Real-time 3D Face Verification with a Consumer Depth Camera

Gregory P. Meyer Minh N. Do
University of Illinois at Urbana-Champaign

***Abstract*—We present a system for accurate real-time 3D face verification using a low-quality consumer depth camera. To verify the identity of a subject, we built a high-quality reference model offline by fitting a 3D morphable model to a sequence of low-quality depth images. At runtime, we compare the similarity between the reference model and a single depth image by aligning the model to the image and measuring differences between every point on the two facial surfaces. The model and the image will not match exactly due to sensor noise, occlusions, as well as changes in expression, hairstyle, and eye-wear; therefore, we leverage a data driven approach to determine whether or not the model and the image match. We train a random decision forest to verify the identity of a subject where the point-to-point distances between the reference model and the depth image are used as input features to the classifier. Our approach runs in real-time and is designed to continuously authenticate a user as he/she uses his/her device. In addition, our proposed method outperforms existing 2D and 3D face verification methods on a benchmark data set.**

I. Introduction

Face verification and recognition is the task of authenticating the identity of a person within an image, and it has been an active research topic for the past several decades [1]. Face verification is still a difficult problem due to the wide range of possible head poses and facial expressions. There has been a wealth of 2D face verification methods proposed over the years [2], [3], [4], [5], and there have been some recent advancements using deep convolutional neural networks [6], [7]. However, 2D methods are still not widely used for authentication since they are easily fooled with a photograph. Face verification using 3D sensors has the potential to be a more secure and reliable mode of authentication, and there has been a variety of 3D face verification methods proposed [8], [9], [10], [11]. Unfortunately, these methods often require high-quality 3D data captured from a high-end 3D sensor in a controlled environment and are not always practical for real-world situations.

We propose a system that uses a consumer-grade depth camera, such as Microsoft's Kinect, to perform real-time face verification in an uncontrolled environment. There are several benefits of using a depth camera for face verification. Depth cameras use infrared light to measure the 3D geometry of an environment; therefore, they are less sensitive to external illumination. In addition, with a 3D approach we can utilize a full 3D face model, which allows our technique to be more robust to changes in pose. There are also challenges associated with using a depth camera, as they are often noisy and low resolution.

To be able to verify the identity of a subject, we first construct a high-quality reference model offline by fitting a 3D morphable face model [12] to a sequence of depth images. Given a novel depth image and a reference model, we can authenticate the user at runtime by aligning the reference to the image and measuring the similarity of the two facial surfaces.

To align the model and the image we need to estimate the pose of the subject's head within the image. To accurately estimate the head pose, we utilize a state-of-the-art 3D head pose estimator [13], which uses a combination of particle swarm optimization (PSO) and the iterative closest point (ICP) algorithm to precisely align a 3D face model to a depth image. Unfortunately, the method proposed in [13] does not run in real-time. We propose an extension to their work, which dynamically changes the number of particles within the swarm to reduce computation when possible. We evaluate our modification to [13] on the Biwi Kinect head pose data set [14], and we show that we can significantly improve runtime performance without sacrificing accuracy.

To accurately compare the reference model to the image under a wide variety of real-world situations such as partial occlusions, changes in pose or expression, and alterations in hairstyle or eye-wear, we use a data driven approach. We train a random decision forest [15] to verify the identity of a subject based on the point-to-point distances between the reference model and the depth image. We evaluate our approach on the Eurecom Kinect face data set [16], and we demonstrate superior results compared to existing state-of-the-art 2D and 3D face verification methods on this data set.

Since our proposed method runs in real-time, we can continuously authenticate a person while he/she uses his/her device. For example, when a trusted user sits down at his/her computer, he/she is immediately authenticated and granted access, and when the user leaves or an intruder is detected, we can instantly revoke privileges.

II. Related Work

A wide variety of 2D and 3D face verification methods have been proposed over the decades [1]. A large portion of the proposed techniques perform face verification on 2D images [2], [3], [4], [5], [6], [7]. [2] represents a 2D face image as a linear combination of basis images, also known as Eigenfaces [17], and trained a support vector machine (SVM) to determine whether or not two face images match.

978-1-5386-6481-0/18/$31.00 ©2018 IEEE
DOI 10.1109/CRV.2018.00020

[3] represents the face with a set of Gabor wavelet features. They use the AdaBoost algorithm to select the best wavelet features for distinguishing a subject from other subjects, and they train a SVM using these features. [4] utilizes a convolution neural network to map a face image to a feature space, such that, the Euclidean distance between feature vectors represents how similar two faces are to each other. [5] learned two separate classifiers to perform face verification. One classifier recognizes attributes such as gender, race, age, physique, hairstyle, eye-wear, etc. The presence or absence of these features are used to verify a person's identity. The second classifier measures the similarity of facial regions. [5] combines the results of both classifiers to verify the identity of a subject. [6] proposed a similar method to [4], but uses a deep network to map an image to a feature vector, and they train a SVM using these features. Most recently, [7] proposed another approach that uses a deep neural network to learn a Euclidean embedding of face images for recognition, verification, and clustering. [6], [7] leverage deep convolutional neural networks to sufficiently outperform all previous 2D methods. Face verification techniques that utilize 2D images typically work well under ideal lighting conditions and when the face is viewed from a frontal position, and their accuracy can degrade when this is not the case. Leveraging 3D data for face verification has the potential to overcome these limitations. 3D sensors often emit their own light, so they are less affected by external lighting conditions. In addition, by leveraging a full 3D model of the face, it is possible for 3D methods to be less influenced by changes in head pose.

Several methods that utilize 3D information have been proposed for face verification [8], [9], [10], [11]. [8] aligns the profiles of two 3D facial surfaces, and uses the residual error after alignment to compare the similarity of the two faces. In addition, they compare the difference in the intensity images, which were used to reconstruct the geometry of the face. [9] uses the Hausdorff distance to compare two aligned 3D face models. [10] uses spin images [18] to detect facial features. The location of the features are used to create a normalized depth image of the face. Principal Component Analysis (PCA) is used to project the depth images to an eigenvector subspace. A SVM classifier is trained to verify faces within this eigenspace. [11] constructs a 3D model of a person's face by merging a set of depth images captured by a laser scanner. Given a novel depth image, they align the image to a 3D model using iterative closest point (ICP), and the residual error after alignment is used to verify the subject. Like [8], [11] also considers the 2D appearance of the user by comparing an intensity image to an image synthesized by rendering the 3D model. Previous 3D face verification methods often require high-quality 3D data and/or both 2D and 3D data to accurately identify a user. We propose a method for 3D face verification using only low-quality depth images captured by a consumer-grade camera.

III. Method

A. Model Fitting

Before we can attempt to verify the identity of a subject, we must create a reference model of person for comparison. We create a reference model of the person's face by fitting a 3D morphable face model to a set of depth images. A morphable model consists of an average 3D face shape $\boldsymbol{\mu}$ and a set of 3D face shape bases $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_M)$, where $M = 199$ [12]. A novel face shape is produced through a linear combination of the mean face and face bases,

$$\hat{\boldsymbol{S}} = \boldsymbol{\mu} + \sum_{m=1}^{M} \alpha_m \boldsymbol{s}_m = \boldsymbol{\mu} + \boldsymbol{S}\boldsymbol{\alpha}. \tag{1}$$

To fit the morphable model we need to identify the coefficients $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_M)$, such that, the face shape $\hat{\boldsymbol{S}}$ best matches our subject.

A single depth image is too noisy to accurately fit a model; therefore, we capture a video sequence of the subject's head in a variety of poses. We fit the model to a subset of depth images from the video sequence selected at uniform intervals. For each depth image i, we identify a set of corresponding points $\mathcal{C}_i$ by finding the closest points between the image and the model. We jointly solve for the coefficients of the morphable model $\boldsymbol{\alpha}$ and the pose of each depth image $\{\boldsymbol{R}_i, \boldsymbol{t}_i\}$ by minimizing the following objective function:

$$\min_{\boldsymbol{\alpha}, \{\boldsymbol{R}_i, \boldsymbol{t}_i\}} \sum_i \sum_{j \in \mathcal{C}_i} \Big(w_{ij} \left\| (\boldsymbol{R}_i \boldsymbol{v}_{ij} + \boldsymbol{t}_i) - (\boldsymbol{\mu}_{ij} + \boldsymbol{S}_{ij}\boldsymbol{\alpha}) \right\|^2 \Big) + \lambda \left\| \boldsymbol{\alpha} \right\|^2 \tag{2}$$

where (i, j) is the j-th corresponding point from the i-th image, $\boldsymbol{v}_{ij}$ is a 3D measurement from the i-th depth image, and $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{S}_{ij}$ are the mean and bases of a single vertex from the morphable model corresponding to $\boldsymbol{v}_{ij}$. To reduce the effect of outliers, the point correspondences are weighted inversely proportional to the distance between them, w_{ij}. Also, we use a stiffness term λ to control how much the morphable model is allowed to deform.

We initialize the coefficients $\boldsymbol{\alpha}$ to zero, and we initialize each pose $\{\boldsymbol{R}_i, \boldsymbol{t}_i\}$ using the method described in the following section. We repeatedly solve Eq. (2) to refine our estimate of the coefficients and poses. For each iteration, we update our point correspondence between the images and the model. In addition, we will reduce the stiffness term λ if the previous iteration did not significantly change the coefficients. We iterate until λ below a certain threshold.

We use Ceres Solver's [19] implementation of conjugate gradient method to solve Eq. (2). For all of our experiments, we use 25 images as a trade-off between quality and computational complexity. Examples reference models are shown in Figure 1.

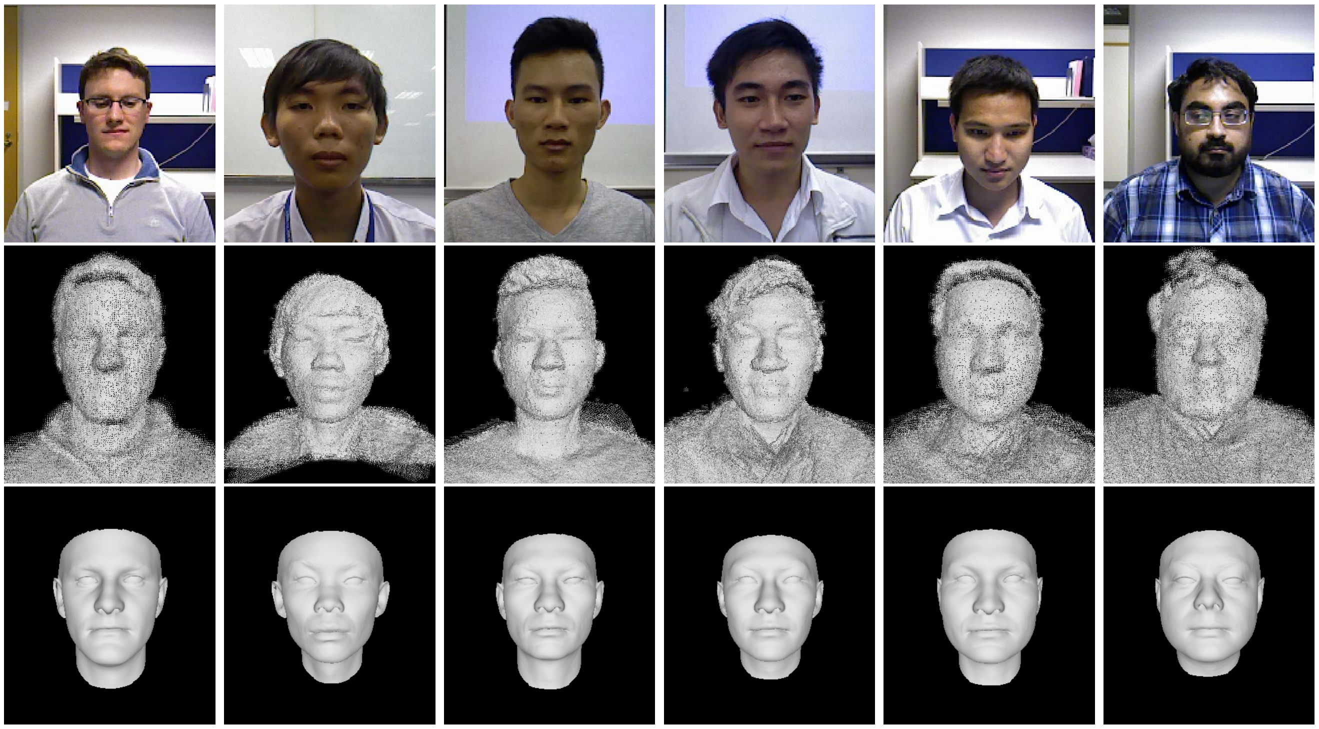
Figure 1: Example reference models (bottom row) constructed by fitting a 3D morphable face model [12] to a point cloud (middle row) assembled from a set of depth images. Color images (top row) are shown only for demonstration purposes, and they are not used by our system.

B. Pose Estimation

To verify whether or not a depth image matches a reference model, we first need to align the model to the image so that we can measure differences. We use the method proposed by Meyer *et al.* [13] to estimate the pose of the subject's head within the image. They use a combination of particle swarm optimization (PSO) and the iterative closest point (ICP) algorithm to accurately align a morphable face model to the depth data [13].

PSO is an evolutionary algorithm that uses a collection of particles to search for a global optimum in a non-convex parameter space [20]. The position of each particle in the swarm represents a potential head pose, $\mathbf{x} = (\theta_x, \theta_y, \theta_z, t_x, t_y, t_z)$, and a pose is evaluated using the following cost function:

$$E(\mathbf{x}) = E_v(\mathbf{x}) + \eta E_c(\mathbf{x}) \tag{3}$$

where the term $E_v(\mathbf{x})$ measures the point-to-plane distance between the image and the model in pose $\mathbf{x}$, and $E_c(\mathbf{x})$ penalizes a pose $\mathbf{x}$ when there are few corresponding points between the model and the image [13]. Each generation, the particles update their position based on their relative position to the other particles in the swarm. Afterwards, ICP is used to push the particles towards a local minima of the parameter space, which is shown by [13] to improve the overall convergence rate of PSO.

The original method proposed in [13] does not run in real-time. However, we observed that the entire swarm of particles is not always necessary to accurately estimate the pose in every frame. If the pose is correctly estimated in the previous frame, then only a few particles are required to update the pose in the subsequent frame. The more particles there are in the swarm, the more computation that needs to be performed. For this reason, we propose a variant of [13] that dynamically updates the number of particles in the swarm. By dynamically resizing the swarm, we can improve the runtime performance of the algorithm without significantly affecting accuracy.

The distance term E_v of the cost function (Eq. 3) gives us an indication of the accuracy of the estimated head pose, $\mathbf{x}^*$. When we correctly identify the pose, the value of $E_v(\mathbf{x}^*)$ should be around the expected measurement error of the depth camera. Therefore, we can update the size of the swarm for the next frame using the following procedure:

$$P_{k+1} = \begin{cases} P_{max} & \text{if } E_v(\mathbf{x}^*) \gg e \\ P_k + 1 & \text{else if } E_v(\mathbf{x}^*) > e \text{ and } P_k < P_{max} \\ P_k - 1 & \text{else if } E_v(\mathbf{x}^*) \leq e \text{ and } P_k > P_{min} \\ P_k & \text{otherwise} \end{cases} \tag{4}$$

where P_k is the number of particles in the current frame k,

$P_{min} = 1$ and $P_{max} = 10$ are the minimum and maximum number of particles in the swarm, respectively, and e is the expected measurement error of our sensor. For the Kinect camera, we experimentally determined $e \approx 5$ mm.

In addition, [13] alternates between estimating the head pose and updating the shape of the morphable model to fit the measured depth. Since we already fit the morphable model to the subject, we are able to skip the model update step and avoid additional computation. As a result, we are able to estimate the pose in real-time.

C. Distance Measurements

Once we align the reference model to the image, we can measure the differences between the two facial surfaces. We use a morphable model to create our reference model for every subject; therefore, each model will have the same number of vertices with the same topology. As a result, we can measure the same set of distances between any model and image regardless of the subject, which allows us to leverage a data driven approach to learn what facial similarities are important for accurately verifying the identity of a subject.

To measure the distance between a model and an image, we project every vertex from the reference model into the depth image,

$$\hat{\boldsymbol{v}}_i = \boldsymbol{R}\boldsymbol{v}_i + \boldsymbol{t}, \quad \begin{bmatrix} x & y & 1 \end{bmatrix}^T = \boldsymbol{K}\hat{\boldsymbol{v}}_i \tag{5}$$

where $\boldsymbol{v}_i$ is the i-th vertex in the reference model, $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation matrix and translation vector that align the reference model to the image, $\boldsymbol{K}$ is the camera's intrinsic calibration matrix, and (x, y) is the corresponding pixel coordinate in the depth image. Assuming a vertex is not self-occluded and the image is not missing data, we can compute the distance between a vertex and its corresponding depth measurement,

$$\delta_i = |D(x, y) - \hat{v}_i^z| \tag{6}$$

where $D(x, y)$ is the (x, y)-th measurement in the depth image and $\hat{v}_i^z$ is the z-component of vertex $\hat{\boldsymbol{v}}_i$. Often, measurements are missing from the depth image due to sensor noise and inaccuracies, and model vertices can be self-occluded due to head pose. In these situations, we set the distance δ_i to positive infinity. To reduce noise in the distances, we take the average over neighboring vertices,

$$\bar{\delta_i} = \frac{\sum_{k \in \mathcal{N}_i} \delta_k \cdot \mathbf{1}_{\delta_k < \infty}}{\sum_{k \in \mathcal{N}_i} \mathbf{1}_{\delta_k < \infty}} \tag{7}$$

where $\mathcal{N}_i$ is the 1-ring neighborhood of vertex $\boldsymbol{v}_i$ defined by the mesh topology of the reference model and $\mathbf{1}_{\delta_k < \infty}$ is an indicator function.

All the distances are aggregated into a distance vector $\boldsymbol{\delta} = (\bar{\delta_1}, \bar{\delta_2}, \ldots, \bar{\delta}_N)$, where the length of the vector is equal to the number of vertices in the reference model ($N = 53490$). Figure 2 visualizes the distances measured by our proposed method.

D. Face Verification

The distance vector $\boldsymbol{\delta}$ could be used in a variety of ways to verify a subject's identity. The simplest way is to compute the mean or medium of all the finite elements in $\boldsymbol{\delta}$ and compare it to a threshold. However, due to occlusions, changes in pose or expression, and alterations in hairstyle or eye-wear, this simple approach may not be sufficient to accurately authenticate a user. Instead, we chose to use a data driven approach to learn how to precisely verify a person using the distance vector as a feature vector. Specifically, we use a random decision forest [15] for its simplicity to train, and its past success in 3D vision [21], [22].

A random decision forest consists of an ensemble of decision trees. Each tree in the forest is trained using a set of training examples. The examples are selected randomly from the data set with replacement. A training example consists of a reference model and a depth image, and a label indicating whether or not the model and the image represent the same person. There are significantly more negative examples (the reference model and the depth image are different people) than positive examples (the reference model and the depth image are the same person), and this imbalance can pose a problem for training. Therefore, we re-weight the examples so that positive examples have a p probability of being drawn and negative examples have a chance of $(1 - p)$. A natural choice for p is 0.5, but we experimentally found that $p = 0.25$ produces better results.

To train a decision tree, we begin by extracting all the features from its set of training examples $\mathcal{S} = \{\boldsymbol{\delta}^n\}$. For each non-leaf node in the tree, we randomly select a subset of features and corresponding thresholds $\{\delta_i, \tau_i\}$, and determine which feature/threshold pair provides the most information gain,

$$\max_{\{\delta_i, \tau_i\}} G(\mathcal{S}', \delta_i, \tau_i) = H(\mathcal{S}') - \left(\frac{|\mathcal{S}_L|}{|\mathcal{S}'|} H(\mathcal{S}_L) + \frac{|\mathcal{S}_R|}{|\mathcal{S}'|} H(\mathcal{S}_R) \right) \tag{8}$$

where $H(\cdot)$ is the class uncertainty measure of a set, $|\cdot|$ is the size of a set, $\mathcal{S}' \subset \mathcal{S}$ is the subset of training examples that reach the node, $\mathcal{S}_L = \{\boldsymbol{\delta}^n \mid \delta_i^n < \tau_i,\ \boldsymbol{\delta}^n \in \mathcal{S}'\}$ is the subset of training examples in $\mathcal{S}'$ with feature δ_i less than τ_i, and $\mathcal{S}_R = \mathcal{S}' \setminus \mathcal{S}_L$ [15]. The class uncertainty measurement or entropy of the set is defined as follows:

$$H(\mathcal{S}) = -\left(\frac{|\mathcal{S}|^+}{|\mathcal{S}|} \log_2 \frac{|\mathcal{S}|^+}{|\mathcal{S}|} \right) - \left(\frac{|\mathcal{S}|^-}{|\mathcal{S}|} \log_2 \frac{|\mathcal{S}|^-}{|\mathcal{S}|} \right) \tag{9}$$

where $|\cdot|^+$ and $|\cdot|^-$ are the number of positive and negative examples in a set, respectively. When a leaf node is reached, either due to reaching the maximum depth of the tree or there are too few training examples at the node, then the probability of an positive example reaching this node, $p = |\mathcal{S}'|^+ / |\mathcal{S}'|$, is recorded.

To test whether or not a novel example is positive, *i.e.* verify a depth image matches a reference model, we extract

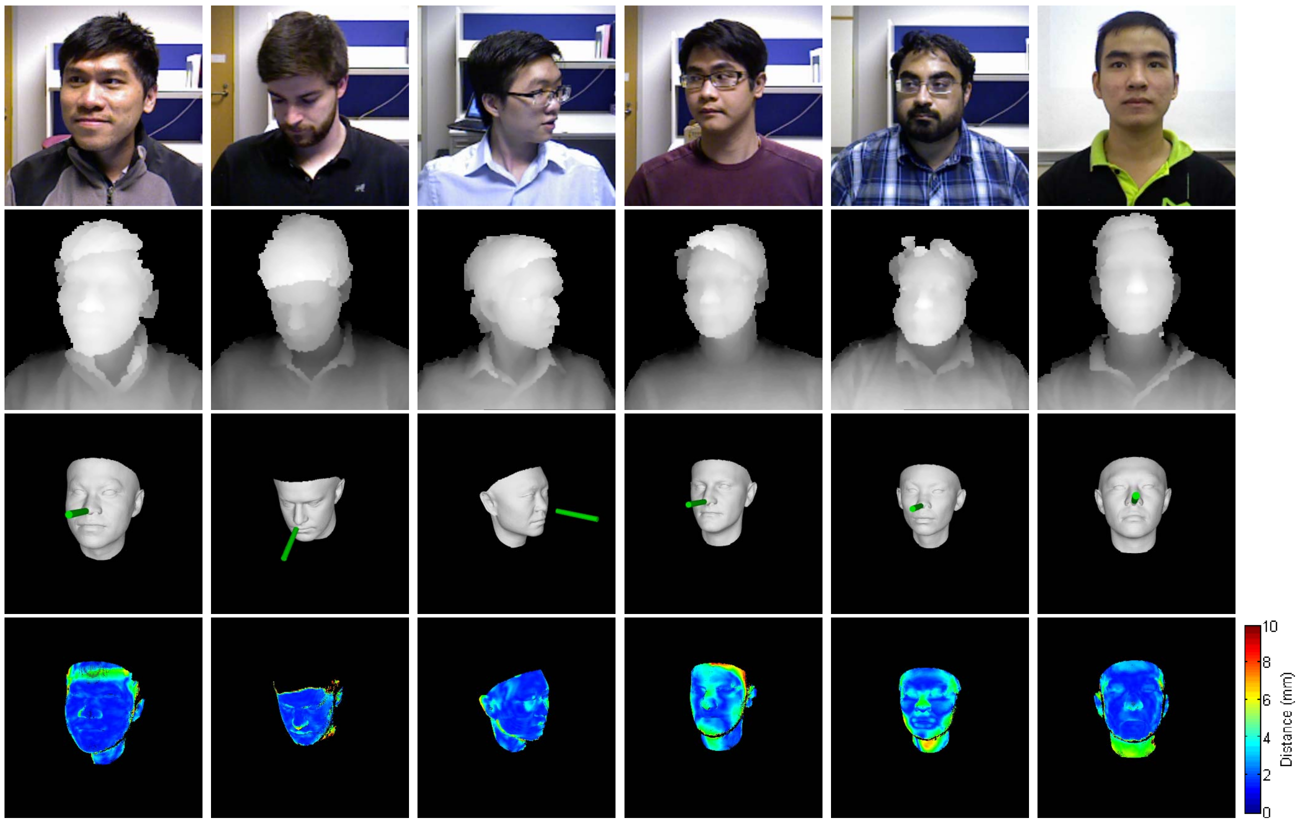

Figure 2: The distances (fourth row) are computed by aligning and projecting our reference model (third row) into a depth image (second row) and computing the difference. These distances are used to verify the identity of a subject. The first three columns depict positive examples where the reference model and the depth image share the same identity. The remaining columns illustrate negative examples. Observe how the distances change between the positive and negative examples. Color images (first row) are only shown for demonstration purposes, and they are not used by our system.

features from the example and pass it to each tree in the random forest. At each non-leaf node, we use the stored feature/threshold pair to determine our path through the tree. Once we reach a leaf node in every tree, we average the stored probabilities to produce an overall probability indicating how likely the image and the model are the same person. If the probability is above a threshold, we claim the image and the model share the same identity.

IV. Experimental Results

A. *Pose Estimation*

We evaluate [13] and our extension to it on the Biwi Kinect head pose data set [14] using an Intel Core i7 CPU and NVIDIA GeForce GTX 660 GPU. The data set contains over 15K RGB and depth images captured by a Kinect camera. For each image, a ground truth head pose (rotation and translation) is provided. The accuracy and runtime performance for each approach is listed in Table I.

The effect of dynamically changing the number of particles in the swarm can be seen by comparing the second and third rows of Table I. The accuracy is reduced by a small amount, but the runtime performance is greatly improved. In addition, the original method proposed by Meyer *et al.* [13] estimates both the head pose and the shape of the morphable model for every frame. Since we fit the morphable model to the subject beforehand, we can avoid this additional computation, and the time saved is shown in the first row of Table I. For more information and additional comparisons to existing head pose estimation methods please refer to [13].

Our modification to [13] enables our system to accurately estimate the head pose in real-time.

B. *Face Verification*

Our goal is to design an real-time system for face verification using low-quality depth images from a consumer-grade depth camera. Therefore, we require a data set that contains multiple video sequences of a variety of subjects captured by a Kinect camera or similar device. We measure the performance of our method, as well as, existing state-of-the-art methods using the Eurecom Kinect face data set [16]. The data set is comprised of 52 subjects, 38 males and 14

Method	Model	Errors				Runtime
		Yaw	Pitch	Roll	Location	
Proposed*	Custom	2.2°	2.3°	2.4°	5.4 mm	39.5 ms
Proposed	Morph	2.3°	2.3°	2.6°	5.4 mm	64.4 ms
Meyer [13]	Morph	2.0°	2.1°	2.3°	5.1 mm	154.6 ms

Table I: The average absolute angular errors, the average translational error, and the average runtime on the Biwi Kinect data set, for our modification of [13], as well as, the original method.

females, with ethnicities including European, Asian, Indian, African, and Hispanic. The data set contains two video sequences per subject captured during different sessions. For each sequence, the subjects are positioned approximately a meter from the camera. The sequences exhibit a wide range of head poses with only moderate changes in expression. The entire data set contains over 50K RGB and depth images.

1) Testing Procedure: For our method and the existing methods we compare against, we use the same procedure to evaluate an approach on the data set. For each video sequence of a subject, we define a model. The model could be an image, a set of images, a 3D mesh, or some other representation of the subject's identity as specified by the method we are testing. Afterwards, we compare the model to all other images in the data set, and compute a similarity value as defined by the method. It is important to note that we do not compare the model to the sequence used to construct it. The similarity value is compared to a threshold to determine the method's true positive and false positive rate. We vary the threshold to obtain the receiver operating characteristic (ROC) of the approach.

2) Evaluation of Our Approach: To evaluate our proposed method on the data set, we use 3-fold cross validation. The data set is randomly partitioned into roughly three equal subsets where each subset contains either 17 or 18 subjects, and we perform three rounds of training and testing. In each round, two subsets are used for training the random decision forest and the remaining subset is used for testing. Each subset contains approximately 13,500 positive examples and 850,000 negative examples.

For each fold, we trained a random forest of T decision trees each with 50,000 randomly selected training examples to a depth of D. At each non-leaf node we randomly select $(100 \times d)$ feature/threshold pairs where d is the depth of the node. We measure the performance of our method by averaging the results across each fold.

We analyze the behavior of our approach using a variety of different configurations. Figure 3 depicts the performance of our method when we vary the number of trees T in the forest and fix the maximum depth to $D = 20$. Beyond 100 trees, we experience diminishing gains in performance. Figure 4 shows the results when we fix the number of trees to $T = 100$ and vary the maximum depth D. Allowing the depth of the trees to go beyond 20 levels does not change the output

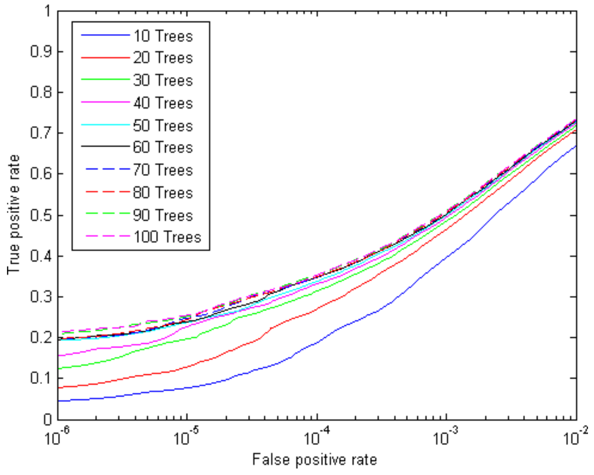

Figure 3: The ROC curves of our proposed method when we vary the number of decision trees within the random forest.

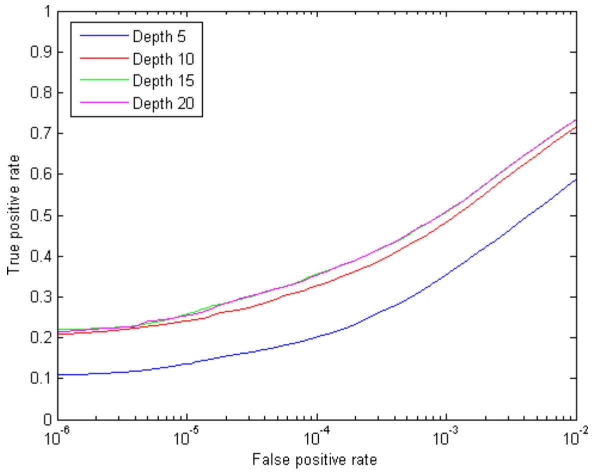

Figure 4: The performance of our approach when we change the maximum depth of the decision trees.

of the random forest because most of the trees do not reach a depth greater than 20 levels. Notice the classifier does not overfit the training data as the number of trees and maximum depth increase; this is a benefit of using the random decision forest [15]. For all the following experiments, we fix the number of trees to $T = 100$ and the maximum depth to $D = 20$.

To gain further insight into what is learned by the random decision forest, we count the number of times an element of our feature vector $\boldsymbol{\delta}$ is used by a decision tree to classify an image. The frequencies are illustrated in Figure 5. The random forest is learning to focus on the parts of the face with the most distinctive shapes, specifically the nose, brow, chin, and cheeks. Notice, the classifier is capable of learning to prioritize parts of the face that are visible, as we can see in Figure 5ac.

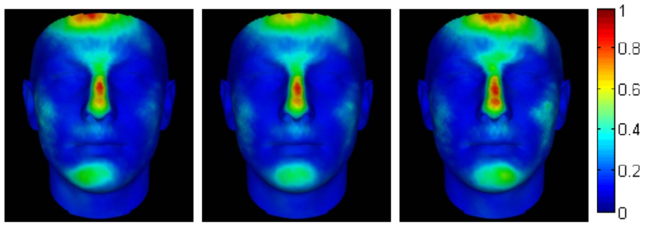

Figure 5: A set of heat maps representing the prevalence of the features within the random decision forest. The heat maps show the number of times a feature is used by the random forest to classify an image when the subject is looking (a) left (yaw angle is less than $-30°$), (b) forward (yaw angle is between $-30°$ and $30°$), and (c) right (yaw angle is greater than $30°$). Notice the subtle difference between (a) and (c) where the classifier learned to focus on the side of the face that is visible.

For all of our experiments, we perform face verification using a single depth image; however, it is possible to improve our method's performance by combining the results from a series of images. To this end, we use a sliding window to collect the output of our algorithm for a sequence of images, and the majority decision is used as the decision for entire window. Figure 6 shows the results when the sliding window contains 1, 5, and 10 seconds of video where the video is captured at 30 frames per second. These results motivate a possible future extension of our work, where multiple depth images are combined together to improve accuracy. One approach could be to integrate a sequence of depth images into a 3D model like in [23] and [24], and perform verification using this model instead of an image. Although, this approach would require more cooperation by the user.

3) Comparison with Existing Methods: We compare our method to existing state-of-the-art 2D and 3D face verification methods on the data set, and the results are shown in Figure 7.

For 2D methods, we compare against FaceNet [7]. FaceNet is a state-of-the-art method that leverages a deep convolutional neural network to map 2D face images to a Euclidean space where distances can be used to measure face similarity [7]. We use OpenFace [25], the open source implementation of FaceNet, to embed each image in the data set into this feature space. For each sequence, we hand select an image to be used as the subject's reference model. The squared $L2$ distance in the Euclidean space is used as the measure of similarity between the model and the data set images. The performance of FaceNet using a single image is depicted as the dashed red line in Figure 7.

Our proposed method uses multiple images from a video sequence to construct the 3D reference model of a subject. In

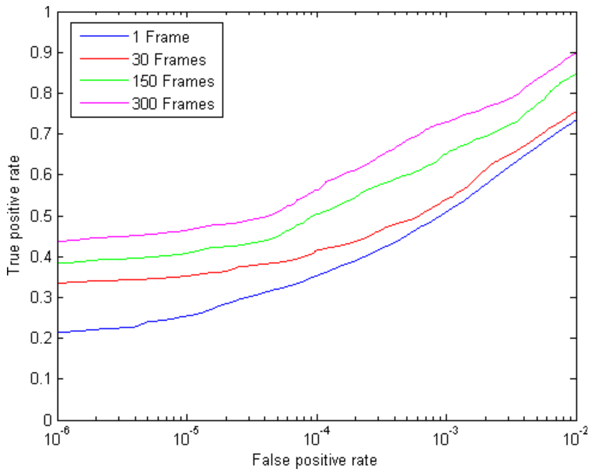

Figure 6: The performance of our method when we use a sliding window to combine the decision for a sequence of images. We show the results for when the sliding window contains 30, 150, and 300 frames, which is approximately 1, 5, and 10 seconds of video, respectively.

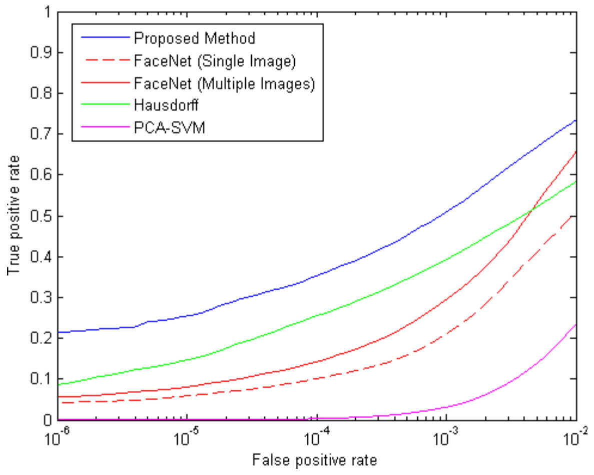

Figure 7: Our proposed method compared to existing state-of-the-art 2D and 3D face verification methods [7], [9], [10].

order to fairly compare our method with FaceNet, we replace the single hand selected image with the same 25 images used to construct our 3D face model. We use the minimum distance between all the reference images and a test image as the measure of similarity. We also experimented with using the mean and medium distance, but the minimum distance performed the best. The performance of FaceNet using multiple images is shown as the solid red line in Figure 7.

To examine how our proposed method and FaceNet deal with variations in head pose, we fix the false positive rate to 10^{-3} and compute the true positive rate as a function of yaw and pitch. As shown in Figure 8, our proposed method

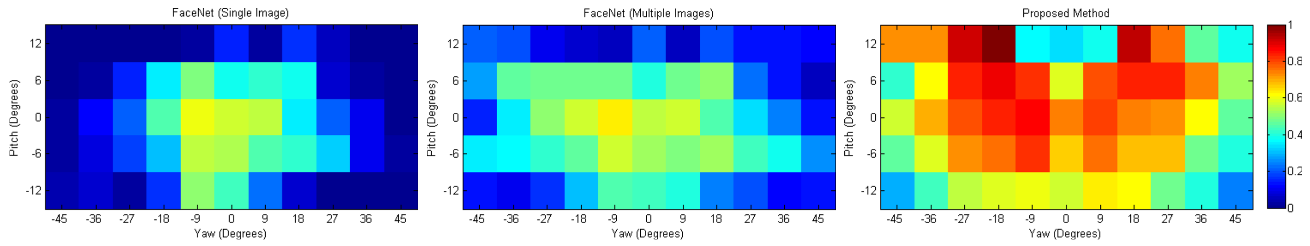

Figure 8: The true positive rate as a function of head pose (yaw and pitch) for our proposed method and FaceNet [7] on the Eurecom Kinect face data set [16]. The false positive rate for each method is fixed to 10^{-3}.

does well across a wide range of head poses, which is a benefit of using a 3D method over a 2D method for face verification.

For 3D methods, we compare against two state-of-the-art methods, [9] and [10]. [9] uses the partial Hausdorff distance to measure the similarity of two facial surfaces. To evaluate [9], we compute the partial Hausdorff distance between our 3D face models and the 3D point clouds generated by back-projecting the depth images in the data set. The results are illustrated as the green line in Figure 7. We believe we are able to obtain superior results to [9] because our approach learns how to adapt to the data.

[10] uses the estimated head pose to frontalize the depth images. Afterwards, the depth images are converted to Gaussian images, and principle component analysis (PCA) is performed on the entire data set. Each image is projected onto the vector space spanned by the top K principal components, and Conde *et al.* [10] trains a support vector machine (SVM) to distinguish between images of and not of the subject within this vector space. For our experiments, we used $K = 2500$ principal components, and the results are shown as the magenta line in Figure 7. We believe [10] does not perform well on this data set because the depth images from the Kinect camera are noisy and contain holes.

Our approach outperforms existing state-of-the-art 2D and 3D face verification methods on the Eurecom Kinect face data set [16]. We believe the strength of our approach is that we measure the distances between every point of the model and the image, and we learn which of these differences are important for authentication.

V. Conclusion

We proposed a real-time system for 3D face verification using a low-cost depth camera. We authenticate a subject by comparing a depth image to a 3D morphable model which was fit to their face. We leverage an existing state-of-the-art head pose estimator which we modify to run in real-time. We trained a random decision forest to learn what facial features are important for measuring the similarity between the image and the reference model. Our proposed method outperforms existing state-of-the-art 2D and 3D methods on a benchmark data set.

References

[1] A. F. Abate, M. Nappi, D. Riccio, and G. Sabatino, “2d and 3d face recognition: A survey,” *Pattern Recognition Letters*, vol. 28, no. 14, pp. 1885–1906, 2007.

[2] K. Jonsson, J. Matas, J. Kittler, and Y. Li, “Learning support vectors for face verification and recognition,” in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. IEEE, 2000, pp. 208–213.

[3] M. Zhou and H. Wei, “Face verification using gaborwavelets and adaboost,” in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 1. IEEE, 2006, pp. 404–407.

[4] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[5] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, “Attribute and simile classifiers for face verification,” in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 365–372.

[6] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “Deepface: Closing the gap to human-level performance in face verification,” in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1701–1708.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[8] C. Beumier and M. Acheroy, “Face verification from 3d and grey level clues,” *Pattern recognition letters*, vol. 22, no. 12, pp. 1321–1329, 2001.

[9] G. Pan, Z. Wu, and Y. Pan, “Automatic 3d face verification from range data,” in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP’03). 2003 IEEE International Conference on*, vol. 3. IEEE, 2003, pp. III–193.

[10] C. Conde and A. Serrano, “3d facial normalization with spin images and influence of range data calculation over face verification,” in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. IEEE, 2005, pp. 115–115.

[11] X. Lu, A. K. Jain, and D. Colbry, “Matching 2.5 d face scans to 3d models,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 1, pp. 31–43, 2006.

[12] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3d faces,” in *Conf. on Computer Graphics and Interactive Techniques*, 1999, pp. 187–194.

[13] G. P. Meyer, S. Gupta, I. Frosio, D. Reddy, and J. Kautz, “Robust model-based 3d head pose estimation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3649–3657.

[14] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, “Random forests for real time 3d face analysis,” *Int. J. Comp. Vision*, vol. 101, no. 3, pp. 437–458, 2013.

[15] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] R. Min, N. Kose, and J.-L. Dugelay, “Kinectfacedb: A kinect database for face recognition,” *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 44, no. 11, pp. 1534–1548, Nov 2014.

[17] M. Turk and A. Pentland, “Eigenfaces for recognition,” *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[18] A. E. Johnson, “Spin-images: a representation for 3-d surface matching,” Ph.D. dissertation, Carnegie Mellon University, 1997.

[19] S. Agarwal, K. Mierle, and Others, “Ceres solver,” http://ceres-solver.org.

[20] J. Kennedy, “Particle swarm optimization,” in *Encyclopedia of Machine Learning*, 2010, pp. 760–766.

[21] G. Fanelli, T. Weise, J. Gall, and L. Van Gool, “Real time head pose estimation from consumer depth cameras,” in *Pattern Recognition*, 2011, pp. 101–110.

[22] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, “Real-time human pose recognition in parts from single depth images,” *Comm. ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[23] M. Hernandez, J. Choi, and G. Medioni, “Laser scan quality 3-D face modeling using a low-cost depth camera,” in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, Aug 2012, pp. 1995–1999.

[24] G. P. Meyer and M. N. Do, “Real-time 3d face modeling with a commodity depth camera,” in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–4.

[25] B. Amos, B. Ludwiczuk, J. Harkes, P. Pillai, K. Elgazzar, and M. Satyanarayanan, “OpenFace: Face Recognition with Deep Neural Networks,” http://github.com/cmusatyalab/openface.