Yuanyi Sun, Sencun Zhu*, and Yu Chen

ZoomP^3: Privacy-Preserving Publishing of Online Video Conference Recordings

Abstract: The COVID-19 epidemic has made online video conferencing extremely popular throughout the world, with many schools, companies and government sectors using video conferencing applications (e.g., Zoom, Google Meet) in a daily basis. These applications also provide local or cloud recording services, which allow the replay or sharing of video conference recordings (VCRs) in a later time. Such convenience, however, can easily cause infringement of privacy as meeting participants' personally identifiable information (e.g., face, name, voice) may be exposed to the public without their awareness or consent. While privacy regulation and training can help relieve the situation, efficient and effective tools are also highly desired to protect the privacy-sensitive users in the VCRs before their public releases. In this work, we propose the first Privacy-Preserving Publishing system (ZoomP^3) that automatically processes video and audio information in VCRs for privacy protection. Besides leveraging and integrating multiple state-of-the-art computer vision and audio processing tools seamlessly into our system, a number of optimization algorithms are proposed to improve the scalability of the system, enabling it to protect the privacy of long video conferences. We have conducted various tests with short and long videos, and the results (with online demos) verified that ZoomP^3 system is suitable for large-scale use. It may be applied as an online service, e.g., by Zoom, or by large organizations such as universities, research institutes and government sectors.

Keywords: Online Video Conference, Privacy-Preserving Publishing, Privacy Protection, Speaker Diarization

DOI 10.56553/popets-2022-0089
Received 2021-11-30; revised 2022-03-15; accepted 2022-03-16.

Yuanyi Sun: Penn State University, E-mail: yus160@psu.edu
***Corresponding Author: Sencun Zhu:** Penn State University, E-mail: sxz16@psu.edu.edu
Yu Chen: Binghamton University, SUNY, E-mail: ychen@binghamton.edu

1 Introduction

Due to the impact of the COVID-19 epidemic, more and more schools, companies, and government sectors have migrated to work online [1]. Consequently, online video conferencing has become the main vehicle for people to communicate in this special period. While people enjoy the convenience, online video conference applications also bring up a number of potential security and privacy issues [19]. For instance, zoombombing is a famous security attack, where attackers break into other people's zoom rooms and disrupt the meeting by making noises or adding insulting annotations to the shared screen [5, 20]. Keystroke inference attacks [36] are also reported, where an attacker tries to infer the keystrokes of a video conference participant from one's observable body movements in the video. Security solutions have also been introduced, e.g., by adding password authentication and waiting rooms, adding end-to-end encryption, etc.

However, privacy issues have not received sufficient attention and understanding yet. Different from many emerging attacks impacting the online conferencing, the consequences of privacy violations are not immediate. The potential of privacy leaks may exist for a long time after the meeting is over, and the victims may not be aware of the privacy risk. Participants are revealed when the video recordings are released into public domains such as YouTube [28]. For example, when schools offer online courses using Zoom, video conference recordings (VCRs) can be easily shared among students or posted on public websites such as YouTube. These videos contain a large amount of facial information about students, and there is a high potential of privacy-violation once an attacker collects lots of video data. Moreover, schools often have privacy rules, stating that if students appear in a class VCR, one may not share the VCR beyond the current semester for the purpose of protecting students' privacy. Consequently, professors are unable to reuse or disclose videos of online courses that include students. While this rule solves the student privacy issue, it also increases the repetitiveness of the professor's lectures. It is compelling to have a holistic solution, by which neither students' privacy nor course video sharing is affected.

Actually, a recent study has shown that VCRs are ideal for attackers to collect user data [28]. If public VCRs are not protected, an attacker can crawl a large number of VCRs to extract facial information and corresponding participants' names, determine the participants' age and gender through machine learning (ML) and collect their phone, email, social network account by searching the name, which will result in a huge database with participants' personally identifiable information (PII). It is an urgent need for an automated and efficient video privacy-preserving solution.

In practice there are lots of challenges to build a privacy-preserving system for VCRs. For example, how can one *continuously and automatically* locate the sensitive areas in the VCR for privacy-sensitive participants despite the frequent switching of view modes in an online conference? how can one locate privacy-sensitive participants in the audio and also the displayed names of privacy-sensitive participants in the VCRs? how can one automatically and effectively protect the various kinds of sensitive information in VCRs? how can one make the whole process of privacy protection scalable and also conveniently fix the missed protection from automatic processing, if any? Although a number of novel systems have emerged over the years as a result of the boom in computer vision and computer speech technologies [26, 31, 34, 35, 37, 39], they have different application scenarios, some for surveillance systems, some for virtual reality systems, and so on. Currently, there is not an automatic privacy protection solution for online VCRs.

In this paper, we propose a **P**rivacy-**P**reserving **P**ublishing system (ZoomP^3) for online VCRs. ZoomP^3 system can run on both the user side and on the cloud as a cloud service. Compared to professional video editors that need a lot of expertise to edit videos in order to protect privacy, ZoomP^3 system is more user-friendly by making the system mostly automatic. A number of performance optimizations (e.g., exploring the visual similarity of adjacent frames, efficient view switching detection, parallelization) are introduced to enable the privacy protection of long videos. We also propose a semi-automatic patching system, which makes it easy for users to quickly fix any missed protection due to the imperfection of computer vision algorithms underlying our automatic system. A proof-of-concept system is implemented and tested in various scenarios. The experimental results show that ZoomP^3 meets the design goals and helps users generate privacy-protected videos quickly and efficiently. In summary, this work makes the following contributions:

– Motivated by the increasing privacy concerns in prevalent use of online video conferencing, we propose ZoomP^3, a system for efficient and effective privacy protection of privacy-sensitive participants in VCRs. To the best of our knowledge, ZoomP^3 system is the first system of its kind.
– Besides leveraging and integrating multiple state-of-the-art computer vision (face detection, face recognition, OCR) and audio processing tools (speaker diarization) *seamlessly* into a single system, a number of optimization algorithms are proposed to improve the scalability of the system, making it possible to protect the privacy of long video conferences.
– We have conducted various benchmark tests with short and long videos, and the results (with online demos [17]) show that ZoomP^3 system is feasible for large-scale use; hence, it may be provided as an online cloud service.

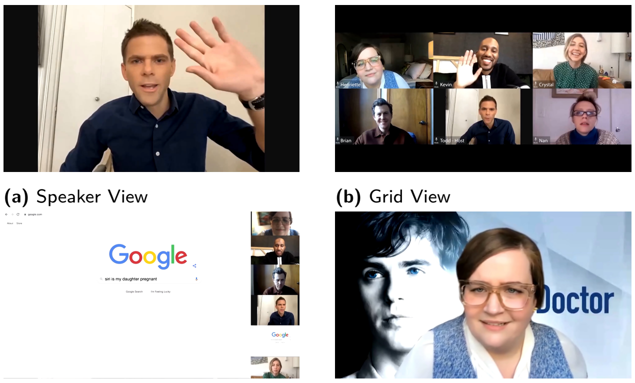
2 Preliminaries

2.1 Background on Video Encoding and Video Conferences

A video consists of two tracks, the video track and the audio track. The two tracks are independent of each other, but integrated in a single file called a container. The most commonly used containers are MP4, AVI, MKV, etc. Containers, however, do not indicate how a video is organized and encoded, but how to store the encoded video in a certain format on the hard drive. Inside a container there are also some metadata, such as the bit rate, frame per second (FPS), and length of the video. As for video encoding formats, the common industrial standards are H.264, H.265, AV1, etc. These encoding formats determine the encoding algorithms for video compression (i.e., how to compress a video frame-by-frame). The most common audio encoding formats are MP3, AAC, and the audio track information is stored inside the same container as the video track information. To process video and audio separately, we usually need to separate the video and audio tracks.

There are typically three display modes in the recording of a video conference [18], as shown in Fig. 1. The active speaker view mode (Fig. 1a) displays only the current speaker's face in the screen. The second is the gridview, also called gallery view mode (Fig. 1b), which on one screen displays multiple users' thumbnail views in a grid layout. The gridview may be dynamic as people join or leave the conference at any time and there is a maximum of number participants on one screen. When the number of participants in a conference is below the max number, the gridview expands and contracts as participants join and leave the meeting. The third mode is the shared screen mode (Fig. 1c), which displays the shared screen of the speaker (e.g., PowerPoint slides). In this mode, the host may also configure the recording settings to record one or multiple thumbnail views, which are normally shown on the right side of the shared screen. Irrespective of the mode, the host may also set up to record participants' names (also called display name or name tag) in the recording, which appear at the bottom left corner of each participant's window view. Moreover, a participant may choose a virtual background (Fig. 1d) instead.

Note that although a participant or a host can choose various kinds of layouts for personal views, the recorded video layouts are basically the above three types, which are independent of the personal view modes chosen by the participants.



(a) Speaker View **(b)** Grid View **(c)** Shared Screen View **(d)** Virtual Background

Fig. 1. Screenshots from NBC Saturday Night Live (SNL) Zoom Video [16].

2.2 Application Scenarios

Our system is designed as a *generic* solution for privacy-preserving publishing of VCRs. Its input is any kind of recorded conference videos, recorded by Zoom, Webex, Google Meet or similar tools, together with small examples of personally identifiable information (PII) to be protected. It is a post-processing system instead of a real-time protection system. Certainly, in the future, conference recording software may provide certain features for better privacy control during the recording time, but it will be unlikely to address all the PII leakage problem beforehand in all real-world applications. On the other hand, while video and audio privacy protection can be handled by a post-production editor (e.g., Premiere and Audition), post-production editing is a very time-consuming and labor-intensive task. Our ZoomP^3 system aims to automate and speed up the process.

ZoomP^3 system has two basic working modes, one is *whitelist mode* and the other is *blacklist mode*. The whitelist mode fits such application scenarios as online teaching. A professor, say Bob, may teach a 50-minute session course via Zoom. After the class, the professor may need to publish the recorded video to a public website or reuse it for a future semester. To conform to university privacy policies, he needs to protect any PII of the students so that the viewers would not know which students participated in the class. In this case, Bob can whitelist himself in the video. Specifically, he may select a picture of himself in the video and choose a few seconds of lecture that only includes his speech. ZoomP^3 will automatically classify all the faces and voices in the video into two categories: Bob or non-Bob. It will protect the face and voice of anyone who is not Bob.

The second mode is the *blacklist* mode, which applies to the scenario where a participant, say Alice, explicitly asks the meeting host to remove her PII from the video. Following the same method by selecting a frame containing Alice's face and a few seconds of video segments including her speech, ZoomP^3 system will learn to identify her in the entire video and protect her PII accordingly. With the support of the above two basic working modes, ZoomP^3 can also handle the cases when multiple users need to be blacklisted or whitelisted. When a user Bob chooses one of the two modes for his VCR, by default, the other participants who are not explicitly whitelisted (or blacklisted) will be blacklisted (or whitelisted).

2.3 Design Goals and Research Challenges

As face and voice information has been widely used for user authentication or identification, each of them is considered as PII to be protected in ZoomP^3 system. Besides, some VCRs may also contain name tags of participants, which should also be protected.

Threat Model In our threat model, we consider outside attackers who crawl and collect VCRs from public domains like YouTube to extract the aforementioned

PII, as described in the recent study [28]. We do not consider an insider attack where an attacker is either a participant of the same video conference (e.g., as classmates or colleagues) or someone with extra knowledge about a participant (e.g., knowing his/her *unique* background or virtual background or clothing). Especially, we consider *user-identification attack*, by which the outside attacker tries to identify a user by his facial biometric or name tag, or *user-linking attack*, by which an *outside* attacker tries to link users in different videos by accurately matching their voice signals. We will present the rationale shortly. With our application scenarios and threat model in mind, we discuss the specific design goals and research challenges below.

High Accuracy To protect the PII of privacy-sensitive users in VCRs, one may apply covering, blurring, morphing, swapping or through other appropriate methods. In this study, for the purpose of demonstration, we choose to blur faces, morph voice and cover name tags of privacy-sensitive users. As image search has become a very powerful tool to identify a person with the input of a face image if the user has personal photos posted somewhere (e.g., social networks, personal webpages), the exposure of a face in a single video frame could result in a total failure of protection, even if the face is protected in all other frames. This can be considered as an all-or-nothing property for visual privacy. In other words, false negatives are not tolerable in our system. On the other hand, we may live with some false positives, that is, (mistakenly) hiding the faces of public users (i.e., privacy insensitive people) in a few frames, although it is unnecessary. The same arguments on false positives and false negatives also applies to name tag protection. The high-accuracy protection of sensitive faces and name tags are necessary for defeating *user identification attack*.

However, for efficiency, our current design does not target very high accuracy on audio information protection. In the past, fingerprint has been commonly used for user authentication/identification, and facial authentication has also been deployed for a few civil applications such as train/airplane boarding in some countries. However, as far as we know there is no large-scale collection of voice biometrics to build an authentication database that is accessible to the public. Companies like Google and Amazon may collect some audio information from their users through their voice assistant devices, such audio information may not be shared to the public due to privacy laws. On the other hand, voice-based authentication or identification is mostly for individual users to log into their smart devices and the voice biometric is stored securely in the devices. Hence, we do not assume the attacker can build or steal a large voice biometric dataset for user identification or leverage an existing search engine. We do not consider the voice based user-identification attack by an *outside* attacker, but focus on defeating the user-linking attack. That is, our design goal is to achieve the best protection accuracy instead of perfect accuracy for audio signals so that an attacker will not be able to link users in different videos through automatic matching of voice signals.

High Efficiency Unlike manual video editing, which is often a very time consuming and tedious process, we will develop highly efficient automated detection and protection methods based on deep learning. However, deep learning algorithms themselves could be very time consuming, negating their advantages over manual editing. Therefore, we aim to optimize the performance so that given a video recording, our privacy protection system will finish the processing in a timely fashion. This will make our system scalable enough to be hosted as a network service in the cloud to serve many users.

3 System Architecture and Design

3.1 Architectural Overview

Figure 2 shows the high-level workflow of ZoomP^3 system. Each rectangle with a solid border is a processing module, each text with an icon is the input and output of the processing module, and the inside of the dashed rectangle indicates that this processing is optional. The input video first goes through the module of target participants selection, where the target partic-

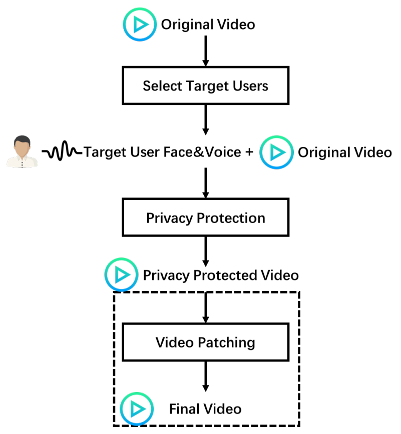


Fig. 2. High Level Workflow of ZoomP^3.

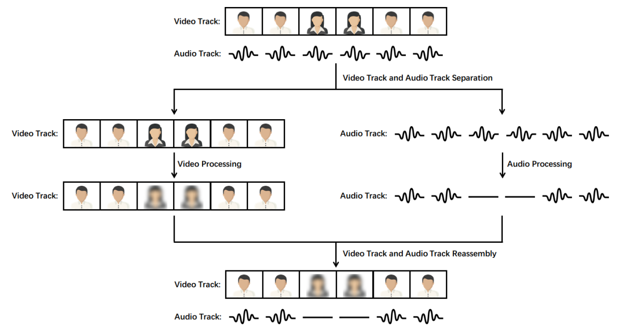

Video Track:
Audio Track:
Video Track and Audio Track Separation
Video Processing
Audio Processing
Video Track and Audio Track Reassembly

Fig. 3. Video Processing Flow of ZoomP^3.

ipant's facial area and a segment of the target participant's voice interval are manually labeled in the original video. The labeled result and the original video are both used as input to the privacy protection module, the output is a privacy protected video. Ideally, the privacy protected video shall have all privacy sensitive participants protected so that the public cannot identify them by faces or voices. The privacy protection process is based on deep learning; however, there is no deep learning that can guarantee 100% accuracy. Therefore, a semi-automatic video patching system is added. If the output video does not protect the privacy sensitive users with 100% accuracy, then the patching system is employed to patch the video for additional privacy protection.

Figure 3 shows the processing flow of our system. Given a VCR (e.g., Zoom video), ZoomP^3 first separates it into the video track and the audio track. The video track is then processed by our video privacy protection module, which blurs sensitive faces in all frames, and the audio track is processed by our audio privacy protection module, which morphs sensitive voices, respectively. These two processes are executed in parallel. Finally, the system resembles the privacy-protected video track and the privacy-protected audio track to output the privacy protected video for direct sharing or patching (if necessary).

3.2 Video and Audio Separation and Reassembly

There are a few well-known video encoding formats like H.264, HEVC, AV1, and audio encoding formats like AAC and MP3. For instance, Zoom video recording uses H.264 and AAC encoding. When separating and reassembling video and audio tracks, if the format of the original input is different from the output format, transcoding will happen. Transcoding one format to another format takes a long time. To improve usability and efficiency, in our system design we make sure transcoding will never happen.

Specifically, even though there are mature video operation libraries (e.g., MoviePy [8]) that can qualify the separation and reassembling job, they do not process videos at the low level, so transcoding will happen with such libraries. In ZoomP^3 system, FFmpeg [4] is selected in the "copy" encoding mode to process videos at a low level to avoid transcoding.

3.3 Video Processing Module

The goal of our video processing module is to automatically identify the privacy-sensitive participants in the video and then protect them. To achieve the goal, the first step is window view segmentation. Each frame is broken into individual views where each view contains one participant, no matter whether the frame is in grid view or speaker view mode. Then in each frame we identify the exact view where a privacy-sensitive participant is located, and accordingly protect his/her entire view. In this way, nothing about this individual will be visible, hence achieving perfect visual privacy.

Technically, Hoffman transform may be employed, which is designed to detect specific shapes (in our case rectangles). However, the Hoffman transform requires a clear boundary to ensure accuracy, whereas in video conferencing, all participant views are next to each other and a view window does not necessarily have a clear border (e.g., when the background of a user is dark or the large margins of a user's view window are black when a user joins a meeting through phone), so it will lead to low accuracy when performing window segmentation. We have also considered using heuristics (manually labeling a view to provide view size information for window segmentation), they do not work in a dynamic session where people may join and leave at any time. Also there are switchings between grid layout and speaker layout, causing the change of view window size.

Due to the above challenges, instead of protecting the entire view of a privacy-sensitive individual, ZoomP^3 system turns to face detection and recognition techniques and protect only the faces.

3.3.1 Face Detection, Recognition and Classification

In ZoomP^3 system, one of the major design requirements is a high face detection/recognition accuracy. Participants may have different light conditions, different face sizes in the views, change facial expression, and may keep moving and turning their heads. From privacy protection point of view, no matter how many participants (and faces) appear in a video, they fall into two classes, the sensitive class (S-class) and the public class (P-class). Depending on the mode of operation (blacklist or whitelist), ZoomP^3 system assigns the output of face recognition to these two classes, and only the faces in the S-class will be protected later on.

3.3.2 Greedy Backward Processing

An intuitive way for video processing is to process the video frame-by-frame, where face detection is applied to each frame of the video, followed by face identification. If a sensitive face identified in a frame, it will be blurred. While it is conceptually simple, frame-by-frame processing is computationally expensive and very time consuming. A possible optimization is to leverage object tracking in videos, which is often more computationally efficient than continuous object detection. However, when multiple faces need to be protected (e.g., in the gridview model), the advantage of object tracking over continuous objection/identification may disappear. Moreover, object tracking may fail in some scenarios, such as switching between gridview mode and speaker view mode.

We propose a Greedy Backward Processing (GBP) method for efficient face protection by exploring the possible similarity between adjacent frames in a conference video. Specifically, GBP divides a video into groups of fixed-length L consecutive video frames, and selects one frame, named *keyframe*, in each group to represent the group. Assume the video frame rate is F FPS, then the time duration of each group T_g is L/F second. For example, when $F = 30$ and $L = 6$, $T_g = 0.2$ second. A keyframe can be any frame in the group as long as all the groups choose the same relative position (e.g., the last frame) as their keyframes. Face detection and identification are only conducted on the keyframe in each group and a very lightweight heuristic algorithm is applied to protect the frames between keyframes. Thus, we drastically reduce the amount of computation.

To quickly get an intuition of our heuristic algorithm, let us assume a simplified scenario with one face only in each frame. GBP algorithm can be easily extended to the gridview mode with multiple faces. Assume there are two participants, a privacy sensitive participant S and a public one P. As mentioned earlier, no matter how many faces are there in the video, they can be classified into two classes. When no face is detected in a frame, the frame is considered as a P frame.

Our GBP algorithm is based on the following observations. First, as long as group size L is not too large, if two adjacent keyframes contain the same face, it is reasonable to assume there is no change of faces in between. Second, if the face is changed either from S to P, or from P to S in these two keyframes, we can make a conservative estimation by assuming all the non-keyframes in between contain the face of S instead of P. Because false positives (i.e., P is mistreated as S) does not affect privacy or hurt usability in a very short time between two adjacent keyframes, whereas false negatives (i.e., mistreating S as P) will cause the break of privacy.

In above cases, as long as a keyframe contains S, GBP will simply protect the same face area as that of S in all the non-keyframes between S and its proceeding keyframe as well as between S and its following keyframe without performing face detection/identification on them. This is our face protection rule (R1). Since faces may move slightly in the views between two adjacent keyframes, we will enlarge the face areas to be protected (e.g. 1.5 times of the default face area returned by the underlying face detection algorithm). In Appendix (Section 10.1) we give a running example of our GBP algorithm with $L = 3$.

The GBP algorithm is very conservative by assuming *all* the $L-1$ non-keyframes between two keyframes contain S instead of P whenever one of the two keyframes contains S. In Appendix (Section 10.2), we further present two optimization algorithms (based on binary search algorithm and perceptual hash, respectively) for accurate identification of the non-keyframe where speaker changes.

3.3.3 Processing in Gridview Mode and Shared Screen View Mode

Next we explain how GBP algorithm works in the gridview mode and shared screen mode. The shared screen mode can be actually considered as a special case of gridview mode with one or multiple thumbnail views on the right side of the shared screen.

GBP algorithm works in the same way in the gridview mode as in the speaker view mode except that in the gridview mode the face detection/identification algorithms on the keyframes return multiple faces in different locations of the screen. If a sensitive user is identified in a grid in the current keyframe, GBP algorithm is applied to this sensitive grid instead of full screen in speaker view mode. As a result, the same area in each of the proceeding $L-1$ non-keyframes as the detected face area in this sensitive grid will be protected blindly. Similar to the case of speaker mode, due to the conservative name of GBP, in the gridview mode over-protection can happen in this sensitive grid. If there are multiple sensitive participants, GBP will do the same to all such participants one-by-one. Our GBP algorithm works the same when switching between different view modes happens although the sizes of faces are different in the gridview mode and the speaker view mode.

3.3.4 Name Tag Protection

The vast majority of online conferencing systems display a name tag at the bottom left corner of each individual's window view. The conference host may select the option to record name tags in the account settings. Name tags reveal information about attendees [28]. Especially, when attendees use their real names, an attacker may search through many videos to find exactly which conferences a real-name user has participated. As the name tags vary in content, size and sometimes appear translucently, it is difficult to detect them accurately through existing object detection methods like *matchtemplate()* in OpenCV.

Our solution is to leverage the optical character recognition (OCR) technology, which, given an image, outputs all the embedded texts as well as the coordinates of the bounding box for each character (in the case of English, also for each string of characters). Then, sensitive names can be identified and protected. Specifically, in the blacklist mode when sensitive names are given for protection, our system simply searches them in the OCR-extracted texts in a video frame. When a match is found, the bounding box of the identified name is blurred in the frame.

However, it is more challenging to handle the whitelist mode. The user may provide some names that should be open to public whereas all other names should be protected. While it is easy to simply blur *all* the text areas except the whitelisted name that is matched in a frame, it mostly likely destroys the normal textual contents in a screen, as a screen may contain multiple types of texts in different locations (e.g., texts in shared screen). To avoid the mistake of over-protection, we need to tell name tags from the other irrelevant texts. Our system uses the given name (either for blacklist or for whitelist) as the *anchor* to locate other name tags. In the gridview mode all the name tags are also organized in grid layout. Based on the location of the anchor name and its font size, we can heuristically locate the other name tags in the same row and the same column as the anchor name, and further repeatedly locate all the other name tags in the entire screen. We do not use the above heuristic for grid segmentation because the recording of name tags is optional, up to the record setting by the host.

In Zoom, when a participant turns off the camera but with audio on, the name will typically appear in the center of the view with black background. A simple heuristic is adopted to address this corner cases. Based on our observation, such names are displayed in white color with font size larger than that of name tags, and we can detect its black background by checking whether there are at least 20 black pixels surrounding each side of the displayed name.

ZoomP^3 system does not need to perform OCR for every frame in a video stream; otherwise, it could be computationally too expensive. Instead, by treating each identified name tag as either a P-class or a S-class object as it treats faces, our system only performs OCR to the keyframes and uses the same GBP (or BSP) algorithm to protect the name tags in the non-keyframes. For efficiency, name-tag protection and face protection are combined in each iteration instead of processing the entire video twice separately.

3.3.5 Performance Gain Leveraging Parallelism

All our algorithms can be executed in parallel in a multicore CPU for higher efficiency. Specifically, suppose we employ K threads to process a video with group size L. With the GBP algorithm, in each round a thread processes one group of frames by applying face detection/identification on the keyframe in its current group. Depending on the label (P or S) of its keyframe and that of the previous keyframe, accordingly it will process the $L-1$ non-keyframes in between. Once the current round is completed, next K groups of frames will be processed in parallel.

3.4 Audio Processing Module

This module has two major tasks: (T1) classify voices into two classes (i.e., P-class and S-class), and (T2) protect the S-class. A key technique to solve T1 is *Speaker Diarization (Diarisation)*, which is a process to partition an audio stream with multiple people into homogeneous segments based on speaker identity. Its output is a list of who-spoke-when [14]. UIS-RNN [42], UIS-RNN-SML [22] are well-known diarization algorithms based on RNN. If a video has two speakers, Alice (sensitive) and Bob (public), the above two algorithms collect lots of audio samples of Alice and Bob as the training dataset to train a classification model, which outputs the list of who-spoke-when. Unfortunately, they do not work when samples are few, because we may have only one video to process. Furthermore, the training process often takes a long time, which will hurt user experience. We need an algorithm that only labels a speaker once and does not require heavy training. Therefore, our system instead adopts Resemblyzer [13] to derive a high-level representation of a voice through a deep learning model (e.g., LSTM). Given a small audio segment, it creates a summary vector (embedding) of 256 values to summarize the characteristics of the spoken voice. Intuitively, we may first divide a given audio stream into small segments and employ Resemblyzer to extract the embedding for each segment. Then, we can apply a clustering algorithm such as **K**-means to cluster these segments into K groups if the audio is known to contain K speakers.

However, the above intuitive idea does not solve task T1 for three reasons. First, it is difficult to predetermine the number of speakers in a VCR before using our system to process it; second, we are dealing with a binary speaker classification problem, which divides each audio segment into either the P-class or the S-class, instead of identifying each individual speaker; third, given a speaker to be whitelisted or blacklisted, our system needs to identify that speaker; that is, speaker identification is required for some participant(s) but not for all. To address the above new problems, we propose the following process to classify speakers.

For each whitelisted (or blacklisted) speaker, say Bob, we create a reference embedding for him based on a few seconds of his audio (manually labelled). Then, the audio stream of the VCR is divided into small segments, and for each segment (no matter if it contains one speaker or more), we compute its embedding and match it with each reference embedding vector. If the similarity (confidence) score is greater than or equal to a pre-defined threshold θ, we claim that the segment belongs to that reference speaker. The sequential segments of the same speaker form a sentence, and sentences are separated by a brief silence.

Here we need to answer two questions: *what is the appropriate segment length* and *what is the appropriate threshold* θ? Using smaller segment (i.e., more segments per second) will lead to a more accurate diarization result but introduce more burden on computation. Based on our empirical study, we find 15 segments per second (i.e., 67 ms per audio segment) can balance accuracy and efficiency well. Our empirical study also finds that $\theta = 0.7$ is a good threshold to balance false positives and false negatives in audio protection. Figure 4 shows an example result of speaker diarization with two speakers, Alice and Bob. We start by first manually labelling an audio segment (a few seconds) for Bob (this is the whitelist mode, blacklist mode works similarly). Through diarization, the system reports the confidence score for Bob on each segment. With $\theta = 0.7$, most of the segments are predicted accurately.

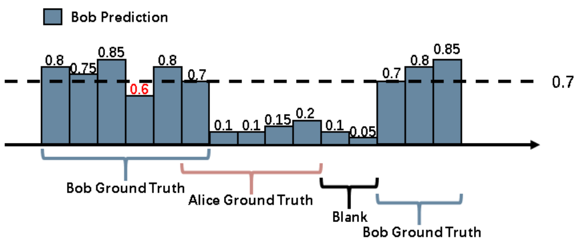


Fig. 4. Visualization of Diarization Result (whitelist mode, segment length 67ms and $\theta = 0.7$).

The example also shows two types of mis-classifications errors. The first is to mis-classify a P-class user to a S-class user, resulting in over-protection. In Fig. 4, the 4th segment has the similarity score 0.6 for Bob, which is below the threshold 0.7, so it will be protected. The second error is to mis-classify a S-class user into a P-class user, leading to under-protection. For the 6th segment in Fig. 4, the system predicts the similarity score 0.7 for Bob, but the ground truth is that this segment belongs to Alice. As a result, this segment of Alice will not be protected.

To reduce mis-classification errors, the adjacent segments are used to correct the mis-identified or non-identified segments in a small gap. This is based on a realistic assumption that speakers rarely speak for a very short time interval with meaningful and identifiable content. Therefore, whenever a small gap is detected in a speaker's speech, we treat the audio segments in the gap as that of the speaker. We empirically set the threshold gap length as 5 segments (i.e., $5/15 = 0.33$ second). As such, the 4th segment will be correctly assigned to Bob. Moreover, as our segment size is very small, the effect of such errors would be very small unless the same type of error lasts in a continuous sequence of segments (e.g., a few seconds long).

Regarding task T2, there are two ways to protect voice: mute or morph. The mute method is preferable when the speech of a sensitive user contains the identifiable information of the speaker or of some other sensitive users. The drawback is that it may more or less disrupt the conversational flow, although this is less of an issue when sensitive users do not talk much, for example, in a low-interaction classroom setting. The voice morph method has the advantage of preserving the conversational flows. There are different ways to change the voice; for instance, commercial software like Morphvox Pro [7] can even naturally change voice across genders and age groups, plus adding vocal effect. In this work, for the purpose of demonstration, we prototype a simple and efficient albeit less natural voice change method in Section 4. Due to its simplicity, this method is not designed to be cryptographically strong but rather an obfuscation method to prevent the user-linking attacks mentioned in our threat model (Section 2.3).

3.5 Semi-Autonomous Video Patching

At present, machine learning or deep learning based models cannot guarantee an accuracy of 100%. This reality imposes a potential risk to our system in which perfect face protection is required for privacy-sensitive users. Therefore, a final patching is needed as the last defense line to avoid the exposed faces that should be protected. There are several scenarios in which protection may fail. First, during face detection, if the system does not detect any face in a video frame where faces actually exist, then this is a missed detection. This happens mostly because a participant's face is too small (e.g., far away from the camera in the 7x7 grid layout) or blurred (due to drastic movement). This may or may not be a privacy concern depending on the sensitivity level of the face. It leads to a privacy leakage only for a sensitive face.

Another type of failure happens due to face recognition errors. As mentioned earlier, each keyframe will go through face detection/identification, and each face output from the process is classified into either the P-class or the S-class (i.e., a binary classification problem). Our system does not require 100% face recognition accuracy as long as a face is classified into the correct class (P-class or S-class). If a face of P-class is mis-classified into S-class ($P \rightarrow S$) it will not be a privacy concern although it may degrade the visual effect. If the opposite $S \rightarrow P$ happens, however, it will result in privacy leakage. The likeliness of protection failure is highly relevant to the video quality and the movement of the faces in a video. Moreover, due to the imperfection of speaker recognition algorithms, protection failures may occur in the audio channel, although the privacy risk is mostly less of an issue.

Because of these potential errors, a review and semi-automatic patching process is introduced as the final stage of our system. A simple graphic user interface is developed, which is similar to a video editor to review and process the video output from the previous processes. Two types of effort is made here to minimize the manual workload for the users. First, based on certain heuristics, our system will report the areas and the frames where the protection failures may occur. Users can review them and fix the problem or ignore them as false alarms. For instance, if Alice's face appeared (especially in a grid) in a keyframe, and there was no face detected (in the same grid) in the next frame or next few keyframes, but Alice's face is detected again in the next frame after that, then probably the system has missed the detection of Alice's face in the middle keyframe. Second, the user is allowed to patch the unprotected faces by simply (click-and-drag) selecting the unprotected face area in a frame, and our system will automatically apply object tracking to track the face area and protect it in the surrounding frames. In this way, one simple patching operation will help the protection in multiple continuous frames with the same problem.

4 Prototype Implementation

To run our proposed algorithms, we first need to prepare the input for them, which are the face templates for whitelisting or blacklisting, certain metadat of voice templates and name tags for targeted participants. Due to the page limit, we present our preparation tool in Appendix (Section 10.3). Next we focus on the implementation details of the algorithms proposed above.

4.1 Video Processing Module

Many face detection algorithms have been designed, like OpenCV with HOG cascade or Haar Cascades [9], Fast-RCNN [33], and YOLO [32]. However, they cannot fully meet our requirements. HOG and Haar have difficulty in detecting side faces [38]. Fast-RCNN and YOLO only make face detection and they require heavy GPU computation. In our work, therefore we adopt SeetaFace [29] as the solution for both detection and identification. SeetaFace has high scalability and can perform face detection and identification without using a GPU. Furthermore, we leverage the parallelism in computing to increase the efficiency of the system. When our system performs video processing tasks in parallel, due to the Global Interpreter Lock (GIL) problem [6] in Python, it is not possible to run multi-thread computation; hence, we replace it with multi-process computation, and the algorithm's logic remains unchanged.

Since name tags are displayed in screen by the conference system in a standard font, most deep learning OCR algorithms are over-qualified. After some exploration, Paddle OCR [10], a recently released open sourced OCR, is selected, which provides high-quality pre-trained models and supports recognition of over 25 language (including English, Korean, Japanese, German, French, Chinese, etc.), digit recognition, vertical text recognition, and long text recognition. With a lightweight network (weight file size only 8.6MB), Paddle OCR can achieve 57 milliseconds per image under GPU acceleration. This extreme low overhead makes it suitable for our system to process keyframes for nametag protection.

4.2 Audio Processing Module

For speaker diarization, an open source project Resemblyzer [13] is selected. As Resemblyzer is originally designed for real-time applications, we modified its real-time interactive parts to turn it into a post-processing tool for video recordings. Every second of audio is divided into 15 segments (67ms each) and each segment is matched to its speaker. Here we (implicitly) assume that an informative speech in a meeting lasts at least 67ms (1/15).

Specifically, the speaker diarization process outputs a number of sentences for a given audio file. To protect a sensitive sentence with a voice morph method, we divide its corresponding audio clip into r pieces. Each piece, represented as a one dimensional vector in time domain, is fed into a Fast Fourier Transform (FFT) process, and the frequency domain output is then multiplied by a factor f. Finally, an inverse Fast Fourier Transform is performed to generate the corresponding morphed audio piece. In the same way, the entire sensitive sentence is morphed. To prevent an attacker from recovering the original audio from a morphed audio clip, randomness is added into the morphing process by choosing r and f randomly for each sentence. Our empirical study shows that the range of r can be a real number in $[5.0, 10.0]$ and the range of f can be a real number in $[1.1, 1.5]$. An attacker may be able to separate morphed sentences correctly from the published video but it is very difficult, if not impossible, to recover the original audio clip for each individual sentence without knowing the exact values of r and f used during the morphing process. In this way, this simple obfuscation method will thwart the user-linking attack because an attacker will have difficulty to automatically match the voice signals of the same user in multiple videos that were morphed under different parameters.

4.3 Patch Processing Module

For the implementation of our semi-automatic patching system, OpenCV's selectROI function is used to enable interactive selection of target areas, that is, selecting a rectangular area with mouse. Then, the patching system leverages the Kernelized Correlation Filter (KCF) algorithm [27] to automatically track the selected face area in the adjacent frames. After the tracking is finished, the patching system will blur the identified face areas in these frames with the target missing.

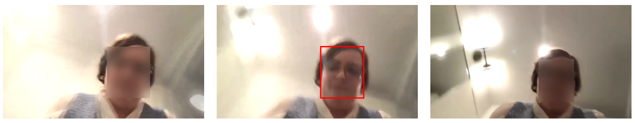


Fig. 5. Video Patching Screenshots

More specifically, after the video processing module, the system saves all keyframe results (e.g. the user's area and name in the current keyframe). These saved results will be used by the patch system to automatically check for missed protections. Figure 5 shows an example of the usage of the patch system. Here, Alice is a privacy-sensitive user. In three consecutive keyframes, her face was successfully detected, recognized and protected in keyframes 1 and 3, respectively, but there is a missed protection in keyframe 2. In this example, there is only one keyframe with missed protection. The system can modify the parameters to increase the size of the missing interval. By analyzing the face information in these consecutive keyframes, the patch system finds that Alice's face is in the same area in the 1st and 3rd keyframes, but her face suddenly disappeared in the corresponding area in the 2nd keyframe (mostly due to the failure of face detection or face recognition tools). Our patch system considers this an unlikely case and predicts the possible face area in keyframe 2 based on the face areas in keyframes 1 and keyframe 3. In this example, the predicted area is automatically labelled by our patch system in the red rectangle in keyframe 2. If such a prediction is inaccurate, the user can manually click and drag over the correct facial area. After this, our patch system will use the tracking algorithm to track the same face area in the adjacent frames and protect them accordingly.

5 Evaluation

5.1 Experiment Setup

Extensive experimental study has been conducted to evaluate the performance of our system. The testing environment is configured as follows: the CPU is Ryzen 3900x, the RAM size is 32GB and the OS is Ubuntu 20.04. The major libraries and their supported versions are Python 3.7, FFmpeg 4.2.4, OpenCV 4.4.0, and Torch 1.6. For all the examples in this section, corresponding videos (both original and after-protection) are available on our demo website [17].

As we did not find any existing VCR dataset, we searched and selected public Zoom videos from YouTube and made video clips to test our system. Table 4 in our appendix (Section 10.4) lists 10 video clips, including the three videos (the top three highlighted in blue color) that are presented with details in the main body of our evaluation and seven other types of videos (e.g., class meetings and business meetings).

5.2 Evaluation of Accuracy

In ZoomP^3, the GBP algorithm is built upon face detection and identification algorithms for accurate face protection, so we start by evaluating the accuracy of face detection and identification in the context of video conferences. While we have downloaded many Zoom recordings from YouTube for testing, they are mostly different in terms of the number of faces in the VCRs and the length of videos. Unlike the case of testing face detection/identification algorithms over static images one-by-one, measuring protection accuracy for such diverse videos is very difficult because labeling all faces frame by frame in all videos is simply infeasible. Besides, many VCRs contain very simple scenarios (few faces, mostly stationary). While testing against a lot of such VCRs can easily give us very good performance reports, it will not reflect the full potential as well as the limitations of our system. Therefore, we turn to evaluate the accuracy with a few representative videos and present the results at greater details.

We start evaluation from the detection accuracy of SeetaFace [29] in the gridview mode with different numbers of participants in one screen, gradually increasing from 1 (minimum) to 49 (maximum). While it is not an easy task to find a suitable video for this test, we were fortunate to find one in Zoom Incorporation's own YouTube channel [15]. This video stream includes 625 1080P frames in total, with participants gradually joining the meeting. Figure 6 shows a screenshot from the video (with face detection by SeetaFace). Note that this is not a video recording by Zoom itself but a recording by an external camera because the bottom menu bar and the margins are also shown. The actual video recording (by Zoom) for this video should have the grid views occupying the entire screen.

Figure 7 illustrates the average width and height of a face in the gridview mode with different number of faces. The smallest size (with 49 grids) is 39×50. The minimal detectable face size reported by Seetaface is 20×20, so the size of face is generally not the barrier for accurate face detection even in the 49-grid view case. Figure 8 further shows the maximum, median and minimum numbers of faces detected by SeetaFace vs. the actual number of faces, respectively. In this specific video, one face is not detected starting from around 20 faces. It is due to the third face in the first row shown in Fig. 6, which is small because the lady sat a bit far from the camera and the background color is close to her skin color. In the worst case, there are four undetected faces in a certain frame. Clearly, the detection accuracy depends on face size, light, and the background. As mentioned earlier, this demo video is smaller than the actual Zoom recording that should be full-screen without the menu bars and the black margins.

While the test shows that SeetaFace is not the most ideal candidate to detect very small faces, there are alternative algorithms. Another relatively new face detection algorithm called DBFace [2] is tested, which is able to detect all the 49 faces shown in Fig. 6. However, our testing also reveals that DBFace is not as good as SeetaFace in detecting large faces (e.g., when the number of faces is below nine), and it does not have face recognition capability as SeetaFace does. In practice, therefore, we can integrate these two algorithms to handle various applications scenarios.

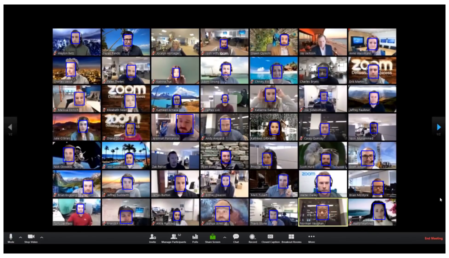


Fig. 6. A screenshot from a demo video in Zoom's Youtube channel [15] with face detection by SeetaFace

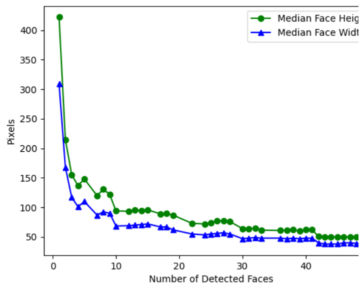


Fig. 7. Face width and height under different number of faces

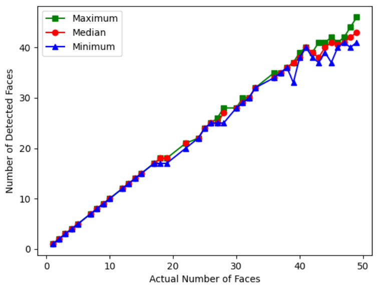
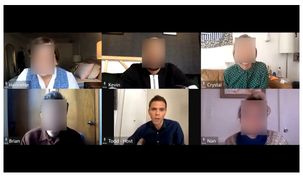


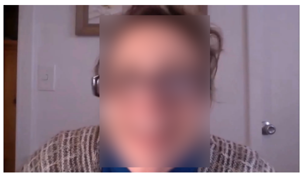
Fig. 8. Number of detected faces vs. actual number of faces

As our system performs binary face classification into P and S classes, it does not require a perfect face recognition accuracy. For the above Zoom video [15], we whitelist the host and protect all the others. The classification accuracy was 100%, neither over-protection nor under-protection was observed for this video.

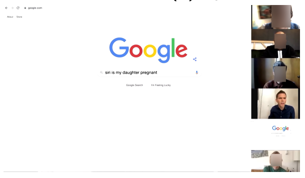
To further showcase the protection accuracy of our system against challenging scenarios, we pick the NBC Saturday Night Live (SNL) Zoom Video [16] as an example. This example video was the most challenging one we could find online. As shown in Fig. 1, this video covers switching among grid view and speaker view, shared screen view, multiple people taking turns to speak and speaking at the same time, and using virtual backgrounds and real backgrounds. It also covers some unusual scenarios such as a participant turned off the camera to show predefined avatar photos, a participant hand-held the camera and kept walking, and multiple participants showed exaggerated facial expressions. Figure 9 shows three screenshots corresponding to three view modes.



(a) Grid View Result



(b) Speaker View Result



(c) Screen Share Result

Fig. 9. Visual Effect of Facial Protection on Demo Video

As can be seen in our demo (Demo1-before-patching [17]), our system protected almost all the sensitive faces perfectly except in a very short moment (in a few frames) one participant turned her face around 90 degree to the left, which caused the face detection algorithm to fail. We then used our semi-automated system to patch it (Demo1-after-patching [17]).

5.3 Evaluation of Proposed Algorithms

Multiple video records may be used to measure the time performance of our system. However, the processing time varies along with the length of video, the number of people in a video and the frequency of scene switching differ among videos. As such, we choose to measure the performance overhead of each individual module of our system with the SNL demo video. The measurement results can be used as the basis to roughly estimate the processing times of other videos of different lengths and with different number of faces. Table 1 lists the details of this demo video.

Table 1. Demo Video Information

Video Encoding	**H.264 (AVC1)**
Video Bitrate	**1502 kb/s**
Audio Encoding	**AAC (mp4a)**
Audio Bitrate	**127 kb/s**
Resolution	**1920x1080**
FPS	**30**
Total Frames	**2832**
Total Length	**94.4 Seconds**
Scene Switching	**25 Times**
Max Number of People in One Frame	**7**
Total Number of Participants	**6**
Avatar and Virtual Background	**2**

Table 2. Time and Memory Use of All Components

Separate Video and Audio Tracks	**0.055 s**
Merge Video and Audio Tracks	**0.075 s**
Maximum Memory for Audio Processing	**2.25 GB**
Time to Read a Frame from File to Memory	**0.0018 s**
Face Detection Time for One Frame	**0.16 s**
Time to Extract Features from One Face	**0.04 s**
Face Matching with One Reference User	**0.000015 s**
Video Encoding Time with CPU	**20.44 s**
Video Encoding Time with GPU	**10.92 s**
Max Memory for Video Processing per Process	**1.13 GB**
KCF Tracking Time for One Frame	**0.064 s**
Audio Diarization	**6.32 s**
Speaker Protection via Voice Morphing	**1.04 s**

5.3.1 Overall Performance

Table 2 shows the median time delay and maximum memory uses of processing steps in our system. The first two items are the time costs, 0.055 and 0.075 seconds, for separating and merging video and audio tracks with FFmpeg, respectively. They are negligible compared to the time for the overall system, which is 62.5 seconds as shown in Table 3. The time for face detection (with Seetaface) on one video frame is 0.16 seconds and it is mostly independent of the actual number of faces in the frame. Once the system detects a face, it extracts facial features (0.04 second per face) and then match against that of each participant for face recognition (negligible time). The most time consuming operation is video encoding (20.44 seconds in CPU), which, taking all the processed video frames as input, encodes them into the video track and finally writes it out to the disk. This operation can also be done in GPU, with about 50% time cost (10.92 seconds). Finally, audio diarization is another time consuming process, which takes 6.32 seconds. The voice morphing process takes about one second. Note that the video and audio tracks are processed in parallel, so the overall time is determined by the video processing time.

Table 3 highlights the time costs of major operations and the whole system, which demonstrate efficiency of our algorithms in terms of processing time. Without our optimization algorithms (i.e., frame-by-frame processing), it took 800 seconds to process the demo video, which is about 8.5 times of the original video length. With our GBP algorithm, at group sizes $L = 5$ and $L = 10$, the time costs are reduced by a factor of 4.5 and 7.7, respectively. The time cost is sublinear to the group size. Furthermore, applying our parallel algorithms over the multi-core processor reduces the processing time significantly. For instance, it takes only 62.5 seconds with $L = 5$ on a quad core processor, which is 2/3 of the video length.

To show the scalability of our system w.r.t. the video length, we replicate 38 copies of the 94.4 second long demo video to extend it into one hour long video. It would be more difficult to see the clear relationship with a different video. Our measurement shows that the time costs are about 38 times of those in Table 3 for the original video, and the total time is about 39 mins with $L = 5$ and Quad core, which indicates the linear relationship. We believe this time cost is reasonable for video processing.

In real world applications, the number of faces and the frequency of scene switching vary. However, a rough estimation on the time cost is obtained based on the data in Table 3. The main time cost that is linear to the number of faces is that for extracting features from one face. In the worst case if this demo video were in the 7×7 gridview mode (49 faces) at all times, we may roughly estimate the total time would be around 328 seconds. On the other hand, in the best case if there were only one face at all times, the total time cost would be around 48 seconds. A server with more CPU cores or a larger group size L can be applied to reduce the time cost. Because our GBP algorithm conservatively treats all the non-keyframes between a sensitive keyframe and a non-sensitive keyframe as sensitive, this over-protection could increase the chance of public participants being protected at a larger group size, slightly losing the visual friendliness.

5.3.2 Audio Processing Module

For our demo video shown in Table 2, audio diarization took a total of 6.32 seconds, and voice processing took 1.04 seconds, which includes voice morphing and writing the audio file to the hard drive. One can see and hear the effect of privacy protection in our demo video [17].

5.3.3 Name Tag Protection

Another online zoom meeting VCR with name tags is chosen to test the effectiveness and efficiency of our Paddle-OCR based name tag protection algorithm. As shown in Fig. 10, the top-left user and the bottom user are selected to be whitelisted. This result validates our algorithm that successfully blur all the others' name tags as desired. The corresponding demo video is also

Table 3. Time Costs of Major Video Processing Algorithms (Demo Video 94.4 Seconds, 2832 Frames, L is group size in our GBP algorithm)

	Frame by Frame	GBP ($L = 5$)	GBP ($L = 10$)	GBP ($L = 5$, Dual Core)	GBP ($L = 5$, Quad)
Read All Frames	5.22 s	5.15 s	5.26 s	2.61 s	1.3 s
Face Detection	459.62 s	91.82 s	45.92 s	46.73 s	24.05 s
Face Recognition	314.72 s	61.53 s	31.47 s	31.62 s	16.58 s
CPU Video Encoding	20.44 s	20.35 s	20.95 s	20.48 s	20.57 s
Total Time	800 s	178.85 s	103.6 s	101.44 s	62.5 s

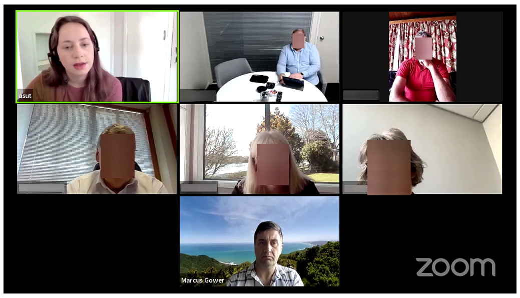


Fig. 10. Name Tag Protection (District Promotions Committee meeting [3] in Waipa District Council, New Zealand).

available [17]. Paddle OCR can achieve 57 milliseconds per video frame under GPU acceleration. The extracted texts are then matched against either whitelisted or blacklisted reference names based on our heuristic algorithm. As name tag protection is done together with our GBP algorithm in a single round instead of a separate round, the time overhead is small compared to that of the video processing algorithms.

5.3.4 Patch Processing Module

Due to the imperfection of machine learning based face detection and face recognition algorithms, for videos recorded in challenging situations (e.g., with small, unclear, moving faces), a final patching is applied to protect missed faces. Our semi-automatic patching system leverages the KCF algorithm to track a user selected facial area in adjacent frames to patch the missed faces, if any. Table 2 shows that the time for tracking a facial area in one frame is 0.064 second, which is small, considering that missed protection is not common for most conference recordings. We did not evaluate the time for manually selecting a missed facial area as it depends on the familiarity of the user with our system.

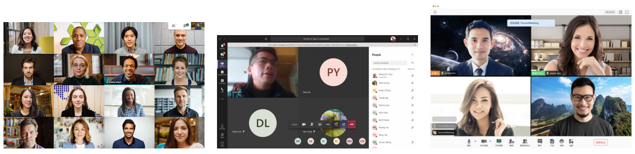
5.3.5 Overall Time Complexity

Here we report the overall time we spent to process the SNL video. In the input preparation phase, it took us less than 10 seconds for selecting and saving the face template for the whitelisted speaker, and less than 10 seconds to select an audio range for him. Since the SNL video does not have name tags, we did not spend time on generating the nametag file for it (we spent a few seconds to input two whitelisted names for the other video shown in Figure 10). As shown above, the total time for our proposed protection algorithms is 62.5 seconds for the SNL video clip. Finally, we patched the video twice and it took us about 20 seconds.

To sum up, if assuming that we already knew which faces and which audio segments to choose as templates, and, if necessary, which frames to patch, and also assuming there were no break time (gap) between the operations, the total time was about 2 minutes for the SNL video. We acknowledge that the actual time will be more because of other subjective factors such as decision making and familiarity with our tools. Nevertheless, our protection algorithms normally take the most time, especially for a long video.

6 Discussions and Limitations

Applicability: Our ZoomP^3 system is compatible with most online conferencing systems, such as Google Meeting, Microsoft Teams, Tencent Meeting, as shown in



(a) Google Meet (b) Microsoft Teams (c) Tencent Meeting

Fig. 11. Online Conferencing Application Screenshots from Google Search

Fig. 11. These systems use the same video encoding method (H.264 AVC1) as Zoom, and they also have speaker view mode and grid view mode, so our system can be directly applied to them for visual privacy. Indeed, because our system uses the FFmpeg decoder, not only can it process online video recordings, but also handle videos recorded by 3rd party software.

Scope and Limitations of Visual Protection: In a VCR, sensitive faces may be from actual participants, from virtual backgrounds or from a photo in the shared screen mode. Our system protects all sensitive faces and voice in a VCR regardless of scenarios because the face detection/recognition algorithms do not differentiate the scenarios. Specifically, in the whitelisting mode, all other faces are considered sensitive (including those faces in the backgrounds or in the shared screens), and they are all blocked. The blocking of faces in the background happened in our SNL demo. In the blacklisting mode, the other faces are not blocked. In this case, if the background contains a sensitive face, one may blacklist it by choosing a snapshot of the face area as the input to our system.

Currently our system does not handle other sensitive texts in shared screens. If such information is rare, a user may protect it with the function of our patching system. That is, a user may select the region containing the sensitive texts and object tracking will help locate the same region in the adjacent frames. However, if there are a lot of sensitive texts, we may use our OCR-based name tag protection method. The idea is to treat all sensitive texts as blacklisted name tags. When processing each frame, they are compared against all the OCR-extracted texts. The bounding box of any matched texts can be blurred accordingly.

Scope and Limitations of Audio Protection: In our demo implementation, we employ the simple FFT-based voice morphing method for voice protection against user-linking attacks that leverage frequency domain features. However, speakers may be linked by their accents/dialects or lexical/syntactical features. Therefore, our simple voice morphing approach cannot handle such attacks. Moreover, in our current implementation, the voice morphing effect is not natural. More advanced methods (e.g., Morphvox Pro [7]) exist to morph voice naturally and they can also defeat the above attacks by changing the voice of each sensitive user into that of a random character, although they are more costly. Note that voice morphing alone cannot address the privacy leakage problem caused by the content of speech, which needs manual identification and removal. For this reason, if user-friendliness is not a key factor here, simply muting all sensitive speakers is a better option.

Currently, our system does not examine the content of speech. As a result, if a non-sensitive participant says sensitive information, one may manually mute that part of voice. An automated tool may be designed to handle such issues with the text input of sensitive information. The high-level idea is to first make the audio-to-text transcription of the audio track (indeed Zoom recently provided the recording option for audio transcript, which will produce a separate audio transcript file for the audio track), then locate the sensitive text, and finally mute the voice at the same position in the audio. The requirement is that the audio-to-text transcription tool has very good accuracy.

We notice that for conference tools like Zoom, once enabled in the setting of recording, the chat log and audio transcript are saved in two separate files other than the video file. To avoid any privacy leakage from the chat log and live transcript, the user should publish the video file only.

7 Related Work

Video Conferencing Security and Privacy: A recent research showed that it was relatively easy to collect a large number of public conference videos and extract from them personal information such as facial features, age, gender, names and usernames of the participants [28]. A data-driven analysis of calls for zoombombing attacks on social media indicate that the vast majority of calls for zoombombing are not made by attackers bruteforcing their meeting ID, but rather by insiders who have legitimate access to the meetings. Drawing from literature on crisis communication and the shared user experiences in the Zoom CEO's blog post, Young [41] showed that Zoom's privacy crisis was not only caused by technical issues but also the company's underestimation on users' privacy expectation.

Surveillance Cameras Privacy: A system named *Visor* [31] was proposed that focuses on protecting the surveillance camera video stream when the cloud service is compromised. Another system was proposed [39] for live video analytics. Data from surveillance cameras can be used for environmental analysis. If uploading the raw data directly to the cloud for analysis, there will be privacy issues. Their proposed solution is to perform facial recognition, block the person and then upload it to the cloud for analysis. A Privacy-preserving

Surveillance as an Edge service (PriSE) method [23, 24] takes advantages of a hybrid architecture comprising a lightweight foreground object scanner and a video protection scheme that operates on edge cameras and fog/cloud-based models to detect privacy attributes like windows, faces, and perpetrators. These approaches are all for surveillance camera privacy and only consider image privacy, which does not consider voice privacy. It is different from our system, which protects both image and voice privacy of an online video conferencing.

Privacy Protection: To our knowledge, existing commercial tools only protect either faces or voice. There is no integrated tool for the protection of faces, voice and name tags altogether. Well-known tools such as Adobe Premiere-Pro and YouTube Face Blurring allow a user to manually choose a face to blur with different masks (oval, rectangle or other shapes) and apply an object tracking algorithm for continuous protection of the same face in the video. However, somewhat similar to our patching mechanism, they provide local protection, because they rely on object tracking to identify the same face in adjacent frames. The protection will break if the face is temporally blocked and reappears later. The user needs repeat the same process for the same lost face.

The visual protection function of our tool is different from these tools in two major ways. First, it can support both blacklist and whitelist modes, which are very useful for protecting VCRs with many participants. Second, it can protect faces throughout the entire video despite the switching of modes among grid view, shared screen view and speaker view modes, which would easily break object tracking. The reason is that our tool is specially designed for VCRs (although it should also work for other types of videos) and it is built upon different algorithms and tools. On the other hand, we acknowledge that these tools can be used for local patching as needed in our system and provide more features on blurring effects (e.g., various shapes of blurring) than that offered by our prototype implementation. As for voice protection, people may use commercial audio editing tools such as Adobe Audition, Audacity or smartphone apps to manually edit an audio file, e.g., cut or split. However, none of them does automatic speaker-based voice processing. Built upon speaker diarization tool Resemblyzer, our tool is able to protect an individual speaker's voice throughout the entire video. As far as we know, there is no existing tool for name tag protection.

There is also some work from the research community. A system is proposed to recognize human activity under an extreme low resolution video [35]. Based on GAN [26], researchers proposed a system to generate pixel-level modifications to anonymize faces [34]. Specifically, the privacy for minors attracted attention from the community and lightweight privacy-preserving methods are customized for edge computing environments [25]. In addition, a deep learning audio-visual model is suggested for isolating a single speech signal from a mixture of sounds such as other voices [21]. We may adopt it for more accurate audio protection at higher time cost.

Audio-Visual Biometric Authentication Face and voice biometrics are the common technology used to verify a person's identity using his/her unique facial and vocal attributes, respectively. In addition to many industrial products, many new techniques have been proposed by the research community over the years. Comprehensive surveys can be found in [30, 40]. In our research, we integrate the state-of-the-art technology on face detection, face identification, speaker diarization seamlessly to achieve our research goal of building an efficient privacy-preserving VCR publishing system. The efficiency and accuracy of our system will improve with the advance of these research fields.

8 Conclusions

In this work, we introduced ZoomP^3, a practical privacy-preserving publishing system for protecting both video and audio of privacy-sensitive participants in online meeting VCRs. The videos can be reused or shared publicly without infringing the privacy rights of such users. Besides leveraging and integrating multiple state-of-the-art computer vision and audio processing tools seamlessly into our system, a number of optimization algorithms are proposed to improve the scalability of the system, making it possible to protect the privacy of long video conferences. To the best of our knowledge, ZoomP^3 system is the first of this kind to protect both video and audio privacy of online conference videos. It may be provided as an online service, e.g., by Zoom, or by large organizations such as universities, research institutes and government sectors.

9 Acknowledgements

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

[1] Covid-19 changed lives. https://foreignpolicy.com/2020/05/07/lockdown-covid-19-changed-lives-around-the-world/. Accessed November 1, 2020.

[2] Dbface-a real-time, single-stage detector for face detection. https://github.com/dlunion/DBFace. Accessed February 18, 2021.

[3] Extraordinary district promotions committee - zoom invite. https://www.youtube.com/watch?app=desktop&v=EiDcZoEcRRk.

[4] Ffmpeg. https://ffmpeg.org. Accessed November 1, 2020.

[5] Find zoom meeting id. https://www.theverge.com/2020/4/2/21206061/zoom-meeting-id-zwardial-automated-tool. Accessed November 1, 2020.

[6] Gil. https://en.wikipedia.org/wiki/Global_interpreter_lock. Accessed November 1, 2020.

[7] Morphvox pro. https://screamingbee.com/morphvox-voice-changer. Accessed March 11, 2020.

[8] Moviepy. https://zulko.github.io/moviepy. Accessed November 1, 2020.

[9] Opencv https://opencv.org/ accessed nov 1,2020.

[10] Paddle ocr. https://github.com/PaddlePaddle/PaddleOCR. Accessed November 1, 2020.

[11] phash: The open source perceptual hash library. https://www.phash.org. Accessed November 26, 2017.

[12] Pydub 0.25.1 website. https://pypi.org/project/pydub/.

[13] Resemblyzer. https://github.com/resemble-ai/Resemblyzer. Accessed November 1, 2020.

[14] Speaker diarisation. https://en.wikipedia.org/wiki/Speaker_diarisation. Accessed November 1, 2020.

[15] Zoom 7x7 view. https://www.youtube.com/watch?v=BP2VdBZtZ6Q. Accessed February 18, 2020.

[16] Zoom call - snl. https://www.youtube.com/watch?v=3byTN8NTCkc. Accessed February 18, 2020.

[17] Zoom privacy demo site. https://github.com/paperdemo888/zoom_privacy.

[18] Zoom recording layout. https://support.zoom.us/hc/en-us/articles/360025561091-Recording-layouts#h_70f4ac90-47f8-4aef-a6b2-370184eb4651. Accessed Feb 8, 2021.

[19] Zoom security and privacy. https://www.tomsguide.com/news/zoom-security-privacy-woes. Accessed December 17, 2020.

[20] zoombombing. https://www.nytimes.com/2020/03/20/style/zoombombing-zoom-trolling.html. Accessed November 1, 2020.

[21] Ephrat, A., Mosseri, I., Lang, O., Dekel, T., Wilson, K., Hassidim, A., Freeman, W. T., and Rubinstein, M. Looking to listen at the cocktail party: a speaker-independent audio-visual model for speech separation. *ACM Trans. Graph. 37*, 4 (2018), 112:1–112:11.

[22] Fini, E., and Brutti, A. Supervised online diarization with sample mean loss for multi-domain data. *arXiv preprint arXiv:1911.01266* (2019).

[23] Fitwi, A., Chen, Y., and Zhu, S. Prise: Slenderized privacy-preserving surveillance as an edge service. In *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)* (2020), IEEE, pp. 125–134.

[24] Fitwi, A., Chen, Y., Zhu, S., Blasch, E., and Chen, G. Privacy-preserving surveillance as an edge service based on lightweight video protection schemes using face de-identification and window masking. *Electronics 10*, 3 (2021), 236.

[25] Fitwi, A., Yuan, M., Nikouei, S. Y., and Chen, Y. Minor privacy protection by real-time children identification and face scrambling at the edge. *EAI Endorsed Transactions on Security and Safety 18*, 3 (2020).

[26] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[27] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence 37*, 3 (2014), 583–596.

[28] Kagan, D., Alpert, G. F., and Fire, M. Zooming into video conferencing privacy and security threats. *arXiv preprint arXiv:2007.01059* (2020).

[29] Liu, X., Kan, M., Wu, W., Shan, S., and Chen, X. Viplfacenet: an open source deep face recognition sdk. *Frontiers of Computer Science 11*, 2 (2017), 208–218.

[30] Mandalapu, H., Reddy P N, A., Ramachandra, R., Rao, K. S., Mitra, P., Prasanna, S. R. M., and Busch, C. Audio-visual biometric recognition and presentation attack detection: A comprehensive survey. *IEEE Access 9* (2021), 37431–37455.

[31] Poddar, R., Ananthanarayanan, G., Setty, S., Volos, S., and Popa, R. A. Visor: Privacy-preserving video analytics as a cloud service. In *29th USENIX Security Symposium (USENIX Security 20)* (Aug. 2020), USENIX Association, pp. 1039–1056.

[32] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *IEEE CVPR* (2016).

[33] Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (2015).

[34] Ren, Z., Lee, Y. J., and Ryoo, M. S. Learning to anonymize faces for privacy preserving action detection. In *Proceedings of the European Conference on Computer Vision (ECCV)* (September 2018).

[35] Ryoo, M. S., Rothrock, B., Fleming, C., and Yang, H. J. Privacy-preserving human activity recognition from extreme low resolution. *arXiv preprint arXiv:1604.03196* (2016).

[36] Sabra, M., Maiti, A., and Jadliwala, M. Zoom on the keystrokes: Exploiting video calls for keystroke inference attacks, 2020.

[37] Speciale, P., Schonberger, J. L., Kang, S. B., Sinha, S. N., and Pollefeys, M. Privacy preserving image-based localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019).

[38] Sun, Y., Chen, S., Zhu, S., and Chen, Y. iryp: a purely edge-based visual privacy-respecting system for mobile cameras. In *Proceedings of the 13th ACM Conference on Se-*

curity and Privacy in Wireless and Mobile Networks (2020), pp. 195–206.

[39] Wang, J., Amos, B., Das, A., Pillai, P., Sadeh, N., and Satyanarayanan, M. A scalable and privacy-aware iot service for live video analytics. In *Proceedings of the 8th ACM on Multimedia Systems Conference* (2017), pp. 38–49.

[40] Wang, M., and Deng, W. Deep face recognition: A survey. *Neurocomputing 429* (Mar 2021), 215–244.

[41] Young, S. Zoombombing your toddler: User experience and the communication of zoom's privacy crisis. *Journal of Business and Technical Communication* (2020), 1050651920959201.

[42] Zhang, A., Wang, Q., Zhu, Z., Paisley, J., and Wang, C. Fully supervised speaker diarization. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 6301–6305.

10 Appendix

10.1 A Demo Example of our GBP Algorithm

Figure 12 shows a simple example with six frames in total and we choose $L = 3$. Since the first frame of the first group is always a keyframe, we have three keyframes at positions 1, 3, and 6, respectively. Here Alice is a sensitive participant and Bob is a public participant. GBP processes the first frame with face detection and identification and knows that it is a public participant, so there is no need for face protection. The second frame is not a keyframe, so there is no need to process the frame at this moment. The third frame is a keyframe with a sensitive participant Alice. According to our protection rule R1, no matter whose face it is in frame #2, #4, #5, they will all be protected. Frame #6 is a keyframe with Bob, so there is no need to protect his face. In the end, all the faces in frames 2-5 are protected.

10.2 Optimization Algorithms for Video Processing

10.2.1 Binary Search Backward Processing

The above GBP algorithm is very conservative by assuming *all* the $L-1$ non-keyframes between two keyframes contain S instead of P whenever one of the two keyframes contains S. For a small L (e.g., $L = 6$, corresponding to 0.2 second for 30 FPS videos), the false positives (i.e., over-protection) caused by treating P as S will not introduce noticeable degradation of visual effect, but the saving on computation is still not significant enough as the number of keyframes in a long video could still be large. On the other hand, a larger L will save more on computation because there will be less keyframes. For example, when $L = 30$, which corresponds to 1 second for 30 FPS videos, only 1/30 of frames will be processed as keyframes. For a video with relative stable views (e.g., a speaker in the speaker view keeps talking for tens of seconds or more), most of the keyframes do not change (i.e., $S \rightarrow S$, or $P \rightarrow P$ between two adjacent keyframes), a larger L saves more computation without introducing false positives in most of the time. Now, when a change occurs between two adjacent keyframes (e.g., $P \rightarrow S$ or $S \rightarrow P$), it will be more visually friendly with less "smear effect" by treating the $L-1$ non-keyframes in between more accurately instead of blindly treating all of them as S frames for a large L.

An accurate protection requires to detect the exact non-keyframe between two adjacent keyframes where speaker changes happen. We call this process *speaker change detection*. A simple way of implementing speaker change detection is to process frame-by-frame with face detection/identification for all the $L-1$ non-keyframes. Clearly, this is inefficient. Therefore, a binary-search-like processing (BSP) algorithm is adopted in our system. The search continues recursively in the half whose leftmost and rightmost frames contain one P frame and one S frame. In this way, additional $\lceil log_2(L-1)\rceil$ frames are processed for accurate protection. Certainly, a further tradeoff can be made by stopping the BSP algorithm when the interval of the current half falls below a threshold (e.g., 5).

10.2.2 Perceptual Hash based Speaker Change Detection

To reduce the computational cost with improved accuracy in the GBP algorithm, an optimization method is proposed for speaker view mode, which is a more efficient yet accurate way to solve the speaker change detection problem. The rationale behind the new method is that in the speaker view mode, only when the speaker is changed, there will be a dramatic change in two adjacent frames due to differences in faces and backgrounds. For the same speaker, even though his/her head may move, the overall visual change in adjacent frames should be much smaller than that caused by the switching of speakers. As such, as long as we detect a significant visual change between two adjacent frames, we can report it as a speaker changing frame.

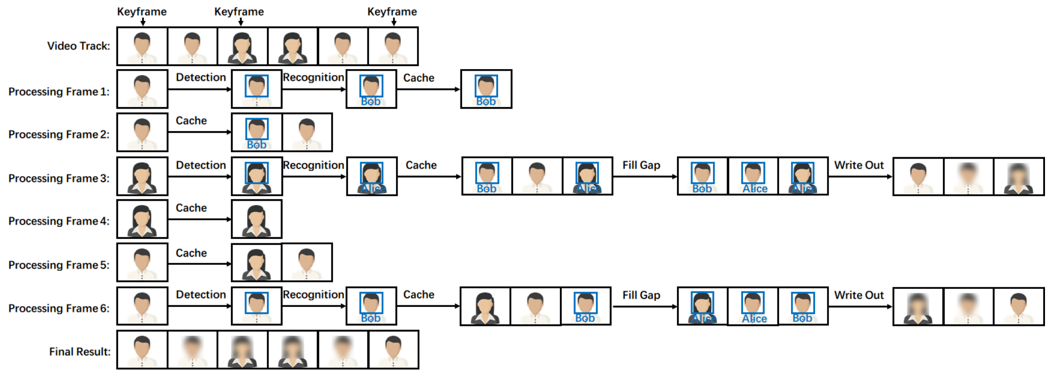


Fig. 12. Example of Greedy Backward Processing ($L = 3$)

Table 4. 10 Zoom Videos Tested by Our System

Video Clip Title	Video URL
Zoom 7x7 View	https://www.youtube.com/watch?v=BP2VdBZtZ6Q
Zoom Call - SNL	https://www.youtube.com/watch?v=3byTN8NTCkc
Extraordinary District Promotions Committee - Zoom Invite	https://www.youtube.com/watch?v=EiDcZoEcRRk
Zoom Catch-Up - SNL	https://www.youtube.com/watch?v=vdqsMY5Z8E8
Extraordinary Te Awamutu Community Board Meeting - Annual Plan Submission	https://www.youtube.com/watch?v=Nt_1PgTLxTY
Ask Eric: Zoom Platform Security Updates	https://www.youtube.com/watch?v=SHw9QcK5R_g
Calculus Class Zoom Meeting 4/8/2020	https://www.youtube.com/watch?v=mjWxYI4oTTw
Demo of Online Meeting via Zoom with Participants from Around the World	https://www.youtube.com/watch?v=261OCjeg9GI
Zoom Class Meeting Downing Soc 220-001 1/21/2021	https://www.youtube.com/watch?v=gWwdGGG4Pkc
Finance Meeting for April 20, 2020	https://www.youtube.com/watch?v=Fa-hbpZx6OQ

To detect visual change efficiently, we use a perceptual hash function called pHash [11], which is an image hash that outputs 64 bits given an image as the input. The difference between a perceptual hash and a cryptographic hash is that a perceptual hash can tolerant small changes in the input. If an input image is changed slightly, its perceptual hash value may be the same or only change by a few bits. In contrast, a single bit change in the input to a cryptographic hash can cause random changes in the output due to an avalanche effect. Therefore, if the hamming distance between the perceptual hash values of two adjacent non-keyframes is greater than a certain threshold, a speaker change case is detected.

An experiment has been conducted to verify this rationale. We record two zoom videos in speaker view mode, both of which have 123 frames and switch from one speaker to another speaker in the middle. The main difference between the two videos is the virtual backgrounds used by two speakers. In one video (Video #1), both speakers use the same background (solid white) and in the other video (Video #2) they use their own image virtual background (a Golden Bridge view vs. a mountain-forest-lake view). We wanted to estimate the visual differences caused by speakers only and by both the speaker and the background, respectively.

Figure 13 depicts the changes of perceptual hashes between a frame and its previous frame (i.e., the hamming distance between the pHash values of two adjacent frames) for both videos. The green line and the red line represent the cases of video #1 and video #2, respectively. Each line has a spike (over 25), which corresponds to the moment of speaker switching, while the change due to the movement of head and visible body is stable and much lower (below 13). Therefore, based on the experimental study a threshold of 20 is selected in our system to detect speaker changes.

Since perceptual hashing is extremely efficient, it can be employed for frame-by-frame processing to enhance the accuracy of our GBP algorithm. It is worth noting that our perceptual hash based speaker change detection is best suitable to speaker view mode, because in gridview mode, it is in general difficult to determine the exact position of the window view each participant

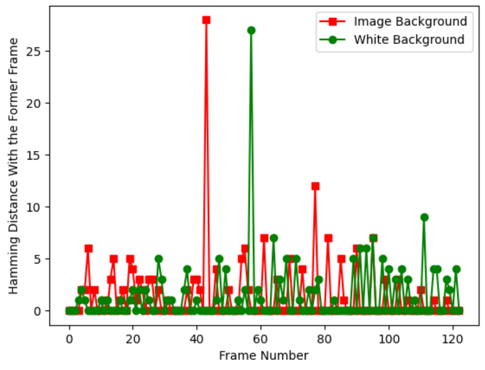


Fig. 13. Hamming distance between each frame and its previous frame.

is located in due to the dynamics of the views in the screen.

10.3 Our Tool for Input Preparation

Before running our algorithms, we first need to produce the face templates, certain metadata of voice templates, and name tags for the participants to be whitelisted or blacklisted in a VCR. For users' convenience, we developed a preparation tool for video processing, audio processing and name tag input. The tool plays a video with basic functions such as play, pause, drag bar (mainly implemented with an OpenCV api "videoCapture.read()" to read frame-by-frame). After locating a frame with a target face, we select with mouse the rectangular area surrounding the face (implemented by OpenCV's selectROI function) and the selected facial area is automatically saved into a PNG image file, which is the face template file for this user.

The audio processing part of our tool is implemented by an open source library named Pydub [12]. It enables us to manually select the range (start and end) of a short voice segment (e.g., a few seconds of Bob's speech as his voice template) for each of the speakers to be explicitly whitelisted or blacklisted. It writes all the range information (start, end) into a single template file, which will be the input to the speaker diarization tool Resemblyzer. Finally, the tool provides a simple UI (implemented by Python input() function) for users to input the names to be blacklisted or whitelisted, and the result is written into a nametag file for the following OCR-based name tag protection.

10.4 Videos Tested by ZoomP^3

Table 4 lists 10 videos tested by our system, which cover a variety of application scenarios as well as extreme demo videos.