A Combined Generalized and Subject-Specific 3D Head Pose Estimation

David Joseph Tan[1], Federico Tombari[1,2], Nassir Navab[1]
[1]CAMP, Technische Universität München
[2]DISI, Università di Bologna
{tanda,tombari,navab}@in.tum.de, federico.tombari@unibo.it

Abstract

We propose a real-time method for 3D head pose estimation from RGB-D sequences. Our algorithm relies on a Random Forest framework that is able to regress the head pose at every frame in a temporal tracking manner. Such framework is learned once from a generic dataset of 3D head models and refined online to adapt the forest to the specific characteristics of each subject. Through the qualitative experiments under different conditions, it demonstrates remarkable properties in terms of robustness to occlusions, computational efficiency and capacity of handling a variety of challenging head poses. In addition, it also outperforms the state of the art on the reference benchmark dataset with regards to the accuracy of the estimated head poses.

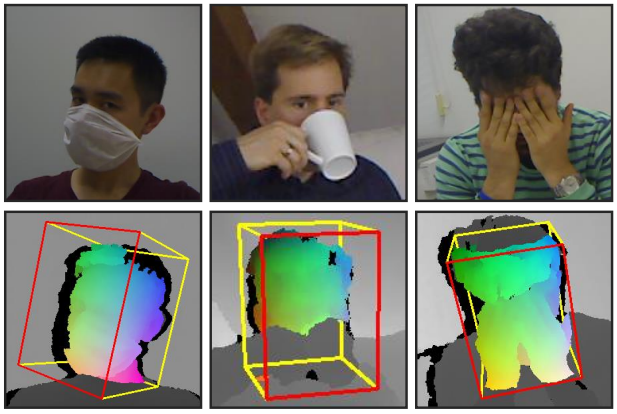

Figure 1. Close-up examples of the pose estimation results with different levels of occlusion. The bounding box and the colors symbolize the location of the back-projected points from the depth image in the model coordinate frame of the head.

1. Introduction

Estimating the pose of the heads that appear in a sequence of frames is an increasingly relevant task in computer vision, given the wealth of applications that rely on such task as a required step to achieve higher level goals. For instance, this is the case of several human-computer interaction systems that estimates input commands from the user by specific configurations of its facial features, which are now particularly relevant, *e.g.* in the gaming industry. In addition, head pose estimation is relevant for human behavior analysis and gaze analysis applied to faces, *e.g.* to automatically understand when the driver of a vehicle falls asleep. Finally, it is a key step for augmented reality applications, *e.g.* in the fashion industry for virtual mirrors applied to headwear, jewellery and eyewear, as well as in the context of 3D avatar creation for video conferencing and special effects.

Since head pose estimation aims to find the 6-degree-of-freedom of the head's rigid transformation in 3D space at each frame, state-of-the-art techniques [7, 14, 15, 17] employ 3D data as input in order to provide higher accuracy, such as the data provided by consumer RGB-D cameras (Microsoft Kinect, Asus Xtion). In particular, these methods are based on the tracking-by-detection paradigm, where the head detection and pose estimation is carried out in each frame independently from the previous ones. To achieve this goal, Hough Forests and Regression Forests proved to be effective in estimating the head's pose, after being trained on thousands of real depth images acquired from multiple people while moving their heads.

Instead, our work aims at performing head pose estimation from RGB-D data by means of a frame-to-frame temporal tracking approach, that incorporates the temporal information to estimate the head's pose throughout the video sequence. It is inspired by the object temporal tracker [19, 20] that uses depth images and Regression Forests to estimate the pose of an object from one frame to the next. However, in contrast to tracking an object with a precise 3D CAD model in [19, 20], we are addressing problems that generalizes the tracker for different variations of the head structures (*i.e.*, small deformations from a generic model). It involves combining a generalized model-based tracker with an online learning method to capture subject-specific structures that makes the tracker more robust against noise, occlusions and facial deformations.

Moreover, as a temporal tracking approach, it requires an initialization when a new face appears in the sequence or when the tracker loses the target. For this reason, we integrate an initialization stage based on a face detector [21] through the RGB images available from the RGB-D sequences. This allows our method to handle a variety of situations such as the absence of people in some frames of the sequence, as well as the simultaneous presence of more than one person in the scene.

Therefore, the proposed algorithm focuses on achieving a wide range of applications, motivated by the following fundamental criteria:

(1) *Robustness*. Accuracy is the most important aspect to consider in head pose estimation because it is the input to the applications that observe the interaction of the user with the objects or interaction among users. It follows that the accuracy must be robust in the presence typical occlusions such as eyeglasses and hand gestures, as well as robust in the presence of sensor artifacts such as holes and noise from depth cameras.

(2) *Efficiency*. We evaluate the efficiency of the algorithm in terms of the runtime per frame with the required processing power and memory consumption. It is ideal to attain low runtime with a low processing power and a low memory consumption.

(3) *Lax users*. The algorithm is capable of estimating the pose within the camera's field of view such that the users can move freely without being mindful of its position on the image. It must not restrict the users to stay on a specific distance from the camera or remain in a static position that is close to the principal point of the image.

(4) *Ease of use*. When running the algorithm, the users are not required to have any special skills or to perform any prerequisite steps. It implies that the algorithm does not require any a priori input information about the user.

Hence, our approach is guided by these criteria to ensure that our algorithm is easily adaptable for several diverse applications. Conforming to them, the quantitative results in our evaluation, obtained on the benchmark dataset [7] for 3D head pose estimation, demonstrate the remarkable accuracy, efficiency and robustness of our approach, which outperforms the state-of-the-art methods in terms of accuracy under different nuisances while retaining a much higher efficiency. In addition, the qualitative results also demonstrate the effectiveness and generality of our method under different occlusions, various user movements and extreme poses.

2. Related work

A survey on different head pose estimation algorithms is available in [13]. Several works on head pose estimation that rely on RGB images only focus on tracking facial features or landmarks instead of estimating the 3D pose of the head [1, 5, 6, 11, 12, 18] or on discretizing the pose to generate crude approximations [9, 10]. Unlike a rigid object with a constant structure that has a one-to-one relation from the image space to 3D space, the head structure differs from one subject to the next, which means that the landmarks do not have corresponding 3D points and their estimated pose is inaccurate due to the lack of depth perception of the subject in RGB images.

As for methods based on depth images, Breitenstein *et al*. [4] proposed to build a set of hypotheses by means of high-resolution depth images to locate the nose through 3D shape signatures and evaluate the hypotheses by computing the error from the reference pose image. Due to its computational requirements, this work uses GPU to run in real-time. Later, Fanelli *et al*. [8] also used high-resolution images but achieved real-time performance without GPU implementation and can handle small occlusions. Another relevant work from Fanelli *et al*. [7] proposes a tracking-by-detection framework based on Random Forest from consumer depth camera data. Their splitting features involves locating two 2D patches with random offsets and sizes from a pixel and computing the difference of the mean depth values enclosed in the patches. Each pixel predicts the yaw, pitch and roll angles for the head's rotation and a 3D vector to locate the nose.

With a similar 2D image features in learning as well as the parametrization of the three angles and a vector, several forest-based methods [14, 15, 17] continued the work of [7]. Among them, Schulter *et al*. [17] proposes Alternating Regression Forests (ARFs) that relates the trees in the forest by optimizing a global loss function, where their approach on head pose estimation uses the same 2D features as [7]. In [14], they evaluate four 2D patches as features in learning and introduce the Probabilistic Locally Enhanced Voting (PLEV) to locally aggregate the predictions of their Hough Forest. Finally, Riegler *et al*. [15] presents a combination of Hough Forests and Convolutional Neural Network for head pose estimation, which they call Hough Networks (HN). This approach extracts overlapping 2D patches in a regular grid on the image, where each patch is classified as foreground or background, and use the foreground patches to predict the pose parameters.

Primarily, these learning-based methods [7, 14, 15, 17] use 2D features to directly predict the 3D pose. However, by omitting the 2D-3D correspondence, when the head is translated around the camera's field of view, the 2D features cannot distinguish the relation between object's projetive view and the corresponding transformation parameters. Fig. 2 illustrates a simple example of the problem. Similar to [7, 14, 15, 17], the learning dataset includes head poses at the center of the image. One of the images in the learning dataset is shown in Fig. 2(a). After learning, we observe the pose prediction when translating the head in one

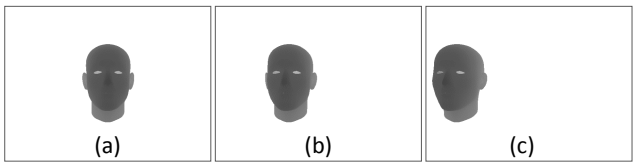

Figure 2. (a) is a rendered depth image of a head model located at the center of the image while (b-c) are rendered images of the same model after imposing a 3D translation towards the direction of the x-axis of the image.

direction from Fig. 2(a) to Fig. 2(b-c). Since the translation does not affect the rotation matrix, the ground truth rotation parameters for all images in Fig. 2 are the same. But, due to the projective transformation of the camera, the view of the head changes when translated. It follows that the values generated by the 2D features are different across Fig. 2(a-c). This leads to different predictions of the rotation parameters in Fig. 2(b-c) compared to Fig. 2(a), which are incorrect. When looking closely at Fig. 2(b-c), the farther the translation from Fig. 2(a), the projected image becomes more distinct and introduces larger errors.

One can argue to include the poses that are not at the center of the image such as Fig. 2(b-c) into the learning dataset. Nonetheless, the problem in using 2D features can also be extended to have multiple identical projective views on the image but with distinct transformation parameters. For instance, we can generate a similar view as Fig. 2(c) that is located at the center of the image with a different set of rotation and translation parameters. Then, when incorporating all of them into the learning dataset, the learning algorithm becomes confused in distinguishing which set of transformation parameters to associate each of the identical views.

In contrast, we do not have these problems since we are utilizing 3D points to describe the 3D structure of the head and to predict the 3D pose. Although the standard dataset from [7] restricts the location of the subject to be close to the principal point of the image, we illustrate several examples in Fig. 7 where the lax users can freely roam around the image. In addition, the problems related to the 2D features explain why our method generate high accuracy with only half the error of the other methods [7, 14, 15, 17] in Sec. 4.

We also demonstrate the efficiency of the algorithm to track in less than 2 ms per frame. Furthermore, while other approaches generalize their learning method to estimate the head pose of any subject, our work emphasizes the value of a combined generalized and subject-specific learning framework to become robust to handle large occlusions (Fig. 1) and extreme poses (Fig. 7(b)), where a significant amount of the facial features are not visible.

3. Proposed head pose estimation method

Given a sequence of RGB-D images, the objective is to find the 3D pose of the head in each frame. Suppose that the coordinates of the head are located in the model coordinate system, we define the pose as a 4×4 rigid transformation matrix $\mathbf{T}$ that transforms the points from the model coordinate system to the camera coordinate system, which is parameterized with:

$$\mathbf{T} = \mathbf{R}(\alpha, \beta, \gamma) \cdot \begin{bmatrix} \mathbf{I}_{3\times3} & \tilde{\mathbf{t}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{1}$$

where α, β and γ are the yaw, pitch and roll angles, while $\tilde{\mathbf{t}} = [t_x, t_y, t_z]^\top$ is the translation vector. The six transformation parameters construct the vector $\boldsymbol{\mu} = [\alpha, \beta, \gamma, t_x, t_y, t_z]^\top$. Hence, given n_h 3D points on the head as $\{\mathbf{X}_h\}_{h=1}^{n_h}$, we fit these points into the depth image $\mathbf{D}$ by observing the individual signed displacement function:

$$\epsilon_h^v(\mathbf{T}; \mathbf{D}) = \mathbf{N}_v^\top \left(\mathbf{T}^{-1}\mathcal{D}(\mathbf{x}_h) - \mathbf{X}_h\right) \tag{2}$$

where $\mathbf{x}_h$ are the projection of $\mathbf{T}\mathbf{X}_h$ onto the image, $\mathcal{D}(\mathbf{x})$ is the back-projection of the pixel $\mathbf{x}$ in $\mathbf{D}$ and $\mathbf{N}_v$ is a unit vector which is the direction of the displacement.

Our method is a temporal tracker that relays the pose from one frame to the next. It uses Random Forest [3] to learn the relation between the displacements ϵ_h^v from the n_h points on the head and the six parameters of $\mathbf{T}$. Then, at time t, we utilize the given pose at $t-1$ in the error function as $\epsilon_h^v(\mathbf{T}_{t-1}; \mathbf{D}_t)$ to predict the changes in transformation $\hat{\mathbf{T}}_t$ from $t-1$ to t through learned forest and update the subsequent pose as $\mathbf{T}_t = \mathbf{T}_{t-1}\hat{\mathbf{T}}_t$. To initialize the tracker, the face is detected at t_0 by means of a face detection algorithm that processes the RGB frame generating, as output, a 2D bounding box. This acts as input to the tracker, where the box needs to be back-projected in the 3D domain, since the tracker only uses the depth images to compute the head's pose throughout the sequence.

As for tracking, we propose to combine a generalized model-based tracker with a subject-specific online learning to adapt the generalized forest with the unique head structures of the subject. Between them, the generalized tracker learns the subject-independent *facial* structures based on multiple head models to predict the poses. Then, while tracking with the generalized model, we incrementally learn new trees from the input depth images to describe subject-dependent *head* structures. In this way, the subject-specific trees stabilize the entire forest to handle large occlusions and extreme head poses, where the generalized model is restricted to perform.

On one hand, the advantage of the generalized method is its capacity to align the tracked poses from multiple subjects into one coordinate system. In contrast, the subject-specific online learning builds the forest on the fly and does not consider the relation of one subject to the rest. Then, the resulting pose also becomes subject-specific and difficult to use in real applications. On the other hand, in the process

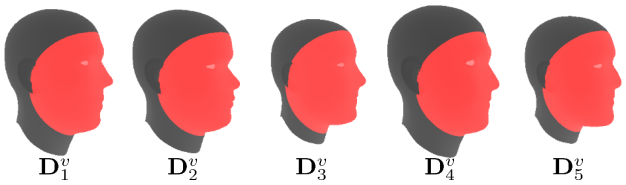

Figure 3. The head model from different subjects using the database of [7] is rendered at a constant v-th camera view, where the common structure is highlighted in red.

of generalizing, it can only track the common facial structures across multiple subjects, which creates the limitation in the amount of poses it can handle and in the robustness to handle occlusions. This is where the subject-specific online learning becomes valuable, since it incorporates the head structures that are unique to an individual subject, which makes it handle extreme poses and robust to large occlusions. As a result, by combining the two methods, we combine the advantage of a blueprint that consistently aligns different head models into a unified coordinate system and the advantage of subject-specific structures that alleviates the limitations of a generalized model.

3.1. Initialization

We initialize the tracker using the face detector of Viola and Jones [21] on the RGB image, which yields a 2D rectangular region for each detected face in the current frame. At the centroid of the rectangle $\mathbf{x}_c$, the corresponding pixel value in the depth image is back-projected, which constructs the initial transformation of the tracker given as:

$$\mathbf{T}_{t_0-1} = \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathcal{D}(\mathbf{x}_c) \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{3}$$

Thereafter, we solely use the tracker to estimate the head's pose in the current frame with $\mathbf{T}_{t_0} = \mathbf{T}_{t_0-1}\hat{\mathbf{T}}_{t_0}$, as well as the poses in the successive frames.

3.2. Generalized model-based tracking

From a set of CAD models for different subjects, the objective is to build a generalized temporal tracker based on Random Forest [3] that performs the pose estimation of the head on any given subject, irrelevant of whether it is part of the set of models or not. In this case, learning is conducted completely on the synthetic depth images of the model and does not require real depth images.

Common structure. To generalize the tracker, the first goal is to determine the common structures of the head across multiple subjects that are visible in their respective 3D CAD models as well as captured by the depth images for tracking. By observing Fig. 3, the face is the most similar structure across different subjects that is consistently visible in the models and the data captured by the camera.

Assuming that the models from different subjects are aligned, we define the *common structure* as the 3D points on the model that are τ_s away from the origin as shown in Fig. 3. This implies that we only use these points for tracking and the generalized model only tracks the facial structure of the head.

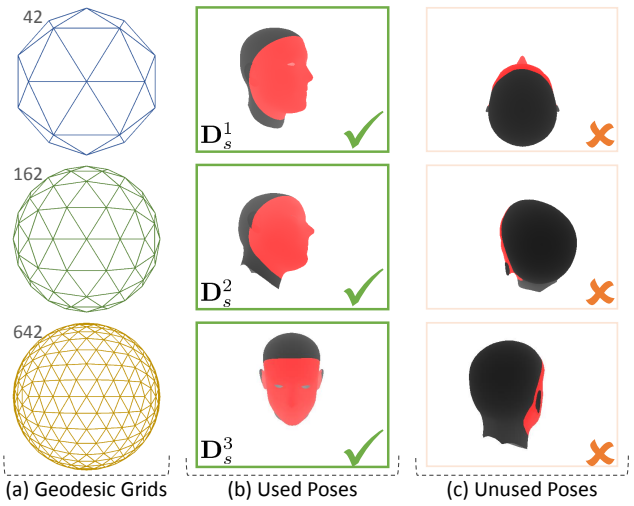

Figure 4. (a) Geodesic grids [16] with different n_v are constructed by iteratively dividing an icosahedron. From different camera views, (b-c) are examples of the rendered depth images of a head model. Depending on the visibility of the common structure (in red), (b) are poses used for learning while (c) are not.

Camera views. Instead of learning one forest for the entire model, we learn individual forests for different camera views of the model [19, 20], so that each forest focuses on predicting the transformation parameters from points on the model that are seen by the camera. Using a geodesic grid [16] with n_v vertices as shown in Fig. 4(a), the different camera views of a model is synthetically rendered by locating the camera on the vertices of the grid and placing the model in its center. Considering that there are some rendered images where the common structure are not visible as illustrated in Fig. 4(c), we only use a subset of geodesic grid's vertices, where the azimuth angle of the spherical coordinate system is between $[0, \pi]$ while the polar angle is between $[0, \frac{3\pi}{4}]$. For the v-th camera view and the s-th subject, the rendered depth image is denoted as $\mathbf{D}_s^v$, where the model is transformed with $\mathbf{T}_v$ from the camera coordinate system. As a consequence, we learn the v-th forest using the depth images across different subjects $\{\mathbf{D}_s^v\}_{s=1}^{n_s}$. In addition, we also define $\mathbf{N}_v$ from Eq. 2 as the unit vector directed from the model's origin to the camera location in the geodesic grid.

Learning dataset. When looking at the v-th view of the n_s subjects used for training, a unified depth image is constructed through its mean, computed as:

$$\bar{\mathbf{D}}^v = \frac{1}{n_s}\sum_s \mathbf{D}_s^v. \tag{4}$$

Again, the points on the common structure are collected and n_h points are randomly selected among them. These points are then transformed to the model coordinate system with $\mathbf{T}_v^{-1}$, and the results are denoted as the set of points on the model $\{\mathbf{X}_h^v\}_{h=1}^{n_h}$. Hence, in tracking, we are optimizing to fit these points onto the depth image through Eq. 2.

Since tracking aims to update the transformation parameters from $t-1$ to t, random transforms $\hat{\mathbf{T}}_r$ are imposed to simulate the location of $\{\mathbf{X}_h^v\}_{h=1}^{n_h}$ at $t-1$. The points are transformed by $\mathbf{T}_r = \mathbf{T}_v\hat{\mathbf{T}}_r^{-1}$ such that an update of $\hat{\mathbf{T}}_r$ brings them back to their ground truth location. As a result, the error introduced by the random transform is described in Eq. 2, hence the error from the n_h points can be accumulated with the vector $\boldsymbol{\epsilon}_r^v = [\epsilon_h^v(\mathbf{T}_r; \mathbf{D}_s^v)]_{h=1}^{n_h}$. Therefore, together with the transformation parameters τ_r of $\hat{\mathbf{T}}_r$, the learning dataset is assembled as $\mathcal{S} = \{(\boldsymbol{\epsilon}_r^v, \tau_r)\}_{r=1}^{n_s \cdot n_r}$ after transforming with n_r random poses on each of the n_s subjects. It is noteworthy to mention that $\bar{\mathbf{D}}^v$ is only used to determine the set of points on the model and not used to create the learning dataset.

Learning the forests. Similar to [19, 20], a tree in the forests learns the relation of $\boldsymbol{\epsilon}_r^v$ and a parameter of $\boldsymbol{\mu}_r$. Through the learning dataset, the nodes of the tree continuously split $\mathcal{S}$ into two subsets and pass the subsets down to the children.

At node N, splitting is accomplished by a feature θ_N and a threshold κ_N, applied on the subset of the learning dataset that arrives on the node $\mathcal{S}_N$. The feature is an index of the vector $\boldsymbol{\epsilon}_r^v$ and the threshold is a scalar value that splits $\mathcal{S}_N$ into:

$$\mathcal{S}_l = \{(\boldsymbol{\epsilon}_r^v, \tau_r) \in \mathcal{S}_N \,|\, \boldsymbol{\epsilon}_r^v[\theta] \geq \kappa\} \tag{5a}$$

$$\mathcal{S}_r = \{(\boldsymbol{\epsilon}_r^v, \tau_r) \in \mathcal{S}_N \,|\, \boldsymbol{\epsilon}_r^v[\theta] < \kappa\} \tag{5b}$$

where $\boldsymbol{\epsilon}_r^v[\theta]$ takes the scalar value of the θ-th index in the vector, while $\mathcal{S}_l$ and $\mathcal{S}_r$ are the subsets of $\mathcal{S}_N$ that go to the left and right child, respectively. To choose θ_N and κ_N, we individually test all indices of $\boldsymbol{\epsilon}_r^v$ and, for each index, threshold using a range composed of ten linearly spaced values. The feature and threshold that best splits the set is measured by the information gain:

$$G(\theta) = \sigma(\mathcal{S}_N) - \sum_{i \in \{l,r\}} \frac{|\mathcal{S}_i|}{|\mathcal{S}_N|}\sigma(\mathcal{S}_i) \tag{6}$$

where $\sigma(\mathcal{S})$ takes the standard deviation of a parameter of all $\boldsymbol{\mu}$ in $\mathcal{S}$. Hence, the pair of (θ_N, κ_N) is the one with the highest information gain. After iteratively splitting $\mathcal{S}_N$ to produce deeper trees, the splitting stops either when the tree reaches its maximum depth, or when $\sigma(\mathcal{S}_N)$ is low, which means that the parameters are homogeneous. Then, this node stores the mean and standard deviation of the parameters in all $\boldsymbol{\mu}$ from $\mathcal{S}_N$.

To learn the multi-view forests, the same process is applied to different parameters in the $\boldsymbol{\mu}$ as well as to different camera views of the head.

Tracking. With the transformation from $t-1$, we compute the camera location $\tilde{\mathbf{X}}_{t-1}^c = -\tilde{\mathbf{R}}_{t-1}^\top \tilde{\mathbf{t}}_{t-1}$ in the model coordinate system, where $\tilde{\mathbf{R}}_{t-1}$ is the 3×3 rotation matrix of $\mathbf{T}_{t-1}$ and $\tilde{\mathbf{t}}_{t-1}$ is its translation vector. From the learned multi-view forests, only the neighboring camera views from the geodesic grid that are within τ_n angular distance are taken for evaluation. The input for each view is the vector $\boldsymbol{\epsilon}^v = [\epsilon_h^v(\mathbf{T}_{t-1}; \mathbf{D}_t)]_{h=1}^{n_h}$. Then, the splitting parameters on the nodes manoeuvre the input towards a leaf, where the predicted parameter of $\boldsymbol{\mu}$ are stored. After evaluating all trees in the neighboring views, the final parameter values take the average of the 10% of the predictions with the least standard deviation. These parameters assemble the transformation matrix $\hat{\mathbf{T}}_t$ and updates the pose with $\mathbf{T}_t = \mathbf{T}_{t-1}\hat{\mathbf{T}}_t$. Lastly, we iteratively refine the predictions for each frame.

3.3. Subject-specific online learning

While tracking with the generalized model, the subject-specific online learning aims to learn new trees that incorporate specific head structures of the subject being tracked and remove the restriction in tracking only the facial structure of the generalized model. As input to learning at time t, we have the depth image $\mathbf{D}_t$ and the resulting pose $\hat{\mathbf{T}}_t$. Based on the pose, we locate the corresponding camera location $\tilde{\mathbf{X}}_t^c$ in the model coordinate system and use a 3D bounding box to segment the head in the image.

In contrast to Sec. 3.2, the n_h points $\{\mathbf{X}_h^v\}_{h=1}^{n_h}$ are randomly selected within the 3D points on the depth image that lie within the bounding box, which are transformed to the model coordinate system using $\mathbf{T}_t^{-1}$. Then, n_r random transformations $\hat{\mathbf{T}}_r$ are imposed on these points by transforming with $\mathbf{T}_r = \mathbf{T}_t\hat{\mathbf{T}}_r^{-1}$ to generate the error vector $\boldsymbol{\epsilon}_r = [\epsilon_h(\mathbf{T}_r; \mathbf{D}_s^v)]_{h=1}^{n_h}$ and build the learning dataset $\mathcal{S} = \{(\boldsymbol{\epsilon}_r, \boldsymbol{\tau}_r)\}_{r=1}^{n_r}$. Thereafter, learning using $\mathcal{S}$ is carried out in the same way as Sec. 3.2.

As a consequence, during tracking, the neighborhood of camera views from learning incorporates both the generalized and subject-specific trees. After comparing the resulting camera location in tracking and the camera locations from all the learned views through the angular distance between them, the trees that are within τ_n are evaluated. The final transformation parameters are the mean of the predictions from the trees.

3.4. Failure detection

We determine if tracking fails based on the confidence of the predictions from the final iteration. Considering the mean of the standard deviation from different trees, we impose that the confidence of the prediction is low if the mean is above τ_c. After having low confidence for n_f consecutive

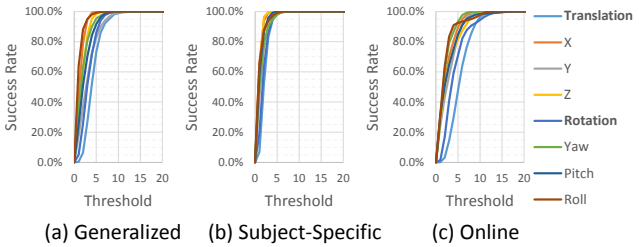

Figure 5. Success rates with varying thresholds for the error in translation (in mm) and rotation (in degrees), based on the (a) generalized (b) subject-specific or (c) online learning scheme.

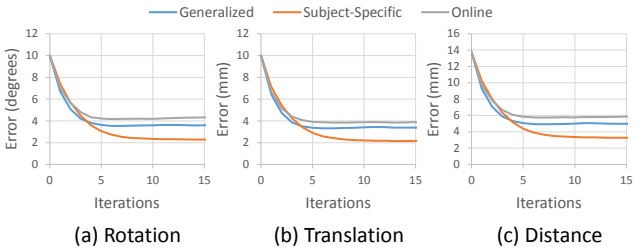

Figure 6. Convergence rates of the (a) rotation, (b) translation, and (c) distance from the model's vertices transformed using the estimates pose to the vertices with the ground truth pose.

frames, we conclude that tracker failed. Subsequently, the tracker performs re-initialization using Sec. 3.1.

4. Evaluation

We evaluate our algorithm using the Biwi Kinect Head Pose Database [7], which is arguably the most relevant public dataset with ground truth for head pose estimation from 3D data. It consists of 24 sequences from 20 different people with a total of approximately 15K images, where all images are labeled with the ground truth pose of the head. Each sequence includes one subject out of the 20 people appearing in the dataset, who stands approximately 1 meter away from the camera and close to the principal point of the image. The objective is to estimate the head pose while the subject rotates his head within $\pm 75^\circ$ yaw, $\pm 60^\circ$ pitch and $\pm 50^\circ$ roll angles. In addition, the database also includes subject-specific 3D CAD model for each sequence.

The error of the estimated pose is evaluated by computing the difference of the translation in the x-, y- and z-axis, as well as the rotation in the yaw, pitch and roll angles from the ground truth. Among the errors, we list the mean and standard deviation in Table 1, with the exclusion of the missed poses which have a translation error above 20 mm [14] or 50 mm [7, 15, 17]. Similarly, we also compute the success rate based on the percentage of estimated poses that have errors below a specified threshold in Table 2 and Fig. 5. Based on these outcomes, we compare our results with those reported in the papers of the four state of the art methods, namely Hough Forest (HF) [7], Hough Forest with Probabilistic Locally Enhanced Voting (PLEV) [14], Alternating Regression Forests (ARF*) [17] and Hough Networks (HN) [15].

While this standard procedure aims at evaluating the 3D pose accurately at each frame, we are also interested in evaluating the convergence rate of our tracker. To achieve this goal, we multiply the ground truth pose at each frame with a random rigid transformation, so to mimic the noisy pose from the previous frame. Then, for each iteration, we plot in Fig. 6 the average error in rotation and translation as well as the average distance from of the model's vertices transformed using the estimated pose to vertices transformed using the ground truth.

In this section, we evaluate and compare three learning strategies from our method – generalized model-based learning in Sec. 4.1, subject-specific model-based learning in Sec. 4.2 and the combined generalized tracker with subject-specific online learning in Sec. 4.3. Individually, our tracking approaches achieve better accuracy and efficiency than the state-of-the-art methods [7, 14, 15, 17]. Furthermore, the qualitative results in Fig. 7 demonstrate the robustness and the range of applicability in real scenarios.

For the generalized tracker, the geodesic grid is constructed with 642 vertices, where 332 of them are used after filtering through the azimuth and polar angles in the spherical coordinate system. In each camera view, we use $n_r = 2500$ random transformation for each subject and $n_h = 20$ points on the model to learn one tree per parameter with a maximum depth of 20 and maximum standard deviation of 0.1. The same learning parameters are used for the two subject-specific learning methods. Moreover, following Sec. 3.1, the tracker is initialized using OpenCV's face detector [2] which takes approximately 9.1 ms per frame. All evaluations are conducted using Intel(R) Core(TM) i7-3820QM CPU with 16 GB RAM.

4.1. Generalized model-based tracking

To evaluate the generalizability of our algorithm, we use the same experimental framework as [7], where we learn using 18 subjects and evaluate on the remaining two. However, our learning scheme relies purely on synthetic data without using the acquired real RGB-D images. In effect, we render depth images through the given 3D model of the head from the 18 subjects in the dataset. Subsequently, we compare our results with [7, 14, 15, 17], where all competing approaches use real images in the database for learning.

Robustness. Table 1 emphasizes that our results decreases the errors in rotation and translation of the state of the art by half. In addition, we achieve better results across all parameters without any missed pose estimation. These results entail a 100% success rate with a threshold of 20 mm for translation and 20° for rotation in Table 2. Our method improves the best results [15] by a 5% increase in transla-

	HF [7]	PLEV [14]	ARF* [17]	HN [15]	*Generalized*	*Subj.-Specific*	*Combined*
Translation (*mm*)	12.8 ± 6.8	7.2 ± 12.1	10.8 ± 6.9	8.1 ± 5.3	**4.2 ± 1.8**	2.1 ± 1.0	5.4 ± 2.3
X	6.9 ± 6.7	–	5.5 ± 5.6	3.8 ± 4.5	**1.3 ± 0.9**	0.9 ± 0.7	2.3 ± 1.9
Y	7.4 ± 5.6	–	6.2 ± 6.1	4.6 ± 4.0	**3.1 ± 2.1**	1.4 ± 1.1	2.8 ± 2.1
Z	4.7 ± 3.4	–	4.1 ± 3.1	3.7 ± 3.0	**1.8 ± 1.2**	0.9 ± 0.6	2.7 ± 2.4
Rotation (*degrees*)	14.3 ± 10.0	7.3 ± 5.9	12.2 ± 9.0	9.8 ± 8.0	**3.2 ± 1.6**	1.9 ± 1.1	4.2 ± 2.7
Yaw	5.7 ± 6.1	4.1 ± 6.9	5.5 ± 5.5	3.8 ± 3.7	**1.9 ± 1.6**	1.3 ± 1.2	2.0 ± 1.6
Pitch	9.7 ± 8.9	3.9 ± 4.0	7.8 ± 7.9	6.7 ± 6.6	**2.3 ± 1.7**	1.0 ± 0.9	2.7 ± 2.5
Roll	5.9 ± 5.2	3.2 ± 3.0	5.0 ± 4.4	4.3 ± 4.9	**1.0 ± 1.0**	1.0 ± 1.0	2.1 ± 2.2
Missed	5.0%	5.0%	3.0%	1.0%	**0.0%**	0.0%	0.0%
Runtime (*ms*)	17.8	–	–	66.7–100.0	**1.8**	1.4	26.9 + 1.2

Table 1. Based on the Biwi Kinect Head Pose Database [7], comparison of the mean and standard deviation of the errors in translation and the rotation angles, the percentage of missed pose estimation, where the error of the translation is above 20 mm for [14] or 50 mm for [7, 15, 17], and runtime per frame. We compare our *generalized* model-based learning method, *subject-specific* model-based learning method, and *combined* generalized model-based and subject-specific online learning method with other existing approaches [7, 14, 15, 17].

	Trans. (20 mm)	Rotation (20°)
HF [7]	82.99%	73.29%
PLEV [14]	95.00%	–
ARF* [17]	88.30%	80.37%
HN [15]	95.11%	88.86%
Generalized	**100.00%**	**100.00%**
Subj.-Specific	100.00%	100.00%
Combined	100.00%	100.00%

Table 2. Success rate of different methods where the error in translation is less than 20 mm or the rotation angle is less than 20°. The list of competing methods are the same as Table 1.

tion and 11% increase in rotation. Considering that other works achieve a maximum success rate of 95.11% in Table 2, Fig. 5(a) shows that we reached 95.2% with a threshold of 8 mm in translation and 97.7% with a threshold of 7° in rotation. Throughout the evaluations, based on Fig. 6, ten iterations guarantee that the error values have converged.

Efficiency. Our algorithm runs at 1.8 ms per frame with a single core CPU and a memory consumption of 46.0 MB. Compared to the other works [7, 15] that runs in real-time in Table 1, this is approximately one order of magnitude lower.

It is noteworthy to mention that other works [7, 15] run in real-time because the scene in the dataset is almost empty: each frame includes one person, visualized from head to torso. Since they employ a tracking-by-detection framework that jointly estimates the pose and segments the head in every frame independently, their runtime increases when the scene is not controlled and the detector is required to distinguish the face from the surrounding objects, such as a scene where a person is sitting on a couch or holding tools. Nonetheless, since we are utilizing a temporal tracker, our runtime is constant and is independent of the surrounding objects in the scene.

4.2. Subject-specific model-based tracking

Instead of using a generalized model as in Sec. 4.1, we also learn using the subject-specific CAD models and compare with the other results. As a consequence, the subject-specific learning scheme generates the least error in Table 1, while maintaining the 100% success rate in Table 2 and achieving the best success rate for individual parameters in Fig. 5(b). Furthermore, it continues converging to a smaller error after four iterations in Fig. 6. In terms of efficiency, it requires approximately 3.5 MB in memory and tracks 1.4 ms per frame with a single core.

4.3. Subject-specific online learning

The superiority of the subject-specific learning across all evaluations in Sec. 4.2 demonstrates the value of tracking unique structures that is particular to the subject. However, the learning methods of both Sec. 4.1 and Sec. 4.2 are done offline. It follows that the reconstruction of the CAD models for individual subjects is done beforehand but requiring an a priori knowledge of the subject hinders its application to directly track the head of any user in the camera's field of view. Hence, the subject-specific model-based learning method is time-consuming and limited in real applications.

One of the main novelties of our work is to incorporate subject-specific structures through online learning by incrementally accumulating trees from different viewpoints of the subject's head. Using the generalized method from Sec. 4.1 to track, we take the depth image of the current frame and the tracked pose to learn new trees, which are added into the existing forest. In this way, we continuously track and learn the head structure to increase the number of subject-specific trees in the forest.

Standard dataset. We also evaluate the combined generalized tracker with the subject-specific online learning in Table 1. Our combined method performs slightly worse

Figure 7. These video sequences are taken (a) while observing a Skype conversation from multiple subjects and (b) while the subject wears a mask and moves around. These illustrate occlusions from (a) hand gestures and (b) external objects. In addition, (b) shows the robustness of the pose estimation in extreme poses.

than the model-based generalized tracker with 1.2 mm more error in translation and 1.0° more in rotation. But they have the same performance with regards to the success rate in Table 2 and Fig. 5(c), as well as the convergence rate in Fig. 6. Evidently, the reason for having a slightly higher error value of the combined method than the generalized model-based tracker is because the online learning is initially guided by the generalized tracker. This implies that the resulting tracker cannot have less error than the generalized tracker. However, Table 1 also highlights that the combined method outperforms any of the other competing works [7, 14, 15, 17].

In terms of efficiency, the forest consumes the same initial memory as the generalized method with an additional 13 KB for the trees associated to each of the learned image. With 8 CPU cores, it runs at 26.9 ms for online learning and 1.2 ms for tracking. It is noteworthy to mention that not all tracked frames are used for learning new trees, and the geodesic grid limits the number of trees for the viewpoint that are close to each vertex.

Limitation of the generalized model. Although the error of the subject-specific online learning in Sec. 4.3 is slightly higher compared to the generalized model-based tracker, its advantage is demonstrated when running real-world applications as shown in Fig. 7 and the *Supplementary Materials*.

Considering that there are no ground truth poses for the two video sequences in Fig. 7, we compare the tracker with and without the online learning by computing the percent of frames where the algorithm detects failure and requires re-initialization as discussed in Sec. 3.4. The generalized tracker detects 53.5% of the frames as failure in Fig. 7(a) and 42.5% in Fig. 7(b). Conversely, the combined method does not detect any failure. This is because, as opposed to the dataset of [7], the users are no longer constrained and can move freely within the camera's field of view. Here, occlusions on the face from hand gestures in Fig. 7(a) are more common, which affect the robustness of the generalized tracker. Basically, the generalized model-based tracking learns the facial structure while online learning learns the entire head structure. This makes the generalized model more sensitive to facial deformations such as talking as well as facial occlusions.

Another limitation of the generalized tracker is the extreme poses, such as Fig. 7(b), which are excluded in learning as shown in Fig. 4(c). Nonetheless, the online learner adapts the forest to handle the extreme poses, where the substantial structures of the facial features are no longer visible.

5. Conclusion

Given a video sequence of RGB-D images, we address the problem of head pose estimation, where the RGB image is used in the first frame to detect the face [21] and initialize the tracker, while the depth image is used to temporally track the head pose throughout the sequence using a Random Forest algorithm.

This paper highlights three types of tracking – the generalized model-based tracking, the subject-specific model-based tracking as well as the combined generalized tracker with the subject-specific online learning. Individually, they achieve state-of-the-art results on the standard dataset [7] with higher success rates and lower error values compared to other methods [7, 14, 15, 17].

The choice of using one of the trackers depends on the application at hand. Notably, the difference between the model-based learning and online learning is that: (1) the former tracks the common facial structure while the latter tracks the head; and, (2) the former has a unified reference coordinate system while the latter has local coordinate system that varies from one tracker to the rest. Therefore, for a controlled environment where the face is visible such as the dataset in [7], the generalized model-based tracker is sufficient; and, if the subject's head model is given, the subject-specific model-based tracker is the best option. However, for a wider range of applications where the subjects occlude their faces with hand gestures or move freely around the camera's field of view as shown in Fig. 7, the choice of using the combined method becomes necessary due to its robustness to handle large occlusions and extreme poses.

References

[1] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Incremental face alignment in the wild. In *Conference on Computer Vision and Pattern Recognition*, 2014. 2

[2] G. Bradski. The opencv library. *Doctor Dobbs Journal*, 2000. 6

[3] L. Breiman. Random forests. *Machine learning*, 2001. 3, 4

[4] M. D. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister. Real-time face pose estimation from single range images. In *Conference on Computer Vision and Pattern Recognition*, 2008. 2

[5] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001. 2

[6] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool. Real-time facial feature detection using conditional regression forests. In *Conference on Computer Vision and Pattern Recognition*, 2012. 2

[7] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool. Random forests for real time 3d face analysis. *International Journal of Computer Vision*, 2013. 1, 2, 3, 4, 6, 7, 8

[8] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *Conference on Computer Vision and Pattern Recognition*, 2011. 2

[9] X. Geng and Y. Xia. Head pose estimation based on multivariate label distribution. In *Conference on Computer Vision and Pattern Recognition*, 2014. 2

[10] M. Kan, S. Shan, H. Chang, and X. Chen. Stacked progressive auto-encoders (spae) for face recognition across poses. In *Conference on Computer Vision and Pattern Recognition*, 2014. 2

[11] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Conference on Computer Vision and Pattern Recognition*, 2014. 2

[12] S. Koterba, S. Baker, I. Matthews, C. Hu, J. Xiao, J. Cohn, and T. Kanade. Multi-view aam fitting and camera calibration. In *International Conference on Computer Vision*, 2005. 2

[13] E. Murphy-Chutorian and M. M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 2

[14] C. Redondo-Cabrera, R. Lopez-Sastre, and T. 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. 1, 2, 3, 6, 7, 8

[15] G. Riegler, D. Ferstl, M. Rüther, and H. Bischof. Hough networks for head pose estimation and facial feature localization. In *British Machine Vision Conference*, 2014. 1, 2, 3, 6, 7, 8

[16] R. Sadourny, A. Arakawa, and Y. Mintz. Integration of the nondivergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere 1. *Monthly Weather Review*, 1968. 4

[17] S. Schulter, C. Leistner, P. Wohlhart, P. M. Roth, and H. Bischof. Alternating regression forests for object detection and pose estimation. In *International Conference on Computer Vision*, 2013. 1, 2, 3, 6, 7, 8

[18] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition*, 2014. 2

[19] D. J. Tan and S. Ilic. Multi-forest tracker: A chameleon in tracking. In *Conference on Computer Vision and Pattern Recognition*, 2014. 1, 4, 5

[20] D. J. Tan, F. Tombari, S. Ilic, and N. Navab. A versatile learning-based 3d temporal tracker: Scalable, robust, online. In *International Conference on Computer Vision*, 2015. 1, 4, 5

[21] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 2004. 2, 4, 8