Real Time Head Model Creation and Head Pose Estimation On Consumer Depth Cameras

Manuel Martin
Fraunhofer IOSB
Fraunhoferstr. 1
76131 Karlsruhe, Germany

Florian van de Camp
Fraunhofer IOSB
Fraunhoferstr. 1
76131 Karlsruhe, Germany

Rainer Stiefelhagen
Karlsruhe Institute of Technology
Vincenz-Priessnitz-Str. 3
76131 Karlsruhe, Germany

Abstract

Head pose estimation is an important part of the human perception and is therefore also relevant to make interaction with computer systems more natural. However, accurate estimation of the pose in a wide range is a challenging computer vision problem. We present an accurate approach for head pose estimation on consumer depth cameras that works in a wide pose range without prior knowledge about the tracked person and without prior training of a detector. Our algorithm builds and registers a 3D head model with the iterative closest point algorithm. To track the head pose using this head model an initialization with a known pose is necessary. Instead of providing such an initialization manually we determine the initial pose using features of the head and improve this pose over time. An evaluation shows that our algorithm works in real time with limited resources and achieves superior accuracy compared to other state of the art systems. Our main contribution is the combination of features of the head and the head model generation to build a detector that gives accurate results in a wide pose range.

1. Introduction

Head pose estimation is an important part of the non-verbal communication of humans. People subconsciously use it to determine the direction of attention of others to get an idea of their intention. The same principle can also be used in computer systems to react to users in a natural way. Example uses include assisting technologies like a collision warning system in a car which mutes an alarm if the car detects that the driver is already aware of the threat, but also include natural user interfaces. Such interfaces allow users to interact with novel input modalities such as gestures instead of traditional input devices like mouse and keyboard. In such settings head pose can be utilized as an additional modality, either for direct interaction or to determine if the users' focus is on the computer system to only react if input is directed at the system.

Our approach is based on tracking a head model using the iterative closest point algorithm but unlike other approaches the creation of the head model is part of our algorithm as well. This has the advantage that the model fits exactly to the user and can even incorporate things like glasses. But it also poses some challenges: The head has to be segmented to generate the model and the orientation of the generated model can be different every time, therefore an additional transformation from the result of the model tracking to the head pose is necessary. Both of these problems can be solved efficiently. We will show how the head pose can be used to segment the head in a straightforward way and how the required transformation can be determined using features of the head.

2. Related Work

There are many different approaches to head pose estimation. Research of non intrusive head pose estimation started with mono and stereo cameras. Murphy-Chutorian et al. give a good summary of such approaches [6]. They can be divided into feature based and appearance based approaches.

Feature based detectors extract features of the head to estimate the pose. This has the advantage that most natural features do not change much from person to person, so the inter person variance is low [14]. The disadvantage is that most features lie on the face which limits the detectable pose range because most of the facial features are at least partially occluded in side views. Another drawback is the need for high resolution to detect these features.

Appearance based detectors use the whole visible face and do not extract additional features. To build such a detector, multi layer perceptrons have shown to give good results [13]. The advantage of this approach is that a much lower resolution is sufficient to estimate the head pose, with the drawback that the general appearance varies much from person to person making it harder to keep the head pose estimation person independent.

Different lighting conditions can be a great challenge for cameras in the visible light spectrum because they can change the appearance of faces drastically. Consumer depth cameras on the other hand do not suffer from this problem because their depth sensing method is mostly independent from ambient lighting. Apart from the invariance to illumination depth sensing cameras open the way to completely different approaches to head pose estimation because they offer easy and cheap access to dense depth images. There are different ways to use this information. It is possible to use machine learning algorithms like random decision forests [3, 9] directly on the depth image, with the advantage that a single depth image is sufficient to estimate the head pose. Other approaches first extract some information out of the depth image like a point cloud around the nose [11] or a reference depth image of the head [7] and then match this information to the following frames to estimate the pose change relative to the initialization. These methods have the disadvantage that they assume that the head pose is initially known in order to track the head pose in following frames. A third method for depth based head pose estimation uses some form of precomputed personalized head template [2, 15] and then matches this template to the live data using iterative closest point to estimate the head pose. The drawback of this approach is that the generation of the template is not part of the algorithm itself. The template may, for example, be generated with a morphable head model [15], but this approach often requires manual initialization of key points which is not optimal. A generic head model can also be used with good results by precomputing reference range images for different head poses and comparing them all to the current input in a massively parallelized algorithm on the gpu [1]. It is also possible to estimate the head pose by extracting features in the point cloud. The approach used by Gurbuz et al. [4] extracts a face plane and the eye positions in 3D and uses this information to determine the pose. This approach has the advantage that it also works on single depth images but the detectable pose range is severely limited because both eyes have to be visible which is not the case in side views of the head.

To conclude this section we can state that each of the discussed approaches has some disadvantages. Template based approaches need a personalized template, feature based approaches can only detect a limited pose range and tracking based approaches need a initialization with a known head pose. The key idea of our algorithm is to combine these approaches to get most of the advantages without the limitations.

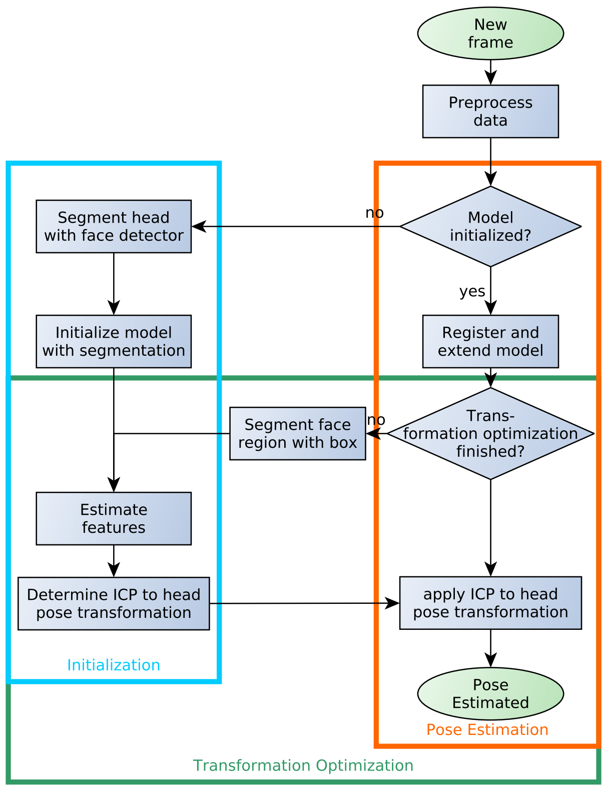

New frame
Preprocess data
Model initialized?
no
yes
Segment head with face detector
Initialize model with segmentation
Register and extend model
Trans-formation optimization finished?
no
Segment face region with box
Estimate features
Determine ICP to head pose transformation
apply ICP to head pose transformation
Pose Estimated
Initialization
Pose Estimation
Transformation Optimization

Figure 1. The flowchart of the combined head pose detector.

3. Implementation

Our approach consists of the following building blocks:

1. preprocessing, which is required to remove noise in the depth image
2. head segmentation, that is necessary to build the model
3. generation of the head model
4. head pose estimation using features of the head

Figure 1 shows how these parts are combined. Our algorithm starts without any information about the person in front of the camera. To find the person and to initialize the model it uses a face detector. The initialization works in the head pose range where the face detector works. The orientation of the generated model can therefore differ at each initialization. Accordingly, with the registration of the model alone it is only possible to determine the pose change from frame to frame, but it is not possible to determine the head pose. To solve this problem the algorithm uses features of the head at the initialization to determine the transformation from the model orientation to the head pose. With this transformation the algorithm is then able to determine the head pose and to extend the model in following frames without the use of features of the head. Estimating this transformation from features of a single frame can be unreliable the algorithm therefore improves the transformation on some further frames.

In the following, we describe each part individually and then describe how the parts are combined to form the final detector.

3.1. Preprocessing

The algorithm uses the color image of the consumer depth camera to initially find the head. To be able to use the found region on the depth image we register the depth image to the color image with the intrinsic and extrinsic parameters of the consumer depth camera.

The depth image of consumer depth cameras, however, usually contains some noise. This noise causes the registration algorithm, used later on to register the head model, to fail in difficult situations. We therefore use a smoothing algorithm to remove most of the noise. The smoothing algorithm has to be edge aware to prevent artifacts on depth discontinuities. The bilateral filter would be a choice to solve this problem but it is too time consuming. We therefore implement our own smoothing algorithm which is faster but with the drawback that it can produce some artifacts. Our smoothing filter first applies a Gaussian filter to the depth image. This causes many artifacts on depth discontinuities and on edges with invalid depth values in the neighborhood. These artifacts become apparent if the depth image is converted to a point cloud (see Figure 2a). To remove most of these artifacts the algorithm subtracts the smoothed image from the original image and applies a threshold. All pixels in the resulting image with values higher than the threshold are considered artifacts. We choose a threshold of 10 mm to ensure that all remaining artifacts are smaller than any feature of the head. There are often additional artifacts that are smaller than the threshold in the neighborhood of larger artifacts. To remove them as well, the algorithm applies a morphological dilation to extend the mask generated by thresholding (see Figure 2b). Lastly the smoothingc algorithm sets all the identified artifacts in the smoothed depth image to their original values. This approach results in smoothing of surfaces with the disadvantage that the final image may contain some remaining artifacts and that the edges stay noisy and frayed out (see Figure 2c).

After smoothing, the depth image is converted to a point cloud using the intrinsic parameters of the depth sensor. We use iterative closest point with the point to plane metric to register the head model later on and therefore need the normals of the points in the point cloud. We determine the normals with a simple cross product of the neighboring points. Normals created this way on the original sensor data would be to inaccurate for registration but with our smoothing algorithm this approach is suitable and fast.

3.2. Segmentation

We use two different approaches to segment the head depending on how much information is already available about the current frame.

If no information about the current frame is available we use a face detector to determine the location of the user's face. We use a commercial face detector [12] to detect the face because it also estimates the location of the eyes which is used by the second segmentation algorithm. We can then use the bounding box of the face, obtained from the color image to cut out the corresponding part of the depth image to create the segmentation.

If the head pose is already known in the current frame F it can nevertheless be necessary to segment the head to extend the head model. In this case the head pose can be used to extract the head in a robust way. To achieve this the segmentation algorithm defines an axis aligned bounding box with the parameters C_{min} and C_{max} and positions it at the origin of the coordinate system. It then transforms each point of the current point cloud with the inverse head pose P^{-1} and tests if the point lies inside the bounding box. This results in a point cloud T which only contains the head and is defined by following equation:

$$T = (x | x \in F \wedge P^{-1}x > C_{min} \wedge P^{-1}x < C_{max}) \quad (1)$$

A bounding box with a default size is enough to segment the head in general but it can occur that additional clutter like the headrest of a chair are included in the segmentation. The result can therefore be improved by estimating the size of the box more accurately based on facial features. Our head pose algorithm determines the tip of the nose and a point inside the head. The face detector additionally determines the positions of the eyes. Based on this information the width of the box can be calculated from the distance between the eyes, the depth from the distance between the origin and the nose tip and the height from the distance between the center of the eyes and the line between nose tip and origin (see Figure 3). With those features it is then possible to determine the size of the box more accurately, but it is also possible to determine boxes for different head parts like the eyes, the nose or the facial area.

3.3. Model Structure

The head model can be described as a collection $M = \{P_i, T_i\}_{i=1}^N$ of model parts, where each part i consists of a point cloud P_i and its orientation T_i in the model. N is the number of current model parts. It increases as new parts are added to the model. Data is therefore not integrated into a single consistent representation but consists of independent parts. This approach reduces the cost of model creation significantly but also sacrifices consistency. Each part of the model represents a certain view of the head. The point

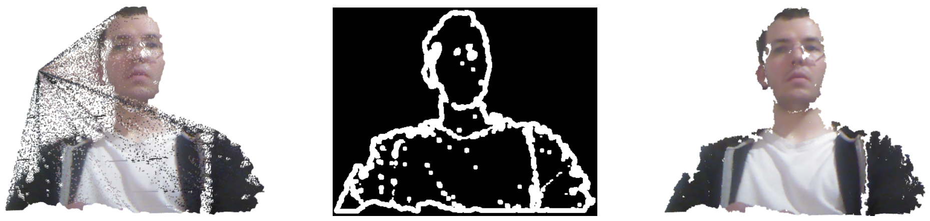

(a) (b) (c)

Figure 2. (a) shows a point cloud reconstructed from a depth image smoothed with a simple Gaussian filter causing the artifacts. (b) shows the mask used to identify the artifacts and (c) shows the result of our smoothing algorithm.

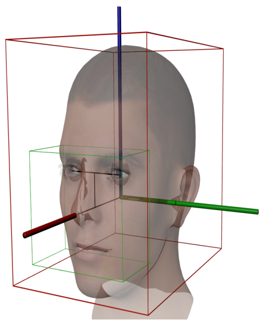


Figure 3. Scheme of the box segmentation. The red box denotes the segmentation of the whole head, the green box the segmentation of the facial feature area and the black lines visualize the distances used to size the boxes.

clouds stay unmodified after their segmentation, therefore the saved transformation is necessary to describe the location and orientation of the part in the model. This transformation is the registration result of the head model at the time of the part's creation. The first model part is created without prior registration, its transformation is therefore the identity matrix.

This model design has advantages for efficient registration because given a hypothesis of the current pose it is possible to select a suitable part of the model with an appropriate view and therefore to discard large parts of the model which are not visible anyway.

3.4. Head Model Creation

The head model is initialized at the start of the algorithm using the face detector based segmentation to extract the point cloud of the first model part. The transformation of this part is the identity matrix because it is generated without prior registration and therefore fits directly into the model. On all following frames the algorithm continually registers the model with the iterative closest point algorithm using the result of the previous frame as the initialization of the registration algorithm. After each successful registration the algorithm determines if a new model part should be created by checking if the angular distance $d(T_i, T_c)$ between the current registration result T_c and the transformation of all model parts T_i is greater as a threshold d_m:

$$N(T_c) = \begin{cases} true & \forall T_i \in M : d(T_i, T_c) > d_m \\ false & \text{else} \end{cases} \quad (2)$$

T_c controls the angular distance between all parts of the model. A small distance results in model parts that are very similar and overlap to a high degree. A large distances on the other hand can result in a model with no overlap and unfillable holes. If a new model part is necessary it is created with the segmentation based on the bounding box. We evaluated d_m systematically and found that a value of 0.4 radians results in the best performance.

3.5. Model Registration

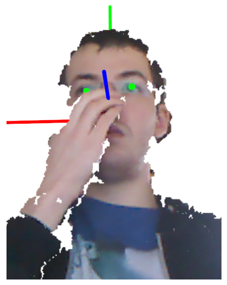
Iterative closest point is a standard algorithm to find a transformation that minimizes the distance between point clouds. We use it here to register the head model continually to the live data. The algorithm has a few characteristic steps:

1) Finding Correspondences: A major part of the algorithm is the selection of correspondences. We use back projection of the model points onto the depth image with the intrinsic parameters. This approach introduces some inaccuracies compared to a direct nearest neighbor search with a kd-tree but it is much faster [8].

2) Correspondence Filtering: The found correspondences often contain outliers which are filtered out in two steps. Firstly, every correspondence with a distance larger than 20cm is filtered out to remove any correspondences with the background. Secondly, the median of the distances of all correspondences is determined and all correspondences with a distance larger than twice the median are filtered out. This step increases the robustness to occlusions and facial expressions because points that do not fit to the common shape of the model and the current frame are ignored (see Figure 4).

3) Transformation Estimation: After filtered correspondences are determined the transformation is estimated with the point-to-plane metric, which is minimized with a linear least squares approach [5].

4) Termination Check: Iteration is stopped at the latest after 30 iterations but if this occurs the registration is regarded as a failed attempt and is not used. Iteration stops


(a)

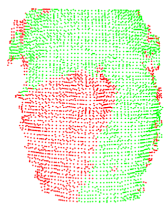

(b)

Figure 4. (a) shows the input point cloud annotated with the head pose and (b) shows the used model part. Red dots denote the points filtered out by the correspondence rejection stage.

before if the rotation and translation change from iteration to iteration falls below a threshold.

As previously explained, our model consists of independent point clouds each representing a specific view of the head. This can be used by the registration algorithm to improve the speed and robustness of the registration. The registration algorithm uses the registration result of the last frame as the initialization in the current frame T_{guess}. Additionally, it uses this estimate to search for the part of the model with the lowest angular distance to the estimate:

$$P_{selected}(T_{guess}) = \{P_i \in M : \min_{i \in N} d(T_{guess}, T_i\} \quad (3)$$

This part represents the view of the head which is most similar to the expected view in the current frame. To register it the transformation estimate T_{guess} has to be corrected with the transformation $T_{selected}$ of the selected part:

$$\hat{T}_{guess} = T_{guess} * T^{-1}_{selected} \quad (4)$$

After the registration this correction has to be reverted to get the final result:

$$T_{result} = \hat{T}_{result} * T_{selected} \quad (5)$$

If the registration of the nearest part fails the selection of a model part and the registration is repeated excluding the already tried parts. This has the effect that gradually further away parts are tried which can help with facial deformations and occlusions because those parts may not contain the problematic areas and may work even if parts with a smaller angular distance fail to work. At the moment we limit the tried parts to 4 because of performance reasons. All remaining parts of the model are ignored which increases the speed of the registration algorithm because it allows to register only a fraction of the whole model on each frame. Additionally, it increases robustness because most of the model that is not represented in the current frame is not used for registration and can therefore not create wrong correspondences and wrong registration results.

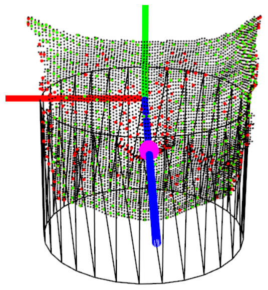


Figure 5. An example of the feature based pose estimation. The green points denote the points that support the cylinder estimation and the red points are the identified outliers. The pink sphere shows the estimate of the nose tip.

3.6. Head Pose Estimation with Features

The feature based pose estimation algorithm does not rely directly on features of the head. It works on hypotheses of the pose coordinate system axes and of the pose coordinate system origin. Those hypotheses can be generated from different sources and can also be accumulated over time to make the estimation more robust. We choose the nose tip and the vertical head axis as features because they are easy to find on a segmentation of the depth image of the head. In the following we first explain how the features are detected and then how the head pose is generated with the detected features.

Detection of the vertical head axis works by fitting a cylinder with random sample consensus to the given segmentation of the head. This algorithm has the advantage that it can work even if many outliers exist. In our case these outliers are all points that are not part of the general face curvature like the nose or the sides of the head. The random sample consensus algorithm determines for each point of the point cloud if it is an inlier and supports the model or if it is an outlier. The algorithm only proceeds with the nose detection if at least 20% of the points of the head segmentation are labeled as inliers.

Nose tip detection works on the simple assumption that the nose tip is the point nearest to the sensor in the head segmentation. This assumption is only true if the head is roughly oriented in the direction of the sensor. To greatly extend the region where the nose can be detected the point cloud can first be transformed with the inverse head pose to always get an almost frontal view. The inverse head pose can be from different sources. It can either be the estimated head pose of the current frame or the head pose of the last frame. If no estimate is available the algorithm uses the identity matrix.

After the detection of the features, the algorithm generates the hypotheses for the coordinate axes. It generates a hypothesis of the origin by projecting the nose tip onto the estimated cylinder. It generates a hypothesis of the z-axis as the vector between the estimated origin and the nose tip and uses the estimated cylinder axis as the hypothesis of the y-axis (see Figure 5). It can not generate a hypothesis for the x-axis directly out of the features but it can compute such a hypothesis with a cross product of the two other axes hypotheses. The axes hypotheses are orthogonal to each other so a pose could be directly generated, but the accuracy of the pose can be improved if more features are used which makes it impossible to determine the pose in such a simple way because the hypotheses are not necessarily orthogonal to each other. The basic problem is to find a rotation matrix that is closest to the degenerated rotation described by the axis hypotheses. To solve this problem for arbitrary numbers of hypotheses we formulate it as a search for a rigid transformation between known point correspondences. There are different algorithms to solve this problem. We use the algorithm developed by Schöneman [10]. The correspondence to each of the axis hypotheses is the corresponding axis of the standard euclidean coordinate system. The algorithm starts with a covariance matrix of the features which is defined by following equation:

$$C = \sum_{i}^{N} f_i^T * c_i \tag{6}$$

where f_i is the i-th axis hypothesis and c_i is the corresponding coordinate axis of the standard euclidean coordinate system. The result of the algorithm is a rotation matrix that minimizes the the euclidean distance between the corresponding points. This rotation matrix can then be combined with the average of all accumulated origin hypotheses to the complete head pose.

3.7. Fusion of the Building Blocks

The following description follows the three groups depicted in 1. The algorithm first initializes, then it can estimate the head pose and while estimating the head pose it can optimize the needed transformation from the model registration result to the head pose.

Initialization works by first initializing the model generating algorithm. This creates the first model part which has the identity matrix as its transformation (see Section 3.4). The problem is that it is not known how the head is oriented inside the model. Therefore the head features are estimated and the head pose is determined based on these features. This head pose determined with features constitutes a transformation from the registration result of the model to the head pose in all following frames. The head pose then depends on the transformation determined by the registration T_{ICP} and on the head pose determined with features T_{init} at the time of initialization. The head pose is determined with following equation:

$$T_{headpose} = T_{ICP} * T_{init} \tag{7}$$

This equation also works on the initialization frame because T_{ICP} is the identity matrix by default.

Pose Estimation after the initialization works by simply registering the model with ICP and using equation 7 with T_{init} - the transformation determined with features at the initialization of the model.

Transformation Improvement of T_{init} is necessary because as our evaluation shows the head pose determined with features is not as accurate. However, the error made in this transformation affects all following pose estimations. To improve this transformation features are collected on more frames and the transformation T_{init} is determined with all of them which averages out the error. The basic principle how these features are combined is already explained in section 3.6. The remaining problem is that the head moves from frame to frame so the extracted features of different frames are not oriented in the same direction. However, this can easily be corrected by multiplying the features with the inverse ICP result T_{ICP}^{-1}. This approach also has the advantage that it is not necessary to determine the features on each frame to improve the transformation. Instead it is possible to wait until the estimated pose indicates a view where the features can be extracted more reliably. We therefore limit the pose range where additional features are estimated to near frontal faces with a yaw angle of $\pm 25°$, because in this range the features are estimated with the best precision. We limit the improvement to 100 frames because doing it on more frames does not have much effect.

An algorithm based on tracking has to reinitialize if tracking is lost. In this case our algorithm uses the face detector to find the head again and to create a head segmentation. The segmentation is then used by the feature based head pose detector to generate a new pose hypothesis for the registration of the model with iterative closest point.

4. Evaluation

To evaluate our algorithm we use an annotated test data set created by Fanelli et al [3]. It consists of 15678 annotated frames, recorded with the Microsoft Kinect, with head rotations of $\pm 75°$ for yaw, $\pm 60°$ for pitch, and $\pm 50°$ for roll. Each frame contains the segmented upper body of a person. The frames are split into 24 streams with 14 male and 6 female participants. The ground truth was automatically generated with a template based head pose detector that achieves a reported accuracy of about $1°$. Some of the streams contain challenging data with fast rotations where iterative closest point sometimes fails to converge correctly. Additionally there are streams where long moving hair make model generation difficult. Some of the streams also contain gaps presumably because the automatic annotation system failed on these frames. Those gaps sometimes lead to failed tracking in our algorithm and ac-

Table 1. The performance statistics of the head pose detector based on features and of the combined detector.

	Mean Absolute Error [°]				Success
	Roll	Pitch	Yaw	Angle	rate [%]
Feature based detector	6.82	4.88	5.14	11.18	94.99
Combined detector after first iteration	3.26	2.34	2.61	5.47	94.41
Combined detector after fifth iteration	3.62	2.54	2.57	5.77	97.58

count for some of the missed frames. In the following evaluation we will exclude all frames where our algorithms could not estimate the pose but we always note the percentage of used frames, called the success rate from here on. In addition to the mean absolute error of the Euler angles we also determine the absolute angle error. It is defined as the L2-norm of the Euler angles and allows to compare different results more easily.

First we evaluate the performance of the feature based head pose. In the combined algorithm it is used to estimate and improve the transformation from the model to the head pose T_{init}. In this case, the feature based head pose estimation is initialized with the facial area segmented with the box segmentation and with the current head pose to find the nose. To evaluate this step independent of the rest of our algorithm we need the current head pose independent of the rest of our algorithm. We therefore use the ground truth of each frame as the initialization to find the nose and we use the the box segmentation as usual.

The combined algorithm builds the head model from scratch. It is therefore expected that the algorithm has a lower success rate in the beginning than later on when a full model is already available. To test this we started the algorithm on each stream of the test dataset without a model and let it iterate five times over the images of the stream to give the algorithm time to create the model.

Table 1 shows the result of our evaluation. The feature based detector alone does not achieve highly accurate results. The combined algorithm on the other hand achieves a precision twice as good as the feature based detector. This shows that the estimation of the transformation from the model to the head pose T_{init} determined with features collected on different frames improves the accuracy notably. The table also shows that the errors of the combined algorithm in the first iteration is slightly lower than the error in the last iteration but that the success rate increases by 3%. This indicates that in the first iteration some more difficult frames do not work that work in later iterations where the model already contains more information. This also explains the slightly lower performance in later iterations because more difficult frames are likely estimated with less accuracy. The performance does not change further with more iterations, because after three iterations the results do not change anymore as all possible head parts are added to the model.

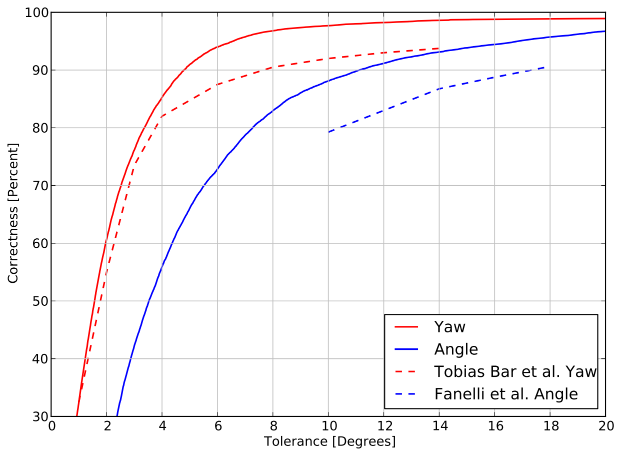


Figure 6. The correct classification rate depending on the accepted level of inaccuracy of our algorithm compared to other systems.

Figure 6 shows the result of our detector after the fifth iteration in comparison to two other state of the art systems that have been evaluated on the same dataset. The system developed by Tobias Bär et al. [2] uses a personalized head template and iterative closest point and the system developed by Fanelli et al. [3] uses random decision forests. Accepting a tolerance of 4 degrees in yaw, our algorithm estimates 85% of the head poses correctly while the system developed by Tobias Bär et al. estimates 82% correctly.

While the frame rate can drop down to 28fps on fast head movements, caused by the iterative closest point algorithm needing more iterations to converge, our algorithm achieves an average frame rate of 48 fps on the test dataset on a single Core of an Intel Core i7-2600K. This is more than sufficient as the depth sensors only provide frames at 30fps.

Figure 7 shows some examples of corner cases of our algorithm. We noticed that our model tracking algorithm can only cope with limited non rigid changes of the head. Especially long hair that fall into the face cause some problems. Another problem are fast rotations around the yaw axis which cause our model to sometimes get stuck on the cheek. On the other hand, we noticed that our algorithm can cope well with occlusions and that it can detect even extreme rotations in yaw of up to ±120° if the ears are not occluded by hair.

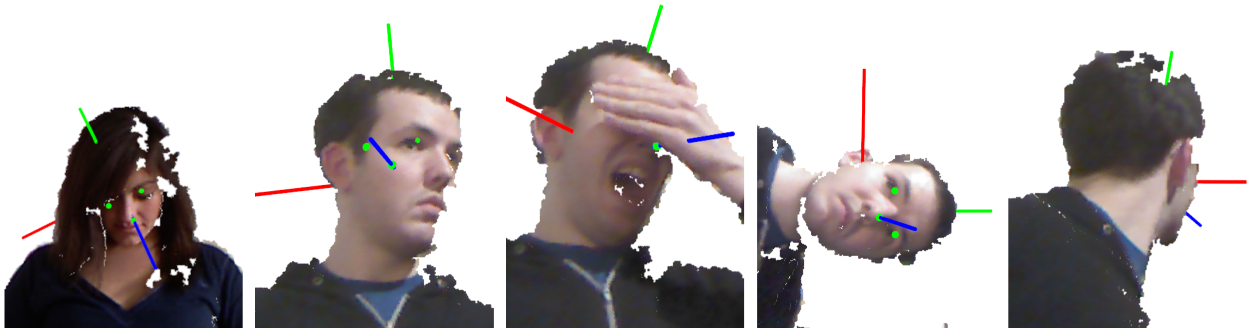


Figure 7. Pictures of different results of our detector. (a) shows a problem occurring with long moving hair, (b) shows a wrong registration on fast head movements, (c) shows the robustness to facial expressions and occlusions ,and (d) and (e) show results for extreme angles.

5. Conclusion and Future Work

We have presented a novel approach for head pose estimation with consumer depth cameras, that works with high precision without prior knowledge of the tracked person and without the training of a detector. To achieve this, we combined an algorithm to generate and track a model of the head with feature based head pose estimation. Our evaluation shows that our algorithm achieves a mean absolute error of $3.62°$ in roll, $2.54°$ in pitch and $2.57°$ in yaw over a high pose range of up to $\pm 120°$ in yaw. It runs with an average frame rate of 48fps on a single core of a modern processor.

In the future we plan to improve our algorithm for generating the head model with means to better ensure the consistency of the model and to improve its accuracy with a method to solve the loop closure problem. We also plan to improve the robustness of the detection of the used features and will also explore the use of other features.

References

[1] M. D. Breitenstein, D. Kuettel, T. Weise, L. Van Gool, and H. Pfister. Real-time face pose estimation from single range images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[2] T. Bär, J. Reuter, and J. Zöllner. Driver head pose and gaze estimation based on multi-template icp 3-d point cloud alignment. In *Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1797 –1802, 2012.

[3] G. Fanelli, T. Weise, J. Gall, and L. Van Gool. Real time head pose estimation from consumer depth cameras. In *Proceedings of the 33rd Annual Symposium of the German Association for Pattern Recognition (DAGM)*, 2011.

[4] S. Gurbuz, E. Oztop, and N. Inoue. Model free head pose estimation using stereovision. *Pattern Recognition*, 45(1):33–42, 2012.

[5] K.-L. Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 2004.

[6] E. Murphy-Chutorian and M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626, 2009.

[7] P. Padeleris, X. Zabulis, and A. Argyros. Head pose estimation on depth data based on particle swarm optimization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 42–49, 2012.

[8] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings of the 3rd International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[9] S. Schulter, C. Leistner, P. Wohlhart, P. M. Roth, and H. Bischof. Alternating regression forests for object detection and pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 417–424. IEEE, 2013.

[10] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[11] Y. Tu, H.-S. Lin, T.-H. Li, and M. Ouhyoung. Depth-based real time head pose tracking using 3d template matching. In *Proceedings of SIGGRAPH Asia*, SA '12, pages 13:1–13:4. ACM, 2012.

[12] Videmo. FaceSDK. `http://videmo.de/`. Accessed: 01.07.2014.

[13] M. Voit, K. Nickel, and R. Stiefelhagen. Neural network-based head pose estimation and multi-view fusion. In R. Stiefelhagen and J. Garofolo, editors, *Multimodal Technologies for Perception of Humans*, volume 4122 of *Lecture Notes in Computer Science*, pages 291–298. Springer Berlin Heidelberg, 2007.

[14] J.-G. Wang and E. Sung. Em enhancement of 3d head pose estimated by point at infinity. *Image Vision Computing*, 25(12):1864–1874, 2007.

[15] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. In *Proceedings of ACM SIGGRAPH*, SIGGRAPH '11, pages 77:1–77:10. ACM, 2011.