Security-Preserving Live 3D Video Surveillance

Zhongze Tang
Rutgers University
Piscataway, NJ, USA
zhongze.tang@rutgers.edu

Huy Phan
Rutgers University
Piscataway, NJ, USA
huy.phan@rutgers.edu

Xianglong Feng
Miami University
Oxford, OH, USA
fengx17@miamioh.edu

Bo Yuan
Rutgers University
Piscataway, NJ, USA
bo.yuan@soe.rutgers.edu

Yao Liu
Rutgers University
Piscataway, NJ, USA
yao.liu@rutgers.edu

Sheng Wei
Rutgers University
Piscataway, NJ, USA
sheng.wei@rutgers.edu

ABSTRACT

3D video surveillance has become the new trend in security monitoring with the popularity of 3D depth cameras in the consumer market. While enabling more fruitful surveillance features, the finer-grained 3D videos being captured would raise new security concerns that have not been addressed by existing research. This paper explores the security implications of live 3D surveillance videos in triggering biometrics-related attacks, such as face ID spoofing. We demonstrate that the state-of-the-art face authentication systems can be effectively compromised by the 3D face models presented in the surveillance video. Then, to defend against such face spoofing attacks, we propose to proactively and benignly inject adversarial perturbations to the surveillance video in real time, prior to the exposure to potential adversaries. Such dynamically generated perturbations can prevent the face models from being exploited to bypass deep learning-based face authentications while maintaining the required quality and functionality of the 3D video surveillance. We evaluate the proposed perturbation generation approach on both an RGB-D dataset and a 3D video dataset, which justifies its effective security protection, low quality degradation, and real-time performance.

CCS CONCEPTS

• **Information systems → Multimedia information systems**; • **Security and privacy → Systems security**.

KEYWORDS

3D surveillance, face authentication, live streaming

ACM Reference Format:
Zhongze Tang, Huy Phan, Xianglong Feng, Bo Yuan, Yao Liu, and Sheng Wei. 2023. Security-Preserving Live 3D Video Surveillance. In *Proceedings of the 14th ACM Multimedia Systems Conference (MMSys '23), June 7–10, 2023, Vancouver, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3587819.3590975

1 INTRODUCTION

With the introduction of 3D depth cameras in the consumer market [11, 13, 15], 3D video surveillance has emerged as a new and advanced means of finer-grained monitoring for the physical world events than the traditional 2D video surveillance [14]. In particular, the addition of depth information in the camera view provides the opportunities of applying video surveillance in many traditional and new fields of businesses, such as security surveillance [16], logistics [5], transportation [7], and retail [19]. For example, in security surveillance, 3D cameras can better differentiate suspicious subjects in the surroundings from normal ones by leveraging the depth information [16]. In retail business, 3D cameras can accurately count the number of people in multiple zones, monitor the dwell time, and form heat-maps to help store managers with decision making [19]. With its growing popularity and strong potential, 3D video surveillance has become one of the major application domains boosting the demand of 3D cameras, which was valued at USD 1.92 billion in 2018 and projected to expand at a compound annual growth rate of 35.8% from 2019 to 2025 [11].

However, the other side of the coin for the 3D video surveillance involves significant challenges in multiple dimensions, such as the bandwidth and performance for delivering the 3D content [52], as well as the security and privacy concerns around the content of 3D videos [63]. To date, the community has mainly focused on the bandwidth and performance challenges with many effective content delivery and rendering solutions [28, 52, 66]. However, the security and privacy aspect of 3D surveillance has not been maturely studied, which leaves a gap between the readiness of the 3D surveillance equipment/technology and the immaturity in utilizing it in practice due to the ever increasing security/privacy concerns in video surveillance [8, 32, 74].

Although the community has developed security techniques to protect the traditional 2D surveillance videos [77], the extra depth information in the 3D surveillance video poses unique security challenges that did not exist in the 2D counterpart. For example, human face is probably the most security sensitive object in a surveillance video that must be protected. Recent multimedia security research has shown that, given the face images extracted from video frames, the adversaries may conduct face ID spoofing attack to get recognized as the victim user in front of a deep learning-based face authentication system [26, 41, 72]. Then, to defend against such spoofing attack, the state-of-the-art face authentication systems have evolved to require depth information [25, 45, 46, 78] and liveness detection [10, 40, 69] to differentiate a real human face from the one extracted from a video. Although these defense methods can successfully mitigate the security concerns in 2D videos, the 3D surveillance video presents both depth and liveness, leading to successful face ID spoofing attacks.

To address the security issues involved in 3D video surveillance, we propose to inject perturbations to the sensitive 3D objects (e.g., human faces) in the surveillance video to prevent potential threats (e.g., face spoofing attack). In a nutshell, from the security perspective, the injected perturbations are expected to fail the face authentication attempts by misleading the authentication workflow or algorithm to generate erroneous results; therefore, there would be no security impact posed to the victim user whose facial information may have been intercepted by the adversary during the transmission and display of the surveillance video. On the other hand, while generating the perturbations, we must maintain the original functionality and performance of the 3D video surveillance, leading to the following requirements: (1) **Quality requirement**: The injected perturbation should not blur the video to the extent that compromises the original purpose of the video surveillance. For example, in the case of 3D security surveillance cameras [16], the subjects in the RGB frames should still be sufficiently visible to a human inspector for the purpose of surveillance and forensics. Also, the depth channel of the surveillance video should not be significantly altered for the purpose of identifying the distances of the subjects. (2) **Real-time performance requirement**: The perturbation generation should not incur significant delay to impact the real-time processing and live delivery of the surveillance video.

Taking the security goal and the quality/performance requirements into consideration, we develop a real-time perturbation generation mechanism for live 3D surveillance video, namely *LiVSec*. To accomplish the security goal while minimizing the quality impact, *LiVSec* makes a benign and novel use of adversarial attack for deep neural networks (DNNs) [24, 35, 43], which is originally a malicious attack method but being adopted as a benign protection scheme for 3D surveillance videos in this work. Under this context, the benign perturbation generated by *LiVSec* would fail the target DNN (i.e., the malicious face authentication/spoofing attempt) but maintain the original visual quality of the original object. More importantly, to meet the performance requirement, *LiVSec* develops a real-time adversarial perturbation generator for 3D face models to perform fast perturbation generation (at the magnitude of milliseconds per frame), which overcomes the limitations of slow generation (at the magnitude of seconds per frame) in the classical adversarial attacks [24, 35, 43]. We implement the proposed *LiVSec* approach in a live 3D video streaming system and evaluate its security, quality, and performance using two representative datasets including RGB-D images and 3D video. The evaluation results demonstrate the effectiveness of *LiVSec* in protecting live 3D video surveillance.

2 BACKGROUND

2.1 Live 3D Video Surveillance

The live 3D video surveillance system adopts 3D cameras and fast network communications to deliver the immersive view of the targets under surveillance. It has become feasible with the popularity of 3D cameras in the consumer market, and the industry foresees a fast growing market trend incorporating 3D video surveillance as one of the major use cases of 3D cameras [11]. Without loss of generality, we consider an end-to-end surveillance video system as demonstrated in Figure 1. The system involves three major components that jointly achieve the goal of live 3D surveillance: (1) the video source, which is a public security camera that captures the live video with both the RGB and depth information, potentially including the faces of various people who are concerned about security; (2) the network, which delivers the live surveillance video from the source to the monitor in a timely manner; and (3) the monitor, which receives and displays the live 3D video at the receiver end for the purpose of surveillance or further analysis.

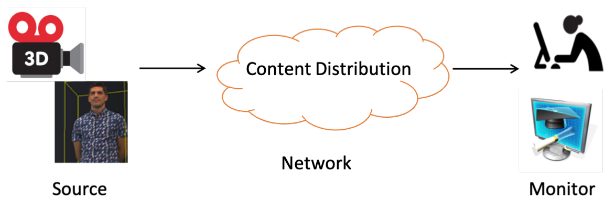

Figure 1: Overview of 3D video surveillance system.

The purpose of the 3D video surveillance system is to capture the suspicious behavior (e.g., moving subjects) accurately and in a timely manner. To achieve this goal, the state-of-the-art 3D security cameras [16] typically involve a 2-step workflow: (1) The captured 3D video is processed by a computer vision or machine learning model to automatically detect the target behavior (e.g., motion). This step does not have human involvement, and the depth information in the video plays an important role in achieving more accurate detections [16]. (2) If suspicious behavior is detected in step (1), the surveillance system would send an alert followed by streaming the surveillance video to the remote user. The user can then visually inspect the video and verify the suspicious behavior or infer more details that were not reported by the system automatically. In this step, the user only inspects the RGB portions of the frames without depth information.

In a 3D video surveillance system, there are two important properties that must be preserved to achieve fruitful surveillance. First, the processing of the source video must be *real-time* for the monitor to capture the dynamic situations at the source. Second, the quality of the video perceived at the monitor must be sufficient for user inspections, which is not as stringent as the general quality of experience requirement in entertainment videos [20], but the noise or perturbation involved should not mislead the visual perception of critical objects in the video. Our design of the *LiVSec* approach achieves these two important properties by injecting perturbations to the video frames in real time, which leads to the malfunction of face spoofing attack without significantly impacting the visual quality of user inspection (i.e., step (2) in the aforementioned 3D surveillance workflow). Meanwhile, since the injected perturbations would not alter the object motion or significantly change the depth to confuse with other types of objects, *LiVSec* is not expected to affect the functionality of motion and depth-based 3D monitoring (i.e., step (1) in the workflow).

2.2 Face Authentication and Spoofing Attack

A face authentication system is a biometric identification system that takes the user's face image as input to authenticate the user as an known individual [2, 4, 10]. With the assistance of deep learning techniques, the face authentication system can intelligently compare the input image with pre-stored user facial information and generate a pass/not pass result based on their similarities. Recently, face ID-based authentication has been deployed in various scenarios given its convenience and effectiveness in accomplishing the authentication task [2, 4, 10].

Since the emergence of face authentication, face spoofing attacks have also been evolving aiming to compromise the face ID-based authentications. In the early days, the traditional face authentication system employs 2D face images with face recognition algorithms for user authentication [2, 4], which can be easily compromised by image-based face spoofing attacks presenting a photo of the victim user to the face authentication system [26, 41, 72]. As a solution to image-based spoofing attacks, the authentication systems have now evolved to examine two additional features to differentiate between the real user and a spoofed face image. One category of approaches employ 3D face models with the consideration of the depth information as an integral part of the authentication [25, 45, 46, 78]. The other category introduces liveness detection to distinguish real human faces from the spoofed face images or models [10, 40, 69].

3 SECURITY THREATS IN 3D SURVEILLANCE

We note that the 3D surveillance video involving 3D user face models can be abused to compromise the aforementioned state-of-the-art 3D and liveness-based face authentication mechanisms. It is because the 3D requirement can be fulfilled by the 3D face models in the video, and the liveness requirement can be achieved by the sequence of the video frames interpreted as the liveness of real human. Therefore, the 3D surveillance video system introduced in Section 2.1 is subject to security issues if left unprotected, which is the focus of this work. In particular, we assume that the sensitive 3D objects can be intercepted by adversaries at any phase of the live video streaming workflow presented in Figure 1, especially at the monitor end that the potential victim users under surveillance have no control over. This security model is in line with the general public concerns about video surveillance [8, 32, 74], due to the lack of knowledge about the remote, untrusted party (i.e., the monitor) who may have access to the surveillance video.

The 3D face model obtained by the attacker can be leveraged to spoof third-party face authentication services, such as in Face ID-based ATM [12], store payment [17], and airport security check-in [18]. The threat model we target in this work is that the victim's face (captured in the surveillance video) may be abused by the adversary (at the monitor end) to falsely pass the face authentication services on behalf of the victim and thus gain benefits (e.g., purchase or withdraw cash using the victim's account). Note that this attack does not need to be a targeted attack on a specific victim, as the attacker can benefit by spoofing an arbitrary victim.

To verify and demonstrate the potential face spoofing attack resulted from the 3D surveillance video, we conduct a proof-of-concept case study of the potential threat model by using an RGB-D image obtained from a 3D video [9] and a state-of-the-art 3D-based face authentication system based on [47]. The architecture of the face authentication system is shown in Figure 2. It takes a user input image with depth information (i.e., in the RGB-D format) and employs a pre-trained Siamese neural network, which consists of two identical ResNet18 networks, to estimate the cosine similarity of the input image with a pre-recorded reference RGB-D face image. If the similarity score is larger than a certain threshold (e.g., 0.9), the face authentication system would deem the user as passing the authentication. We choose this face authentication system based on the following considerations. First and foremost, it involves depth information as a new dimension of the input data, which is one of the most attractive features provided by the 3D surveillance video. Second, for such a system deployed in the real world, it is impossible to retrain the entire model when new users' facial information is added. Therefore, techniques like one-shot learning [67] must be used, which is one of the great features of the Siamese networks. Last but not least, it is a well recognized open-source project with open-source dataset to train and test the model, which provides us with full control of the face authentication system and thus more flexibility to design and customize our defense strategy.

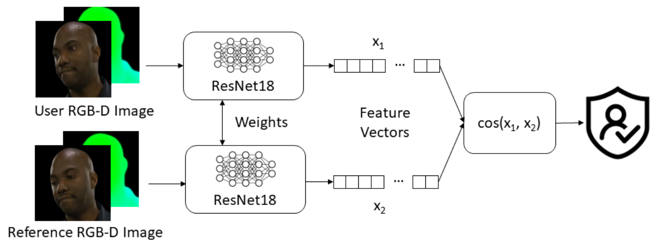

Figure 2: The architecture of the face authentication model [47]. Two ResNet18 networks share the same weights, creating a Siamese neural network.

Figure 3 presents the workflow of the attack, where the attacker feeds the face authentication system with an RGB-D face model extracted from the surveillance video, with the malicious attempt of passing the authentication and thus spoofing the victim users. In our experiments with 121 images obtained from the target 3D video [9], all of the images result in a similarity score larger than the threshold, indicating successful face spoofing attacks. Since a successful attack only requires a single point of vulnerability (i.e., one image), such a high percentage of success rate indicates a significant security vulnerability in 3D video surveillance that must be addressed.

While we only demonstrate one instance of the face authentication attack enabled by exploiting facial information in unprotected 3D surveillance videos, the category of face authentication attacks targeting similar image or video applications have been demonstrated in the literature of various sources [3, 27, 63, 69]. Moreover, in addition to face authentication attacks, the exploited facial information from images/videos can enable an even broader range of security attacks, such as Deepfake attacks [57, 68] and privacy leakage [22, 32, 33, 50, 59, 64, 74]. Overall, the threat model of exploiting facial information in the victim videos have drawn significant security concerns that must be addressed.

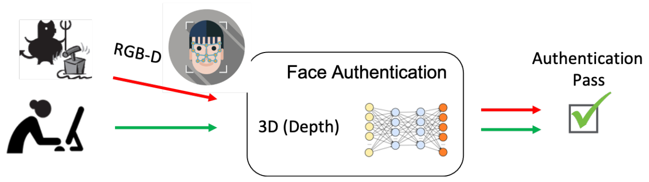

Figure 3: Workflow of the 3D surveillance threat model: Face ID spoofing attack.

4 PROPOSED DEFENSE: *LIVSEC*

4.1 System Overview

The aforementioned quality and security requirements for 3D video surveillance motivate us to adopt an adversarial attack method [24, 35, 43] discussed in the machine learning community to form the basis of the defense mechanism, because a successful adversarial attack would add minimum perturbation to make the neural network produce incorrect inference results while preserving the quality of the image. This feature aligns with our intended security and quality requirements in this work.

Inspired by the features of the adversarial perturbations, we develop a live 3D surveillance defense mechanism, namely *LiVSec*, to counter the face spoofing attack demonstrated in Section 3. In essence, *LiVSec* accomplishes the defense by adding the adversarial perturbations to the 3D surveillance videos in a benign manner, so that even if the perturbed (and thus protected) 3D facial data is extracted by an attacker, it cannot be used to issue a successful face spoofing attack.

The end-to-end workflow of a 3D video surveillance system is illustrated in Figure 1. To enable security protection, we propose to integrate *LiVSec* into the video source, to ensure that the video delivered to the network for distribution is protected. The generation of protected frames is the core of our defense mechanism, which should work transparently to both the video source and the monitor (i.e., meet the quality and real-time performance requirement).

Figure 4 shows the workflow of *LiVSec* in generating the protected 3D video frames at the video source. We first conduct preprocessing to extract the 3D face model from the target surveillance video (discussed in Section 4.2). Then, we develop a real-time perturbation generation model to inject the desired perturbations that meet the security, quality, and performance requirements (discussed in Section 4.3). The protected face are then used to replace the original in the target video for streaming.

4.2 3D Face Extraction

A frame of a 3D video, as demonstrated in Figure 5, has a black background with two components (i.e., RGB and depth). The upper part of the frame is the RGB part like 2D videos, and the lower part represents the depth values of the corresponding pixels in the RGB part. It is worth noting that the depth values are represented by RGB colors as well, as illustrated in the Hue color bar in Figure 5. For example, we can observe that the face is closer to the camera than the legs and feet. To protect the video frame, there is no need to add perturbations to the entire frame but only the face part (both RGB and depth), because (1) only the human face is considered security

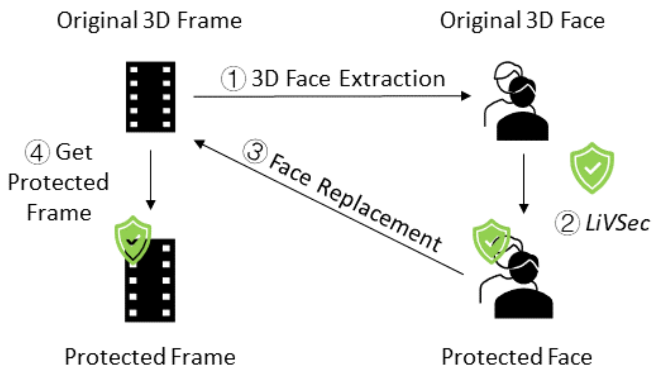

Figure 4: The procedure of protected frame generation.

sensitive that we aim to protect; and (2) we must minimize the injected perturbation and meet the quality requirement. Based on the features of this 3D video, to obtain the RGB-D face, we must (1) locate the face in the frame and extract the RGB facial information; and (2) extract the depth facial information by translating the RGB-based depth information to 1-D numerical values.

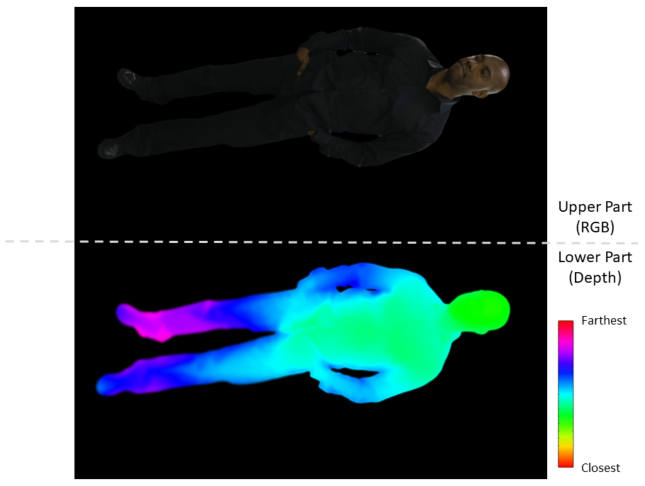

Figure 5: Example frame from a 3D video [9], which is adopted for the design and evaluation of *LiVSec*.

4.2.1 RGB Facial Information Extraction. We use a 256×256 rectangle filter to extract the RGB-D facial information from the frame shown in Figure 5. The core idea is to find the column containing the pixels on the top of the head, which is implemented in a binary search algorithm. We determine if all the pixels in column *mid* are black (i.e., the sum of values in this column is 0). If it is true, it means that the column we aim to find is to the left of this column, otherwise it is to the right. The coordinate of the filter's top left pixel is defined as (h, w), where w is calculated by the left search boundary L minus 270, and we adopt a fixed h value in the current implementation, as the subject moves very little in the video scene.

4.2.2 *Depth Facial Information Extraction.*

The RGB-based depth information is located by applying an offset to move the aforementioned rectangle filter from the upper part to the lower part. According to the design of the 3D video, in the RGB-based depth channel, the hue value of the pixel can be used as the normalized depth value (i.e., the normalized distance between that pixel and the camera). The conversion process to obtain the normalized hue value H from RGB is shown in Equation (1):

$$H = \begin{cases} 0 & if\ \Delta = 0 \\ (\frac{G-B}{\Delta} mod 6)/6 & if\ C_{max} = R \\ (\frac{B-R}{\Delta} + 2)/6 & if\ C_{max} = G \\ (\frac{R-G}{\Delta} + 4)/6 & if\ C_{max} = B \end{cases} \tag{1}$$

where R, G, and B are normalized to $[0, 1]$, and C_{max} and Δ are defined as follows:

$$C_{max} = max(R, G, B)$$
$$C_{min} = min(R, G, B)$$
$$\Delta = C_{max} - C_{min}$$

Finally, we create a concatenation of the facial RGB and depth data x as the input of the perturbation generation in *LiVSec*.

4.3 Protected Face Generation

The key step in generating a protected face in *LiVSec* is to derive a generative model $U(x)$, which generates the perturbation δ for the input x to defeat potential face spoofing attacks. To achieve minimum perturbation and thus meet the quality requirement, we take advantage of the unique depth information in the 3D video and design a filter *mask* that discards the unnecessary perturbations added in the background. The depth information represents the distance between that pixel and the 3D camera and, therefore, it is useful to distinguish the face and the background given that the depth of the background is much larger or even infinity. By applying *mask*, the value of the perturbation δ is updated to $\delta \times mask$, and the $clamp(\cdot)$ function is called to ensure that the abstract value of δ is smaller than the perturbation strength *pert*, which is a pre-defined parameter. After that, another $clamp(\cdot)$ function clips the values of the perturbed face $x + \delta$ into a valid range ($[-1, 1]$ in our case) to produce the final protected face x'. A protected frame to be sent for streaming is then generated by replacing the original face with the protected one, which is the last step of the protected 3D frame generation.

4.3.1 *Perturbation Generation.*

We employ U-Net [54] as the generative model for perturbation generation, which is originally designed for image semantic segmentation when it was first introduced but later applied in generating adversarial perturbations [49, 51]. It is lightweight and only requires a single forward to produce the perturbations, which is much faster than the iteration-based adversarial attacks (e.g., C&W attack [24]) and thus helps achieve the real-time performance. We follow the architecture of the original U-Net but change the number of input channels to 4, and the input channel sizes of the four down-sampling layers to 32, 64, 128 and 256. The output channel number of the fourth down-sampling layer is 512. The input/output channel numbers of the up-sampling layers are changed accordingly.

4.3.2 *Training* LiVSec.

Algorithm 1 describes the training process of *LiVSec*. During the training, the reference input x_{ref} is the current image loaded, while the user input x is randomly chosen from the other images of the same user. There is no need to use images from other users because the similarity scores between them are already smaller than the threshold. After getting the protected face x' following the steps described in both Section 4.3 and Algorithm 1, we calculate the cosine similarity score between $F(x')$ and $F(x_{ref})$ to obtain the value of our loss function, where $F(\cdot)$ is the face authentication model, and its output is a feature vector of the input. The cosine similarity score between the two vectors x_1 and x_2 is defined as follows [6]:

$$cosine\ similarity = \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon)} \tag{2}$$

where ϵ is a small number to avoid division by zero. Furthermore, an optimizer is used to help update the weights of $U(\cdot)$ to minimize the value of the loss function. The training keeps updating $U(\cdot)$ until the iteration number reaches *max_epochs*.

Algorithm 1: *LiVSec* Training Algorithm

```
Input: Training dataset $X = \{x_{11}, ..., x_{1m}, x_{21}, ..., x_{nm}\}$,
       where $n$ is the number of users and $m$ is the number
       of poses of each user, the corresponding depth mask
       $M = \{mask_{11}, ..., mask_{nm}\}$, perturbation strength
       $pert$, maximal number of epochs $max\_epochs$.
Result: Trained perturbation generator $U(\cdot)$.
Freeze the weights of the face authentication model $F(\cdot)$.
Randomly initialize $U(\cdot)$.
for $k \leftarrow 1$ to $max\_epochs$ do
    for $j \leftarrow 1$ to $n$ do
        for $i \leftarrow 1$ to $m$ do
            $x_{ref} \leftarrow x_{ji}$;
            $x \leftarrow$ randomly choose one from
              $\{x_{j1}, ..., x_{jm}\} - \{x_{ji}\}$;
            $\delta \leftarrow U(x)$;
            $\delta \leftarrow \delta \times mask_{ji}$;
            $x' \leftarrow x + clamp(\delta, -pert, pert)$;
            $x' \leftarrow clamp(x', -1, 1)$;
            $Loss \leftarrow CosineSimilarity(F(x_{ref}), F(x'))$;
            update $U(\cdot)$ to minimize $Loss$;
        end
    end
end
```

5 IMPLEMENTATION AND CASE STUDY

We implement and deploy *LiVSec* into an end-to-end live 3D video streaming system, following the 3D video surveillance workflow illustrated in Section 2.1. In addition, we also implement the face authentication system introduced in Section 3 to evaluate *LiVSec* against face authentication attacks. In this section, we discuss the system implementation details, as well as a case study to demonstrate the basic workflow and functionality of *LiVSec*.

5.1 3D Surveillance Video Streaming System

Figure 6 shows the architecture of the end-to-end *LiVSec*-based video streaming system, which involves a live video content server and a client. At the server end, the live video source is captured by a 3D camera, perturbed by *LiVSec*, and served on a web server. The client requests and streams the video from the content server for surveillance monitoring.

- **Live Surveillance Video.** In our implementation, we adopt a 3D video dataset (Dataset #2 described in Section 6.1.1 [9]) to serve as the live video source captured from a 3D camera. We adopt FFmpeg [65] as the decoder to read the video frame by frame in the *image2pipe* mode, with the *rgb*24 pixel format and the *rawvideo* codec.
- **LiVSec.** *LiVSec* accepts each raw frame from the input pipe and produces protected 3D surveillance frames to the output pipe, which is described in Section 4. Both the raw frame and the protected frame are transmitted in byte arrays.
- **Virtual Camera.** After the protected frames are generated by *LiVSec*, FFmpeg is further used as the encoder to write the frames to a virtual 3D camera device, which is simulated by v4l2loopback [79]. The virtual camera works as the source for the *LiVSec*-protected live video stream deployed on the web server following the DASH video streaming standard [58]. In our implementation, FFmpeg receives the protected frames from the output pipe and writes them to the *v4l2* camera device under the speed of 24 FPS. The frames are sent with the same format, i.e., 2048×2048 resolution, *rawvideo* codec, and *rgb*24 pixel format. Then, we adopt DashCast [56] as the packager to generate live video streams from the virtual camera with 1-second segment duration and 5*s* time shift, i.e., all the video segments will be kept on the server side since the video is 5-second long.
- **Web Server.** The generated live video segments are deployed on the web server for the client to request via HTTP following the DASH standard [58]. In our system implementation, we employ a web server for live DASH streaming [36].
- **Client.** On the client side, we use GPAC MP4Client [37] to receive and playback the protected video. In our system, we stream both the RGB and depth (in color map) of the frame and display them separately at the player, considering the RGB and depth information may be used separately for different surveillance tasks at the receiver/monitor, as described in Section 2.1.

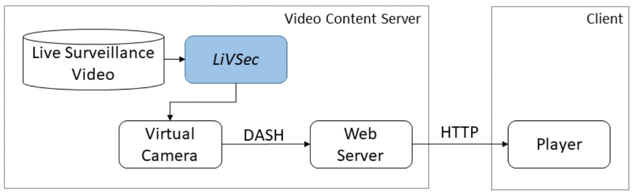

Figure 6: Architecture of the end-to-end *LiVSec*-based live surveillance video streaming system.

5.2 3D Face Authentication System

We implement the 3D face authentication system (described in Figure 2) to facilitate the evaluation of *LiVSec* against face authentication attacks. The face authentication model takes two $300 \times 300 \times 4$ RGB-D images as inputs; one of them is the user input image extracted from the 3D surveillance video, and the other is a pre-recorded reference image. Inside the authentication model, two ResNet18 models share identical weights and construct a Siamese network to generate two 128×1 feature vectors for the two inputs. In the next step, cosine similarity (formulated as Equation (2)) is employed to measure the distance between these two feature vectors, and the measured result turns out to be a number between -1 and 1. Based on the definition of cosine similarity, -1 means the farthest distance (i.e., the values of two images are opposite) while 1 means the closest (i.e., two images are identical). Lastly, the face authentication system determines if the user should pass the authentication by checking if the similarity score is larger than 0.9 (i.e., passing the authentication) or not. For cosine similarity formulated in Equation (2), ϵ is set as 1e-8.

We use Dataset #1 (described in Section 6.1.1) [31] to train the face authentication system. Also, we choose Cosine Embedding Loss as the loss function to maximize the dissimilarity between two inputs x_1, x_2, i.e., the loss between two inputs is minimized. The loss is computed based on the cosine similarity between the inputs and their corresponding label y. When inputs x_1 and x_2 come from the same user, the label y is 1, otherwise, it is -1. The loss is defined as

$$loss(x_1, x_2) = \begin{cases} 1 - cos(x_1, x_2) & y = 1 \\ max(0, cos(x_1, x_2) - margin) & y = -1 \end{cases} \quad (3)$$

where *margin* is set as 0.5 in the training. We use stochastic gradient descent (SGD) as the optimizer to train the 3D face authentication system, and the learning rate, weight decay and momentum for it are 0.01, 0.0001, and 0.9, respectively. In addition, the batch size is 128, while the maximum number of epochs is set as 200.

5.3 Case Study

We perform a case study to demonstrate the procedure and the defense effectiveness of *LiVSec* using a sample image from Dataset #2 (described in Section 6.1.1) [9], which is illustrated in Figure 7. The first column shows the 3D face image of the user, which is a reference input that has been pre-registered with the face authentication system. The second column shows the 3D face data extracted by an adversary from the 3D surveillance video and can be used as the user input for face authentication. Without enabling *LiVSec*, the similarity score between the reference input and the user input is 0.986, which indicates that the adversary can impersonate the legitimate user by feeding such face model to the face authentication system. After deploying *LiVSec*, as shown in the third column, the similarity score decreases to 0.779, which makes the adversary no longer capable of accomplishing a face spoofing attack. Meanwhile, the quality impact caused by the protected face is minimal, meeting the quality requirement and thus ensuring that the surveillance task at the monitor side is uninterrupted. More detailed visual results with additional images, as well as sample videos visually demonstrating the original and protected videos, can be found in the repository of the project at https://github.com/hwsel/LiVSec.

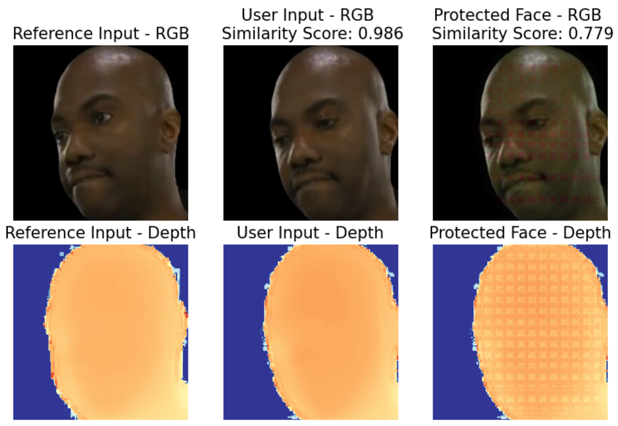

Figure 7: A case study on using *LiVSec* to defend against face spoofing attack using a sample image from the dataset [9]. From left to right, the three columns are the RGB-D data of: reference input pre-registered in the face authentication system, user input extracted from a 3D surveillance video, and the protected user input by applying *LiVSec*.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

6.1.1 Datasets. We adopt two datasets for the evaluation of *LiVSec*:

Dataset #1 [31] contains 1581 RGB-D images from 31 users, captured by a Microsoft Kinect V1 depth camera [1]. For each user, there are 17 different poses, including 13 face orientations and 4 facial expressions (i.e., smiling, sad, yawn, and angry). 3 images are collected for each different pose of each user. The RGB-D images of 26 out of 31 users are used for face authentication system training, and those of the rest 5 users are used for validation. In other words, 1326 images are used for training, and 255 images are used for validation. During the training/validation process, we use all images as the reference input sequentially and choose different random images as the user input. The validation dataset is also used in our evaluation, where we choose a fixed image of each user as the reference input and the rest 50 images of that user as the user input. Since we use the images of 5 users for evaluation, in total we have $50 \times 5 = 250$ test cases. In this dataset, RGB images are $1280 \times 960px$ as 32-bit bitmaps, while the depth information has been pre-processed by the dataset provider and is stored in text files in the format of integer numbers to represent the depth values, with the resolution of $640 \times 480px$. The images from Dataset #1 are center-cropped and resized to $300{\times}300px$ for the face authentication system and the perturbation generator.

Dataset #2 consists of 121 continuous RGB-D frames extracted from a 5-second 3D video clip, which is from a demo video shot by Azure Kinect depth camera and edited by Depthkit software [9]. The demo contains a 3D video of one user, and each frame of the video contains the face of the user. In our experiments, we use the first frame as the reference input and all the frames as the user input. This is because, during the streaming, it could be impossible to pre-store a user's reference input, so we naturally choose the first frame as the reference. It is different from Dataset #1 because this frame needs to be protected as well. Therefore, in total, we have 121 test cases for Dataset #2. In this dataset, the video frames are in the size of 2048×2048. The top half of a frame is the RGB data of the user and the bottom half is the depth information. The images from Dataset #2 are cropped to 256×256 by applying the face locator discussed in Section 4.2.

6.1.2 Parameter Settings. For training, the perturbation strength is set as 16/255, the maximal number of epochs *max_epochs* is 200. Stochastic gradient descent (SGD) is used as the optimizer with learning rate 0.01 and weight decay 0.0001. The similarity threshold for the face authentication system is set as 0.9. For RGB facial information extraction, h is fixed as 140.

6.1.3 Evaluation Metrics. We employ the following security, quality, and performance metrics in our evaluations.

Security Metric: Success Rate. Recall that we use cosine similarity (in Equation (2)) to measure the similarity of two inputs, so the higher the similarity score is, the more similar the two images are. For two identical images, their similarity score would be 1. Based on this observation, we define a user input whose original similarity score is larger than the authentication threshold (i.e., 0.9 in our experiments) as a *valid test*, because originally it can be used to bypass the face authentication system. Then, if the similarity score of a protected image falls below the authentication threshold, it is counted as an *effective defense*. The success rate metric is then defined as the ratio between the *effective defenses* and the *valid tests*.

Quality Metrics: Normalized L2 Norm and Learned Perceptual Image Patch Similarity (LPIPS). To evaluate whether the protected 3D surveillance video meets the quality requirement described in Section 1, we first use normalized L2 norm to measure the quality of protected frames, which is widely adopted in the machine learning community [24, 29, 61] and in the related work [63]. We also use LPIPS [30, 76] to evaluate the perceptual similarity (i.e., the quality) between the protected frames and the original frames, which is a widely adopted perceptual similarity metric in the community better reflecting human visual experience than the traditional PSNR or SSIM metrics. It utilizes deep features of a well-trained image classification model to achieve perceptual similarity judgments. Similar to the normalized L2 norm, lower values mean closer/similar. We adopt the open-source LPIPS project [75] to perform our evaluations with version 0.1 and AlexNet. Note that we only use LPIPS to evaluate the RGB part of the frames.

Real-Time Performance Metric: Frame Rate. We employ frame rate as the metric to evaluate if the proposed approach meets the real-time performance requirement:

$$frame\ rate = \frac{\#\ frames\ sent}{time\ to\ send\ the\ frames} \tag{4}$$

where the time to send the frames includes the time for the system to read the frames from the video, add perturbations, and send them for video streaming. This metric only applies to Dataset #2 that has a video sequence involved.

6.2 Security Evaluation

The security evaluation results are shown in Table 1. We observe that, after deploying *LiVSec*, the average similarity scores decrease by 44% and 18% for Dataset #1 and #2, respectively. Both average similarity scores with *LiVSec* are lower than the threshold (i.e., 0.9), which indicates successful security defense. To better illustrate the similarity score changes, we present the distributions of all test cases for Datasets #1 and #2 in Figure 8 and Figure 9, respectively. We observe an obvious boundary between the two cases with and without *LiVSec*, which justifies that *LiveSec* meets the security requirement. Furthermore, the defense success rate is 100% for both datasets. Specifically, in Dataset #1, there are 228 out of the 228 valid cases that prevent the face spoofing attack successfully. Similarly, in Dataset #2, the defense is successful for all the 121 cases.

Table 1: Security evaluation results for both datasets.

	Dataset #1	Dataset #2
Avg. Similarity Score w/o *LiVSec*	0.966	0.988
Avg. Similarity Score w/ *LiVSec*	0.539	0.806
Success Rate	100% (228/228)	100% (121/121)

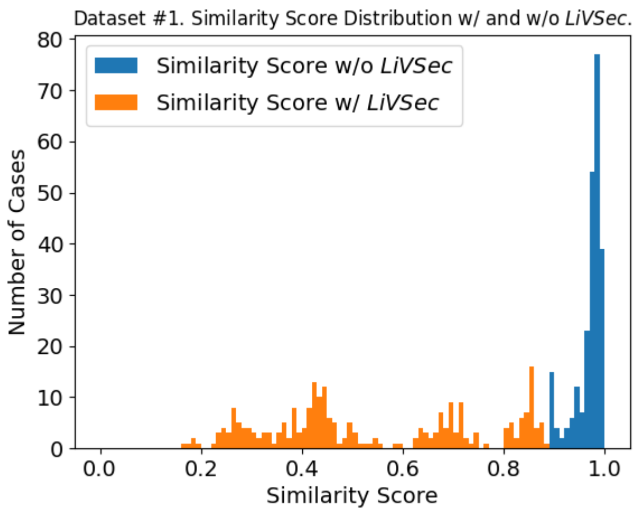

Figure 8: Similarity score distributions for Dataset #1.

6.3 Quality Evaluation

We use normalized L2 norm and LPIPS to evaluate the quality of the protected faces. The distribution of the results is shown in Table 2. For Dataset #1, the average L2 norm is 0.0315 and the average LPIPS is 0.0442, and for Dataset #2 the results are 0.0713 (average L2 norm) and 0.1764 (average LPIPS). Overall, the L2 norm and LPIPS results of both datasets fall in a low and narrow range of values, indicating the robustness of *LiVSec* in meeting the quality requirement.

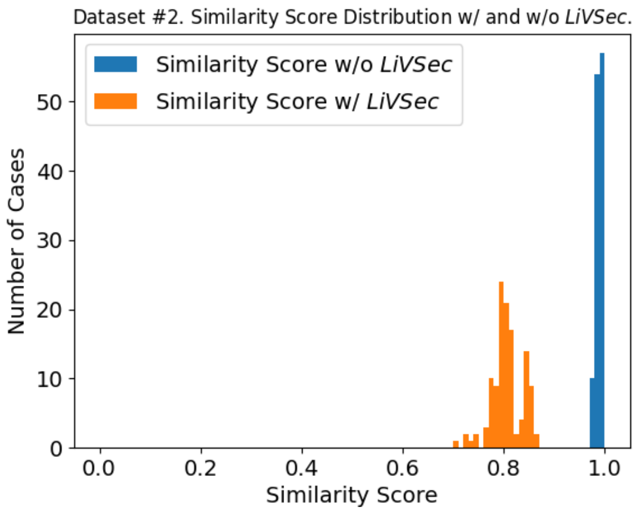

Figure 9: Similarity score distributions for Dataset #2.

Table 2: Quality evaluation results for both datasets. Normalized L2 norm and LPIPS are used as the evaluation metrics.

	Norm. L2 norm		LPIPS	
	Dataset #1	Dataset #2	Dataset #1	Dataset #2
Minimum	0.0267	0.0692	0.0288	0.1551
First Quartile	0.0298	0.0709	0.0370	0.1727
Median	0.0311	0.0714	0.0443	0.1787
Thrid Quartile	0.0334	0.0718	0.0514	0.1823
Maximum	0.0357	0.0754	0.0627	0.1887
Average	0.0315	0.0713	0.0442	0.1764

6.4 Performance Evaluation

We further evaluate the performance of *LiVSec* in term of meeting the real-time requirement using Dataset #2, which is shown in Table 3. In the original security strategy (i.e., the "Reuse-1" column), it takes 3.84 seconds to process and send all the 121 frames, which results in a 31.51 FPS performance. To further explore the potential of the frame rate, we deploy a "Reuse-X" strategy, where we reuse the perturbation of the current frame for the next $X - 1$ frame(s), to save the perturbation generation time and improve the frame rate.

Table 3 shows that we can gain improved performance by applying the "Reuse-X" strategy. For example, Reuse-5 increases the frame rate from 31.51 to 34.38. Meanwhile, we observe that the success rate remains at 100% in all the reuse scenarios, which preserves the security of the system. Also, the average similarity scores with *LiVSec* deployed decreases in both Reuse-5 and Reuse-10 cases but only slightly. For quality, both the normalized L2 norm and the LPIPS almost do not change when applying the reuse strategies. Overall, our experiments show that the "Reuse-X" strategy helps improve the performance while posing minimum impact to the security and quality metrics. It is worth noting that the original frame rate of the 3D surveillance video is 23.98 FPS, which can be sufficiently supported by *LiVSec* without performance downgrade;

Table 3: Security, quality, and performance evaluation results w/ Reuse-X strategy using Dataset #2 with 121 frames.

	Reuse-1	Reuse-5	Reuse-10
Avg. Similarity Score w/o LiVSec	0.988	0.988	0.988
Avg. Similarity Score w/ LiVSec	0.806	0.804	0.799
Success Rate	100% (121/121)	100% (121/121)	100% (121/121)
Avg. Norm. L2 Norm	0.0713	0.0714	0.0714
Avg. LPIPS	0.1764	0.1762	0.1758
Time Cost (Sec)	3.84	3.52	3.43
Frame Rate (FPS)	31.51	34.38	35.28

furthermore, the experiment results imply that *LiVSec* can work with higher framerate videos (i.e., up to 35 FPS).

Furthermore, we conduct a timing breakdown analysis on an arbitrary frame from Dataset #2 to better understand the performance of the *LiVSec* system (i.e., the workflow illustrated in Figure 4). The timing breakdown results are shown in Figure 10. We observe that the key step of *LiVSec*, Protected Face Generation (i.e., Step ② in Figure 4), only takes 5.88 ms (around 20% of the total time). Specifically, the perturbation generation itself only takes 3.99 ms, accounting for around 13% of the total time. Overall, we note that the most time-consuming components in the system are the I/O operations (i.e., writing/reading data to/from disk, reading an original frame from the pipe, and writing the protected frame back to the pipe), which account for over 70% of the total time. Based on the timing analysis, we can conclude that the perturbation generator has efficient timing performance compared to other components in the system, and the system can be further optimized by improving the I/O capability.

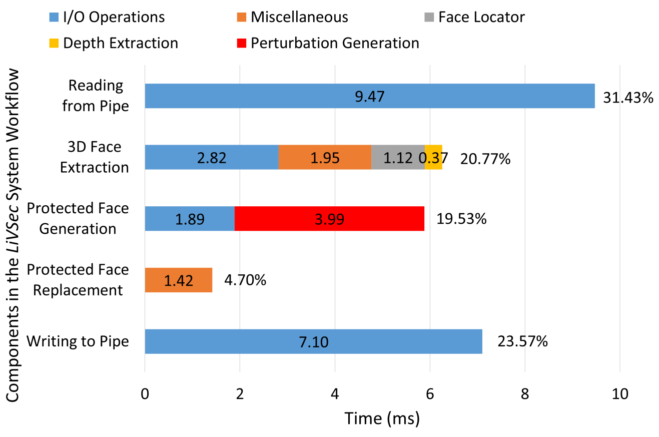

Figure 10: Timing analysis of the *LiVSec* system.

7 RELATED WORK

Among all the related works, *VVSec* [63] is probably the most closely related work targeting similar video application (volumetric video) and threat models (face authentication attack) with *LiVSec*. However, *VVSec* only targets the video-on-demand (VOD) scenario, where the video is pre-recorded that allows offline processing without the real-time processing requirements. Under this context, *VVSec* adopted an offline, iterative perturbation generation approach to protect the victim video, and the perturbation generation process requires a significantly long processing time, at the magnitude of a few seconds, which does not meet the real-time performance requirement for live 3D surveillance videos targeted by *LiVSec*. To confirm this observation, we conduct a quantitative experimental comparison with *VVSec* (based on the *VVSec* open source release in [62]) in terms of security, quality, and performance between the two approaches, the results of which are shown in Table 4. In the experiments, only the videos in the validation part of Dataset #1 are used. We observe that both systems successfully defend all the attacks, and *LiVSec* runs almost 100x faster than *VVSec* as benefiting from the proposed real-time generative model for perturbation generation. It is worth noting that *VVSec* produces less perturbation (in terms of L2 norm and LPIPS) than *LiVSec* given the finer-grained, iterative perturbation process and, therefore, it can support the entertainment video streaming scenario with higher quality requirement than the surveillance video targeted by *LiVSec*. We will discuss about the quality limitation of *LiVSec* in Section 8.

VR/AR Security/Privacy. Security/privacy has been deemed as an important topic in the multimedia and VR community given the rich 3D representation of the multimedia content and the significant real user interactions. The existing security/privacy works have targeted several directions. The first direction is to restrict the access to sensitive input (e.g., sensor) data [32, 33, 59, 64]. For example, both DARKLY [33] and PlaceAvoider [64] employ computer vision algorithms to address such concerns, which target 2D images/videos in computer vision applications and first-person cameras, respectively. Jana et al. [32] propose a new OS abstraction to build a fine-grained permission system that only exposes higher-level objects instead of raw sensor data to AR applications. PrivacyEye [59] designs a mechanical camera shutter to occlude the first-person camera private scenes. The second direction targets security or privacy concerns caused by malicious or buggy VR/AR outputs [38, 55]. For instance, Arya et al. [38] utilize policy constraints to detect and eliminate malicious VR outputs. The third direction aims to protect users' biometric information or unique patterns to keep them anonymized, in which differential privacy (DP) has been studied [23, 39]. In addition, several recent works have focused on addressing the security challenges of user authentication and interactions in VR/AR applications [21, 42, 44]. Overall, these existing works all target the *explicit* security/privacy attacks that either alter or gain unauthorized accesses to the sensitive data; hence, they do not need to consider the quality requirements. Also, only several solutions can achieve real-time performance [32, 39, 59] in their specific application scenarios. Different from the existing works, *LiVSec* mainly focuses on the *implicit* security/privacy implications from the rich multimedia content, which is an orthogonal dimension with novel research challenges and contributions.

Table 4: Comparison between VVSec [63] and *LiVSec* using Dataset #1.

Paper	Application Scenario	Success Rate	Avg. L2 Norm	Avg. LPIPS	Avg. Running Time (s)
VVSec [63]	Volumetric video	100%	0.0053	0.0017	4.3
LiVSec	3D surveillance	100%	0.0315	0.0442	0.045

Surveillance Video Security/Privacy. Existing works of surveillance video security/privacy mainly focus on 2D surveillance videos. Surveillance videos may contain biometric (e.g., face, iris, and fingerprint) and non-biometric (e.g., text and license plate) information that are security or privacy sensitive. Since (a) humans are often the subject of interest in surveillance videos, and (b) face and body attributes provide personal privacy information like identity, age, race, and gender, many existing works [34, 48, 53] aim to protect face and body in the surveillance videos. Common methods to solve these security/privacy concerns include (1) encrypting the selected portion (i.e., the region of interest) of the video [48]; (2) de-identification [53], which generates an alternative of the original face to hide the identity; (3) pixelation [74], which blurs the sensitive images; and (4) privacy-preserving cameras have been designed to achieve pre-capture privacy by either using other sensors (e.g., a thermal sensor) to avoid collecting RGB data directly [50], or applying well-designed optical lens to the camera to filter unwanted information [22]. These existing protection mechanisms only consider 2D videos, while our target application scenario is 3D video surveillance. Moreover, solutions that fall into categories (1), (3) and (4) would block the human vision (i.e., users can no longer recognize the face features from the protected video), and solutions like (2) would alter the appearance of the face, which would also block the human vision. Therefore, all the 4 categories of solutions do not meet the requirements for protecting the security of live 3D surveillance videos.

Adversarial Attacks. Adversarial attacks have been a popular security topic discussed in the machine learning community. The classical adversarial attack methods [24, 35, 43] add small perturbations to the input images of deep learning models to maliciously alter the inference results. Inspired by the classical adversarial attack, a small number of recent works utilize adversarial attacks for benign use cases as a means of obfuscation [60, 70, 73]. *LiVSec* was also inspired by these adversarial attack techniques to develop the perturbation generation method; however, none of the existing approaches have targeted live 3D video with the real-time performance requirement, which presents unique challenges to address in the design of *LiVSec*. Furthermore, the defense mechanisms against adversarial attacks have been an active research area in the machine learning related communities [71], which can motivate further research on the potential threat models against *LiVSec*.

8 LIMITATIONS OF *LIVSEC*

During the evaluations, we observe that a notable limitation of *LiVSec* is the non-ideal quality of the protected video frames with the amount of perturbations generated by the real-time generative model. For example, *LiVSec*-protected frames would present perturbations visible to human eyes, which is less competitive than iterative-based perturbation generation approaches (e.g., *VVSec* [63], as compared in Table 4.) Due to this limitation, the current version of *LiVSec* can only be applied to the application scenarios that do not require significantly high viewing quality; for example, it can meet the requirement of surveillance videos targeted by this paper but cannot support the entertainment videos where such amount of perturbations could impact the users' quality of experience. While investigating this limitation, we found that it is due to the trade-off between the perturbation generation speed and quality, as similar quality limitations can also be identified in other works involving real-time generative models [49, 51]. In the future work, we plan to conduct a more in-depth study on optimizing the generative model to address the quality limitation, e.g., by fine-tuning the model parameters to seek a balance between the speed and quality or by accelerating the perturbation generation process.

Also, LiVSec only supports a single person/face in the surveillance video; therefore, our evaluations are limited to surveillance videos that contain one person. Also, the face cropping/extraction is a necessary step for LiVSec, which we presented in Section 4.2. Moving forward, to support surveillance video containing multiple persons, the fundamental methodology of LiVSec would not change, but it requires more system acceleration efforts (e.g., via parallel processing) to inject multiple perturbations while maintaining the same real-time performance. In addition, improving and evaluating *LiVSec* under adaptive streaming scenarios (e.g., varying network conditions) would require more in-depth studies on the streaming of perturbed videos, which we target as future work.

9 CONCLUSION

We for the first time investigated the security implications of live 3D video surveillance. We identified the security vulnerabilities caused by the 3D sensitive objects (e.g., human face) in the surveillance video that lead to spoofing attacks. To address the security issue, we developed a novel perturbation generation method, namely *LiVSec*, which employs a real-time generative neural network model to inject small perturbations to the 3D video frames and protect the sensitive objects. *LiVSec* makes non-conventional use of the traditionally malicious attack method as a benign defense mechanism, which achieves the desired security goals. We evaluated *LiVSec* on two datasets involving RGB-D images and 3D video, which demonstrates effective security protection, acceptable quality degradation, and real-time performance, meeting the requirements of live 3D video surveillance. The repository of the project is at https://github.com/hwsel/LiVSec.

ACKNOWLEDGMENTS

We would like to thank the reviewers and the shepherd for their constructive feedback. This work was partially supported by the National Science Foundation under awards 1912593 and 2200048.

REFERENCES

[1] 2015. Kinect Docs. https://github.com/Kinect/Docs.
[2] 2017. Apple Event Keynote 2017: iPhoneX Face ID and Animoji. https://www.youtube.com/watch?v=eRvBU_tKGjE.
[3] 2017. Windows 10's face authentication defeated with a picture. https://www.theverge.com/2017/12/21/16804992/microsoft-windows-10-windows-hello-bypass-security.
[4] 2017. Windows Hello face authentication. https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/windows-hello-face-authentication.
[5] 2019. 3D machine vision guides robotic system for logistics e-fulfillment. https://www.vision-systems.com/cameras-accessories/article/16736111/3d-machine-vision-guides-robotic-system-for-logistics-efulfillment.
[6] 2019. CosineSimilarity - PyTorch Docs. https://pytorch.org/docs/stable/generated/torch.nn.CosineSimilarity.html.
[7] 2019. How 3D sensors are transforming the transportation and logistics industry. https://www.zebra.com/us/en/blog/posts/2019/this-one-technology-is-delivering-a-solution-to-the-parcel-problem.html.
[8] 2020. Balancing privacy concerns with video monitoring capabilities. https://www.sdmmag.com/articles/98278-balancing-privacy-concerns-with-video-monitoring-capabilities.
[9] 2020. DepthKit. https://depthkit.tv.
[10] 2020. Online face recognition software demo – The BioID playground. https://www.bioid.com/playground/.
[11] 2021. 3D camera market size, share and trends analysis report. https://www.grandviewresearch.com/industry-analysis/3d-camera-market.
[12] 2021. Cash withdrawal in an ATM with face biometrics – Use case. https://www.electronicid.eu/en/blog/post/cash-withdrawal-atm-face-biometrics/en.
[13] 2021. Time-of-flight camera - An introduction. https://www.mouser.com/applications/time-of-flight-robotics/.
[14] 2021. Why 3D is the key to unlocking vital video surveillance data. https://oyla.ai/why-3d-is-the-key-to-unlocking-vital-video-surveillance-data/.
[15] 2021. ZED 2 camera. https://www.stereolabs.com/zed-2/.
[16] 2022. 3D surveillance with Leica BLK247 and CORTROL VMS: Smart 3D surveillance system. https://ganzsecurity.com/subpage/1308/3d-surveillance-with-leica-geosystems-blk247-and-cortrol.
[17] 2022. Mastercard launches tech that lets you pay with your face or hand in stores. https://www.cnbc.com/2022/05/17/mastercard-launches-tech-that-lets-you-pay-with-your-face-or-hand.html.
[18] 2022. TSA PreCheck: Touchless identity solution. https://www.tsa.gov/biometrics-technology/evaluating-facial-identification-technology.
[19] 2023. Spectrum series 3D people counter. https://www.trafsys.com/spectrum-3d-people-counter/.
[20] Shivang Aggarwal, Sibendu Paul, Pranab Dash, Nuka Saranya Illa, Y Charlie Hu, Dimitrios Koutsonikolas, and Zhisheng Yan. 2020. How to evaluate mobile 360° video streaming systems?. In *International Workshop on Mobile Computing Systems and Applications (HotMobile)*. 68–73.
[21] Abdullah Al Arafat, Zhishan Guo, and Amro Awad. 2021. VR-Spy: A Side-Channel Attack on Virtual Key-Logging in VR Headsets. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 564–572.
[22] Bijie Bai, Yi Luo, Tianyi Gan, Jingtian Hu, Yuhang Li, Yifan Zhao, Deniz Mengu, Mona Jarrahi, and Aydogan Ozcan. 2022. To image, or not to image: Class-specific diffractive cameras with all-optical erasure of undesired objects. *arXiv preprint arXiv:2205.13122* (2022).
[23] Efe Bozkir, Onur Günlü, Wolfgang Fuhl, Rafael F Schaefer, and Enkelejda Kasneci. 2021. Differential privacy for eye tracking with temporal correlations. *Plos one* 16, 8 (2021), e0255979.
[24] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*. 39–57.
[25] Huangxun Chen, Wei Wang, Jin Zhang, and Qian Zhang. 2019. EchoFace: Acoustic Sensor-Based Media Attack Detection for Face Authentication. *IEEE Internet of Things Journal* 7, 3 (2019), 2152–2159.
[26] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. 2019. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4690–4699.
[27] Nguyen Minh Duc and Bui Quang Minh. 2009. Your face is not your password face authentication bypassing lenovo–asus–toshiba. *Black Hat Briefings* 4 (2009), 158.
[28] Serhan Gül, Dimitri Podborski, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge. 2020. Low-Latency Cloud-Based Volumetric Video Streaming Using Head Motion Prediction. 27–33.
[29] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens Van Der Maaten. 2017. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117* (2017).
[30] Qingying Hao, Licheng Luo, Steve TK Jan, and Gang Wang. 2021. It's not what it looks like: Manipulating perceptual hashing based applications. In *ACM Conference on Computer and Communications Security (CCS)*. 69–85.
[31] RI Hg, Petr Jasek, Clement Rofidal, Kamal Nasrollahi, Thomas B Moeslund, and Gabrielle Tranchet. 2012. An RGB-D database using Microsoft's Kinect for Windows for Face Detection. In *International Conference on Signal Image Technology and Internet Based Systems*. 42–46.
[32] Suman Jana, David Molnar, Alexander Moshchuk, Alan Dunn, Benjamin Livshits, Helen J Wang, and Eyal Ofek. 2013. Enabling fine-grained permissions for augmented reality applications with recognizers. In *USENIX Security Symposium (Security)*. 415–430.
[33] Suman Jana, Arvind Narayanan, and Vitaly Shmatikov. 2013. A scanner darkly: Protecting user privacy from perceptual applications. In *IEEE Symposium on Security and Privacy (S&P)*. 349–363.
[34] Pavel Korshunov, Claudia Araimo, Francesca De Simone, Carmelo Velardo, J-L Dugelay, and Touradj Ebrahimi. 2012. Subjective study of privacy filters in video surveillance. In *International Workshop on Multimedia Signal Processing (MMSP)*. 378–382.
[35] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).
[36] Jean Le Feuvre, Cyril Concolato, Nassima Bouzakaria, and Viet-Thanh-Trung Nguyen. 2015. MPEG-DASH for low latency and hybrid streaming services. In *ACM international conference on Multimedia (MM)*. 751–752.
[37] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac. 2007. GPAC: open source multimedia framework. In *ACM international conference on Multimedia (MM)*. 1009–1012.
[38] Kiron Lebeck, Kimberly Ruth, Tadayoshi Kohno, and Franziska Roesner. 2017. Securing augmented reality output. In *IEEE Symposium on Security and Privacy (S&P)*. 320–337.
[39] Jingjie Li, Amrita Roy Chowdhury, Kassem Fawaz, and Younghyun Kim. 2021. Kalεido: Real-time privacy control for eye-tracking systems. In *USENIX Security Symposium (Security)*. 1793–1810.
[40] Yan Li, Yingjiu Li, Qiang Yan, Hancong Kong, and Robert H. Deng. 2015. Seeing your face is not enough: An inertial sensor-based liveness detection for face authentication. In *ACM Conference on Computer and Communications Security (CCS)*. 1558–1569.
[41] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. Sphereface: Deep hypersphere embedding for face recognition. In *IEEE conference on computer vision and pattern recognition (CVRP)*. 212–220.
[42] Shiqing Luo, Anh Nguyen, Chen Song, Feng Lin, Wenyao Xu, and Zhisheng Yan. 2020. OcuLock: Exploring human visual system for authentication in virtual reality head-mounted display. In *The Network and Distributed System Security Symposium (NDSS)*.
[43] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
[44] Florian Mathis, John H Williamson, Kami Vaniea, and Mohamed Khamis. 2021. Fast and secure authentication in virtual reality using coordinated 3D manipulation and pointing. *ACM Transactions on Computer-Human Interaction (ToCHI)* 28, 1 (2021), 1–44.
[45] Yue Ming and Xiaopeng Hong. 2016. A unified 3D face authentication framework based on robust local mesh SIFT feature. *Neurocomputing* 184 (2016), 117–130.
[46] Abdelmalik Ouamane, Mebarka Belahcene, Abdelhamid Benakcha, Salah Bourennane, and Abdelmalik Taleb-Ahmed. 2016. Robust multimodal 2D and 3D face authentication using local feature fusion. *Signal, Image and Video Processing* 10, 1 (2016), 129–137.
[47] Norman Di Palo. 2018. How I implemented iPhone X's FaceID using deep learning in Python. https://towardsdatascience.com/how-i-implemented-iphone-xs-faceid-using-deep-learning-in-python-d5dbaa128e1d
[48] Fei Peng, Xiao-wen Zhu, and Min Long. 2013. An ROI privacy protection scheme for H. 264 video based on FMO and chaos. *IEEE Transactions on Information Forensics and Security* 8, 10 (2013), 1688–1699.
[49] Huy Phan, Yi Xie, Siyu Liao, Jie Chen, and Bo Yuan. 2020. CAG: A real-time low-cost enhanced-robustness high-transferability content-aware adversarial attack generator. In *AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 34. 5412–5419.
[50] Francesco Pittaluga, Aleksandar Zivkovic, and Sanjeev J Koppal. 2016. Sensor-level privacy for thermal cameras. In *International Conference on Computational Photography (ICCP)*. 1–12.
[51] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4422–4431.
[52] Feng Qian, Bo Han, Jarrell Pair, and Vijay Gopalakrishnan. 2019. Toward practical volumetric video streaming on commodity smartphones. In *International Workshop on Mobile Computing Systems and Applications (HotMobile)*. 135–140.
[53] Slobodan Ribaric and Nikola Pavesic. 2015. An overview of face de-identification in still images and videos. In *IEEE International conference and workshops on automatic face and gesture recognition (FG)*, Vol. 4. 1–6.
[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. 234–241.

[55] Kimberly Ruth, Tadayoshi Kohno, and Franziska Roesner. 2019. Secure multi-user content sharing for augmented reality applications. In *USENIX Security Symposium (Security)*. 141–158.

[56] Arash Shafiei, Cyril Concolato, and Jean Le Feuvre. 2013. DashCast, a live DASH streaming server. In *International Workshop on Multimedia Signal Processing (MMSP)*.

[57] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. 2019. First order motion model for image animation. *Advances in Neural Information Processing Systems* 32 (2019).

[58] Iraj Sodagar. 2011. The MPEG-DASH standard for multimedia streaming over the Internet. *IEEE MultiMedia* 18, 4 (2011), 62–67.

[59] Julian Steil, Marion Koelle, Wilko Heuten, Susanne Boll, and Andreas Bulling. 2019. Privaceye: privacy-preserving head-mounted eye tracking using egocentric scene image and eye movement features. In *ACM Symposium on Eye Tracking Research & Applications*. 1–10.

[60] Pu Sun, Yuezun Li, Honggang Qi, and Siwei Lyu. 2020. Landmark breaker: Obstructing Deepfake by disturbing landmark extraction. In *2020 IEEE International Workshop on Information Forensics and Security (WIFS)*. 1–6.

[61] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[62] Zhongze Tang, Xianglong Feng, Yi Xie, Huy Phan, Tian Guo, Bo Yuan, and Sheng Wei. 2020. VVSec code on GitHub. https://github.com/hwsel/vvsec.

[63] Zhongze Tang, Xianglong Feng, Yi Xie, Huy Phan, Tian Guo, Bo Yuan, and Sheng Wei. 2020. VVSec: Securing volumetric video streaming via benign use of adversarial perturbation. In *ACM International Conference on Multimedia (MM)*. 3614–3623.

[64] Robert Templeman, Mohammed Korayem, David J Crandall, and Apu Kapadia. 2014. PlaceAvoider: Steering first-person cameras away from sensitive spaces.. In *The Network and Distributed System Security Symposium (NDSS)*. 23–26.

[65] Suramya Tomar. 2006. Converting video formats with FFmpeg. *Linux Journal* 2006, 146 (2006), 10.

[66] Jeroen van der Hooft, Tim Wauters, Filip De Turck, Christian Timmerer, and Hermann Hellwagner. 2019. Towards 6DoF HTTP adaptive streaming through point cloud compression. In *ACM Multimedia Conference*. 2405–2413.

[67] Lingxiao Wang, Yali Li, and Shengjin Wang. 2018. Feature learning for one-shot face recognition. In *IEEE International Conference on Image Processing (ICIP)*. 2386–2390.

[68] Mika Westerlund. 2019. The emergence of Deepfake technology: A review. *Technology Innovation Management Review* 9, 11 (2019).

[69] Yi Xu, True Price, Jan-Michael Frahm, and Fabian Monrose. 2016. Virtual U: Defeating face liveness detection by building virtual models from your public photos. In *USENIX Security Symposium (Security)*. 497–512.

[70] Zirui Xu, Fuxun Yu, Chenchen Liu, and Xiang Chen. 2019. HAMPER: high-performance adaptive mobile security enhancement against malicious speech and image recognition. In *Asia and South Pacific Design Automation Conference (ASPDAC)*. 512–517.

[71] Chao-Han Yang, Jun Qi, Pin-Yu Chen, Xiaoli Ma, and Chin-Hui Lee. 2020. Characterizing speech adversarial examples using self-attention u-net enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3107–3111.

[72] Xiaowen Ying, Xin Li, and Mooi Choo Chuah. 2018. LiveFace: A multi-task CNN for fast face-authentication. In *International Conference on Machine Learning and Applications (ICMLA)*. 955–960.

[73] Fuxun Yu, Zirui Xu, Chenchen Liu, and Xiang Chen. 2019. Masker: Adaptive mobile security enhancement against automatic speech recognition in eavesdropping. In *Annual Design Automation Conference (DAC)*. 1–6.

[74] Hyunwoo Yu, Jaemin Lim, Kiyeon Kim, and Suk-Bok Lee. 2018. Pinto: Enabling video privacy for commodity IoT cameras. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1089–1101.

[75] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. LPIPS code on GitHub. https://github.com/richzhang/PerceptualSimilarity.

[76] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual Metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 586–595.

[77] Wei Zhang, S.S. Cheung, and Minghua Chen. 2005. Hiding privacy information in video surveillance system. In *IEEE International Conference on Image Processing (ICIP)*, Vol. 3. II–868.

[78] Bing Zhou, Zongxing Xie, and Fan Ye. 2019. Multi-modal face authentication using deep visual and acoustic features. In *IEEE International Conference on Communications (ICC)*. 1–6.

[79] IOhannes M Zmölnig. 2005. v4l2loopback - a kernel module to create V4L2 loopback devices. https://github.com/umlaeute/v4l2loopback.