# Liveness Verifications for Citizen Journalism Videos

Mahmudur Rahman
FIU
Miami, FL
mrahm004@cs.fiu.edu

Mozhgan Azimpourkivi
FIU
Miami, FL
mozhganaz@gmail.com

Umut Topkara
JW Player
New York City, NY
topkara@gmail.com

Bogdan Carbunar
FIU
Miami, FL
carbunar@cs.fiu.edu

## ABSTRACT

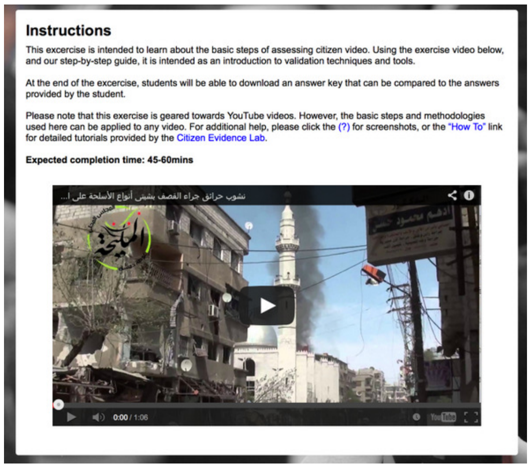
Citizen journalism videos increasingly complement or even replace the professional news coverage through direct reporting by event witnesses. This raises questions of the integrity and credibility of such videos. We introduce Vamos, the first user transparent video "liveness" verification solution based on video motion, that can be integrated into any mobile video capture application without requiring special user training. Vamos' algorithm not only accommodates the full range of camera movements, but also supports videos of arbitrary length. We develop strong attacks both by utilizing fully automated attackers and by employing trained human experts for creating fraudulent videos to thwart mobile video verification systems.

We introduce the concept of video motion categories to annotate the camera and user motion characteristics of arbitrary videos. We share motion annotations of YouTube citizen journalism videos and of free-form video samples that we collected through a user study. We observe that the performance of Vamos differs across video motion categories. We report the expected performance of Vamos on the real citizen journalism video chunks, by projecting on the distribution of categories. Even though Vamos is based on motion, we observe a surprising and seemingly counterintuitive resilience against attacks performed on relatively "static" video chunks, which turn out to contain hard-to-imitate involuntary movements. We show that the accuracy of Vamos on the task of verifying whole length videos exceeds 93% against the new attacks.

## 1. INTRODUCTION

The citizen journalism revolution, enabled by advances in mobile and social technologies, transforms information consumers into collectors and disseminators of news. Major news outlets have started to fill out professional journalistic gaps with videos shot on mobile devices. Exam-

*WiSec* '15 , June 22 - 26, 2015, New York, NY, USA
Copyright 2015 ACM 978-1-4503-3623-9/15/06
DOI: http://dx.doi.org/10.1145/2578726.2578744 ...$15.00.



**Instructions**

This excercise is intended to learn about the basic steps of assessing citizen video. Using the exercise video below, and our step-by-step guide, it is intended as an introduction to validation techniques and tools.

At the end of the excercise, students will be able to download an answer key that can be compared to the answers provided by the student.

Please note that this exercise is geared towards YouTube videos. However, the basic steps and methodologies used here can be applied to any video. For additional help, please click the (?) for screenshots, or the "How To" link for detailed tutorials provided by the Citizen Evidence Lab.

**Expected completion time: 45-60mins**

**Figure 1: Snapshot of Citizen Evidence Lab [3] training session exercise. It consists of steps to verify the source of the video (i.e., account of uploader, upload time) and its content (e.g., clothes, accents, flags, landmarks).**

ples range from videos of conflicts in areas with limited professional journalism representation (e.g., Syria, Ukraine) to spontaneous events (e.g., tsunamis, earthquakes, meteorite landings, authority abuse). Such videos are often distributed through sites such as CNN's iReport [5], NBC's Stringwire [11] and CitizenTube [4].

The increasing popularity of citizen journalism is starting however to raise important questions concerning the credibility of impactful videos (see e.g., [3, 15, 34, 18]). Videos from other sources can be copied, projected and recaptured, cut and stitched before being uploaded as genuine on social media sites.

Citizen Evidence Lab [3] and Witness.Org [14] provide tutorials to train the public to create and to assess citizen reports, including those captured with mobile devices (see Figure 1 for a tutorial snapshot). InformaCam [7] provides mechanisms to ensure that the media was captured by a specific device at a certain location and time. InformaCam is however ineffective against adversaries that capture videos of projected videos: The resulting videos have been shot with the device, but are fraudulent. While manual verifications can identify such attacks, they do not scale well to the high number of videos on social media sites.

To address this problem, we exploit the observation that for plagiarized videos, the motion encoded in the video stream is likely inconsistent with the motion from the inertial sensor streams (e.g., accelerometer) of the device. Movee [33], a video liveness verification solution that uses this principle, has important weaknesses: i) it is not user transparent to the extent that it imposes an explicit verification step on users, ii) it severely limits the movements in the verification step to one of four pan movements, and iii) it is vulnerable to "stitch" attacks in which the attacker creates a fraudulent video by first live recording a genuine video and then pointing the camera to a pre-recorded target video.

In this paper, we introduce Vamos, a Video Accreditation through Motion Signatures system. Vamos provides liveness verifications for videos of arbitrary length. It is resistant to a wide range of attacks including those by fully automated systems and those employing trained human experts. Vamos is completely transparent to the users; it requires no special user interaction, nor change in user behavior.

Instead of enforcing an initial verification step, Vamos uses the entire video and acceleration stream for verification purposes: It divides the video and acceleration data into fixed length chunks. It then classifies each chunk and uses the results, along with a suite of novel features that we introduce, to classify the entire sample. This process enables Vamos to efficiently detect several potent attacks, including stitch attacks.

Vamos does not impose a dominant motion direction, thus, does not constrain the user movements. Instead, Vamos verifies the liveness of the video by extracting features from *all* the directions of movement, from both the video and acceleration streams. Vamos improves on the free-form video motion verification accuracy of Movee by more than 15% in the domain of 6 second *Cluster Attack* videos, and by more than 30% in the domain of whole length *Cluster and Stitch Attack* videos (see Section 3 for a discussion of the adversary model).

Removed video length and movement constraints provide additional flexibility for attackers to create fraudulent videos. In order to study the security of the new unconstrained setting, we i) propose a novel, motion based video classification system, ii) introduce several attacks targeted at sensor based video liveness verification, and iii) show experimental evidence on a wide range of data collected through user studies and from public sources.

We have collected 150 citizen journalism videos from YouTube and performed a user study to collect 160 free-form videos and corresponding acceleration samples. Experiments with 6s chunks extracted from the free-form videos confirm (through $\chi^2$ and Fisher's exact tests) that the classification performance of Vamos depends on the video category. Furthermore, we show that the proposed motion based video classification can be used to predict the accuracy of Vamos on videos for which we currently lack associated sensor streams (e.g., YouTube videos). To summarize, this paper makes the following contributions:

- **Targeted attacks**. Introduce a sensor based attack model and develop novel attacks targeted against video verification mechanisms [§ 3].
- **Video motion classification**. Introduce a novel classification of mobile videos [§ 4].
- **Vamos**. Develop a video liveness verification solution to detect fraudulent video and inertial sensor chunks that encode arbitrary motions. Introduce Vamos, a system that detects fraudulent video and accelerometer streams of arbitrary length, and is resilient to powerful attacks [§ 5].
- **Video data collection**. Collect datasets of free-form and citizen journalism videos [§ 6]. Show that the performance of Vamos is dependent on the video motion classification [§ 7.2]. Predict the classification of Vamos on sensor-less citizen journalism videos.

In our experiments we observe a surprising and seemingly counter-intuitive resilience of Vamos against attacks on "stationary" video chunks. We argue this is due to the ability of Vamos to exploit the involuntary user hand shakes that occur during video capture sessions. Furthermore, our experiments show that Vamos differentiates between genuine and fraudulent video and acceleration samples of unconstrained length and motion, with an accuracy that exceeds 93%.

## 2. THE PROBLEM, MOTIVATION AND RELATED WORK

The problem of verifying the authenticity of videos uploaded to a social media site (e.g., from conflicts in Syria, Ukraine or Venezuela) is paramount to the ability to use such videos as evidence or trusted sources for journalism. Citizen Evidence Lab observes that it is common occurrence during complex emergencies and natural disasters for old pictures and videos to be recycled as new online, and go viral due to uncritical re-sharing through social networks [3].

This problem has several dimensions, that include assessing the location and time of capture, or the content of the video. For instance, CitizenEvidenceLab provides tutorials to train the public to asses citizen videos from YouTube (see Figure 1 for a snapshot). InformaCam [7] leverages the unique noise of the device camera to sign content it produces, along with the output of other sensors (e.g., GPS). This enables InformaCam to authenticate that content has been produced with a certain camera. InformaCam assumes that all sensor data is valid and has not been fabricated. It is also vulnerable to plagiarism attacks where the attacker points the camera to a projected video.

In this paper we focus on the liveness dimension of video verifications: verify that the video was captured on a mobile device, and has not been fabricated using material from other sources.

Movee [33] is a liveness verification solution that imposes a 6 second verification step on video capturing experiences: before being allowed to shoot the desired video, the user is presented with a target (bullseye) symbol displayed randomly either on the top, bottom, left or the right side of the screen. The user needs to align the center of the screen to the bullseye, by moving the device in its direction.

Movee uses the correspondence between the motion sensors and video motion to provide a preliminary liveness verification solution. Movee is however severely limited, as (i) the "verification" step is constrained to the initial few seconds of the video, and (ii) the system dictates the user to pan the camera in a specific direction rather than gracefully accept the natural motion of the user. These limitations significantly impact the practical application of Movee.

Furthermore, Movee is vulnerable to the potent attacks that we study in this paper. For example, an attacker starts

Movee and points to a portion of a target video playing on a projection screen, performs a pan motion as specified by Movee, then points the camera to the whole frame of the fraudulent video. Since Movee only uses the initial 6s chunk, the resulting sample passes Movee's verifications. Furthermore, in Section 7.2 we quantitatively show the ineffectiveness of Movee for free-form movements even in 6s chunks: on the attacks we introduce, Movee's false positive rate is as low as 38% and its false negative rate is 28%.

We introduce Vamos to address these limitations and provide the first video liveness verification system that works on unconstrained, free-form videos, does not impose a "verification" step on users, and is resilient to a suite of powerful, sensor based attacks.

Vamos is orthogonal to the problem of finding pirated video on the Internet [28]. Indyk et al. [28] proposed to extract a small number of pertinent features (temporal fingerprints) based on the shot boundaries of a video sequence, and match them against a database of videos. This solution requires thus access to a large dataset of videos, e.g., all the videos uploaded on YouTube. Vamos however works on videos shot on mobile devices and requires access to the simultaneously captured accelerometer data. This makes Vamos resilient to complex plagiarism attacks.

We note that mobile device accelerometers have been shown to be valuable sources for surreptitiously learning typed text. Cai et al. [24] use motion data produced by typing on smartphone virtual keyboards to infer the typed keys. Marquardt et al. [31] show that mobile device accelerometer data, through relative physical position and distance between each vibration, can be used to extract keys typed even on nearby keyboards.

Several video watermarking algorithms has been proposed for video content authentication [35, 25]. The goal of Vamos is however not to authenticate the recorded video, but to verify the video liveness claim. We note that watermarking only works if all the videos in the world employ it. Furthermore, the defenses provided by invisible watermarks are defeated by projection attacks.

Liu et al. [30] proposed a solution for summarizing (i.e., extracting important frames from) mobile videos captured simultaneously with acceleration and orientation streams. The acceleration values are used to exclude outliers. Abdollahian et al. [19] define a "camera view" concept, and use camera motion parameters to temporally segment, summarize and annotate user generated videos. It will be interesting to evaluate a more efficient, video summary based Vamos: detect fraud by identifying discrepancies between video and acceleration summaries.

## 3. SYSTEM AND ADVERSARY MODEL

We consider a system that consists of a service provider (e.g., YouTube [16], iReport [5], Stringwire [11]) and multiple subscribers. The service provider offers an interface for subscribers to upload videos captured on their mobile devices. We assume subscribers own devices equipped with a camera and inertial sensors (i.e., acceleration). Devices have Internet connectivity, which, for the purpose of this work may be intermittent. Each user is required to install an application on her mobile device, which we henceforth denote as the "client".

The client simultaneously captures video and acceleration streams from the device. It then uploads them to her account hosted by the service provider. The provider verifies the data uploaded. If the verification is successful, the provider makes the video publicly accessible. The provider keeps however the acceleration stream secret, or even discards it.

We assume that while the service provider is honest, users can be malicious. Such users can fraudulently claim ownership (creation) of videos they upload. As described above, we assume that attackers do not have access to the acceleration stream of videos they intend to plagiarize. Thus, they are required to fabricate acceleration streams for the uploaded videos.

An acceleration based video liveness verification solution needs to both attest the integrity and authenticity of the acceleration data and to verify the consistency between the acceleration and the video streams. In this paper we focus on the latter problem.

For the former problem, Liu et al. [29] have leveraged ARM TrustZone extensions to implement sensor attestation techniques on an ARM platform. Specifically, in TrustZone, hardware interrupts (of sensors and I/O) can trap directly into the secure monitor code, which can route them to either the secure or the normal world. This allows sensors to map all their interrupts to the secure world. This essentially protects the integrity of the sensors and prevents adversaries from creating and feeding synthetic acceleration data that emulates the movement inferred from the video frames. In addition, apps running in the normal (untrusted) world can use the smc (secure monitor call) instruction to call back into the secure world and access sensor readings.

We note that a system that provides accelerometer attestation does not solve the acceleration to video correlation problem of liveness verification. For instance, the adversary can use a trusted mobile device to capture a projection of a video displayed on a big screen. The resulting video and acceleration samples are genuine, yet, the video is plagiarized.

We introduce several manual and automatic attacks that produce acceleration streams that "match" targeted videos. Let $\mathcal{A}$ denote the algorithm used by the attacker. Let $V$ denote the "target" video that the attacker wants to pass as his own. Let $\Gamma_{\mathcal{A}}$ denote additional information used by the attacker. For instance, $\Gamma_{\mathcal{A}}$ may include other videos and corresponding acceleration streams. We denote the output of $\mathcal{A}$ as $\mathcal{A}(V, \Gamma_{\mathcal{A}}) = \overline{Acc}$, the acceleration stream produced for $V$.

We introduce first the "sandwich" attack, that enables the attacker to manually produce the acceleration data:

**Sandwich attack**. The attacker studies the video $V$ and emulates the observed movement. For instance, $\mathcal{A}$ stacks two devices. The attacker plays the target video $V$ on the top device. He then moves the device stack to emulate the movement seen on the top device. The device on the bottom records the resulting acceleration data, $\overline{Acc}$. $\mathcal{A}$ outputs $\overline{Acc}$.

In the following, we describe the cluster attack, an automatic technique to produce fraudulent data: pair the target video with the acceleration stream copied from a "similar" but genuine sample.

**Cluster attack**. $\mathcal{A}$ captures a dataset of genuine (video, acceleration) samples and stores them in $\Gamma_{\mathcal{A}}$. $\mathcal{A}$ uses a clustering algorithm (e.g., K-means [21]) to cluster the videos based on their movement. $\mathcal{A}$ classifies the target $V$ according to its movement and assigns it to one of the previously generated clusters: the cluster containing videos whose movement is

| Category ID | Distance to subject | User Motion | Camera Motion |
|---|---|---|---|
| 1 | Close | Standing | Stationary |
| 2 | Far | Standing | Stationary |
| 3 | Close | Walking | Stationary |
| 4 | Far | Walking | Stationary |
| 5 | Close | Standing | Scanning |
| 6 | Far | Standing | Scanning |
| 7 | Close | Walking | Scanning |
| 8 | Far | Walking | Scanning |
| 9 | Close | Standing | Following |
| 10 | Far | Standing | Following |
| 11 | Close | Walking | Following |
| 12 | Far | Walking | Following |

**Table 1: Video motion categories, based on (i) camera distance to the subject, (ii) the user motion and (iii) camera motion.**

closest to $V$. $\mathcal{A}$ randomly picks one of the genuine (video, acceleration) samples in the cluster. Let $(V', Acc')$ be the chosen sample. $\mathcal{A}$ outputs $Acc'$.

Next we introduce the stitch attack, that concatenates a plagiarized (video, acceleration) chunk with several genuine chunks. In Section 6.3 we construct stitched samples from multiple fraudulent and genuine chunks.

**Stitch attack**. $\mathcal{A}$ takes as input parameters the target video $V$ and two integers, $g > 0$ and $0 \leq k \leq g$. $\mathcal{A}$ first creates a set of genuine video and acceleration chunks, $\Gamma_{\mathcal{A}} = \{(V_1, Acc_1), .., (V_1, Acc_g)\}$, e.g., by capturing them on the mobile device. $\mathcal{A}$ uses either the cluster or the sandwich attack to fabricate $\overline{Acc}$, an acceleration stream for $V$. $\mathcal{A}$ then "stitches" the fake chunk $(V, \overline{Acc})$ with the genuine chunks $\Gamma_{\mathcal{A}}$, according to the index $k$. Let $||$ denote the concatenation operation, applicable both to video and acceleration streams. Then, $\mathcal{A}$ outputs $(V_a, Acc_a)$, where $V_a = V_1||..V_{k-1}||V||V_{k+1}..||V_g$ and $Acc_a = Acc_1 \; || \; .. \; Acc_{k-1} \; || \; \overline{Acc} \; || \; Acc_{k+1} \; .. \; || \; Acc_g$.

## 4. A CLASSIFICATION OF MOBILE VIDEOS

We posit that the success rate of the attacks previously introduced depends on the type of motions encoded in the video. For instance, it seems intuitive that videos where the hand-held device is stationary are easier to plagiarize. To verify our conjecture, we propose a general classification of videos captured on mobile devices, based on the following dimensions:

- **User motion:** We consider two types of recorder motions, "standing" and "walking", but no motions such as jumping or driving.
- **Camera motion:** We consider three types of camera motions: "stationary", "scanning" and "following". "Scanning" means the camera moves in a direction (e.g., left to right) at a pace independent of the subject of the video. "Following" means the camera moves to maintain the subject within the confines of the video. We have not considered videos shot with head mounted cameras.
- **Distance to subject:** We consider video subjects that are either "close" or "far" to the camera. If the camera focuses on the subject of the video and only a limited area of the background is observed in the video, we say the subject is "close". Otherwise, the subject is "far".

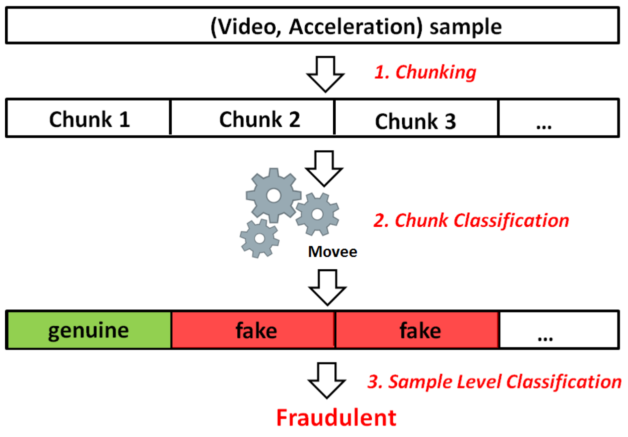


**Figure 2: Illustration of the Vamos architecture and operation. Vamos consists of three steps, (i) "chunking", to divide the (video, acceleration) sample, (ii) chunk level classification, and (iii) sample level classification.**

Table 1 shows the resulting 12 mobile video categories. Figure 5 shows the category distribution of YouTube and free-form video sets we collected (§ 6.1 and § 7.1).

## 5. VAMOS: VIDEO ACCREDITATION THROUGH MOTION SIGNATURES

In this section we introduce Vamos (Video Accreditation Through Motion Signatures) an un-constrained video liveness analysis system. The verifications of Vamos leverage the entire video and acceleration sample. This is in contrast with Movee, that relies only on the initial section of the sample. Vamos consists of the three step process illustrated in Figure 2. First, it divides the input sample into equal length chunks. Second, it classifies each chunk as either genuine or fraudulent. Third, it combines the results of the second step with a suite of novel features to produce a final decision for the original sample. In the following, we detail each of these steps.

### 5.1 Chunk Extraction

The "chunking" process divides a video and acceleration sample $S = (V, Acc)$ into fixed length chunks. We consider a 1s granularity of division. While 6s is the chunk length we use in the experiments, we consider here a parameter $l$ to denote the length in seconds of the chunks. We call a *transition point* (TP) to be the time when the sample transitions from one video motion category to another (e.g., from category 4 to category 8). Let a *transition chunk*, denote a $l$ second chunk that contains parts that belong to multiple video categories. Let $V[s, t]$ and $Acc[s, t]$ denote a segment of $V$ and $Acc$ that starts at second $s$ and ends at second $t$. The chunking process produces a set $C$ of chunks, initially empty. Let $L$ denote the length of the $(V, Acc)$ sample. We propose three chunking techniques, illustrated in Figure 3:

**Sequential chunking**. Divide $(V, Acc)$ into sequential chunks,

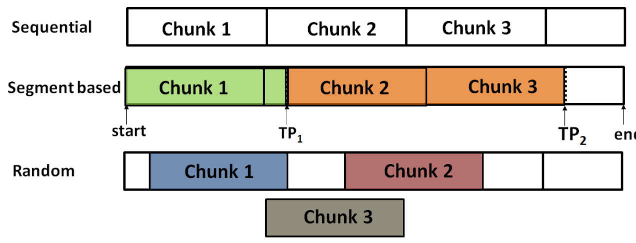


**Figure 3: Chunk extraction illustration. For segment based chunking, the first segment produces a single usable chunk. For random chunking, chunk 3 overlaps both chunks 1 and 2.**

starting with the beginning. Let $n = |C| = \lfloor L/l \rfloor$. Then, $C = \{(V[0, l-1], Acc[0, l-1]), (V[l, 2l-1], Acc[l, 2l-1])..(V[l(c-1), lc], Acc[l(c-1), lc])\}$.

**Segment based chunking**. Identify the transition points of the sample $(V, Acc)$. Let a sample *segment* denote the part of a sample between either (i) the beginning of the sample and the first transition point, (ii) two transition points, or (iii) the last transition point and the end of the sample. Discard all segments of $(V, Acc)$ whose length is less than $l$. Divide remaining segments according to the sequential chunking described above.

**Randomized chunking**. Produces $k$ chunks, $0 < k \leq L$, where $k$ is an input argument, as follows. Generate $k$ different index values within the sample, $0 \leq i_1, .., i_k \leq L$ such that for any $s$ and $t$, $1 \leq s, t \leq k$, $i_s + l \neq i_t$. For each $i_j$, $j = 1..k$, if $i_j \leq L - l$, then $C = C \cup (V[i_j, i_j + l - 1], Acc[i_j, i_j + l - 1])$. Otherwise, $C = C \cup (V[i_j - l, i_j], Acc[i_j - l, i_j])$.

Sequential chunking may produce transition chunks, that contain one or more transition points. Segment based chunking will not produce transition chunks. However, segment based chunking requires a mechanism to identify transition points. Randomized chunking can produce transition chunks and also overlapping chunks. Sequential and segment based chunking produce strictly non-overlapping chunks.

## 5.2 CL-Vamos: Chunk Level Verification

In the second step, Vamos classifies each chunk produced by the first step, as either genuine or fraudulent. While Movee [33] works on fixed length chunks, it is limited to video and inertial sensor streams that encode one of 4 movements (up, down, to the left, or to the right). Specifically, 3 of the 14 features of Movee are (i) the placement of the bullseye relative to the center of the screen, (ii) the dominant video motion direction and (iii) the dominant sensor motion direction.

We introduce here CL-Vamos, the first liveness verification solution that works on free form chunks, that encode unrestricted movements. Similar to Movee, CL-Vamos analyzes the consistency of the inferred motion from the simultaneously and independently captured video and acceleration streams. First, it uses an efficient image processing method to infer a motion vector over the timeline of the video from frame-by-frame progress. Second, it converts the raw inertial sensor readings into a motion vector over the same timeline. Subsequently, CL-Vamos uses the Dynamic Time Warping algorithm (DTW) [32] to find the set of operations that minimizes the cost of converting one vector to the other.

CL-Vamos is not restricted to the dominant direction of movement and removes the features extracted from it. Instead, we have investigated a wide range of features on both the x and y axes. Due to lack of space we report and evaluate here (see Section 7) the feature combination that achieved the best performance.

Specifically, CL-Vamos computes the DTW between the motion vectors extracted from the projections of the video and acceleration streams on both the x and y axes. For each axis, DTW returns the number of diagonal, expansion and contraction moves that convert one vector to the other, and the cost of the resulting transformation. CL-Vamos uses this information to generate the following features, for both the x and y axes:

- The DTW distance (transformation cost) between the video frame shift and acceleration streams.
- The ratio of overlap points: the number of overlapping points in the two motion vectors divided by the length of the vectors.
- The ratio of diagonal, expansion and contraction moves to the number of points in the vectors.

CL-Vamos uses these features, along with other Movee features (e.g., the cumulative shift of the video and accelerometer on the $x$ and $y$ axes), with supervised learning to train classifiers. For each chunk $C_i$ in $C$, let $c_i \in \{genuine, fake\}$ denote the classification produced by CL-Vamos, and let $a_i \in \{genuine, fake\}$ denote the actual status of the chunk. We consider a "positive" to denote a fake chunk, and a "negative" to denote a genuine chunk.

We observe that the false positive rate of CL-Vamos, FPR $= Pr(c_i = fake | a_i = genuine)$. That is, the false positive rate denotes the probability that a chunk is classified as fake (positive), given that the chunk is in fact genuine. Similarly, the false negative rate is FNR $= Pr(c_i = genuine | a_i = fake)$, the true positive rate is TPR $= Pr(c_i = fake | a_i = fake)$ and the true positive rate is TNR $= Pr(c_i = genuine | a_i = genuine)$.

## 5.3 Vamos: Whole Video Classification

Let us assume that for a sample $S = (V, Acc)$, $f$ chunks in $C$ have been classified as fraudulent and $g$ chunks have been classified as genuine. Let $n = f + g = |C|$. We say $S$ is genuine iff $\forall i = 1..n$, $a_i =$ "gen". $S$ is fake if $\exists i$, $i = 1..n$, s.t., $a_i =$ "fake". We can write the probability that the sample $S = (V, Acc)$ is fake, $Pr(S = fake)$, given the above classification result, as

$$Pr[S = fake | \bigwedge_{i=1}^{g} (c_i = gen), \bigwedge_{i=g+1}^{n} (c_i = fake)] =$$

$$= 1 - \Pi_{i=1}^{g} Pr(a_i = gen | c_i = gen) \times$$

$$\Pi_{i=g+1}^{n} Pr(a_i = gen | c_i = fake).$$

Let $\alpha = Pr(a_i = gen | c_i = gen)$, for any of the chunks $C_i$ in $C$. Similarly, let $\beta = Pr(a_i = gen | c_i = fake)$. Then, we have that $Pr(S = fake) = 1 - \alpha^g \times \beta^f$. Now, based on Bayes' theorem, we have that $\alpha = \frac{TNR \times Pr(a_i = genuine)}{TNR \times Pr(a_i = genuine) + FNR \times Pr(a_i = fake)}$. Similarly, we have that $\beta = \frac{FPR \times Pr(a_i = genuine)}{FPR \times Pr(a_i = genuine) + TPR \times Pr(a_i = fake)}$. We can compute thus $\alpha$ and $\beta$ as a function of $Pr(a_i = genuine)$ and $Pr(a_i = fake)$. We obtain these probability values statistically, based on the performance of CL-Vamos on a large number of chunks. Specifically, $Pr(a_i = fake) = \frac{Nr.\ of\ fake\ chunks}{Total\ nr.\ of\ chunks}$ and $Pr(a_i = genuine) = \frac{Nr.\ of\ genuine\ chunks}{Total\ nr.\ of\ chunks}$, see Section 7.4.

We introduce several mechanisms to classify samples as genuine or fraudulent. First, we propose a majority voting approach, where a sample $S = (V, Acc)$ is classified as fraudulent if more than a threshold of the chunks of $S$ have been classified by CL-Vamos as fraudulent: $\frac{f}{f+g} > thr$. The threshold $thr$ is a parameter that will be determined experimentally. Second, we consider a probabilistic approach that labels a sample as fake if $Pr(S = fake) = 1 - \alpha^g \times \beta^f$ is larger than a threshold value. We experiment with threshold values in Section 7.4. Third, we propose a classifier based approach, that uses the following novel features:

- **Results of CL-Vamos**: The number of fraudulent chunks, $f$ and the number of genuine chunks $g$. The classification results $c_i, \forall i = 1..n$. The probability that the sample $S$ is fake, $Pr(S = fake)$.
- **Aggregate features**: For each of the 18 features of CL-Vamos, compute the minimum, maximum, average and standard deviation of the feature's values over $c_i$, $\forall i = 1..n$, as new features.

Vamos uses these features with supervised learning to train classifiers for samples of arbitrary length and encoding arbitrary motions.

## 6. DATA COLLECTION

We have collected datasets of citizen journalism videos from YouTube and of free-form (video, accelerometer) samples from real users. We have also created datasets of fraudulent samples following the attacks introduced in Section 3. In the following we detail each dataset.

### 6.1 YouTube Video Collection

We have collected 150 random citizen journalism videos from YouTube, in the following manner. First, we have identified relevant topics using Wikipedia's "Current Events" site [13], BBC [1] and CNN [2]. They include political events (e.g., Ukraine, Venezuela, Middle East), natural disasters (e.g., earthquakes, tsunamis, meteorite landing), extreme sports and wild life encounters. We have used keywords from such events to identify videos in YouTube that have been captured by a regular person, using a mobile camera. We have discarded videos shot by a professional cameraman or using a head mounted camera. We collected the 150 videos from 139 users accounts. We have made public this list of videos [17]. The total length of the 150 videos is 13,107 seconds. We analyze this dataset in Section 7.1.

### 6.2 Free-Form Video Collection

We performed a user study to simultaneously collect mobile videos shot by users without any motion restrictions, and the associated accelerometer readings. We first briefly describe the implementation of Vamos, then detail the free-form video collection process.

**Vamos implementation**. We have implemented the Vamos client using Android, and a server component using C++, R and PHP. We used the OpenCV (Open Source Computer Vision) library [8] for the video motion analysis. We used Nexus 4 smartphones (Android OS Jelly Bean version 4.2 with 1.5 GHz CPU) to experiment with Vamos. Nexus 4 captures video at 30 fps (frames per second) and samples accelerometer readings at a rate of 16.67Hz [10].

| Category | Chunk count | Category | Chunk count |
|---|---|---|---|
| 1 | 26 | 8 | 28 |
| 2 | 50 | 9 | 26 |
| 3&7 | 82 | 10 | 35 |
| 4 | 18 | 11 | 28 |
| 5 | 44 | 12 | 22 |
| 6 | 42 | | |

**Table 2: Number of chunks of the free-form dataset, per category. Details in Section 7.1.**

**Ethical considerations**. We have used the Vamos application to collect video and acceleration samples from real life users. We have worked with the Institutional Review Board (protocol number IRB-13-0582) at FIU to ensure an ethical interaction with the users and collection of the (video, acceleration) samples.

**Free-form data set**. We have collected data from 16 users[1]. Each user was asked to use Vamos, following the instructions shown on the screen: move the device in any direction to capture videos. Each user contributed 10 free-form videos (and associated accelerometer data), producing a free-form dataset of 160 videos. We have manually annotated the free-form dataset video samples according to the categories described in Table 1.

We have divided each sample of the free-form dataset into 6s chunks, using segment based chunking (see Section 5), producing a total of 401 genuine chunks. Table 2 shows the distribution of the chunks into categories. We have made the free-form dataset publicly available [26].

### 6.3 Attack Datasets

We have used the free-form dataset (see Section 6.2) to create the following attack datasets.

**Sandwich attack dataset**. Two skilled users have performed the sandwich attack on the 160 free-form video dataset. We have used the following procedure, for each whole video (not at chunk level). The attacker watches the target video an unlimited number of times. The attacker stacks two phones. The attacker plays the target video on the top device. The bottom device records the acceleration readings during the session. The attacker can shoot any number of takes, until satisfied with the result.

We combine the original video with the resulting attack acceleration sample to produce a "sandwich sample". We used the segment based chunking method to divide each sandwich sample into 6s (video,acceleration). Thus, each sandwich chunk corresponds to one of the free-form chunks. The sandwich chunk dataset contains thus also 401 chunks.

**Cluster attack dataset**. We ran K-means clustering [21] on the free-form chunk dataset, to cluster the chunks according to their motion (see Cluster attack). We applied the v-fold cross-validation algorithm [20] to determine the optimal number of clusters in our dataset. The outcome was $K = 6$. The cluster attack dataset consists of two subsets, of gen-

[1]11 are males and 5 females, aged 23-32, occupation including biology, fashion design, unemployed, and software, civil and electrical engineering

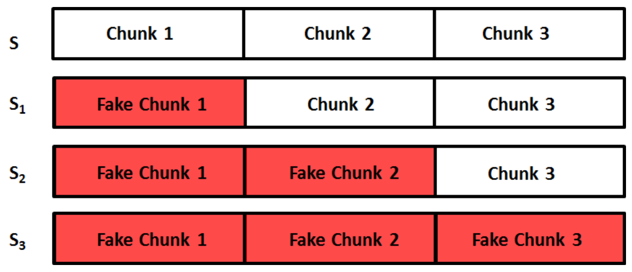


**Figure 4: Stitch attack example. For a genuine sample of 3 chunks, the attacker produces 3 fake samples, with 1 to 3 fake (red) chunks. The genuine chunks are copied from the genuine sample.**

uine and fraudulent (video, acceleration) chunks. We used the free-form chunk dataset as the genuine data. To create the fraudulent subset, for each genuine chunk, we randomly chose another chunk from the same (motion) cluster. We then coupled the video from the first chunk with the inertial sensor data of the randomly selected chunk. Thus, the genuine and fraudulent subsets of the cluster attack dataset each contain 401 chunks.

**Stitch attack datasets**. We have built two stitch attack datasets, one based on the fake cluster chunks and one on the fake sandwich chunks of the previous two attack datasets. The construction process is the following. First, we discarded 4 out of the 160 free-form samples, as they do not have a 6s chunk belonging to a single category. We then discarded 43 samples that have only one chunk. For each of the 113 remaining samples (that has at least 2 chunks), we construct 3 fraudulent samples. For instance, for a 2 chunk genuine sample, we create a fraudulent sample having the first chunk fake, the second genuine, one where the first chunk is genuine, but the second is fake, and one where both chunks are fake. For samples with more than 3 chunks, the position of the fake chunks in any of the 3 created fake samples is randomly selected. The fake chunks are from either the sandwich or the cluster chunk datasets.

Figure 4 illustrates the generation of fraudulent samples given a genuine free-form sample of 3 chunks. The reason for dropping samples with less than 2 chunks is that we need to create the same number of fake samples given any genuine sample (3 fakes per genuine sample). Samples with 1 chunk cannot produce 3 fake stitch samples, thus had to be discarded. The resulting stitch datasets based on the cluster and sandwich attacks have thus each 339 fake samples $((160 - 4 - 43) \times 3)$.

## 7. EVALUATION

We first report our experience in classifying the collected video datasets. We then evaluate the ability of CL-Vamos to classify 6s chunks from the free-form dataset, as either genuine or fraudulent. Finally, we evaluate the performance of Vamos on the whole length samples from the free-form dataset.

### 7.1 Video Dataset Classification

Two users (paper authors) have manually annotated the YouTube and free-form datasets based on the criteria described in Section 4. Since a single video can include sections belonging to different motion categories, the result of the annotation process consists of tuples of the form (*start_time*, *end_time*, *category_id*), where the first two fields denote the start and end time of a video section (measured in seconds) and the last field denotes the id of the video category (integer ranging from 1 to 12). At the end of the process, we have computed a tally of the number of seconds of video belonging to each of the 12 video motion categories.

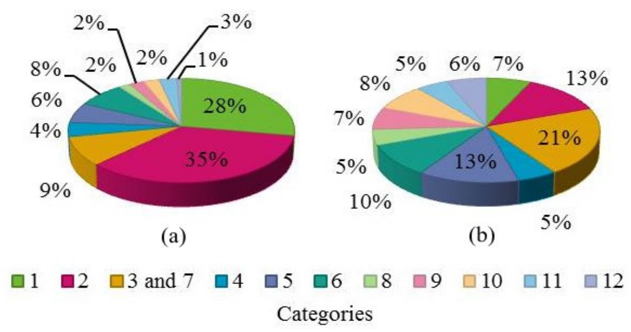


**Figure 5: (a) Motion category distribution for YouTube dataset. (b) Distribution for free-form dataset. Table 1 defines the 12 categories.**

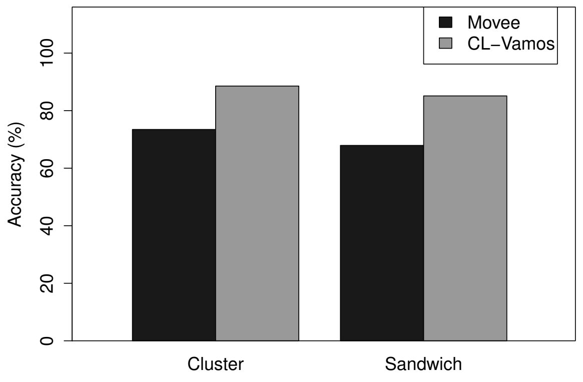


**Figure 6: Accuracy of CL-Vamos and Movee. CL-Vamos improves by more than 15% on Movee, for both cluster and sandwich attacks.**

We have noticed discrepancies between the annotations of the two users: a walking user recording a nearby scene without moving the camera, produces a video that can be (visually) categorized as either 3 or 7. We have labeled these video with a category denoted 3&7.

Figure 5(a) shows the resulting category distribution of the YouTube dataset. The motion categories 2 and 1 have the largest representation, whereas category 12 has the smallest representation. Figure 5(b) shows the category distribution of the free-form dataset, and Table 2 shows the category distribution of the free-form chunks. The difference in distributions of the YouTube and free-form datasets is likely due to the fact that the free-form video collection scenarios have different dynamics from citizen journalism scenarios.

### 7.2 CL-Vamos on Motion Categories

**Experiment setup**. CL-Vamos and Vamos use trained classifiers to determine if a video is genuine. We have experimented with several classifiers, including MultiLayer Per-

| Category | TPR(%) | FPR(%) | FNR(%) | Acc(%) |
|---|---|---|---|---|
| 1 | 75.0 | 16.67 | 25.0 | 80.0 |
| 2 | 82.13 | 16.67 | 17.87 | 83.33 |
| 3&7 | 87.97 | 27.0 | 12.03 | 77.08 |
| 4 | 75.0 | 25.0 | 25.0 | 75.0 |
| 5 | 80.0 | 2.86 | 20.0 | 83.33 |
| 6 | 68.33 | 13.9 | 31.67 | 79.17 |
| 8 | 75.0 | 0.0 | 25.0 | 80.0 |
| 9 | 77.66 | 16.67 | 22.34 | 80.0 |
| 10 | 91.67 | 38.86 | 8.33 | 75.0 |
| 11 | 83.25 | 6.25 | 16.75 | 85.0 |
| 12 | 75.0 | 25.0 | 25.0 | 81.25 |

**Table 3: CL-Vamos accuracy on cluster attack is as high as 85% (on category 11.)**

| Category | TPR(%) | FPR(%) | FNR(%) | Acc(%) |
|---|---|---|---|---|
| 1 | 75.0 | 10.0 | 25.0 | 84.0 |
| 2 | 66.67 | 16.67 | 33.33 | 73.33 |
| 3&7 | 81.34 | 22.58 | 18.66 | 78.75 |
| 4 | 83.34 | 12.5 | 16.66 | 80.0 |
| 5 | 66.0 | 31.32 | 34.0 | 68.75 |
| 6 | 69.34 | 28.0 | 30.66 | 70.0 |
| 8 | 86.0 | 6.66 | 14.0 | 88.0 |
| 9 | 74.34 | 20.0 | 25.66 | 84.0 |
| 10 | 66.67 | 8.0 | 33.33 | 77.14 |
| 11 | 83.34 | 27.34 | 16.66 | 72.0 |
| 12 | 76.68 | 0.0 | 23.32 | 85.0 |

**Table 4: CL-Vamos accuracy on sandwich attack ranges from 68% to 88%.**

ceptron (MLP) [27], Decision Trees (DT) (C4.5), Random Forest (RF) [23] and Bagging [22]. We have used the Weka data mining suite [12] to perform the experiments, with default settings. For the backpropagation algorithm of the MLP classifier, we set the learning rate to 0.3 and the momentum rate to 0.2.

**CL-Vamos vs. Movee**. In a first experiment we compare the efficacy of CL-Vamos and Movee [33] using the following variation of cross validation. For each category $c$, we split the data into 10 equal sized folds. Then, in each of 10 iterations, we train the classifier on 9 folds from all the categories, and test on the data of the remaining fold from $c$. Repeat this operation 10 times, ensuring each fold appears once in the test dataset.

Figure 6 summarizes our results. On the cluster attack, CL-Vamos achieves 88% accuracy when using MLP (Random Forest 83%, Bagging 81%, Decision Trees 78% and SVM 80%). Movee achieves highest accuracy when using the Random Forest (73%). On the sandwich attack, CL-Vamos achieves 85% accuracy when using Random Forest (MLP 71%, Bagging 78%, Decision Trees 74% and SVM 80%). Movee achieves the highest, 67% accuracy, when using either Random Forest or Bagging classifiers. This substantial improvement of CL-Vamos corresponds to an FNR of 6-14% and FPR of 15-17% on these attacks. In contrast, Movee's FNR is between 21-28% and FPR is between 31-38%.

**CL-Vamos: per-category efficacy**. Table 3 shows the TPR, FPR, FNR and accuracy results for CL-Vamos on the cluster chunk dataset, on each of the 11 motion categories. The accuracy ranges between 75% and 85%. Table 4 shows the per-category TPR, FPR, FNR and accuracy results of CL-Vamos on the sandwich chunk dataset. The accuracy ranges from 68% to 88%. We conjecture that its good accuracy for several video categories is due to the difficulty for a human attacker to correctly emulate the movement of the camera, including to estimate distances, observed in a video.

**Category relevance**. We now verify the intuition that the variation in FNR, FPR and accuracy of CL-Vamos is due to its dependence on the video motion categories. We have performed both Pearson's $\chi^2$ and Fisher's exact test on the results CL-Vamos for the sandwich attack dataset. The null hypothesis is that the true positive, false positive, true negative and false negative values are independent of the proposed video categories. The $\chi^2$'s p-value is 0.0001166 and Fisher's p-value is 0.00015. Thus, we reject the null hypothesis and conclude that the performance of CL-Vamos depends on the video motion category.

**Experiment conclusion**. While we expected that certain motion categories are easier to plagiarize, our results are surprising: CL-Vamos does not perform worst on categories 1 and 2, captured by a standing user with a stationary camera. Based on observations from our experiments, we believe that CL-Vamos exploits the ability of accelerometers to capture the small, involuntary hand shakes that occur during video capture sessions. Instead, CL-Vamos has high FPR values for the sandwich attack on categories 5, 6 and 11. This shows that in our experiments, humans are better at plagiarizing videos shot while scanning or following subjects. In Section 7.4 we show that Vamos' overall accuracy exceeds 93% even for the sandwich attack.

## 7.3 CL-Vamos on Citizen Journalism

Current YouTube videos do not have acceleration data. CL-Vamos however only works for video chunks for which we have acceleration data. We propose to use the classification of the collected YouTube videos (see Section 7.1) and the performance of CL-Vamos on the free-form video and acceleration samples (see Section 7.2) to predict its performance on fixed length chunks of citizen journalism videos from YouTube.

Let $Acc(i, FreeForm, AT)$ denote the accuracy of CL-Vamos on videos from the $i$-th category ($i$=1..11) of the free-form dataset, for a given attack type AT. We define the predicted accuracy of CL-Vamos for YouTube and the attack type AT, $Acc_p(YouTube, AT)$, as the weighted sum of its per-category accuracy on the free-form dataset:
$Acc_p(YouTube, AT) = \sum_{i=1}^{11} w_i \times Acc(i, FreeForm, AT)$.
We define the weight $w_i$ to be the percentage of chunks of category $i$ in the YouTube dataset, as shown in Figure 5(a). In the YouTube dataset categories 1 and 2 have the highest weight. The predicted accuracy of CL-Vamos for the cluster attack on the YouTube dataset is then 80.9%, and for the sandwich attack is 77.19%.

## 7.4 Vamos Evaluation

We now evaluate the performance of Vamos on entire video and acceleration samples. We note that a sample can consist of multiple chunks that belong to different motion categories. We have performed experiments using the stitch attack datasets (based on chunk-level cluster and sandwich attacks) described in Section 6.3. The stitch attack datasets

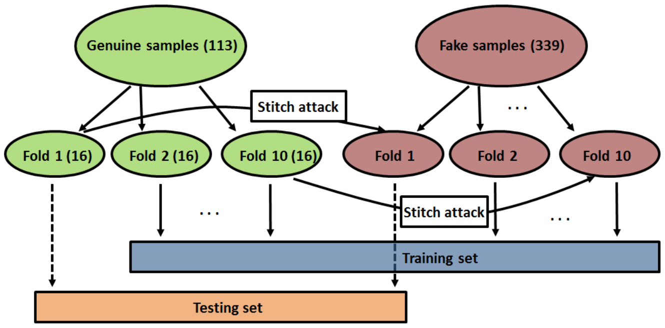


**Figure 7: Setup of Vamos experiment. Each genuine fold produces a stitch attack fold. In each experiment, 9 genuine folds and the corresponding stitch attack folds are used for training. The rest are used for testing.**

| Algo | TPR(%) | FPR(%) | FNR(%) | Acc(%) |
|---|---|---|---|---|
| Maj. Vote | 91.69 | 7.95 | 8.31 | 91.78 |
| Prob. | 91.69 | 7.95 | 8.31 | 91.78 |
| Bagging | 97.35 | 5.08 | 2.65 | 95.53 |

**Table 5: Vamos efficacy on cluster based stitch attack. The classifier approach performs best.**

consist of both genuine and fraudulent free-form samples.

Vamos makes the sample level classification decision based on the classification of the chunks of the sample. In order to avoid a case where the same chunk appears in both training and testing sets, we propose the following experimental design, illustrated in Figure 7.

First, divide the dataset of 113 samples of at least 2 chunks each, into $k$ folds, $gen.fold(i)$, $i = 1..k$, and the corresponding 339 attack sample dataset (either cluster or sandwich attack based) into $k$ folds, $attack.fold(i)$, $i = 1..k$. The split takes place such that the samples from the $gen.fold(i)$ were used to generate the attack samples from $attack.fold(i)$. Then, for each $i = 1..k$, pick all the samples from $gen.fold(j)$ and $attack.fold(j)$, $j = 1..k$, $j \neq i$, and use their chunks to train CL-Vamos. Run the trained CL-Vamos on all the chunks from $gen.fold(i)$ and $attack.fold(i)$. Given the classified chunks of the samples from $gen.fold(i)$ and from $attack.fold(i)$, run the sample level classification step to classify the samples. For instance, to compute $Pr(S = fake)$ for a sample $S$, compute $Pr(a_i = genuine)$ and $Pr(a_i = fake)$ based on the number of fake and genuine chunks in the training folds (see Section 5). In our experiments, we set $k$ to 10.

**Experiment results**. Table 5 reports the performance of Vamos on the cluster based stitch attack dataset. We have

| Algo | TPR(%) | FPR(%) | FNR(%) | Acc(%) |
|---|---|---|---|---|
| Maj. Vote | 74.19 | 35.83 | 25.81 | 71.69 |
| Prob. | 69.95 | 32.50 | 30.05 | 69.34 |
| Bagging | 83.7 | 3.63 | 16.3 | 93.199 |

**Table 6: Vamos performance on sandwich/stitch attack. The classifier approach performs best.**

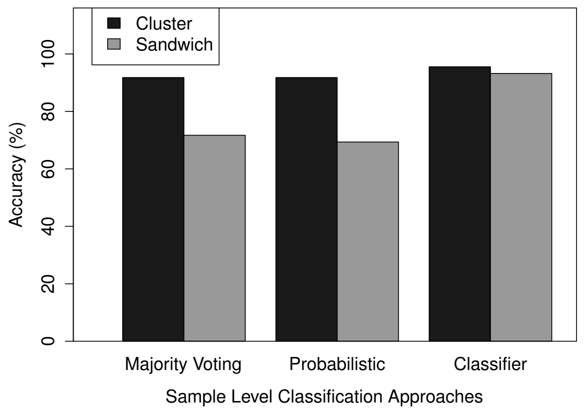


**Figure 8: Vamos accuracy on stitch attacks. Even for the sandwich stitch attack, the classifier approach (using Bagging) exceeds 93% accuracy.**

experimented with multiple threshold values. For majority voting, a threshold of 0.1 performed best: both FPR and FNR values are under 9%. For the probabilistic approach, a threshold of 0.7 achieves similar performance. However, we note that the classifier approach, when using the Bagging algorithm, significantly outperforms the other solutions, with an FPR of around 5% and an FNR of 2.65%.

Table 6 shows the performance of the majority voting, probabilistic and classifier based approaches of Vamos, on the sandwich based stitch attack dataset. For majority voting and probabilistic approaches, the sandwich based stitch attack is significantly more efficient: The majority voting has no threshold where both FPR and FNR are below 35%. The probabilistic approach achieves its optimum for a threshold of 0.8, when its FPR and FNR values are barely under 35%. In contrast, the classifier approach, again when using Bagging, exhibits a significantly improved performance, with an FPR of under 4% and an FNR of 16.3%. Figure 8 summarizes the accuracy of the three approaches of Vamos for the cluster and sandwich based stitching attacks.

We emphasize the importance of this result: while the Movee [33] algorithm exhibits an accuracy of 60-70% on fixed length chunks, Vamos achieves an accuracy that exceeds 93% even on arbitrary length videos, under a *combination* of potent attacks.

## 8. LIMITATIONS

We have not evaluated Vamos against a sandwich attack variant, where a robotic arm [6, 9] holding the mobile device is used to reproduce the motion observed in a target video. While low cost, easy to program robotic arms do exist, they are only capable of jerky, robot-like movements. Such movements are likely different from the fluid movements encoded in human captured video streams; they can thus be detected by Vamos. Therefore, in order for such a robot based attack to bypass Vamos, the adversary needs to invest in a system able to fluidly replicate the wide variety of whole body movements. However, robotic arms that are capable of fluid, human-like movements are significantly more expensive. Thus, Vamos raises the bar for attackers, that need to invest in expensive components.

Furthermore, we have introduced and evaluated Vamos against manual, automatic and mixed attacks. We leave the exhaustive exploration of the attack space for future work.

## 9. CONCLUSIONS

We proposed Vamos, the first length and motion un-constrained video liveness verification system. Vamos uses the entire video and acceleration streams to identify video fraud. We introduced a motion based classification of videos. We evaluated Vamos on data collected from a user study and on citizen journalism videos from YouTube. Vamos has an accuracy exceeding 93% on novel, complex attacks.

## 10. ACKNOWLEDGMENTS

We thank the shepherd and the anonymous reviewers for their excellent feedback. This research was supported in part by DoD grant W911NF-13-1-0142 and NSF grant EAGER-1450619.

## 11. REFERENCES

[1] British Broadcasting Corporation. `http://www.bbc.com/`.

[2] Cable News Network. `www.cnn.com`.

[3] Citizen Evidence Lab. `http://citizenevidence.org/`.

[4] Citizentube. `http://www.citizentube.com/`.

[5] CNN iReport. Explore. Discover. Contribute. `http://ireport.cnn.com/`.

[6] Global Specialties R680 C-Programmable Banshi Robotic Arm. `http://www.testequipmentdepot.com/products.htm?item=R680&ref=gbase&gclid=CNnOrqi2zsICFe7m7AodMS8A8A`.

[7] InformaCam: Verified Mobile Media. `https://guardianproject.info/apps/informacam/`.

[8] Open Source Computer Vision. `http://opencv.org/`.

[9] RobotGeek Snapper Robotic Arm. `http://www.trossenrobotics.com/robotgeek-snapper-robotic-arm?feed=Froogle&gclid=CNOOps22zsICFcxQ7AodOGUArw`.

[10] Sensor Delay. `http://developer.android.com/reference/android/hardware/SensorManager.html`.

[11] Stringwire. Live Video. Made Social. `https://stringwire.com/`.

[12] Weka. `http://www.cs.waikato.ac.nz/ml/weka/`.

[13] Wikipedia current events. `http://en.wikipedia.org/wiki/Category:Current_events`.

[14] Witness. See it. Film it. Change it. `witness.org`.

[15] Witness.org. `http://witness.org/`.

[16] YouTube. `http://www.youtube.com`.

[17] Youtube videos. `http://seclab.cs.fiu.edu/resources`.

[18] Us intelligence officials working to establish authenticity of video of sotloff being killed, reportedly by the same fighter who murdered james foley. http://www.theguardian.com/world/2014/sep/02/isis-video-steven-sotloff-beheading, September 2014.

[19] G. Abdollahian, C. M. Taskiran, Z. Pizlo, and E. J. Delp. Camera motion-based analysis of user generated video. *IEEE Transactions on Multimedia*, 12(1):28–41, 2010.

[20] S. Arlot. Model selection by resampling penalization, 2007.

[21] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[22] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, Aug. 1996.

[23] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[24] L. Cai and H. Chen. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. In *Proceedings of the 6th USENIX Conference on Hot Topics in Security (HotSec)*, 2011.

[25] S.-C. Chu, L. C. Jain, H.-C. Huang, and J.-S. Pan. Error-resilient triple-watermarking with multiple description coding. *Journal of Networks*, 5(3):267–274, 2010.

[26] R. free form dataset. `http://users.cis.fiu.edu/~mrahm004/RATC/`.

[27] S. I. Gallant. Perceptron-based learning algorithms. *Trans. Neur. Netw.*, 1(2):179–191, June 1990.

[28] P. Indyk, G. Iyengar, , and N. Shivakumar. Finding pirated video sequences on the internet. Technical report, Stanford University, 1999.

[29] H. Liu, S. Saroiu, A. Wolman, and H. Raj. Software abstractions for trusted sensors. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 365–378, 2012.

[30] Y. Liu, H. Liu, Y. Liu, and F. Sun. User-generated-video summarization using sparse modelling. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 3909–3915, July 2014.

[31] P. Marquardt, A. Verma, H. Carter, and P. Traynor. (sp) iPhone: Decoding Vibrations from Nearby Keyboards using Mobile Phone Accelerometers. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, pages 551–562. ACM, 2011.

[32] M. MÃijller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, 2007.

[33] M. Rahman, U. Topkara, and B. Carbunar. Seeing is Not Believing: Visual VeriïňĄcations Through Liveness Analysis using Mobile Devices. In *Proceedings of the 29th Annual Computer Security Applications Conference (ACSAC)*, 2013.

[34] T. Shelton. The most disturbing fake videos making the rounds in Syria. `http://www.globalpost.com/dispatch/news/regions/middle-east/syria/121109/fake-syria-videos-images`, November 2012.

[35] W. Zhang, R. Zhang, X. Liu, C. Wu, and X. Niu. A video watermarking algorithm of h. 264/avc for content authentication. *Journal of Networks*, 7(8):1150–1154, 2012.