Feature-based head pose estimation from images

Conference Paper · November 2007

DOI: 10.1109/ICHR.2007.4813889 · Source: IEEE Xplore

Sven Behnke
University of Bonn
515 PUBLICATIONS **8,163** CITATIONS

Feature-based Head Pose Estimation from Images

Teodora Vatahska, Maren Bennewitz, and Sven Behnke
University of Freiburg
Computer Science Institute
D-79110 Freiburg, Germany
Email:{vatahska, maren, behnke}@informatik.uni-freiburg.de

***Abstract*— Estimating the head pose is an important capability of a robot when interacting with humans since the head pose usually indicates the focus of attention. In this paper, we present a novel approach to estimate the head pose from monocular images. Our approach proceeds in three stages. First, a face detector roughly classifies the pose as frontal, left, or right profile. Then, classifiers trained with AdaBoost using Haar-like features, detect distinctive facial features such as the nose tip and the eyes. Based on the positions of these features, a neural network finally estimates the three continuous rotation angles we use to model the head pose. Since we have a compact representation of the face using only few distinctive features, our approach is computationally highly efficient. As we show in experiments with standard databases as well as with real-time image data, our system locates the distinctive features with a high accuracy and provides robust estimates of the head pose.**

I. Introduction

Successful interaction with humans requires robust and accurate perception and tracking of their body parts to infer non-verbal signals of attention and intention. In order to establish a joint focus of attention [1], estimating the head pose is crucial since it usually coincides with the gaze direction. Furthermore, head pose estimation is also essential for analyzing complex meaningful gestures such as pointing gestures or head nodding and shaking.

In this paper, we present an approach to estimate the head pose from monocular images. We model the head pose in the three-dimensional space by three Euler angles of rotation around three axis orthogonal to each other (see Fig. 1). While most of the existing approaches for head pose estimation deal only with poses that vary around the vertical axis, our system provides continuous head pose estimates in all three rotation directions. We propose a method which is based on distinctive features such as eyes, nose, mouth corners, and ears. Our approach proceeds in a hierarchical fashion. Starting with a face detector that roughly classifies the pose into one of the three classes frontal, left profile, or right profile, our system extracts distinctive facial features within the bounding box of the detected face. Finally, it refines the pose estimate by considering the positions of the features. By using few local features instead of the whole subimage containing the face, we achieve a compact representation that allows for training a computationally efficient estimator. Using facial components also contributes to the robustness of the system since the appearance of a single facial feature varies less under different illumination conditions than the appearance of the whole face.

We construct the individual feature detectors using AdaBoost in combination with Haar-like features [20]. As we show in our experiments, our detectors accurately locate the distinctive facial features. We use a neural network trained with resilient backpropagation [12] to estimate the three rotation angles of the head pose. The input features of the neural net are computed based on the positions of detected facial features. Our experimental results show the accuracy of the pose estimates with respect to all three rotation directions.

This paper is organized as follows. The next section reviews related work. Section III presents the AdaBoost algorithm which we use to train our feature detectors. Section IV defines the facial features and their search areas and describes the training data acquisition and the application of the detectors. Section V introduces neural networks and Section VI defines the input features and the topology of the network we use. Finally, Section VIII presents the experimental results.

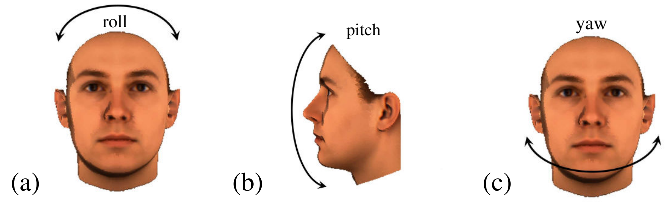


Fig. 1. The three degrees of freedom of the head pose. (a) roll angle denoted as θ_x, (b) pitch angle denoted as θ_y, and (c) yaw angle denoted as θ_z.

II. Related Work

In the last few years, much research has focused on head pose estimation based on monocular images. Existing methods can be categorized in appearance-based and model-based methods.

Appearance-based techniques use the whole subimage containing the face. Most of them concentrate on face detection and consider the pose estimation problem as a classification problem. The range of head orientations is divided into a limited number of classes and classifiers for each class are trained. The number of the classes defines the accuracy of the final pose estimation that can be achieved. Among these techniques are the statistical method of Schneiderman and Kanade [14], the approach proposed by Meynet et al. [10] who train a tree of classifiers by hierarchically sub-sampling the pose space, and the technique of of Li and Zhang [8] who apply a detection pyramid which contains classifiers with increasingly finer resolution. Using only a limited number of classes for the head orientation, however, is not sufficient to recognize head gestures like nodding or shaking. Further appearance-based approaches are the systems developed by Stiefelhagen [17] and Rae and Ritter [11] which are based on neural networks. Appearance-based techniques are quite efficient regarding computation time. However, the mentioned approaches do not yield estimates for all three rotation angles. They concentrate on estimating the yaw and the pitch angle only. Our system, in contrast, provides accurate and continuous head pose estimates in all three rotation directions.

Model-based approaches for head pose estimation use a geometric model of the face. For example, the methods proposed by Stiefelhagen et al. [18] and Gee and Cipolla [4] extract a set of facial features such as eyes, mouth, and nose and map the features onto the 3D model using perspective projection. Schödl et al. [15] combine the 3D model with a texture. They transform an extracted texture of the face to the frontal view and project it onto the model. Dornaika and Ahlberg [2] apply an active appearance model and use a very detailed description of the contours and features of the face. The disadvantage of model-based approaches, however, is that they are computationally more expensive and furthermore, most of them need to be hand-initialized.

The head pose estimation system presented in this paper, combines ideas from several approaches mentioned above and combines their advantages. Our approach first localizes faces in the image using classifiers for frontal and left/right profile faces. In this way, we have already a rough guess about the head pose. Then, we refine the pose estimate using a feature-based technique. Instead of explicitly finding correspondences with a 3D head model, we learn the correlations between the positions of facial features and the head pose from training data.

Facial feature detection itself is a difficult task. In contrast to other methods that use low-level edge features or simple grayscale features [4], [18] to locate facial features, we apply a reliable and fast appearance-based technique.

III. The AdaBoost Algorithm

We learn classifiers for the individual facial features using AdaBoost which is a supervised learning algorithm. Boosting refers to the concept of building a strong, highly accurate classifier by combining weak, not very accurate classifiers. In this work, we apply the adaptive boosting variant of the algorithm AdaBoost which was proposed by Freund and Schapire [3].

Input to the AdaBoost algorithm is a set of labeled (positive/negative) training examples $(x_n, y_n), n = 1, \dots N$, where each x_n is an example and y_n is a boolean value indicating whether x_n is a positive or negative example. In each round $t = 1, \dots, T$, the algorithm computes a distribution D_t over the training examples. Then, AdaBoost selects a weak classifier $h_t : X \rightarrow \{0, 1\}$ that best separates the positive and the negative examples with respect to the current distribution D_t. Based on the error of the classifier, the weights of the examples are updated. The idea is to modify the distribution D_t by decreasing the weights of correctly classified examples and increasing the weights of misclassified examples. In this way, in the next round the algorithm is forced to concentrate on the difficult examples which have been incorrectly classified before. The final strong classifier h is a weighted majority vote of the T best weak classifiers. The weight of a hypothesis h_t is larger the smaller its error ϵ_t is. The complete AdaBoost algorithm is given in Tab. I.

TABLE I
THE ADABOOST ALGORITHM ACCORDING TO VIOLA AND JONES [20].

- Input: Set of labeled examples $(x_1, y_1), \dots, (x_N, y_N)$, where $y_n = 1$ for positive examples and $y_n = 0$ for negative examples.
- Let m be the number of negatives examples and l be the number of positive examples. Initialize the weights $w_{1,n} = \frac{1}{2m}, \frac{1}{2l}$ depending on the value of y_n.
- For $t = 1, \dots, T$:
  1) Normalize the weights to get a probability distribution D_t on the training set $D_t(i) = \frac{w_{t,i}}{\sum_{n=1}^{N} w_{t,n}}$.
  2) Generate a weak classifier h_j for each feature f_j.
  3) Determine the error ϵ_j of classifier h_j with respect to D_t:
  $$\epsilon_j = \sum_{n=1}^{N} w_{t,n} \left| h_j(x_n) - y_n \right| .$$
  4) Choose the classifier h_j with the lowest error ϵ_j and set $(h_t, \epsilon_t) = (h_j, \epsilon_j)$.
  5) Update the weights $w_{t+1,n} = w_{t,n} \beta_t^{1-e_n}$, where $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ and $e_n = 0$, if example x_n is classified correctly by h_t and 1, otherwise.
- The final strong classifier is given by:
$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \log \frac{1}{\beta_t} h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & \text{otherwise.} \end{cases}$$

We apply the variant of Viola and Jones [20] in which a weak classifier h_j is built from a single scalar feature f_j:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \phi_j \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

Here, ϕ_j is a threshold and the parity p_j represents the direction of the inequality. For each weak classifiers h_j, optimal values ϕ_j and p_j are determined so that a minimum number of training examples is misclassified.

For the construction of the classifiers, we use Haar-like features [20], [9]. When applied to an image patch, the value of each Haar-like feature is computed very efficiently using the differences of the sum of the pixel values within neighboring rectangular regions.

In order to reduce computation time, Viola and Jones proposed to use a cascade of classifiers [20]. To account for the fact that the majority of sub-windows in an image are negatives, the detector is constructed so as to process negative instances as efficiently as possible. Examples that are evaluated as positives at some stage of the cascade are processed at the next stage, while examples that are classified as negatives are immediately rejected. To find a trade-off between efficiency and accuracy and to meet given detection rates, constraints are imposed on the individual stage classifiers. New layers are added to the cascade until the overall target detection rate is reached.

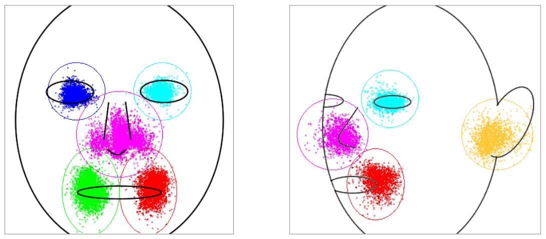


Fig. 2. Distribution of the positions of the features in a normalized face bounding box. The ellipses indicate the search areas for the individual features.

IV. Facial Feature Detection

We apply a face detection system that is also based on the AdaBoost algorithm and uses a boosted cascade of the same Haar-like features. The system works fast, has high detection rates, and yields accurate positions of the faces. We use two trained cascades that are provided by Intel's OpenCV library [6]: one for frontal faces and one for left (and right) profiles. The two detectors cover approximately the range of $[-25°, +25°]$ for the roll angle, the range of $[-40°, +40°]$ for the pitch angle, and the range of $[-90°, +90°]$ for the yaw angle. This is sufficient for our application scenario since angles outside these ranges correspond to atypical head poses of humans during an interaction.

A. Facial Features and their Search Areas

Given a detected face in the image, we localize distinctive facial features within the corresponding bounding box. We use two different facial feature sets for the frontal and profile faces. For frontal features, we use the following five features: left eye, right eye, nose tip, left mouth corner, and right mouth corner. If the yaw angle gets bigger, a part of the face becomes occluded. Therefore, for profile faces, we use only the features from the entirely visible part of the face, i.e., one of the eyes, the nose, one of the mouth corners, and additionally, the ear (if not covered by hair). Due to the symmetry of faces, we train feature detectors only for the left part of the face. By flipping the image patches and applying the detectors for the left features, the right counterparts can be detected.

To focus the search for a feature to a small region of the face which is most likely to contain the particular feature, we define individual search areas for the features in a scale-invariant face bounding box. Fig. 2 shows results from the facial feature detectors for the frontal and left profile views. To match the distribution of the positions of an individual feature as good as possible, the search areas are defined to have either circular or elliptical form.

B. Training Data

To obtain training examples for the detectors, we use patches from images of faces with hand-labeled features. The size of these sub-images is correlated to the size of the bounding box of the face. In order to use for learning the context in which facial features appear, we extract patches with a size of one quarter of the face bounding box. Positive instances are centered at the positions of labeled features. Fig. 3 depicts example images and the annotated positions for all types of facial features that are used for modeling the frontal and the profile faces.

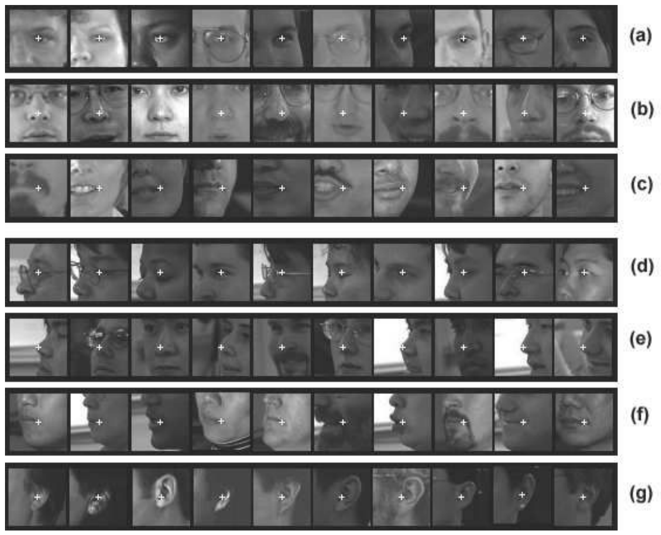


Fig. 3. Positive examples of pose-specific facial features with positions of labeled features. Rows (a), (b), and (c) depict the features in the left half of frontal faces, i.e., eye, nose, and mouth corner. Rows (d), (e), (g), and (f) illustrate features in left profiles, i.e., eye, nose, mouth corner, and ear.

To train a detector on negative examples, we collect them from the search area of the corresponding facial feature. For each labeled positive instance, we extract multiple negative examples which are used for training. The patches of the negative examples have the same size as the corresponding positive examples and are centered at random positions within the search ellipse of the specific facial feature. We add candidate patches only to the negative set if their position has a certain Euclidean distance to the position of the corresponding facial feature. Obviously, the resulting negative examples often also contain the facial feature, however, it is not in the center of the image patch. In this way, the detectors are trained to distinguish with high precision the actual features from the surrounding context. Exactly locating features in the face is crucial for the overall accuracy of the pose estimation.

C. Application of the Feature Detectors

To detect facial features, we use the trained classifier cascade and scan the image at multiple locations and scales. As explained above, we restrict the search to the expected location areas of the specific features in the face. We shift the search window by one pixel each time so that the detector examines all locations within the particular search ellipse.

The size of the facial features is correlated to the face size, given the extracted face bounding box. Therefore, during the search, we only consider a small number of scales of the image patch. In particular, we detect facial features with very high accuracy by using only three different scalings, centered at the scale derived from the bounding box of the detected face.

D. Integration of Multiple Detections

The facial feature detectors are usually insensitive to small changes in translation and scale. For this reason, multiple

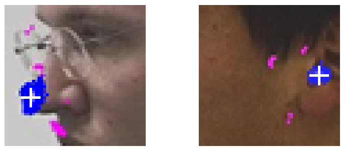


Fig. 4. Examples for facial feature detections. Blue points indicate detections that were considered to compute the final estimated position (white cross), while pink points denote detections that were classified as outliers.

detections of the same feature occur around the true facial feature position. However, for pose estimation a single accurate position is needed. Usually, the true positive detections form a cluster around the true position of a feature and a small number of false positive detections appear as outliers at a larger distance from this cluster. To exclude outliers and average over the positions of the remaining detections, we apply a mean shift algorithm that initializes the mean by averaging over the positions of all detections. It then iteratively shifts the mean by excluding detections that have a larger distance to the current mean than a certain threshold. We determined individual thresholds for the different facial features depending on the size of their search area. Fig. 4 shows two examples for the computation of the estimated position of a nose and an ear.

V. Artificial Neural Networks

To estimate the head pose with respect to the three rotation angles roll, pitch, and yaw from simple features, we apply a neural network. We use a multilayer feed-forward network, in which the units are organized in layers. The units from each layer are connected with directed weighted links to the units of the subsequent layer. Each unit computes its output by passing the incoming weighted signal through an activation function.

Given a set of labeled training examples, the function implied by the data can be learned by updating the connection weights w_{ij} of the network. A common approach to implement training in neural networks is to minimize the network error E which is based on the difference between the target output and the actual output. Using backpropagation [13], the error is fed back through the network and the connection weights are changed so as to reduce the error by some small amount according to a learning rate. We apply a variant of standard backpropagation proposed by Riedmiller and Braun [12]. Resilient backpropagation (RPROP) is an adaptive local search learning algorithm which adapts the weights according to the behavior of the error function:

$$\Delta w_{ij}^t = \begin{cases} -\Delta_{ij}^t & \text{if } \frac{\partial E^t}{\partial w_{ij}} > 0 \\ +\Delta_{ij}^t & \text{if } \frac{\partial E^t}{\partial w_{ij}} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The update values Δ_{ij}^t are computed as follows:

$$\Delta_{ij}^t = \begin{cases} \eta^+ \cdot \Delta_{ij}^{t-1} & \text{if } \frac{\partial E^{t-1}}{\partial w_{ij}} \cdot \frac{\partial E^t}{\partial w_{ij}} > 0 \\ \eta^- \cdot \Delta_{ij}^{t-1} & \text{if } \frac{\partial E^{t-1}}{\partial w_{ij}} \cdot \frac{\partial E^t}{\partial w_{ij}} < 0 \\ \Delta_{ij}^{t-1} & \text{otherwise,} \end{cases} \tag{3}$$

where $0 < \eta^- < 1 < \eta^+$ are constant factors (we chose 0.5 and 1.2, respectively). In the beginning, the update values Δ_{ij} are initialized with some small value Δ_0. RPROP is one of the best performing learning algorithms in neural networks in terms of convergence speed, accuracy, and robustness.

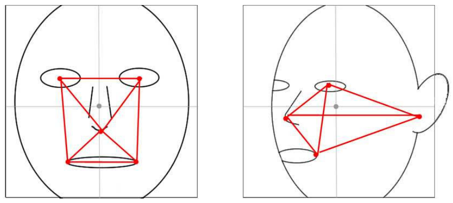


Fig. 5. Input features for the frontal (left) and the profile (right) pose estimation neural network in a normalized face bounding box. We use the positions of the facial features as well as normalized distances between them.

VI. Pose Estimation

Our network learns continuous rotation angles from input data consisting of the relative positions of distinctive features in the face and relative distances between pairs of features. Since we first apply the profile and frontal face detectors, we already have a coarse estimate of the pose. We train separate neural networks to estimate head rotation for the frontal face category and for the profile view category. Due to the symmetry of faces, we only need to train one network for the estimation of profile poses.

A. Network Topology and Input Data

We use a feedforward network with one hidden layer containing six units, three output units describing the three rotation angles roll θ_x, pitch θ_y, and yaw θ_z. As activation function we use the sigmoid function. The number of input units varies with the number of facial features used to model the different poses. The input to the neural network consists of two types of scalar features:

- Coordinates x and y denoting the position of a facial feature. The origin of the coordinate system is positioned at the center of the face bounding box. x and y are normalized in the range $[-0.5, +0.5]$.
- Distances $d_x = x^i - x^j$ and $d_y = y^i - y^j$ between pairs of facial features i and j. These values are also normalized in the range $[-0.5, +0.5]$.

The input features for the frontal face pose estimation are based on the positions of five facial features that are the eyes, the nose tip, and the lip corners. The number of input units of the neural network is 26 (five features, eight distances). The profile face is described by four features that are visible under the specific pose, i.e., one of the eyes, one of the mouth corners, one of the ears, and the nose tip. The number of input values is in this case 20 (four features, six distances). The full set of facial features and the used distances between them are depicted in Figure 5.

B. Network Output

The range of poses that the neural nets estimate is constrained to the approximate ranges of poses the face detectors cover (i.e., $\theta_x \in [-25°, +25°]$ and $\theta_y \in [-40°, +40°]$). The output yaw angle θ_z^f of frontal faces is determined to lie in the interval $[-40°, +40°]$, whereas the yaw angle θ_z^p of profile faces is limited to $[-90°, -30°] \cup [+30°, +90°]$. In the overlapping region, the values of the two pose estimators can be used to improve accuracy. We normalize the output values of the neural net so that they vary between -0.5 and $+0.5$, where 0 corresponds to the middle of the range of the allowed poses for each particular rotation direction.

VII. Datasets

To train our facial feature detectors and the neural networks, we collected image data from different standard datasets as well as synthetically generated images of varying head poses.

For training and testing the feature detectors for *frontal* poses, we used disjoint image sets from the BioID database [7] and from the PIE dataset [16]. As training data for the feature detectors of *profile* poses we chose images from the PIE dataset. We tested the performance of the detectors on a different set of images of the PIE database as well as on images of the Pointing'04 dataset [5]. Before training, a few preprocessing steps are applied. The image patches are scaled to a fixed size of 24×24 pixels and converted to grayscale. To minimize the effect of different illumination conditions, additionally, brightness and contrast normalization are applied.

To train the neural nets for pose estimation, we do not use the database images since the pose definitions would have to be adjusted. Instead, we use 3D human head models from the MPI Face Database [19] to generate training data. We rendered these models with natural texture in order to obtain images of various poses. In particular, we rotated the head models in all three directions simultaneously. By rendering the models, we generated a large number of face images under different pose, scale, and illumination conditions, and with different background colors. The advantage of using this set of synthetic images is that the rotation angles of the head poses are directly available. We additionally scaled the head models disproportionally in the different directions to have a rich dataset.

VIII. Experimental Results

We carried out a series of experiments an database images as well as on real video images to evaluate of our approach.

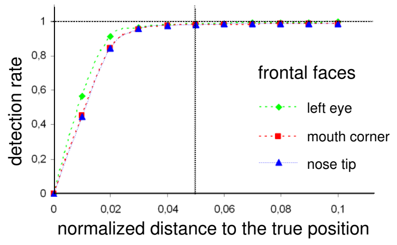
A. Accuracy of Facial Feature Detectors

We evaluated the accuracy of the individual facial feature detectors on independent test sets consisting of 2,726 and 1,537 images of frontal and left-profile faces, respectively. Each image in the test set contains a single face. Using our detectors, we estimated the positions of the individual facial features in a previously extracted face bounding box.

To measure the classification performance, we compute the detection rate which is given by

$$\text{detection rate} = \frac{\text{number of correct feature detections}}{\text{number of test images}}. \quad (4)$$

In order to calculate the detection rate, we need to define a criterion for successful facial feature detection. We use the Euclidean distance between the detected feature position (if it is detected) and the ground truth. To make this distance



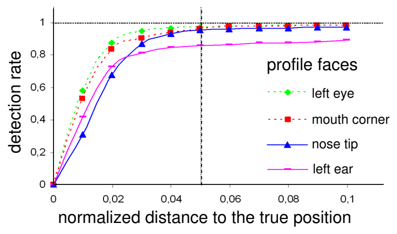


Fig. 6. Detection performance of the facial feature classifiers. With a threshold of 0.05 for the distance between the estimated feature position and the true position in a normalized face bounding box, we achieve a detection rate of 98%.

independent of the image size, the difference is normalized by the bounding box of the detected face. Obviously, there is a trade-off between the detection rate and the precision of the positions of detected facial features. To illustrate this, Fig. 6 plots the detection rates achieved for a number of distance thresholds for the frontal and the profile facial feature detectors, respectively. As can be seen, excellent results of approximately 98% correct detections are achieved for a threshold of 0.05. The mean distance to the true position over all detections that are considered to be correct is 0.015. The remaining 2% false detections include detections that were outside the given radius as well as missing detections. Only the detector for ears in profiles showed worse performance with a detection rate of about 85% detection rate for the same threshold. Upon visual inspection, we discovered that the test set contains a few subjects with haircuts that (partially) cover the ears. This is the major source of error for this detector.

B. Head Pose Estimation

We evaluated the performance of our pose estimation system on a separate test set consisting of synthetically generated images. We used one of the 3D head models which was not used for training. The faces and facial features were automatically detected by our detectors. For testing the neural network used for pose estimation, we used all correctly detected faces with a full set of facial features. We used 250 images for evaluation of the frontal pose estimator and 320 images for the profile pose estimator. To evaluate the performance of the head pose estimation, we computed the mean absolute error for the three rotations (see Tab. VIII-B). As can be seen, the frontal poses are approximated with a high accuracy, while the profile poses seem to be more difficult to learn. We made the observation that the bounding boxes the profile face detector provides have a high variation in size and position relative to what we would consider as the actual face. Thus, as a preprocessing step before the profile head pose estimation starts, we adjust the bounding box so that it better covers the skin-colored area.

TABLE II

MEAN ABSOLUTE ERROR OF THE ESTIMATED ROTATION ANGLES.

	Roll	**Pitch**	**Yaw**
Frontal	$1.65°$	$2.74°$	$3.13°$
Profile	$10.5°$	$5.6°$	$7.3°$

To deal with cases in which a feature is not detected, we trained further neural networks whose input is based on the remaining features. The estimation results were slightly worse in cases of missing features.

C. Real-Time Head Detection and Pose Estimation

Additionally, we performed qualitative experiments to evaluate the capability of our approach to estimate the head pose from images in real-time. Using a standard webcam with a resolution of 640×480 pixels, we currently achieve a rate of 10 fps on a standard PC. To improve the robustness of our system and to smooth the estimated angles, we apply independent Kalman filters to track each facial feature and each rotation angle over time. Fig. 7 shows some example images and the corresponding estimated pose which is illustrated by the rendered head model on the right hand side. As can be seen, the head pose is estimated quite accurately. An exception is the last example.

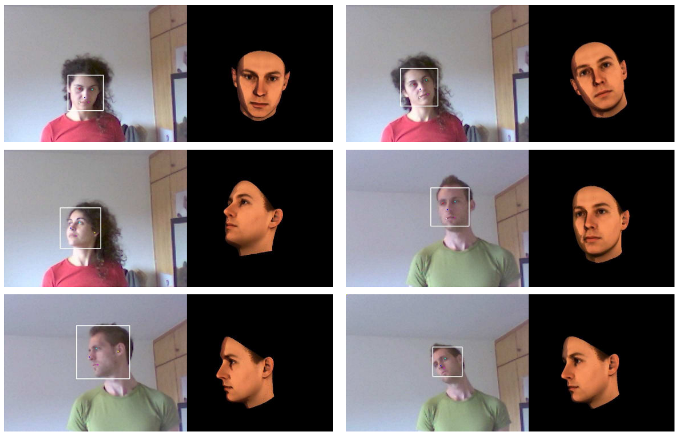


Fig. 7. Qualitative pose estimation results. The rendered head model illustrates the estimated head pose.

IX. CONCLUSIONS

In this paper, we presented a feature-based system that estimates the head pose from monocular images. Our system works in three stages. First, a face detector classifies the pose into one of the general classes frontal, left, or right profile. Then, classifiers detect distinctive features within the face. The classifiers for the individual facial features are learned using AdaBoost with Haar-like features. Given the positions of the individual facial features, a neural network finally estimates the three rotation angles roll, pitch, and yaw of the head pose. Since we use local features instead of the whole subimage containing the face, we have a compact representation which results in a computationally efficient estimator. Our system system yields continuous estimates of all three rotation angles.

We performed a series of experiments using standard image databases as well as real-time image data. Our system robustly detects the distinctive features and yields highly accurate pose estimates, especially for the frontal face class. We currently aim for analyzing head gestures by modeling the head pose over time.

ACKNOWLEDGMENT

This project is supported by the DFG (Deutsche Forschungsgemeinschaft), grant BE 2556/2-2.

REFERENCES

[1] M. Bennewitz, F. Faber, D. Joho, S. Schreiber, and S. Behnke. Towards a humanoid museum guide robot that interacts with multiple persons. In *Proc. of the IEEE/RSJ Int. Conf. on Humanoid Robots (Humanoids)*, 2005.

[2] F. Dornaika and J. Ahlberg. Fast and reliable active appearance model search for 3D face tracking. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4):1838–1853, 2004.

[3] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[4] A. H. Gee and R. Cipolla. Determine the gaze of faces in images. *Image and Vision Computing*, 12(10):639–647, 1994.

[5] N. Gourier, D. Hall, and J. L. Crowley. Estimating face orientation from robust detection of salient facial features. In *Int. Workshop on Visual Observation of Deictic Gestures at POINTING04*, 2004.

[6] Intel. Open source computer vision library. http://www.intel.com/technology/computing/opencv/, 2007.

[7] O. Jesorsky, K. Kirchberg, and R. Frischholz. Robust face detection using the Hausdorff distance. In *3rd Int. Conf. on Audio and Video based Person Authentication (AVBPA)*, 2001.

[8] S. Z. Li and Z. Q. Zhang. Floatboost learning and statistical face detection. *IEEE Trasactions on Pattern Analysis and Machine Intelligence*, 26(9), 2004.

[9] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2002.

[10] J. Meynet, T. Arsan, J. C. Mota, and J. Thiran. Fast multiview tracking with pose estimation. Technical Report TR-ITS.2007.01, EPFL, 2007.

[11] R. Rae and H. Ritter. Recognition of human head orientation based on artificial neural nets. *IEEE Transactions on Neural Networks*, 9(2):257–265, 1998.

[12] M. Riedmiller and H. Braun. RPROP - A fast adaptive learning algorithm. In *Proc. of the Int. Symposium on Computer and Information Science VII*, 1992.

[13] R. Rojas. A graph labelling proof of the backpropagation algorithm. *Commun. ACM*, 39(12es), 1996.

[14] H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3):151–177, 2004.

[15] A. Schödl, A. Haro, and I. A. Essa. Head tracking using a textured polygonal model. Technical Report GIT-GVU-98-24, Georgia Institute of Technology, 1998.

[16] T. Sim, S. Baker, and M. Bsat. The CMU pose, illumination, and expression database (PIE). *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(12):1615 – 1618, 2003.

[17] R. Stiefelhagen. Estimating head pose with neural networks – Results on the Pointing04 ICPR workshop evaluation data. In *Pointing '04 ICPR Workshop of the Int. Conf. on Pattern Recognition*, 2004.

[18] R. Stiefelhagen, J. Yang, and A. Waibel. A model-based gaze tracking system. In *Proc of the IEEE Int. Joint Symposia on Intelligence and Systems*, 1996.

[19] N. Troje and H. H. Bülthoff. Face recognition under varying poses: The role of texture and shape. Vision Research 36, Max Planck Institute for Biological Cybernetics, 1996.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2001.