Towards Untrusted Social Video Verification to Combat Deepfakes via Face Geometry Consistency

Eleanor Tursman Marilyn George Seny Kamara James Tompkin
Brown University

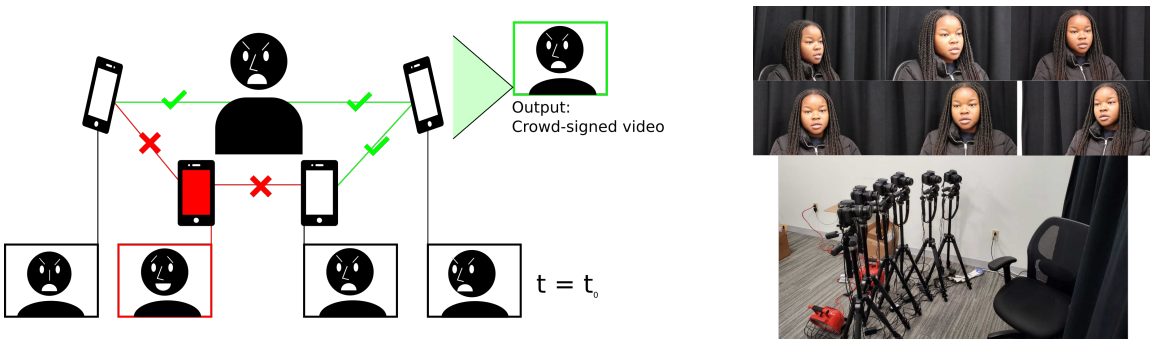


Figure 1: *Left:* Our imagined social verification setting: Multiple people capture a speaker with their smartphones to verify the truth of an event, even if one person is digitally manipulating the face (red). *Right*: Our current proof of concept capture setting.

Abstract

Deepfakes can spread misinformation, defamation, and propaganda by faking videos of public speakers. We assume that future deepfakes will be visually indistinguishable from real video, and will also fool current deepfake detection methods. As such, we posit a social verification system that instead validates the truth of an event via a set of videos. To confirm which, if any, videos are being faked at any point in time, we check for consistent facial geometry across videos. We demonstrate that by comparing mouth movement across views using a combination of PCA and hierarchical clustering, we can detect a deepfake with subtle mouth manipulations out of a set of six videos at high accuracy. Using our new multi-view dataset of 25 speakers, we show that our performance gracefully decays as we increase the number of identically faked videos from different input views.

1. Introduction

Current facial manipulation tools produce video which can fool humans. While these manipulations can be benign or humorous, like a transplant of Nicholas Cage's face onto *every other actor*, many manipulations are malicious. With simple-to-use open-source software, anyone can create convincing propaganda of a politician, can submit manipulated video as evidence in court, or can post revenge pornography of an ex. These faked videos of humans are often created with deep learning [18], and have been dubbed 'deepfakes.'

Deepfake detection approaches combat the spread of misinformation. These methods focus on classifying individual videos as either real or fake, often by detecting visual artifacts such as temporal flicker [14, 7, 11]. While state-of-the-art deepfake generation produces video with minor artifacts, we hypothesize that in the future these 'tells' will be whittled away, and current detection methodologies will struggle. Consider two videos found online, where one is a real video of a speaker, and the other is a visually perfect deepfake of the real video. We assume that we cannot establish which, if either, of these two videos is unmodified.

To provide an additional tool to combat deepfakes, we consider detection via *social verification* at capture time (Figure 1): the arbiters of truthfulness are a *group* of video cameras that synchronously capture a speaker, collectively reach consensus, and then sign their videos in real time as 'true'. This setting makes the creation of fakes more difficult in two ways: 1) attackers must make real time modifications, and 2) attackers must make modifications which fool the consensus-driven representation. Forming this representation is challenging. Cameras must agree that the depiction of the speaker is the same even though they capture the scene at different angles, while also rejecting cameras presenting manipulated content to the group. Further, the operation to reach consensus must be fast to compute and constitute a small amount of data for real-time transfer between the group, e.g., via a peer-to-peer wireless network. However, if we can accomplish this, then we can 'crowd-sign' videos for

authenticity, and provide a basis of truth for verifying other videos uploaded after the event.

Through this process, we assume that individual cameras do not trust one another, but that all parties are motivated to share data with the group by the desire to produce a trustworthy signed video. We imagine citizen journalists with no shared affiliation recording a speech at a protest, or a group of news broadcasters with different political leanings wishing to protect against video stream hacking. Finally, a complete solution to this problem would also include an analysis of the captured audio; our paper currently focuses on the visual appearance of human speakers only.

Our approach Given n video streams of an event, we assume that some k videos are manipulated in real time by an attacker, with $0 \leq k < n$ and $m = n - k$ unedited videos. We assume that $m > k$, i.e., that the largest set of consistent videos are unedited. Our goal is to define a measure by which we can identify the majority of unedited videos. This allows us to decide to sign videos only when the majority is equal to the size of the group, or under more relaxed tolerance constraints, e.g., where one malicious attacker is ostracized.

For this, we look to establish a measure with properties analogous to the collision-resistance of a cryptographic hash function. In a cryptographic hash function, multiple inputs can map to the same output, and any pair of inputs that maps to the same output is known as a collision. A hash function is collision-resistant if it is computationally infeasible to efficiently find a collision in the function. In our visual setting, we look for a visual hash that can be computed *per camera feed*, such that it is difficult to find another 'meaningfully visually different but still convincingly manipulated' video that has the same hash value.

We experiment with *face geometry* as a visual hash. Within our scope, we focus on mouth manipulations, though our method may generalize to other face part manipulation. Intuitively, manipulations to the speaker's mouth shape will create differences in geometry between edited and unedited videos. Face 3D geometry will be view invariant, as long as the speaker's face is mostly un-occluded for all cameras, and will be difficult to replicate, since a convincing manipulation will be reflected in a geometric representation.

First, we collect a multi-view dataset of 25 human speakers from six cameras. Next, we show that off-the-shelf methods cannot reliably fit two established 3D multi-linear face models to our videos with sufficient accuracy to detect mouth differences. Then, we show that a simple and fast variance-based measure on 2D mouth landmarks, computed per camera as a geometric cue, can detect faked mouth motion in a way which is robust to angle changes up to at least 65°, and is somewhat robust to increasing number of fakes. We show that this is better than a naive baseline landmark distance method, and a more complex wavelet-based signal clustering method. In sum, our work provides evidence that untrusted social verification systems are possible, and may prove to be an additional useful tool to combat deepfakes.

Within the span of this work, we contribute:

- A proposal for social verification for this problem space,
- A multi-view video dataset of 25 human speakers in an indoor environment, with deepfakes of each video which manipulates participant mouths via speech changes, and
- A distance based on hierarchical clustering of variance in 2D mouth landmark motion over time and across views.

2. Related Work

2.1. 3D face geometry

The human face can be modelled both spatially and temporally. Parametric models are popular, and are typically constructed from datasets of laser scans [3]. The 3DMM model is a widely used parameterized face model [6]; it is represented by a set of shape, texture, and expression parameters as coefficients to a basis for the model space. We also consider the FLAME model, which similarly parameterizes the face, but uses a more diverse set of input data [12]. In our experiments, we assess 2D landmark models [4] and fitting both the 3DMM 2019 and FLAME models to our data.

2.2. Deepfake creation

Current deepfake manipulations include synthesizing a fully fake face, swapping identities [24], manipulating specific facial attributes, or transferring expressions [27, 23]. Many deepfake creation methods focus uniquely on identity swapping, or on a mix of identity and expression swapping [24, 16]. Expression transfer exemplifies more subtle geometric manipulation than full identity swapping, so to establish the robustness of our proposed geometry-based method, we focus on related works on expression transfer.

Expression transfer can be accomplished with auto-encoders [17], or with GANs paired with a variety of problem-specific losses and modifications [28, 20, 9]. For creating our own fakes for analysis, we use LipGAN, a publicly available model which generates a talking face given a single image or input video, and desired audio output. K R et al. use this network to automate language agnostic translation in video with lip synchronization [9].

Some works are not 'deep.' Thies et al. create Face2Face to transfer the expression of a source actor to a target video in real-time [22]. This method is used to generate a subset of the fakes in FaceForensics++ [19], a dataset used to train and evaluate deepfake detection methods. Averbuch-Elor et al. warp portrait photos to mimic a source video, and transfer fine details like wrinkles to heighten realism. This work can generate convincing manipulations as long as there are no large changes in head pose [2].

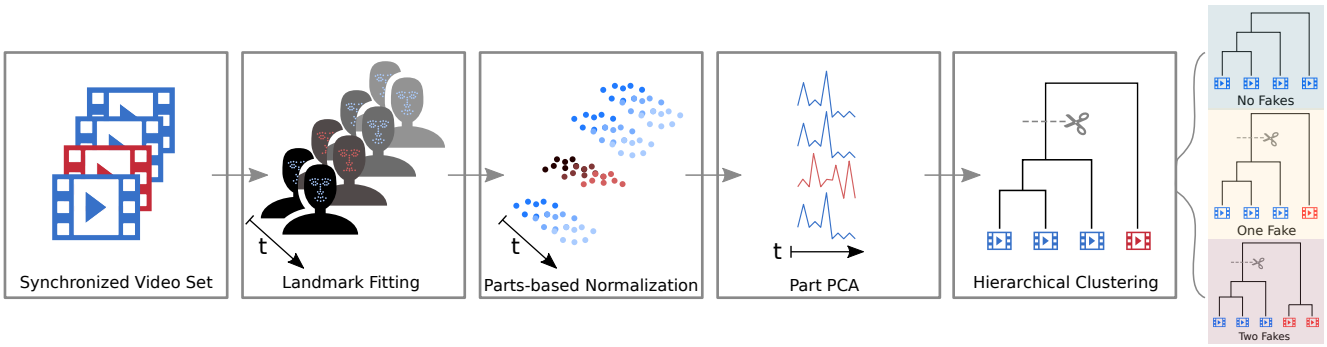


Figure 2: Given a set of time-synchronized input videos with a subset of fakes (red), we first compute 2D facial landmarks per video. Next, we normalize landmark motion across temporal windows per video, then perform PCA on the landmarks. From this, we measure the shape variance as the distance of each timestep's landmarks from the center of PCA space. Finally, we hierarchically cluster distances per window, and threshold the resulting trees to determine which videos are fake. We illustrate hierarchical tree cutting in the no fake, one fake, and two fakes scenarios, with the connection height proportional to its cost.

2.3. Deepfake detection

Detection methods primarily focus on finding artifacts created by sensor noise, compression, or edits. These often use neural networks to decide whether or not a single video has been modified, which can overfit to their training data—the performance of many state-of-the-art detection methods dramatically decays for in-the-wild scenarios that have large variation in compression, noise, and blur [27, 23].

The work of Guera et al. combines a CNN for feature detection and an LSTM for temporal information to predict if a video is fake given two seconds of input. As their training set is small, this work is unlikely to generalize to in-the-wild scenarios [7]. Li et al. look for face warping artifacts caused by resolution changes between source videos and fake output videos [14]. Neves et al. show that GANs leave 'fingerprints' in the high frequency information of their output videos. They show that many existing detection methods rely on detecting these fingerprints, and demonstrate that detection accuracy drops dramatically when they remove them [15].

Detectors can also be trained on subject behaviors. Li et al. look at eye blink rates and abnormal physiological cues to detect fakes [13]. Agarwal et al. show that fakes can be detected by looking for consistent action unit models for persons of interest. They codify a speaker's normal behaviors as clusters in a high dimensional space, and detect fakes if they fall outside these clusters [1]. However, these approaches require an existing corpus of data for a particular speaker.

We can aid detection by creating and publicly releasing training data. Khodabakhsh et al. create a dataset of manipulated images—Fake Faces in the Wild [11]. Rossler et al. create FaceForensics++, a dataset of fake videos manipulated using a variety of methods and at different resolutions to try to overcome detector weakness to the variety of in-the-wild video. They find that expression transfer methods like Face2Face are most difficult for humans to detect, which motivates our use of relatively subtle manipulations in image content [19]. These datasets are all single view, so they are not sufficient for our social verification problem space. Most recently, Jiang et al. create the DeeperForensics-1.0 video dataset, with a set of diverse actors shot from seven views [8]. This data is not yet publicly available.

One final consideration for detection methods is interpretability. Verdoliva describes that successful detection methods based on neural networks are hard to interpret, which is not sufficient for many scenarios where manipulated video could be considered, e.g., legal [27].

3. Method

Suppose a set of C synchronously capturing cameras $c_1, ..., c_i, ..., c_C$, where some subset of cameras produce fake output (Fig. 2). First, we fit landmarks per video per frame using 2D FAN [4]. For our landmark-based clustering, we isolate and normalize our landmarks by face part. Given a set of landmarks per frame $\{\vec{m}_{c_i,1}, \vec{m}_{c_i,2}, ..., \vec{m}_{c_i,n}\}$, where each video c_i is n frames long, we perform classical principal component analysis (PCA) on each set of landmarks corresponding to a face part (e.g., the mouth) to obtain a signal per camera which is representative of the variation in face part motion (Sec. 3.1). Finally, we cluster our PCA results to separate our cameras into real and fake subsets (Sec. 3.2). As a comparison, we also outline our 3D model fitting process (Sec. 3.3), which we use to evaluate parameterized model fitting (Sec. 4.2).

3.1. Landmark PCA

For each camera c_i, we define an $n \times 2p$ matrix M_{c_i}:

$$M_{c_i} = \begin{bmatrix} \vec{m}_{c_i,1} \\ \vec{m}_{c_i,2} \\ \vdots \\ \vec{m}_{c_i,n} \end{bmatrix}$$

$$= \begin{bmatrix} x_{1,1} & y_{1,1} & x_{2,1} & y_{2,1} & \dots & x_{p,1} & y_{p,1} \\ x_{1,2} & y_{1,2} & x_{2,2} & y_{2,2} & \dots & x_{p,2} & y_{p,2} \\ \vdots & & & & \vdots & & \\ x_{1,n} & y_{1,n} & x_{2,n} & y_{2,n} & \dots & x_{p,n} & y_{p,n} \end{bmatrix}.$$

Each of the p part landmarks have some (x, y) pixel coordinates per frame. We then perform classical PCA on each of the matrices $M_{c_1}, ..., M_{c_C}$. Our intuition is that PCA will be able to capture the underlying variance in the landmark movement over time, which we can then use to detect anomalies across face part motion. To represent this variation compactly as a signal over time per camera c_i, at each frame we represent the face's pose as the distance from its projected position in the PCA subspace for M_{c_i}, to the center of that same PCA subspace. We calculate this length using the Mahalanobis distance metric, which can be used to find outliers in multivariate comparisons. We define this $n \times 1$ distance vector for c_i as $\vec{d_i}$, where

$$\vec{d_i} = \sqrt{\sum_{cols} (M_{c_i}^{proj} - A)^T cov(M_{c_i})^{-1} (M_{c_i}^{proj} - A)} \quad (1)$$

$$M_{c_i}^{proj} = (M_{c_i} - \bar{M}_{c_i})E \quad (2)$$

In Eq. 2, $M_{c_i}^{proj}$ represents the data matrix minus its mean multiplied by the $2p \times l$ eigenvector matrix E of the covariance matrix of that same data matrix. In other words, $M_{c_i}^{proj}$ is the centered data matrix projected into PCA space. In Equation 1, $cov(M_{c_i})$ is the covariance matrix of M_{c_i}, and A is an $n \times l$ matrix which represents the $1 \times l$ center of the PCA space of the data matrix with l eigenvectors repeated n times. Our distance vector is then a sum of the resulting inner matrix over columns of the inner matrix result.

Intuitively, we quantify the variance of each of the p (x, y) part landmarks over some set of n frames, with the understanding that regardless of the pose of the head, if these variances are similar, the face parts are moving in similar ways throughout the given set of frames. In this way, we can robustly quantify unusual face motion (see Figure 3).

We use the Matlab LIBRA toolbox for computing classical PCA and Mahalanobis distances per frame [25, 26].

3.2. Social verification

We perform hierarchical clustering of our set of vectors $\vec{d_1}, \vec{d_2}, ..., \vec{d_C}$ to obtain a weighted binary tree t. To perform hierarchical clustering, we first calculate the l_2 distance between all of our leaves, or distance vectors. We pair the two closest leaves with a parent node, which is then weighted with the l_2 distance between the two leaves. We build a weighted binary tree by repeatedly joining the two clusters that contain the two leaves separated by the minimum distance. We use the final tree's shape and costs to determine which, if any, of the inputs are faked, or inconsistent with the majority (Fig. 2).

Given a tree t with k connections and a distance weight w per connection, we calculate

$$\phi = w_k / w_{k-1} \quad (3)$$

Our ϕ represents the ratio of the cost of the ultimate and penultimate connections in t. If ϕ is greater than a set threshold, we say that there is inconsistency within our input set of cameras, and at least one fake is present. We then cluster the tree using a maximum cluster size of two, where one cluster represents the real inputs, and the other represents the fakes. The smaller of the two clusters is then called the fake cluster. If ϕ is instead below this threshold, we say that there are no fakes present.

We limit ourselves to the most difficult scenario where any fakes that are present are manipulated in the exact same way, and will therefore be clustered as a consistent sub-tree in t. In the more trivial cases where all of the fakes are manipulated in different ways, t will have many solitary branches with a high connection cost. In this scenario, we would calculate ϕ moving from the top of the tree down towards the bottom of the tree, pruning off any section that falls above our threshold. For the scope of this work, we focus on the most complex scenario, to ensure that our methodology is feasible and robust in the worst-case scenario.

3.3. Model fitting

We initially use RingNet, a deep learning method, to fit the parameterized FLAME model to our video input [21]. However, we found that RingNet was not able to capture mouth motion well, and appeared to be converging towards a mean shape for each participant. Some visuals of this fitting are in Appendix A. We instead frame model fitting for both 3DMM and FLAME as an optimization problem, where we solve for model parameters and rigid transformation parameters that minimize the re-projection error of model vertices to their corresponding 2D landmark locations. Our results have average re-projection error per pixel on the order of 10^{-1}. We run this fitting for each of our input videos to obtain a set of model parameters per frame per video.

4. Experiments

For our method to represent a visual hash, in the loose cryptographic sense, we need to be certain that our signals from real cameras are view invariant, and that these signals are hard to replicate while maintaining a visual dissimilarity from the real video. We collect data and set up a series of experiments to test both of these qualities.

4.1. Data

Real-world data We collect multi-view video of 25 participants using six Canon EOS T7i Rebel cameras fitted with 18-55mm lenses. These are arranged in an arc of 65°

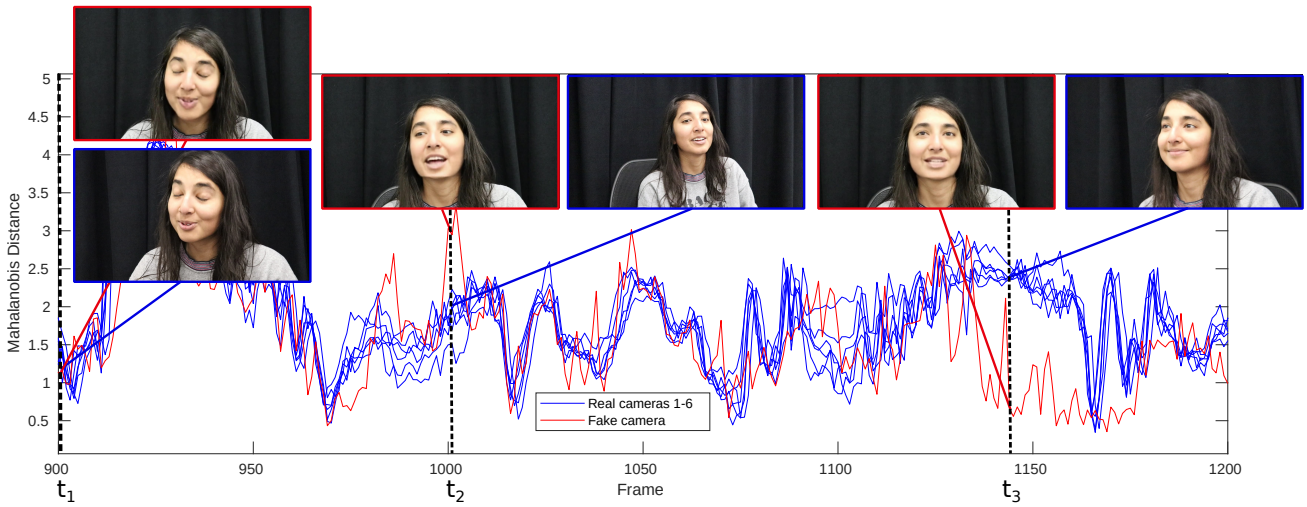
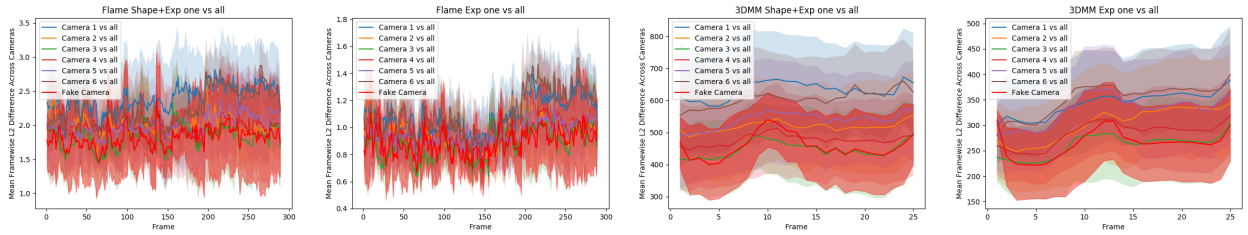


Figure 3: Our Mahalanobis distances plotted over time. We highlight areas where our clustering results could correctly differentiate mouth shape. At t_1, our real video (blue) and fake video (red) are close by Mahalanobis distance, and the real and fake corresponding mouth poses are the same. At t_2, the fake mouth is open wider, and we see a visual spike in the graph. At t_3, we see our biggest difference between the real and fake mouths, where the fake is open, and the real is closed.



(a) FLAME fit with all parameters

(b) FLAME fit with expression parameters

(c) 3DMM fit with all parameters

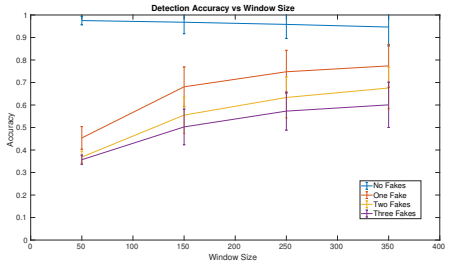
(d) 3DMM fit with expression parameters

Figure 4: To isolate a fake camera from normal fitting variance, the fake camera (red) must be separable from the lines corresponding to real cameras. Both the FLAME and 3DMM parameter sets are unable to differentiate between the real and fake geometries. This result holds when comparing both the full parameter vectors, and the expression parameter vectors. We visualize one standard deviation from the mean.

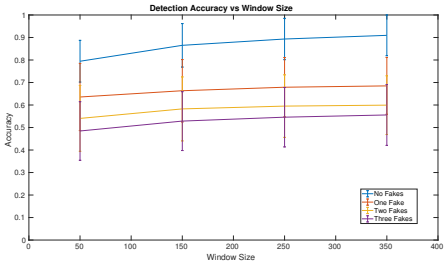
facing a seated speaker set against a black cloth backdrop. External lights are used to keep the scene bright, and camera settings are consistent across all cameras. We capture video synchronously using a trigger box, where each video is on average five thousand frames long. Participants are asked a fixed series of innocuous questions to trigger expressive responses (e.g., "which do you prefer: dogs or cats?"), with the goal of collecting video with a variety of expressions.

Deepfake data We use LipGAN [9] to generate our deepfake videos. We pass the network our input video, one of the original camera results, and an audio track from the LibriTTS corpus dataset [29]. LipGAN synthesizes a new video with modified mouth and jaw area from the input video. This process creates faked scenarios that are more difficult to detect than an identity swap alone. We create a deepfake for each of the six real videos per participant, using the same multi-minute-long audio sequence for all fakes.

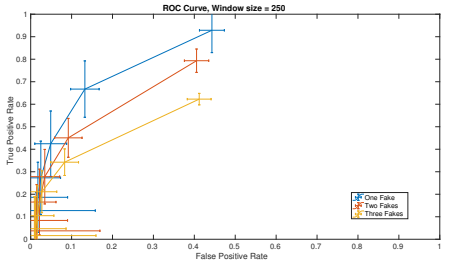
Synthetic data As a proof of concept, we determine whether we are able to capture differences in geometry at the model level in parameter space. Given an idealized scenario where fitting a model to video via landmarks is perfect, we take a mesh represented through the FLAME model, then warp it in mesh space with two different audio samples using VOCA [5]. We show that differences in geometry caused by mouth movement are detectable in the model's parameter space in our supplemental, Appendix A.



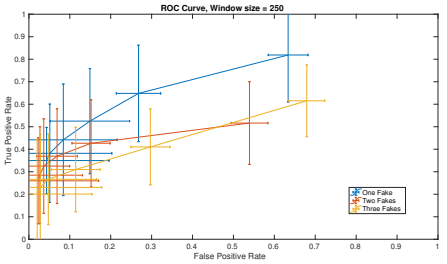
(a) Our method's accuracy



(b) DWT method's accuracy



(c) Our method's ROC curve



(d) DWT method's ROC curve

Figure 5: Accuracy and ROC curves for our method and DWT. Error bars represent one standard deviation from the mean across all participants. Horizontal error in the ROC curves represents uncertainty in the FPR, and vertical error represents uncertainty in the TPR. We plot our ROC curves at a fixed window size of 250 frames. In summary, our method has higher accuracy for all but the smallest window size. Our method is more robust to false positives.

4.2. Experimental Methods

Model evaluation For parameterized models 3DMM and FLAME to be viable, we must confirm that the parameters of the fitted models to the fake video are differentiable from the normal variation in the fits to the real six views.

To compare parameter vectors across videos, we work in a one vs. all manner. Suppose we have the parameter vectors $\vec{p}_{c,1}, ..., \vec{p}_{c,n}$ associated with the n frames of camera c. For every frame, we calculate the mean of the Euclidean distances between all pairs of cameras. More formally, for every frame f, and every camera c, we calculate: $\frac{1}{C}\sum_{j=1,...,C,j\neq c} \|\vec{p}_{c,f} - \vec{p}_{j,f}\|$, where we have C cameras.

Fake detection For these experiments, we use the top five principle components from PCA, $l = 5$, determined empirically. To test our method, we first generate a fake version of the second, third, and fourth camera for every subject, where the fake is as long as the real videos at three and a half minutes, and is generated using the first 4000 frames of the real video. These cameras are the most frontal facing, and therefore have the best LipGAN output quality.

We run two detection experiments, one using full video sequences, and one using sliding windows. For both experiments, we evaluate two baselines in addition to our method. The first baseline is a simple measure of how open the mouth is at a given frame. We take the l_2 distance between the upper and lower lip for each frame of each camera, given the feature points in the middle of the mouth. Each camera then has a vector of distances, which we hierarchically cluster as described in Section 3.2. The second baseline is a direct decomposition of the data matrices M_{c_i} using a discrete wavelet transform (DWT). We use the haar wavelet with the maximum number of levels permitted for the signal length. Then, we hierarchically cluster the sets of wavelet coefficients per camera, as described in Section 3.2. We do not compare against existing detection methods, since we assume in our problem setup that the fake is already able to fool single-video-based detectors.

We create four sets of inputs to use for both experiments:

1. Real = $\{1, 2, 3, 4, 5, 6\}$
2. Real = $\{1, 2, 3, 5, 6\}$ and Fake = $\{4\}$
3. Real = $\{1, 2, 5, 6\}$ and Fake = $\{3, 4\}$
4. Real = $\{1, 5, 6\}$ and Fake = $\{2, 3, 4\}$

The worst case scenario for our problem space occurs when we have multiple attackers manipulating video streams in identical ways, such that the geometry across fakes is consistent. We test how many fakes we can tolerate in this setup, while still detecting the correct subset of real videos.

Full Sequence Experiment In the first experiment, we test whether our method can pick out which, if any, of the

Table 1: Accuracy and false positive rates for the compared methods; DWT and our method have ratio threshold of 1.3. The landmark-based simple mouth metric fails for all window sizes; DWT performs better; our method consistently performs better still at all but the smallest window sizes. We bold the highest accuracy per ratio of real to fake inputs for non maximum-window sizes, and their corresponding false positive rates. We also include whole video window sizes (*'max'*). The false positive rate is not computable for the 'max' case since all frames are faked, making false positives and true negatives zero.

Method	Window	Accuracy for # fakes / # real				False positive rate for # fakes / # real			
	# frames	0 / 6	1 / 5	2 / 4	3 / 3	0 / 6	1 / 5	2 / 4	3 / 3
Simple	50	0.32 ± 0.03	0.20 ± 0.06	0.16 ± 0.03	0.15 ± 0.03	0.68 ± 0.03	0.79 ± 0.03	0.81 ± 0.03	0.84 ± 0.02
Mouth	150	0.33 ± 0.05	0.21 ± 0.08	0.16 ± 0.05	0.16 ± 0.04	0.67 ± 0.05	0.78 ± 0.05	0.80 ± 0.04	0.83 ± 0.05
	250	0.35 ± 0.08	0.22 ± 0.10	0.17 ± 0.06	0.16 ± 0.05	0.65 ± 0.08	0.76 ± 0.07	0.78 ± 0.08	0.82 ± 0.06
	350	0.35 ± 0.09	0.22 ± 0.11	0.19 ± 0.07	0.17 ± 0.05	0.65 ± 0.09	0.75 ± 0.09	0.75 ± 0.10	0.81 ± 0.07
	max	0.40 ± 0.50	0.24 ± 0.44	0.08 ± 0.28	0.00 ± 0.00	-	-	-	-
DWT	50	0.79 ± 0.09	0.64 ± 0.15	0.54 ± 0.15	0.48 ± 0.13	0.21 ± 0.09	0.31 ± 0.15	0.21 ± 0.10	0.39 ± 0.14
	150	0.87 ± 0.10	0.66 ± 0.14	0.58 ± 0.14	0.53 ± 0.13	0.13 ± 0.10	0.28 ± 0.15	0.16 ± 0.09	0.32 ± 0.14
	250	0.89 ± 0.09	0.68 ± 0.13	0.60 ± 0.14	0.55 ± 0.13	0.11 ± 0.09	0.27 ± 0.15	0.15 ± 0.10	0.30 ± 0.15
	350	0.91 ± 0.09	0.69 ± 0.13	0.60 ± 0.13	0.56 ± 0.14	0.09 ± 0.09	0.27 ± 0.16	0.16 ± 0.12	0.29 ± 0.17
	max	0.60 ± 0.50	0.48 ± 0.51	0.36 ± 0.49	0.28 ± 0.46	-	-	-	-
Ours	50	$\mathbf{0.98 \pm 0.02}$	0.45 ± 0.05	0.37 ± 0.03	0.36 ± 0.02	$\mathbf{0.02 \pm 0.02}$	0.03 ± 0.02	0.04 ± 0.02	0.04 ± 0.03
	150	0.97 ± 0.05	0.68 ± 0.09	0.56 ± 0.08	0.50 ± 0.08	0.03 ± 0.05	0.08 ± 0.05	0.06 ± 0.06	0.06 ± 0.05
	250	0.96 ± 0.06	0.75 ± 0.10	0.63 ± 0.09	0.57 ± 0.08	0.04 ± 0.06	0.13 ± 0.07	0.09 ± 0.08	0.08 ± 0.07
	350	0.95 ± 0.08	$\mathbf{0.77 \pm 0.10}$	$\mathbf{0.68 \pm 0.09}$	$\mathbf{0.60 \pm 0.10}$	0.05 ± 0.08	$\mathbf{0.18 \pm 0.10}$	$\mathbf{0.13 \pm 0.10}$	$\mathbf{0.11 \pm 0.10}$
	max	0.88 ± 0.33	1.00 ± 0.00	0.92 ± 0.28	0.76 ± 0.44	-	-	-	-

inputs are fake. Given all four input scenarios, we calculate a clustering with a threshold of 1.3 for both baselines and our method. We average our results across the video of all 25 participants.

Sliding Window Experiment In a real-world scenario, we will not have the luxury of processing three minutes of video at a time, where one video is fully fake throughout. We therefore set up a second experiment where we use sliding windows through time. We increase the complexity of the fake by interleaving it with its real counterpart. For example, if camera four is our one fake camera, we use the LipGAN output for the first and last third of the video, and use the real camera four data for the middle third of the fake video. We average our results across the video of all 25 participants.

Tolerance to viewpoint angle We evaluate the pose-invariance of our method by calculating detection accuracy over pairs of real cameras which are a fixed number of degrees apart. Our input to our method is a pair of real cameras, and the fake of camera four. We take the average accuracy over all sets whose real camera pair is separated by the same angular distance. These results are further compacted as an average across all 25 participants. We estimate angular distance between cameras from our real world setup, use a window size of 250 frames, and use a threshold of 1.3.

4.3. Results

Model accuracy As shown in Figure 4, for both models, the fake camera (red) is not notably different from the variance in the fit of the real geometry. This result holds true regardless of whether we look at the subset of parameters related to expression. Even though the overall re-projection error of our fitting methods is low, the landmark-driven fit to the mouth is not accurate enough to capture the geometric differences between the real and fake videos. Therefore this model fitting approach for FLAME and 3DMM is not a currently viable method for establishing if the cameras are filming consistent face geometry.

Detection accuracy Based on our results (see Table 1), we see that in scenarios where we have one video that is faked at every frame, our method outperforms DWT for every combination of fakes and reals.

In the windowed experiment, we see in Figure 5, while the overall accuracy of DWT is similar to that of our method, its false positive rate is also much more drastic. Our method is more robust to the scenarios with no fakes. We see that our method's performance decays as anticipated as we increase the number of fake cameras. We test window sizes of 50, 150, 250, and 350 frames. For our ROC curves, we generate data using thresholds 1.1, 1.3, 1.5, 1.7, 1.9, and 2.1. Our method, implemented in Matlab on a CPU, runs in 0.0112 seconds on average per 250 frame window.

To better understand how our accuracy results change over window size, and to visualize where our method fails to detect a fake, we plot histograms of the number of properly detected fakes and the number of missed fakes in Figure 6. The x-axis demonstrates the average l_2 difference between LipGAN's mouth landmarks and the real video's mouth landmarks. Intuitively, when this distance is small, the fake and real mouths look similar to one another. We see that when this distance is small, and the window size is also

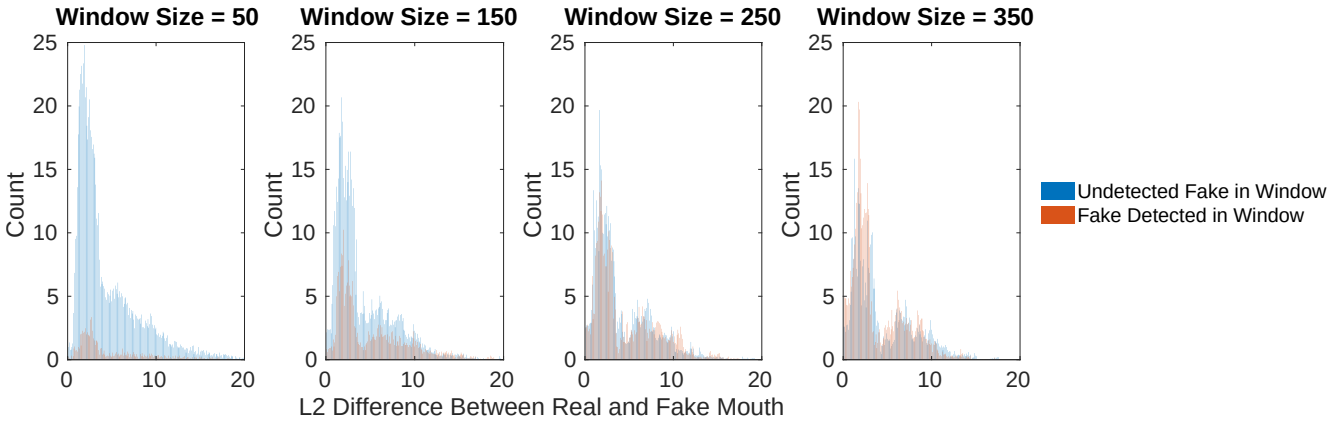


Figure 6: As window size increases, our method becomes better at picking out fakes, even when the lip motion of the fake is visually similar to that of the real mouth. For all histograms, we have one fake, and use a threshold of 1.3.

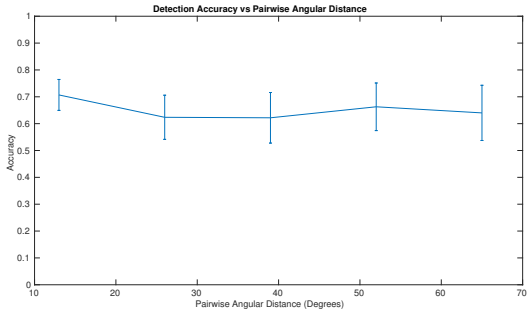


Figure 7: Accuracy vs. pairwise camera angle. We compute clustering accuracy across all pairs of real cameras separated by a set number of degrees, plus one fake camera. Our accuracy remains consistent as angular distance increases, showing that our results are robust to different camera poses.

small, it is difficult for our method to detect fakes. However, as the window size increases, our method's performance improves dramatically. Most of the error associated occurs when the motion of the fake and real mouths is similar.

Viewpoint angle As per Figure 7, our detection accuracy remains stable around 0.65, with a small uptick towards 0.70 for cameras that are closest together. We see through this experiment that we are also able to detect fakes at 65% accuracy with window sizes of 250 frames, even though we have decreased the number of real cameras from six to two.

4.4. Limitations

Our method will not be robust if the mouth itself is small relative to the overall video size, since that will lead to smaller and less distinctive mouth motion. We hypothesize that we see a drop in accuracy between processing full sequences and windows for two reasons. The first is that in a window, we are more beholden to the l_2 difference between the real and fake mouth landmarks. The second is that our method processes windows independently from one another, so we do not temporally propagate a fake detection.

However, it is worth noting that for our desired application, we seek to have our method running while cameras are actively filming content. To properly detect a fake, we do not necessarily have to detect every faked frame perfectly—we just need to isolate fakes over time with high accuracy, while keeping all of our real videos in one set.

5. Conclusion

We move towards a social verification system to combat deepfakes. We can detect mouth manipulations by hierarchically clustering signals based on the quick-to-compute variance of mouth landmark motion in a set of videos. Based on our experiments, we require a window size of around 8 seconds to detect one fake out of a set of six at 75% accuracy. As we compute this metric *per video*, we side-step complex multi-view reconstruction issues, and propose a system where we rely on majority rule for cross-video consistency where no individual video needs to have trustworthy content.

Moving forward, we aim to implement the system on a set of smartphones for real time capture. While expensive to compute, we also plan to explore latent representations for the geometry in the video, e.g., comparing face embeddings in the StyleGAN latent space [10]. Finally, and conceptually, no technology is a panacea. A social verification system can still be gamed, e.g., a 'deepfake flashmob' could verify an event given sufficient bad actors. Our work explores the feasibility of the underlying social verification as a way to provide an additional tool to combat deepfakes.

Acknowledgements

Thank you to Jessica Forde and Kweku Kwegyir-Aggrey for work in initial explorations. Thank you to Purvi Goel, Naveen Srinivasan, and Natalie Lindsay for data collection and snacks. Thank you to NVIDIA for donating a GPU.

References

[1] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–45, 2019.

[2] Hadar Averbuch-Elor, Daniel Cohen-Or, Johannes Kopf, and Michael F. Cohen. Bringing portraits to life. *ACM Transactions on Graphics (Proceeding of SIGGRAPH Asia 2017)*, 36(6):196, 2017.

[3] Stefano Berretti, Mohamed Daoudi, Pavan Turaga, and Anup Basu. Representation, analysis, and recognition of 3d humans: A surveyrossler2019faceforensics. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(1s):16, 2018.

[4] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.

[5] Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael Black. Capture, learning, and synthesis of 3D speaking styles. *Computer Vision and Pattern Recognition (CVPR)*, pages 10101–10111, 2019.

[6] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Luthi, Sandro Schönborn, and Thomas Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 75–82. IEEE, 2018.

[7] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[8] Liming Jiang, Wayne Wu, Ren Li, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. *arXiv preprint arXiv:2001.03024*, 2020.

[9] Prajwal K R, Rudrabha Mukhopadhyay, Jerin Philip, Abhishek Jha, Vinay Namboodiri, and C V Jawahar. Towards automatic face-to-face translation. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, pages 1428–1436, New York, NY, USA, 2019. ACM.

[10] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. *CoRR*, abs/1912.04958, 2019.

[11] Ali Khodabakhsh, Raghavendra Ramachandra, Kiran Raja, Pankaj Wasnik, and Christoph Busch. Fake face detection methods: Can they be generalized? In *2018 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–6. IEEE, 2018.

[12] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), 2017.

[13] Y. Li, M. Chang, and S. Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, Dec 2018.

[14] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[15] Joao C Neves, Ruben Tolosana, Ruben Vera-Rodriguez, Vasco Lopes, and Hugo Proença. Real or fake? spoofing state-of-the-art face synthesis detection systems. *arXiv preprint arXiv:1911.05351*, 2019.

[16] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *ICCV*, 2019.

[17] Shengju Qian, Kwan-Yee Lin, Wayne Wu, Yangxiaokang Liu, Quan Wang, Fumin Shen, Chen Qian, and Ran He. Make a face: Towards arbitrary high fidelity face manipulation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10033–10042, 2019.

[18] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics: A large-scale video dataset for forgery detection in human faces. *arXiv preprint arXiv:1803.09179*, 2018.

[19] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–11, 2019.

[20] Enrique Sanchez and Michel Valstar. A recurrent cycle consistency loss for progressive face-to-face synthesis. In *IEEE Int'l Conf. on Automatic Face and Gesture Recognition (FG)*, 2020.

[21] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael Black. Learning to regress 3D face shape and expression from an image without 3D supervision. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7763–7772, June 2019.

[22] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos.

In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2387–2395, 2016.

[23] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection, 2020.

[24] @torzdf, @andenixa, and @kvrooman. Faceswap.

[25] Sabine Verboven and Mia Hubert. Libra: a matlab library for robust analysis. *Chemometrics and intelligent laboratory systems*, 75(2):127–136, 2005.

[26] Sabine Verboven and Mia Hubert. Matlab library libra. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):509–515, 2010.

[27] Luisa Verdoliva. Media forensics and deepfakes: an overview, 2020.

[28] Wayne Wu, Yunxuan Zhang, Cheng Li, Chen Qian, and Chen Change Loy. Reenactgan: Learning to reenact faces via boundary transfer. In *ECCV*, September 2018.

[29] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

A. Appendix

A.1. Synthetic Data Collection

We collect synchronized video of a seated speaker from four different views. The speaker is prompted to make a wide variety of facial expressions, and to turn their head dramatically in the space. FLAME is then fit to each frame using RingNet. For each frame, we save the shape, expression, and jaw pose parameters of the FLAME model.

Next, we create synthetic adversarial examples in the FLAME model space using VOCA. Given a neutral FLAME model, and an audio sample, VOCA animates the model to synchronously mouth the audio. We create two fake videos: In the first, we keep the expression neutralized, and maintain the shape, or identity-related, parameters. In the second, we match the average expression across the real videos in addition to matching the shape parameters (Figure 9).

A.2. Synthetic Data Experiments

We aim to confirm that in an idealized scenario, where we are perfectly able to capture face geometry from an image to create a detailed model, that we are able to detect small changes in mouth pose. First, we need to determine whether the FLAME models fit across different views look the same. Qualitatively, we see in Figure 9 that the shape, expression, and jaw pose remain invariant across views.

In Table 2, we see that a fake video can most easily vary in jaw pose while remaining undetected. The fit of RingNet is much more sensitive to the shape and expression parameter space across views. To determine if our fakes are detectable, given this sensitivity to the model fitting, we employ a one versus all approach. For each frame, we look at the l_2 difference between the parameter vectors for every possible pair of cameras. We use both the expression and jaw pose parameters alone, and the full set of shape, expression, and jaw pose parameters.

In Figure 8, we see that if we use expression parameters, we are only able to isolate the easy fake case from the rest of the pack. However, if we look at the full set of parameters, we are able to isolate both the easy and hard fake cases from our real video, except for a small section at frame 900. When we plot the mean l_2 difference between the set of real cameras, and sets including different fakes, we see that we can differentiate between the real and fake cases.

We find that the parameter space is fairly invariant across our real views. We can also detect fake videos that only differ from the real videos in mouth pose and expression. These results demonstrate a proof of concept for using geometric cues with social verification to detect video manipulation.

Table 2: To detect a fake, the fake must be outside of the range of normal fitting error among the real cameras. Shape is most consistent across cameras, and jaw pose, which controls how open the mouth is, has the most variation.

Parameters	Mean frame-wise standard deviation across real cameras
Shape	± 0.1037
Expression	± 0.2025
Jaw pose	± 1.1690

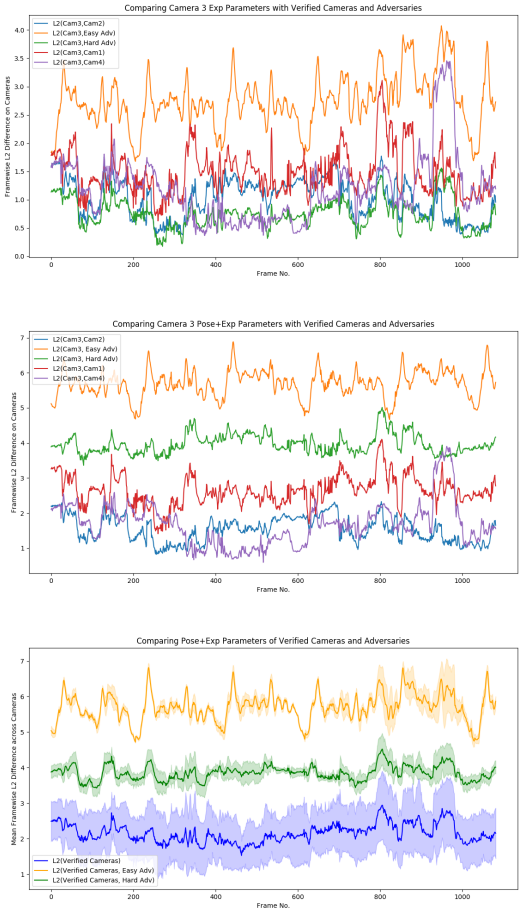


Figure 8: *Top:* We compare expression parameters of camera three against all other inputs. While the easy adversarial case (orange) is separable from most of the real data, the hard adversarial case (green) is not. *Middle:* We compare the full parameter set of camera three against all other inputs. Both easy and hard fake cases are separable from the real video. *Bottom:* We look at the mean frame-wise l_2 difference across the real cameras only (blue), the real cameras and the easy fake case (yellow), and the real cameras and the hard fake case (green). We plot one standard deviation from the mean in all cases. All fake cases are separable from the real cases.

Figure 9: *Top:* Four input frames from four different views, synchronized in time. *Middle:* We show the RingNet fitting output, which stays fairly consistent across views, though it is not able to capture the furrowed brow, or much interesting face shape. *Bottom:* On the left, the easy faked case has a neutral expression, and a different viseme from the original input. On the right, the hard faked case has the averaged expression of the input models, and the new viseme from the easy case.