SilhouetteTell: Practical Video Identification Leveraging Blurred Recordings of Video Subtitles

Guanchong Huang
School of Computer Science
University of Oklahoma
guanchong.huang@ou.edu

Song Fang
School of Computer Science
University of Oklahoma
songf@ou.edu

Abstract

Video identification attacks pose a significant privacy threat that can reveal videos that victims watch, which may disclose their hobbies, religious beliefs, political leanings, sexual orientation, and health status. Also, video watching history can be used for user profiling or advertising and may result in cyberbullying, discrimination, or blackmail. Existing extensive video inference techniques usually depend on analyzing network traffic generated by streaming online videos. In this work, we observe that the content of a subtitle determines its silhouette displayed on the screen, and identifying each subtitle silhouette also derives the temporal difference between two consecutive subtitles. We then propose *SilhouetteTell*, a novel video identification attack that combines the spatial and time domain information into a spatiotemporal feature of subtitle silhouettes. *SilhouetteTell* explores the spatiotemporal correlation between recorded subtitle silhouettes of a video and its subtitle file. It can infer both online and offline videos. Comprehensive experiments on off-the-shelf smartphones confirm the high efficacy of *SilhouetteTell* for inferring video titles and clips under various settings, including from a distance of up to 40 meters.

Keywords

Video inference, Subtitle analysis, Spatiotemporal feature extraction

1 Introduction

Video viewing has become increasingly popular. According to a survey with 528 unique respondents conducted in November 2022, people on average watch 17 hours of videos per week [69]. Video identification attacks aim to infer videos that victims are watching without authorization. They pose an increasing privacy threat, as an individual's video viewing history or preference may reveal their political, financial, and personal interests [19], and others may judge them based on such data [31], causing cyberbullying and discrimination. Moreover, scammers may use sensitive video viewing history to conduct blackmail, threatening to release it to victims' families, friends, coworkers, or social network contacts [24, 30]. In the United States, the Video Privacy Protection Act (VPPA) [15] was enacted in 1988 after Robert Bork's video rental history was published during his Supreme Court nomination, making it illegal to disclose video viewing history without the watchers' consent [1].

Proceedings on Privacy Enhancing Technologies 2026(1), 470–485

https://doi.org/10.56553/popets-2026-0024

A traditional and naive method to identify a video is video content matching (e.g., [44]). An adversary may steal frames or audio data from a video snippet (e.g., via shoulder surfing attacks or recording with a camera or microphone), using them to generate features (i.e., video fingerprints) to characterize the video clip. Such features are then matched with a database of video fingerprints built with known videos. Videos containing content similar to the snippet are thus identified. However, such attacks rely on direct views of the victim's screen by capturing a clear view of the video content or eavesdropping accompanying audio information.

Recent research focuses on developing video identification attacks by analyzing streaming video traffic. Each video generates a distinct traffic pattern due to its unique content. Such a mapping can be pre-built for inferring what video is currently streaming. Those attacks, however, often require the attacker to either compromise the router that the victim's device connects to [35, 36, 75] or infect the victim's device with malware (e.g., via rogue websites [64]) to capture streaming traffic. Both requirements impose practical hurdles, making such invasive attacks no longer viable, as modern networks usually adopt anti-malware tools and are also secured with a password unknown to the sniffer. Some attacks (e.g., [19, 43]) using passive network reconnaissance are non-invasive, while they require special equipment such as Universal Software Radio Peripheral (USRP) to measure traffic. Moreover, most video streaming services (e.g., Netflix [53]) offer offline viewing, generating no traffic and failing all existing traffic analysis based techniques.

Individuals often take precautions to protect video privacy, including preventing others from directly viewing the video content or hearing the audio, securing the video streaming network, or predownloading videos. In this paper, we investigate whether users are vulnerable to video identification attacks in general settings, after taking the aforementioned precautions. We discover it is possible to infer the video a victim is watching, whether online or offline, by pointing a single RGB camera at the victim's screen from a distance and obtaining the subtitle silhouettes of the video. We refer to the proposed attack as *SilhouetteTell*. The silhouette (e.g., height or line count) of the video's subtitle on screen varies with the subtitle content. Such information can be captured by a camera at a distance, where the video frames, audio, and subtitles are completely illegible. The obtained coarse information related to subtitles can be exploited to infer videos, as shown in Figure 1.

Subtitles are text representing the contents of the audio in a video, often displayed with each scene at the bottom of the screen. They are necessary for people with hearing impairments, facilitating following the dialog. Also, translated subtitles are necessary for media in a foreign language [26, 42]. According to a survey conducted in 2023, 63% of young adults under 30 prefer watching

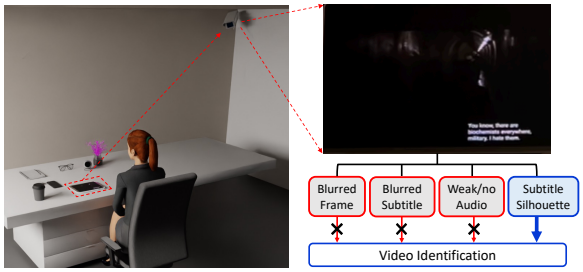

Figure 1: Subtitle silhouette information for inferring videos.

videos with subtitles even in a language they know, and only 27% choose subtitles off [20]. Another study of 5,616 participants shows that 69% view video with sound off in public places and 25% watch with sound off in private places, and instead, they turn on subtitles for understanding videos in these scenarios [49]. It has been well known that subtitles can be utilized to classify movie genres [58], improve movie recommender systems [29], and retrieve relevant videos from a large corpus [45]. However, in our general threat model, due to the long shooting distance and non-specific viewing angle, the attacker cannot record the content of the video that the victim is watching, nor can they obtain the subtitle content. Typically, when a user is watching videos that they wish to keep private, they tend to take actions, such as adjusting the screen away from the bystander. However, if others are too far away to see the screen, they often pay less attention and become less vigilant.

Intuitively, Optical Character Recognition (OCR) algorithms may help recognize subtitles, by converting optically sensed document text (i.e., typed, handwritten, or printed text in images) into machine-encoded text [52]. We pick two most popular and state-of-the-art algorithms, Tesseract OCR Engine [66] and Convolutional Recurrent Neural Network (CRNN) [65], which are deep learning-based and open-source. They both fail due to the poor recording quality. For the frame in Figure 1, the subtitle is: "*You know, there are*" (first line), "*biochemists everywhere,*" (second line), and "*military. I hate them.*" (third line). By running the latest Tesseract version 5.0 [12] to this frame, the recovery result is "*vou eon few ow*" (first line), "*Scien*" (second line), and "*rata | hate Pam*" (third line). Also, with the current CRNN implementation [3], the recognition output of the frame is nothing. Incorrect or empty extraction shows the inability of OCR methods to recover subtitles from blurry recordings.

We also wonder whether image captioning techniques, which generate natural language descriptions of images [41], can gather useful information for inferring videos. Vision-language pre-training (VLP) is currently the dominant training method using pre-trained large-scale models for visual recognition tasks [32]. One widely used pre-trained model is CLIP (Contrastive Language-Image Pre-Training) [55]. We utilize ClipCap [50], a recent work based on CLIP and a pre-trained language model (GPT-2 [56]), to generate captions for images. Similar to the challenges faced by OCR methods in recognizing blurry subtitles, it becomes evident that no correct caption can be produced for those blurry recordings. For example, in Figure 1, the true caption is "Two men are talking to each other", while the result of applying the ClipCap implementation [2] is "a scene from the movie", providing no help in video identification.

Traditional ML-based techniques (e.g., OCR, CRNN, and ClipCap) have inherent limitations in recognizing subtitle or image content from blurry frames captured at a distance. In contrast, contentless subtitle silhouettes are much easier to distinguish under such conditions. Building on this observation, *SilhouetteTell* extracts spatial and temporal features for inferring videos. First, we normally can observe a seemingly "continuous white area" in each video frame when a subtitle is present. We refer to this area as the "subtitle area". Different lengths of subtitles may result in subtitle areas with different shapes (silhouettes). This allows us to compare line counts among subtitles. Meanwhile, the duration of a subtitle may span multiple frames, that is, the subtitle remains unchanged while the video scene changes. We extract the line count of each subtitle as its spatial feature. Second, identifying subtitle areas for each frame helps derive the temporal difference between two successive frames. As aforementioned, the shape of the subtitle area varies with the subtitle content. However, for some subtitles with similar lengths, the shapes of their subtitle areas would be quite similar. For such cases, we observe that the overall brightness (i.e., the sum of image pixel values) of subtitle areas would differ as the subtitle contents change. Accordingly, we utilize both the shape and brightness of subtitle areas to distinguish whether the subtitle changes across neighboring frames. Such pattern variation of subtitle areas over time is referred to as *temporal feature* of subtitles, which discloses how long (or how many frames) a subtitle appears on the screen.

SilhouetteTell initially needs to extract correct subtitle silhouettes in blurry videos, from where we barely identify any texts. As a blurry subtitle in videos recorded from a far distance often appears as a white chunk, we change the problem from recognizing texts to localizing white areas for obtaining subtitle areas, and design a subtitle silhouette extraction scheme using Mask Region-based Convolutional Neural Network (Mask R-CNN), a deep learning technique that can segment and identify the pixel-wise boundaries of each object [38]. Moreover, an attacker must compare the correlations among subtitles with those among observed subtitle silhouettes. This requires a self-contained spatiotemporal feature that can quantify such correlations and be compared against others. We create such a feature and corresponding approaches to compare observed subtitle silhouettes to possible candidate videos using this feature. Our technique incorporates a mechanism to retain high accuracy in the presence of subtitle silhouette recognition errors.

In summary, we mainly make the following contributions.

- We propose a new type of video identification attack using only video captured from a distance through commodity phone cameras. Our method does not require online video viewing, professional equipment, or a connection to the same network as the playback device.
- We develop an algorithm to map obtained contentless subtitle silhouettes in recorded blurry videos into a video clip, and achieve video inference by modeling, extracting, and correlating their self-contained spatiotemporal features.
- Extensive real-world experiments on top of 300 movies from YouTube, Amazon Prime Video, and Netflix, show that with a 2-minute blurry video clip, the average probabilities of recognizing videos in the top 1, 5, and 10 candidates can reach as high as 91.1%, 98.7%, and 99.8%, respectively.

2 Preliminaries

Subtitles: A subtitle, in the form of one or more lines of written text, typically appears at the bottom center of the screen in sync with a video's audio [23, 34]. It may appear in a rectangular overlay with a colored or opaque background.

Subtitles can be open or closed. Open subtitles are embedded in the video and cannot be turned off, whereas closed subtitles are on a separate text track and can be turned on/off by the viewer. Subtitles can be downloaded in specific formats, e.g., SubRip text (.srt), Web Video Text Tracks Format (.vtt), and SubStation Alpha (.ssa) [74]. The .srt format, as one of the most popular and almost a de facto standard in the web [37, 74], contains the subtitle index, start time, end time, and text content. In a .srt file, subtitles are numbered sequentially, and the timecode format used is hours:minutes:seconds, milliseconds, with time units fixed to two zero-padded digits and fractions fixed to three zero-padded digits (00:00:00,000).

Mask R-CNN: Object detection deals with the task of segmenting instances of semantic objects in digital images and videos. We input an image to an object detection model, which generally outputs coordinates of bounding boxes in the input image that contains specific objects. Faster R-CNN [60] makes a series of improvements on the initial object detection algorithm of Regions with CNN features (R-CNN) [33]. Mask R-CNN is an extension of Faster R-CNN and augments object detection by adding object segmentation.

3 Threat Model and Assumptions

We consider a general user (victim) who might be vulnerable while watching subtitled videos on computers/tablets/phones in open places such as libraries, airports, cafes, and cubicles. An adversary records the scenario where the victim is watching videos from a distance via a single RGB camera (e.g., a phone camera) and processes the recorded blurry video to identify videos the victim watches. An attacker is assumed to have access to the victim's space, either physically or remotely, to monitor their screen activity.

No Traffic Access: Our work differs significantly from prior work (e.g., [19, 28, 35, 43, 46, 59]) in that we do not rely on traffic information of video streaming sessions. *SilhouetteTell* thus works even in scenarios where the victim watches videos offline, while all previous traffic analysis based video identification attacks fail.

No User-specific Training and Directly Observing Video Content: We assume that the attack is opportunistic, requiring no labeled data or prior observation of the victim, and the victim is alert to conventional shoulder-surfing attacks in the sense that the attacker cannot get too close to them when watching a video.

Recording Required: While directly observing an individual may sometimes reveal sensitive information (e.g., their lifestyle), such inferences typically require prolonged observation and may be ineffective if the victim does not outwardly exhibit personal traits. The attacker does not need her recordings to contain the full screen of the target device on which the victim watches the video, while we assume that the attacker can observe at least a partial blurry view of the screen containing the subtitles. Such scenarios are quite common in practice as it is challenging for the victim to block observing the screen from all angles.

Subtitle Library: We also assume that the attacker has access to the subtitles of all videos in the suspect set. Such knowledge can be easily acquired as subtitle files are often available and downloaded from subtitle websites (e.g., [11]) or video streaming services. Appendix A discusses specific methods for crawling subtitle files.

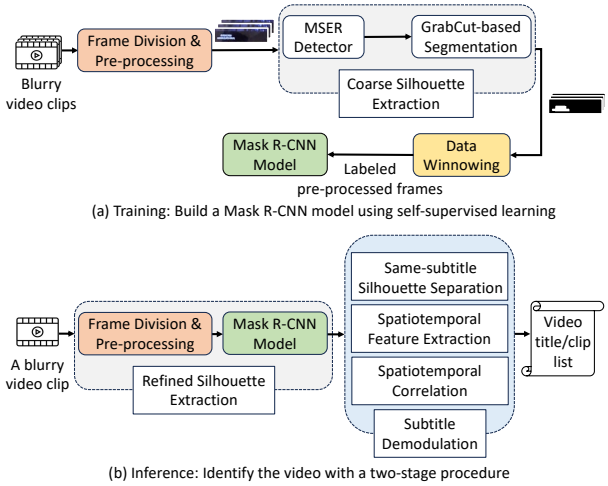

Figure 2: Overview of *SilhouetteTell*.

4 Video Inference Attack

4.1 Attack Overview

SilhouetteTell performs a two-phase process to infer videos from observed blurry videos: *training* and *inference*.

Figure 2 plots an overview of *SilhouetteTell*. Figure 2a depicts the offline training phase, where a Mask R-CNN model is trained for accurately detecting subtitle silhouettes. The attacker first divides the recorded video into frames, and pre-processes them to focus on the subtitle areas on the victim's screen. The pre-processed frames are input into a module of *Coarse Silhouette Extraction*, built on Maximally Stable Extremal Regions (MSER) [48] and GrabCut [62], to obtain raw training data, which are then sanitized via a module of *Data Winnowing*. Figure 2b shows the inference phase, consisting of two modules, *Refined Silhouette Extraction* and *Subtitle Demodulation*. For the initial module, the attacker obtains pre-processed frames using the same procedure in the training phase, and inputs them to the trained Mask R-CNN model. In the second module, we separate the captured silhouettes originating from different subtitles, and obtain a silhouette sequence. Each element of the sequence belongs to a corresponding distinguishable subtitle. Next, we extract the spatiotemporal feature of the silhouette sequence, and correlate it with a subtitle sequence presented in subtitle files, shrinking the candidates of the target video title and clip.

4.2 Training Phase

4.2.1 Frame Division & Pre-processing. We divide the recorded video clip into individual raw video frames using OpenCV, an open-source library for computer vision [6]. A frame may record the background beyond the victim's screen, where certain video scenes may confuse the subtitle area recognition. Accordingly, we perform a dual-cropping operation on each frame uniformly to eliminate interference and only include the area where subtitles may appear.

First-cropping: If the victim's screen is not moved in the recording, we can mark the screen's four corners to designate the portion of the

video frame and clip the corresponding area. If the screen's location varies in the recording (e.g., when the recording camera is not stationary), the manual cropping may be laborsome. Alternatively, we use an existing object detection method (i.e., a pre-trained Mask R-CNN model [17]) to crop out the screen area efficiently.

Perspective Transformation: We may record the screen from different angles. Thus, the subtitle area in the raw recording may be skewed or compressed. To reduce such effects, we apply perspective transformation to convert an original screen recording image into the plane of the screen (i.e., the bird's eye view) before applying the second cropping. Specifically, we first mark the 4 points on the video to indicate the target screen's planar surface. We later compute a homography matrix **H** between this planar surface and the video frame's perspective, and finally transform the source image with the matrix **H**, using two OpenCV functions, $perspectiveTransform$ and $warpPerspective$ [9]. Let (x, y) denote the 2D coordinate of a point in the source image. By multiplying (x, y) with **H**, we obtain its corresponding point in the plane of the screen.

Second-cropping: Occasionally, the screen may have regions that appear similar to a subtitle area. Minimizing such similarity would be beneficial for recognizing subtitle areas. Empirically, we observe that the subtitles of videos for popular video streaming services (e.g., YouTube, Netflix, and Amazon Prime Video) are usually displayed in the bottom quarter of the screen. We then crop this area as the input to the next step for subtitle silhouette extraction. If subtitles appear in other parts of the screen, the second cropping will target that area, adapting to corresponding subtitle displacements.

4.2.2 Coarse Silhouette Extraction. This step aims to extract subtitle silhouettes leveraging the MSER feature extractor and GrabCut-based segmentation. Such a process still suffers from segmentation errors, and we refer to it as coarse silhouette extraction.

MSER Feature Extractor: In the pre-processed frames, normally, the subtitle area and non-subtitle area (i.e., background) differ in color, while within each area, the color is nearly constant. MSER is a blob detector that finds the stable connected regions in a gray-scale image over a wide range of thresholds. To transfer an RGB image into a gray-scale one, we utilize an OpenCV function $cvtColor$ [7], which represents each pixel in the original image with a single grayscale intensity according to its RGB values. The intensity is denoted with an 8-bit integer giving 256 possible different shades of gray from black to white. After applying MSER to each pre-processed frame (with the grayscale format) with another OpenCV function - $MSER\_create$ [8], a sequence of rectangular bounding boxes will be obtained. Next, for each bounding box, we calculate the corresponding edge-to-edge distances of the pre-processed frame and this box, resulting in four distances. If any of the distances is smaller than an empirically pre-defined threshold, we determine that this box is near the edge of the pre-processed frame, and then filter it out, as it usually may not contain the subtitle.

To select the most relevant bounding box corresponding to the subtitle area, we propose a bounding box selection algorithm based on the Non-Maximum Suppression (NMS) method [40, 54]. Let A and B denote two bounding boxes. Their Intersection over Union (IoU) is the ratio between the area of overlap to the area of union between A and B, i.e., $\frac{A \cap B}{A \cup B}$. We use the standard IoU threshold of 0.5 [40] and perform the following steps. First, a confidence score μ for each bounding box is defined as the ratio of its height to its width. We sort the boxes in descending order of μ, and denote the one with the highest μ as H. Empirically, we find H usually recognizes the object more accurately than the rest. Next, we compute IoU of H with every remaining box. Let C denote one of the remaining boxes. If IoU of H and C is greater than 0.5, we remove C. After traversing all boxes, if it ends up with two or more unique ones, we observe this case often appears when there are some small bounding boxes encapsulating inference of white points, rather than the subtitle. Accordingly, we keep the one with the largest size.

GrabCut-based Segmentation: The next step is to obtain subtitle silhouettes from the bounding boxes. GrabCut is often used to segment the foreground of an image from the background, while it is required to first manually frame and select the target area [62]. Instead, we utilize the module of the MSER feature extractor to automatically generate the input of GrabCut, replacing the low-efficiency process of manually choosing the segmentation range. Generally, for each bounding box, the segmentation results using an OpenCV function - $grabCut$ [5], return the subtitle silhouette as the foreground and the rest as the background.

4.2.3 Data Winnowing. The results of coarse silhouette extraction may contain errors. We thus perform a consistency check to see whether the segmented subtitle silhouettes are correct. Particularly, we winnow out the pre-processed frames, whose subtitle silhouettes are incorrectly recognized. Meanwhile, the rest pre-processed frames, together with their corresponding detected subtitle silhouettes, become training data for building a Mask R-CNN model.

4.2.4 Mask R-CNN Model Building. We utilize a commercial smartphone (Samsung Galaxy Z Fold4) to record a laptop playing movies on three streaming services (YouTube, Amazon Prime Video, and Netflix) at distances larger than 4 meters. In these recordings, both the movie and the subtitle contents appear blurry.

Dataset Splitting: For each streaming service, we select 10 popular movies, and for every movie, we randomly record a 2-minute clip with 60 FPS for processing. As a result, we have a total of $3 \times 10 \times 120 \times 60 = 216,000$ raw video frames as input. The coarse silhouette extraction recognizes 109,326 frames that contain subtitle areas. After applying data winnowing, we obtain 78,591 pieces of correctly labeled data (referred to as *silhouette-contained data*). Also, among falsely labeled data, we obtain 3,244 pieces, in each of which the corresponding pre-processed frame actually has no subtitle silhouette. For such data, we replace the incorrect original label with a segmentation result of no subtitle silhouette, and refer to these revised data as *silhouette-free data*. The only component that requires training is the Mask R-CNN model. With the two obtained datasets, silhouette-contained and silhouette-free, we split each of them using the common 80:20 train-test ratio for fine-tuning a Mask R-CNN model. We use ResNet50 [38, 39] as the backbone and stochastic gradient descent (SGD) [22] for optimization.

4.3 Inference Phase

4.3.1 Refined Silhouette Extraction. Compared with coarse silhouette extraction, refined silhouette extraction can achieve higher accuracy. We verify this with a new dataset. For this dataset, we randomly select 15 new movies (not shown in the training dataset)

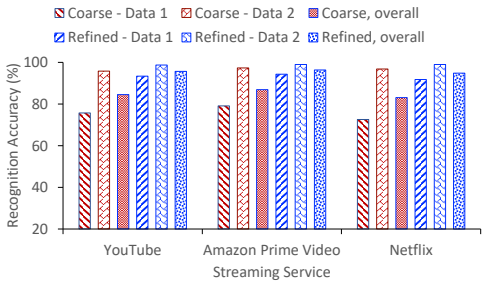

Figure 3: Performance of coarse and refined silhouette extraction (denoted as "Coarse" and "Refined") for silhouette-contained (Data 1) and silhouette-free frames (Data 2).

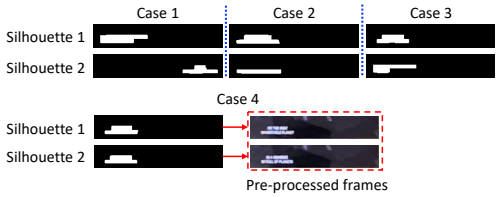

Figure 4: Silhouette comparison: entirely separate (case 1); partially overlap with varying or the same line counts (case 2/3); fully overlap and check pre-processed frames (case 4).

from the three streaming services, with 5 for each. We then record a randomly picked 2-minute video for every movie with 60 FPS. In total, we have $15 \times 2 \times 60 \times 60 = 108,000$ frames. This dataset is only used to validate the superior performance of the refined silhouette extraction method over the coarse one.

Figure 3 compares the recognition rates of coarse and refined silhouette extraction methods applied to silhouette-contained and silhouette-free frames. We can observe that the refined silhouette extraction method consistently achieves higher recognition accuracy compared to the coarse one. Also, it always maintains high recognition accuracy regardless of the streaming service or whether the input frame has a subtitle. Specifically, the overall recognition accuracy for refined silhouette extraction is consistently above 95%, whereas for coarse silhouette extraction, this value is consistently below 86%. These results convincingly demonstrate the necessity of building a Mask R-CNN model for recognizing subtitle silhouettes. Also, Appendix B shows example cases where refined silhouette extraction works while coarse silhouette extraction fails.

4.3.2 Subtitle Demodulation. Subtitle demodulation converts a sequence of consecutive subtitle silhouettes into corresponding subtitles of a video, enabling video identification. We start by distinguishing silhouettes originating from the same subtitle and designing a spatiotemporal feature to be applied to a subtitle silhouette sequence. This feature must be suitable for narrowing down the search space of possible candidates. Subsequently, we demonstrate how to apply this feature to multiple subtitle silhouette sequences.

Same-subtitle Silhouette Separation: The same subtitle usually generates highly similar silhouettes at the same positions of a frame. If two silhouettes are different, they normally come from different subtitles. Figure 4 presents typical scenarios involving two consecutive silhouettes. Intuitively, it is easy to determine that in the first three cases, the two silhouettes, which either have no overlap or partially overlap, come from different subtitles. Specifically, to reduce the impact of the no-silhouette area, we place two marked silhouettes S_1 and S_2 into the same frame, and then compute their IoU, which is denoted as $IoU_s = \frac{S_1 \cap S_2}{S_1 \cup S_2}$, ranging from 0 to 1.

Let IoU_s^{diff} and IoU_s^{same} denote the IoU of two neighboring silhouettes coming from different and the same subtitles. To explore their practical values, we randomly select 10,000 pairs of neighboring silhouettes coming from the same and different subtitles, respectively. Figure 5 plots the empirical cumulative distribution functions (CDFs) of IoU_s^{diff} and IoU_s^{same}. We see that IoU_s^{same} is always near 1 (above 0.93), while IoU_s^{diff} is less than 0.93 with a probability of 97%. Thus, with an IoU threshold of 0.93 for distinguishing silhouettes, the true positive (i.e., silhouettes originating from the same subtitle are correctly identified) and true negative (i.e., silhouettes from the different subtitle are correctly identified) would be 100%, while the false positive (silhouettes from different subtitles are incorrectly identified as from the same subtitle) is 3%. This appears as the IoU-based method cannot distinguish between two silhouettes from different subtitles when they have similar sizes and appear at similar locations. For example, in case 4 of Figure 4, both silhouettes are similar with IoU equaling 0.95, while the corresponding subtitles differ. We further compare perspective pre-processed frames associated with them. This is based on the observation that different subtitles have distinct words and sentence structures, leading to variation in pixel value distribution in subtitle areas (i.e., difference in pre-processed frames).

When the IoU of two neighboring silhouettes is above or exceeds the threshold, we first put two marked silhouettes into the same frame, and identify the area of intersection. We then extract the portion marked by this intersection area in each of the two corresponding pre-processed frames. We refer to the extracted two regions as A_1 and A_2, respectively, and calculate the absolute difference between every pixel in A_1 and the corresponding pixel in A_2. Finally, we sum all such pixel differences and denote the sum as $\mathcal{P}$. We set up another threshold $\mathcal{T} = \mathcal{T}_0 \cdot N$, where N is the number of pixels in A_1 or A_2 and $\mathcal{T}_0$ indicates the mean maximum allowable pixel variance along two neighboring frames for the same subtitle. We introduce a new metric, called *similarity ratio*, and denoted as $R = \frac{\mathcal{P}}{\mathcal{T}}$ to compare $\mathcal{P}$ and $\mathcal{T}$. If $R > 1$, we have $\mathcal{P} > \mathcal{T}$, and regard that the silhouettes originate from different subtitles; otherwise, when $0 \leq R \leq 1$, they are determined as from the same subtitle.

Empirically, we choose $\mathcal{T}_0 = 1.5$, achieving high accuracy. We also pick another 10,000 pairs coming from different subtitles while each such pair has an IoU larger than 0.93. Figure 6 presents the CDFs of R_{same} and R_{diff} that denote the similarity ratio for a pair of neighboring silhouettes coming from the same and different subtitles, respectively. We can see that R_{same} is always below 1 while R_{diff} is consistently above 1, indicating a 100% success rate in distinguishing silhouettes originating from the same or different subtitles. Particularly, for case 4 in Figure 4, we have $\mathcal{T} = 17,145$, and $\mathcal{P} = 20,298 > \mathcal{T}$, where the two silhouettes are correctly determined as coming from different subtitles.

Spatiotemporal Feature Extraction: Ideally, the feature extracted from a subtitle silhouette sequence would enable us to uniquely determine the video clip (and thus the video title). Suppose the

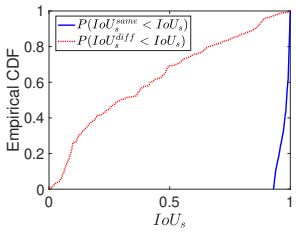

Figure 5: CDFs of IoU of two silhouettes.

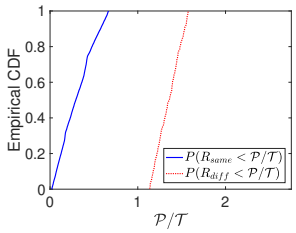

Figure 6: CDFs of similarity ratio of two silhouettes.

duration of the recorded clip is T and we can accordingly extract C clips with a period of T within the suspect library of videos. A perfect feature would classify the C video clips into C groups, each having one member only, such that an input subtitle silhouette sequence can uniquely match a video clip based on this feature. Our strategy is thus to figure out a feature that can divide all video clips into as many groups as possible, to achieve high distinguishability.

To quantify the distinguishability of a feature in dividing all candidates, we define *uniqueness score* as a new metric, as the ratio C_s/C_0, where C_0 is the number of considered video clips and C_s represents the number of sets obtained by dividing C_0 video clips with the selected feature. The uniqueness score should be maximized for the best partitioning of the video clips. To calculate the number of sets divided by each, we randomly select 10,000 different clips with a duration of T from 100 movies streaming on YouTube, Amazon Prime Videos (Amazon for short), and Netflix respectively. We vary T from 1 to 3 minutes, in increments of half a minute.

Temporal Feature Representation: From a temporal perspective, the intuitive feature of a video clip is the number of subtitles displayed, i.e., the length of the subtitle silhouette sequence. The sequence length can be obtained by counting the number of varying subtitle silhouettes within the period of the video clip since two successive frames showing the same subtitle would exhibit the same subtitle silhouettes. We find that all selected 2-minute 10,000 video clips for Netflix are 1-94 subtitles long. If we choose the sequence length as the only feature, we can divide all suspect Netflix video clips into 94 sets. On average, each set has $10,000/94 \approx 106.4$ video clips. The uniqueness score is then $94/10,000 = 0.0094$.

Spatial Feature Representation: From a spatial perspective, the shapes of subtitle silhouettes disclose how they differ in terms of their width or height. Normally, the video subtitles have no more than three lines, avoiding obstructing the video scene. Also, the height of a subtitle silhouette is proportional to the line count. Thus, we use it to characterize each subtitle silhouette. Silhouettes with similar heights are clustered in the same group. Next, we sort and index all groups in ascending order of their heights. We denote the height similarity information of a silhouette sequence as $[s_1, \cdots, s_r]$, where r is the number of distinct silhouette height groups that appear, and s_i ($i \in \{1, \cdots, r\}$) denotes how many times a silhouette in the group with the i^{th} smallest index appears.

Considering that a subtitle has up to three lines, we have up to three categories according to the silhouette height. For instance, for a sequence of 5 subtitles whose lines are 2, 2, 1, 2, and 1, respectively, its height similarity information is [2, 3], as there are two different line counts, with the smaller one appearing twice and the other appearing three times. Using the height similarity information, we

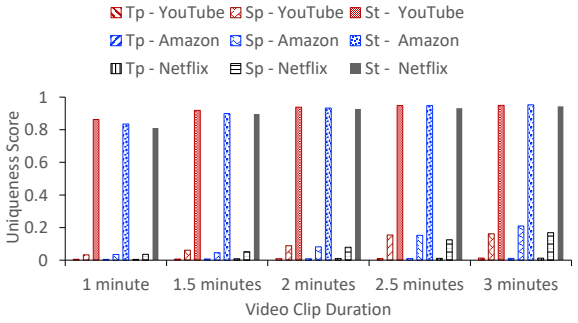

Figure 7: Uniqueness scores of video clips.

can divide all selected 10,000 2-minute Netflix video clips into a total of 789 sets. The uniqueness score equals $789/10,000 = 0.0789$.

Spatiotemporal Feature Characterization: The height similarity information has better distinguishability than the length feature, as its larger uniqueness score yields a smaller set cardinality, and hence a reduced search space to map an input silhouette sequence to a video clip. The feature of height similarity information gives the statistics of subtitles with varying lines/heights, while it does not consider their exact positions in the sequence. We expect a more advanced feature can further increase the uniqueness score by indicating both the length and height similarity information of the silhouette sequence, as well as the height variation from silhouette to silhouette. Let $\mathbf{s} = [s_1, s_2, \cdots, s_n]$ denote a sequence of n different subtitles or subtitle silhouettes within a video clip of a period of T. We define its *spatiotemporal vector* as

$$V : \mathbf{s} = [s_1, s_2, \cdots, s_n] \longmapsto \mathbf{l} = [l_1, l_2, \cdots, l_n], \tag{1}$$

where l_i ($i \in \{1, 2, \cdots, n\}$) denotes the mapping of s_i in the spatiotemporal vector. To construct V, the following steps are performed.

- We first cluster n subtitles or silhouettes based on the line count of subtitles or the silhouette's height, and thus obtain r sets. Subtitles with the same line count, or silhouettes with comparable height, aggregate into a separate set.
- Each set is associated with a line count or height as a label. Sort r sets based on this label and index them from 1 to r.
- Finally, s_i ($i \in \{1, 2, \cdots, n\}$) will be mapped into the index of the set that s_i belongs to, i.e., l_i.

We build the spatiotemporal vector for each video clip and ultimately partition the 10,000 2-minute Netflix video clips into 9,271 sets. The corresponding uniqueness score is $9,271/10,000 = 0.9271$, much larger than those of the previously discussed two features.

Empirically, we find that the uniqueness scores for video clips of varying durations are not evenly distributed. Figure 7 compares the uniqueness scores when we search candidates using sequence length (i.e., temporal feature or "Tp"), height similarity (i.e., spatial feature or "Sp"), and spatiotemporal vector (i.e., spatiotemporal feature or "St"). For all cases, the spatiotemporal vector greatly outperforms the other two. There is no significant difference in uniqueness scores among different streaming services. However, it is evident that as the video clip becomes longer, it becomes more uniquely structured, resulting in a higher uniqueness score for each feature. For example, with the spatiotemporal vector as the feature to divide chosen Netflix videos, the uniqueness score for a one-minute video clip is 0.81, while that for a three-minute video clip

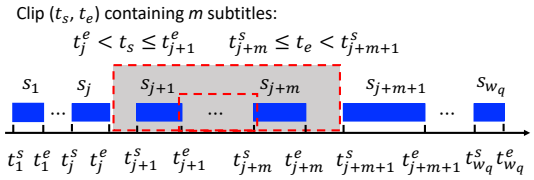

Figure 8: A sliding window of m subtitle.

reaches 0.95. Across all streaming services, video clips of 2 minutes or longer, maintain uniqueness scores above 0.93.

Spatiotemporal Correlation: Suppose the suspect library has K videos. With the subtitle file of the q^{th} ($q \in \{1, 2, \cdots K\}$) video, we can obtain its subtitle sequence $[s_1, s_2, \cdots, s_{w_q}]$, which w_q denotes the video's total amount of subtitles, as well as the starting and ending time points of each subtitle, i.e., t_i^s and t_i^e ($i \in \{1, 2, \cdots, w_q\}$). For a recorded blurry video clip with a duration of T, its subtitle silhouette sequence is denoted with $\mathbf{S} = [S_1, S_2, \cdots, S_m]$, where m denotes the total number of silhouettes. We can then obtain its spatiotemporal vector as $\mathbf{V}_o = V(\mathbf{S})$. Let C represent the demodulated set containing the candidates for this recording.

We utilize a sliding window of m subtitles, with a step size of one subtitle, applied to the entire subtitle sequence of each video. Thus, the q^{th} video will be divided into $w_q - m + 1$ segments, and each can be denoted with $\mathbf{G} = [s_{j+1}, s_{j+2}, \cdots, s_{j+m}]$ ($j \in \{0, 1, \cdots, w_q - m\}$). Figure 8 illustrates how we extract a segment with m subtitles, i.e., $s_{j+1}, \cdots, s_{j+m}$. The starting and ending time points are denoted with t_s and t_e. Accordingly, we obtain the following two constraints for them, $t_j^e < t_s \le t_{j+1}^e$, and $t_{j+m}^s \le t_e \le t_{j+m+1}^s$. Particularly, we let t_0^e and $t_{w_q+1}^s$ be 0 and T, indicating the start and end of the video, respectively. Beginning with searching from the first suspect video, we perform the following steps to decode the recorded blurry video. Initially, we set $q = 1$, $j = 0$, and $C = \emptyset$.

(1) When the time constraints for the recording are not satisfied, i.e., $t_{j+m+1}^s - t_j^e < T$ or $t_{j+m}^s - t_{j+1}^e > T$, this m-subtitle window has no way to exhibit an m-silhouette sequence of duration T. We will thus skip this no-candidate segment.
(2) When the two time constraints are satisfied, we calculate the spatiotemporal vector of $\mathbf{G}$ as $\mathbf{V}_c = V(\mathbf{G})$. If $\mathbf{V}_c$ matches with $\mathbf{V}_o$, this segment $\mathbf{G}$ would be a candidate and we add it into the set C; otherwise, we skip this segment.
(3) If j equals w_q, jump to step (4); otherwise increase j by 1, and jump to step (1).
(4) If $q < K$, we increment q by 1, reset $j = 1$, and jump to step (1); otherwise, return.

With the video clip candidates in C, we can then retrieve their video titles to achieve video identification.

Joint Demodulation: While watching a video, the victim may move their body to cover partial or all screen, leading to the attacker not recording the subtitle area on the victim's screen. Also, a pedestrian or obstacle may block the line-of-sight between the attacker and the victim's screen. Under these circumstances, the attacker may miss recording certain subtitle silhouettes, resulting in obtaining several discontinuous clips. In a general case, we assume that the attacker obtains M clips, denoted as $\mathcal{A} = [\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_M]$. The duration of these M clips are represented by $T_1, T_2, \cdots$, and T_M. Also, the duration between two neighboring clips (referred to as the inter-sequence interval) is denoted by $T_{i,i+1}$ ($i \in \{1, 2, \cdots, M-1\}$). Our goal is to find a long video clip with duration $\sum_{i=1}^{M-1}(T_i + T_{i,i+1}) + T_M$ that corresponds to the M subtitle silhouette sequences. While each silhouette sequence could have several candidate short video clips with a matching structure, the combination of respective candidates for neighboring clips should satisfy the following two requirements: (1) both candidates come from the same video; (2) the interval between the two candidates within the video should match the inter-sequence interval observed by the attacker. Accordingly, the attacker first finds initial candidate short video clips for each subtitle silhouette sequence $\mathbf{S}_j$ ($j \in \{1, 2, \cdots, M\}$), denoted as $\mathbf{G}_j = [\mathbf{G}_j^1, \mathbf{G}_j^2, \cdots, \mathbf{G}_j^{N_j}]$, where N_j denotes the total amount of the candidates. The starting and ending time points for $\mathbf{G}_j^n$ ($n \in \{1, 2, \cdots, N_j\}$) are denoted with T_j^s and T_j^e. The attacker then iterates over all M clips with $j = 2$ initially, and performs the following steps, returning when $j > M$.

(i) Concatenate the candidates for $\mathbf{S}_{j-1}$ and $\mathbf{S}_j$, obtaining $N_{j-1} \cdot N_j$ candidates.
(ii) For each concatenated candidate, if $\mathbf{G}_j^n$ and $\mathbf{G}_{j-1}^m$ ($m \in \{1, 2, \cdots, N_{j-1}\}$) come from two different videos (i.e., with varying video titles), this candidate will be discarded; otherwise, we further compare $T_{j-1,j}$ and $T_j^s - T_{j-1}^e$.
(iii) If $|T_{j-1,j} - (T_j^s - T_{j-1}^e)| > \delta$, where δ denotes the sum of the durations of the last and first subtitles of $\mathbf{G}_{j-1}$ and $\mathbf{G}_j$, respectively, the candidate will also be discarded.
(iv) Update $\mathbf{S}_{j-1} = \mathbf{S}_{j-1} || \mathbf{S}_j$ (the concatenation of two sequences), and $\mathbf{G}_{j-1}^m$ with candidates survive steps (ii) and (iii).
(v) Increase j by 1 and jump to step (i).

4.3.3 Error Tolerance. The built Mask R-CNN may introduce errors in subtitle silhouette extraction due to model imperfections and interference from contents in the recordings. Such errors may further lead to inaccurate spatiotemporal vectors, generating invalid or even no candidates. We consider three types of errors: (i) *Type 1 (Substitution)*, which involves classifying silhouettes into inaccurate categories; (ii) *Type 2 (Deletion)*, where valid subtitles are missed; (iii) *Type 3 (Insertion)*, which extracts non-existed silhouettes.

For Type 1 errors, we handle them by adjusting the criterion of determining the consistency between spatiotemporal vectors ($\mathbf{V}_o$ and $\mathbf{V}_c$) of the target silhouette sequence and a potential candidate video clip. Particularly, for step (2) of Spatiotemporal Correlation, as presented in Section 4.3.2, we calculate the Euclidean distance d between the two spatiotemporal vectors, i.e., the square root of the sum of the squared differences between the two vectors. We can then set up a threshold d_0, and if $d > d_0$, both vectors are regarded as inconsistent; otherwise, they will be regarded as consistent. Thus, we can control d_0 to adjust the tolerance against type 1 errors. Empirically, we enable $d_0^2 = \lceil 0.1 \times |\mathbf{V}_o| \rceil$, always successively overcome substitution errors, where $|\mathbf{V}_o|$ denotes the total number of silhouettes, and $\lceil x \rceil$ denotes the ceil function, returning the smallest integer greater than or equal to x.

To handle Type 2 or 3 errors, we develop a heuristic solution by guessing and adding or deleting erroneous elements in the silhouette sequence. We assume there are D deletion errors and I insertion errors. For deletion errors, we consider that each missing

silhouette can be at any position in the sequence, and it can be divided into any cluster according to its height. When we add a missing silhouette into a silhouette sequence with length m, there are m+1 positions to choose from, and its mapping l_{miss} in the spatiotemporal vector of the new silhouette sequence will match with any value, i.e., $|l_{miss} - l| = 0$, where l denotes any value. Similarly, regarding insertion errors, we consider that the extra incorrectly recognized silhouette can be any element in the silhouette sequence. For each insertion error, we thus iteratively remove one element from the silhouette sequence, starting from the first and proceeding to the last. In general, D and I are bounded. We demodulate all resultant possible silhouette sequences and combine their returned results to form the set of final candidates.

4.3.4 Impact of User Operations. If the victim pauses the video and resumes later, a complete silhouette sequence can still be obtained, while the recording duration T no longer reflects the actual duration of uninterrupted video playback. The attacker can detect pauses (e.g., by comparing pixel differences of successive frames) and get the pause interval (T'). Next, the duration of the silhouette sequence will be updated as $T - T'$, eliminating the impact of pausing.

When the victim rewinds or fast-forwards a video from time t_1 to t_2, the attacker would obtain two separate video clips with respective silhouette sequences, one before time t_1, and one after t_2. The inter-sequence duration $T_{1,2} = t_2 - t_1$ no longer correctly reflects the duration between the two clips when the video plays normally. However, the candidates for the two silhouette sequences should belong to the same video. Also, for rewinding and fast-forwarding, the starting time of the second clip should be smaller and larger than the ending time of the first clip, respectively. Thus, we need to revise the joint demodulation algorithm. We first demonstrate each silhouette sequence. With each candidate pair ($\mathbf{G}_1$, $\mathbf{G}_2$), where $\mathbf{G}_i$ ($i \in \{1, 2\}$) is one candidate for the i^{th} silhouette sequence, we check whether both $\mathbf{G}_1$ and $\mathbf{G}_2$ belong to the same video. If not, we discard this candidate pair; otherwise, we continue to check whether the starting time of $\mathbf{G}_2$ and the ending time of $\mathbf{G}_1$ satisfy the above requirement. If not, we discard the candidate pair, otherwise, this candidate pair will be then added to the final candidate set.

5 Evaluation

We develop an app to implement *SilhouetteTell*, which can run on an off-the-shelf Android/iPhone smartphone.

5.1 Experimental Setup

The attacker aims to identify a victim's video in a known set. This problem aligns with existing video identification studies (e.g., [19, 43, 64]). We construct the dataset by capturing 300 subtitle files from the three most popular media providers (YouTube, Amazon Prime Video, and Netflix). For each provider, we randomly select 100 movies. The duration of a selected movie ranges from 33 to 200 minutes, with the number of subtitles varying between 413 and 4,071. This 300-video dataset is solely for evaluation. The victim watches a video in the dataset, whether online or offline, on a typical personal device using the corresponding streaming service. We consider an attacker who is at a distance (4 meters or above) from the victim and uses their smartphone camera to record a video of the victim's screen. The recorded video is consistently set to 1080p (1920 × 1080 pixels) at 60 FPS, which is a common camera setting for today's phones. We also investigate the impact of the recording resolution in Appendix C.

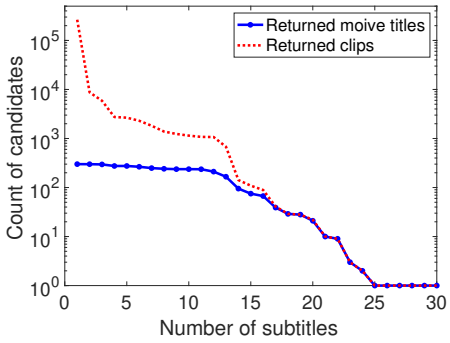

Figure 9: The evolution of the count of candidates.

We calculate *top-k accuracy* in terms of video title or clip, which is defined as the probability that the top k guesses from the obtained N candidates contain the target. If $k > N$, we have $\alpha = 1$; otherwise, $\alpha = \frac{{}^1C_1 \cdot {}^{(N-1)}C_{(k-1)}}{{}^NC_k} = \frac{k}{N}$, where ${}^NC_k$ is the number of combinations by choosing k from N numbers.

Meanwhile, we also utilize traditional text/image recognition algorithms (Tesseract OCR and CRNN for text recognition, and ClipCap for image recognition) to test whether they can identify meaningful information from the recorded blurry videos.

5.2 Case Study

In this example, the victim watches a movie ("The Tomorrow War") on Amazon Prime Video with a 16-inch MacBook Pro. The attacker uses a Samsung Galaxy Z Fold4 smartphone to record a 2-minute video targeting the victim's screen while the video plays from its $2,208^{th}$ to $2,327^{th}$ second. The attacker thus obtains a sequence of 30 subtitle silhouettes. The returned results from OCR are always nonsensical strings; the results of CRNN are empty; by enforcing ClipCap, the results are irrelevant to the scene of the frame.

Figure 9 presents the counts of returned video titles and clips using *SilhouetteTell* during the processing of the silhouettes. For a video with T_0 seconds, we consider that it contains N_c T-second video clips, with a second as the basic unit, i.e., $N_c = T_0 - T + 1$. We see for the initial 7 silhouettes, the number of returned video titles has almost no change, remaining equal to or slightly smaller than the total size (i.e., 300) of the suspect video library. This is because the spatiotemporal structure of a short sequence with fewer silhouettes is common, leading to the failure of using it to distinguish varying videos. On the clip level, as different clips with varying subtitles have different periods, the number of returned video clips decreases accordingly. Also, with the number of silhouettes increasing, the numbers of matching video titles and clips both gradually converge to 1, i.e., the correct match, indicating successful recognition.

5.3 Robustness to Influential Factors

The recording duration/distance/angle/device may vary for the attacker; the victim may use different watching devices and playback speeds. We evaluate their impact. For recorded videos, we also use

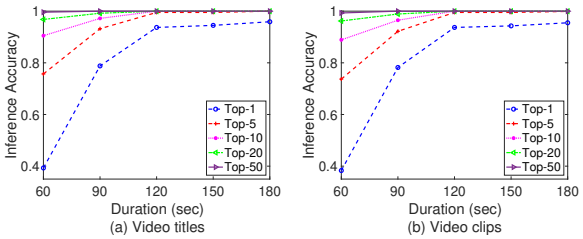

Figure 10: Average top-k accuracy vs. recording duration.

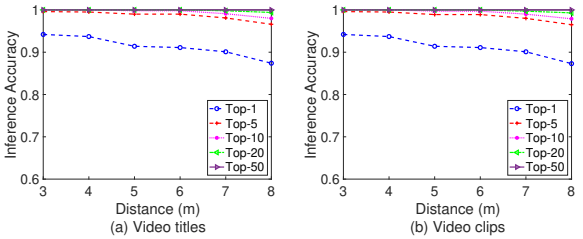

Figure 11: Average top-k accuracy vs. recording distance.

traditional recognition methods (OCR, CRNN, and ClipCap) to extract meaningful information, and none achieves video inference. The impact of the playback speed is presented in Appendix D.

5.3.1 Impact of Recording Duration. We vary the duration of the recorded video clip from 1 to 3 minutes, in increments of 30 seconds. For each clip duration, we randomly record 100 clips. Figures 10a and 10b present the inference performance in terms of the video title and clip. We see from 60 to 120 seconds, the inference accuracy proportionally increases with time, and it maintains high when the duration is 120 seconds or above. When the recording period reaches 2 minutes, the top-10 accuracy stays 100%, and the top-1 accuracy is above 93.7%. Some selected video clips have fewer subtitles due to scenes containing no subtitles (such as gunfight scenes). The spatiotemporal feature of such a video clip may not be rich enough to make the clip distinct, generating multiple candidates. To achieve a desirable inference performance, we employ 2 minutes as the recording period for the following discussion.

5.3.2 Impact of Recording Distance. A traditional shoulder surfer must stay close (e.g., within 2 m) to the victim to directly observe their screen, but this proximity is likely to raise suspicion [73]. To avoid suspicion, we vary the recording distance from 3 to 8 m, in increments of 1. For each recording distance, we randomly record 100 clips. Figure 11 shows the inference performance for different recording distances. We see the inference accuracy remains high for varying distances. Also, with the distance increasing, the average top-1, top-5, and top-10 accuracy values for both video titles and clips slightly decrease. When the distance is 7 m, the average top-1 accuracy values for video titles and clips both exceed 90%. Besides, the top-20 and top-50 accuracy values are always 100%.

Long Recording Distance Tests: Unlike digital zoom, which merely enlarges pixels without enhancing resolution, some smartphones feature optical zoom lenses that adjust the optics to bring the target closer while preserving image clarity. We employ a Samsung S23 Ultra smartphone with 10 times optical zoom [63]. Our long-distance tests were conducted on a 40 m long corridor, as

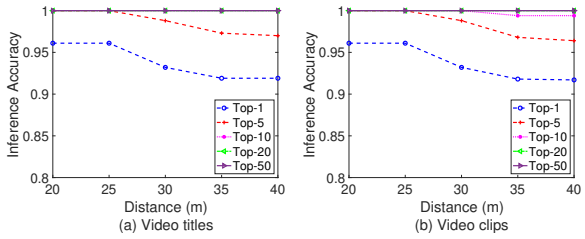

Figure 12: Inference accuracy in long recording distance tests.

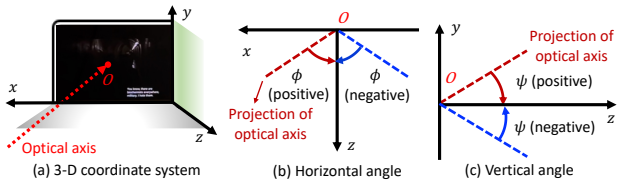

Figure 13: Horizontal and vertical angles.

shown in Appendix E. We vary the recording distance from 20 to 40 m, in increments of 5, and perform 30 trials for each distance. Figure 12 shows the obtained average top-k accuracy. We see that the top-1 accuracy values for video titles and clips slightly decrease with the recording distance but still exceed 92% even at a distance of 40 m. For both video titles and clips, the top-10, top-20, and top-50 accuracies stay at 100% across all distances; the top-5 accuracy is near 100% when the distance ranges from 20 to 30 m and slightly drops to about 97% when the distance equals 35 or 40 m.

5.3.3 Impact of Recording Angle. As shown in Figure 13, the optical axis of the camera intersects the screen at point O; the angle between the projection of the optical axis in the xz-plane and the vertical line (i.e., z axis), which passes through the point O and also is perpendicular to the screen, is referred as horizontal angle (ϕ); the angle between the projection of the optical axis in the yz-plane and the vertical line is referred as vertical angle (ψ). We vary ϕ and ψ both from -60° to 60°, in increments of 30°. We maintain one of two angles fixed and perform 100 trials of *SilhouetteTell* for each value of the other angle, with each trial randomly recording a video clip. Figures 14 and 15 present the corresponding average top-k accuracy. "Top-k, t" and "Top-k, c" refer to top-k accuracy for video titles and clips, respectively. We see the top-k accuracy values for video titles and clips are consistently high. This appears as there is usually one video clip in a video emerging as the candidate. Also, as the recording angle (ϕ or ψ) increases, the top-1 and top-5 accuracies for video titles and clips all slightly decrease. This is because a larger recording angle may result in more distortion of subtitle silhouettes, making the spatiotemporal features of the extracted silhouette sequence less distinguishable and thus producing more candidates. When $|\phi| \leq 30°$ or $|\psi| \leq 30°$, *SilhouetteTell* achieves a top-1 accuracy exceeding 91% for either video titles or clips. Besides, the top-10, top-20, and top-50 accuracy values for all cases are 100%.

5.3.4 Impact of Recording Device. We experiment with three popular smartphones, Samsung Galaxy Z Fold4, iPhone 15 Pro Max, and OnePlus 12 (referred to as Samsung, iPhone, and OnePlus, respectively), and use each to randomly record 100 video clips. Other factors (e.g., recording distance, angle, and resolution) maintain the

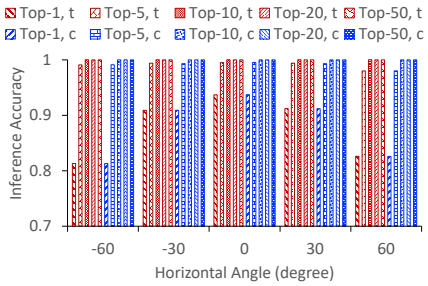

Figure 14: Impact of horizontal angles.

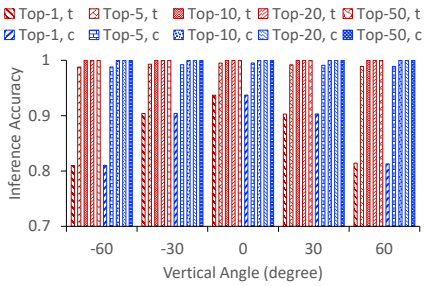

Figure 15: Impact of vertical angles.

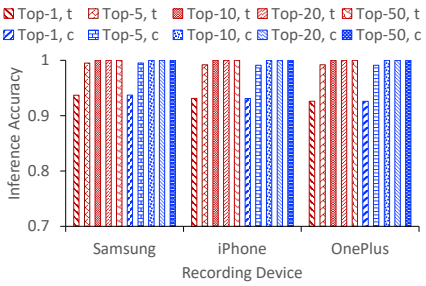

Figure 16: Impact of recording devices.

same. Figure 16 shows the obtained inference accuracy. We see that top-k accuracy values of all recording devices for video titles and clips are consistently high, and there is no obvious difference in attack performance among varying recording devices.

5.3.5 Impact of Watching Device. A victim may watch videos on different devices, with varying screen and subtitle sizes. We test four common watching devices, a mobile phone (Samsung S23 Ultra) with a 6.8-inch screen, a tablet (iPad Pro) with a 12.9-inch screen, a laptop (MacBook Pro) with a 16-inch screen, and a desktop with a 23.8-inch ACER monitor. For each device, we randomly record 100 video clips. During recordings, we control all the rest factors consistently. Figure 17 plots the resultant inference performance. We see that for both video titles and clips, the top-1 and top-5 accuracy values increase with the screen size, while the top-10, top-20, and top-50 accuracy values maintain 1. Particularly, when the victim watches videos with a smartphone, our attack still achieves top-1 accuracy of above 81% for inferring video titles or clips.

5.4 Overall Attack Impact

We perform 5 different attack trials for watching each video in the built dataset, and thus have $5\times3\times100=1,500$ video clips in total. For comparison, we also decode these blurry video clips with OCR, CRNN, and ClipCap, and fail to extract any useful information.

We sort the movies in ascending order of the candidate video title number, and for movies whose candidate video titles are the same, we further sort them in ascending order of the candidate video clip number. We then index the sorted movies from 1 to 300 in increments of 1. Figure 18 presents the corresponding average numbers of video title and clip candidates when the victim watches each movie. We see that the average video title and clip candidate counts are consistently low (under 4.6). Particularly, there are 143 movies whose video title and clip candidate counts are both 1, indicating that our attack can uniquely identify the video title and clip. Figure 19 shows the average top-k accuracy across different streaming services. We see that all top-k accuracy ($k \in \{1, 5, 10, 20, 50\}$) values for both video titles and clips are consistently high (above 91%). Also, there is no obvious performance difference for varying streaming services. These results convincingly indicate that our attack is robust against different movies and service providers. In addition, we also examines the effects of video libraries of different sizes, as presented in Appendix F.

Side-Channel Entropy Analysis: We use a subtitle silhouette sequence within a 2-minute sliding time window as the analysis unit to evaluate the entropy of the proposed subtitle silhouette side channel. Let T_v denote the total number of distinct subtitle sequences. The corresponding T_v subtitle silhouette sequences result in T_v spatiotemporal vectors. By clustering the same vectors into the same class, we obtain $\mathcal{U}$ unique classes. Consequently, the subtitle silhouette sequence X's entropy $H(X)$ can be computed as $-\sum_{i=1}^{\mathcal{U}} P(x_i)log_2P(x_i)$, where $P(x_i)$ represents the probability that $X = x_i$ holds. From our dataset of 300 videos, we extract 731,228 unique classes of spatiotemporal vectors. The largest class contains 14,370 elements, while the smallest class contains only a single element. Accordingly, the maximum entropy is $\log_2(14,370) \approx 13.81$ bits, and the minimum entropy is 0 bits. Also, the average entropy equals 2.58×10^{-5} bits, indicating low uncertainty and a highly predictable structure in the subtitle silhouette side channel.

Table 1: False positive rates by duration in open-world setting ("Avg." denotes average; "#" represents "number").

Duration (sec)	# of FP Clips	FP Rate (%)	Avg. # of Matched Clips	Avg. # of Matched Titles
60	312	10.4	7.33	3.85
90	216	7.2	5.27	2.30
120	109	3.6	1.39	1.38
150	40	1.3	1.30	1.15
180	29	1.0	1.24	1.10

Open-World Scenario: We also investigate the performance of *SilhouetteTell* in an open-world setting, where the target video is not present in the fingerprint database. Specifically, we randomly divide the 300 videos into 10 groups, each containing 30 videos. We then iteratively use each group as the fingerprinting target set, while the remaining nine groups serve as the open-world database. For each target video, we randomly sample 10 clips at a given recording duration, resulting in a total of $10 \times 30 \times 10 = 3,000$ clips per duration. We consider five different recording durations, including 60, 90, 120, 150, and 180 seconds.

For each target clip, we find that the inference result is either empty or consists of a list of clips from the open-world database. A clip is considered a false positive if it matches any clip in the open-world database. Table 1 presents the false positive rates and the average matched clips and titles for false positives in the open-world setting. We observe that with the recording duration increasing, the FP rate significantly decreases. For a recording duration of 60 seconds, 312 out of 3,000 clips are incorrectly matched, resulting in a false positive (FP) rate of 10.4%. As the recording duration increases to 120 and 180 seconds, the FP rate decreases to 3.6% and 1.0%, respectively. In addition, the average number of matched clips and titles per false positive case also decreases with longer recording

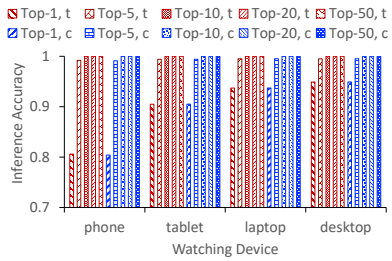

Figure 17: Impact of watching devices.

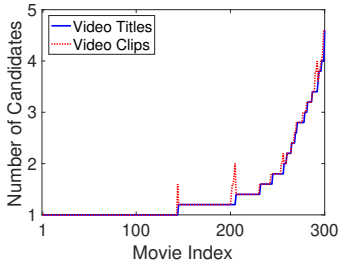

Figure 18: Candidate count distribution.

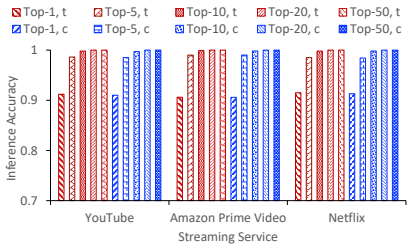

Figure 19: Performance vs. services.

Figure 20: Real-world scenarios.

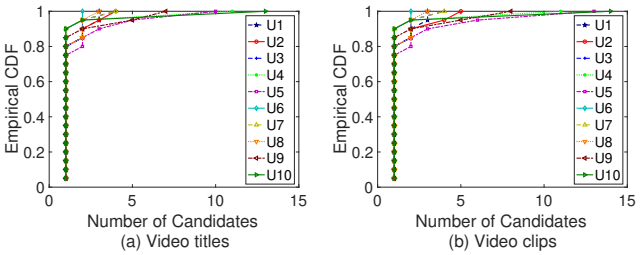

Figure 21: CDFs of the numbers of obtained candidates.

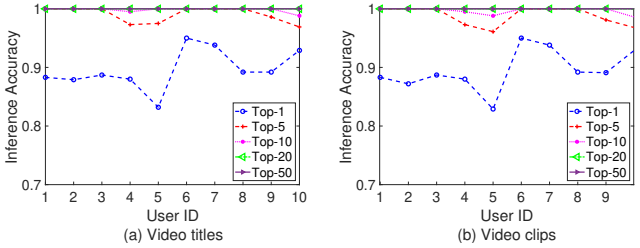

Figure 22: Average top-k accuracy in typical scenario.

durations. These results indicate that increasing the recording duration reduces the ambiguity in spatiotemporal patterns of subtitle silhouettes, leading to lower FP rates.

In the closed world, *SilhouetteTell* achieves 90.9% top-1 accuracy for clip recognition, indicating that for most library videos, *SilhouetteTell* can correctly recognize it, even when provided with only a single candidate. Meanwhile, in the open world, with a recording duration of 120 s, *SilhouetteTell* outputs a non-empty candidate list for only 109 out of 3,000 clips that are not in the library, and correctly determines that the remaining are absent from the library.

5.5 Controlled Experiments

We recruited 10 volunteers (U1-U10; aged 23-32 years old; 5 females and 5 males).[1] We perform experiments in a cafeteria-like environment, where people may walk around or move chairs. Each participant was instructed to sit at a table and watch a movie from the suspect library on a 16-inch MacBook Pro laptop. Two typical cases are considered, as shown in Figure 20: (a) a general scenario where there is no need to have the optical zoom and adjust it to bring the target closer and we run *SilhouetteTell* on a Samsung Galaxy Z Fold4 smartphone at a distance above 4 m behind the victim; (b) a long-distance scenario, where we run *SilhouetteTell* on a Samsung S23 Ultra smartphone and turn on 10 times optical zoom towards the target at a distance of above 20 m away from the victim, making the attack more difficult to be noticed by the victim.

Participants can freely adjust their sitting positions and choose to pause, fast-forward, or rewind videos as usual. For each participant, we performed 20 independent attempts with random video clips selected by the participant. Similarly, for comparison, we utilize OCR, CRNN, and ClipCap to process each recording and generate no meaningful information related to the target video. We present the results of *SilhouetteTell* to the participant, who determines whether the watched clip is in the inferred list. For all trials, the target clip is always in the inferred result. Figure 21 plots the CDFs of the numbers of obtained candidates per user for video titles and clips. We see that the maximum numbers of candidates among all users for video titles and clips are 13 and 14, respectively. Also, regardless of users, the probabilities of uniquely inferring the target video title and clip are both at least 75%. Figure 22 presents the inference performance. We observe that our attack consistently achieves high average top-1 and top-5 accuracies (above 83% and 97% respectively) for all users in terms of video titles and clips; the average top-10, top-20, and top-50 accuracies are all near or equal to 100%.

We have similar observations in the long-distance scenario: for all users, the numbers of candidate video titles and clips range from 1 to 13, and from 1 to 15, respectively; the minimum average top-1 accuracies for video titles and clips are 88.9% and 88.6%, respectively.

Comparison with Baselines: For comparison, we evaluate the performance of human attackers using the two baseline methods: (i) Bare-eye recognition: 10 participants with normal or corrected-to-normal vision (either unaided or with glasses) attempt to recognize the videos from a distance of 40 meters, relying solely on their vision. (ii) 10× zoom recognition: The same participants repeat the above experiment using the 10× zoom lens of a Samsung Galaxy S23 Ultra phone. In each trial, we randomly select a video from the 300-movie dataset and play a 2-minute clip. Each participant performs 30 attempts per above baseline. The results show that no participant correctly identifies any movie under bare-eye or 10× zoom conditions, yielding a 0% success rate for video identification.

[1]This study has been reviewed and approved by our institution's IRB.

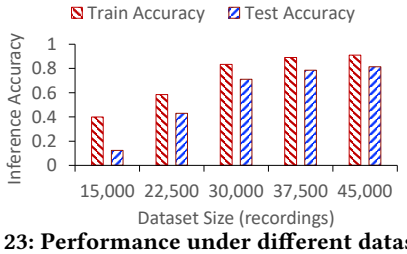

Figure 23: Performance under different dataset sizes.

In addition, we also test a visual input attack, i.e., CNN-LSTM based recognition. The CNN-LSTM hybrid architecture [27, 70] is a well-established model for spatiotemporal visual understanding tasks. We use the ResNet-50 network as the backbone of the spatial feature extractor. The extracted spatial features are fed into an LSTM layer to capture temporal dependencies. We fine-tune the LSTM and final classification layers using our dataset of 300 videos.

To generate the dataset for building the model, we adopt a grid-based multi-angle recording strategy. We vary the horizontal viewing angle across five settings (0°, ±30°, ±60°), and the vertical angle similarly across five settings (0°, ±30°, ±60°), resulting in 25 distinct camera viewpoints. Also, we vary the recording distance from the target screen between 2 and 5 meters, in 1-meter increments. At each spatial configuration (i.e., a unique combination of angle and distance), we record each video once, yielding 100 recordings per video. We then get $300 \times 100 = 30,000$ recordings in total. We experiment with varying the number of recordings per video. To increase the dataset size, we randomly select 25 or 50 spatial configurations and add one new recording at each, resulting in 25 or 50 additional recordings per video. To reduce the dataset size, we similarly select 25 or 50 configurations at random and remove the existing recordings at those locations. This allows us to evaluate performance under different dataset sizes of 50, 75, 125, and 150 recordings per video. For each dataset size, we apply the standard 80:20 train-test split to the data of each video, ensuring that the CNN-LSTM model is trained with samples from all 300 videos (i.e., classes).

Figure 23 presents the obtained train and test accuracies of the CNN-LSTM model. We see that both train and test accuracies improve as the dataset size increases. With 100 recordings per class, the model achieves 83.3% and 71.0% accuracy for training and testing, respectively; increasing to 150 recordings per class raises these to 91.0% and 81.0%. To evaluate the model's performance for longer recording distances, we record 20 different videos at distances of 6, 7, and 8 meters, respectively. The results show that inference accuracy declines as recording distance increases. The top-1 accuracies at 6, 7, and 8 meters are 0.60, 0.35, and 0.25, respectively. This decline is likely due to reduced spatial resolution and increased noise at greater distances, which degrade the model's ability to extract meaningful features. These results demonstrate that the CNN-LSTM model, when trained with sufficient data, achieves high accuracy on configurations seen during training. However, its performance drops significantly under unseen recording conditions. Moreover, the data collection and training process is costly, which further limits the model's scalability.

For clip-level recognition, *SilhouetteTell* can achieve a top-1 accuracy of 0.91, while when 150 recordings per class are used for training, the top-1 accuracy for testing of CNN-LSTM attains is just 0.81. Meanwhile, with the recording distance increasing, the CNN-LSTM degrades sharply (top-1 = 0.60/0.35/0.25 at 6/7/8 meters). In contrast, *SilhouetteTell* shows only a slight decline, achieving top-1 accuracies of 0.91, 0.90, and 0.87, respectively.

6 Discussions

6.1 Limitation

Necessity of Turning on Subtitles: *SilhouetteTell* does not work for cases when the victim watches a video without subtitles enabled. A recent survey shows that 50% of Americans, and particularly 70% of young viewers aged 18 to 25, watch content with subtitles most of the time [51]. Also, subtitles are almost a prerequisite for following videos with foreign language content or in a noisy environment [67]. Our attack thus poses a serious practical privacy threat.

Unknown Subtitle Files: *SilhouetteTell* builds a suspect video library by downloading subtitles files and can only infer videos in the built library. Our experimental scale is comparable to that in most recent studies (e.g., [19, 35, 75]) but still limited. If the victim watches a subtitled video that does not appear in the library, the inference may fail. However, conducting larger-scale experiments by pre-downloading or pre-extracting subtitle files of more videos may further evidence the efficacy of *SilhouetteTell*.

6.2 Defense Strategies

6.2.1 Disabling Subtitles. Intuitively, to thwart *SilhouetteTell*, a user may close subtitles, which, however, are necessary in many cases, e.g., for translating foreign-language content. Also, disabling subtitles may reduce viewing experience.

We randomly select 25 2-minute video clips and play each with subtitles off. As no subtitle silhouettes appear in the recorded video, *SilhouetteTell* fails to recognize the video title/clip across all trials.

6.2.2 Rendering Obfuscated Subtitle Silhouettes. We may confuse the attacker by rendering the same or randomized subtitle silhouettes. Currently, the shapes (e.g., line counts) of a video's subtitles display on the screen strictly following the video's subtitle file. We can pad each subtitle of a video frame with special characters, which aims to prevent disclosing the original true subtitle silhouettes. As a result, there is no confirmed correlation between a video clip's subtitles and the observed corresponding subtitle silhouettes. *SilhouetteTell* would thus fail. This method, however, may increase the complexity of rendering subtitles and also degrade the viewing experience. To further mislead attackers, the video can generate and insert fake subtitle silhouettes into frames that originally contain no subtitles. These artificial subtitles are designed to appear visually similar to real ones but do not contain meaningful text, ensuring they do not impact user comprehension.

To evaluate this defense, we modify the original subtitle (.srt) files by appending padding characters ("#") at the end of each subtitle line. This ensures that every frame containing subtitles displays two lines of subtitles, each consisting of 40 characters. This modification preserves the readability of the subtitles while significantly altering their visual appearance. As a result, the subtitles in the recorded videos produce identical silhouette patterns, and *SilhouetteTell* consistently misclassifies these modified silhouettes, incorrectly identifying all randomly selected 25 video clips. Figure 24

Figure 24: Illustration of padded subtitles.

shows the impact of subtitle padding. Without padding, the original two frames contain one and two lines of subtitles, respectively, resulting in distinguishable subtitle silhouettes in blurry recordings. In contrary, with padding, both blurry frames exhibit identical subtitle silhouettes, effectively confusing *SilhouetteTell*.

6.2.3 Applying privacy screen protectors. Another defense is to use privacy screen protectors (e.g., [16]) with special filters that allow light to pass through from certain angles (e.g., narrow front angles), which could be further blocked by the victim's body. It would be then difficult for the attacker to find a workable recording angle to video-record clear subtitle silhouettes. However, putting privacy films on screens would reduce the viewing quality for the victim, as the screen's brightness and color may be deteriorated. It also requires the victim to cooperate to help block the angles from which the light is not filtered by the films and can pass through.

We evaluate a common privacy screen protector [14] on a 16-inch MacBook Pro. This protector is designed to reduce screen brightness and limit visibility beyond a horizontal viewing angle of ±30°. When the attacker records from outside this range, the captured video appears nearly black, rendering subtitle silhouette extraction infeasible. In this setting, across 25 randomly selected video clips, *SilhouetteTell* again consistently fails to identify the correct videos. Figure 25 shows the privacy screen protector's impact. When the protector is in place, the screen becomes effectively invisible from side angles, revealing no subtitle silhouettes.

6.3 Potential Extensions

While *SilhouetteTell* is specifically designed to leverage the spatiotemporal feature of subtitle silhouettes, the underlying methodology can be generalized to other forms of on-screen text. For instance, news tickers commonly appear as horizontally or vertically scrolling text, typically located at the bottom or top of the screen. By tracking the appearance frequency, shape, and position of these text regions over time, we can construct spatiotemporal fingerprints similar to those used for subtitles. Such fingerprints can then be used for video identification.

7 Related Work

CV-based Text/image Recognition: By recognizing images or text (e.g., subtitles) in a video, we may infer the video by content matching. Computer Vision (CV) based recognition methods exploit the features from video data and achieve recognizing personnel or objects within images (e.g., [47, 61]). OCR and Scene Text Recognition (STR) [25, 68] are two widely used methods for text recognition. However, all of these methods require clear, detailed, and high-quality recordings to extract text or images accurately, rendering them incapable of handling scenarios when recordings

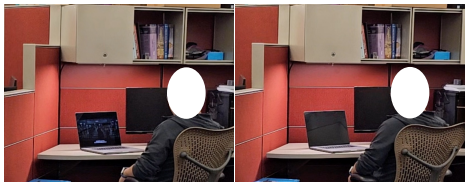

Figure 25: Illustration of the privacy screen protector defense.

are blurred. On the contrary, our work is the first practical video inference technique that can handle scenarios with blurring recordings when traditional CV-based methods fail.

Besides, analyzing reflections from nearby objects has been explored to recover sensitive content displayed on a screen [18, 57, 71, 72]. For example, [18] is one pioneering study to infer on-screen content from blurred images, while it requires using an astronomical camera that costs over 6,000 USD. In contrast, *SilhouetteTell* works with a low-cost RGB camera. Meanwhile, [57] and [72] target keystroke inference leveraging key pop-out events and fingertip motion, respectively. These techniques study a different privacy threat from ours. Also, [71] infers TV content by leveraging flickering patterns caused by brightness changes due to scene transitions. However, this method is limited to nighttime conditions with no ambient lighting and has only been evaluated on large displays (24/30/50 inches). Conversely, *SilhouetteTell* has no such restrictions and remains effective on small screens (e.g., 6.8 inches).

Traffic Analysis based Video Inference: There have been extensive studies using traffic analysis to infer videos. Different videos may thus result in distinct traffic patterns. In general, traffic-assist video inference attacks can be divided into the following categories according to the source of traffic. (1) *Wired-based:* the attacker may hack the router that the victim connects and then connect to the router through an Ethernet cable for traffic sniffing [35, 75]. (2) *WiFi-based:* the attacker may connect to the same WiFi as the victim to capture wireless traffic [28, 46]. (3) *Cellular-based:* recent studies (e.g., [19, 43]) also show success in achieving video identification by collecting and analyzing LTE traffic. In contrast, our attack does not rely on traffic for video inference and works regardless of whether the victim watches videos online or offline.

8 Conclusion

We propose *SilhouetteTell*, a novel and practical video inference technique, with the following advantages over previous methods. (1) Non-invasive: there is no need to pre-infect the victim's device with malware. (2) Traffic-independent: it works for both online and offline video streaming. (3) No user-specific training is needed: no labeled data is needed from the victim and it can handle cases when traditional text/image recognition algorithms fail. (4) It requires no close proximity: it can be launched several meters, even tens of meters, away from the target. *SilhouetteTell* is the first to identify and build the spatiotemporal correlation between the captured blurry video frames and embedded subtitles. Our extensive evaluation on top of off-the-shelf smartphones verifies that *SilhouetteTell* can achieve high accuracy in identifying video titles and clips that the victim is watching from a distance of up to 40 meters.

Acknowledgments

The authors would like to thank all anonymous reviewers for their insightful comments. This work was supported in part by the National Science Foundation under Grants No. 2155181 and No. 2424439.

References

[1] 2018. Developments in the Law: MORE DATA, MORE PROBLEMS. *Harvard Law Review* 131, 6 (2018), 1714–1811.
[2] 2024. CLIP prefix captioning. https://github.com/rmokady/CLIP_prefix_caption.
[3] 2024. Convolutional Recurrent Neural Network. https://github.com/bgshih/crnn.
[4] 2024. Download subtitles from Youtube, Viki, Viu, Wetv, Kocowa and more. https://downsub.com/.
[5] 2024. Interactive Foreground Extraction using GrabCut Algorithm. https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html.
[6] 2024. OpenCV. https://docs.opencv.org/3.4/.
[7] 2024. OpenCV: Color Space Conversions. https://docs.opencv.org/3.4/d8/d01/group__imgproc__color__conversions.html.
[8] 2024. OpenCV: cv::MSER Class Reference. https://docs.opencv.org/3.4/d3/d28/classcv_1_1MSER.html.
[9] 2024. OpenCV: Geometric Image Transformations. https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html.
[10] 2024. Subtitle Converter. https://gotranscript.com/subtitle-converter.
[11] 2024. Subtitles - download movie and TV Series subtitles. https://www.opensubtitles.org/en/search/subs.
[12] 2024. Tesseract OCR. https://github.com/tesseract-ocr/tesseract.
[13] 2024. youtube-transcript-api. https://pypi.org/project/youtube-transcript-api/.
[14] 2025. Peslv Magnetic Privacy Screen for MacBook Pro 16 Inch (2021–2024, M1–M4). Amazon. https://www.amazon.com/dp/B09K72GKJ4
[15] 112th Congress. 2012. Video Privacy Protection Act Amendments Act of 2012. https://www.govinfo.gov/content/pkg/CRPT-112srpt258/html/CRPT-112srpt258.htm.
[16] 3M. 2024. Subtitles - download movie and TV Series subtitles. https://www.3m.com/3M/en_US/privacy-screen-protectors-us/.
[17] Waleed Abdulla. 2017. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN.
[18] Michael Backes, Tongbo Chen, Markus Duermuth, Hendrik P.A. Lensch, and Martin Welk. 2009. Tempest in a Teapot: Compromising Reflections Revisited. In *2009 30th IEEE Symposium on Security and Privacy*. 315–327. https://doi.org/10.1109/SP.2009.20
[19] Sangwook Bae, Mincheol Son, Dongkwan Kim, CheolJun Park, Jiho Lee, Sooel Son, and Yongdae Kim. 2022. Watching the Watchers: Practical Video Identification Attack in LTE Networks. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 1307–1324.
[20] Jamie Ballard. 2023. Most American adults under 30 prefer watching TV with subtitles — even when they know the language. https://today.yougov.com/entertainment/articles/45987-american-adults-under-30-watching-tv-subtitles.
[21] Kayce Basques, Jecelyn Yeen, and Sofia Emelianova. 2018. Open Chrome DevTools. https://developer.chrome.com/docs/devtools/open.
[22] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 177–186.
[23] Andy Brown, Rhia Jones, Mike Crabb, James Sandford, Matthew Brooks, Mike Armstrong, and Caroline Jay. 2015. Dynamic Subtitles: The User Experience. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video* (Brussels, Belgium) *(TVX '15)*. Association for Computing Machinery, New York, NY, USA, 103–112.
[24] Jackie Callaway. 2018. The new email scam that threatens to send your pornography browsing history to friends and family. https://www.abcactionnews.com/news/national/the-new-email-scam-that-threatens-to-send-your-pornography-browsing-history-to-friends-and-family.
[25] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. 2021. Text Recognition in the Wild: A Survey. *ACM Comput. Surv.* 54, 2, Article 42 (mar 2021), 35 pages.
[26] Jorge Díaz Cintas and Aline Remael. 2014. *Audiovisual translation: subtitling*. Routledge.
[27] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2625–2634.
[28] Ran Dubin, Amit Dvir, Ofir Pele, and Ofer Hadar. 2017. I Know What You Saw Last Minute—Encrypted HTTP Adaptive Video Streaming Title Classification. *IEEE Transactions on Information Forensics and Security* 12, 12 (2017), 3039–3049.
[29] Sagi Eden, Amit Livne, Oren Sar Shalom, Bracha Shapira, and Dietmar Jannach. 2022. Investigating the Value of Subtitles for Improved Movie Recommendations. In *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization* (Barcelona, Spain) *(UMAP '22)*. Association for Computing Machinery, New York, NY, USA, 99–109.
[30] Kate Fazzini. 2019. Email sextortion scams are on the rise and they're scary — here's what to do if you get one. https://www.cnbc.com/2019/06/17/email-sextortion-scams-on-the-rise-says-fbi.html.
[31] Dan Frankowski, Dan Cosley, Shilad Sen, Loren Terveen, and John Riedl. 2006. You are what you say: privacy risks of public mentions. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 565–572.
[32] Taraneh Ghandi, Hamidreza Pourreza, and Hamidreza Mahyar. 2023. Deep Learning Approaches on Image Captioning: A Review. *ACM Comput. Surv.* 56, 3, Article 62 (oct 2023), 39 pages.
[33] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 580–587.
[34] Henrik Gottlieb. 2012. Subtitles: readable dialogue? *Eye tracking in audiovisual translation* (2012), 37–82.
[35] Jiaxi Gu, Jiliang Wang, Zhiwen Yu, and Kele Shen. 2018. Walls Have Ears: Traffic-based Side-channel Attack in Video Streaming. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1538–1546.
[36] Jiaxi Gu, Jiliang Wang, Zhiwen Yu, and Kele Shen. 2019. Traffic-Based Side-Channel Attack in Video Streaming. *IEEE/ACM Transactions on Networking* 27, 3 (2019), 972–985.
[37] Thomas Hanke, Marc Schulder, Reiner Konrad, and Elena Jahn. 2020. Extending the Public DGS Corpus in size and depth. In *sign-lang@ LREC 2020*. European Language Resources Association (ELRA), 75–82.
[38] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. 2017. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2961–2969.
[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[40] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. 2017. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4507–4515.
[41] MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2019. A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Comput. Surv.* 51, 6, Article 118 (feb 2019), 36 pages.
[42] Yongtao Hu, Jan Kautz, Yizhou Yu, and Wenping Wang. 2015. Speaker-Following Video Subtitles. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 2, Article 32 (jan 2015), 17 pages.
[43] Nitya Lakshmanan, Abdelhak Bentaleb, Byoungjun Choi, Roger Zimmermann, Jun Han, and Min Suk Kang. 2022. On Privacy Risks of Watching YouTube over Cellular Networks with Carrier Aggregation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 1, Article 19 (mar 2022), 22 pages.
[44] Sunil Lee and Chang D. Yoo. 2008. Robust video fingerprinting for content-based video identification. *IEEE Transactions on Circuits and Systems for Video Technology* 18, 7 (2008), 983–988.
[45] Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. 2020. Tvr: A large-scale dataset for video-subtitle moment retrieval. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*. Springer, 447–463.
[46] Ying Li, Yi Huang, Richard Xu, Suranga Seneviratne, Kanchana Thilakarathna, Adriel Cheng, Darren Webb, and Guillaume Jourjon. 2018. Deep Content: Unveiling Video Streaming Content from Encrypted WiFi Traffic. In *IEEE International Symposium on Network Computing and Applications (NCA)*. 1–8.
[47] Mehdi Masmoudi, Hamdi Friji, Hakim Ghazzai, and Yehia Massoud. 2021. A Reinforcement Learning Framework for Video Frame-Based Autonomous Car-Following. *IEEE Open Journal of Intelligent Transportation Systems* 2 (2021), 111–127.
[48] J Matas, O Chum, M Urban, and T Pajdla. 2004. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing* 22, 10 (2004), 761–767. British Machine Vision Computing 2002.
[49] TJ McCue. 2019. Verizon Media Says 69 Percent Of Consumers Watching Video With Sound Off. https://www.forbes.com/sites/tjmccue/2019/07/31/verizon-media-says-69-percent-of-consumers-watching-video-with-sound-off.
[50] Ron Mokady, Amir Hertz, and Amit H Bermano. 2021. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734* (2021).
[51] Nadiia Mykhalevych. 2024. Survey: Why America is obsessed with subtitles. https://preply.com/en/blog/americas-subtitles-use/.
[52] G. Nagy. 2000. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 1 (2000), 38–62.
[53] Netflix. 2024. How to download titles to watch offline. https://help.netflix.com/en/node/54816.

[54] Alexander Neubeck and Luc Van Gool. 2006. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, Vol. 3. IEEE, 850–855.
[55] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763. https://proceedings.mlr.press/v139/radford21a.html
[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
[57] Rahul Raguram, Andrew M White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. 2011. iSpy: automatic reconstruction of typed input from compromising reflections. In *Proceedings of the 18th ACM conference on Computer and communications security*. 527–536.
[58] Nikhil Kumar Rajput and Bhavya Ahuja Grover. 2022. A Multi-Label Movie Genre Classification Scheme Based on the Movie's Subtitles. *Multimedia Tools Appl.* 81, 22 (sep 2022), 32469–32490.
[59] Andrew Reed and Michael Kranch. 2017. Identifying HTTPS-Protected Netflix Videos in Real-Time. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy* (Scottsdale, Arizona, USA) *(CODASPY '17)*. Association for Computing Machinery, New York, NY, USA, 361–368.
[60] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (2017), 1137–1149.
[61] Xiang Ren, Min Sun, Xianfeng Zhang, Lei Liu, Hang Zhou, and Xiaoping Ren. 2022. An improved mask-RCNN algorithm for UAV TIR video stream target detection. *International Journal of Applied Earth Observation and Geoinformation* 106 (2022), 102660.
[62] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.* 23, 3 (aug 2004), 309–314.
[63] Samsung. 2024. Galaxy S23 Ultra. https://www.samsung.com/us/smartphones/galaxy-s23-ultra/.
[64] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. 2017. Beauty and the Burst: Remote Identification of Encrypted Video Streams. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 1357–1374.
[65] Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 11 (2017), 2298–2304.
[66] R. Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 629–633.
[67] VITAC. 2024. The Growing Demand for Captions and Subtitles in Today's Media Landscape. https://vitac.com/the-growing-demand-for-captions-and-subtitles-in-todays-media-landscape.
[68] Peng Wang, Hui Li, and Chunhua Shen. 2022. Towards End-to-End Text Spotting in Natural Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 7266–7281. https://doi.org/10.1109/TPAMI.2021.3095916
[69] Wyzowl. 2023. Video Marketing Statistics 2023. https://www.wyzowl.com/video-marketing-statistics.
[70] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. PMLR, 2048–2057.
[71] Yi Xu, Jan-Michael Frahm, and Fabian Monrose. 2014. Watching the watchers: Automatically inferring tv content from outdoor light effusions. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 418–428.
[72] Yi Xu, Jared Heinly, Andrew M White, Fabian Monrose, and Jan-Michael Frahm. 2013. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 1063–1074.
[73] Guixin Ye, Zhanyong Tang, Dingyi Fang, Xiaojiang Chen, Kwang In Kim, Ben Taylor, and Zheng Wang. 2017. Cracking android pattern lock in five attempts. In *Proceedings of the 2017 Network and Distributed System Security Symposium 2017 (NDSS 17)*. Internet Society.
[74] Soledad Zárate. 2021. *Captioning and subtitling for d/deaf and hard of hearing audiences.* UCL Press.
[75] Xiyuan Zhang, Gang Xiong, Zhen Li, Chen Yang, Xinjie Lin, Gaopeng Gou, and Binxing Fang. 2023. Traffic Spills the Beans: A Robust Video Identification Attack Against YouTube. *Computers & Security* (2023), 103623.

Table 2: Average top-k accuracy for video titles and clips (referred to as "T" and "C") vs. recording resolution.

Resolution	Top-1		Top-5		Top-10		Top-20		Top-50	
	T	C	T	C	T	C	T	C	T	C
4k	0.953	0.953	0.995	0.995	1.0	1.0	1.0	1.0	1.0	1.0
1080p	0.937	0.937	0.995	0.995	1.0	1.0	1.0	1.0	1.0	1.0
720p	0.903	0.901	0.995	0.994	1.0	1.0	1.0	1.0	1.0	1.0

A Subtitle Data Crawling

We can directly copy the URL of a YouTube video to web applications such as DownSub [4], and then download its subtitles. We can also write a script to automatically get the subtitles for a given YouTube video using a Python API (i.e., youtube-transcript-api [13]). Besides, the following steps can be performed to obtain subtitle files from Amazon Prime Video and Netflix: (1) open DevTools (i.e., a set of web developer tools) from Chrome menus [21]; (2) in the Network tab, filter ".ttml2" (for Amazon Prime Video) or a string "?o=" (for Netflix) when playing a video online; (3) capture subtitle files from the returned results and further convert them into .srt files via a subtitle converter tool (e.g., [10]).

B Silhouette Extraction Comparison

Figure 26 compares coarse and refined silhouette extraction methods in dealing with five different cases, where coarse silhouette extraction fails while refined silhouette extraction manages to generate correct recognition results. In cases 1-3, coarse silhouette extraction makes errors in recognizing silhouette width, height, and both width and height, respectively; in cases 4-5, the input frames have no subtitles, while coarse silhouette extraction incorrectly recognizes interference shapes as subtitle silhouettes.

C Impact of Recording Resolution

Usually, external subtitles (e.g., SRT files) of a video are independent and separate from the video, and their resolution (i.e., quality) will not change with the resolution of the video being played. Therefore, if the video resolution changes, the silhouettes of the subtitles of this video may not change accordingly. On the contrary, the adversary may utilize different resolutions to do recording, capturing silhouettes that may have inconsistent clarity. We consider three typical recording resolutions for the recording device (Samsung Galaxy Z Fold4): 720p (1280×720), 1080p (1920×1080), and 4k (3840×2160). For each resolution, we randomly record 100 video clips.

Table 2 presents the top-k accuracy for inferring video titles and clips under different recording resolutions. We can see our attack can consistently obtain pretty high (above 90%) inference performance. Particularly, the top-10, top-20, and top-50 accuracy values are always 1 regardless of the recording resolution. Also, with the resolution decreasing, the top-1 accuracy slightly decreases, and top-5 accuracy almost maintains a high value (i.e., 99.5%).

D Impact of Playback Speed

We test four different playback speeds: 0.5×, 1.0×, 1.5×, and 2.0×. For each playback speed, we randomly record 100 2-minute video clips while keeping other factors (e.g., recording distance, angle, and resolution) consistent. Figures 27 present the resultant inference

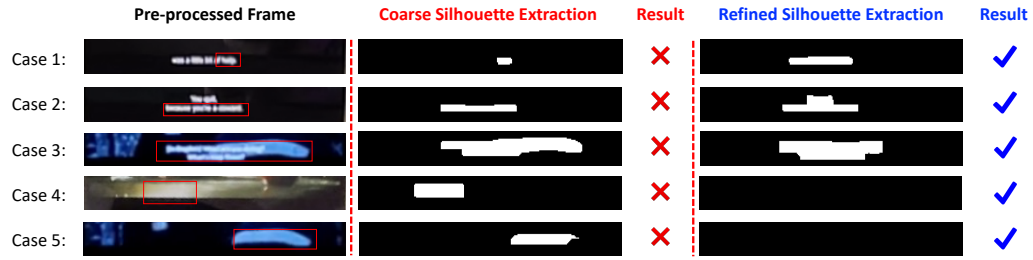

Figure 26: Comparison of extraction results between coarse and refined silhouette extraction.

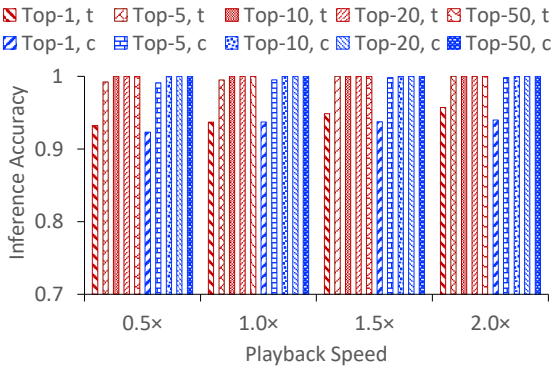

Figure 27: Impact of playback speed.

Figure 28: Recording the target across the 40 m corridor.

performance. We observe that for both video titles and clips, the top-1 and top-5 accuracy values slightly increase as the playback speed increases, while the top-10, top-20, and top-50 accuracy values remain at 100%. This improvement mainly results from an increase in the length of the obtained subtitle vector at higher playback speeds, which reduces the number of candidates. Particularly, the top-1 accuracy for video titles ranges from 93.3% to 95.7% regardless of the playback speed.

E Long-distance Testing Environment

Figure 28 shows our long-distance testing environment.

F Scalability Analysis

Scalability Analysis In existing video inference studies [19, 35, 59, 75], the number of videos used typically ranges from 100 to 300.

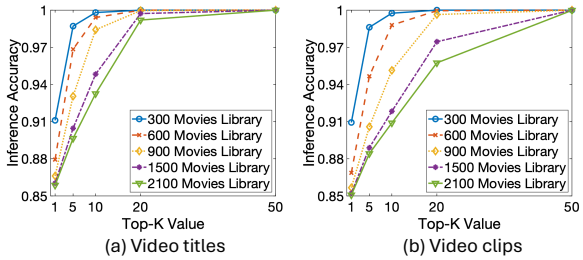

Figure 29: Average top-k accuracy vs. library size.

Due to the widespread availability of subtitle files, *SilhouetteTell* can easily expand its video database for matching. We examine the effects of video libraries of five different sizes: 300, 600, 900, 1,500, and 2,100. Accordingly, we randomly select 100, 200, 300, 500, and 700 videos from each video provider. The video durations range from 24 to 242 minutes. Figure 29 (a) and (b) illustrate the corresponding average video titles and clips inference accuracy. We observe that increasing the library size does not significantly impact inference accuracy. Particularly, the top-1 and top-20 accuracy values remain above 85% and 96%, respectively, across all video library sizes.