Real Time Head Pose Estimation with Random Regression Forests

Gabriele Fanelli[1] Juergen Gall[1] Luc Van Gool[1,2]
[1]BIWI, ETH Zurich [2]ESAT-PSI / IBBT, KU Leuven
{fanelli,gall,vangool}@vision.ee.ethz.ch vangool@esat.kuleuven.be

Abstract

Fast and reliable algorithms for estimating the head pose are essential for many applications and higher-level face analysis tasks. We address the problem of head pose estimation from depth data, which can be captured using the ever more affordable 3D sensing technologies available today. To achieve robustness, we formulate pose estimation as a regression problem. While detecting specific face parts like the nose is sensitive to occlusions, learning the regression on rather generic surface patches requires enormous amount of training data in order to achieve accurate estimates. We propose to use random regression forests for the task at hand, given their capability to handle large training datasets. Moreover, we synthesize a great amount of annotated training data using a statistical model of the human face. In our experiments, we show that our approach can handle real data presenting large pose changes, partial occlusions, and facial expressions, even though it is trained only on synthetic neutral face data. We have thoroughly evaluated our system on a publicly available database on which we achieve state-of-the-art performance without having to resort to the graphics card.

1. Introduction

Automatic and robust algorithms for head pose estimation can be beneficial to many real life applications. Accurately localizing the head and its orientation is either the explicit goal of systems like human-computer interfaces (*e.g.*, reacting to the user's head movements), or a necessary preprocessing step for further analysis, such as identification or facial expression recognition. Due to its relevance and to the challenges posed by the problem, there has been considerable effort in the computer vision community to develop fast and reliable algorithms for head pose estimation.

Methods relying solely on standard 2D images face serious problems, notably illumination changes and textureless face regions. Given the recent development and availability of 3D sensing technologies, which are becoming ever more affordable and reliable, the additional depth information can finally allow us to overcome some of the problems inherent of methods based on 2D data. However, existing depth-based methods either need manual initialization, cannot handle large pose variations, or are not real-time. An exception are approaches like the one presented by [4], where the authors achieve real-time performance by exploiting the massive parallel processing power of a GPU. Their approach relies on a geometric descriptor which provides nose location hypotheses which are then compared to a large number of renderings of a generic face template, done in parallel on the GPU. The fast computation time reported is only achievable provided that specific graphics hardware is available.

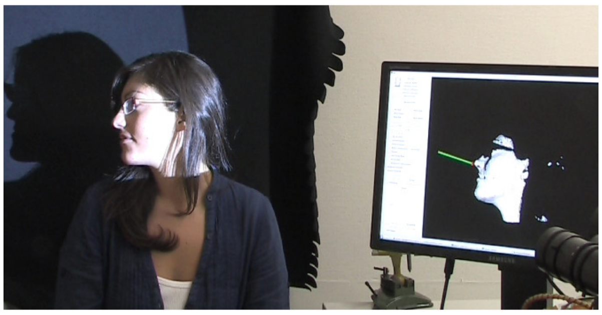
Figure 1. Real time head pose estimation example.

GPUs, however, present a very high power consumption which limits their use for certain kinds of application. Hence, we propose an approach for 3D head pose estimation which does not rely on specific graphics hardware and which can be tuned to achieve the desired trade-off between accuracy and computation cost, which is particularly useful when resources are limited by the application. We formulate the problem as a regression, estimating the head pose parameters directly from the depth data. The regression is implemented within a random forest framework [2, 10], learning a mapping from simple depth features to a probabilistic estimation of real-valued parameters such as 3D nose coordinates and head rotation angles. Since random forests (as any regressor) need to be trained on labeled data and the accuracy depends on the amount of training, data acquisition is a key issue. We solve this problem by training only on synthetic data, generating an arbitrary number of training examples without the need of laborious and error-prone annotations. Our system works in real-time on a frame-by-frame basis, without any manual initialization or expensive calculations. In our experiments, we show that it works for unseen faces and can handle large pose changes, variations such as facial hair, and partial occlusions, *e.g.*, due to glasses, hands, or missing parts in the 3D reconstruction. Moreover, as it does not rely on specific features, *e.g.*, for the nose tip detection, our method can be adapted to the localization of other parts of the face. The performance of the system is evaluated on a challenging publicly available database and our results are comparable or superior to the state-of-the-art.

2. Related Work

Head pose estimation is the goal of several works in the literature [19]. Existing methods can conveniently be divided depending on the type of data they rely on, *i.e.*, 2D images or depth data. Within the 2D image-based algorithms, we can further distinguish between appearance-based and feature-based methods. While the former look at the entire face region in the image, the latter rely on the localization of specific facial feature points.

A common appearance-based approach is to discretize the head poses and learn a separate detector for each pose, *e.g.*, [13, 18]. Approaches like [1, 6] focus on the mapping from the high-dimensional space of facial images into lower-dimensional, smooth manifolds; Osadchy *et al.* [21], for example, use a convolutional network, detecting faces and their orientation in real-time. Several works rely on statistical models of the face shape and appearance, *e.g.*, Active Appearance Models (AAMs) [8] and their extensions [9, 23, 25], but their focus is usually on detection and tracking of facial features.

Feature-based methods need either the same facial features to be visible in all poses, *e.g.*, [26, 29, 16], or use pose-dependent features; for example, Yao and Cham [30] select feature points manually and match them to a generic wireframe model. The authors of [28] use a combination of the face appearance and a set of specific feature points, which bounds the range of recognizable poses to the ones where both eyes are visible.

In general, methods relying solely on 2D images are sensitive to illumination, lack of features, and partial occlusions. Moreover, the annotation of head poses from 2D images is an error-prone task in itself. Fortunately, recent 3D technologies have achieved high quality at affordable costs, *e.g.*, [27]. The additional depth information can help in solving some of the limitations of image-based methods, therefore several recent works use depth either as primary cue [4, 15] or as an addition to standard 2D images [5, 18, 24]. Seemann *et al.* [24] presented a neural network-based system fusing skin color histograms and depth information. It runs at 10 fps but requires the face to be first detected in frontal pose. The work of [5] uses a linear deformable face model for real-time tracking of the head pose and facial movements using depth and appearance cues. Their system focuses on tracking facial features and thus no evaluation is presented for its head pose tracking performance. The approach presented in [17] uses head pose estimation only as a preprocessing step to face recognition, and the reported errors are only calculated on faces belonging to the same people. Breitenstein *et al.* [4] proposed a real-time system which can handle large pose variations, partial occlusions (as long as the nose remains visible), and facial expressions from range images. The method uses geometric features to generate nose candidates which suggest many head position hypotheses. Thanks to the massive parallel computation power of the GPU, they can simultaneously compare all suggested poses to a generic face template previously rendered in many different orientations and finally choose the pose minimizing a predefined cost function. Also the authors of [15] use range images and rely on the localization of the nose; however, their reported results are computed on a database generated by synthetically rotating frontal scans of several subjects.

Random forests [2] have become a popular method in computer vision [11, 10, 20, 14, 12] given their capability to handle large training datasets, high generalization power, fast computation, and ease of implementation. Recent works showed the power of random forests in mapping image features to votes in a generalized Hough space [11] or to real-valued functions [10, 12]. Recently, multiclass random forests have been proposed in [12] for real-time head pose recognition from 2D video data. To the best of our knowledge, we present the first approach that uses random regression forests for the task of head pose estimation from range data.

3. Head Pose Estimation with Random Regression Forests

Our goal is to jointly estimate the 3D coordinates of the nose tip and the angles of rotation of a range image of a head, *i.e.*, 2.5D data output of a range scanner like [27]. We use a random regression forest (Sec. 3.1), trained as explained in Sec. 3.2 on a large dataset of synthetically generated range images of faces (Sec. 3.4). The way the actual regression is performed is explained in Sec. 3.3.

3.1. Random Regression Forests

Classification and regression trees [3] are powerful tools capable of mapping complex input spaces into discrete or respectively continuous output spaces. A tree achieves highly non-linear mappings by splitting the original problem into smaller ones, solvable with simple predictors. Each node in the tree consists of a test, whose result directs a data sample towards the left or the right child. During training, the tests are chosen in order to group the training data in clusters where simple models achieve good predictions. Such models are stored at the leaves, computed from the annotated data which reached each leaf at train time.

Breiman [2] shows that, while standard decision trees alone suffer from overfitting, a collection of randomly trained trees has high generalization power. Random forests are thus ensembles of trees trained by introducing randomness either in the set of examples provided to each tree, in the set of tests available for optimization at each node, or in both. Figure 2 shows a very simple example of the regression forest used in this work.

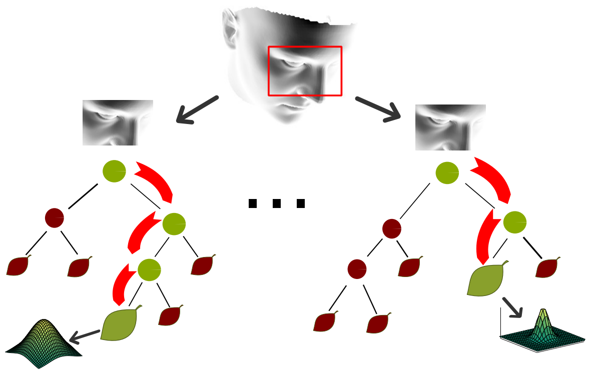

Figure 2. Example of regression forest. For each tree, the tests at the non-leaf nodes direct an input sample towards a leaf, where a real-valued, multivariate distribution of the output parameters is stored. The forest combines the results of all leaves to produce a probabilistic prediction in the real-valued output space.

3.2. Training

The learning is supervised, *i.e.*, training data is annotated with values in $\mathbb{R}^D$, where D is the dimensionality of the desired output. In our setup, training examples consist of range images of faces annotated with 3D nose location and head rotation angles. We limit ourselves to the problem of estimating the head pose, thus assume that the head has been already detected in the image. However, a random forest could be trained to jointly estimate the head position in the range image together with the pose, as in [11, 20].

Each tree T in the forest $\mathcal{T} = \{T_t\}$ is constructed from a set of patches $\left\{\mathcal{P}_i = \left(\mathcal{I}_i^f, \boldsymbol{\theta}_i\right)\right\}$ randomly sampled from the training examples. $\mathcal{I}_i^f$ are the extracted visual features for a patch of fixed size; in the current setup, we use one to four feature channels, namely depth values and, optionally, the X, Y, and Z values of the geometric normals computed over neighboring, non-border pixels. The real-valued vector $\boldsymbol{\theta}_i = \{\theta_x, \theta_y, \theta_z, \theta_{yaw}, \theta_{pitch}, \theta_{roll}\}$ contains the pose parameters associated to each patch. The components θ_x, θ_y, and θ_z represent an offset vector from the point in the range scan falling on the center of the training patch to the nose position in 3D, while θ_{yaw}, θ_{pitch}, and θ_{roll} are the head rotation angles denoting the head orientation.

We build the trees following the random forest framework [2]. At each non-leaf node, starting from the root, a test is selected from a large, randomly generated set of possible binary tests. The binary test at a non-leaf node is defined as $t_{f,F_1,F_2,\tau}(\mathcal{I})$:

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

where I^f indicates the feature channel, F_1 and F_2 are two rectangles within the patch boundaries, and τ is a threshold. The test splits the training data into two sets: When a patch satisfies the test it is passed to the right child, otherwise, the patch is sent to the left child. We chose to take the difference between the average values of two rectangular areas (as the authors of [10]) rather than single pixel differences (as in [11]) in order to be less sensitive to noise. Figure 3 shows a patch (marked in red) and the two randomly generated regions F_1 and F_2 as part of a binary test; the arrow indicates the 3D offset vector stretching from the patch center (in red) to the annotated nose location (green).

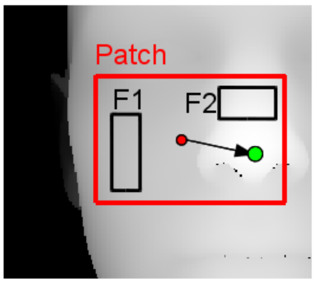

Figure 3. Example of a training patch (larger, red rectangle) with its associated offset vector (arrow) between the 3D point falling at the patch's center (red dot) and the ground truth location of the nose (marked in green). The rectangles F1 and F2 represent a possible choice for the regions over which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

During the construction of the tree, at each non-leaf node, a pool of binary tests $\left\{t^k\right\}$ is generated with random values for f, F_1, F_2, and τ. The set of patches arriving at the node is evaluated by all binary tests in the pool and the test maximizing a predefined measure is assigned to the node. Following [10], we optimize the trees by maximizing the information gain defined as the differential entropy of the set of patches at the parent node $\mathcal{P}$ minus the weighted sum of the differential entropies computed at the children $\mathcal{P}_L$ and $\mathcal{P}_R$:

$$IG = H(\mathcal{P}) - (w_L H(\mathcal{P}_L) + w_R H(\mathcal{P}_R)), \quad (2)$$

where $\mathcal{P}_{i\in\{L,R\}}$ is the set of patches reaching node i and w_i is the ratio between the number of patches in i and in its parent node, *i.e.*, $w_i = \frac{|\mathcal{P}_i|}{|\mathcal{P}|}$.

We model the vectors $\boldsymbol{\theta}$ at each node as realizations of a random variable with a multivariate Gaussian distribution, *i.e.*, $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}; \overline{\boldsymbol{\theta}}, \boldsymbol{\Sigma})$. Therefore, Eq. (2) can be rewritten as:

$$IG = \log|\boldsymbol{\Sigma}(\mathcal{P})| - \sum_{i\in\{L,R\}} w_i \log|\boldsymbol{\Sigma}_i(\mathcal{P}_i)|. \quad (3)$$

Maximizing Eq. (3) favors tests which minimize the determinant of the covariance matrix $\boldsymbol{\Sigma}$, thus decreasing the uncertainty in the votes for the output parameters cast by each patch cluster.

We assume the covariance matrix to be block-diagonal $\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}^v & 0 \\ 0 & \boldsymbol{\Sigma}^a \end{pmatrix}$, *i.e.*, we allow covariance only among offset vectors ($\boldsymbol{\Sigma}^v$) and among head rotation angles ($\boldsymbol{\Sigma}^a$), but not between them. Eq. (3) thus becomes:

$$IG = \log(|\boldsymbol{\Sigma}^v| + |\boldsymbol{\Sigma}^a|) - \sum_{i\in\{L,R\}} w_i \log(|\boldsymbol{\Sigma}_i^v| + |\boldsymbol{\Sigma}_i^a|). \quad (4)$$

A leaf l is created when the maximum depth is reached or a minimum number of patches are left. Each leaf stores the mean of all angles and offset vectors which reached it, together with their covariance, *i.e.*, a multivariate Gaussian distribution.

3.3. Testing

Given a new, unseen range image of a head, patches (of the same size as the ones used for training) are densely sampled and passed through all trees in the forest. At each node of a tree, the stored binary test evaluates a patch, sending it either to the right or left child, all the way down until a leaf. Arrived at a leaf l, a patch gives an estimate for the pose parameters in terms of the stored distribution $p(\boldsymbol{\theta}|l) = \mathcal{N}(\boldsymbol{\theta}; \overline{\boldsymbol{\theta}}, \boldsymbol{\Sigma})$.

Because leaves with a high variance are not very informative and add mainly noise to the estimate, we discard all Gaussians with a total variance greater than an empiric threshold max_v. We also first locate the nose position to remove outliers before estimating all other parameters. We thus perform 10 mean-shift [7] iterations using a spherical kernel and keep only the random variables whose means fall within the mean-shift kernel. The kernel radius is defined as a fraction of the size of the smallest sphere enclosing the average human face, which we consider to be the mean of the PCA model constructed from aligned range scans of many different people [22]. The initialization for the mean-shift step is the mean of all votes returned by the forest, which we assume to be close to the true nose location, where most of the votes usually cluster. In our experiments, removing outliers has shown to be crucial in test images where the head undergoes large rotations and/or is partially occluded by glasses or facial hair. We finally sum all the remaining random variables θ, producing a Gaussian whose mean is the ultimate estimate of our output parameters and whose covariance represents a measure of the estimate's uncertainty. An example test frame is shown in Fig. 4, where the small blue spheres are all votes cast by the forest, and the green, larger ones represent the votes selected after mean-shift. The final estimate of the nose position and head direction is represented by the green cylinder.

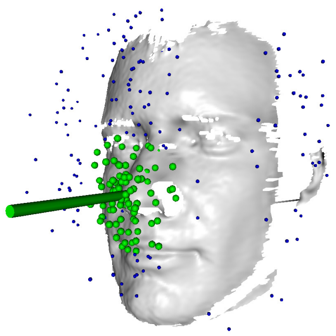
Figure 4. Example test image: the green spheres are the ones selected after the filtering of the outliers (blue spheres) by mean shift. The large green cylinder stretches from the final estimate of the nose center in the estimated face direction.

3.4. Training Data Generation

Random forests can be built from large training datasets in reasonable time and are very powerful in learning the most distinctive features for the problem at hand. We therefore generated a large database of 50K, 640x480 range images of faces by rendering a 3D morphable model [22] in many randomly generated poses. The rotations span $\pm 95^\circ$ for yaw, $\pm 50^\circ$ for pitch, and $\pm 20^\circ$ for roll. Moreover, we randomly translated the 3D model along the z axis within a 50 cm range. We further perturbed the first 30 PCA modes of the shape model by ± 2 of the standard deviation of each mode, thus introducing random variations also in the identity of the faces. We stored the 3D coordinates of each visible point on the face surface for each pixel in each image, together with the ground truth 3D coordinates of the nose tip and the three rotation angles. Figure 5 shows a few of the training faces, with the red bar pointing out from the nose indicating the head direction.

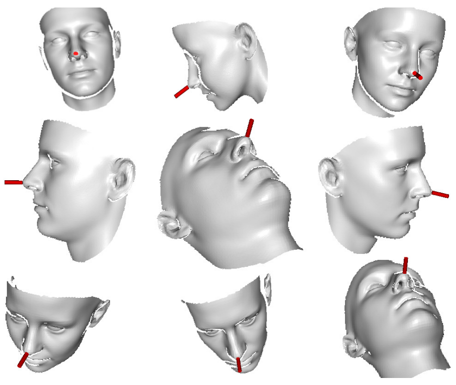

Figure 5. Sample images from our synthetically generated training set. The heads show large 3D rotations and variations in the distance from the camera and also in identity. The red cylinder attached to the nose represents the ground truth orientation of the face.

4. Experiments

In order to assess the performance of our algorithm on realistic data, we used the ETH Face Pose Range Image Data Set [4]. The database contains over 10K range images of 20 people (3 females, 6 subjects recorded twice, with and without glasses) turning their head while captured at 28 fps by the range scanner of [27]. The images have a resolution of 640x480 pixels, and a face typically consists of around 150x200 pixels. The head pose range covers about $\pm 90\,^\circ$ yaw and $\pm 45\,^\circ$ pitch rotations. The provided ground truth for each image consists of the 3D nose tip coordinates and the coordinates of a vector pointing in the face direction.

Our system outputs real-valued estimates for the 3D nose location and the three head rotation angles. However, the above database does not contain roll rotations, which are not encoded in the directional vector provided as ground truth. Therefore, we evaluated our performance with regard to orientation by computing the head direction vector from the estimated yaw and pitch angles and report the angular error with the vector provided as ground truth. For the nose localization, the Euclidian distance is used as error measure.

Our method is controlled by a few parameters. In the present setup, some parameters have been fixed intuitively, like the size of the patches (120x80 pixels), the maximum size of the sub-patches defining the areas F_1 and F_2 in the tests (fixed to 40 pixels), and the number of mean-shift iterations used to remove outliers from the set of random variables obtained from the regression forest, set to 10. Additional parameters are the maximum accepted variance max_v of the Gaussians output of the leaves and the ratio defining the radius of the mean-shift spherical kernel. In order to find the best combination of these last two parameters, we used a small part of the ETH database (1000 images) and ran a coarse grid search for the best average errors. We picked the values $max_v = 400$ and $ratio = 5$ as they showed the best compromise for the errors in both the nose localization and angle estimation task. Additionally, the number of trees to be used and a stride in the sampling of the patches for testing can be adapted to find the desired balance between accuracy and speed of the estimation process. This is a very important feature of our system since it allows to adapt the algorithm to the computational constraints imposed by the application.

In the following experiments, we always trained each tree sampling 25 patches from each of 3000 synthetically generated images, while the full ETH database was used for testing.

The plots in Fig. 6*(a,d)* show the time in ms needed to process one frame (once loaded into memory) using our method, depending on the number of loaded trees and on the stride. It is important to note that the current implementation is not optimized, and could greatly benefit from using integral images, possibly reducing the computation time of a factor of 2. Fig. 6*(a)* plots the average runtime (computed over 500 frames) for a stride fixed to 20 as a function of the number of trees, while in Fig. 6*(d)* 15 trees are loaded and the stride parameter changes. The red circular markers represent the performance of the system when all available feature channels are employed (*i.e.*, depth and geometric normals), while the blue crosses refer to the setup where only the depth channel is used. We used a computer with an Intel Core i7 CPU @ 2.67GHz, without resorting to multithreading. It can be seen that, for a number of trees less or equal to 15 and a stride greater or equal to 20, both curves are well below 40ms, *i.e.*, we achieve framerates above 25 frames per second. If not otherwise specified, the remaining results and images have been produced using these values for the number of trees and the stride.

The additional computations needed to extract the normals (a 4-neighborhood is used - all done on the CPU) pay off in terms of estimation accuracy, as visualized in the Figs. 6*(b,c,e,f)*. The plot in Fig. 6*(b)* shows the percentage of correctly estimated images as a function of the success threshold set for the angular error, while in Fig. 6*(e)* the thresholds are defined on the nose localization error. As can be seen in all plots, using all the available feature channels performs consistently better than taking only the depth information. More in detail, for a conservative threshold of $10\,^\circ$ on the angular error, our method succeeded in 90.4% of the cases, 95.4% for $15\,^\circ$, and 95.9% for $20\,^\circ$. With the same thresholding and on the same dataset, [4] reports percentages of 80.8%, 97.8%, and 98.4%, *i.e.*, we have improved the accuracy of about 10% for the most conservative threshold, which is also the most relevant one. The performance for nose localization is shown in Fig. 6*(e)* where the success rate for thresholds between 10 and 35 mm is plotted, *e.g.*, setting a 20 mm threshold leads to 93.2% accuracy.

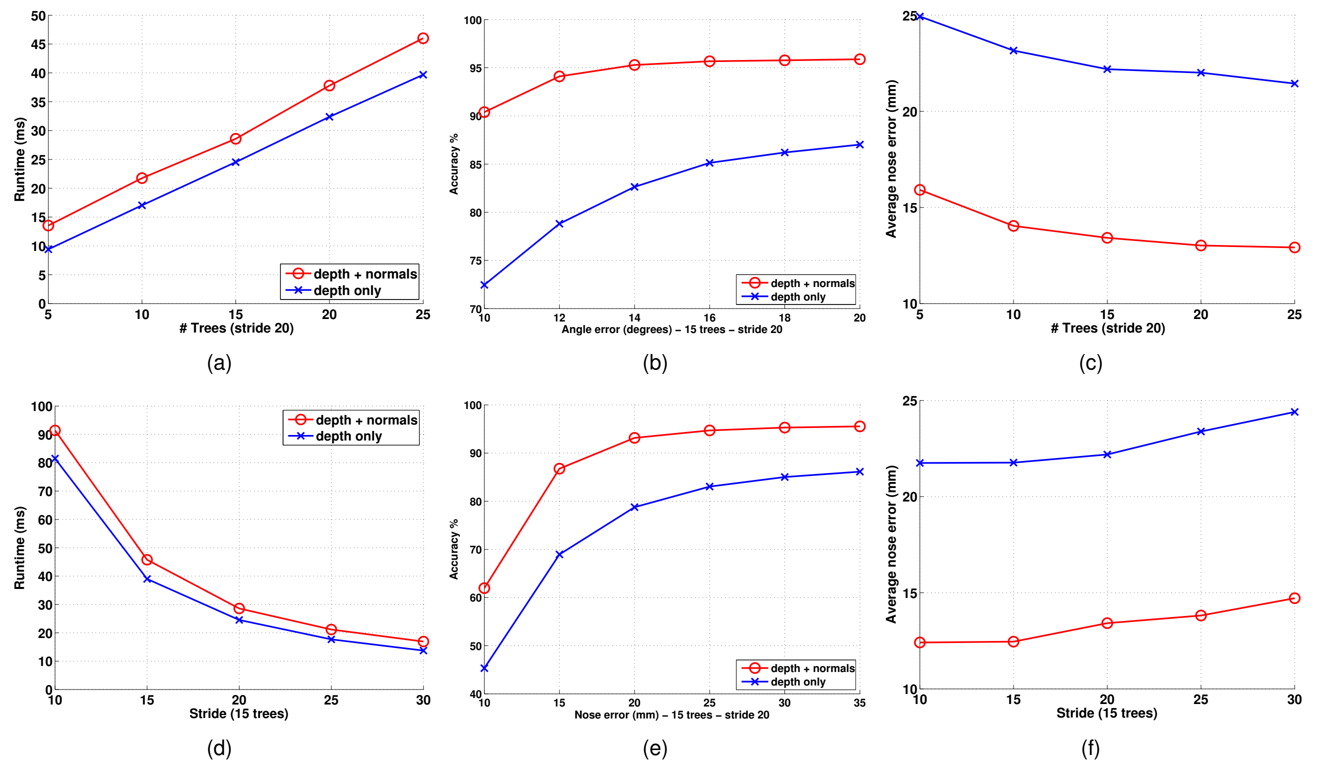

Figure 6. All plots are labeled in blue and crosses when only the depth channel is used by the system, and in red and circles when all four channels are used (depth plus X,Y, and Z components of the normals). a) Regression time as a function of the number of trees in the forest when the stride is fixed to 20 pixels, the reported values are averaged over 500 images. b) Accuracy of the system in terms of percentage of correctly estimated poses as a function of the angle error threshold. c) Average nose error in mm, as a function of the number of trees (with stride=20). d) Runtime of the system for a forest of 15 trees as a function of the stride parameter. e) Accuracy plotted against nose error thresholds in mm. f) Nose localization error for a forest of 15 trees and varying stride. In the current, unoptimized settings, the system achieves real-time performance for forests with $\leq$ 15 trees and a stride $\geq$ 20. The additional information coming from the normals consistently boosts the performance.

	Nose error	Yaw error	Pitch error	Direction estimation accuracy
Our approach	$13.4/21.1$ mm	$5.7/15.2\,^\circ$	$5.1/4.9\,^\circ$	90.4%
Breitenstein et al.	$9.0/14.0$ mm	$6.1/10.3\,^\circ$	$4.2/3.9\,^\circ$	80.8%

Table 1. Comparison of our results with [4]. The first three columns show mean and standard deviation for the Euclidean error in the nose tip localization task (mm) and for the yaw and pitch estimation (degrees). The values in the last column are the percentages of correctly estimated images for a threshold on the angular error (to the ground truth provided with the ETH database) of 10 degrees.

In 6.5% of the range images in the ETH database, our system did not return an estimate, *i.e.*, no votes were left under the mean-shift kernel; these images were discarded before computing the following average errors. Figs. 6*(c,f)* show the average error in the nose localization task, plotted as a function of the number of trees when the stride is fixed to 20 (6*(c)*) and as a function of the stride when 15 trees are used (6*(f)*). Again, using the normals in addition to the depth channel (red curve) shows considerably superior performance. The mean and standard deviation of the pose estimation error are reported in Table 1 and compared to the errors reported in [4]. However, the first three columns of Table 1 are not directly comparable, because the errors reported by [4] are computed only on estimates with a high nose detection confidence (positive rate of 80% and a false positive rate of 3%), but the percentage of retained images is not provided. The last column of Table 1 is instead a fair comparison of the two systems because both values were computed on the full ETH database, showing the percentage of correctly classified images assuming a $10\,^\circ$ threshold on the angular error.

Fig. 7 shows the success rate when the angle error threshold is set to $15\,^\circ$, and the nose error to 20 mm. In this case, we loaded 15 trees as before, but set the stride to 10. The test dataset was divided based on the heads' angles of rotations for pitch and yaw, in areas of $15\,^\circ \times 15\,^\circ$. Only areas

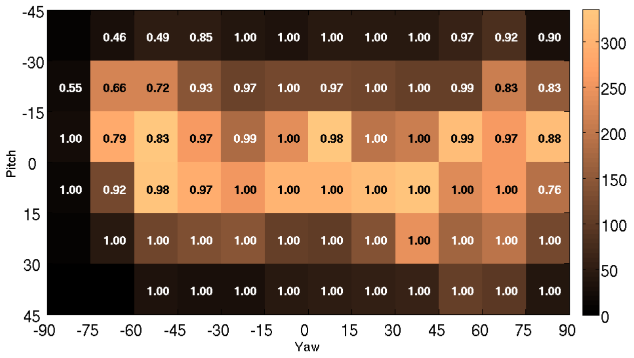

Figure 7. Normalized success rates of the estimation. The database was discretized in $15^\circ \times 15^\circ$ areas and the accuracy compute for each range of angles separately. The color encodes the number of images falling in each region, as explained by the bar on the side.

with more than 10 images were considered and are filled in the image; the color-coding represents the amount of data contained in the angle range of each area, as described by the color bar on the right. It can be noted how the results are close to 100% for the central region of the map, where most of the images in the ETH database fall. The performance usually decreases in the corner regions, where the rotations are very large, and the number of images low. In general, our results are comparable to a similar plot presented in [4].

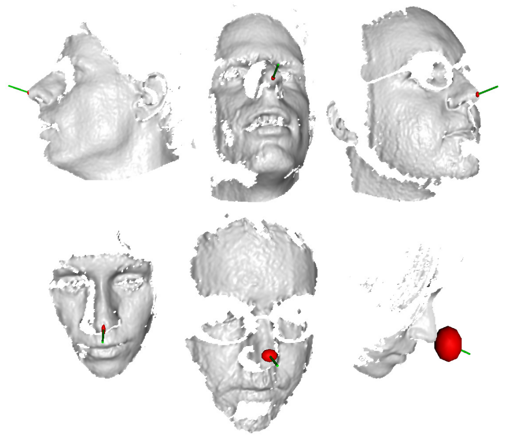

Figure 8. Correctly estimated poses from the ETH database. Large rotations, glasses, and facial hair do not pose major problems in most of the cases. The green cylinder represents the estimated head rotation, while the red ellipse is centered on the estimated 3D nose position and scaled according to the covariance provided by the forest (scaled by a factor of 10 to ease the visualization).

Fig. 8 shows some successfully processed frames from the ETH database. The red ellipse represents the estimated nose location and it is scaled according to the covariance output of the regression forest. The green cylinder is centered on the nose tip and stretches 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, which is not reconstructed by the scanner). Two example failures are rendered in Fig. 9.

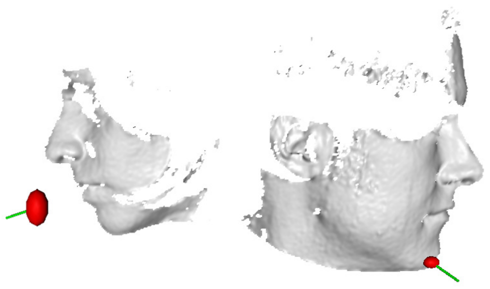

Figure 9. Example failure images from the ETH database. The large ellipse denotes that the regression predicted a high variance for the estimate of the nose location.

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 [27]. Fig. 10 shows some example frames, with our method successfully estimating the head pose even when the nose is badly occluded and thus most of the other approaches based on 3D data would fail. Facial expressions also do not seem to cause problems to the regression in most of the cases, even though the synthetic training dataset contains only neutral faces.

5. Conclusions

In this work, we have presented an approach for real-time head pose estimation from depth data. In our experiments, we have performed a thorough quantitative evaluation on a publicly available database where our approach achieves state-of-the-art performance. Because we do not rely on specific hardware like a GPU, our approach is also suitable for applications where hardware constraints limit the computational resources. In particular, the introduction of a stride for patch processing gives an easy-to-use parameter to steer the trade-off between accuracy and computation cost for an optimal setting. Furthermore, our approach is robust against severe occlusions, because we do not localize a few, specific facial features but estimate the pose parameters from all surface patches within a regression framework.

6. Acknowledgements

The authors acknowledge financial support from the SNF project Vision-supported Speech-based Human Machine Interaction (200021-130224) and the EU project RADHAR (FP7-ICT-248873). We also thank Thibaut Weise for providing the 3D scanner and useful code.

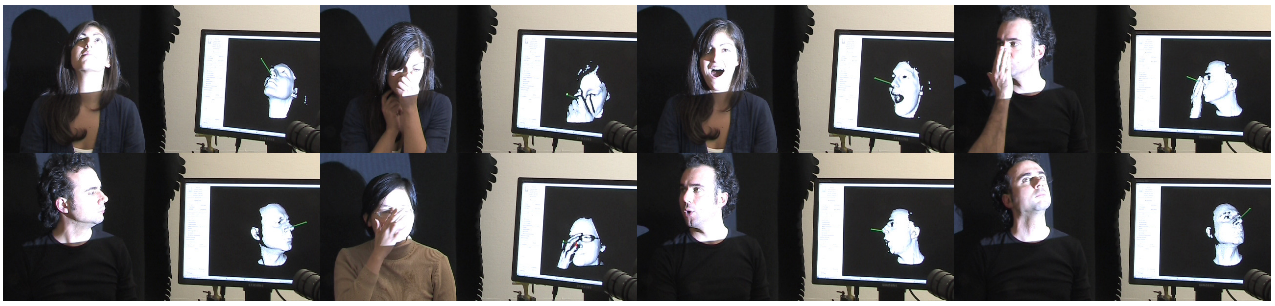
Figure 10. Example frames from our real-time head pose estimation system, showing how the regression works even in the presence of facial expressions and partial occlusions.

References

[1] V. N. Balasubramanian, J. Ye, and S. Panchanathan. Biased manifold embedding: A framework for person-independent head pose estimation. In *CVPR*, 2007.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[4] M. D. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister. Real-time face pose estimation from single range images. In *CVPR*, 2008.

[5] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang. 3d deformable face tracking with a commodity depth camera. In *ECCV*, 2010.

[6] L. Chen, L. Zhang, Y. Hu, M. Li, and H. Zhang. Head pose estimation using fisher manifold learning. In *Workshop on Analysis and Modeling of Faces and Gestures*, 2003.

[7] Y. Cheng. Mean shift, mode seeking, and clustering. *TPAMI*, 17:790–799, 1995.

[8] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *TPAMI*, 23:681–685, 2001.

[9] T. F. Cootes, G. V. Wheeler, K. N. Walker, and C. J. Taylor. View-based active appearance models. *Image and Vision Computing*, 20(9-10):657–664, 2002.

[10] A. Criminisi, J. Shotton, D. Robertson, and E. Konukoglu. Regression forests for efficient anatomy detection and localization in ct studies. In *Medical Computer Vision Workshop*, 2010.

[11] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 2011.

[12] C. Huang, X. Ding, and C. Fang. Head pose estimation based on random forests for multiclass classification. In *ICPR*, 2010.

[13] M. Jones and P. Viola. Fast multi-view face detection. Technical Report TR2003-096, Mitsubishi Electric Research Laboratories, 2003.

[14] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *TPAMI*, 28:1465–1479, 2006.

[15] S. Malassiotis and M. G. Strintzis. Robust real-time 3d head pose estimation from range data. *Pattern Recognition*, 38:1153 – 1165, 2005.

[16] Y. Matsumoto and A. Zelinsky. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. In *Aut. Face and Gestures Rec.*, 2000.

[17] A. Mian, M. Bennamoun, and R. Owens. Automatic 3d face detection, normalization and recognition. In *3DPVT*, 2006.

[18] L.-P. Morency, P. Sundberg, and T. Darrell. Pose estimation using 3d view-based eigenspaces. In *Aut. Face and Gestures Rec.*, 2003.

[19] E. Murphy-Chutorian and M. Trivedi. Head pose estimation in computer vision: A survey. *TPAMI*, 31(4):607–626, 2009.

[20] R. Okada. Discriminative generalized hough transform for object dectection. In *ICCV*, 2009.

[21] M. Osadchy, M. L. Miller, and Y. LeCun. Synergistic face detection and pose estimation with energy-based models. In *NIPS*, 2005.

[22] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *Advanced Video and Signal based Surveillance*, 2009.

[23] K. Ramnath, S. Koterba, J. Xiao, C. Hu, I. Matthews, S. Baker, J. Cohn, and T. Kanade. Multi-view aam fitting and construction. *IJCV*, 76:183–204, 2008.

[24] E. Seemann, K. Nickel, and R. Stiefelhagen. Head pose estimation using stereo vision for human-robot interaction. In *Aut. Face and Gestures Rec.*, 2004.

[25] M. Storer, M. Urschler, and H. Bischof. 3d-mam: 3d morphable appearance model for efficient fine head pose estimation from still images. In *Workshop on Subspace Methods*, 2009.

[26] T. Vatahska, M. Bennewitz, and S. Behnke. Feature-based head pose estimation from images. In *Humanoids*, 2007.

[27] T. Weise, B. Leibe, and L. Van Gool. Fast 3d scanning with automatic motion compensation. In *CVPR*, 2007.

[28] J. Whitehill and J. R. Movellan. A discriminative approach to frame-by-frame head pose tracking. In *Aut. Face and Gestures Rec.*, 2008.

[29] R. Yang and Z. Zhang. Model-based head pose tracking with stereovision. *Aut. Face and Gestures Rec.*, 2002.

[30] J. Yao and W. K. Cham. Efficient model-based linear head motion recovery from movies. In *CVPR*, 2004.