myjournal manuscript No.
(will be inserted by the editor)

Random forests for real time 3D face analysis

Gabriele Fanelli · Matthias Dantone · Juergen Gall · Andrea Fossati · Luc Van Gool

Received: date / Accepted: date

Abstract We present a random forest-based framework for real time head pose estimation from depth images and extend it to localize a set of facial features in 3D. Our algorithm takes a voting approach, where each patch extracted from the depth image can directly cast a vote for the head pose or each of the facial features. Our system proves capable of handling large rotations, partial occlusions, and the noisy depth data acquired using commercial sensors. Moreover, the algorithm works on each frame independently and achieves real time performance without resorting to parallel computations on a GPU. We present extensive experiments on publicly available, challenging datasets and present a new annotated head pose database recorded using a Microsoft Kinect.

Keywords Random forests · Head pose estimation · 3D facial features detection · Real time

Gabriele Fanelli · Matthias Dantone · Juergen Gall · Andrea Fossati · Luc Van Gool
Computer Vision Laboratory, ETH Zurich,
Sternwartstrasse 7, 8092 Zurich, Switzerland
E-mail: fanelli@vision.ee.ethz.ch

Matthias Dantone
E-mail: dantone@vision.ee.ethz.ch

Juergen Gall
Perceiving Systems Department, Max Planck Institute for Intelligent Systems,
Spemannstrasse 41, 72076 Tübingen, Germany
E-mail: juergen.gall@tue.mpg.de

Andrea Fossati
E-mail: fossati@vision.ee.ethz.ch

Luc Van Gool
Department of Electrical Engineering/IBBT, K.U. Leuven,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
E-mail: luc.vangool@esat.kuleuven.be

1 Introduction

Despite recent advances, people still interact with machines through devices like keyboards and mice, which are not part of natural human-human communication. As people interact by means of many channels, including body posture and facial expressions, an important step towards more natural interfaces is the visual analysis of the user's movements by the machine. Besides the interpretation of full body movements, as done by systems like the Kinect for gaming, new interfaces would highly benefit from automatic analysis of facial movements, as addressed in this paper.

Recent work has mainly focused on the analysis of standard images or videos; see the survey of [47] for an overview of head pose estimation from video. The use of 2D imagery is very challenging though, not least because of the lack of texture in some facial regions. On the other hand, depth-sensing devices have recently become affordable (e.g., Microsoft Kinect or Asus Xtion) and in some cases also accurate (e.g., [66]).

The newly available depth cue is key for solving many of the problems inherent to 2D video data. Yet, 3D imagery has mainly been leveraged for face tracking [6,11,67,65], often leaving open issues of drift and (re-)initialization. Tracking-by-detection, on the other hand, detects the face or its features in each frame, thereby providing increased robustness.

A typical approach to 3D head pose estimation involves localizing a specific facial feature point (e.g., one not affected by facial deformations like the nose) and determining the head orientation (e.g., as Euler angles). When 3D data is used, most methods rely on geometry to localize prominent facial points like the nose tip [39, 12,60,10,9] and thus becoming sensitive to its occlusion. Moreover, the available algorithms are either not

real time, rely on some assumption for initialization like starting with a frontal pose, or cannot handle large rotations.

We introduce a voting framework where patches extracted from the whole depth image can contribute to the estimation task. As in the Implicit Shape Model [36], the intuition is that patches belonging to different parts of the image contain valuable information on global properties of the object which generated it, like pose. Since all patches can vote for the localization of a specific point of the object, it can be detected even when that particular point is occluded.

We propose to use random regression forests for real time head pose estimation and facial feature localization from depth images. Random forests [7] (RFs) have been successful in semantic segmentation [58], keypoint recognition [37], object detection [28,29], action recognition [70,29], and real time human pose estimation [57,30]. They are well suited for time-critical applications, being very fast at both train and test time, lend themselves to parallelization [56], and are inherently multi-class. The proposed method does not rely on specific hardware and can easily trade-off accuracy for speed. We estimate the desired, continuous parameters directly from the depth data, through a learnt mapping from depth to parameter values. Our system works in real time, without manual initialization. In our experiments, we show that it also works for unseen faces and that it can handle large pose changes, variations in facial hair, and partial occlusions due to glasses, hands, or missing parts in the 3D reconstruction. It does not rely on specific features like the nose tip.

Random forests show their power when using large datasets, on which they can be trained efficiently. Because the accuracy of a regressor depends on the amount of annotated training data, the acquisition and labeling of a training set are key issues. Depending on the expected scenario, we either synthetically generate annotated depth images by rendering a face model undergoing large rotations, or record real sequences using a consumer depth sensor, automatically annotating them using state-of-the-art tracking methods.

A preliminary version of this work was published in [25], where we introduced the use of random regression forests for real time head pose estimation from high quality range scans. In [26], we extended the forest to cope with depth images where the whole body can be visible, i.e., discriminating depth patches that belong to a head and only using those to predict the pose, jointly solving the classification and regression problems involved. In this work, we provide a thorough experimental evaluation and extend the random forest to localize several facial landmarks on the range scans.

2 Related work

After a brief overview of the random forest literature, we present an analysis of related works on head pose estimation and facial features localization.

2.1 Random Forests

Random forests [7] have become a popular method in computer vision [28,29,57,30] given their capability to handle large training datasets, their high generalization power and speed, and the relative ease of implementation. Recent works showed the power of random forests in mapping image features to votes in a generalized Hough space [28] or, in a regression framework, to real-valued functions [19]. In the context of real time pose estimation, multi-class random forests have been proposed for the real time determination of head pose from 2D video data [32]. In [57], random forests have been used for real time body pose estimation from depth data. Each input depth pixel is first assigned to a specific body part, using a classification forest trained on a very large synthetic dataset. After this step, the location of the body joints is inferred through a local mode-finding approach based on mean shift. In [30], it has been shown that the body pose can be more efficiently estimated by using regression instead of classification forests. Inspired by the works [28,19], we have shown that regression forests can be used for real time head pose estimation from depth data [25,26].

A detailed introduction to decision forests and their applications in computer vision can be found in [18].

2.2 Head pose estimation

With application ranging from image normalization for recognition to driver drowsiness detection, automatic head pose estimation is an important problem. Several approaches have been proposed in the literature [47]; before introducing 3D approaches, which are more relevant for this paper, we present a brief overview of works that take 2D images as input. Methods based on 2D images can be subdivided into appearance-based and feature-based classes, depending on whether they analyze the face as a whole or instead rely on the localization of some specific facial features.

2D Appearance-based methods. These methods usually discretize the head pose space and learn separate detectors for subsets of poses [33]. Chen et al. [13] and Balasubramanian et al. [2] present head pose estimation systems with a specific focus on the mapping from the high-dimensional space of facial appearance

to the lower-dimensional manifold of head poses. The latter paper considers face images with varying poses as lying on a smooth low-dimensional manifold in a high-dimensional feature space. The proposed Biased Manifold Embedding uses the pose angle information of the face images to compute a biased neighborhood of each point in the feature space, prior to determining the low-dimensional embedding. In the same vein, Osadchy et al. [50] instead use a convolutional network to learn the mapping, achieving real time performance for the face detection problem, while also providing an estimate of the head pose. A very popular family of methods use statistical models of the face shape and appearance, like Active Appearance Models (AAMs) [16], multi-view AAMs [53], and 3D Morphable Models [5, 59]. Such methods usually focus on tracking facial features rather than estimating the head pose, however. In this context, the authors of [40] coupled an Active Appearance Model with the POSIT algorithm for head pose tracking.

2D Feature-based methods. These methods rely on some specific facial features to be visible, and therefore are sensitive to occlusions and to large head rotations. Vatahska et al. [62] use a face detector to roughly classify the pose as frontal, left, or right profile. After this, they detect the eyes and nose tip using AdaBoost classifiers, and the detections are fed into a neural network which estimates the head orientation. Similarly, Whitehill et al. [69] present a discriminative approach to frame-by-frame head pose estimation. Their algorithm relies on the detection of the nose tip and both eyes, thereby limiting the recognizable poses to the ones where both eyes are visible. Morency et al. [45] propose a probabilistic framework called Generalized Adaptive View-based Appearance Model integrating frame-by-frame head pose estimation, differential registration, and keyframe tracking.

3D methods. In general, approaches relying solely on 2D images are sensitive to illumination changes and lack of distinctive features. Moreover, the annotation of head poses from 2D images is intrinsically problematic. Since 3D sensing devices have become available, computer vision researchers have started to leverage the additional depth information for solving some of the inherent limitations of image-based methods. Some of the recent works thus use depth as primary cue [10] or in addition to 2D images [11, 44, 54].

Seemann et al. [54] presented a neural network-based system fusing skin color histograms and depth information. It tracks at 10 fps but requires the face to be detected in a frontal pose in the first frame of the sequence. The approach in [43] uses head pose estimation only as a pre-processing step to face recognition, and the low reported average errors are only calculated on faces of subjects that belong to the training set. Still in a tracking framework, Morency et al. [44] use instead a intensity and depth input image to build a prior model of the face using 3D view-based eigenspaces. Then, they use this model to compute the absolute difference in pose for each new frame. The pose range is limited and manual cropping is necessary. In [11], a 3D face model is aligned to an RGB-depth input stream for tracking features across frames, taking into account the very noisy nature of depth measurements coming from commercial sensors.

Considering instead pure detectors on a frame-by-frame basis, Lu and Jain [39] create hypotheses for the nose position in range images based on directional maxima. For verification, they compute the nose profile using PCA and a curvature-based shape index. Breitenstein et al. [10] presented a real time system working on range scans provided by the scanner of [66]. Their system can handle large pose variations, facial expressions, and partial occlusions, as long as the nose remains visible. The method relies on several candidate nose positions, suggested by a geometric descriptor. Such hypotheses are all evaluated in parallel on a GPU, which compares them to renderings of a generic template with different orientations, finally selecting the orientation which minimizes a predefined cost function. Real-time performance is only met thanks to the parallel GPU computations. Unfortunately, GPUs are power-hungry and might not be available in many scenarios where portability is important, e.g., for mobile robots. Breitenstein et al. also collected a dataset of over 10K annotated range scans of heads. The subjects, both male and female, with and without glasses, were recorded using the scanner of [66] while turning their heads around, trying to span all possible yaw and pitch rotation angles they could. The scans were automatically annotated, tracking each sequence using ICP in combination with a personalized face template. The same authors also extended their system to use lower quality depth images from a stereo system [9]. Yet, the main shortcomings of the original method remain.

2.3 Facial features localization

2D Facial Features. Facial feature detection from standard images is a well studied problem, often performed as preprocessing for face recognition. Previous contributions can be classified into two categories, depending on whether they use global or local features. Holistic methods, e.g., Active Appearance Models [16, 17, 41], use the entire facial texture to fit a generative model to a test image. They are usually affected by

lighting changes and a bias towards the average face. The complexity of the modeling is an additional issue. Moreover, these methods perform poorly on unseen identities [31] and cannot handle low-resolution images well.

In recent years, there has been a shift towards methods based on independent local feature detectors [61, 1,3]. Such detectors are discriminative models of image patches centered around the facial landmarks, often ambiguous because the limited support region cannot cope with the large appearance variations present in the training samples. To improve accuracy and reduce the influence of wrong detections, global models of the facial features configuration like pictorial structures [27, 23] or Active Shape Models [20] are needed.

3D Facial Features. Similar to the 2D case, methods focusing on facial feature localization from range data can be subdivided into categories using global or local information. Among the former class, the authors of [46] deform a bi-linear face model to match a scan of an unseen face in different expressions. Yet, the paper's focus is not on the localization of facial feature points and real time performance is not achieved. Also Kakadiaris et al. [35] non-rigidly align an annotated model to face meshes. Constraints need to be imposed on the initial face orientation, however. Using high quality range scans, Weise et al. [67] presented a real time system, capable of tracking facial motion in detail, but using personalized templates. The same approach has been extended to robustly track head pose and facial deformations using RGB-depth streams provided by commercial sensors like the Kinect [65].

Most works that try to directly localize specific feature points from 3D data take advantage of surface curvatures. For example, the authors of [60,55,12] all use curvature to roughly localize the inner corners of the eyes. Such an approach is very sensitive to missing depth data, particularly for the regions around the inner eye corners, frequently occluded by shadows. Also Mehryar et al. [42] use surface curvatures by first extracting ridge and valley points, which are then clustered. The clusters are refined using a geometric model imposing a set of distance and angle constraints on the arrangement of candidate landmarks. Colbry et al. [15] use curvature in conjunction with the Shape Index proposed by [22] to locate facial feature points from range scans of faces. The reported execution time of this anchor point detector is 15 sec per frame. Wang et al. [64] use point signatures [14] and Gabor filters to detect some facial feature points from 3D and 2D data. The method needs all desired landmarks to be visible, thus restricting the range of head poses while being sensitive to occlusions. Yu et al. [72] use genetic algorithms to combine several weak classifiers into a 3D facial landmark detector. Ju et al. [34] detect the nose tip and the eyes using binary neural networks, and propose a 3D shape descriptor invariant to pose and expression.

The authors of [73] propose a 3D Statistical Facial Feature Model (SFAM), which models both the global variations in the morphology of the face and the local structures around the landmarks. The low reported errors for the localization of 15 points in scans of neutral faces come at the expense of processing time: over 10 minutes are needed to process one facial scan. In [48], fitting the proposed PCA shape model containing only the upper facial features, i.e., without the mouth, takes on average 2 minutes per face.

In general, prior work on facial feature localization from 3D data is either sensitive to occlusions, especially of the nose, requires prior knowledge of feature map thresholds, cannot handle large rotations, or does not run in real time.

3 Random forest framework for face analysis

In Section 3.1 we first summarize a general random forest framework [7], then give specific details for face analysis based on depth data in Sections 3.2 and 3.3.

3.1 Random forest

Decision trees [8] can map complex input spaces into simpler, discrete or continuous output spaces, depending on whether they are used for classification of regression purposes. A tree splits the original problem into smaller ones, solvable with simple predictors, thus achieving complex, highly non-linear mappings in a very simple manner. A non-leaf node in the tree contains a binary test, guiding a data sample towards the left or the right child node. The tests are chosen in a supervised-learning framework, and training a tree boils down to selecting the tests which cluster the training such as to allow good predictions using simple models.

Random forests are collections of decision trees, each trained on a randomly sampled subset of the available data; this reduces over-fitting in comparison to trees trained on the whole dataset, as shown by Breiman [7]. Randomness is introduced by the subset of training examples provided to each tree, but also by a random subset of tests available for optimization at each node.

Fig. 1 illustrates a random regression forest mapping feature patches extracted from a depth image to a distribution stored at each leaf. In our framework, these distributions model the head orientation (Section 3.2)

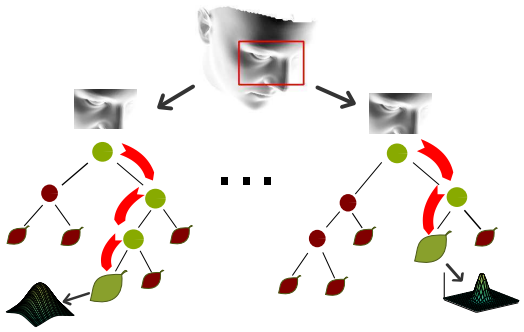


Fig. 1 Example of regression forest for head pose estimation. For each tree, the tests at the non-leaf nodes direct an input sample towards a leaf, where a real-valued, multivariate distribution of the output parameters is stored. The forest combines the results of all leaves to produce a probabilistic prediction in the real-valued output space.

or locations of facial features (Section 3.3). In the following, we outline the general training approach of a random forest and give the application specific details in Sections 3.2 and 3.3.

A tree T in a forest $\mathcal{T} = \{T_t\}$ is built from a set of annotated patches, randomly extracted from the training images: $\mathcal{P} = \{P_i\}$, where $\mathcal{I}_i$ is the appearance of the patch. Starting from the root, each tree is built recursively by assigning a binary test $\phi(\mathcal{I}) \rightarrow \{0, 1\}$ to each non-leaf node. Such test sends each patch (according to its appearance) either to the left or right child, in this way, the training patches $\mathcal{P}$ arriving at the node are split into two sets, $\mathcal{P}_L(\phi)$ and $\mathcal{P}_R(\phi)$.

The best test ϕ^* is chosen from a pool of randomly generated ones ($\{\phi\}$): all patches arriving at the node are evaluated by all tests in the pool and a predefined information gain of the split $IG(\phi)$ is maximized:

$$\phi^* = \arg\max_{\phi} IG(\phi) \tag{1}$$

$$IG(\phi) = \mathcal{H}(\mathcal{P}) - \sum_{i \in \{L,R\}} w_i \mathcal{H}(\mathcal{P}_i(\phi)), \tag{2}$$

where $w_i = \frac{|\mathcal{P}_i(\phi)|}{|\mathcal{P}|}$ is the ratio of patches sent to each child node and $\mathcal{H}(\mathcal{P})$ is a measure of the patch cluster $\mathcal{P}$, usually related to the entropy of the clusters' labels. The measure $\mathcal{H}(\mathcal{P})$ can have different forms, depending on whether the goal of the forest is regression, classification, or rather a combination of the two. The measures that are relevant for face analysis are discussed in Sections 3.2 and 3.3. The process continues with the left and the right child using the corresponding training sets $\mathcal{P}_L(\phi^*)$ and $\mathcal{P}_R(\phi^*)$ until a leaf is created when either the maximum tree depth is reached, or less than a minimum number of training samples are left.

In order to employ such a random forest framework for face analysis from depth data, we have to

- acquire annotated training data $\mathcal{P}$,
- define binary tests ϕ,
- define a measure $\mathcal{H}(\mathcal{P})$,
- define a distribution model to be stored at the leaves.

These issues are discussed in the following sections.

3.2 Head pose estimation

3.2.1 Training data

Building a forest is a supervised learning problem, i.e., training data needs to be annotated with labels on the desired output space. In our head pose estimation setup, a training sample is a depth image containing a head, annotated with the 3D locations of a specific point, i.e., the tip of the nose, and the head orientation. Fix-sized patches are extracted from a training image, each annotated with two real-valued vectors: While $\boldsymbol{\theta}^1 = \{\theta_x, \theta_y, \theta_z\}$ is the offset computed between the 3D point falling at the patch center and the nose tip, the head orientation is encoded as Euler angles, $\boldsymbol{\theta}^2 = \{\theta_{ya}, \theta_{pi}, \theta_{ro}\}$. In order for the forest to be more scale-invariant, the size of the patches can be made dependent on the depth (e.g., at its center), however, in this work we assume the faces to be within a relatively narrow range of distances from the sensor.

In order to deal with background like hair and other body parts, fixed-sized patches are not only sampled from faces but also from regions around them. A class label c_i is thus assigned to each patch P_i, where $c_i = 1$ if it is sampled from the face and 0 otherwise. The set of training patches is therefore given by $\mathcal{P} = \{\mathcal{P}_i = (\mathcal{I}_i, c_i, \boldsymbol{\theta}_i)\}$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}^1, \boldsymbol{\theta}^2)$. $\mathcal{I}_i$ represents the image features $\mathcal{I}_i^f$ computed from a patch P_i. Such features include the original depth values plus, optionally, the geometric normals, i.e., for a depth pixel $d(u, v)$, the average of the normals of the planes passing through $d(u, v)$ and pairs of its 4-connected neighbors. The x, y, and z coordinates of the normals are treated as separate feature channels.

3.2.2 Binary tests

Our binary tests $\phi_{f,F_1,F_2,\tau}(\mathcal{I})$ are defined as:

$$|F_1|^{-1} \sum_{\boldsymbol{q} \in F_1} \mathcal{I}^f(\boldsymbol{q}) - |F_2|^{-1} \sum_{\boldsymbol{q} \in F_2} \mathcal{I}^f(\boldsymbol{q}) > \tau, \tag{3}$$

where f is the feature channel's index, F_1 and F_2 are two asymmetric rectangles defined within the patch, and τ is a threshold. We use the difference between the average values of two rectangular areas as in [25,19], rather than single pixel differences as in [29] in order to

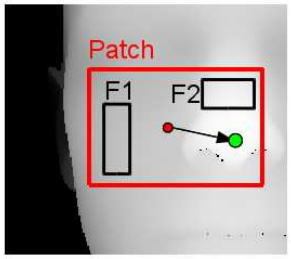


Fig. 2 Example of a training patch (larger, red rectangle) with its associated offset vector (arrow) between the 3D point falling at the patch's center (red dot) and the ground truth location of the nose (marked in green). The two rectangles F1 and F2 represent a possible choice for the regions over which to compute a binary test, i.e., the difference between the average values computed over the two boxes.

be less sensitive to noise; the additional computation is negligible when integral images [63] are used. Tests defined as Equation (3) represent a generalization of the widely-used Haar-like features [51]. An example test is shown in Figure 2: A patch is marked in red, containing the two regions F_1 and F_2 defining the test (in black); the arrow represents the 3D offset vector ($\boldsymbol{\theta}^1$) between the 3D patch center (in red) and the ground truth location of a feature point, the nose tip in this case (green).

3.2.3 Goodness of split

A regression forest can be applied to head pose estimation from depth images containing only faces [25]; in this case, all training patches are positive ($c_i = 1\ \forall i$) and the measure $\mathcal{H}(\mathcal{P})$ is defined as the entropy of the continuous patch labels. Assuming $\boldsymbol{\theta}^n$, where $n \in \{1, 2\}$, to be realizations of 3-variate Gaussians, we can represent the labels in a set $\mathcal{P}$ as $p(\boldsymbol{\theta}^n) = \mathcal{N}(\boldsymbol{\theta}^n; \overline{\boldsymbol{\theta}^n}, \boldsymbol{\Sigma}^n)$, and thus compute the differential entropy $H(\mathcal{P})^n$ for n:

$$H(\mathcal{P})^n = \frac{1}{2}\log\Big((2\pi e)^3\,|\boldsymbol{\Sigma}^n|\Big). \tag{4}$$

We thus define the regression measure:

$$\mathcal{H}_r(\mathcal{P}) = \sum_n \log(|\boldsymbol{\Sigma}^n|) \propto \sum_n H(\mathcal{P})^n. \tag{5}$$

Substituting Equation (5) into Equation (2) and maximizing it actually favors splits which minimize the covariances of the Gaussian distributions computed over all label vectors $\boldsymbol{\theta}^n$ at the children nodes, thus intuitively decreasing the regression uncertainty.

Goal of the forest, however, is not only to map image patches into probabilistic votes in a continuous space, but, as in [26], also to decide which patches are actually allowed to cast such votes. In order to include a measure of the classification uncertainty in the information gain defined by Equation (2), we use the measure $\mathcal{H}_c(\mathcal{P})$ of the cluster's class uncertainty, defined as the entropy:

$$\mathcal{H}_c(\mathcal{P}) = -\sum_{k=0}^{K} p(c=k|\mathcal{P}) \log\left(p(c=k|\mathcal{P})\right), \tag{6}$$

where $K = 1$. The class probability $p(c=k|\mathcal{P})$ is approximated by the ratio of patches with class label k in the set $\mathcal{P}$.

The two measures (5) and (6) can be combined in different ways. One approach, as in the work of Gall et al. [29], is to randomly select one or the other at each node of the trees, denoted in the following as the *interleaved* method. A second approach (*linear*) was proposed by Okada [49], i.e., a weighted sum of the two measures:

$$\mathcal{H}_c(\mathcal{P}) + \alpha \max\Big(p\big(c=1|\,\mathcal{P}\big) - t_p, 0\Big)\mathcal{H}_r(\mathcal{P}). \tag{7}$$

When minimizing (7), the optimization is steered by the classification term alone until the purity of positive patches reaches the activation threshold t_p. From that point on, the regression term starts to play an ever important role, weighted by the constant α, until the purity reaches 1. In this case, $\mathcal{H}_c = 0$ and the optimization is driven only by the regression measure $\mathcal{H}_r$.

In [26], we proposed a third approach, where the two measures are weighted by an *exponential* function of the depth:

$$\mathcal{H}_c(\mathcal{P}) + (1 - e^{-\frac{d}{\lambda}})\mathcal{H}_r(\mathcal{P}), \tag{8}$$

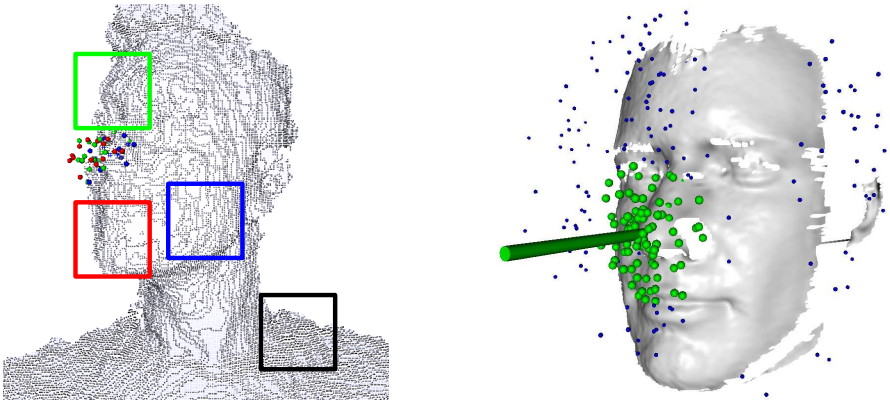
where d is the depth of the node in the tree. In this way, the regression measure is given increasingly higher weight as we descend deeper in the tree towards the leaves, with the parameter λ specifying the steepness of the change.

Note that, when only positive patches are available, $\mathcal{H}_c = 0$, i.e., Equations (7) and (8) are both proportional to the regression measure $\mathcal{H}_r$ alone, and both lead to the same selected test ϕ^*, according to Equation (1).

In our experiments (see Section 4.2.2), we evaluate the three possibilities for combining the classification measure $\mathcal{H}_c$ and the regression measure $\mathcal{H}_r$ for training.

3.2.4 Leaves

For each leaf, the class probabilities $p\big(c=k|\,\mathcal{P}\big)$ and the distributions of the continuous head pose parameters $p(\boldsymbol{\theta}^1) = \mathcal{N}(\boldsymbol{\theta}^1; \overline{\boldsymbol{\theta}^1}, \boldsymbol{\Sigma}^1)$ and $p(\boldsymbol{\theta}^2) = \mathcal{N}(\boldsymbol{\theta}^2; \overline{\boldsymbol{\theta}^2}, \boldsymbol{\Sigma}^2)$ are stored. The distributions are estimated from the training patches that arrive at the leaf and are used for estimating the head pose as explained in the following section.



(a) (b)

Fig. 3 (a) Example votes, casted by different patches. The green, red, and blue patch are classified as positives and therefore cast votes for the nose position (correspondingly colored spheres). On the other hand, the black patch at the shoulder is classified as negative and does not vote. (b) Example test image: the green spheres represent the votes selected after outliers (blue spheres) are filtered out by mean shift. The large green cylinder stretches from the final estimate of the nose center in the estimated face direction.

3.2.5 Testing

When presented with a test depth image, patches are densely sampled from the whole image and sent down through all trees in the forest. Each patch is guided by the binary tests stored at the nodes, as illustrated in Fig. 1. A stride parameter controls how densely patches are extracted, thus easily steering speed and accuracy of the regression.

The probability $p\big(c = k|\,\mathcal{P}\big)$ stored at the leaf judges how informative the test patch is for class k. This probability value tells whether the patch belongs to the head or other body parts. Since collecting all relevant negative examples is harder than collecting many positive examples, we only consider leaves with $p\big(c = k|\,\mathcal{P}\big) = 1$. For efficiency and accuracy reasons, we also filter out leaves with a high variance, which are less informative for the regression, i.e., all leaves with $\mathrm{tr}\left(\Sigma^1\right)$ greater than a threshold max_v. The currently employed threshold ($max_v = 400$) has been set based on a validation set. Although the two criteria seem to be very restrictive, the amount of sampled patches and leaves is large enough to obtain reliable estimates.

The remaining distributions are used to estimate $\boldsymbol{\theta}^1$ by adding the mean offsets $\overline{\boldsymbol{\theta}^1}$ to the patch center $\boldsymbol{\theta}^1(\mathcal{P})$:

$$\mathcal{N}(\boldsymbol{\theta}^1; \boldsymbol{\theta}^1(\mathcal{P}) + \overline{\boldsymbol{\theta}^1}, \boldsymbol{\Sigma}^1). \tag{9}$$

The corresponding means for the position of the nose tip are illustrated in Fig. 3. The votes are then clustered, and the clusters are further refined by mean shift in order to remove additional outliers. As kernel for the mean shift, we use a sphere with a radius defined as one sixth of the radius of the average face in the model of [52]. A cluster is declared as a head if it contains a large enough number of votes. Because the number of votes is directly proportional to the number of trees in the forest (a tree can contribute up to one vote for each test patch), and because the number of patches sampled is inversely proportional to the square of the stride, we use the following threshold:

$$\beta \frac{\#trees}{stride^2}. \tag{10}$$

For our experiments, we use $\beta = 300$.

For each cluster left, i.e., each head detected, the distributions left are averaged, where the mean gives an estimate for the position of the nose tip $\boldsymbol{\theta^1}$ and the head orientation $\boldsymbol{\theta^2}$ and the covariance measures the uncertainty of the estimates.

3.3 Facial features localization

Since the framework for head pose estimation is very general and can be used in principle for predicting any continuous parameter of the face, the modifications for localizing facial features are straightforward. Instead of having only two classes as in Section 3.2.1, we have $K + 1$ classes, where K is the number of facial feature points we wish to localize. The set of training patches is therefore given by $\mathcal{P} = \{\mathcal{P}_i = (\mathcal{I}_i, c_i, \boldsymbol{\theta}_i)\}$, where $\boldsymbol{\theta}_i = \{\boldsymbol{\theta}_i^1, \boldsymbol{\theta}_i^2, \ldots \boldsymbol{\theta}_i^K\}$ are the offsets between the patch center and the 3D locations of each of the K feature points. Accordingly, (5) is computed for the K fiducials

and (6) is computed for the $K+1$ classes, where $c=0$ is the label for the background patches.

The testing, however, slightly differs. In Section 3.2.5, all patches are allowed to predict the location of the nose tip and the head orientation. While this works for nearly rigid transformations of the head, the location of the facial features depends also on local deformations of the face, e.g., the mouth shape. In order to avoid a bias towards the average face due to long distance votes that do not capture local deformations, we reduce the influence of patches that are more distant to the fiducial. We measure the confidence of a patch P for the location of a feature point n by

$$\exp\left(-\frac{\|\overline{\boldsymbol{\theta}^n}\|^2}{\gamma}\right), \tag{11}$$

where $\gamma = 0.2$ and $\overline{\boldsymbol{\theta}^n}$ is the average offset relative to point n, stored at the leaf where the patch P ends. Allowing a patch to vote only for feature points with a high confidence, i.e., above a feature-specific threshold, our algorithm can handle local deformations better, as our experiments show. The final 3D facial feature points' locations are obtained by performing mean-shift for each point n.

4 Evaluation

In this section, we thoroughly evaluate the proposed random forest framework for the tasks of head pose estimation from high quality range scans (Section 4.1), head pose estimation from low quality depth images (Section 4.2), and 3D facial features localization from high resolution scans (Section 4.3). Since the acquisition of annotated training data is an important step and a challenge task itself, we first present the used databases[1] in each subsection.

4.1 Head pose estimation - high resolution

4.1.1 Dataset

The easiest way to generate an abundance of training data with perfect ground truth is to synthesize head poses. To this end, we synthetically generated a very large training set of 640x480 range images of faces by rendering the 3D morphable model of [52]. We made such model undergo 50K different rotations, uniformly sampled from $\pm 95^\circ$ yaw, $\pm 50^\circ$ pitch, and $\pm 20^\circ$ roll. We also randomly varied the model's distance from the camera and further perturbed the first 30 modes of the PCA shape model sampling uniformly within ± 2 standard deviation from the mean, thus introducing variations also in identity[2].

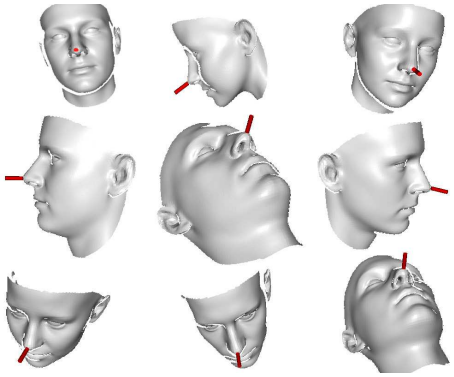


Fig. 4 Sample images from our synthetically generated training set. The heads show large 3D rotations and variations in the distance to the camera and also in identity. The red cylinder attached to the nose represents the ground truth face orientation.

Such a dataset was automatically annotated with the 3D coordinates of the nose tip and the applied rotations, represented as Euler angles. Figure 4 shows a few of the training faces, with the red cylinder pointing out from the nose indicating the annotation in terms of nose position and head direction. Note that the shape model captures only faces with neutral expression and closed mouth. Furthermore, important parts of the head like hair or the full neck are missing. This will be an issue in Section 4.2, where we discuss the limitations of synthetic training data.

For testing, we use the real sequences of the ETH Face Pose Range Image Data Set [10]. The database contains over 10K range images of 20 people (3 females, 6 subjects recorded twice, with and without glasses) recorded using the scanner of [66] while turning their head around, trying to cover all pitch and yaw rotations. The images have a resolution of 640x480 pixels, and a face typically consists of around 150x200 pixels. The heads undergo rotations of about $\pm 90^\circ$ yaw and $\pm 45^\circ$ pitch, while no roll is present. The data was annotated using person-specific templates and ICP tracking, in a similar fashion as what will be later described in 4.2.1 and shown in Figure 15. The provided ground truth contains the 3D coordinates of the nose tip and the vector pointing from the nose towards the facing direction.

[1] Most of the datasets are publicly available at `http://www.vision.ee.ethz.ch/datasets`.

[2] Because of the proprietary license for [52], we cannot share the above database. The PCA model, however, can be obtained from the University of Basel.

4.1.2 Experiments

In this section, we assume a face to be the prominent object in the image. That means that all leaves in a tree contain a probability $p(c = 1|\mathcal{P}) = 1$ and thus all patches extracted from the depth image will be allowed to vote, no matter their appearance.

Training a forest involves the choice of several parameters. In the following, we always stop growing a tree when the depth reaches 15, or if there are less than 20 patches left for training. Moreover, we randomly generate 20K tests for optimization at each node, i.e., 2K different combinations of f, F_1, and F_2 in Equation (3), each with 10 different threshold τ. Other parameters include the number of randomly selected training images, the number of patches extracted from each image (fixed to 20), the patch size, and the maximum size of the subpatches defining the areas F_1 and F_2 in the tests (set to be half the size of the patch). Also the number of feature channels available is an important parameter; in the following, we use all features (depth plus normals) unless otherwise specified.

A pair of crucial test-time parameters are the number of trees loaded in the forest and the stride controlling the spatial sampling of the patches from an input image. Such values can be intuitively tuned to find the desired trade-off between accuracy and temporal efficiency of the estimation process, making the algorithm adaptive to the constraints of different applications.

In all the following experiments, we use the Euclidean distance in millimeters as the nose localization error. For what concerns the orientation estimation, the ETH database does not contain large roll variations, and in fact these rotations are not encoded in the directional vector provided as ground truth. We therefore evaluated our orientation estimation performance computing the head direction vector from our estimates of the yaw and pitch angles and report the angular error in degrees with the ground truth vector.

Figure 5 describes the performance of the algorithm when we varied the size of the training patches and the number of samples used for training each tree. In Figure 5(a), the blue, continuous line shows the percentage of correctly classified images as a function of the patch size, when 1000 training images are used. Success is declared if the nose error is smaller than 20 mm and the angular error is below 15 degrees. Although this measure might be too generous for some applications, it reflects the relative estimation performance of the approach and is therefore a useful measure for comparing different settings of the proposed approach. The red, dashed line shows instead the percentage of false positives, i.e., missed detections, again varying with the size

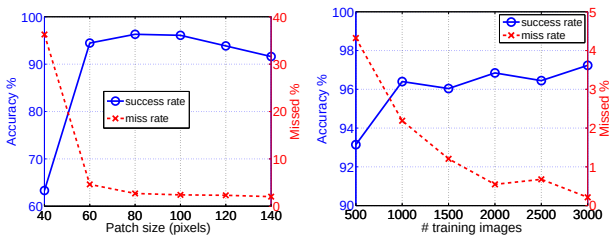


Fig. 5 (a) Success rate of the system depending on the patch size (when using 1000 training samples), overlaid to the missed detection rate. (b) Success and missed detection rate depending on the number of training data (for 100x100 patches). Success is defined for a nose error below 20 mm and angular error below 15 degrees.

of the patch. The plot shows that a minimum size for the patches is critical since small patches can not capture enough information to reliably predict the head pose. However, there is also a slight performance loss for large patches. In this case, the trees become more sensitive to occlusions and strong artifacts like holes since the patches cover a larger region and overlap more. Having a patch size between 80x80 and 100x100 pixels seems to be a good choice where the patches are discriminative enough to estimate the head pose, but they are still small enough such that an occlusion affects only a subset of patches. Figure 5(b) also shows accuracy and missed detections rate, this time for 100x100 patches, as a function of the number of training images. It can be noted that the performance increases with more training data, but it also saturates for training sets containing more than 2K images. For the following experiments, we trained on 3000 images, extracting 20 patches of size 100x100 pixels from each of them.

In all the following graphs, red circular markers consistently represent the performance of the system when all available feature channels are used (i.e., depth plus geometric normals), while the blue crosses refer to the results achieved employing only the depth channel.

The plots in Figure 6 show the time in milliseconds needed to process one frame, once loaded in the RAM. The values are reported as a function of the number of trees used and of the stride parameter. The numbers were computed over the whole ETH database, using an Intel Core i7 CPU @ 2.67GHz processor, without resorting to multithreading. Figure 6(a) plots the average run time for a stride fixed to 10 pixels as a function of the number of trees, while in Figure 6(b) 20 trees are loaded and the stride parameter changes instead. For strides equal to 10 and greater, the system always performs in real time. Unless otherwise specified, we use these settings in all the following experiments. Obviously, having to compute the normals (done on the CPU

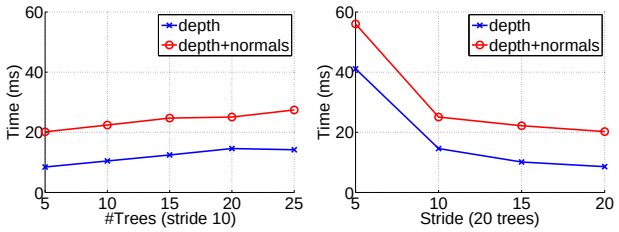


Fig. 6 Processing time: a) Regression time as a function of the number of trees in the forest when the stride is fixed to 10 pixels. b) Run time for a forest of 20 trees as a function of the stride parameter.

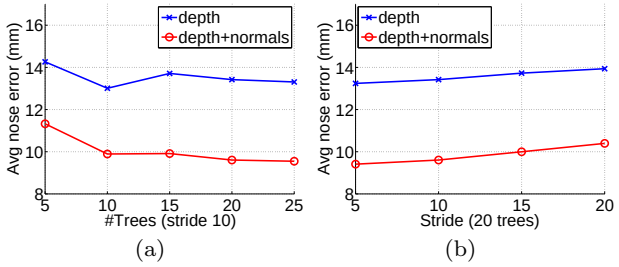


Fig. 7 Mean errors (in millimeters) for the nose localization task, as a function of the number of trees in the forest (a) and of the stride (b).

using a 4-neighborhood) increases processing time, but, for the high-quality scans we are dealing with, the boost in accuracy justifies the loss in terms of speed, as can be seen in the next plots.

Figure 7(a) shows the average errors in the nose localization task, plotted as a function of the number of trees when the stride is fixed to 10, while in Figure 7(b) 20 trees are loaded and the stride is changed. Similarly, the plots in Figures 8(a) and 8(b) present the average errors in the estimation of the head orientation. When comparing Figures 6, 7, and 8, we can conclude that it is better to increase the stride than reducing the number of trees when the processing time needs to be reduced. Using normals in addition also improves the detection performance more than increasing the number of trees. In particular, using depth and normals with a stride of 10 gives a good trade-off between accuracy and processing time for our experimental settings.

In Figure 9, the plots show the accuracy of the system computed over the whole ETH database, when both depth and geometric normals are used as features. Specifically, the curves in Figure 9(a) and Figure 9(b) represent the percentage of correctly estimated depth images as functions of the success threshold set for the nose localization error, respectively for the angular error. Using all the available feature channels performs consistently better than relying only on the depth in-

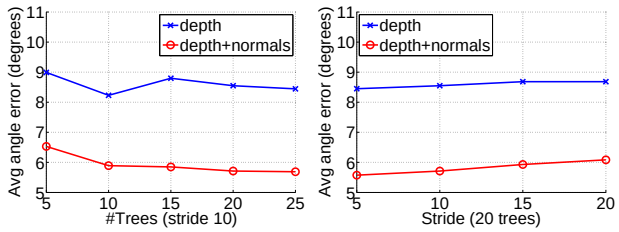


Fig. 8 Mean errors (degrees) for the orientation estimation task, as a function of the number of trees in the forest (a) and of the stride (b).

formation. The plots show also the success rate of the method of [10], applied to the same data[3]; their algorithm uses information about the normals to generate nose candidates, but not for refining the pose estimation on the GPU, where a measure based on the normalized sum of squared depth differences between reference and input range image is used.

Our approach proves better at both the tasks of nose tip detection and head orientation estimation. We improve over the state-of-the-art especially at low thresholds, which are also the most relevant. In particular, for a threshold of 10 mm on the nose localization error, our improvement is of about 10% (from 63.2% to 73.0%), and even better for a threshold of 10 degrees on the angular error: Our system succeded in 94.7%, compared to 76.3% of Breitenstein et al.

Table 1 reports mean and standard deviation of the errors, compared to the ones of [10]. The first columns show mean and standard deviation for the Euclidean error in the nose tip localization task, the orientation estimation task, and for the yaw and pitch estimation errors taken singularly. The last two columns give the percentages of correctly estimated images for a threshold on the angular error of 10 degrees, and on the nose localization error of 10 millimeters. The average errors were computed from the ETH database, where our system did not return (i.e., no cluster of votes large enough was found) an estimate in 0.4% of the cases, while the approach of Breitenstein failed 1.6% of the time; only faces where both the systems returned an estimate were used to compute the average and standard deviation values.

Figure 10 shows the success rate of the system applied to the ETH database (using 20 trees and a stride of 10) for an angular error threshold of 15 ° and a nose error threshold of 20 mm. The heat map shows the database divided in 15 ° × 15 ° bins depending on the head's pitch and yaw angles. The color encodes the amount of images in each bin, according to the side

[3] We used the source code provided by the authors.

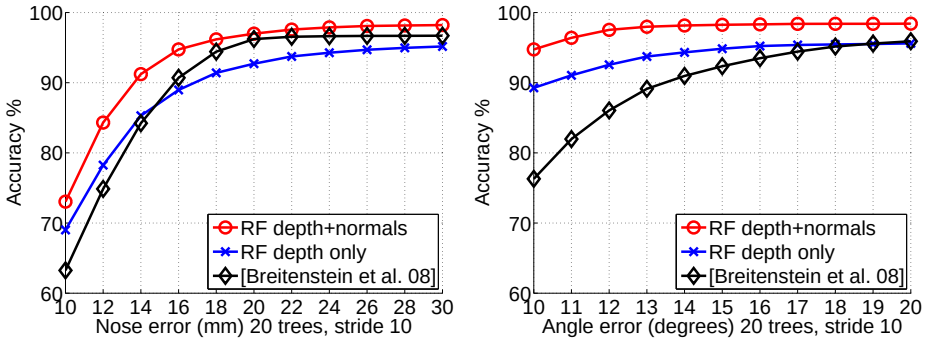


Fig. 9 Accuracy: (a) Percentage of correctly estimated poses as a function of the nose error threshold. (b) Accuracy plotted against the angle error threshold. The additional information coming from the normals (red curves) consistently boosts the performance. The black curve represents the accuracy of [10] on the same dataset.

	Nose error	Direction error	Yaw error	Pitch error	dir. ($\leq 10\,^\circ$)	nose (≤ 10 mm)
Random Forests	9.6 ± 13.4 mm	$5.7 \pm 8.6\,^\circ$	$4.4 \pm 2.7\,^\circ$	$3.2 \pm 2.7\,^\circ$	94.7%	73.0%
Breitenstein 08	10.3 ± 17.5 mm	$9.1 \pm 12.6\,^\circ$	$7.0 \pm 13.4\,^\circ$	$4.8 \pm 4.9\,^\circ$	76.3%	63.2%

Table 1 Comparison of our results with the ones of [10]. Mean and standard deviation are given for the errors on nose localization, direction estimation, and singularly for yaw and pitch angles. The values in the last two columns are the percentages of correctly estimated images for a threshold on the angular error of 10 degrees, and on the nose localization error of 10 millimeters. We used a forest with 20 trees, leveraging both depth and normals as features, testing with a stride of 10 pixels.

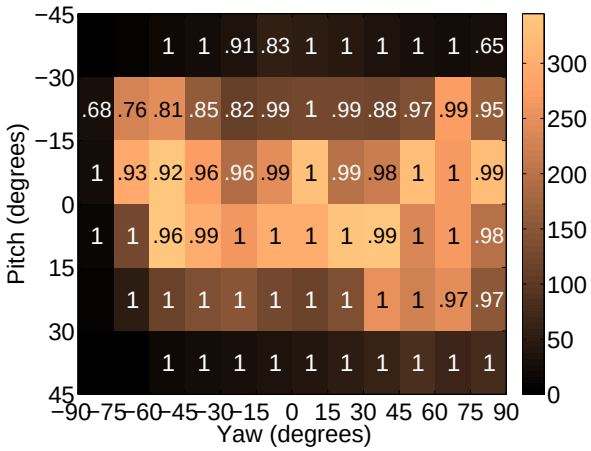


Fig. 10 Normalized success rates of the estimation, equivalent of Figure 10 in [10]. The database was discretized in $15\,^\circ \times 15\,^\circ$ areas and the accuracy computed for each range of angles separately. The color encodes the number of images falling in each region, as explained by the side bar. Success is declared when the nose error is below 20 mm and the angular error is below 15 degrees.

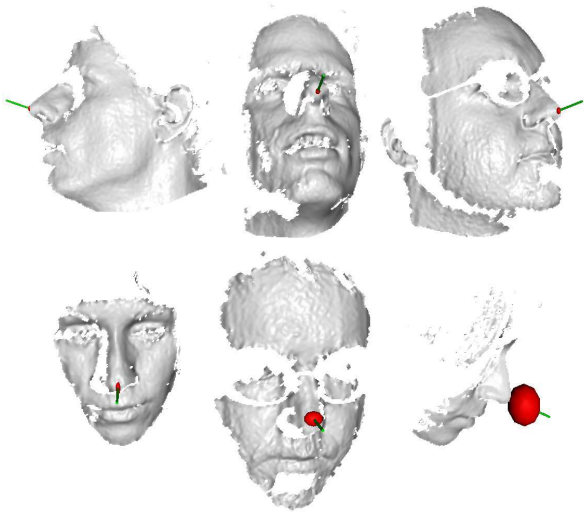


Fig. 11 Correctly estimated poses from the ETH database. Large rotations, glasses, and facial hair do not pose major problems in most of the cases. The green cylinder represents the estimated head rotation, while the red ellipse is centered on the estimated 3D nose position and scaled according to the covariance provided by the forest (scaled by a factor of 10 to ease the visualization).

color bar. The results are 100% or close to 100% for most of the bins, especially in the central region of the map, which is where most of the images fall. Our results are comparable or superior to the equivalent plot in [10].

Figure 11 shows some successfully processed frames from the ETH database. The red ellipse is placed on the estimated nose tip location and scaled according to the covariance output of the regression forest. The green cylinder stretches from the nose tip along the estimated head direction. Our system is robust to large rotations and partial facial occlusions (note the girl at the bottom right, with most of the face covered by hair,

Fig. 14 Example frames from our real time head pose estimation system, showing how the regression works even in the presence of partial occlusions. notably of the nose. Facial expressions also can be handled to a certain degree, even though we trained only on neutral faces.

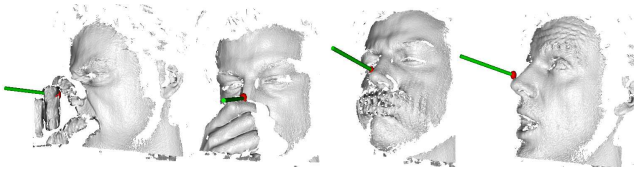


Fig. 12 Example frames from a sequence acquired with the 3D scanner of [66]. Occlusions (even of the nose) and facial expressions can be handled by our system.

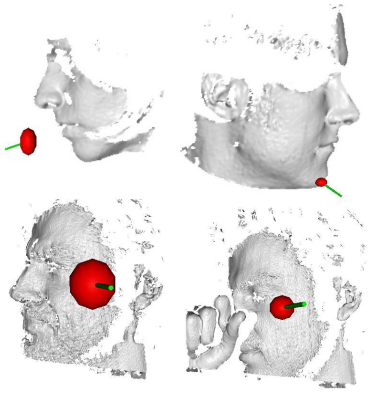


Fig. 13 Example failure images from the ETH database. The large ellipse denotes a high variance for the estimate of the nose location.

which is not reconstructed by the scanner). Additional results are shown in Figure 12, demonstrating how the proposed algorithm can handle a certain degree of facial expression and occlusion, maintaining an acceptable accuracy of the estimate.

We ran our real time system on a Intel Core 2 Duo computer @ 2GHz, equipped with 2GB of RAM, which was simultaneously used to acquire the range data as explained in [66]. Figure 14 shows some example frames from the video[4]. Our method successfully estimates the head pose even when the nose is totally occluded and thus most of the other approaches based on 3D (e.g., [10]) would completely fail. Some degree of facial dynamics also does not seem to cause problems to the regression in many cases, even though the synthetic training dataset contains only neutral faces; only very large mouth movements like yawning result in a loss of accuracy.

Some example failures are rendered in Figure 13. Note how the red ellipse is usually large, indicating a high uncertainty of the estimate. These kind of results are usually caused by a combination of large rotations and missing parts in the reconstruction, e.g., because of hair or occlusions; in those circumstances, clusters of votes can appear in the wrong locations and if the number of votes in them is high enough, they might be erroneously selected as the nose tip.

4.2 Head pose estimation - low resolution

4.2.1 Dataset

To train and test our head pose estimation system on low quality depth images coming from a commercial sensor like the Kinect, synthesizing a database is not an easy task. First of all, such a consumer depth camera is built specifically for being used in a living-room environment, i.e., capturing humans with their full body. This means that heads are always present in the image together with other body parts, usually the torso and the arms. Because regions of the depth image different than the head are not informative about the head pose, we need examples of negative patches, e.g., coming from the body, together with positive patches extracted from the face region. Lacking the human body model and MoCap trajectories employed by [57], we resorted to record a new database using a Kinect. The dataset comprises 24 sequences of 20 different subjects (14 men and 6 women, 4 people with glasses) recorded while sitting about 1 meter away from the sensor. All subjects rotated their heads trying to span all possible

[4] `www.vision.ee.ethz.ch/~gfanelli/head_pose/head_forest.html`

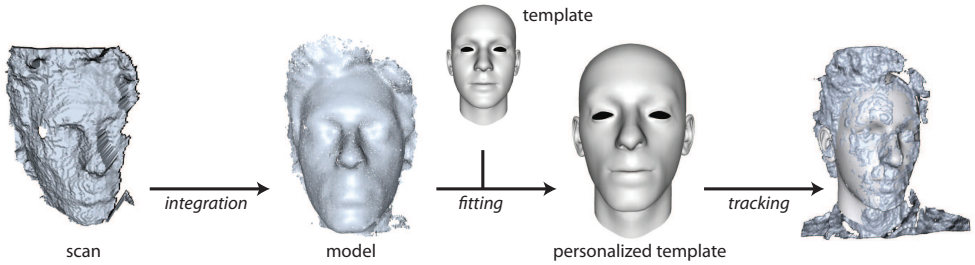


Fig. 15 Automatic pose labeling: A user turns the head in front of the depth sensor, the scans are integrated into a point cloud model and a generic template is fit to it. The personalized template is used for accurate rigid tracking.

ranges of yaw and pitch angles, but also some roll is present in the data.

To label the sequences with the position of the head and its orientation, we processed the data off-line with a state-of-the-art template-based head tracker [65][5], as illustrated in Figure 15. A generic template was deformed to match each person's identity as follows. First, a sequence of scans of the users' neutral face recorded from different viewpoints were registered and fused into one 3D point cloud as described by [68]. Then, the 3D morphable model of [52] was used, together with graph-based non-rigid ICP [38], to adapt the generic face template to the point cloud. Each sequence was thus tracked with the subject's template using ICP [4], obtaining as output for each frame the 3D location of the head (and thus of the nose tip) and the rotation angles.

Using such automatic method to acquire the ground truth for our database allowed us to annotate over 15K frames in a matter of minutes. Moreover, we found that the mean translation and rotation errors were around 1 mm and 1 degree respectively. Please note that such personalized face model is only needed for labeling the training data: Our head pose estimation system does not assume any initialization phase nor person-specific training, and works on a frame-by-frame basis.

The resulting Biwi Kinect Head Pose Database contains head rotations in the range of around $\pm 75^\circ$ for yaw, $\pm 60^\circ$ for pitch, and $\pm 50^\circ$ for roll. Faces are 90x110 pixels in size on average. Besides the depth data which we used for our algorithm, the corresponding RGB images are also available, as shown in Figure 16.

4.2.2 Experiments

Training patches must now be distinguished between positives (extracted from the head region) and negatives (belonging to other body parts). When we randomly extracted patches from the Biwi Kinect Head Pose Database, we labeled them as positive only if the Euclidean distance between the 3D point falling at the center of the patch and the closest point on the face model used for annotation was below 10 millimeters. In this way, negative patches were extracted not only from the torso and the arms, but also from the hair. Figure 17 shows this process.

In the following experiments, unless explicitly mentioned otherwise, all training and testing parameters are kept the same as in the previous evaluation done on high resolution scans. We only reduce the size of the

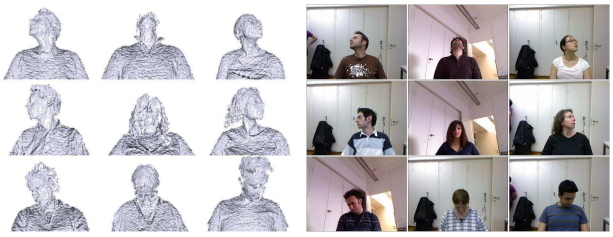


Fig. 16 Example frames from the Biwi Kinect Head Pose Database. Both depth and RGB images are present in the dataset, annotated with head poses. In this paper, we only use the depth images for the head pose estimation algorithm.

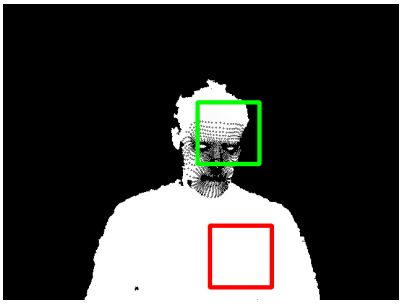
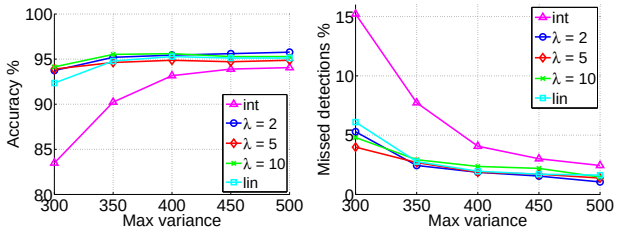


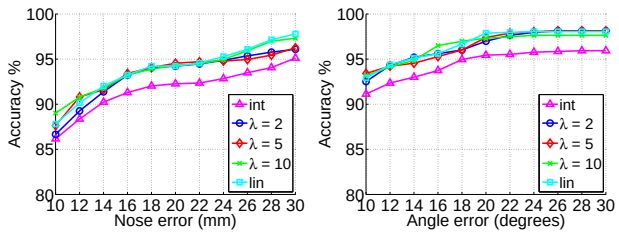
Fig. 17 Training patches extracted from the annotated depth images of the Biwi Kinect Head Pose Database acquired with a Microsoft Kinect. The green box represents a positive patch, while the red one is an example of a negative patch. The dark dots on the face represent the model's vertices used to define the patch label: Only if the center of the patch falls near such vertices, the patch is considered as positive.

[5] Commercially available: `http://www.faceshift.com`



(a) (b)

Fig. 18 Accuracy (a) of the tested methods as a function of the maximum variance parameter, used to prune less informative leaves in the forest. Success is defined when the nose estimation error is below $20mm$ and the thresholds for the orientation estimation error is set to 15 degrees. The plots in (b) show the percentage of images were did not return an estimate (false negatives), again as a function of the maximum variance. It can be noted that the evaluated methods perform rather similarly and the differences are small, except for the interleaved scenario, which consistently performs worse.



(a) (b)

Fig. 19 Accuracy for the nose tip estimation error (a), respectively the angle error (b) of the tested methods. The curves are plotted for different values of the threshold defining success.

patches to 80x80 because the heads are smaller in the Kinect images than in the 3D scans. Furthermore, we extract 20 negative patches per training image in addition to the 20 positive patches. For testing, patches ending in a leaf with $p(c|\mathcal{P}) < 1$ and $\mathrm{tr}\left(\Sigma^1\right) \geq max_v$ are discarded. Given the much lower quality of the depth reconstruction, using the geometric normals as additional features does not bring any improvement to the estimation, therefore we only use the depth channel in this section. Because the database does not contain a uniform distribution of head poses, but has a sharp peak around the frontal face configuration, as can be noted from Figure 21, we bin the space of yaw and pitch angles and cap the number of images for each bin.

In Section 3.2.3, we described different ways to train forests capable of classifying depth patches into head or body and at the same time estimating the head pose from the positive patches. In order to compare the discussed training strategies (*interleaved*, *linear*, and *exponential*), we divided the database into a testing and training set of respectively 2 (persons number 1 and 12) and 18 subjects.

Depending on the method used to combine the classification and regression measures, additional parameters might be needed. In the *interleaved* setting [29], each measure is chosen with uniform probability, except at the two deepest levels of the trees where the regression measure is always used. For the *linear* weighting approach (7), we set α and t_p as suggested by [49], namely to 1.0 and 0.8. For the *exponential* weighting function based on the tree depth (8), we used λ equal to 2, 5, and 10. All comparisons were done with a forest of 20 trees and a stride of 10.

Results are plotted in Figures 18 and 19. The success rate of the algorithm is shown in Figure 18(a), as the max_v parameter increases, i.e., as more and more leaves are allowed to vote. Success means that the detected nose tip is within $20mm$ from the ground truth location, and that the angular error is below $15\,^\circ$. Figure 18(b) shows, again as a function of the maximum leaves' variance, the percentage of missed detections. In general, low values of the parameter max_v have a negative impact on the performance, as the number of votes left can become too small. However, reducing the maximum variance makes only the most certain votes pass, producing better estimates if there are many votes available, e.g., when the face is covering a large part of the image; moreover, reducing max_v can also be used to speed up the estimation time. The parameter shows how well the different schemes minimize the classification and regression uncertainty. Because only the leaves with low uncertainties are used for voting, trees with a large percentage of leaves with a high uncertainty will yield a high missed detection rate, as shown in Figure 18(b). In this regards, all tested methods appear to behave similarly, except for the interleaved scenario, which consistently performs worse, indicating that the trees produced using such method had leaves with higher uncertainty. We also note that the exponential weighting scheme with $\lambda = 5$ returns the lowest number of missed detections.

The plots in Figures 19(a) and 19(b) show the success rate as function of a threshold on the nose error, respectively on the orientation error. We note again the lower accuracy achieved by the interleaving scheme, while the other methods perform similarly.

We performed a 4-fold, subject-independent cross-validation on the Biwi Kinect Head Pose Database, using an exponential weighting scheme with λ set to 5. All other parameters were kept as described earlier. The results are given in Table 2, where mean and standard deviation of the nose tip localization, face orientation estimation, yaw, pitch and roll errors are shown together with the percentage of missed detections and the average time necessary to process an image, depending on

Stride	Nose (mm)	Direction (°)	Yaw (°)	Pitch (°)	Roll (°)	Missed (%)	Time (ms)
5	12.2 ± 22.8	5.9 ± 8.1	3.8 ± 6.5	3.5 ± 5.8	5.4 ± 6.0	6.6	44.7
10	12.6 ± 23.4	6.1 ± 8.8	4.0 ± 7.1	3.6 ± 6.0	5.5 ± 6.2	6.5	17.8
15	13.4 ± 26.9	6.4 ± 9.4	4.2 ± 7.8	3.8 ± 6.4	5.5 ± 6.2	6.5	10.7

Table 2 Mean and standard deviation of the errors for the nose position and Euler angles estimation, together with rate of false negatives and average runtime, as functions of the stride. The values are computed by 4-fold, subject independent cross validation on the entire Biwi Kinect Head Pose Database.

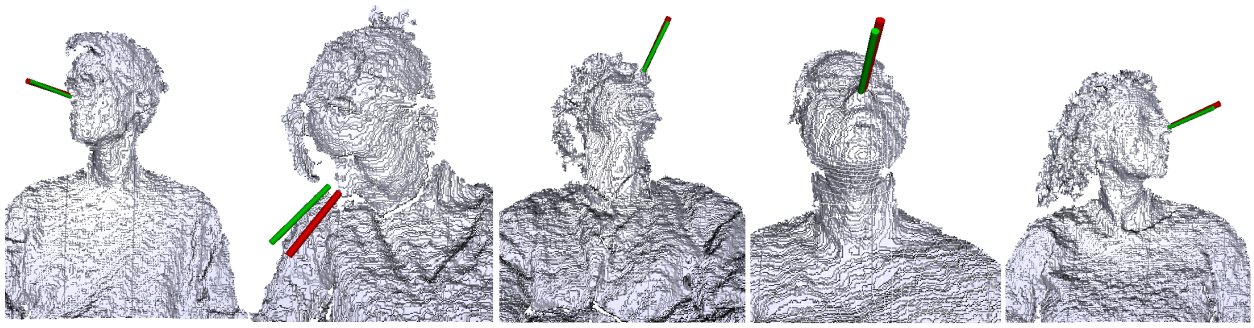


Fig. 20 Examples of successfully estimated depth images out of our Kinect database. The green cylinder represents the estimated head pose, while the red one encodes the ground truth.

the stride parameter. It can be noted that the system performs beyond real time for strides greater than or equal to 10 (needing less than 20ms to process a frame on a 2.67GHz Intel Core i7 CPU, i.e., running at over 50 frames per second), still maintaining a small number of missed detections and low errors. Some examples of successful estimations are given in Figure 20, where the green cylinder encodes the estimated head pose, while the red one represents the ground truth.

Some typical failure cases are shown in Figure 22, with examples of missed detections, wrong detections, and a case of a false positive (the magenta cylinder on the hand). Apart from very large rotations, common issues of the current system include long hair covering part of the head, and distracting objects like hands or clothing. Adding more negatives samples to the training set (e.g., of hands) would alleviate some of these problems.

Figure 21 is the equivalent of Figure 10, i.e., the results of the cross-validation (stride 10) are given as ratios of successfully estimated frames for each 15x15 degrees bin. Success is again declared for nose localization errors $\leq 20mm$ and angular errors $\leq 15\,^\circ$. The map is colored according to the number of images present in each bin. It can be noted how the central areas contain a lot more frames than the border ones, thus the necessity of binning the database before random sampling for training.

As a last experiment, we rendered depth images of the face templates which were used to annotate the database; see Figure 15. We simulated a Kinect by using the same intrinsics camera matrix. In this way, we created a dataset of synthetic depth images of heads, undergoing the same global movements as the original data. Also the identity of the templates are consistent with the recorded dataset. We thus extracted the positive patches from the synthetic data, while using the original depth data to sample negatives. Using the same settings as for Table 2, we achieved the results presented in Table 3. All errors are higher, in particular the ones related to the nose tip.

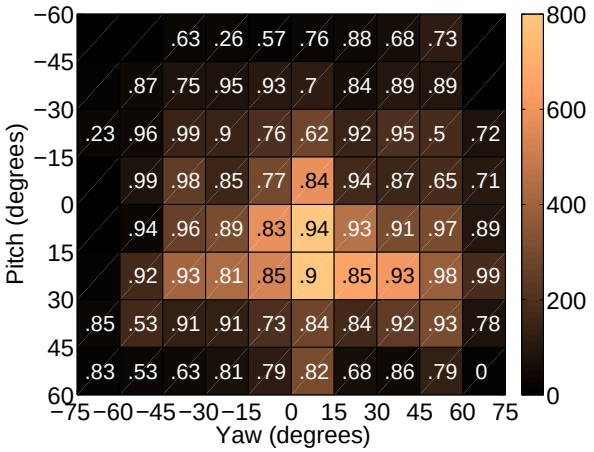


Fig. 21 Equivalent of Figure 10: Normalized success rates of the estimation, where success means nose error below 20 mm and angular error below 15 degrees. The database was discretized in $15\,^\circ \times 15\,^\circ$ areas and the accuracy computed for each range of angles separately. The color encodes the number of images falling in each region, as explained by the side bar.

Stride	Nose (mm)	Direction (°)	Yaw (°)	Pitch (°)	Roll (°)	Missed (%)	Time (ms)
5	19.7 ± 46.5	8.5 ± 12.9	6.0 ± 11.5	4.8 ± 7.1	5.8 ± 6.8	9.3	44.0
10	20.2 ± 47.3	8.7 ± 13.1	6.2 ± 11.8	4.9 ± 7.3	5.8 ± 6.8	9.2	15.3
15	21.7 ± 50.7	9.3 ± 14.0	6.6 ± 12.6	5.2 ± 7.7	6.0 ± 7.1	8.7	10.0

Table 3 Results of the cross-validation experiments, when synthetic data was used to extract positive training patches.

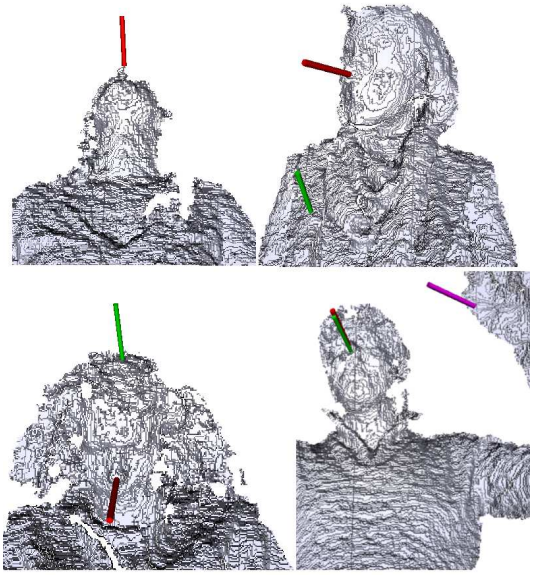
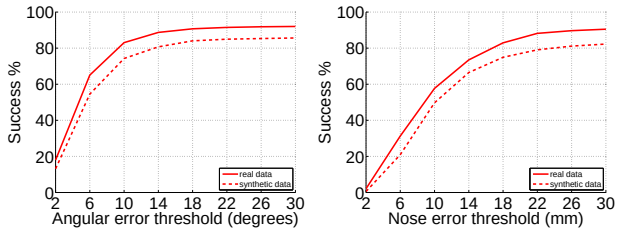


Fig. 22 Some typical failure cases of the algorithm, showing a missed detection and two false positives. The estimated head pose is shown in green, the ground truth is in red. The magenta cylinder indicates a (wrong) second detection.



(a) (b)

Fig. 23 Percentage of correctly estimated images (4-fold cross validation) depending on the (a) head localization and (b) angular error thresholds. The continuous lines represent the performance when real data is used for training, while the dashed lines are the results of the forests trained on positive patches extracted from synthetically generated heads. The whole Kinect dataset is always used for testing.

Fig. 24 Example frame from the $B3D(AC)^2$ database. High quality range scans come together with RGB texture (not used in this paper) and a high definition generic template deformed to match the facial expression in each frame.

The plots in Figure 23 compare the accuracy of the system when trained on real data and when using the synthesized heads as positive samples. The continuous lines are the results obtained using real data, while the dashed lines represent the accuracy of the system when trained on synthetic positive samples (and tested on real data). Specifically, Figure 23(a) plots the success rate as a function of the orientation estimation error, while Figure 23(b) as a function of the nose error.

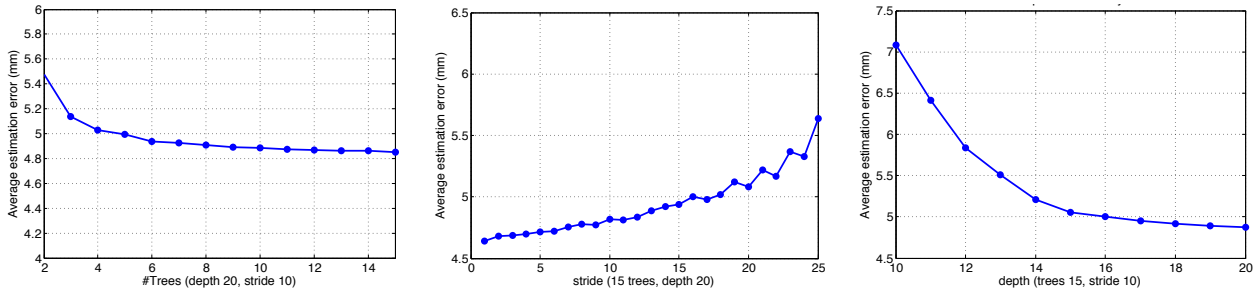
Using the synthetic heads decreases the performance, though not in a very incisive manner. The loss in performance can be explained by the incomplete head model that does not model hair and anything below the neck as shown in Figure 15. The incompleteness of models for generating training data seems to be indeed a limitation of synthetic training data. Another source for the performance loss is the missing sensor noise in the synthetic data.

4.3 Facial features localization

4.3.1 Datasets

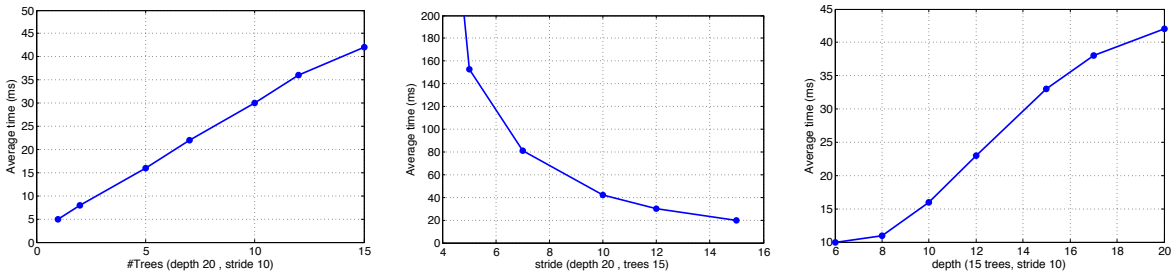
When extending the random forest framework for the purpose of facial features localization, once again a large dataset of annotated range images of faces is needed.

As a first dataset, we chose $B3D(AC)^2$ [24]. This is a relatively naturalistic and large set of high quality, dynamic facial scans, with subjects recorded using the 3D scanner of [66] while pronouncing a set of 40 predefined sentences both in a neutral and in an induced emotional state. The dataset includes 14 subjects, 8 females and 6 males, and a total of over 1100 sequences, 4.67 seconds long on average, for over 120K depth images. Figure 24 shows an example frame: Depth and RGB images come together with a template of over 23K vertices, deformed to fit the specific expression. Thanks to such annotation, we could select a set of 14



(a) (b) (c)

Fig. 25 (a) Average Euclidean error for the localization of all the fiducials in millimeters depending on the number of trees, for a stride fixed to 10 and maximum depth set to 20. (b) Error depending on the stride parameter, with 15 trees in the forest and depth 20. (c) Error depending on the depth of the trees, with 15 trees in the forest and stride 10. For most of the configurations, the average error is below 5 mm.



(a) (b) (c)

Fig. 26 Estimation time for all the 14 feature points, averaged over 500 randomly selected frames. As can be noted, the values are low and our system runs at frame rates higher than 25 fps for most of the configurations. (a) Processing time depending on the number of trees, when the stride is fixed to 10. (b) Run time depending on the stride, having fixed the number of trees to 15. (c) Time in ms needed to process a frame, depending on the maximum depth which the trees are allowed to grow.

facial features on the generic template and automatically extract their 3D locations from all frames in the dataset. In our facial features detection algorithm, we only use the depth images from the above database, i.e., we do not rely on the RGB data.

As a second dataset, we used *BU3DFE* [71], which contains a larger number of subjects (100, 56 females and 44 males), and stronger facial deformations. Each subject performed the six basic expressions plus neutral in front of a 3D face scanner. Each of the six prototypic expressions (happiness, disgust, fear, angry, surprise and sadness) includes four levels of intensity, i.e., there are 25 static 3D expression models for each subject, resulting in a total of 2500 faces. Because the dataset comes in form of 2.5 face models, we could render them into depth images, first without rotations, then with randomly varying the pitch, yaw, and roll angles, sampling uniformly between ±20 degrees. All models come with manually annotated 83 facial features locations in 3D, from which we extracted the 14 fiducial which interested us: Eye, nose, and mouth corners, plus outer midpoints on the lips and the two extremes of the eyebrows.

4.3.2 Experiments

When building a forest for localizing facial features from range scans, we sample training patches both from the inside and the outside of the face region. A patch is considered as a positive training sample for facial feature k if the norm of the corresponding offset vector is below a threshold, i.e., if $||\boldsymbol{\theta}^k|| \leq 0.2r$ where r is the radius of the average face. Since the definition of the class $c = k$ already localizes patches in a neighborhood of each feature, we use only the classification measure (6) for training. Using an additional regression measure did not change the performance in this setting.

Since facial features depend more on local deformations of the face compared to the head pose, we use smaller patches of size 40x40 pixels. Since we have also more classes, we increased the depth of the trees to 20 and also the number of sampled patches for training. Each tree is built from 5000 randomly sampled images, each contributing with 50 patches, 30 extracted from within the face boundary (i.e., the bounding box defined by the ground truth facial feature locations) and 20 from outside the face. During testing, each patch reaching a leaf votes for feature point k if $P(c = k|\mathcal{P}) \geq$

0.5, $\mathrm{tr}(\boldsymbol{\Sigma}^k) < max_v^k$, and the confidence (11) is above a threshold. The threshold and the values max_v^k for each facial feature point k are estimated by grid search over a validation set. In particular, we extract patches from 2000 randomly selected training images out of the $BA3D(AC)^2$ database. We only use the depth channel, without resorting to additional features like the geometric normals.

We experimentally evaluated the influence of the number of trees in the random forest, the stride, and the maximum depth of the trees. For these experiments, we trained on 12 of the subjects of the $B3D(AC)^2$ database and tested on the remaining two subjects, one man and one woman. As shown in Fig. 25, increasing the number of trees, letting them grow deeper, and reducing the stride, all have positive effects on the quality of the results. The plots show the mean Euclidean error, in millimeters, averaged over all the feature points and all the frames in the test set. For most of the configurations shown, the average error is below 5 millimeters. However, increased accuracy comes at the cost of a higher computation time. Fig. 26 shows the time in milliseconds needed to process a test image once loaded into memory (the values are averaged over 500 randomly selected frames), as a function of the number of trees, stride, and maximum depth of trees. As can be seen, for a stride of 10 pixels, we achieve real time performance, i.e., frame rates above 25 fps, when loading up to 15 trees of depth 20. In all the following experiments, we thus use a forest of 15 trees, each with a maximum depth of 20 and set the stride to 10 pixels.

We further performed 5-fold, subject-independent cross validations on the $B3D(AC)^2$ database [24], and the $BU3DFE$ database rendered both in frontal pose and with random rotations added. Table 4 shows mean and standard deviation of the errors in millimeters for all the analyzed facial features. Moreover, the success rates (for all feature points on the whole database) are given for two conservative thresholds of 10 and 5 millimeters. The outer brow corners are the points most often misplaced; this is not surprising, as the brows present limited variation in the depth channel. For the $BU3DFE$ database, where the mouth deforms more, the lower lip midpoint is also sometimes wrongly estimated.

The plot in Fig. 27 shows the percentage of correctly estimated points for all the tested databases, as a function of the threshold defining success. For the $B3D(AC)^2$ dataset, we localized the feature points with an error below or equal to 5 mm in 87.7% of the cases, which becomes 98.2% for a threshold of 10 mm. For the $BU3DFE$ database in its frontal renderings, we correctly localized 76.8% of the points for a 5 mm threshold and 96.9% for a 10 mm one; such accuracies are lower for the database with synthetically introduced rotations, namely 62.4% and 92.2%.

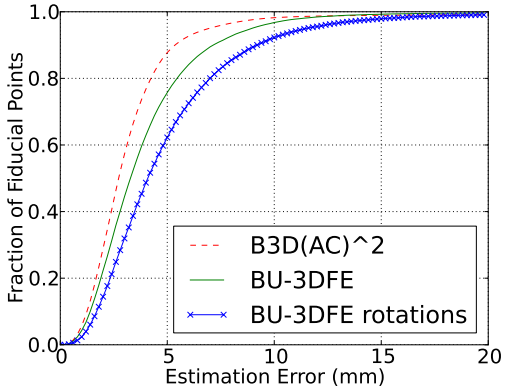


Fig. 27 Accuracy of the algorithm on the full database (5-fold cross validation), as the threshold defining success changes. The accuracy is the percentage of correctly estimated facial feature points.

Some examples of successful detections of the 14 facial feature points on test range images from the test datasets are shown in Figure 28. The three images on the left side come from the $B3D(AC)^2$ database and depict people talking, while the ones on the right side are renderings of the $BU3DFE$ database, with larger facial deformations due to posed expressions. Some failure examples, where not all fiducials were correctly localized, are shown in Figure 32. Most errors occur around the mouth regions due to the large deformations and the noisy reconstruction of the teeth and oral cavity.

As a last experiment, in order to test the performance with regard to partial occlusions and missing reconstructions, we tested our system on synthetically corrupted range images. First, we randomly selected parts of the depth images in the $B3D(AC)^2$ database and set them to zero, then, we rendered a hand model in front of the faces from the $BU3DFE$ dataset, in order to simulate more realistic occlusions. In both cases, we trained on the original data and tested on the corrupted images, in a 5-fold cross-validation experiment.

Figure 29 shows the mean error, averaged over all the facial feature points, as a function of the amount of synthetically removed reconstructions on the $B3D(AC)^2$ corpus. The extent of missing data is measured as the percentage of the area covered by the face bounding box, i.e., the smallest rectangle enclosing all projections on the depth image of the facial feature points ground truth locations, enlarged by the patch size (40 pixels) on both dimensions. The occluding patches are required to fall within the face bounding box and sample test faces are rendered over the curve to ease visualization. As

	$B3D(AC)^2$		BU-3DFE		BU-3DFE with rotations	
fiducial	succ. % (5/10mm)	mean ± std	succ. % (5/10mm)	mean ± std	suc. % (5/10mm)	mean ± std
outEyeL	85.37/98.67	3.29 ± 3.56	81.20/99.35	3.36 ± 2.09	66.01/95.93	4.66 ± 3.39
innEyeL	97.12/99.19	2.58 ± 2.94	97.72/99.95	2.32 ± 1.28	92.12/99.83	2.87 ± 1.71
innEyeR	95.20/98.83	3.41 ± 2.87	97.68/99.95	2.44 ± 1.47	91.43/99.83	2.94 ± 1.72
outEyeR	72.86/96.89	4.69 ± 6.42	83.55/99.26	3.32 ± 2.32	64.14/94.27	4.77 ± 4.69
noseL	96.80/99.88	2.41 ± 1.52	88.34/99.87	3.11 ± 1.58	81.03/99.55	3.48 ± 1.84
noseR	94.53/99.30	2.60 ± 2.47	87.45/99.75	3.24 ± 1.67	81.52/99.43	3.56 ± 1.92
mouthL	88.88/98.97	3.04 ± 2.15	69.38/95.53	4.46 ± 3.25	55.70/87.77	6.04 ± 5.44
mouthR	85.13/98.54	3.38 ± 3.38	69.95/95.93	4.41 ± 3.21	54.20/88.55	5.82 ± 4.77
upLip	94.55/99.85	2.95 ± 1.46	87.33/99.30	3.10 ± 2.09	73.04/98.05	4.05 ± 2.44
lowLip	86.17/98.34	3.38 ± 2.61	75.76/95.41	4.52 ± 5.39	49.61/89.52	6.45 ± 6.68
outBrowL	68.31/95.56	4.50 ± 3.66	49.49/88.10	5.90 ± 3.64	32.48/77.46	7.37 ± 4.22
innBrowL	93.95/98.39	2.86 ± 3.85	71.29/98.45	4.42 ± 2.56	51.23/92.89	5.42 ± 2.95
innBrowR	92.50/97.83	3.34 ± 4.16	68.77/97.68	4.59 ± 2.74	49.00/92.28	5.61 ± 3.08
outBrowR	77.01/94.66	4.99 ± 7.05	47.86/88.46	6.10 ± 3.93	32.19/75.39	7.71 ± 4.59

Table 4 Summary of the performance of our method, applied to a 5-fold cross validation on all the tested databases, for each fiducial. Together with mean and standard deviation of the Euclidean errors, the success rates for conservative thresholds of 5, respectively 10 millimeters are shown.

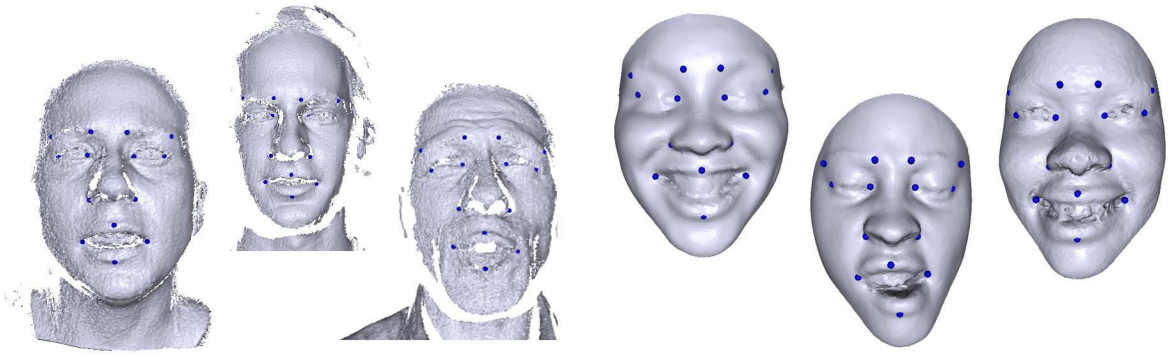


Fig. 28 Successfully localized facial features localization on some test scans from the $B3D(AC)^2$ database (left) and the $BU3DFE$ dataset (right).

can be seen from the plot, the proposed method is robust to such missing reconstructions: Even when 50% of the data was missing, we still obtained an average error below 8 mm.

Figures 30 and 31 relate to the artificially occluded $BU3DFE$ data. In particular, in Figure 30, the success rate (averaged over all feature points) is plotted against the percentage of occlusion. The blue, continuous line, relates to a threshold of 10 mm, while the red, dashed line correspond to a threshold of 5 mm. The amount of occlusion is calculated as the ratio of pixels in the face which are covered by the hand. Similarly, Figure 31 shows the average error in the localization, as the occlusion increases. As our system was trained only on positive patches coming from depth images of faces, this experiment proved more challenging than the previous one, and the errors grow faster as the hand occludes a higher percentage of the face surface.

Some examples of successful detections on corrupted images are shown in Figure 34. On the left, the miss-

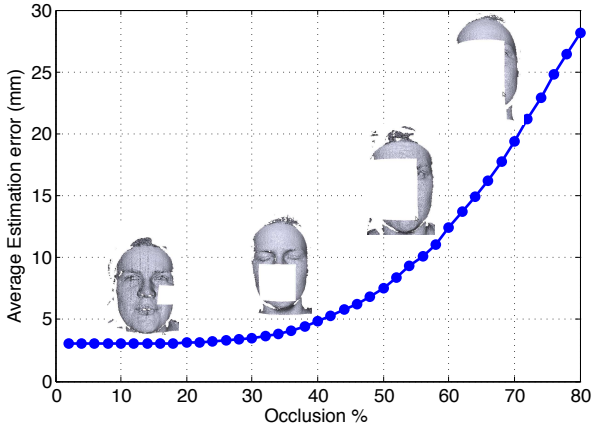


Fig. 29 Mean errors (averaged over all the feature points) as a function of the amount of synthetically removed reconstruction from the $B3D(AC)^2$ database, measured as % of the bounding box enclosing the ground truth locations of the fiducials. Example image are overlaid on the plot, more examples are shown in Figure 34, left.

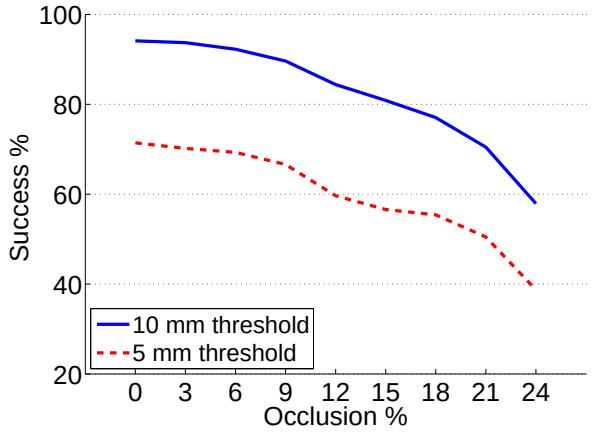


Fig. 30 Success rate, for a threshold of 10, respectively 5 mm, for the facial features localization task. The curves are plotted as functions of the percentage of face pixels occluded by the hand, in the renderings of the $BU3DFE$ dataset. Examples are shown in Figure 34, right.

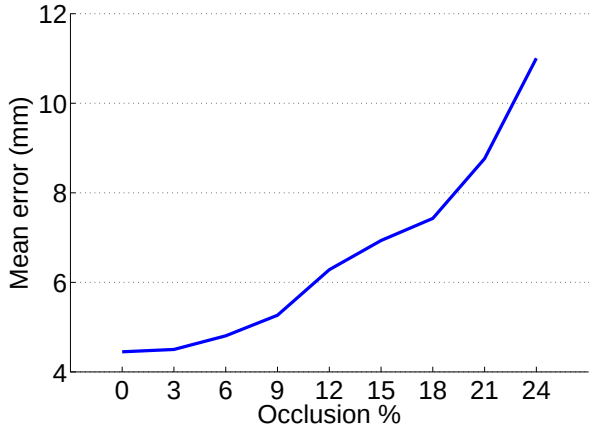


Fig. 31 Average errors for the facial features localization task, applied to the synthetically occluded images from the $BU3DFE$ dataset. The curve is a functions of the percentage of face pixels occluded by the hand. Examples are shown in Figure 34, right.

ing reconstructions in the $B3D(AC)^2$ database, on the right, the hand-occluded renderings of the $BU3DFE$ dataset.

In order to qualitatively evaluate the performance of our algorithm, we also tested it on new subjects, directly as they were scanned by a structured light scanner similar to [66]. We used a forest trained on the full $B3D(AC)^2$ database. We asked the subjects to perform different motions, also partly occluding their face with their hands or sunglasses. As shown in Figure 33, the results are robust to such occlusions. The video also shows how the algorithm is able to run in real time, at around 15 frames per second on a computer equipped with a 2GHz processor and 2GB of RAM, acquiring

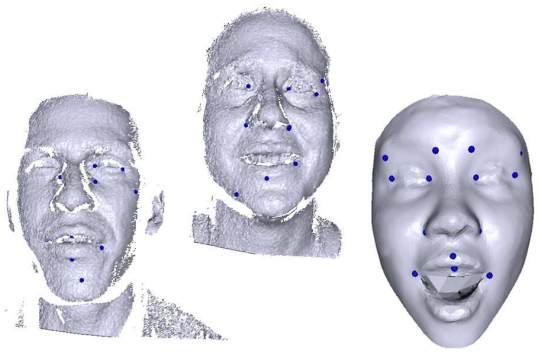


Fig. 32 Examples failure cases for the facial feature detector. The mouth feature points and the brow's endpoints are the fiducial most often misplaced.

the range scans while estimating the 3D locations of the facial features.

5 Conclusions

We have proposed a fast and robust framework based on random forests for real time head movement analysis. Intuitive parameters like number of trees and sampling stride provide straight-forward tools for adapting the system to different levels of computing power availability. We described in details its application for head pose estimation using both high quality range scans and low resolution depth images, and for 3D facial features localization. Our method runs on a frame-to-frame basis and therefore does not suffer from the usual shortcomings of tracking approaches, lending itself as a valuable tool for (re-)initialization of such methods.

We have demonstrated the accuracy and robustness of the proposed method on challenging and realistic datasets which are available to the community. Moreover, for our experiments on real time head pose estimation from consumer depth cameras, we acquired and annotated a new database containing different subjects rotating their heads, recorded using a Microsoft Kinect, which we made available for download.

Our framework relies on the abundance of annotated training depth data. New and more realistic training databases are required, covering all the scenarios which should be expected at test time. In our future work, we intend to train on full upper body models instead of isolated faces in order to better handle hair and other non-face body parts. Synthesis of such databases is very challenging due to the need of generating different hair styles, facial expressions, and head-wears. On the other hand, acquiring and annotating real-life scenes, to be used for testing new algorithms, would probably prove even more challenging.

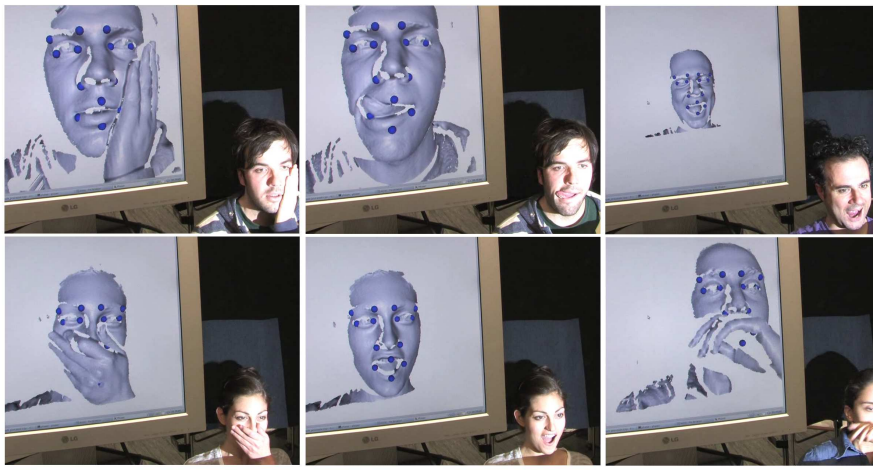


Fig. 33 Qualitative results, on subjects not present in the dataset, of the system running in real time.

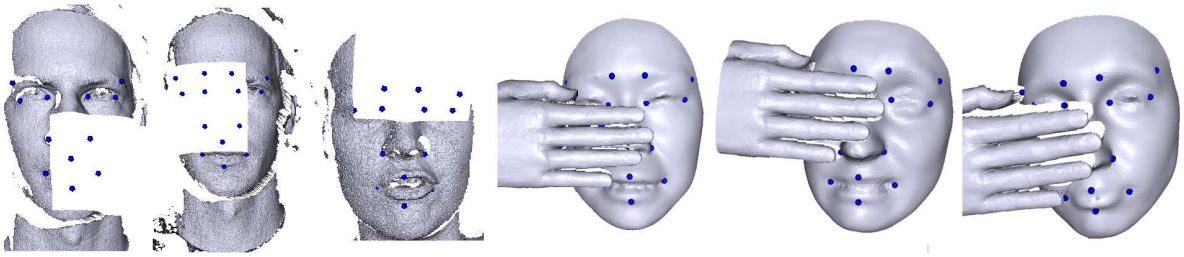


Fig. 34 Some badly occluded test images where our algorithm still manages to predict plausible locations of the feature points.

The use of depth data solves many of the inherent problems of standard images, however, is bounded by the availability of such sensors. Even though prices have recently dropped, the distribution of depth cameras is still limited compared to standard video recording devices and most have problems in outdoor scenarios. Our recent work [21] shows how to join real time head pose estimation and facial features localization for 2D images of faces acquired "in the wild".

Acknowledgements We thank Thibaut Weise for useful code and discussions. We acknowledge financial support from EU projects RADHAR (FP7-ICT-248873) and TANGO (FP7-ICT-249858), and from the SNF project Vision-supported Speech-based Human Machine Interaction (200021-130224).

References

1. Amberg, B., Vetter, T.: Optimal landmark detection using shape models and branch and bound slides. In: International Conference on Computer Vision (2011)
2. Balasubramanian, V.N., Ye, J., Panchanathan, S.: Biased manifold embedding: A framework for person-independent head pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2007)
3. Belhumeur, P.N., Jacobs, D.W., Kriegman, D.J., Kumar, N.: Localizing parts of faces using a consensus of exemplars. In: IEEE Conference on Computer Vision and Pattern Recognition (2011)
4. Besl, P., McKay, N.: A method for registration of 3-d shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence **14**(2), 239–256 (1992)
5. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), pp. 187–194 (1999)
6. Breidt, M., Buelthoff, H., Curio, C.: Robust semantic analysis by synthesis of 3d facial motion. In: Automatic Face and Gesture Recognition (2011)
7. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
8. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA (1984)
9. Breitenstein, M.D., Jensen, J., Hoilund, C., Moeslund, T.B., Van Gool, L.: Head pose estimation from passive stereo images. In: Scandinavian Conference on Image Analysis (2009)
10. Breitenstein, M.D., Kuettel, D., Weise, T., Van Gool, L., Pfister, H.: Real-time face pose estimation from single range images. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
11. Cai, Q., Gallup, D., Zhang, C., Zhang, Z.: 3d deformable face tracking with a commodity depth camera. In: European Conference on Computer Vision (2010)
12. Chang, K.I., Bowyer, K.W., Flynn, P.J.: Multiple nose region matching for 3d face recognition under varying facial expression. IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(10), 1695–1700 (2006)

13. Chen, L., Zhang, L., Hu, Y., Li, M., Zhang, H.: Head pose estimation using fisher manifold learning. In: Analysis and Modeling of Faces and Gestures (2003)
14. Chua, C.S., Jarvis, R.: Point signatures: A new representation for 3d object recognition. International Journal of Computer Vision **25**, 63–85 (1997)
15. Colbry, D., Stockman, G., Jain, A.: Detection of anchor points for 3d face verification. In: IEEE Conference on Computer Vision and Pattern Recognition (2005)
16. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**, 681–685 (2001)
17. Cootes, T.F., Wheeler, G.V., Walker, K.N., Taylor, C.J.: View-based active appearance models. Image and Vision Computing **20**(9-10), 657 – 664 (2002)
18. Criminisi, A., Shotton, J., Konukoglu, E.: Decision forests for classification, regression, density estimation, manifold learning and semi-supervised learning. Tech. Rep. TR-2011-114, Microsoft Research (2011)
19. Criminisi, A., Shotton, J., Robertson, D., Konukoglu, E.: Regression forests for efficient anatomy detection and localization in ct studies. In: Recognition techniques and applications in medical imaging (2010)
20. Cristinacce, D., Cootes, T.: Automatic feature localisation with constrained local models. Journal of Pattern Recognition **41**(10), 3054–3067 (2008)
21. Dantone, M., Gall, J., Fanelli, G., Van Gool, L.: Real-time facial feature detection using conditional regression forests. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
22. Dorai, C., Jain, A.K.: COSMOS - A Representation Scheme for 3D Free-Form Objects. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**(10), 1115–1130 (1997)
23. Everingham, M., Sivic, J., Zisserman, A.: Hello! my name is... buffy - automatic naming of characters in tv video. In: British Machine Vision Conference (2006)
24. Fanelli, G., Gall, J., Romsdorfer, H., Weise, T., Van Gool, L.: A 3-d audio-visual corpus of affective communication. IEEE Transactions on Multimedia **12**(6), 591 – 598 (2010)
25. Fanelli, G., Gall, J., Van Gool, L.: Real time head pose estimation with random regression forests. In: IEEE Conference on Computer Vision and Pattern Recognition (2011)
26. Fanelli, G., Weise, T., Gall, J., Van Gool, L.: Real time head pose estimation from consumer depth cameras. In: German Association for Pattern Recognition (2011)
27. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. International Journal of Computer Vision **61**(1), 55–79 (2005)
28. Gall, J., Lempitsky, V.: Class-specic hough forests for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2009)
29. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2011)
30. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. International Conference on Computer Vision (2011)
31. Gross, R., Matthews, I., Baker, S.: Generic vs. person specific active appearance models. Image and Vision Computing **23**(12), 1080 – 2093 (2005)
32. Huang, C., Ding, X., Fang, C.: Head pose estimation based on random forests for multiclass classification. In: International Conference on Pattern Recognition (2010)
33. Jones, M., Viola, P.: Fast multi-view face detection. Tech. Rep. TR2003-096, Mitsubishi Electric Research Laboratories (2003)
34. Ju, Q., O'keefe, S., Austin, J.: Binary neural network based 3d facial feature localization. In: International Joint Conference on Neural Networks (2009)
35. Kakadiaris, I.A., Passalis, G., Toderici, G., Murtuza, M.N., Lu, Y., Karampatziakis, N., Theoharis, T.: Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(4), 640–649 (2007)
36. Leibe, B., Leonardis, A., Schiele, B.: Robust object detection with interleaved categorization and segmentation. International Journal of Computer Vision **77**(1-3), 259–289 (2008)
37. Lepetit, V., Lagger, P., Fua, P.: Randomized trees for real-time keypoint recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2005)
38. Li, H., Adams, B., Guibas, L.J., Pauly, M.: Robust single-view geometry and motion reconstruction. ACM Transactions on Graphics (Proceedings SIGGRAPH Asia 2009) **28**(5) (2009)
39. Lu, X., Jain, A.K.: Automatic feature extraction for multiview 3d face recognition. In: Automatic Face and Gesture Recognition (2006)
40. Martins, P., Batista, J.: Accurate single view model-based head pose estimation. In: Automatic Face and Gesture Recognition (2008)
41. Matthews, I., Baker, S.: Active appearance models revisited. International Journal of Computer Vision **60**(2), 135–164 (2003)
42. Mehryar, S., Martin, K., Plataniotis, K., Stergiopoulos, S.: Automatic landmark detection for 3d face image processing. In: Evolutionary Computation (2010)
43. Mian, A., Bennamoun, M., Owens, R.: Automatic 3d face detection, normalization and recognition. In: 3D Data Processing, Visualization, and Transmission (2006)
44. Morency, L.P., Sundberg, P., Darrell, T.: Pose estimation using 3d view-based eigenspaces. In: Automatic Face and Gesture Recognition (2003)
45. Morency, L.P., Whitehill, J., Movellan, J.R.: Generalized adaptive view-based appearance model: Integrated framework for monocular head pose estimation. In: Automatic Face and Gesture Recognition (2008)
46. Mpiperis, I., Malassiotis, S., Strintzis, M.: Bilinear models for 3-d face and facial expression recognition. IEEE Transactions on Information Forensics and Security **3**(3), 498 –511 (2008)
47. Murphy-Chutorian, E., Trivedi, M.: Head pose estimation in computer vision: A survey. Transactions on Pattern Analysis and Machine Intelligence **31**(4), 607–626 (2009)
48. Nair, P., Cavallaro, A.: 3-d face detection, landmark localization, and registration using a point distribution model. IEEE Transactions on Multimedia **11**(4), 611–623 (2009)
49. Okada, R.: Discriminative generalized hough transform for object dectection. In: International Conference on Computer Vision (2009)
50. Osadchy, M., Miller, M.L., LeCun, Y.: Synergistic face detection and pose estimation with energy-based models. In: Neural Information Processing Systems (2005)
51. Papageorgiou, C., Oren, M., Poggio, T.: A general framework for object detection. In: International Conference on Computer Vision (1998)

52. Paysan, P., Knothe, R., Amberg, B., Romdhani, S., Vetter, T.: A 3d face model for pose and illumination invariant face recognition. In: Advanced Video and Signal based Surveillance (2009)
53. Ramnath, K., Koterba, S., Xiao, J., Hu, C., Matthews, I., Baker, S., Cohn, J., Kanade, T.: Multi-view aam fitting and construction. International Journal of Computer Vision **76**(2), 183–204 (2008)
54. Seemann, E., Nickel, K., Stiefelhagen, R.: Head pose estimation using stereo vision for human-robot interaction. Automatic Face and Gesture Recognition (2004)
55. Segundo, M., Silva, L., Bellon, O., Queirolo, C.: Automatic face segmentation and facial landmark detection in range images. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **40**(5), 1319 –1330 (2010)
56. Sharp, T.: Implementing Decision Trees and Forests on a GPU. In: European Conference on Computer Vision (2008)
57. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from single depth images. In: IEEE Conference on Computer Vision and Pattern Recognition (2011)
58. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
59. Storer, M., Urschler, M., Bischof, H.: 3d-mam: 3d morphable appearance model for efficient fine head pose estimation from still images. In: Workshop on Subspace Methods (2009)
60. Sun, Y., Yin, L.: Automatic pose estimation of 3d facial models. In: International Conference on Pattern Recognition (2008)
61. Valstar, M., Martinez, B., Binefa, X., Pantic, M.: Facial point detection using boosted regression and graph models. In: IEEE Conference on Computer Vision and Pattern Recognition (2010)
62. Vatahska, T., Bennewitz, M., Behnke, S.: Feature-based head pose estimation from images. In: International Conference on Humanoid Robots (2007)
63. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision **57**(2), 137–154 (2004)
64. Wang, Y., Chua, C., Ho, Y.: Facial feature detection and face recognition from 2d and 3d images. Pattern Recognition Letters **10**(23), 1191–1202 (2002)
65. Weise, T., Bouaziz, S., Li, H., Pauly, M.: Realtime performance-based facial animation. ACM International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH) (2011)
66. Weise, T., Leibe, B., Van Gool, L.: Fast 3d scanning with automatic motion compensation. In: IEEE Conference on Computer Vision and Pattern Recognition (2007)
67. Weise, T., Li, H., Van Gool, L., , Pauly, M.: Face/off: Live facial puppetry. In: Symposium on Computer Animation (2009)
68. Weise, T., Wismer, T., Leibe, B., Van Gool, L.: In-hand scanning with online loop closure. In: 3-D Digital Imaging and Modeling (2009)
69. Whitehill, J., Movellan, J.R.: A discriminative approach to frame-by-frame head pose tracking. In: Automatic Face and Gesture Recognition (2008)
70. Yao, A., Gall, J., Van Gool, L.: A hough transform-based voting framework for action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (2010)
71. Yin, L., Wei, X., Sun, Y., Wang, J., Rosato, M.J.: A 3d facial expression database for facial behavior research. In: Face and Gesture Recognition (2006)
72. Yu, T.H., Moon, Y.S.: A novel genetic algorithm for 3d facial landmark localization. In: Biometrics: Theory, Applications and Systems (2008)
73. Zhao, X., Dellandréa, E., Chen, L., Kakadiaris, I.: Accurate landmarking of three-dimensional facial data in the presence of facial expressions and occlusions using a three-dimensional statistical facial feature model. IEEE transactions on systems, man, and cybernetics, Part B: Cybernetics **41**(5), 1417–1428 (2011)