FaceRevelio: A Face Liveness Detection System for Smartphones with a Single Front Camera

Habiba Farrukh
hfarrukh@purdue.edu
Purdue University

Reham Mohamed Aburas
raburas@purdue.edu
Purdue University

Siyuan Cao
cao208@purdue.edu
Purdue University

He Wang
hw@purdue.edu
Purdue University

ABSTRACT

Facial authentication mechanisms are gaining traction on smartphones because of their convenience and increasingly good performance of face recognition systems. However, mainstream systems use traditional 2D face recognition technologies, which are vulnerable to various spoofing attacks. Existing systems perform *liveness detection* via specialized hardware, such as infrared dot projectors and dedicated cameras. Although effective, such methods do not align well with the smartphone industry's desire to maximize screen space.

This paper presents a new liveness detection system, *FaceRevelio*, for commodity smartphones with a single front camera. It utilizes the smartphone screen to illuminate a user's face from multiple directions. The facial images captured under varying illumination enable the recovery of the face surface normals via photometric stereo, which can then be integrated into a 3D shape. We leverage the facial depth features of this 3D surface to distinguish a human face from its 2D counterpart. On top of this, we change the screen via a *light passcode* consisting of a combination of random light patterns to provide security against replay attacks. We evaluate *FaceRevelio* with 30 users trying to authenticate under various lighting conditions and with a series of 2D spoofing attacks. The results show that using a passcode of 1s, *FaceRevelio* achieves a mean EER of 1.4% and 0.15% against photo and video attacks, respectively.

CCS CONCEPTS

• **Security and privacy** → *Biometrics*; *Mobile and wireless security*.

KEYWORDS

Liveness detection, user security and privacy, 3D reconstruction, face authentication

ACM Reference Format:
Habiba Farrukh, Reham Mohamed Aburas, Siyuan Cao, and He Wang. 2020. FaceRevelio: A Face Liveness Detection System for Smartphones with a Single Front Camera. In *The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21–25, 2020, London, United Kingdom.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3372224.3419206

MobiCom '20, September 21–25, 2020, London, United Kingdom

ACM ISBN 978-1-4503-7085-1/20/09…$15.00
https://doi.org/10.1145/3372224.3419206

1 INTRODUCTION

Considering the growingly extensive use of smartphones in all aspects of our daily life, reliable user authentication for securing private information and mobile payments is an absolute necessity. Recent years have witnessed a rising usage of face authentication on smartphones as a promising alternative to traditional password-based protection mechanisms. Most of the existing face authentication systems use traditional 2D face recognition technologies, which suffer from vulnerability to *spoofing* attacks where the attacker uses 2D photos/videos or 3D masks to bypass the authentication system.

Recently, some smartphone manufacturers have introduced *liveness detection* features to some of their high-end products, e.g. iPhone X/XR/XS and HUAWEI Mate 20 Pro. These phones are embedded with specialized hardware components on their screens to detect the 3D structure of the user's face. For example, Apple's TrueDepth system [1] employs an infrared dot projector coupled with a dedicated infrared camera beside its traditional front camera.

Although effective, deployment of such specialized hardware components, adding a notch on the screen, is against the bezel-less trend in the smartphones' market. Customers' desire for higher screen-to-body ratio has consequently forced manufacturers to search for alternative methods. For example, Samsung recently launched S10 as its first phone with face authentication and an Infinity-O hole-punch display. However, S10's lack of any specialized hardware for capturing facial depth, made it an easy target for 2D photo or video attacks [8].

Therefore, in this paper we ask the following question: How can we enable liveness detection on smartphones only relying on a single front camera?

Prior works on face liveness detection for defense against 2D spoofing attacks have relied on computer vision techniques to detect and analyze textural features for facial liveness clues like nose and mouth features [15, 25], and skin reflectance [36]. Usually, extracting such characteristics from a face requires ideal lighting conditions, which are hard to guarantee in practice. Another common approach is the use of challenge-response protocols where the user is asked to respond to a random challenge, such as pronouncing a word, blinking or other facial gestures. These techniques, however, are unreliable because facial gestures can be simulated using modern technologies, such as media-based facial forgery [29]. A time-constrained protocol was recently introduced to defend against these attacks, which however still required the users to make specific expressions [37]. The additional time-consuming efforts and their reliance on users' cooperation, make such protocols harder to use in many scenarios, including but not limited to elderly usage and emergency cases.

In this paper, we introduce a novel face liveness system, *FaceRevelio*, that only uses the front camera on commodity smartphones. Our system reconstructs 3D models of users' faces in order to defend against 2D-spoofing attacks. *FaceRevelio* exploits smartphone screens as light sources to illuminate the human face from different angles. Our main idea is to display combinations of light patterns on the screen and simultaneously record the reflection of those patterns from the users' faces via the front camera. We employ a variant of photometric stereo [20] to reconstruct 3D facial structures from the recorded videos. To this end, we recover four *stereo images* of the face from the recorded video via a least squared method and use these images to build a normal map of the face. Finally, the 3D model of the face is reconstructed from the normal map using a quadratic normal integration approach [34]. From this reconstructed model, we analyze how the depth changes across a human face compared to model reconstructed from a photograph or video and train a deep neural network to detect various spoofing attacks.

Implementing our idea of reconstructing the 3D face structure for liveness detection using a single camera involved a series of challenges. First, displaying simple and easily forgeable light patterns on the screen makes the system susceptible to replay attacks. To secure our system from replay attacks, we designed the novel idea of a *light passcode*, which is a random combination of patterns in which the screen intensity changes during the process of authentication, such that an attacker would be unable to correctly guess the random passcode. Second, in the presence of ambient lighting, the intensity of the reflection of our light passcode was small, hence difficult to separate from ambient lighting. In order to make *FaceRevelio* practical in various realistic lighting conditions, we carefully designed light passcodes to be orthogonal and "zero-mean" to remove the impact of environment lighting. In addition, we had to separate the impact of each pattern from the mixture of captured reflections to accurately recover the stereo images via the least square method. For this purpose, we linearized the camera responses by fixing camera exposure parameters and reversing gamma correction [7]. Finally, the unknown direction of lighting used in the four patterns causes an uncertainty in the surface normals computed from the stereo images which could lead to inaccurate 3D reconstruction. We designed an algorithm to find this uncertainty using a general template for human surface normals. We used landmark-aware mesh warping to fit this general template to users' face structures.

FaceRevelio is implemented as a prototype system on Samsung S10 smartphone. By collecting 4900 videos with a resolution of 1280×960 and a frame rate of $30fps$, we evaluated *FaceRevelio* with 30 volunteers under different lighting conditions. *FaceRevelio* achieves an EER of 1.4% for both dark and day light settings, respectively against 2D printed photograph attacks. It detects the replay video attacks with an EER of 0.0%, and 0.3% for each lighting, respectively.

The contributions in this paper are summarized as follows:

(1) We design a liveness detection system for commodity smartphones with only a single front camera by reconstructing the 3D surface of the face, without relying on any extra hardware or human cooperation.
(2) We introduce the notion of *light passcodes* which combines randomly-generated lighting patterns on four quarters of the screen. Light passcode enables reconstructing 3D structures from stereo images and more importantly, defends against replay attacks.
(3) We implement *FaceRevelio* as an application on Android phones and evaluate the system performance on 30 users in different scenarios. Our evaluations show promising results on applicability and effectiveness of *FaceRevelio*.

2 BACKGROUND

In this section, we introduce photometric stereo and explain how it is used for 3D reconstruction under known/unknown lighting conditions.

Photometric stereo is a technique for recovering the 3D surface of an object using multiple images in which the object is fixed and lighting conditions vary [40]. Its key idea is to utilize the fact that the amount of light that a surface reflects depends on the orientation of the surface with respect to the light source and the camera.

Computing Normals under Known Lighting Conditions: Besides the original assumptions under which photometric stereo is normally used [40] (e.g. point light sources, uniform albedo, etc.), we now assume that the illumination is known.

Given three point light sources, the surface normal vectors S can be computed by solving the following linear equation based on the two known variables:

$$I^T = L^T S, \tag{1}$$

where $I = [I_1, I_2, I_3]$ is the stacked three stereo images exposed to different illumination, and $L = [L_1, L_2, L_3]$ is the lighting direction for these three images. Note that at least three images under variant lighting conditions are required to solve this equation and to make sure that the surface normals are constrained.

Computing Normals under Unknown Lighting Conditions: Now we consider the case when the lighting conditions are unknown. The matrix of intensity measurements is further denoted as M, which is of size $m \times n$ where m is the number of images and n is the number of pixels in each image. Therefore

$$M = L^T S. \tag{2}$$

For solving this approximation, M is factorized using Singular Value Decomposition (SVD) [38]. Using SVD the following is obtained

$$M = U \Sigma V^T. \tag{3}$$

This decomposition can be used to recover L and S in the form of $L^T = U\sqrt{\Sigma}A$ and $S = A^{-1}\sqrt{\Sigma}V^T$, where A is an 3×3 linear ambiguity matrix. [20] provides the details about how this equation can be solved with four images under different lighting conditions.

3 FACEREVELIO SYSTEM OVERVIEW

FaceRevelio is a liveness detection system designed to defend against various spoofing attacks on face authentication systems.

Figure 1 shows an overview of *FaceRevelio*'s architecture. It begins its operation by dividing the phone screen into four quarters and using each of them as a light source. *Random Light Passcode Generator* module is used to select a random light passcode which is a collection of four orthogonal light patterns displayed in the

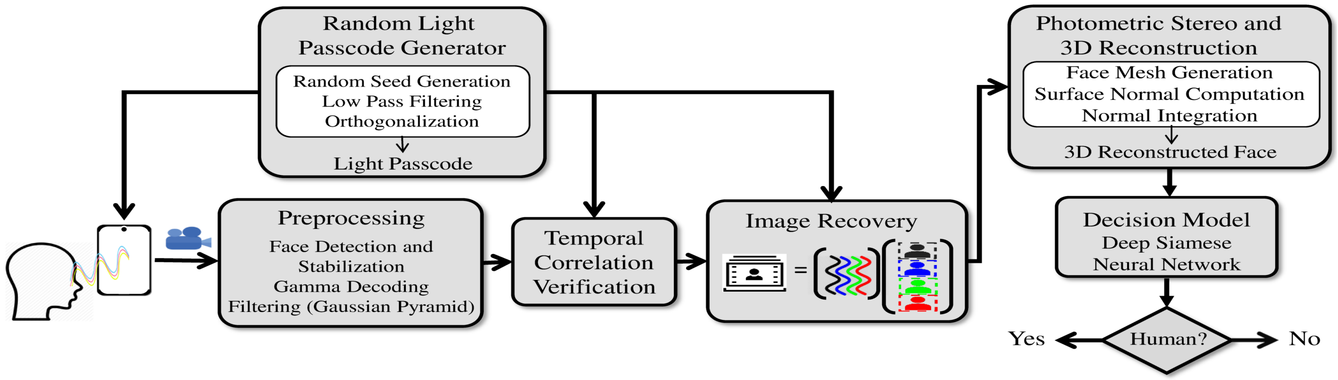

Figure 1: System overview

four quarters of the phone screen. The front camera records a video clip containing the reflection of these light patterns from the user's face. These light patterns are not only used during video recording, but also help reconstruct 3D structure of the face and detect replay attacks. The recorded video then passes through a preprocessing module where first face region is extracted and aligned in each adjacent video frame. This is followed by an inverse gamma calibration operation applied to each frame to ensure linear camera response. Finally, the video is filtered by constructing its Gaussian Pyramid [13], where each frame is smoothed and subsampled to remove noise. After preprocessing, a temporal correlation between the passcode in the video frames and the one generated by the *Random Light Passcode Generator* is checked. If a high correlation is verified, the filtered video frames along with the random light passcode are fed into an *Image Recovery* module. The goal of this module is to recover the four stereo images corresponding to the four light sources, by utilizing the linearity of the camera response. The recovered stereo images are then used to compute face surface normals under unknown lighting conditions using a variant of photometric stereo technique [20]. A generalized human normal map template and its 2D wired mesh connecting the facial landmarks are used to compute these normals accurately. A 3D face is finally reconstructed from the surface normals by using a quadratic normal integration method [34]. Once the 3D structure is reconstructed, it is passed on to a liveness detection decision model. Here, a Siamese neural network [24] is trained to extract depth features from a known sample human face depth map and the reconstructed candidate 3D face. These feature vectors are then compared via L1 distance and a sigmoid activation function to give a similarity score for the two feature vectors. The decision model declares the 3D face as a real human face if this score is above a threshold and detects a spoofing attack otherwise.

4 FACEREVELIO ATTACK MODEL

Attacks to face authentication techniques can be classified into static and dynamic attacks. In a 2D static attack, a still object such as a photograph or mask is used, such that the face recognition algorithms would not be able to differentiate these presented objects from an actual face. Dynamic attacks aim at spoofing systems where some form of user action is required like making an expression or a gesture. In these attacks, a video of the user is replayed performing the requested action. These videos can easily be forged by merging user's public photos with its facial characteristics. Adversaries can also launch a 3D static attack by using 3D models of the face. However, this requires advanced 3D printing capabilities which requires high cost. Similarly, 3D dynamic attacks involving building a 3D model in virtual settings, are impractical as described in [37].

In this paper, our goal is to prevent adversaries from spoofing face authentication systems with 2D static and dynamic attacks. We assume that an attacker has access to high-quality images of the legitimate user's face. We also assume that the adversary can record a video of the user while using *FaceRevelio*. In this case, the recorded video will capture the light patterns' reflections from the user's face. The attacker prepares these videos beforehand and launches an offline attack on our system by displaying them on a laptop screen/monitor. An adversary can possibly conduct an online attack if they have access to high-speed cameras, powerful computers, and a specialized screen with a fast refreshing rate such that it can capture and recognize the random passcode displayed on the screen on each use of the system, forge appropriate face responses depending on the passcode, and present the forged responses to the system. However, because of these difficult requirements for conducting such an attack, we believe that 2D attacks with photos/videos are still the major threat and the main focus of our paper.

5 FACEREVELIO SYSTEM DESIGN

5.1 Light Passcode Generator

To apply photometric stereo, we need to generate four images of the face illuminated under various light sources, from different directions. In order to simulate these light sources using the phone screen, we divide the screen into four quarters where each quarter is assumed a light source. During the video recording, each of these quarters is illuminated alternately in four equal intervals, while the other three quarters are dark. Figure 2 shows how the screen changes with different patterns during the four intervals and an example of the 3D reconstruction of the face using these patterns.

Random Passcode Generator: It could be argued that using these basic light patterns, the system would be prone to replay attacks. Keeping this in mind, we consider the idea of illuminating all the four quarters together for a certain period and changing the

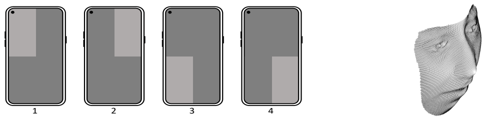

Figure 2: An example of 3D reconstruction using four basic light patterns displayed on four quarters of the screen.

screen lighting randomly at each time instance and each quarter to a random value drawn from a continuous range between −1 and 1. Now, each quarter of the screen is illuminated simultaneously with a random pixel value, simulating four light sources. Based on this, we define a *light passcode* as a collection of four random light patterns displayed in the four quarters. In the rest of the paper, we will use *passcode* as a short-term for *light passcode*.

For the passcode, a random light pattern P_j is generated for a quarter j. During a time interval t_s, P_j is the light pattern represented as a sequence of random numbers, between −1 and 1, of length t_s. The light pattern represents what each pixel of the screen is set to in the quarter j. In order to account for the smartphone screen refreshing rate, we apply an ideal low pass filter with a frequency threshold of $3Hz$ to each of the four light patterns. Although current smartphone screens support a refreshing rate of $60Hz$, there is a delay when the screen is gradually updated from top to bottom. As a result, when the frequency threshold is set to a higher value, the intensity within each quarter may not be consistent. Additionally, setting a higher frequency threshold would result in rapid changes in the screen intensity, making it uncomfortable for users' eyes. These filtered patterns are then normalized such that each pattern is zero-mean.

One problem in illuminating the four quarters together is that the recorded video has a mixture of reflections of the four light patterns from the face. To be able to recover the stereo images from the mixture of reflections, we guarantee independence when combining the light patterns into a passcode. On top of ensuring their independence, we also introduce orthogonality between these four patterns. We apply Gram-Schmidt [18] process to the four light patterns to get their orthogonal basis and use these as patterns. Orthogonality assures a good separation between the impact of the four patterns on the human face and hence helps in the recovery of stereo images. Using induction and the fact that Gram-Schmidt process is linear, we can prove that if each of the original patterns satisfies the frequency threshold of 3Hz, the resulting orthogonal patterns are also within 3Hz. Figure 3 shows an example of a passcode with four patterns and the FFT of these patterns before and after the application of Gram-Schmidt process. We can see that the FFT of the patterns generated after applying Gram-Schmidt to the filtered random sequence only has components below $3Hz$. On a side note, the above process is analogous to code-division multiple access (CDMA) [39] used in radio communications. In our case, the face is analogous to the shared media, the camera is the receiver and our orthogonal patterns are like the codes in CDMA. The stereo images generated by each independent quarter are like the data bit sent by each user. The difference is that in our case, we design and use patterns of continuous values that satisfy a frequency bound requirement.

As a result of the above steps, we obtain four orthogonal zero-mean light patterns, forming a passcode. Each value in the passcode is then multiplied with an amplitude of 60 and finally the passcode is added on top of a constant base pixel intensity value of 128 to be displayed on the screen. Here, note that the Section 5.5 describes how the passcodes are used to defend against replay video attacks.

5.2 Video Preprocessing and Filtering

After generating a random passcode, the corresponding light patterns are displayed on the smartphone screen. Meanwhile, a video of their reflections from a user's face is recorded using the front camera. From the recorded video, first, we locate and extract the face in each frame by identifying the facial landmarks (83 landmarks) using Face++ [2]. We then use these landmarks to align the face position in every adjacent frame to neutralize the impact of slight head movements and hand tremors.

Since our following algorithms focus on how the changes in lighting conditions affect the captured face images, we preprocess the recorded video by converting each frame from the color space to the HSV space [10]. Only the V component will be kept and the other two components are discarded since the V component reflects the brightness of an image. Then, each video frame represented by the V component is further processed using Gaussian pyramid [13] which is a standard technique used in signal processing to filter noise and achieve a smoother output. We used Gaussian pyramid to remove any inherent camera sensor noise. Additionally, pyramids reduce the size of the input video frames by decreasing the spatial sampling density while retaining the important features within the frame, which in turn reduces the system's processing time. We use two levels of pyramid and select the peak of the pyramid in the subsequent steps for video analysis.

5.3 Image Recovery

Recall that in photometric stereo, at least three stereo images with different single light sources are needed for computing the surface normals. However, what we obtained so far is a series of frames, in which the lighting on the face at any given time is a combined effect of all four light patterns on the screen. Therefore, we need to recover these stereo images for each quarter from the preprocessed video frames, which is different from the traditional way of directly collecting stereo images used for photometric stereo.

Based on the theory that the intensities of incoherent lights add linearly [22], we propose to recover the stereo images by directly solving the equation, $G = WX$, where G is a $f \times n$ matrix representing the light intensity values received on each pixel in the recorded video frames, where f is the number of frames and n is the number of pixels in one frame. W represents the $f \times 4$ light patterns $[P_1; P_2; P_3; P_4]$ used while recording the video. X $(= [I_1; I_2; I_3; I_4])$ is a $4 \times n$ matrix representing the four stereo images that we aim to recover. This equation utilizes the fact that under a combined lighting condition, the light intensity received on a certain pixel is a weighted sum of four light intensities with a single light from each quarter.

However, we cannot directly use the above equation unless under the assumption that camera sensors can accurately capture light intensities and reflect the actual values. Problems, e.g. inaccurate image recovery, will arise if we ignore the possible effects of camera

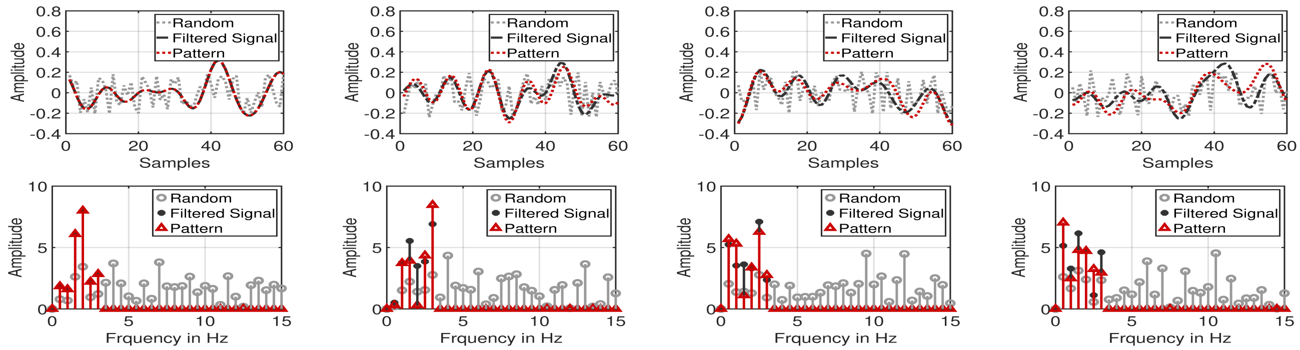

Figure 3: An example of a random passcode. The top row shows the four random patterns in the passcode before and after low-pass filtering and the final patterns after applying Gram-Schmidt process to the filtered pattern. The bottom row shows the FFT of these patterns before and after applying Gram-Schmidt process. The frequency bound still holds after applying Gram-Schmidt process.

parameters and sensitivity. Recently, smartphone camera APIs[1] started supporting manual camera mode which gives the user full control of the exposure parameters, i.e. aperture, shutter speed (exposure time) and sensitivity (ISO). In automatic mode, the camera continuously adjusts its ISO to compensate for lighting changes in the scene. In order to have a smoother camera response with changing light intensity, we use the camera in manual mode where its ISO is set to a fixed value.

Although the camera response curve is smooth after fixing the ISO, we still need to linearize the relationship between the image captured and the light from the screen to be able to use the equation for solving G. For this purpose, we dig deep into the mechanics of the camera sensor and image processing involved in generating the final output images. Cameras typically apply a series of operations on the raw camera sensor data to give us the final output images. These include linearization of sensor data, white balancing, demosaicing [6] and gamma calibration [7]. Gamma calibration is where non-linearity arises between the captured pixel values and the light intensity from the scene. In order to make use of linear relationship between these two, we apply an inverse of the gamma calibration, to the recorded video frames obtained from the camera. As a result, the resulting pixel values in the range between black and saturation level have a linear relationship with the actual light present in the scene. This relationship can be formulated as the linear model, $y = kx + b$, where b is the y-intercept introduced to account for the non-zero black level of the camera sensor. This inverse calibration is applied to each frame in the video preprocessing before face extraction. Now by generalizing the linear model to every frame, containing multiple pixels, we get

$$K = kG + B, \tag{4}$$

where K is the video frames that the camera actually captured for the duration of the passcode. By substituting the definition of G into Equation 4, we get

$$K = kWX + B. \tag{5}$$

[1] Android supports manual camera mode starting from Android Lollipop 5.1

Finally, we use the least square method to solve

$$WX = \frac{1}{k}(K - B) \tag{6}$$

which can be written as

$$X = (W^T W)^{-1} W^T (\frac{1}{k}(K - B)) \tag{7}$$

Here, notice that B is a constant matrix and since each of the four patterns in the passcode W are zero-mean, the term $W^T B$ will be eliminated. Hence Equation 7 becomes:

$$X = (W^T W)^{-1} W^T (\frac{1}{k}K) \tag{8}$$

Note that this solution X will have an uncertainty of a scale factor. For any $\alpha > 0$, let $X' = \alpha X$, $k' = \frac{1}{\alpha}k$. X', k' will also minimize the above function.

However, this will not have an impact on the reconstructed surface normals. Recall, that surface normals are computed by taking SVD of the stereo images. So, when X and X' are both factorized using SVD, the decompositions are

$$X = U\Sigma V^T, \tag{9}$$

$$X' = U(\alpha\Sigma)V^T. \tag{10}$$

The surface normal V^T will stay the same in these two cases. From the above observation, we can set $k = 1$ without any impact on the surface normals. Now, we can solve for X' by

$$X' = (W^T W)^{-1} W^T K \tag{11}$$

So far, we assumed that the only light present in the scene is due to the passcode displayed on the screen. However, we still need to consider the ambient light present in the scene as well as the base intensity value of the screen on top of which the passcode is added. To account for these other light sources, Equation 5 now becomes

$$K = kWX + B + C \tag{12}$$

where C is the constant light present in the scene. Again, since C is a constant, because of the orthogonal and zero-mean nature of our passcode, W, $W^T C$ will become 0. As a result, Equation 11 will give a solution for X even when ambient light is present.

Due to the inherent delay in the camera hardware, the recorded video may have some extra frames and the timestamps for each

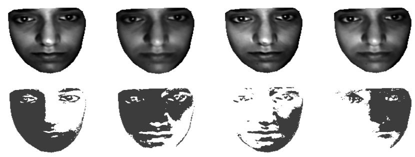

Figure 4: The recovered stereo images corresponding to the four patterns in the passcode. The bottom row shows a binary representation to emphasize the differences in these stereo images.

video frame captured and the four patterns displayed on the screen at that point may differ. To ensure that we obtain a correct and fine alignment between these two, we first compute the average brightness of each frame and then apply a low pass filter on the average brightness across frames. The peaks and valleys in the average brightness are matched with those of the passcode and finally, DTW [11] is used to align the two series correctly. Once aligned, the result is the video frames which exactly represent the reflection of the passcode from the face. These video frames are then given as input to Equation 11 to recover the four stereo images as X. We define the average brightness of these video frames as the recorded passcode for later sections.

An example of the recovered four stereo images corresponding to every single light i.e. four patterns displayed in each quarter is shown in Figure 4. The top 4 images are the recovered stereo images. The bottom images are the binary representation of these stereo images such that in each image, a pixel value is 1 if the pixel in the corresponding stereo image is larger than the mean value of the same pixel in the other three stereo images. This binary representation is just to visually emphasize how different these stereo images are and how they represent the face illuminated from lighting in four different directions.

5.4 Photometric Stereo and 3D Reconstruction

The stereo images recovered from the least squared method approximate the facial images taken with four different point lights. Now, we can use these stereo images to compute the surface normals of the face as described in Section 2.

However, as mentioned earlier, these surface normals have an ambiguity of matrix A. We design an algorithm illustrated in Algorithm 1 to compute the normals without this uncertainty. We use a generalized template, N_t, for the surface normals of a human face and use this to solve for A. This template can be the surface normals of any human face recovered without any ambiguity like surface normals computed when the lighting is known. Note that obtaining this template is a one-time effort and the same normal template is used for all users. Along with the normal map, we also have a $2D$ wired triangulated mesh, M_t, connecting the facial landmarks (vertices), for this template. Now, when computing the normals of a user subject, we use the facial landmarks detected from an RGB image of the face to build a triangulated mesh of the face, M, using M_t as a reference for connecting the vertices and triangles. A representation of this mesh can be seen in Figure 5 (left). An affine transformation from the template mesh, M_t to M is then found independently for each corresponding pair of triangles in the two

ALGORITHM 1: Surface Normal Computation

Data: normal map template N_t, template mesh M_t, stacked four stereo images I and face RGB image R
Result: surface normals S

$V \leftarrow getFaceLandmarks(R)$
$M \leftarrow buildMesh(V, M_t)$
$\hat{S}, \hat{L^T} \leftarrow SVD(I)$
$N'_t \leftarrow transform(N_t, M_t, M)$
Solve $N'_t = A\hat{S}$ for A
$S' \leftarrow A\hat{S}$
$M_s \leftarrow symmetrizeMesh(M)$
$S \leftarrow transform(S, M, M_s)$
$S \leftarrow adjustNormalValues(S)$

function TRANSFORM(Z, T_1, T_2)
  for each pair of triangles $< t_1, t_2 > \in T_1, T_2$ **do**
    $a \leftarrow affineTransformation(t_1, t_2)$
    $Z_{out} \leftarrow warp(Z(t_1), a)$
    $Z(t_2) \leftarrow Z_{out}$
end function

meshes and applied to the matching piece in N_t. As a result, the transformed normal map template, N'_t now fits the face structure of the user. This transformed template can finally be used to find the unknown A, by solving $N'_t = A\hat{S}$ where $\hat{S}$ are the approximate normals recovered from SVD, and obtain the surface normals, S'. The last step in normal map computation is to make the normal map symmetric. This is needed to reduce noise in the recovered stereo images and hence the surface normals. We first find the center axis of the $2D$ face mesh using landmarks on the face contour, nose tip and mouth. Once the center is found, each pairing landmarks like eyes, eyebrow corner etc. are adjusted such that they have equal distance to the center to get a symmetric mesh. After symmetrizing the mesh, we fit S into this symmetrized mesh. Now, we can easily apply inverse symmetry to the x component of S and symmetrize the values in y and z components of S. Note that by introducing symmetry, we might loose some tiny details of the facial features as all human faces are not symmetrical. However, since our goal is to distinguish the human face from their spoofing counterpart and not another human, the information retained in the surface normals is more than sufficient. Figure 5 (right) shows an example of the x, y and z components of a normal map generated from our algorithm.

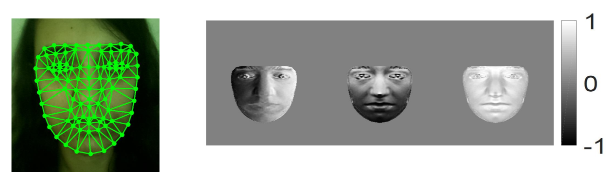

Figure 5: Normal map calculation (left) shows $2D$ triangulated face mesh generated by using facial landmarks. (right) shows the X, Y and Z components of the normal map generated from Algorithm 1.

After we have successfully recovered the surface normals, we can reconstruct the 3D surface of the face from them. For 3D reconstruction, we follow the quadratic normal integration approach described in [34]. The results of 3D reconstruction are shown in Figure 6. Side and top view are shown for each reconstructed model.

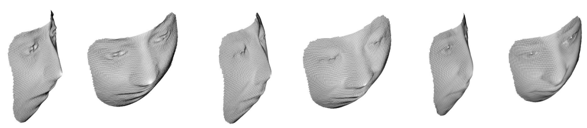

Figure 6: Examples of 3D reconstruction from human faces. Side and top views are shown.

5.5 Liveness Detection

FaceRevelio aims to provide security against two broad categories of spoofing attacks: 2D printed photograph attack and video replay attack.

2D Printed Photograph Attack: To defend against the 2D printed photograph attacks, we need to determine whether the reconstructed 3D face belongs to a real/live person or a printed photograph. Figure 7 shows examples of 3D reconstruction from a printed photograph using the approach described in the previous section. It is interesting to note here that the same general human face normal map template is used for computing the surface normals of a photograph. As a result, the overall structure of the reconstructed model looks similar to a human face. However, even when using this human normal map template, the freedom provided by solving for A is only up to 9 dimensions. Therefore, despite having a similar structure, the reconstruction from the 2D photograph lacks depth details in facial features, e.g. nose, mouth and eyes, as is clear in the examples in Figure 7.

Based on these observations, we employ a deep neural network to extract facial depth features from the 3D reconstruction and classify it as a human face or a spoofing attempt. We train a Siamese neural network adapted from [24] for this purpose. The Siamese network consists of two parallel neural networks whose architecture is the same, however, their inputs are different. One of these networks takes in a known depth map of a human face while the other is given the candidate depth map obtained after the 3D reconstruction. Therefore, the input to the Siamese network is a *pair of depth maps*. Both the neural networks in the Siamese network output a feature vector for their inputs. These feature vectors are then compared using L1 distance and a sigmoid activation function. The final output of the Siamese network is the probability of the candidate depth map being that of a real human face. If this output value is above a predefined threshold, τ_s, the system detects a real face. Otherwise,

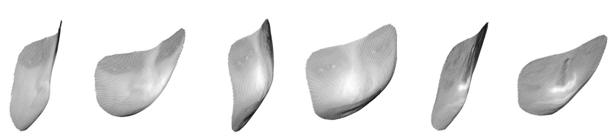

Figure 7: Examples of 3D reconstruction from 2D printed photographs.

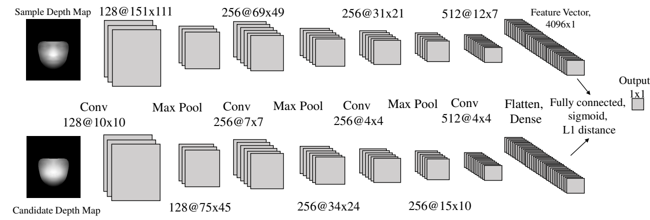

Figure 8: Architecture of the Siamese neural network. One of the twin neural networks takes a known human depth map as input while the other is passed the candidate 3D reconstruction.

a spoofing attempt is identified. Figure 8 shows the architecture of the Siamese network.

To elaborate the training process for our Siamese network, suppose we have N depth maps collected from human subjects and N depth maps from their photos/videos. From these depth maps, we obtain $N(N-1)/2$ pairs of positive samples where both the depth maps in the pair are from human subjects. For the negative samples, we have N^2 pairs where one depth map is of a human subject while the other is from a photo/video. Since the total number of negative samples is larger than the positive samples, we randomly select $N(N-1)/2$ samples from the negative pairs. These positive and negative samples are then used as input to train the Siamese network. Every time a subject tries to authenticate using *FaceRevelio*, the reconstructed $3D$ model along with a sample human depth map is fed as the input pair to the Siamese network. Here, note that the sample human depth map can be any depth map obtained from our reconstruction algorithm from a human subject in the training set and does not require registration by the test subject.

Since Siamese network uses the concept of one-shot learning [19] and takes pairs as input for training, the amount of data required for training is much smaller than traditional convolutional neural networks. Here, one may argue that why not train the model with the raw images/videos captured by the front camera, for the duration of passcode, instead of the depth map to decide if the subject is a human or not? Although intuitive, training such classifiers would require huge amounts of data; datasets for different environment settings, different light passcodes, different distances between the face and phone and various face orientations. In contrast, our image recovery module and approach for reconstructing the 3D surface of the face account for the ambient lighting and different orientations of the face before generating the depth map. Training a model using these depth maps ensures that input to our network is not impacted by the various ambient environment conditions; hence, much less data is required for training. Furthermore, models with video input are more complex with larger number of trainable parameters, resulting in higher storage and computation costs.

Video Replay Attacks: *FaceRevelio* has a two-fold approach for defending against video replay attacks. The first line of defense is to utilize the randomness of the passcode. When a human subject tries to authenticate via *FaceRevelio*, the passcode displayed on the screen is reflected from the face and captured by the camera. As a result, the average brightness of the video frames across time has a high correlation with the light incident upon the face i.e. the sum

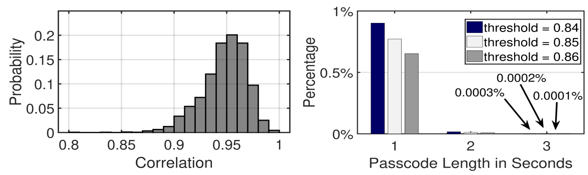

Figure 9: Video Replay Attack: (left) shows the distribution of correlation between recorded passcodes from human face and the original passcode. (right) shows the percentage of passcodes which have a correlation with another random passcode higher than a threshold for different thresholds.

of the four patterns in the passcode displayed on the screen. Figure 9 (left) shows a distribution of the correlation between recorded passcodes and the original passcode for experiments conducted with humans. The correlation between the two passcodes is higher than 0.85 for more than 99.9% of the cases. An adversary may try to spoof our system by recording a video of a genuine user while using *FaceRevelio* and replay this video on a laptop screen or monitor in front of the phone later. In this case, the video frames captured by the camera will have the reflections of the passcode on the phone screen as well as the passcode present in the replay video. Since *FaceRevelio* chooses a totally random passcode each time as described in 5.1, the probability that the passcode displayed on the screen and the passcode in the video has a high correlation is extremely low. To give an idea, for a passcode duration of $3s$, if we compare 300 million pairs of random passcodes, only 0.0003% of the pairs will have a correlation greater than 0.84. Figure 9 (right) shows the percentage of passcode with a correlation higher than threshold values 0.84, 0.85 and 0.86 for passcode lengths of 1, 2 and $3s$. Hence, just by computing and setting a threshold on the correlation between the recorded passcode and the sum of passcode from the screen, the chances of detecting a replay attack are very high.

For the rare cases when the correlation is higher than the predefined threshold, our second line of defense comes into play. Similar to 2D photograph attack, video replay attacks can also be detected using the reconstructed 3D model. The reconstruction from the replayed video suffers from two main problems. First, it is hard for the adversary to accurately synchronize playing the attack video with the start of the passcode display on the smartphones. Second, even if the correlation passes the threshold, there will be some differences in the replayed passcode and *FaceRevelio*'s passcode. Because of this, the DTW matching will not match the recorded video frames with the displayed passcode very well. Hence, the four stereo images, X, obtained by solving equation 11 will not be representative of the subject's face being illuminated from four different lighting directions. As a result, the surface normals and 3D reconstruction from these wrong stereo images do not capture the 3D features of the face and is sufficient to identify a spoofing attempt.

6 EVALUATION

We describe the implementation and evaluation of our system in this section. We first describe the experiment settings and the data collection details and then the performance of our system in different settings.

6.1 Implementation and Data Collection

We implemented a prototype for *FaceRevelio* on Samsung S10 which runs Android 9, with 10 MP front camera that supports Camera2 API. The videos collected for our authentication system have a resolution of 1280x960 and a frame rate of 30fps. For each experiment setting, we display the passcode patterns on the smartphone screen and record a video of the reflections from the user's face via the front camera. We use Face++ [2] for landmark detection and OpenCV in the image recovery and reconstruction modules of our system. Python libraries for TensorFlow [9] and Keras were used to train the neural network for liveness detection while TensorFlow Lite was used for inference on Android.

We evaluated *FaceRevelio* with 30 volunteers using our system for liveness detection. The volunteers included 19 males and 11 females with ages ranging from 18 to 60. These volunteers belonged to different ethnic backgrounds including Americans, Asians, Europeans and Africans. During the experiments, the volunteers were asked to hold the phone in front of their faces and press a button on the screen to start the liveness detection process. Once the button was clicked, the front camera started recording a video for the duration of the passcode. During all experiments, we collected a high-quality image of the user to test the performance of our system against photo attacks. For the video replay attack, we used the videos collected from real users and replayed them to the system.

We collected a total of 4900 videos from the 30 volunteers over a duration of 3 weeks. We evaluated the performance of our system in natural daylight as well as in completely dark environment (0 lux). For the daylight setting, all experiments were conducted during daytime however the light intensity varied (between 200 to 5000 lux) based on the weather conditions on the day and time of the experiment. Each volunteer performed 10 trials of liveness detection using our system for each of the two light settings. A random passcode of $1s$ duration was added on top of a gray background (grayscale intensity value of 128) for these trials. We also tested *FaceRevelio* with passcode durations of 2 and $3s$ in the two light settings. We also evaluated the impact of indoor lighting ($\sim$ 250 lux), the distance between the face and the smartphone screen and the orientation of the face, on the performance of our system. For these scenarios, we collected data from 10 volunteers with a passcode duration of $1s$. These volunteers used the system 30 times for each scenario. In addition, we also explored whether using a background image affects *FaceRevelio*'s performance.

We used the Siamese neural network described in section 5.5 to test each user. We employed a leave-one-out method for each test user where we used the depth maps generated from the data collected from the remaining 29 users for training. From these 29 users' data, we used 80% of the data as the training set while the remaining 20% was used for validation. Hence, the test user's data remained unseen by the network during the training process. At inference time, the depth map from the test subject along with a sample human depth map, randomly selected from the human depth maps collected from the other 29 subjects, was given as the input pair to the Siamese network. The predefined threshold, τ_s, for classifying the test subject as a real human or not, was set to 0.7 for the evaluation.

6.2 Performance Results

For evaluating *FaceRevelio* system performance, we answer the following questions:

(1) What is the overall performance of *FaceRevelio*?

To determine the overall performance of our system, we evaluated our system's ability to defend against 2D printed photographs and video replay attacks. We report the accuracy of our system as the true and false accept rate for the two light settings. We also determine the equal error rate (EER) for the attacks.

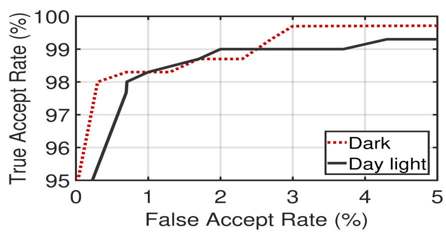

Figure 10: ROC curve for detecting photo attack in dark and daylight setting with a passcode of 1*s*. The detection rate is 99.7 and 99.3% when true accept rate is 98% and 97.7% for the two settings respectively.

First, we describe our system's performance against printed photograph attack. Figure 10 shows the ROC curve for *FaceRevelio*'s defense against photo attack in the dark and daylight setting with a passcode duration of 1*s*. For dark setting, with a true accept rate of 98%, the false accept rate is only 0.33%. This means that a photo attack is detected with an accuracy of 99.7% when the real user is rejected in 2% of the trials. The EER for the dark setting is 1.4%. In daylight, the photo attack is detected with an accuracy of 99.3% when the true accept rate is 97.7%. The EER in this case is also 1.4%. *FaceRevelio* performs better in dark setting because the impact of our light passcode is stronger when the ambient lighting is weaker. Hence, the signal-to-noise ratio in the recorded reflections from the face is higher, resulting in a better 3D reconstruction.

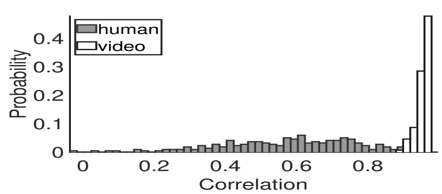

Figure 11: Distribution of the correlation between the passcode on the phone and the camera response from real human and video attack combined for dark and daylight setting.

We also evaluated our system against video replay attacks by using videos collected from the volunteers during the experiments. Each video was played on a Lenovo Thinkpad laptop, with a screen resolution of 1920 x 1080, in front of a Samsung S10 with *FaceRevelio* installed. Our system detected these video replay attacks with an EER of 0% in dark and 0.3% in daylight settings. Figure 11 shows a histogram of the correlation between the passcode displayed on the phone and the camera response for all experiments with 1*s* long passcode. The correlation for all the attack videos is less than 0.9. In contrast, 99.8% of the videos from real human users have a correlation higher than 0.9.

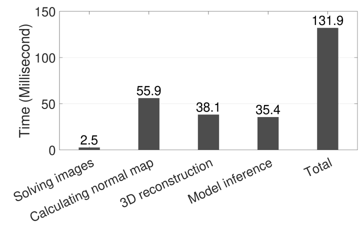

Figure 12: Processing time of the different modules of the system for a passcode of 1*s* duration.

Another performance metric is the total time it takes to detect liveness with *FaceRevelio*. Figure 12 shows the processing time of the different modules of our system. On top of the signal duration of the passcode, the liveness detection process only takes 0.13*s* in total. The stereo images recovery only takes 3.6*ms*. The most expensive computation step is the normal map computation, taking 56*ms*, since it involves two 2D warping transformations. 3D reconstruction and feature extraction and comparison via the Siamese network take 38.1 and 35.4 ms respectively.

(2) What is the effect of the duration of the light passcode?

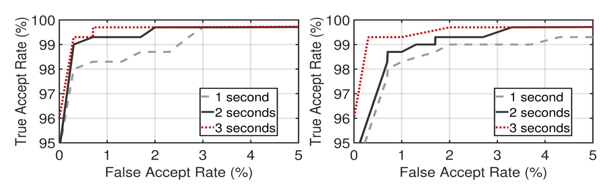

Figure 13: ROC curve for passcode durations of 1, 2 and 3 seconds in dark (left) and daylight (right) settings.

To answer this question, we tested the performance using passcodes of time durations 1, 2 and 3*s*. Figure 13 shows the ROC curve for photo attack with different passcode duration in dark (left) and daylight (right) settings. In dark, the attacks are detected with an accuracy of 99.7% for passcodes of length 1, 2 and 3 seconds each. These accuracies are achieved when the true accept rate is 98%, 99% and 99.3% for the three time durations respectively. The EER is 1.44% for 1*s* and 0.7% for 2 and 3 each. For daylight, the detection accuracy is 99.3% for 1*s* and 2*s*. For 3*s*, the photograph attack is detected with an accuracy of 99.7%. These accuracies are achieved when the true accept rate is 97.7%, 98.3% and 99.3% for 1, 2 and 3*s* respectively. We observe that the performance of *FaceRevelio* improves as we increase the duration of the passcode. Although the true accept rate deteriorates when a passcode of $1s$ is used, achieving a higher attack detection accuracy within a short duration is the priority of an effective liveness detection system.

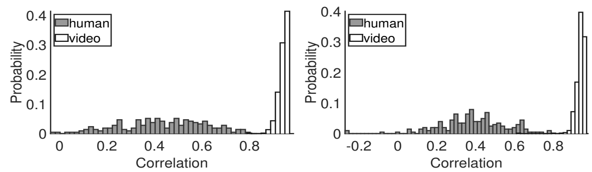

Figure 14: Distribution of the correlation between passcode on the phone and the camera response from real human and video attack for $2s$ (left) and $3s$ (right) long passcodes.

We also evaluated the effect of passcode duration on detecting video attacks. Figure 14 shows the correlation distribution for human and video attack combined for passcode duration of 2 (Figure 14 left) and 3 (Figure 14 right) seconds in the two light settings. For $2s$, all the video attacks have a correlation less than 0.84 while 99.8% of the human data have a correlation higher than 0.86. In case of $3s$, 99.8% of the real human experiments have a correlation higher than 0.8. In comparison, all attack videos have correlation of less than 0.8.

We also determine the effect of the passcode duration on the processing time in the authentication phase. The duration of the passcode only affects the time taken to determine the least squared solution for recovering the four stereo images as that depends on the number of frames in the recorded video. The computation time for the other components of the system stays consistent across different passcode duration. The total processing time remains below $0.15s$ for all three passcode durations.

(3) How well does *FaceRevelio* perform in indoor lighting?

To evaluate the effect of indoor lighting, we conducted experiments with 10 volunteers in a room with multiple lights on. The goal was to determine if this extra light had any impact on the efficacy of our light passcode. In these experiments, we used $1s$ long passcodes. For a true accept rate of 98%, *FaceRevelio*'s accuracy against 2D attacks is 99.7%. It achieves an EER of 1.4% which is comparable to the dark setting. Hence, we conclude that *FaceRevelio* performs well even when artificial light is present in the scene.

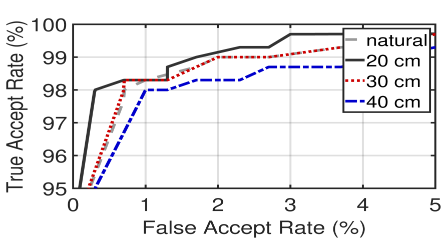

Figure 15: ROC curve for different face to smartphone screen distances with a passcode duration of $1s$.

(4) What is the impact of the smartphone's distance from the face on *FaceRevelio*'s performance?

We evaluated the effect of distance between the face and the smartphone screen by conducting experiments with 10 volunteers. First, we asked the volunteers to hold the smartphone naturally in front of their face such that their face is within the camera view and use our system. We measured the distance in this scenario for each volunteer and observed that the average distance between the face and the screen during these experiments was $27cm$. Later, we guided the volunteers to use *FaceRevelio* while holding the smartphone at various distances from their face, more specifically, at $20cm$, $30cm$ and $40cm$.

Figure 15 shows the ROC curve for *FaceRevelio* performance against 2D attack for various distances between the face and the smartphone screen. For both the natural distance and $30cm$, *FaceRevelio* detects the 2D attack with an accuracy of 99.3% when the true accept rate is 98%. The detection accuracy is 99.7% with a true accept rate of 98% when the distance between the face and the screen is $20cm$. We also observe that *FaceRevelio*'s detection accuracy remains 99.3% when the smartphone's distance from the face is increased to $40cm$. This shows that *FaceRevelio* can defend spoofing attempts even when the distance is relatively large. The true accept rate deteriorates slightly to 96.7% in this case. The lower true accept rate, however, does not impact the usability (since users usually hold the phone at a closer distance) and more importantly, the security (since detection accuracy is still high) of *FaceRevelio*.

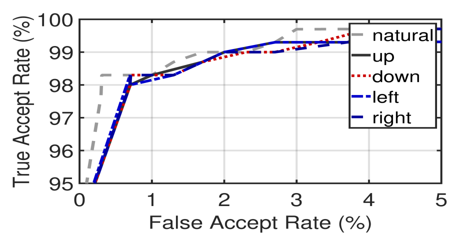

Figure 16: ROC curve for different face orientations with a passcode duration of $1s$.

(5) Does the orientation of the face affect *FaceRevelio*'s performance?

For evaluating the impact of face orientation on *FaceRevelio*'s performance, we first requested the volunteers to hold the phone naturally while keeping their face vertically aligned to the smartphone screen and use our system. We then instructed them to rotate their head *up*, *down*, $left$ and *right* and perform trials for each face orientation. Figure 16 shows the performance of our system for the various face orientations. For the natural case, *FaceRevelio*'s detection accuracy is 99.7% when the true accept rate is 98.3%. *FaceRevelio* can detect the 2D attacks with an accuracy of 99.3% with a true accept rate of 98%, 98.3%, 98% and 98.3% for the *up*, *down*, $left$ and *right* face orientations respectively. The EER for the natural face orientation as well as the four rotated face poses is 1.44%. This shows that *FaceRevelio* can defend against spoofing attempts for different orientations of the face attributing to the facial landmark aware mesh warping used in the surface normal computation described in section 5.4.

(6) What is the effect of displaying the signal on a background image?

So far, we used gray image as a base for the light passcode displayed on the screen to evaluate our system. Here we want to determine how the system performance change if we used an RGB

Algorithm	Attack Resistance	Special Hardware?	User Interaction Required?	Limitation	Accuracy
FaceID [1]	2D & 3D	TrueDepth	No	3D head mask attack	$> 99.9\%$
Samsung FR [4]	None	No	No	Photo Attack	-
EchoFace [14]	2D photo	No	No	Audible sound	96%
FaceCloseup [30]	2D photo/video	No	Requires moving the phone	Slow response	99.48%
EchoPrint [42]	2D photo/video	No	No	Audible sound, low accuracy in low illumination	93.75%
Face Flashing [37]	2D photo/video	No	Requires expression	Slow response	98.8%
FaceHeart [16]	2D photo/video	No	Place fingertip on back camera	Low accuracy in low illumination	EER 5.98%
FaceLive [29]	2D photo/video	No	Requires moving the phone	Slow, low accuracy in low illumination	EER 4.7%
Patel et al. [33]	2D photo/video	No	No	Device dependent, low accuracy in low illumination	96%
Chen et al. [15]	2D photo/video	No	Requires moving the phone	Slow response	97%

Table 1: Summary of existing face liveness detection methods

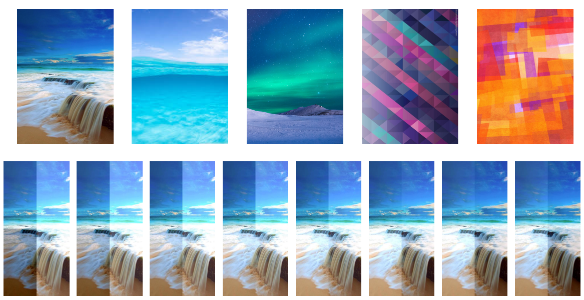

Figure 17: Top row shows images chosen as background for the light passcode. Bottom row shows what the passcode looks like with an image as background

image for the passcode instead of the gray background. For this purpose, we selected a total of 5 background images (shown in Figure 17(top)). Figure 17 also shows an example of what the passcode frames look like with an image background across time. We performed experiments with 10 users where each user performed 10 trials in daylight setting using the 5 background images. Our system achieves an EER of 1.15% against the spoofing attacks. A photo attack is detected with an accuracy of 99.4% when the true accept rate for humans is 97%. These results show that *FaceRevelio*'s process can be made more user friendly by using images of the user's choice as a base for the passcode.

(7) What is the power consumption of *FaceRevelio*?
We additionally investigated the power consumption of *FaceRevelio* by performing several trials of our system and recorded the battery consumption. During these measurements, the brightness level of the screen was set to maximum level by *FaceRevelio* during operation. A single use of our system consumes $1.08mAh$ on average. Assuming that users typically unlock their smartphones about 100 times a day [5] and the average battery size of modern flagship smartphones is $3500mAh$ [3], *FaceRevelio* will consume an average of only 3.4% of the total battery per day.

(8) Where does *FaceRevelio* stand compared to existing face liveness detection methods?

Table 1 gives an overview of the existing methods for face liveness detection on smartphones. It shows the type of attacks these methods can defend against and if they require any extra hardware or user interaction for doing so. Among the commercial solutions, Samsung's face recognition is vulnerable to simple 2D photo attacks and needs to be combined with other authentication methods for security [4]. Apple's *FaceID* [1] is the most secure method against 2D and 3D spoofing attacks, owing to the *TrueDepth* camera [1] system. Since *FaceID* is an authentication system, it generates 3D reconstruction of the face which is capable of capturing the subtle differences in the facial features of different humans. However, among liveness detection methods that do not rely on any extra specialized hardware [16, 29, 30, 37], *FaceRevelio* achieves the highest accuracy in detecting 2D photo and video attacks with the fastest response time of $1s$. Tang et al. [37] use a challenge-response protocol to achieve a high detection accuracy, however, their approach relies on the user to make facial expressions as instructed and takes $6s$ or more (depending on the number of video frames collected) to perform well. In contrast, *FaceRevelio* detects the spoofing attempts in $1s$, without requiring any user interaction, increasing its overall usability. Another important comparison metric is the performance variation in different lighting conditions. For methods like [16, 33, 42], the performance mentioned in table 1 is achieved under controlled lighting conditions and deteriorates in dark environments. EchoFace [14] achieves a good accuracy by using an acoustic sensor based approach however their sound frequency is within human audible range, (owing to smartphones' speaker limitation [28]) making it less user friendly.

7 RELATED WORK

Several software-based face liveness detection techniques have been proposed in the literature. These depend on features and information extracted from face images captured without additional hardware. Texture-based methods detect the difference in texture between real face and photographs/screens. In [36], local binary patterns were used to detect the difference in local information of a real face and a 2D image using binary classification. Another technique, [23], measures the diffusion speed of the environmental light which helps distinguish a real face. [16] operates by comparing photoplethysmograms independently extracted from the face and fingertip videos captured by front and back cameras. Similarly [31] uses a combination of rPPG and texture features for spoof detection. These works do not perform well in poor lighting conditions and are affected by the phone camera limitations. Some works [33, 41], make use of the degraded image quality of attack photos or videos. However, with modern cameras and editing softwares, an adversary can easily obtain high quality images and videos to conduct an attack. In contrast to these approaches, *FaceRevelio* works in different lighting conditions and is not dependent on the quality of the videos captured.

Other techniques use the involuntary human actions such as eye blinking [21] or lips movement [26] to detect spoofing, but these techniques fail against video replay attacks. Challenge-response protocols require the user to respond to a random challenge, such as blinking, face expression, head gesture, etc [32]. These systems are limited by their unconstrained response time and are still prone to replay attacks. Another work, [37], used a time constrained challenge-response technique that shows different colors on the screen and detects the difference in the time of reflection between a real face and an attack. This work differs from *FaceRevelio* as they utilize the random challenge on the screen to perform a timing verification whereas we use the screen lighting to reconstruct the 3D surface of the face. Also, [37] requires the user to make a face expression to defend against static attacks. Some works like [15, 29] require the user to move the phone in front of their face and analyze the consistency between the motion sensors' data and the recorded video to detect liveness. These approaches require some form of user interaction unlike our system which operates independently of the user.

Some hardware-based techniques require extra hardware or different sensors to detect more features of the human face structure. FaceID was introduced by Apple in the iPhone X to provide secure 3D face authentication using a depth sensor [1]. However, the extra hardware consumes screen space and requires additional cost. [42] developed an authentication system that uses the microphone with the front camera to capture the 3D features of the face. However, this technique does not work well in poor lighting and depends on deep learning which requires large training datasets. Similarly, [14] uses acoustic sensors to detect the 3D facial structure. Both these techniques play audible sound for detection, which makes their system less user friendly. Some other techniques use thermal camera [17], 3D-camera or multiple 2D-cameras [27]. Again, these techniques suffer from the setup cost for these extra devices.

8 DISCUSSION

FaceRevelio depends on light emitted from the screen, therefore it is sensitive to rapid changes in the ambient lighting like when a user is in a moving car. The accuracy of our system would be affected in such scenarios. This requires investigating other camera features to recognize the small light changes produced by our passcode in the presence of strong, changing ambient light.

Recently, some advanced machine learning based attacks [12, 35] have been successful in spoofing state-of-the-art face recognition systems. However, *FaceRevelio* can defend against these attacks because the random light passcode changes with every use of our system and does not have any relation to the passcodes used previously. Hence, learning a machine learning model to guess the password on the fly and replaying it to the system is not possible. Here, we want to admit that as *FaceRevelio* performs liveness detection by exploiting the differences in the 3D layout of the human face and 2D photos/videos, it is limited in defending against non-2D object with curvature features bearing similarity to the human face. Similarly, *FaceRevelio* may be spoofed by a sophisticated 3D printed mask of the subject mimicking the skin reflection properties and the depth features of the human face. However, these attacks are costly and difficult to execute given the nature of the 3D printing materials commonly available. Keeping this in mind, *FaceRevelio* focused on defending and raising the bar against the commonly existing 2D spoofing attacks. In future, we plan on investigating how our reconstruction algorithm and Siamese network can be adapted to defend against 3D attacks as well.

In our system, we divided the phone screen into four quarters for displaying four random patterns in the passcode. These passcodes helped us achieve a good accuracy in detecting replay attacks. However, we can further push the randomness involved in our passcodes by dividing the screen into smaller regions or using combination of different shapes to display the light patterns. We also plan to increase the system usability by using more sophisticated light patterns, such as a picture of blinking stars or animated waterfall.

FaceRevelio provides a promising solid idea for secure liveness detection without any extra hardware. Our technique can be integrated with existing 2D face recognition technologies on smartphones. Detecting the 3D surface of the face through our system before face recognition would help them in identifying spoofing attacks at an early stage. This will improve the overall accuracy of these state-of-the-art technologies. Apart from this, since our system reconstructs the 3D surface of the face, it has the potential to be used for 3D face authentication. To distinguish the faces of different human beings, our reconstruction algorithm will need be modified to retain the tiny details in the facial features during the mesh warping step of the surface normal computation. We leave this to a future work of our system.

9 CONCLUSION

This paper proposes a secure liveness detection system, *FaceRevelio*, that uses a single smartphone camera with no extra hardware. *FaceRevelio* uses the smartphone screen to illuminate the human face from various directions via a random light passcode. The reflections of these light patterns from the face are recorded to construct the 3D surface of the face. This is used to detect if the authentication subject is a human or not. *FaceRevelio* achieves a mean EER 1.4% and 0.15% against photo and video replaying attacks, respectively.

10 ACKNOWLEDGMENTS

We sincerely thank our shepherd and the anonymous reviewers for their insightful comments and valuable suggestions.

REFERENCES

[1] [n. d.]. About Face ID advanced technology. https://support.apple.com/en-us/HT208108
[2] [n. d.]. Face++. https://www.faceplusplus.com/landmarks/

[3] [n. d.]. Fact check: Is smartphone battery capacity growing or staying the same? https://www.androidauthority.com/smartphone-battery-capacity-887305/

[4] [n. d.]. How does Face recognition work on Galaxy Note10, Galaxy Note10+, and Galaxy Fold? https://www.samsung.com/global/galaxy/what-is/face-recognition/

[5] [n. d.]. How many times do you unlock your phone? https://www.techtimes.com/articles/151633/20160420/how-many-times-do-you-unlock/-your-iphone-per-day-heres-the-answer-from-apple.htm

[6] [n. d.]. Processing Raw Images in MATLAB. https://rcsumner.net/raw_guide/RAWguide.pdf

[7] [n. d.]. Understanding Gamma Correction. https://www.cambridgeincolour.com/tutorials/gamma-correction.htm

[8] [n. d.]. You should probably turn off the Galaxy S10's face unlock if you value basic security. https://www.androidauthority.com/galaxy-s10-face-unlock-insecure-964276/

[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[10] Max K Agoston. 2005. *Computer Graphics and Geometric Modeling: Implementation and Algorithms*. Springer London.

[11] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series.. In *KDD workshop*, Vol. 10. Seattle, WA, 359–370.

[12] Avishek Joey Bose and Parham Aarabi. 2018. Adversarial Attacks on Face Detectors using Neural Net based Constrained Optimization. arXiv:cs.CV/1805.12302

[13] Peter J Burt and Edward H Adelson. 1983. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (TOG)* 2, 4 (1983), 217–236.

[14] H. Chen, W. Wang, J. Zhang, and Q. Zhang. 2020. EchoFace: Acoustic Sensor-Based Media Attack Detection for Face Authentication. *IEEE Internet of Things Journal* 7, 3 (March 2020), 2152–2159. https://doi.org/10.1109/JIOT.2019.2959203

[15] Shaxun Chen, Amit Pande, and Prasant Mohapatra. 2014. Sensor-assisted facial recognition: an enhanced biometric authentication system for smartphones. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 109–122.

[16] Y. Chen, J. Sun, X. Jin, T. Li, R. Zhang, and Y. Zhang. 2017. Your face your heart: Secure mobile face authentication with photoplethysmograms. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2017.8057220

[17] Tejas I Dhamecha, Aastha Nigam, Richa Singh, and Mayank Vatsa. 2013. Disguise detection and face recognition in visible and thermal spectrums. In *Biometrics (ICB), 2013 International Conference on*. IEEE, 1–8.

[18] Kimberly A. Dukes. 2014. *Gram–Schmidt Process*. American Cancer Society. https://doi.org/10.1002/9781118445112.stat05633 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat05633

[19] Li Fei-Fei, Rob Fergus, and Pietro Perona. 2006. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence* 28, 4 (2006), 594–611.

[20] Hideki Hayakawa. 1994. Photometric stereo under a light source with arbitrary motion. *JOSA A* 11, 11 (1994), 3079–3089.

[21] Hyung-Keun Jee, Sung-Uk Jung, and Jang-Hee Yoo. 2006. Liveness detection for embedded face recognition system. *International Journal of Biological and Medical Sciences* 1, 4 (2006), 235–238.

[22] Francis A Jenkins and Harvey E White. 1937. *Fundamentals of optics*. Tata McGraw-Hill Education.

[23] Wonjun Kim, Sungjoo Suh, and Jae-Joon Han. 2015. Face liveness detection from a single image via diffusion speed model. *IEEE transactions on Image processing* 24, 8 (2015), 2456–2465.

[24] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, Vol. 2. Lille.

[25] Klaus Kollreider, Hartwig Fronthaler, and Josef Bigun. 2009. Non-intrusive liveness detection by face images. *Image and Vision Computing* 27, 3 (2009), 233–244.

[26] Klaus Kollreider, Hartwig Fronthaler, Maycel Isaac Faraj, and Josef Bigun. 2007. Real-time face detection and motion analysis with application in liveness assessment. *IEEE Transactions on Information Forensics and Security* 2, 3 (2007), 548–558.

[27] Andrea Lagorio, Massimo Tistarelli, Marinella Cadoni, Clinton Fookes, and Sridha Sridharan. 2013. Liveness detection based on 3D face shape analysis.. In *IWBF*. 1–4.

[28] Patrick Lazik and Anthony Rowe. 2012. Indoor pseudo-ranging of mobile devices using ultrasonic chirps. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. 99–112.

[29] Yan Li, Yingjiu Li, Qiang Yan, Hancong Kong, and Robert H Deng. 2015. Seeing your face is not enough: An inertial sensor-based liveness detection for face authentication. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1558–1569.

[30] Yan Li, Zilong Wang, Yingjiu Li, Robert Deng, Binbin Chen, Weizhi Meng, and Hui Li. 2019. A Closer Look Tells More: A Facial Distortion Based Liveness Detection for Face Authentication. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security (Asia CCS âĂŹ19)*. Association for Computing Machinery, New York, NY, USA, 241âĂŞ246. https://doi.org/10.1145/3321705.3329850

[31] Bofan Lin, Xiaobai Li, Zitong Yu, and Guoying Zhao. 2019. Face Liveness Detection by RPPG Features and Contextual Patch-Based CNN. In *Proceedings of the 2019 3rd International Conference on Biometric Engineering and Applications (ICBEA 2019)*. Association for Computing Machinery, New York, NY, USA, 61âĂŞ68. https://doi.org/10.1145/3345336.3345345

[32] Gang Pan, Lin Sun, Zhaohui Wu, and Yueming Wang. 2011. Monocular camera-based face liveness detection by combining eyeblink and scene context. *Telecommunication Systems* 47, 3-4 (2011), 215–225.

[33] Keyurkumar Patel, Hu Han, and Anil K Jain. 2016. Secure face unlock: Spoof detection on smartphones. *IEEE Transactions on Information Forensics and Security* 11, 10 (2016), 2268–2283.

[34] Yvain Quéau, Jean-Denis Durou, and Jean-François Aujol. 2017. Variational Methods for Normal Integration. *CoRR* abs/1709.05965 (2017). arXiv:1709.05965 http://arxiv.org/abs/1709.05965

[35] Qing Song, Yingqi Wu, and Lu Yang. 2018. Attacks on State-of-the-Art Face Recognition using Attentional Adversarial Attack Generative Network. arXiv:cs.CV/1811.12026

[36] Xiaoyang Tan, Yi Li, Jun Liu, and Lin Jiang. 2010. Face liveness detection from a single image with sparse low rank bilinear discriminative model. In *European Conference on Computer Vision*. Springer, 504–517.

[37] Di Tang, Zhe Zhou, Yinqian Zhang, and Kehuan Zhang. 2018. Face Flashing: a Secure Liveness Detection Protocol based on Light Reflections. *arXiv preprint arXiv:1801.01949* (2018).

[38] Lloyd N Trefethen and David Bau III. 1997. *Numerical linear algebra*. Vol. 50. Siam.

[39] Andrew J. Viterbi. 1995. *CDMA: Principles of Spread Spectrum Communication*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[40] Robert J Woodham. 1980. Photometric method for determining surface orientation from multiple images. *Optical engineering* 19, 1 (1980), 191139.

[41] Hang Yu, Tian-Tsong Ng, and Qibin Sun. 2008. Recaptured photo detection using specularity distribution. In *2008 15th IEEE International Conference on Image Processing*. IEEE, 3140–3143.

[42] Bing Zhou, Jay Lohokare, Ruipeng Gao, and Fan Ye. 2018. EchoPrint: Two-factor Authentication using Acoustics and Vision on Smartphones. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 321–336.