A Head Pose Tracking System Using RGB-D Camera

Songnan Li, King Ngi Ngan, and Lu Sheng

The Chinese University of Hong Kong, Hong Kong SAR

Abstract. In this paper, a fast head pose tracking system is introduced. It uses iterative closest point algorithm to register a dense face template to depth data captured by Kinect. It can achieve 33fps processing speed without specific optimization. To improve tracking robustness, head movement prediction is applied. We propose a novel scheme that can train several simple predictors together, enhancing the overall prediction accuracy. Experimental results confirm its effectiveness for head movement prediction.

Keywords: head movement prediction, K-means like retraining, head pose tracking, iterative closest point algorithm.

1 Introduction

RGB-D cameras like Kinect capture both color and depth information in real time. They turn many complex computer vision algorithms into practical systems. In this paper, we introduce a fast Kinect-based head pose tracking system which uses user-specific 3D face template to track head pose. As using 3D face template, eye positions are also provided by the system. Therefore, the system can be used as a natural human computer interface in many attractive applications. For example, it can be applied in gaming or free-viewpoint video to provide user viewpoints; it can be used to control cursor with head motion, which will facilitate people with disabled arms; it can serve as a preprocessing step for free-head-movements gaze estimation, and so on.

In the literature, head pose estimation, face tracking, and eye localization have been investigated intensively. Most of previous studies are based on color information only. Related surveys can be found in [9]. As RGB-D camera becomes affordable, more and more researchers start to base their studies on depth data. For example, depth could be used to locate nose [8] or several nose position candidates [3] to infer head pose. In [5], the head pose estimation algorithm learned a regression model from a large amount of depth patches. In [4], a sparse face model was used for 3D deformable face tracking. Both color and depth information were exploited.

The prior study that is most closely related to our work is perhaps [13]. Although the system introduced in [13] aims at tracking facial expression, rigid head pose tracking is one of its essential processing steps. Like in [13], we generate user-specific dense face template, and try to register this 3D template to depth data using ICP algorithm [2]. The contribution of the paper mainly comes from the registration process. We propose a novel method to train head movement predictors. These head movement predictors can provide a good initialization result for ICP and help it converge more easily. They make our head pose tracking system more robust in case of fast head movements.

This paper is structured as follows. Section 2 describes the system. Functionality and implementation details of each processing component are discussed. Section 3 focuses on the design of head movement predictors. A "K-means" like re-training scheme is proposed, which can be used when several predictors cooperate with each other for prediction. Experimental results verify its effectiveness for head movement prediction. Section 4 concludes the paper.

2 System Description

We use a commercial product to obtain the user-specific face template, which is then segmented manually to exclude regions that typically exhibit strong deformations. This is the offline part of the system which will be investigated further in a future work. Our current research work is only related to the rest of the system, and will be elaborated below.

The workflow of the online part is illustrated in Fig. 1. Either face and nose detection or head movement prediction is used to provide initialization result for ICP tracking, depending on whether or not the previous tracking results are available.

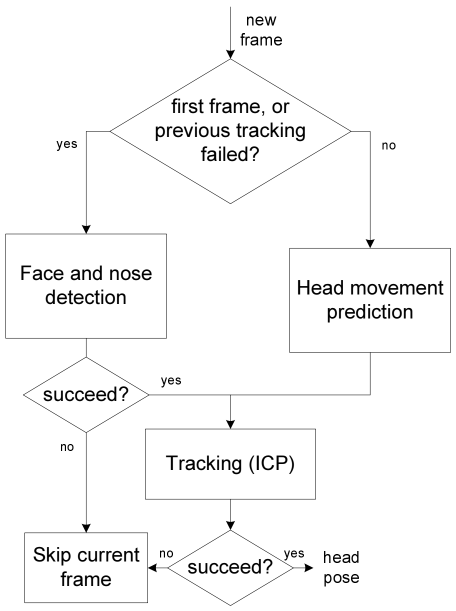


Fig. 1. Workflow of the head pose tracking system

2.1 Face and Nose Detection

As shown in Fig. 1, when it is the first frame or tracking of the previous frame fails, detection using color information is performed. It serves as an initialization/ re-initialization step for tracking. Using Haar-like features and cascade classifiers [12], the system detects the first face region across its search path. Then a nose region is detected using a similar method. To accelerate this detection process, the video resolution is down-sampled from 640×480 to 160×120. The search of nose uses original video resolution, but the search region is constrained to the detected face region.

The 3D position of nose tip $\mathbf{d}_{nose}$ is found by assuming that it is the closest point to the camera in the nose region. Pose of the face template is initialized by registering its nose tip position $\mathbf{f}_{nose}$ to the detected one $\mathbf{d}_{nose}$, and forcing that the template faces towards the camera center. This pose initialization result will be refined in the following step (Section 2.3) using the ICP algorithm.

Besides using detection, previous tracking results if available may be used to predict head pose more accurately. It will help the ICP algorithm to converge quickly. This paper will focus on this part in Section 3.

2.2 Tracking Using the ICP Algorithm

In this system, the face template is considered as a 3D point cloud, $\mathbf{f}_i$, $i \in [1, 2, ..., N_f]$. Given the camera calibration results, the depth image captured by Kinect can be easily transformed into another 3D point cloud, $\mathbf{d}_i$, $i \in [1, 2, ..., N_d]$. ICP algorithm is applied to register the two point clouds, and accordingly generates the head pose information.

The ICP algorithm [2] typically consists of several iteratively processing steps. In our system, it can be summarized as follows:

(1) For each point of the face template $\mathbf{f}_i$, find its matching point $\mathbf{d}_i$ in the point cloud captured by Kinect.
(2) Reject matching pairs $\{\mathbf{f}_i, \mathbf{d}_i\}$ according to predefined conditions.
(3) For the remaining matching pairs $\{\mathbf{f}_i, \mathbf{d}_i\}$, $i \in [1, 2, ..., N_r]$, find an optimal transform matrix $\mathbf{T}$ by minimizing a matching error metric, which quantifies the matching quality of $\{\mathbf{T} \times \mathbf{f}_i, \mathbf{d}_i\}$, $i \in [1, 2, ..., N_r]$.
(4) If the algorithm does not converge, repeat step 1 to 3.

Different approaches can be taken for each processing step, so there are many ICP variants [11]. In our system, projective data association [7] is used to find point correspondences in step one. The choice is mainly because of its superior efficiency compared with, for example, the typically used nearest neighbor method. In step two, a matching pair will be rejected if (a) the distance between $\mathbf{f}_i$ and $\mathbf{d}_i$ is too large (>50mm in our implementation) or (b) $\mathbf{f}_i$ does not have correspondence because of occlusion. Since our face is highly curved, part of it will be self-occluded. The occluded part will not be captured by Kinect, so some of $\mathbf{f}_i$s cannot find their correspondences. The self-occlusion can be detected by analyzing the normal vector of $\mathbf{f}_i$, which is also included in the face template file.

The homogeneous 4×4 transform matrix $\mathbf{T}$ is determined by minimizing this matching error metric:

$$\mathbf{T} = argmin_{\mathbf{T}} \sum_{i=1}^{N_r} \|\mathbf{q}_i \cdot (\mathbf{T} \times \mathbf{f}_i - \mathbf{d}_i)\|^2 \tag{1}$$

where $\mathbf{q}_i$ is the unit normal vector of $\mathbf{f}_i$. A numerical method BFGS [10] is used to optimize this metric. ICP using Eq. (1) as the matching error metric is called point-to-plane ICP. We found experimentally that it performs much better than point-to-point ICP [2] which does not employ normal information. Steps 1 to 3 are repeated until Eq.(1) reaches a local optimum.

With a dual-core 2.33GHz CPU, the face & nose detection algorithm runs at around 15 frames per second (fps). It is only used occasionally for tracking re-initialization. Without specific optimization, tracking using ICP runs at around 33fps. It can be optimized easily using multi-core CPU or GPU due to the parallel nature of the ICP algorithm. The tracking results are very accurate and robust to fast head motion. Fig. 2 illustrates some tracking and re-initialization results. A demo video can be found on the project website[1].

3 Head Movement Prediction

To predict movements, Kalman filter is typically used. Kalman filter is good at modeling linear systems. But head movement is typically nonlinear. For nonlinear movements, usually particle filter is applied. But particle filter needs a large number of samples to estimate a probability distribution, which makes it time consuming. In this section, a new method is proposed for head movement prediction. It uses only three samples and can provide quite accurate prediction results.

3.1 Prediction Model

Perhaps the most intuitive method for 1-dimentional variable prediction should be this simple linear model:

$$x_k = a_0 + a_{-1}x_{k-1} + a_{-2}x_{k-2} + ... + a_{-L+1}x_{k-L+1} \tag{2}$$

where a_is, $i \in \{0, -1, ..., -L+1\}$ are model parameters, x_k is the variable value (e.g., position in millimeter or orientation in degree) at time k, and L is the predictor length. Clearly it cannot model nonlinear head movement. But in our system, the prediction result is not directly used as the head pose of the next frame (as in applications such as head mounted display [1] which directly uses the prediction result as the next head pose to reduce display latency), but as an initialization result for ICP tracking. Therefore, we can use several simple linear models which work together to compensate for their disadvantages.

[1] `http://www.ee.cuhk.edu.hk/~snli/HeadTracking.htm`

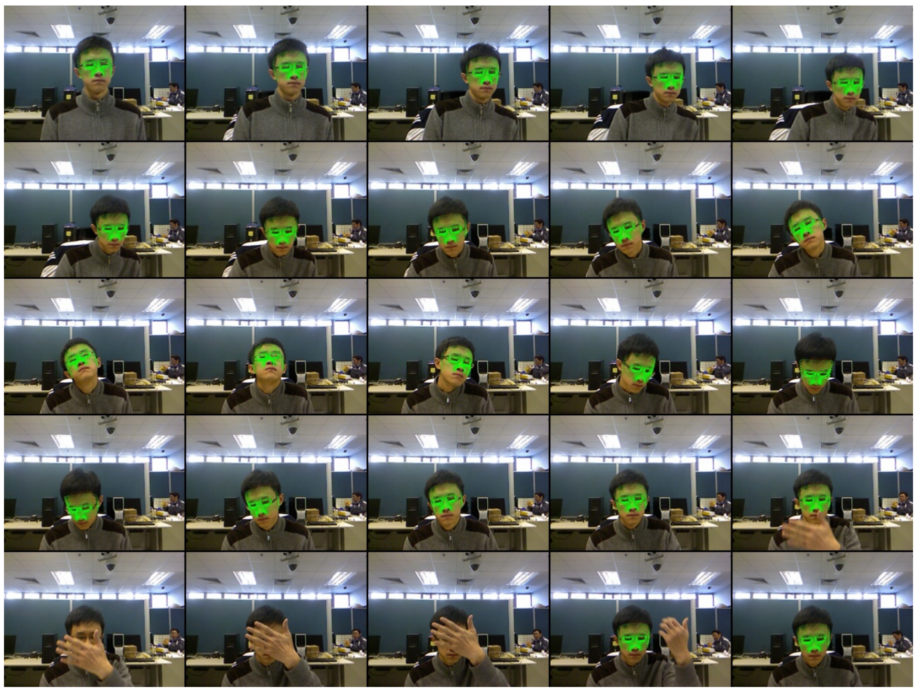


Fig. 2. Illustration of tracking and re-initialization results

In practice, we use the following prediction model instead of (2):

$$\nabla x_k = a_0 + a_{-1}\nabla x_{k-1} + a_{-2}\nabla x_{k-2} + ... + a_{-L+1}\nabla x_{k-L+1} \tag{3}$$

where ∇x_k is equal to $x_k - x_{k-1}$ which indicates change of position or orientation between adjacent frames. Using the same number of parameters, Eq. (3) can better model natural movement. For example, using two parameters, Eq. (3) can model constant velocity ($a_0 = 0$ and $a_{-1} = 1$ assuming the frame sampling rate is constant), while Eq. (2) cannot. As introduced below, parameters are derived by training. It is better to use less parameters, since there will be less possibility for the model to over-fit the training data.

Head motion prediction aims to provide a good initial transform matrix **T** in Eq. (1), so that the ICP algorithm can converge robustly. The 4×4 homogenous transform matrix **T** has only 6-degrees of freedom: $\mathbf{T}_k = [\nabla x_k, \nabla y_k, \nabla z_k, \nabla \phi_k, \nabla \theta_k, \nabla \omega_k]^t$ which corresponds respectively to transitions and rotations along the XYZ axes. For computational efficiency, we assume independence between these six variables and use the same model parameters for prediction:

$$\mathbf{T}_k = a_0\mathbf{1} + a_{-1}\mathbf{T}_{k-1} + a_{-2}\mathbf{T}_k - 2 + ... + a_{-L+1}\mathbf{T}_{k-L+1} \tag{4}$$

3.2 Training

Head movement should have its unique characteristics, which can be captured by tuning the predictor parameters (a_0, a_{-1}, etc.) according to real-world head

Table 1. Three linear predictors with different lengths (L=1, 2, 3), and their prediction errors measured by MSE

	a_0	a_{-1}	a_{-2}	MSE
Predictor A (L=1)	0.009	-	-	240.3
Predictor B (L=2)	0.004	0.585	-	138.7
Predictor C (L=3)	0.005	0.469	0.198	127.7

movement data. As a preliminary experiment, we captured a head movement sequence with $N = 1829$ frames, trying to include many typical head movements. Exhaustive search is used to find a good initialization result for ICP, which converged successfully for all frames in this video sequence. Such a scheme cannot be applied in practice considering its complexity, but can be used to derive ground truth when more accurate tracking devices are unavailable.

It should be noted that transitions (∇x_k, ∇y_k, ∇z_k) are measured in millimeter, while rotations ($\nabla \phi_k$, $\nabla \theta_k$,$\nabla \omega_k$) are measured in degree. The six variables have different variances in the training data. To prevent the training result to be biased towards variables that exhibit larger variances, they are normalized by their respective standard deviation values[2]. Then the predictor parameters are obtained by optimizing this least square equation:

$$\min_{[a_0,a_{-1},\ldots,a_{-L+1}]^t} \left\| \begin{bmatrix} \mathbf{T}_N \\ \mathbf{T}_{N-1} \\ \vdots \\ \mathbf{T}_L \end{bmatrix} - \begin{bmatrix} \mathbf{1}\ \mathbf{T}_{N-1}\ \mathbf{T}_{N-2} \ldots \mathbf{T}_{N-L+1} \\ \mathbf{1}\ \mathbf{T}_{N-2}\ \mathbf{T}_{N-3} \ldots\ \mathbf{T}_{N-L} \\ \vdots \qquad \ddots \qquad \vdots \\ \mathbf{1}\ \mathbf{T}_{L-1}\ \mathbf{T}_{L-2} \ldots\ \mathbf{T}_1 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_{-1} \\ \vdots \\ a_{-L+1} \end{bmatrix} \right\|^2 \quad (5)$$

or

$$\min_{\mathbf{V}} \|\mathbf{P} - \mathbf{Q} \times \mathbf{V}\|^2 \quad (6)$$

which has a closed form solution: $\mathbf{V} = (\mathbf{Q}^t \times \mathbf{Q})^{-1} \times \mathbf{Q}^t \times \mathbf{P}$.

We trained three linear predictors with different lengths L. Their parameters are shown in Table 1. The prediction error is measured by MSE:

$$MSE = \frac{\sum_{k=L}^{N} \|\mathbf{T}_k - \hat{\mathbf{T}}_k\|^2}{6(N - L + 1)} \quad (7)$$

where $\mathbf{T}_k$ is the ground truth, $\hat{\mathbf{T}}_k$ is the predicted one with each of its elements rescaled (by multiplying the corresponding standard deviation value). As expected, the prediction error gradually decreases as the predictor length increases. However, to prevent over-fitting training data, predictor should not be too long. In practice, we chose the maximum predictor length to be 3, as shown in Table 1.

As explained above, in our application we can evaluate several predictions and select the best one as initialization for ICP. Therefore, all three predictions A, B and C can be used together. The notation min(A, B, C) indicates a new

[2] $\sigma_{\nabla x} = 12.86$, $\sigma_{\nabla y} = 8.33$, $\sigma_{\nabla z} = 1.80$, $\sigma_{\nabla \phi} = 0.67$, $\sigma_{\nabla \theta} = 0.97$, $\sigma_{\nabla \omega} = 0.77$.

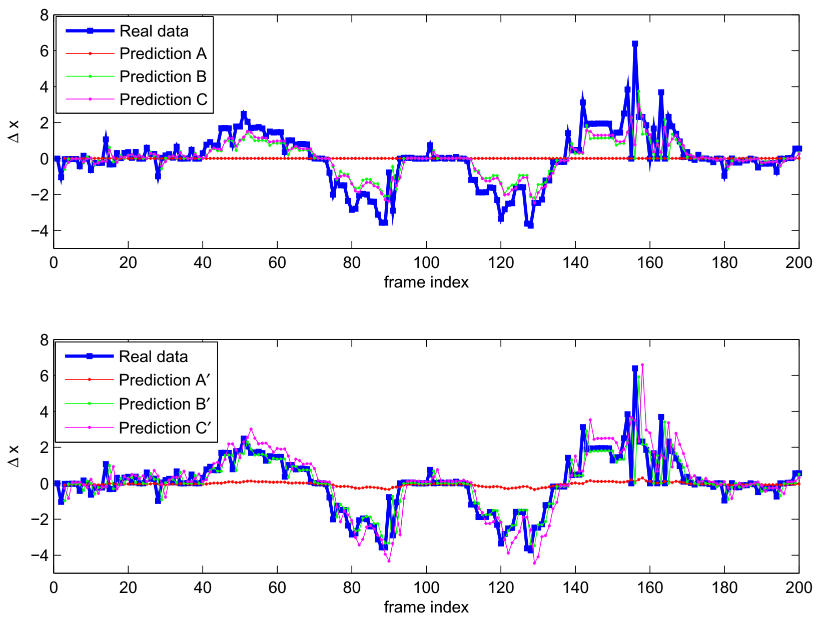


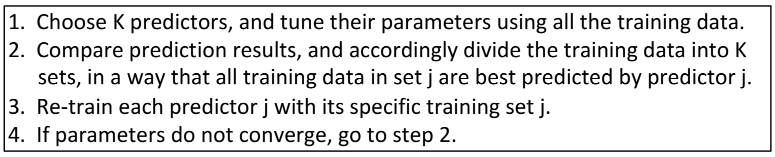
Fig. 3. 200 frames of predictions on ∇x. Upper subfigure illustrates prediction results using predictors A, B, and C as given in Table 1. Bottom subfigure illustrates prediction results using predictors A', B', and C' as given in Table 2.

predictor that combines all three predictors together, and only uses the best predictor among A, B and C for each frame. Compared with the smallest MSE value in Table 1, i.e., 127.7, min(A, B, C) further reduces it to 102.8.

3.3 A "K-means" Like Re-training

The upper subfigure of Fig. 3 shows an example of prediction results, i.e., 200 frames of predictions on ∇x, using three predictors A, B and C given in Table 1. It can be observed that although predictors B and C have different lengths, they have similar prediction results. None of predictors A, B and C can predict large transitions accurately. The reason is that they intend to cope with all types of data, balancing performances of predicting both large and small transitions. This should be difficult for any simple linear predictor.

Therefore, we propose a model re-training scheme. It aims to construct a better prediction algorithm by considering the fact that in practice, several simple predictors may cooperate with each other rather than working alone. It takes a "K-means" like approach. Limited by the paper length, readers interested in K-means cluster algorithm please refer to [6]. A conceptual description of this re-training scheme is given in Fig. 4. In practice, the re-training of predictors A, B and C is conducted in the following way: (1) Set the initial parameters of new predictors A', B', and C' according to Table 1. Parameters of A' are a_0=0.009, a_{-1}=0 and a_{-2}=0. Parameters of B' are a_0=0.004, a_{-1}=0.585 and a_{-2}=0. C'



1. Choose K predictors, and tune their parameters using all the training data.
2. Compare prediction results, and accordingly divide the training data into K sets, in a way that all training data in set j are best predicted by predictor j.
3. Re-train each predictor j with its specific training set j.
4. If parameters do not converge, go to step 2.

Fig. 4. A "K-means" like re-training scheme

Table 2. Predictors after re-training. The last column shows the ratio of best prediction number to all training data for each predictor.

	a_0	a_{-1}	a_{-2}	best/all
Predictor A'	-0.054	0.041	0.039	30%
Predictor B'	-0.008	0.927	0.003	52%
Predictor C'	0.088	0.352	0.888	18%

is the same as C. Notice that the lengths of A', B' and C' are all 3 now. (2) Compare the prediction results, and divide the training data into 3 sets accordingly. Training data in 1^{st}, 2^{nd} and 3^{rd} sets are best predicted by predictors A', B' and C', respectively. (3) Re-train A', B' and C' using training set 1, 2, and 3, respectively. (4) Repeat steps 2 and 3 until the parameter values converge.

The final parameters are shown in Table 2. Table 2 also lists the ratio of best prediction count to all training frames for each predictor. The bottom subfigure of Fig. 3 shows an example of their prediction results. It can be seen that compared with B and C, B' and C' exhibit larger diversity. This will be helpful when using them together to construct a better prediction algorithm, e.g., min(A', B', C'). On the training data, min(A', B', C') further reduces the smallest MSE from min(A, B, C)'s 102.8 to 80.3.

3.4 Performance Test

To test the performance, another head movement sequence is used. It has 717 frames. The ground truth tracking result is derived using exhaustive search as introduced before. The prediction error is measured by MSE as given in Eq. (7) [3]. The model parameters of A, B, C, A', B', C' are shown in Table 1 and 2. From Table 3, it can be seen that the re-training reduces prediction error on this test sequence significantly.

The prediction result is used as an initialization for ICP. The matching error metric of ICP as given by Eq. (1) can be used to evaluate prediction performance. Fig. 5 compares three predictors on this test video. "Identity" corresponds to a naive predictor given by $\mathbf{T}_k = 0$, i.e., all a_i of Eq. (4) are set to zero. Using "Identity", the previous head state is used as an initialization for the next frame. min(Identity, B, C) is similar with min(A, B, C) but A is replaced with

[3] Input data are normalized by the standard deviation values obtained from the training data before prediction. After prediction, the predicted values are rescaled before calculating MSE.

Table 3. Prediction errors of five predictors on a test sequence

	A	B	C	Min(A, B, C)	Min(A', B', C')
MSE	391.0	168.9	153.3	135.7	93.06

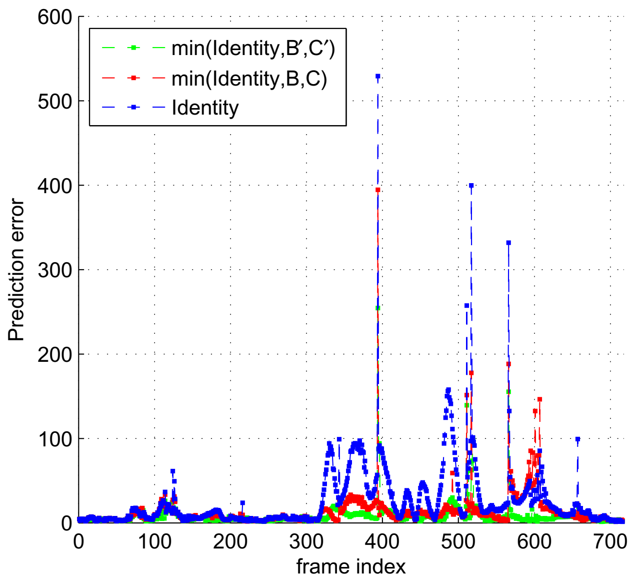


Fig. 5. Prediction errors measured by Eq. (6) on the test sequence

"Identity". It is believed that if the training data is large enough, A should be close to "Identity". And in practice, we find that "Identity" indeed performs better than A. Similarly, min(Identity, B', C') is min(A', B', C') with A' replace by "Identity". It should be noted that for the same frame, inputs of these predictors ($\mathbf{T}_{k-1}$ and $\mathbf{T}_{k-2}$) are different. So it is possible that occasionally "Identity" even performs better than min(Identity, B, C).

As shown in Fig. 5, min(Identity, B', C') outperforms the other two for most frames. Specially, for those extremely difficult frames (head movement with burst acceleration) which are associated with very large prediction errors, min(Identity, B', C') always produce the smallest prediction error, which verifies that the retraining indeed enhances the prediction capability.

4 Conclusion

In this paper, we introduced a Kinect-based head pose tracking system. Point-to-plane ICP is used to register a user-specific face template to the point cloud captured by Kinect. To make sure that ICP can converge robustly even with fast head motion, a reliable head movement prediction algorithm is required to provide ICP with a good initialization result. Our head movement prediction algorithm uses three simple linear predictors. To enhance overall prediction accuracy, a novel method is proposed to train these simple predictors. We name this method as "K-means" like re-training to emphasize its similarity with the K-means clustering algorithm. It automatically clusters the training data into K groups and trains K predictors, respectively. Although any one of these predictors trained using this method cannot provide a good estimate for every frame, there will be one of them that can predict quite accurately. Therefore, when they are used together, good performance can be expected. It should be noted that the proposed training method can also be applied to other predictors, such as Kalman filters. This is one aspect of our future work. Other aspects include using professional tracking device to obtain accurate training data, and code optimization.

References

1. Azuma, R., Bishop, G.: Improving static and dynamic registration in an optical see-through hmd. In: Proceedings of the 21st Conference on Computer Graphics and Interactive Techniques, pp. 197–204. ACM, USA (1994)
2. Besl, P., McKay, H.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2), 239–256 (1992)
3. Breitenstein, M., Kuettel, D., Weise, T., Van Gool, L., Pfister, H.: Real-time face pose estimation from single range images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8 (2008)
4. Cai, Q., Gallup, D., Zhang, C., Zhang, Z.: 3D deformable face tracking with a commodity depth camera. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part III. LNCS, vol. 6313, pp. 229–242. Springer, Heidelberg (2010)
5. Fanelli, G., Weise, T., Gall, J., Van Gool, L.: Real time head pose estimation from consumer depth cameras. In: Mester, R., Felsberg, M. (eds.) DAGM 2011. LNCS, vol. 6835, pp. 101–110. Springer, Heidelberg (2011)
6. Hartigan, J.: Clustering algorithms. John Wiley Sons (1975)
7. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In: Proc. UIST, pp. 559–568 (2011)
8. Malassiotis, S., Strintzis, M.G.: Robust real-time 3D head pose estimation from range data. Pattern Recognition 38, 1153–1165 (2005)
9. Murphy-Chutorian, E., Trivedi, M.: Head pose estimation in computer vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(4), 607–626 (2009)
10. Nocedal, J., Wright, S.: Numerical optimization. Springer series in operations research. Springer (2006)
11. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Proceedings of International Conference on 3-D Digital Imaging and Modeling, pp. 145–152 (2001)
12. Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vision 57(2), 137–154 (2004)
13. Weise, T., Bouaziz, S., Li, H., Pauly, M.: Realtime performance-based facial animation. In: ACM SIGGRAPH 2011, pp. 1–10. ACM, USA (2011)