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*. IEEE, 1–2.

[35] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[36] Nicolas Staelens, Margaret H. Pinson, Philip Corriveau, Filip De Turck, and Piet Demeester. 2015. Measuring video quality in the network: From quality of service to user experience. In *9th International Workshop on Video Processing and Consumer Electronics*. 5–6.

[37] Patrik Sundberg, Thomas Brox, Michael Maire, Pablo Arbeláez, and Jitendra Malik. 2011. Occlusion boundary detection and figure/ground assignment from optical flow. In *Conference on Computer Vision and Pattern Recognition (CVPR'11)*. IEEE, 2233–2240.

[38] Yoshinao Takemae, Kazuhiro Otsuka, and Naoki Mukawa. 2003. Video cut editing rule based on participants' gaze in multiparty conversation. In *11th ACM International Conference on Multimedia*. 303–306.

[39] Chang Tang, Xinzhong Zhu, Xinwang Liu, Lizhe Wang, and Albert Zomaya. 2019. DefusionNet: Defocus blur detection via recurrently fusing and refining multi-scale deep features. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2700–2709.

[40] Xiaohua Tong, Zhen Ye, Yusheng Xu, Xinming Tang, Shijie Liu, Lingyun Li, Huan Xie, Fengxiang Wang, Tianpeng Li, and Zhonghua Hong. 2014. Framework of jitter detection and compensation for high resolution satellites. *Rem. Sens.* 6, 5 (2014), 3944–3964.

[41] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3D convolutional networks. In *IEEE International Conference on Computer Vision*. 4489–4497.

[42] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*. 6450–6459.

[43] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. Quickcut: An interactive tool for editing narrated video. In *29th Annual Symposium on User Interface Software and Technology*. 497–507.

[44] Shuhei Tsuchida, Satoru Fukayama, and Masataka Goto. 2017. Automatic system for editing dance videos recorded using multiple cameras. In *International Conference on Advances in Computer Entertainment*. Springer, 671–688.

[45] Alan M. Turing. 2009. Computing machinery and intelligence. In *Parsing the Turing Test*. Springer, 23–65.
[46] Miao Wang, Guo-Wei Yang, Shi-Min Hu, Shing-Tung Yau, and Ariel Shamir. 2019. Write-a-video: Computational video montage from themed text. *ACM Trans. Graph.* 38, 6 (2019), 1–13.
[47] Mi Wang, Ying Zhu, Jun Pan, Bo Yang, and Quansheng Zhu. 2016. Satellite jitter detection and compensation using multispectral imagery. *Rem. Sens. Lett.* 7, 6 (2016), 513–522.
[48] Yang Wang, Yi Yang, Zhenheng Yang, Liang Zhao, Peng Wang, and Wei Xu. 2018. Occlusion aware unsupervised learning of optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition*. 4884–4893.
[49] Alper Yilmaz, Xin Li, and Mubarak Shah. 2004. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 11 (2004), 1531–1536.
[50] Kai Zeng, Yaonan Wang, Jianxu Mao, Junyang Liu, Weixing Peng, and Nankai Chen. 2018. A local metric for defocus blur detection based on CNN feature learning. *IEEE Trans. Image Process.* 28, 5 (2018), 2107–2115.
[51] Yingxue Zhang, Yingbin Wang, Feiyang Liu, Zizheng Liu, Yiming Li, Daiqin Yang, and Zhenzhong Chen. 2018. Subjective panoramic video quality assessment database for coding applications. *IEEE Trans. Broadcast.* 64, 2 (2018), 461–473.
[52] Wenda Zhao, Fan Zhao, Dong Wang, and Huchuan Lu. 2018. Defocus blur detection via multi-stream bottom-top-bottom fully convolutional network. In *IEEE Conference on Computer Vision and Pattern Recognition*. 3080–3088.
[53] Chunluan Zhou and Junsong Yuan. 2018. Bi-box regression for pedestrian detection and occlusion estimation. In *European Conference on Computer Vision (ECCV'18)*. 135–151.
[54] Vilmos Zsombori, Michael Frantzis, Rodrigo Laiola Guimaraes, Marian Florin Ursu, Pablo Cesar, Ian Kegel, Roland Craigie, and Dick C. A. Bulterman. 2011. Automatic generation of video narratives from shared UGC. In *22nd ACM Conference on Hypertext and Hypermedia*. 325–334.

Received 8 December 2022; revised 18 October 2023; accepted 9 February 2024