Adaptive Learning-based Temporal Tracker for 3D Head Shape Models

David Joseph Tan

tanda@in.tum.de

Nassir Navab

navab@cs.tum.edu

Federico Tombari

tombari@in.tum.de

CAMP

Technische Universität München

Munich, Germany

Abstract

We propose a learning-based 3D temporal tracker that estimates the orientation and location of the head in the 3D scene. The algorithm is based on random forest that learns the 6D pose from a class of head models. A unique attribute of our approach is the capacity to adapt the learned tracker for a specific user, *i.e.*, after learning, the tracker can deform the shape of the learned model to a specific instance of the class in order to match the user's head shape. To find the user's head shape model, we use a fast calibration method to personalize the model for a specific user. As a consequence, this technique enhances the accuracy of the head pose estimation as the personalized model becomes more detailed, and tracks at 1.4 ms per frame using a single CPU core.

1 Introduction and Related Work

The introduction of consumer depth cameras, acquiring dense RGB-D data at 30 Hz from low-cost hardware, created a trend in 3D head pose estimation and tracking. While the previous 3D head pose estimation techniques that were calculated directly from the RGB images are ambiguous due to the lack of depth perception, a number of techniques with RGB-D cameras have been proposed – aimed at estimating, in real-time, the complete 6-Degree-of-Freedom (6DoF) head pose across a sequence of frames. The most recurring trend has been so far based on the tracking-by-detection paradigm [6, 10, 12, 13], where the head pose is estimated independently at each frame with respect to a reference 3D model. As for these methods, learned approaches, based on either hough forests or regression forests [9], are deployed to learn the pose estimation from a training set of real depth images of different subjects under a variety of head motions and poses.

Recently, temporal trackers have also been proposed for the task of 3D head pose estimation, where the 6DoF transformation defining the head pose is updated from one frame to the next one. These approaches can be categorized as based either on energy minimization [8] or learning [11, 14]. By inheriting the concepts behind the Iterative Closest Point (ICP) algorithm [1, 5], the former minimizes a cost function to estimate the frame-to-frame transformation parameters. Since all the iterative computations such as the Jacobian matrices and the linear optimizations are conducted online, this approach is flexible to deal with

changes between the reference model and the input data. For instance, when a personalized head shape is acquired by deforming the mean model to fit a user, trackers based on energy minimization can easily modify the cost function at run-time to evaluate the points on the deformed model.

On the other hand, learning-based temporal trackers [17, 19] perform most of the computations during learning, which makes it significantly faster and more efficient at run-time. Inspired by energy-based methods, they build random forests [9] to describe the relation between the residuals and the transformation parameters such that, in tracking, the forest directly predicts the parameters, and avoids the computation of the Jacobian matrices and the linear optimization. In order to be applicable for different users, they learn a generalized tracker that incorporates different head models to capture the common structure across different users. However, once learned, it becomes complicated – or time consuming – to re-structure the trees of a forest so to adapt them for a specific subject’s characteristics. Thus, in [17, 19], when a personalized head shape model is given as part of the dataset [8], they propose to learn a new forest specialized for the user. As shown by the results in [17, 19], the subject-specific tracker is more accurate but restricted to an individual, and requires a separate forest for every new user; while, the generalized tracker does not require re-learning for a new user but yields lower pose accuracy.

Therefore, we propose a learning-based temporal tracker that uses random forests [9] to leverage the positive traits of the aforementioned methods (see Sec. 5). We learn a generalized tracker that captures the common structure from different users: in contrast to [17, 19], the proposed algorithm is capable of taking the personalized head shape model to adapt the tracker for the specified user, even without re-learning a new set of trees or re-structuring the existing trees. Fundamentally, we learn the forest from a class of head models taken from different head shape models of different users; then, in tracking, we adapt the forest to a specific instance of a head model. It follows that, although the proposed tracker is generalized to be applicable for different users, the resulting accuracy is the same as that of a subject-specific tracker, this yielding a remarkable accuracy with respect to other temporal tracking techniques. At the same time, we maintain the efficiency of learning-based approaches, with an average tracking time of 1.4 ms per frame.

To attain the personalized head shape model, a fast calibration strategy deforms a mean model to acquire a detailed head shape (see Sec. 4). Given the 3D CAD models of various users from the database [8], computing the Principal Component Analysis (PCA) of these models [2, 7, 18] simplifies the deformation into shape coefficients (see Sec. 2). Each setting of the shape coefficients induces a meaningful deformation on the mean model. Our calibration then accumulates a set of depth images of the user, taken from different views, and the corresponding head pose, tracked using the mean model. Thereafter, it simultaneously optimizes the shape coefficients across all the images and transformation parameters for each image to minimize the displacement between the synthetic model and the user’s head on the acquired frames. The resulting shape coefficients are then used as input to the tracker. We wish to point out that advanced parametric 3D head models have been proposed [7, 8], although they fall off the scope of this paper, as they mostly work offline and mostly aim at obtaining a personalized head reconstruction. Conversely, in our work, the personalization is done with the goal of improving the robustness and accuracy of the head tracking.

In this paper, Sec. 2 describes the deformation of the head shape model as well as its transformation through the 6DoF pose. Based on these, Sec. 3 derives an objective function that registers the deformed model to the input depth images, which is used for both the calibration and the temporal tracker. Thereafter, we discuss the fast calibration strategy

Figure 1: (a) The transformation of the points from the mean model to the depth image ($\mathbf{D}$) passes through the deformable model parameterized by β and the pose parameterized by θ . (b) Jacobian matrix for the calibration. (c) Sample input frames for calibration.

in Sec. 4 in order to acquire user-specific head shapes; and, we also discuss the adaptive learning-based temporal tracker in Sec. 5 that takes the personalized head shape parameters in order to deform the learned model. Finally, Sec. 6 quantitatively evaluates our algorithm and highlights our claims in terms of accuracy and efficiency.

2 Head Shape and Pose Model

Given a set of 3D CAD models from various users (*i.e.*, the ones in the BiWi database [9]), our morphable model relies on Principal Component Analysis (PCA) applied on the vertices of the models [2, 21, 28]. A novel head model can then be represented as the linear combination of the mean model and the principal components. Each vertex deforms as

$$V_\mu(\beta) = \mathbf{X}_\mu + \sum_{i=1}^{n_\beta} \beta_i \mathbf{v}_i = \mathbf{X}_p \Rightarrow \text{inversely, } \mathbf{X}_\mu = \mathbf{X}_p - \sum_{i=1}^{n_\beta} \beta_i \mathbf{v}_i \quad (1)$$

where $\mathbf{X}_\mu$ is the location of a vertex on the mean model, $\{\mathbf{v}_i\}_{i=1}^{n_\beta}$ is the set of principal components and $\mathbf{X}_p$ denotes the resulting point. Here, $\hat{\beta} = [\beta_i]_{i=1}^{n_\beta}$ are the shape coefficients that control the deformation of the shape model. Therefore, in order to personalize a head shape model for a specific user, we seek a set of shape coefficients to generate a deformable head model that matches the user as illustrated in Fig. 1(a). Since PCA is a dimensionality reduction scheme, setting the number of principal components to n_β dictates the range of the deformation. A smaller n_β is bounded to the coarser movement, while a larger n_β incorporates the finer details on the deformation (see Sec. 6.1).

Interpolation. Interestingly, when comparing the model and the depth image in the calibration (see Sec. 4) and the tracker (see Sec. 5), their residuals are not restricted to the vertices of the model, but also incorporate arbitrary points on the surface of the mean model. Due to this, the principal components for an arbitrary point are calculated through the barycentric interpolation on the three vertices of the closest face. Consequently, we can map every point from the mean model to a corresponding point on the deformed model.

Head Pose. In this paper, the final goal is to estimate the 3D head pose in the scene. The relation between the model and the scene, acquired through the depth images $\mathbf{D}$, is the 4×4 transformation matrix

$$\mathbf{T} = \begin{bmatrix} \tilde{\mathbf{R}}_{3 \times 3} & \tilde{\mathbf{t}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (2)$$

that incorporates the 3×3 rotation matrix $\tilde{\mathbf{R}}_{3 \times 3}$ and the translation vector $\tilde{\mathbf{t}}$ in 3D space. The 6DoF transformation is parameterized by a 6×1 pose vector $\boldsymbol{\theta}$ that concatenates the rotation parameters and the translation vector. Note that we distinguish the inhomogeneous coordinates with a tilde written as $\tilde{\mathbf{x}}$, in contrast to the homogeneous coordinates $\mathbf{x}$.

3 Objective Function

The objective function compares the relation between the mean model and the depth image through the shape coefficients and the head pose as illustrated in Fig. 1. The goal then is to optimize the shape and pose in order to register the mean model into the scene. Our algorithm follows a tracking framework that temporally relays the transformation parameters from one frame to the next, or, in general, from one iteration to the next (*i.e.* temporal tracking). Assuming that, at time t , the shape coefficients $\hat{\beta}_t$ and the pose $\mathbf{T}_t$ are computed, the motivation of a temporal tracker is to iteratively find $\hat{\beta}_{t+\tau}$ and $\mathbf{T}_{t+\tau}$ at time $t + \tau$.

A forward transformation, based on Fig. 1, from the mean model to the depth image $\mathbf{D}$ goes through the deformation by $\hat{\beta}$ and a pose by $\mathbf{T}$. Thus, a point $\mathbf{X}_\mu$ on the mean model is transformed as

$$\mathbf{X}_c = \mathbf{T}_t \left(\mathbf{X}_\mu + \sum_{i=1}^{n_\beta} \beta_i \mathbf{v}_i \right) \quad (3)$$

where $\mathbf{X}_c$ is a point on the camera coordinate system of the depth image.

Using a projective correspondence to compare the synthetic model and the input depth image, we denote $\mathcal{P}(\mathbf{X})$ as both the projection of $\mathbf{X}$ onto the depth image using the intrinsic parameters as well as the back-projection of the pixel into the 3D coordinates. The residual is then formulated as

$$r(\mathbf{X}_\mu, \hat{\beta}_t, \mathbf{T}_t) = \mathbf{n}_\mu^\top \left[\mathbf{X}_\mu - \left(\mathbf{T}_t^{-1} \mathcal{P}(\mathbf{X}_c) - \sum_{i=1}^{n_\beta} \beta_{i,t} \mathbf{v}_i \right) \right] = \mathbf{n}_\mu^\top \left(\mathbf{X}_\mu - \mathbf{X}_p + \sum_{i=1}^{n_\beta} \beta_{i,t} \mathbf{v}_i \right) \quad (4)$$

where $\mathbf{n}_\mu$ is the surface normal vector of $\mathbf{X}_\mu$ on the mean model and $\mathbf{X}_p = \mathbf{T}_t^{-1} \mathcal{P}(\mathbf{X}_c)$. Collecting the residuals from a set of points $\{\mathbf{X}_\mu\}_{\mu=1}^{n_\mu}$ on the mean model, we compose the energy function

$$E(\hat{\beta}_{\Delta t}, \mathbf{T}_{\Delta t}) = \sum_{\mu=1}^{n_\mu} \left[\mathbf{n}_\mu^\top \left(\mathbf{X}_\mu - \mathbf{T}_{\Delta t} \mathbf{X}_p + \sum_{i=1}^{n_\beta} (\beta_{i,t} + \beta_{i,\Delta t}) \mathbf{v}_i \right) \right]^2 \quad (5)$$

that simultaneously optimizes the change in shape coefficient $\hat{\beta}_{\Delta t}$ and change in pose $\mathbf{T}_{\Delta t}$. With the initial values of $\hat{\beta}_{\Delta t} = \mathbf{0}$ and $\mathbf{T}_{\Delta t} = \mathbf{I}_{4 \times 4}$ which is an identity matrix, this brings (5) back to (4). Optimizing (5) updates the shape as $\hat{\beta}_{t+\tau} = \hat{\beta}_t + \hat{\beta}_{\Delta t}$ and the pose as $\mathbf{T}_{t+\tau} = \mathbf{T}_t \mathbf{T}_{\Delta t}^{-1}$. Note that the calibration in Sec. 4 finds both shape and pose, while the tracker in Sec. 5 solely estimates the pose. In addition, the energy function only takes points that are on the face of the user and not on the entire model, allowing our objective function to focus on the regions that are visible in both the CAD model and the depth images [17, 19].

4 Head Shape Personalization

Driven to find the personalized head shape model for a specific user by deforming the mean model from Sec. 2 with the principal components, this section enforces the objective function from Sec. 3 to estimate the user's head surface through the shape coefficients. The optimization procedure is based on Gauss-Newton algorithm that simultaneously estimates both shape and pose parameters. Here, we decompose the transformation parameters using the small angle approximation written as

$$\mathbf{T}(\theta) \approx \begin{bmatrix} \mathbf{I}_{3 \times 3} + [\tilde{\mathbf{r}}]_{\times} & \tilde{\mathbf{t}} \\ \mathbf{0}^{\top} & 1 \end{bmatrix} \quad \text{such that} \quad \theta = \begin{bmatrix} \tilde{\mathbf{r}} \\ \tilde{\mathbf{t}} \end{bmatrix} \quad (6)$$

where $\tilde{\mathbf{r}} = [r_x, r_y, r_z]^{\top}$ and $[\cdot]_{\times}$ is the skew-symmetric matrix.

To acquire a detailed model, we utilize n_f keyframes composed of depth images captured from different viewpoints of the user's head while tracking using the mean model as shown in Fig. 1(c). In addition to the keyframes, the tracked pose $\mathbf{T}_{t,f}$ for every keyframe f and the initial shape coefficient $\hat{\beta}_{t_0} = \mathbf{0}$ are also included as input to the calibration. Here, the pose of each keyframe is independent from that of the other keyframes, while the shape variation is common to all keyframes. Thus, we are optimizing $6n_f + n_{\beta}$ parameters with n_f poses and one set of shape coefficients.

Points on the Mean Model. In an offline stage, we render the mean model from n_v different viewpoints, by placing a virtual camera around the model at equally spaced vertices of a geodesic grid [13]. For each of the rendered image, we select n_m pixels on the model and transform them back onto the mean model's coordinate system. We store the pairs of point and surface normal to the set $\chi_v = \{(\mathbf{X}_{\mu}, \mathbf{n}_{\mu})\}_{\mu=1}^{n_m}$ for the v -th view. As mentioned in Sec. 2, each point from the mean model has its own set of principal component vectors that deforms based on the shape coefficients.

During optimization, the pose $\mathbf{T}_{t,f}$ estimates the closest viewpoint among the rendered images. The set of points associated to this image is used to formulate the residual in (4) and the energy function in (5). This process is repeated for all the n_f frames. Notably, other approaches can be deployed for the selection of χ_v , *e.g.* by choosing the vertices of the mean model. However, by rendering the mean model, we guarantee that the selected points are located on the visible structure of the head.

Derivatives. Each point in χ_v constitutes one residual in the energy function. Investigating the residual of an arbitrary point from (5), we substitute the pose from (6) to expand the formulation as

$$\hat{\mathbf{r}}_{\mu} = \tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{X}}_{\mu} - \left(\tilde{\mathbf{n}}_{\mu}^{\top} \mathbf{I}_{3 \times 3} \tilde{\mathbf{X}}_p + \tilde{\mathbf{n}}_{\mu}^{\top} [\tilde{\mathbf{r}}]_{\times} \tilde{\mathbf{X}}_p + \tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{t}} \right) + \tilde{\mathbf{n}}_{\mu}^{\top} \sum_{i=1}^{n_{\beta}} (\beta_{i,t} + \beta_{i,\Delta t}) \tilde{\mathbf{v}}_i \quad (7)$$

$$= \tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{X}}_{\mu} - \tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{X}}_p - \tilde{\mathbf{r}}^{\top} [\tilde{\mathbf{X}}_p]_{\times} \tilde{\mathbf{n}}_{\mu} - \tilde{\mathbf{t}}^{\top} \tilde{\mathbf{n}}_{\mu} + \sum_{i=1}^{n_{\beta}} \beta_{i,t} \tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{v}}_i + \sum_{i=1}^{n_{\beta}} \beta_{i,\Delta t} \tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{v}}_i. \quad (8)$$

Then, the partial derivatives of (8) with respect to $\theta_{\Delta t}$ and $\hat{\beta}_{\Delta t}$ are computed as

$$\mathbf{J}_{\theta} = \begin{bmatrix} -[\tilde{\mathbf{X}}_p]_{\times} \tilde{\mathbf{n}}_{\mu} \\ -\tilde{\mathbf{n}}_{\mu} \end{bmatrix}^{\top} \quad \text{and} \quad \mathbf{J}_{\hat{\beta}} = \left([\tilde{\mathbf{n}}_{\mu}^{\top} \tilde{\mathbf{v}}_i]_{i=1}^{n_{\beta}} \right)^{\top}, \quad (9)$$

respectively. Note that $\mathbf{J}_\theta$ and $\mathbf{J}_{\hat{\beta}}$ of a point in $\mathcal{X}_v$ constitutes one row of the Jacobian matrix $\mathbf{J}$ of the energy function.

Optimization. Now, we have all the tools needed to optimize the energy function. The optimizer requires the full Jacobian matrix $\mathbf{J}$ of the $6n_f + n_\beta$ parameters. Due to the independence of the pose between frames, $\mathbf{J}$ is assembled in Fig. 1(b) as a sparse matrix, so that the poses from distinct frames are uncorrelated while the shape coefficients depend on all frames. Using Gauss-Newton, we concatenate the residuals from (4) into the vector ε and find the updated parameters as

$$\begin{bmatrix} [\theta_{\Delta, f}]_{f=1}^{n_f} \\ \hat{\beta}_{\Delta} \end{bmatrix} = -(\mathbf{J}^\top \mathbf{J})^{-1} \mathbf{J}^\top \varepsilon \quad (10)$$

such that the pose of the frame f is updated as $\mathbf{T}_{t+\tau, f} = \mathbf{T}_{t, f} \mathbf{T}_{\Delta, f}^{-1} = \mathbf{T}_{t, f} \mathbf{T}(\theta_{\Delta, f})^{-1}$ while the shape coefficients are updated as $\hat{\beta}_{t+\tau} = \hat{\beta}_t + \hat{\beta}_{\Delta}$. In practice, this optimization is an iterative procedure that refines the previous estimate.

5 Adaptive Temporal Tracker

As previously defined, a temporal tracker relays the transformation from one frame to the next. Assuming that the shape coefficients $\hat{\beta}$ are given, the goal is to estimate the 3D head pose

$$\mathbf{T}(\theta) = \begin{bmatrix} \mathbf{R}_z(r_z) \mathbf{R}_y(r_y) \mathbf{R}_x(r_x) & \tilde{\mathbf{t}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (11)$$

between two consecutive frames that uses the yaw, pitch and roll angles to construct the pose vector $\theta = [r_x, r_y, r_z, \tilde{\mathbf{t}}^\top]^\top$. Contrary to Sec. 4, these pose parameters do not require the small angle approximation because the tracker is a learning-based approach that does not need to compute the Jacobian matrix [13, 14]. Our approach learns random forests [9] to model the relation between the residuals in (4) from n_μ points on the model, written as $\varepsilon(\mathbf{T}) = [r(\mathbf{X}_\mu, \hat{\beta}, \mathbf{T})]_{\mu=1}^{n_\mu}$, and the pose parameters θ . In this way, the tracker takes $\varepsilon(\mathbf{T}_t)$ as input and predicts θ to update the location and orientation of the head pose estimate.

Similarly to Sec. 4, we also employ a geodesic grid to render depth images around the model from different viewpoints. However, for the tracker, we render both the mean model and several personalized CAD models generated by deforming the mean model by $\hat{\beta}$. In this way, we have the ground truth shape coefficients $\hat{\beta}_{\text{gt}}$ and the ground truth pose $\mathbf{T}_{\text{gt}}$ for each image. The points $\mathbf{X}_\mu$ for ε are taken from the mean model while the other models are used to generalize the tracker. Differentiating from the generalized tracker of [13, 14], our approach uses the deformations from the shape coefficients as part of the transformations in the residuals so that the learned structure matches the user, while [13, 14] only aligns the models and does not mathematically consider the difference in shape. As a consequence, our method produces better accuracy in tracking (see Sec. 6.2).

To focus on the visible structure present on the depth image, the temporal tracker learns one tree per parameter for each of the rendered view so that, in tracking, the given pose $\mathbf{T}_t$ determines the viewpoints that are close to the current pose and only use the trees within this neighborhood. In the succeeding paragraphs, we narrate the procedure in learning a tree from one viewpoint but the same scheme applies for the other vertices of the geodesic grid.

Learning Dataset. Learning the relation between ε and θ entails creating a dataset that mimics the optimization procedure. We introduce random transformations $\mathbf{T}_r = \mathbf{T}(\theta_r)$ on the ground truth $\mathbf{T}_{\text{gt}}$ in order to resemble the location in the previous frame. Then, when the points are transformed by $\mathbf{T}_t = \mathbf{T}_{\text{gt}}\mathbf{T}_r$, the forest needs to predict $\mathbf{T}_r^{-1}$ to update the pose as $\mathbf{T}_{t+\tau} = \mathbf{T}_t\mathbf{T}_r^{-1} = \mathbf{T}_{\text{gt}}$ to bring it back to the ground truth. Each random pose introduces a disturbance in the residual that is calculated by $\varepsilon_r = \varepsilon(\mathbf{T}_{\text{gt}}\mathbf{T}_r)$. Therefore, the trees in the forest learn the relation of the changes in the residuals induced by the changes in the random pose. With n_r random transformations, the learning dataset then is a set $\mathcal{S}$ with pairs of residuals and pose vector: $\mathcal{S} = \{(\varepsilon_r, \theta_r)\}_{r=1}^{n_r}$.

Learning. The construction of the trees in the forest iteratively splits the learning dataset $\mathcal{S}$ into two subsets that are inherited by the children. Splitting aims at making the pose parameters have approximately the same values in the leaves, *i.e.* measured by the standard deviation. Suppose that $\mathcal{S}_N$ arrives on the node, the branch thresholds the p -th residual such that $\mathcal{S}_l = \{(\varepsilon_r, \theta_r) \in \mathcal{S}_N \mid \varepsilon_r[p] \geq \kappa\}$ and $\mathcal{S}_r = \{(\varepsilon_r, \theta_r) \in \mathcal{S}_N \mid \varepsilon_r[p] < \kappa\}$ goes to the left and right child, respectively. To choose p , all indices of the residuals are tested with ten thresholds that are linearly spaced between the minimum and maximum. The best split is measured by the minimum sum of standard deviations of $\mathcal{S}_l$ and $\mathcal{S}_r$. As the tree grows, $\mathcal{S}$ continues to split until the standard deviation of the parameter is less than 0.1 mm or 0.1°, or the size of the subset is less than 25. The leaf stores the mean and standard deviation of the pose parameters taken from the subset in the learning dataset that reached this node.

Tracking. Given $\mathbf{T}_t$, tracking aims at estimating $\mathbf{T}_{t+\tau}$ by calculating transformation between two consecutive frames. As mentioned, we use $\mathbf{T}_t$ to find the neighborhood of learned viewpoints that are close to the current pose. The residual vector $\varepsilon(\mathbf{T}_t)$ for each viewpoint maneuvers from the root node down to the leaf, utilizing the learned threshold in each branch. Each leaf stores the mean and the confidence based on the standard deviation. We then take the average of 50% of all the predictions with the lowest standard deviation to assemble the pose vector θ_{pred} . Finally, the pose is updated with $\mathbf{T}_{t+\tau} = \mathbf{T}_t\mathbf{T}(\theta_{\text{pred}})^{-1}$.

6 Evaluation

In this section, we investigate the effects of the principal component in relation to the calibration strategy in Sec. 6.1. Then, we compare the accuracy of different methods [6, 8, 10, 11, 12, 14, 19] on a benchmark in Sec. 6.2, as well as the reported efficiency in Sec. 6.3.

The evaluation is based on the BiWi Kinect Head Pose Database [6]. It consists of 24 sequences with 20 different people. Each image in the sequence is labeled with the ground truth head pose. Interestingly, this dataset also includes the 3D CAD model of each subject. The experimental framework follows the same strategy as [12, 19] where we learn using the 18 subjects and evaluate on the remaining two. The two subjects are randomly chosen and five sets of experiments are conducted. To conduct a fair comparison, for each set, we only take the 18 subjects throughout the learning procedure. This includes creating the mean model and the shape coefficients as well as learning the trees. Hence, the two users are never seen by the learned model which, consequently, evaluate the adaptability to a new user's head shape. Moreover, this evaluation highlights the algorithm's generalization in tracking different individuals even with a limited number of users in the learning dataset.

The goal is to track the subject in each sequence and estimate his/her head pose. We then compare the estimated pose with the ground truth and compute the average error in trans-

Figure 2: Comparison of the accuracy given varying n_β of the 3D head shape model, evaluated by the displacement error of each vertex of the optimized model to the ground truth.

lation and rotation. The evaluation also incorporates the failure rates where the translation error is larger than 20 mm or rotation error is larger than 15° . Note that these errors were not included in the average errors. Also, the compared methods used different threshold values in their evaluation: such values are specified in the caption of Table 1. For the calibration of the head model, we first track the sequence with the mean model and collect three frames with frontal view and angled views of $\pm 30^\circ$. Then, we re-evaluate the sequence with the optimized head shape model.

6.1 Number of Principal Components

This evaluation visually shows the importance of the number of principal components (n_β) in affecting the head shape calibration. Fig. 2 visualizes the displacement error of each vertex from the optimized head shape and the ground truth model from the database [6]. When $n_\beta = 0$, we are using the mean model without any deformation which consequently have large errors on all vertices. As we increase n_β , all the displacement errors decrease and the resulting head shape model becomes more detailed. With $n_\beta = 20$, the errors on all vertices are less than 2.5 mm.

6.2 Accuracy

This evaluation shows the importance of using a detailed head shape model with an increasing number of principal components with regards to the accuracy of the head pose estimation. Evaluating on [6], Table 1 records the average error in rotation (yaw, pitch, roll) and translation (t_x , t_y , t_z) of our method with increasing n_β , and compares the results against [6, 8, [10], [11], [12], [13], [14].

When we track with the mean model only ($n_\beta = 0$), the average error values are inferior to all three methods of [14] but better than [6, 8, [10], [11], [12], [13]], while the failure rate is zero. The reason behind the large error is because the forest only takes the mean model in learning without any generalization to other subjects. When we increase the number of principal components to 5, we achieve comparable results to the generalized tracker [14]; and, the online learning method [14] that starts tracking from the generalized tracker and learns additional trees using the input depth images. Nevertheless, our method yields larger errors than their subject-specific tracker [14] that learns one forest per user. After a closer look at Fig. 2, the optimized model when $n_\beta = 5$ still has large errors in some regions, this affecting the overall accuracy. Then, starting from $n_\beta = 10$, we achieve state-of-the-art results against the other methods in Table 1. With a difference of 0.1 mm in translation and 0.1° in rotation, our errors have no statistical difference from the subject-specific tracker in

	Translation (mm)	t_x (mm)	t_y (mm)	t_z (mm)	Rotation (degrees)	Yaw (degrees)	Pitch (degrees)	Roll (degrees)	Fail (%)	
Other Methods	HF [8]	12.8 ± 6.8	6.9 ± 6.7	7.4 ± 5.6	4.7 ± 3.4	14.3 ± 10.0	5.7 ± 6.1	9.7 ± 8.9	5.9 ± 5.2	5.0
	FTR [8]	–	–	–	–	–	3.0 ± –	3.2 ± –	5.3 ± –	6.9
	TSPF [10]	8.4 ± 22.2	–	–	–	5.5 ± 16.5	3.0 ± 9.6	2.5 ± 7.4	3.8 ± 16.0	0.0
	PLEV [10]	7.2 ± 12.1	–	–	–	7.3 ± 5.9	4.1 ± 6.9	3.9 ± 4.0	3.2 ± 3.0	5.0
	HN [12]	8.1 ± 5.3	3.8 ± 4.5	4.6 ± 4.0	3.7 ± 3.0	9.8 ± 8.0	3.8 ± 3.7	6.7 ± 6.6	4.3 ± 4.9	1.0
	ARF* [14]	10.8 ± 6.9	5.5 ± 5.6	6.2 ± 6.1	4.1 ± 3.1	12.2 ± 9.0	5.5 ± 5.5	7.8 ± 7.9	5.0 ± 4.4	3.0
	Generalized [19]	2.5 ± 1.3	1.0 ± 0.9	1.4 ± 1.2	1.3 ± 1.0	2.1 ± 1.2	1.5 ± 1.4	1.1 ± 0.9	1.2 ± 1.1	0.0
	Online Learning [19]	2.4 ± 1.3	1.0 ± 0.8	1.4 ± 1.3	1.2 ± 0.9	2.0 ± 1.1	1.4 ± 1.3	1.1 ± 0.9	1.2 ± 1.1	0.0
	Subj.-Specific [19]	1.8 ± 0.9	0.8 ± 0.6	1.2 ± 1.0	0.8 ± 0.5	1.5 ± 0.8	0.9 ± 0.8	0.7 ± 0.6	0.6 ± 0.5	0.0
	Our Work	$n_\beta = 0$	3.1 ± 2.2	1.2 ± 1.0	2.2 ± 1.9	1.2 ± 1.4	2.4 ± 1.8	1.2 ± 1.6	1.2 ± 1.1	1.1 ± 1.0
$n_\beta = 5$		2.6 ± 2.4	0.8 ± 0.7	2.1 ± 2.4	0.9 ± 1.0	1.7 ± 1.0	0.7 ± 0.7	0.8 ± 0.7	1.0 ± 0.9	0.0
$n_\beta = 10$		1.9 ± 1.1	0.7 ± 0.7	1.3 ± 1.1	0.7 ± 0.5	1.6 ± 1.0	0.7 ± 0.6	0.7 ± 0.7	0.9 ± 0.9	0.0
$n_\beta = 15$		1.8 ± 1.0	0.7 ± 0.6	1.2 ± 1.0	0.7 ± 0.5	1.6 ± 1.0	0.8 ± 0.7	0.7 ± 0.7	1.0 ± 0.9	0.0
$n_\beta = 20$		1.8 ± 1.0	0.8 ± 0.7	1.3 ± 1.0	0.7 ± 0.6	1.6 ± 0.9	0.7 ± 0.6	0.7 ± 0.7	1.0 ± 0.9	0.0

Table 1: Evaluating on the BiWi dataset [8], the error values of our tracker with varying n_β is compared to different head pose estimation algorithms. The failure rate is the percentage of frames where the translational error is above 20 mm for [10], [13] or 50 mm for [8, 12, 14], the rotational error is above 15° for [8, 13], or when detection fails without any specified thresholds for [10]. For our method, we use 20 mm and 15°.

	HF [8]	FTR [8]	TSPF [10]	HN [12]	Gen. [19]	Online [19]	Subj.-Spec. [19]	<i>Our Work</i>
Time	17.8 ms	21.0 ms	37.9-75.9 ms	66.7-100.0 ms	1.5 ms	30.0 ms	1.1 ms	1.4 ms
Architecture	CPU (1 core)	CPU	CPU (1 core)	GPU	CPU (1 core)	CPU (8 cores)	CPU (1 core)	CPU (1 core)

Table 2: Timings with respect to the hardware architecture reported on different head pose estimation methods [8, 8, 10, 12, 14].

Figure 3: Comparison of the success rate in (a) translation and (b) rotation with varying thresholds as well as the timings in (c) calibration and (d) tracking for different n_β .

[19]. The same trend follows in Fig. 3(a-b), where we plot the success rates with varying thresholds. Here, the curves with n_β equal to 10, 15 and 20 overlap each other in the best results.

Therefore, as we increase the number of principal components from zero to 20, the calibration becomes more detailed as shown in Fig. 2, which, in effect, improves tracking accuracy in Table 1. When parameterizing $n_\beta = 20$, Table 1 also illustrates that our adaptive tracker executes the same accuracy as the subject-specific tracker [19]. Hence, although our adaptive tracker learns from a class of head models, its tracking accuracy is equivalent to a tracker that learns for a specific instance.

6.3 Efficiency

We highlight both the calibration time and tracking time with respect to the hardware requirements. All evaluations were conducted on an Intel(R) Core(TM) i7-3820QM CPU. Evaluating on varying n_β , Fig. 3(c) plots the increasing calibration time with respect to the increasing number of principal components. Notably, when $n_b = 10$, we require 104.0 ms to calibrate with only one core. This trend reaches 166.5 ms with $n_b = 20$. Between 10 and 20 principal components, we perform a fast calibration procedure of less than 200 ms while achieving the state-of-the-art accuracy (see Sec. 6.2). In comparison, the subject-specific tracker [19] requires 18.8 seconds to learn a new forest for a new user.

Moreover, in Fig. 3(d), the curve of the tracking time is constant at 1.4 ms with one CPU core for different n_β . In Table 2, we achieve a lower tracking time with a minimal hardware requirement against [8, 9, 10, 11] and the online learning approach of [19]. On average, our efficiency is similar to the learning-based temporal trackers in [19] with negligible difference, being 0.3 ms slower than the subject-specific tracker and 0.1 ms faster than the generalized.

7 Conclusion

We have presented the first real-time adaptive learning-based 3D temporal tracker that can morph the learned model into a specific instance of a given class of head models. To build a personalized head model for the user, we use a rapid calibration procedure that deforms a mean model to a user-specific 3D head shape model. Integrating the personalized model into the tracker, our approach effectively yields state-of-the-art accuracy in 3D head pose estimation. These results are comparable to a method from [19] that learns one forest for every user. In addition, our implementation reports an efficiency of 1.4 ms per frame while using just a single CPU core.

References

- [1] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.
- [2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.
- [3] Leo Breiman. Random forests. *Machine learning*, 2001.
- [4] Chen Cao, Qiming Hou, and Kun Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on graphics (TOG)*, 33(4): 43, 2014.
- [5] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3), 1992.
- [6] Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 2013.
- [7] Patrik Huber, Guosheng Hu, Rafael Tena, Pouria Mortazavian, P Koppen, WJ Christmas, Matthias Ratsch, and Josef Kittler. A multiresolution 3d morphable face model and fitting framework. In *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.
- [8] Songnan Li, King Ngan, Raveendran Paramesran, and Lu Sheng. Real-time head pose tracking with online face template reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.
- [9] Gregory P. Meyer, Shalini Gupta, Iuri Frosio, Dikpal Reddy, and Jan Kautz. Robust model-based 3d head pose estimation. In *International Conference on Computer Vision*, December 2015.
- [10] Chavdar Papazov, Tim K Marks, and Michael Jones. Real-time 3d head pose and facial landmark estimation from depth images using triangular surface patch features. In *Conference on Computer Vision and Pattern Recognition*, pages 4722–4730, 2015.
- [11] Carolina Redondo-Cabrera, Roberto Lopez-Sastre, and Tinne Tuytelaars. All together now: Simultaneous detection and continuous pose estimation using a hough forest with probabilistic locally enhanced voting. In *British Machine Vision Conference*, 2014.
- [12] Gernot Riegler, David Ferstl, Matthias Rüther, and Horst Bischof. Hough networks for head pose estimation and facial feature localization. In *British Machine Vision Conference*, 2014.
- [13] Robert Sadourny, Akio Arakawa, and Yale Mintz. Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere 1. *Monthly Weather Review*, 1968.

- [14] Samuel Schulter, Christian Leistner, Paul Wohlhart, Peter M Roth, and Horst Bischof. Alternating regression forests for object detection and pose estimation. In *International Conference on Computer Vision*, 2013.
- [15] David Joseph Tan and Slobodan Ilic. Multi-forest tracker: A chameleon in tracking. In *Conference on Computer Vision and Pattern Recognition*, 2014.
- [16] David Joseph Tan, Federico Tombari, Slobodan Ilic, and Nassir Navab. A versatile learning-based 3d temporal tracker: Scalable, robust, online. In *International Conference on Computer Vision*, 2015.
- [17] David Joseph Tan, Federico Tombari, and Nassir Navab. A combined generalized and subject-specific 3d head pose estimation. In *International Conference on 3D Vision*, pages 500–508. IEEE, 2015.
- [18] David Joseph Tan, Thomas Cashman, Jonathan Taylor, Andrew Fitzgibbon, Daniel Tarlow, Sameh Khamis, Shahram Izadi, and Jamie Shotton. Fits like a glove: Rapid and reliable hand shape personalization. In *Conference on Computer Vision and Pattern Recognition*, Las Vegas, United States, June 2016.
- [19] David Joseph Tan, Federico Tombari, and Nassir Navab. Real-time accurate 3d head tracking and pose estimation with consumer rgb-d cameras. *International Journal of Computer Vision*, pages 1–26, 2017.