Real-time Head Orientation from a Monocular Camera using Deep Neural Network

Byungtae Ahn, Jaesik Park, and In So Kweon

KAIST, Republic of Korea
[btahn,jspark]@rcv.kaist.ac.kr, iskweon77@kaist.ac.kr

Abstract. We propose an efficient and accurate head orientation estimation algorithm using a monocular camera. Our approach is leveraged by deep neural network and we exploit the architecture in a data regression manner to learn the mapping function between visual appearance and three dimensional head orientation angles. Therefore, in contrast to classification based approaches, our system outputs continuous head orientation. The algorithm uses convolutional filters trained with a large number of augmented head appearances, thus it is user independent and covers large pose variations. Our key observation is that an input image having 32×32 resolution is enough to achieve about 3 degrees of mean square error, which can be used for efficient head orientation applications. Therefore, our architecture takes only $1ms$ on roughly localized head positions with the aid of GPU. We also propose particle filter based post-processing to enhance stability of the estimation further in video sequences. We compare the performance with the state-of-the-art algorithm which utilizes depth sensor and we validate our head orientation estimator on Internet photos and video.

1 Introduction

Head pose estimation is crucial for face related applications such as face recognition, facial expression recognition, driver state monitoring, gaze estimation, etc. Accordingly, a variety of methods have been proposed for more than two decades [1]. In the context of computer vision, head pose estimation infer the position and orientation (roll, pitch, and yaw) of head from a face image.

Existing approaches can be categorized into two methods: appearance based methods and model based methods. Appearance based methods [2–12] use visual feature of the whole face appearance with machine learning techniques. The methods are relatively robust to large head pose variation and low image resolution. However, most of them utilize discrete head poses for training and treat the head pose estimation as a classification problem. As a result, the estimates are quantized (typically more than 10°) as well. Model based methods [13–18] use geometric cues or non-rigid facial models. Model based methods have advantages that the outputs are continuous values; not discrete. Also they can obtain not only head pose but also facial feature locations for various applications. However, since their performance heavily rely on facial feature localization, the model based methods are sensitive to large variation of head pose, facial expression, and low resolution of input image.

The objective of this paper is to do head orientation estimation that is accurate, continuous, operating beyond real time, and robust to large variation of head pose and low resolution. We achieve this by exploiting deep neural network as a data regression manner. We demonstrate that the proposed estimator outperforms previous literatures. Our approach is adequate for real time applications such as driver drowsiness detection, gaze estimation, and face verification.

2 Related Works

Appearance based methods These methods seek relationship between 3D face pose and its appearance on 2D image. Balasubramanian *et al.* [9] and Foytik and Asari [2] presented manifold embedding frameworks which maps the high-dimensional space of face appearance to low-dimensional manifolds. The latter paper introduces a framework composed of two steps, in which head pose is estimated in a coarse-to-fine manner. Gruji *et al.* [8] utilized image retrieval which compares an input image of head to a set of large exemplars. The initially estimated head orientation is refined using the candidate images in the database. The reported test error of [2, 8] on Pointing'04 dataset [19] is larger than 13°. Huang *et al.* [5] used Gabor feature based random forests as the discrete label classifier. They combined the random forest with linear discriminative analysis (LDA) to improve the discriminative power. Zhu and Ramanan [3] proposed a unified model for face detection, head pose estimation, and facial landmark localization. They use a mixture of tree-structured part models to find topological changes due to rotation along yaw axis. Though it conducts unified task, it classifies just a few discrete yaw angles of head poses, and the computation takes a few seconds per VGA resolution image.

Compared to those discrete labeling approaches, BenAbdelkader [6] and Ji *et al.* [4] treated head pose estimation as a nonlinear regression problem which computes continuous 3D pose. Other approaches [10–12] exploited depth information for continuous head pose estimation. Breitenstein *et al.* [10] aligned a range image with reference poses. Their GPU implementation operates in 10 fps. Fanelli *et al.* [12] introduced a random forest based voting framework for real-time and continuous head pose estimation. They also extended it to 3D facial feature localization. They provide an head pose database containing tuples of color, depth and ground truth head pose. The use of depth data has some advantages that it can be available even at night and can generate 3D face model, but a specific device is required. Also the device cannot be used in outdoors because of its sensing mechanism.

Model based methods In constrast to most of appearnce based methods, model based methods output continuous head pose. Hu *et al.* [13] roughly estimated face pose by using asymmetric distribution of facial components. The pose is refined with 3D-to-2D geometrical model. Active shape models (ASM) [15] and active appearance models (AAM) [16] are very popular statistical models of face. They were proposed for facial landmark localization first, but have been extended for estimating head pose [17]. Morency *et al.* [18] presented generalized adaptive view-based appearance model (GAVAM) for stable head pose estimation, which takes some benefits of automatic initialization, user-independence, and key frame tracking. These methods generally depend on some specific facial landmarks, so they are sensitive to initialization, large variation of head pose or facial expression, occlusions, and resolution of input image.

Deep Convolutional Neural Network As graphic processing unit (GPU) has been developed, and accessibility to big data has become easy, deep learning techniques has been actively studied. Among those deep learning methods, convolutional neural network (CNN) [20] has been successfully applied to computer vision tasks such as image classification [21], pedestrian detection [22], and image denoising [23]. Recently, deep convolutional neural network (DNN) are widely utilized for face related applications and body pose estimation as well. Sun *et al.*[24] and Zhou *et al.* [25] introduced DNN into coarse-to-fine facial feature localization. The former paper proposed three-level cascaded structure composed of one DNN and two shallow neural networks. They also analyzed on effects of some schemes such as absolute value rectification and local weight sharing on facial feature localization. Toshev and Szegedy [26] appiled DNN to human body pose estimation, namely DeepPose. They designed DNN architecture composed of regressor and refiner. The architecture is used for every body joint individually, and the outputs are linked to each other for building the body pose. They report state-of-the art performance.

Inspired by recent success of DNN based approaches, we design a DNN architecture for estimating head orientation. We found that DNN architecture is appropriate for head orientation estimation. In our experiment, we observe that it outperforms previous approach which exploits depth data while we use only gray scale images. Especially, we analyze the effects of input image size, the number of layers, and the number of feature maps. We suggest a novel head orientation estimator showing remarkable accuracy in $1ms$.

3 Preliminaries: Representation of Head Pose

Before introducing our approach, we provide preliminary discussion for describing and displaying head pose. Compared to 6D description of object's pose which is general, the head pose in image coordinate can be described as $(x_h, y_h, \psi, \theta, \phi)$. $\mathbf{x}_h = (x_h, y_h)$ is head position in image coordinate and a triplet (ψ, θ, ϕ) stands for the rotation angles of roll, pitch, and yaw. They are all bounded in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and [0,0,0] denotes frontal view of the head. We use the conventional definition of (ψ, θ, ϕ) in right-handed Cartesian coordinates as shown in Fig. 1. According to the definition, ψ and θ correspond to clockwise rotation angles about x-axis

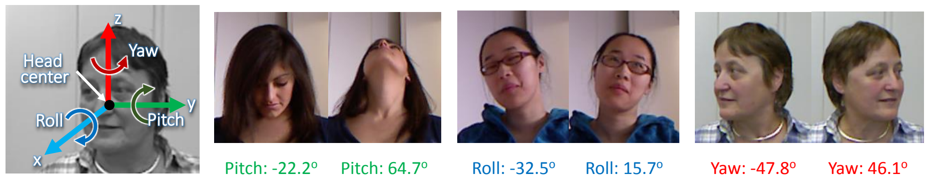

Fig. 1. Representation of head orientation. We use conventional definition of roll, pitch and yaw rotation directions shown in the left figure. Some examples of rotation angles and their corresponding head images are shown in the right side. The dataset is provided by Fanelli *et al.* [12].

and y-axis. ϕ corresponds to counter clockwise rotation angle about z-axis. The 3D head orientation matrix $R_{head} = R_\psi R_\theta R_\phi$ is then determined as

$$R_\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}, R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, R_\phi = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

As a counter conversion, unique (ψ, θ, ϕ) is determined from R_{head} as

$$(\psi, \theta, \phi) = \left(\arctan(\frac{R_{32}}{R_{33}}), \arctan(\frac{-R_{31}}{\sqrt{R_{32}^2 + R_{33}^2}}), \arctan(\frac{R_{21}}{R_{11}}) \right), \quad (2)$$

where R_{ij} is the element of R_{head} at i-th row and j-th column.

The head pose $(x_h, y_h, \psi, \theta, \phi)$, can be visualized by means of the 3D axis and a circle on yz plane around the head as shown in Fig. 6. To do so, we transform (ψ, θ, ϕ) into R_{head} and we project the axes and the circle onto the input image by using an orthographic projection matrix:

$$P = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x_h \\ R_{21} & R_{22} & R_{23} & y_h \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3)$$

where P is defined in homogeneous coordinate.

4 Proposed method

In this section, our head pose estimation approach is introduced. We assume that we have head position and its corresponding scale. In our implementation, we utilize robust head detection algorithm by Zhu *et al.* [3] which uses tree structured part model for elastic deformation.

4.1 Deep Learning Architecture for Head Orientation

We review convolutional neural network (CNN) briefly and introduce our design for head orientation task. Figure 2 illustrates the proposed structure of DNN.

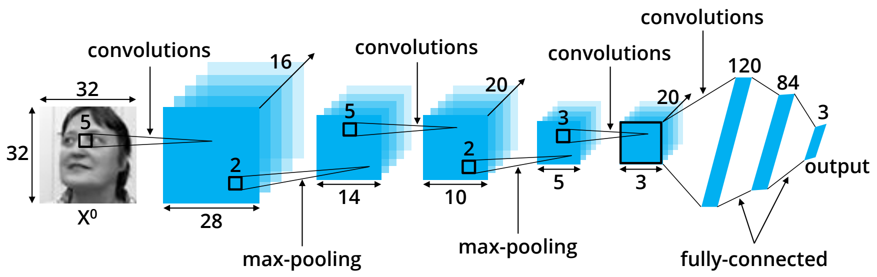

Fig. 2. Proposed structure of deep neural network (referred as N2 in Table 1) for head orientation estimation. It uses 32×32 pixels gray scale image as an input. The output is head orientation (ψ, θ, ϕ).

The trained filters in DNN minimize the following loss error:

$$E(X_i; W) = \sum_i ||Y_i - f(X_i; W)||_2^2, \tag{4}$$

where i indicates an index of training samples, W is a set of weight values in convolution filters, X is estimated angles (ψ, θ, ϕ), and Y denotes the target (ground truth) of head orientation. Training CNN consists of two phases: prediction and update. Prediction means feed forward through the network. Update means evolving weights and biases between layers by error back-propagation. In the prediction phase, one convolutional layer accompanies three steps. First, convolution operation is performed on the input image with trained filters. Second, the outputs of the convolutions are passed through an activation function. Third, they are downscaled (sub-sampling) for introducing small translation invariance and improving generalization. Sub-sampling step can be disregarded according to applications. In update phase, loss errors are calculated at the end node (the output of the network). Based on the errors, the weights and biases of the network are updated from the last layer to the first layer by stochastic gradient descent (SGD). This is called backward propagation of errors (or back-propagation). Hyperbolic tangent, sigmoid, and rectified linear unit (ReLU) [21] functions are commonly used as the activation function. The sigmoid function $f(x) = (1 + e^{-\beta x})^{-1}$ maps $[-\infty, +\infty] \rightarrow [0, 1]$, while hyperbolic tangent function $f(x) = \tanh(x)$ maps $[-\infty, +\infty] \rightarrow [-1, +1]$. Thus, the outputs from the sigmoid function are typically not close to zero on average, while average of the outputs from hyperbolic tangent function is close to zero. In this aspect, with a normalized dataset whose mean and variance are 0 and 1 respectively, the hyperbolic tangent function is recommendable due to convergence during gradient descent [27]. ReLU tends to train faster than other activation functions [21].

Now, we introduce our DNN design for head orientation estimation. Our DNN structure follows a principle introduced by Coates *et al.* [28]. According to the literature, since our dataset may not cover head appearances of every people, we use small filter size (5×5 which is smallest in convention) and smallest

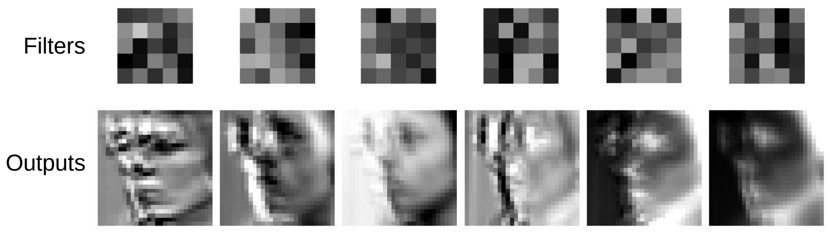

Fig. 3. Some trained filters and their outputs of an input image in the first convolutional layer. The sizes of filters and the outputs are 5×5 and 28×28 pixels respectively.

convolutional stride (1 pixel). Regarding the number of layers, we follow insights from [24], which states that performance improves as the number of layers increases (more than 3 at least). The number of filters is also an important factor on accuracy. Our design is composed of 4 convolutional layers having 16, 20, 20, and 120 filters respectively, which produces an acceptable trade-off between the performance and computational speed.

Our architecture takes an input image of 32×32 pixels which is relatively small compared to other DNN architectures for other face applications [3, 24, 25, 29]. We normalize intensities of an input image, so that the mean and variance are 0 and 1 respectively. This allows us to use hyper-tangent as the activation function. Max-pooling is performed after convolutional layers. The outputs of the first convolutional layer followed by max-pooling is the input of the second convolutional layer. They are convolved with 20 filters of 5×5 pixels. In the same manner, third and forth convolutional layers take the outputs of the previous layers as input, and convolve it with 20 and 120 filters of 5×5 and 3×3 pixels respectively. The max-pooling is not conducted in the third and fourth convolutional layer. The l-th convolutional layer is defined as

$$X_v^{l+1} = \tanh\left(\sum_{u=1}^{I} W_{uv}^l \otimes X_u^l + b_v^l\right), \tag{5}$$

where W_{uv}^l and X_u^l are the trained filter and the image patch, and u and v indicate the index of input and output channels respectively. For example, in the first convolutional layer, $u = 1$ and $v \in \{1, \cdots, 16\}$. Therefore, X_v^{l+1} is the output from v-th channel which is the input to the $(l+1)$-th layer. b_v means the bias vectors, and $\otimes$ denotes convolution operator. Figure 3 shows some of the trained filters in the first convolutional layer. Note that the features are not correlated, and edges and some important parts for estimating head orientation (e.g. eye, nose, and chin) are enhanced in their output.

The first and second fully connected layers following convolutional layers are composed of 120 and 84 neurons respectively. The fully connected layer is performed with function $y_j = \tanh\left(\sum_{i=0}^{m-1} x_i \cdot w_{i,j} + b_j\right)$, for $j \in \{0, \cdots, n-1\}$, where m and n are the number of neurons at the previous layer and current layer respectively. Equation (4) is non-linear due to the activation function. We solve it by back-propagation method using stochastic gradient descent (SGD) as in [21].

4.2 Temporally Stable Head Pose Estimation

Given an input video, if we handle input frames independently, the estimated head orientation may be temporally unstable since the head appearance often changes abruptly due to shadows or occlusions. In order to obtain stable head orientation in the time domain, we apply Bayesian sequential estimation which uses the past observation to update the posterior distribution and to predict the current state. The distribution required for filtering procedure can be effectively approximated by sequential Monte Carlo estimation, or known as particle filter [30]. We empirically choose particle filter instead of linear filter such as Kalman due to high non-linearity of state changes. We operate two particle filters, which are for head orientation and head position due to the multi-modality and weak correlation between the two states. For propagating particles, we use a first-order dynamic model which regards constant angular or positional displacements over the period $[t-1, t]$. In this manner, head orientation state $\mathbf{o} = (\mathbf{s}, \mathbf{d})$ is updated as:

$$\mathbf{s}_t = \mathbf{s}_{t-1} + \mathbf{d}_{t-1}\Delta t + \epsilon_{\mathbf{s}}, \tag{6}$$

$$\mathbf{d}_t = \mathbf{d}_{t-1} + \epsilon_{\mathbf{d}}, \tag{7}$$

where $\mathbf{s} := (\psi, \theta, \phi)$ represents head orientation state, $\mathbf{d} := (d_\psi, d_\theta, d_\phi)$ is angular displacement, subscript t notes time stamp at t, and $\epsilon_{\mathbf{s},\mathbf{d}}$ are process noise come from zero-mean Gaussian distribution. We exploit the *bootstrap filter* where the density of state transition is used for estimating the probability function [31]. The importance weight $w^i_{t,ang}$ for i-th particle $\mathbf{o}^i_t$ is described by:

$$w^i_{t,ang} \propto w^i_{t-1,ang} \times p(\mathbf{o}_{t,obs}|\mathbf{o}^i_t), \tag{8}$$

where $\mathbf{o}_{t,obs} = (\mathbf{s}_{t,obs}, \mathbf{d}_{t,obs})$ is a new observation at t and $\mathbf{o}^i_t$ is the propagated particles. $w_{t-1,ang}$ can be regarded as constant since resampling is performed on fixed number of particles. We define $w^i_{t,ang}$ as:

$$w^i_{t,ang} = \exp\left(\frac{\|\mathbf{o}_{t,obs} - \mathbf{o}^i_t\|^2}{\sigma^2_{ang}}\right), \tag{9}$$

Note that we have another state $\mathbf{h} = (x, y, v_x, v_y)$ which represents head position and its velocity in image domain. For this state, the importance weight $w_{t,pos}$ for particle $\mathbf{h}^i$ is defined as:

$$w^i_{t,pos} = \exp\left(\frac{f(x^i, y^i)^2}{\sigma^2_{pos}}\right), \tag{10}$$

where $f(\cdot)$ is 2D confidence map built up by head detector. Since $\mathbf{h}$ also uses constant velocity model, it is similarly updated as Eq. (6) and Eq. (7).

5 Experimental Result

In this section, we provide experimental results in various aspects. First, we evaluate networks while altering parameters of the networks such as the number of feature maps and size of input image with the depth of networks. We will also discuss the effect of particle filter as a post processing. Finally, the proposed method will be compared with the state-of-the-art method [12].

5.1 Dataset for Evaluation

We evaluate our method using Biwi Kinect Head Pose Database [12]. The dataset contains 15,678 upper body images of 20 people (4 people were recorded twice but they appear different hair style and clothing), and ground truth head pose information from user-specific 3D template based head tracker [32]. It provides 3D rotation matrix for head orientation. By using Eq. (2), we convert the rotation matrix into (ψ, θ, ϕ). The triplet is used for training described in Sec. 4. The head orientation covers about $\pm 75°$ for yaw, $\pm 60°$ for pitch, and $\pm 50°$ for roll. The dataset provides depth to facial center as well. From the perspective camera model without lens distortion, the size of head image patch is determined as $\frac{fR}{Z}$ where f is focal length, R is radius of head, Z is metric depth to head center. We use $R = 120mm$ and fix it over the evaluation. The extracted head images are resized to 100×100 pixels.

Among 15,678 patches, we randomly selected a subset of 2,178 patches as our validation set, and remaining 13,500 patches were used for training. For the training samples, we first did data augmentation on the extracted patches to avoid over-fitting. We did this by randomly cropping the extracted patches. The size of smaller patch varies from 86×86 to 100×100 pixels. Then, the augmented patches are resized to 32×32 pixels for the proposed DNN. At test time, five patches of 86×86 pixels are extracted from each 100×100 pixels of input patch (four from each corner patch and one from center). The five patches are also resized to 32×32 pixels. Note that the size of input patch can be 64×64 pixels as well, which will be discussed in Sec. 5.2. All training and test patches are gray-scaled and their intensity values are modified by histogram normalization. We used GPU accelerated implementation, and training continues until convergence.

5.2 Analysis on Various Network Structures

In order to find most efficient and effective network, we design various types of DNN structures with different parameters (the number of feature maps, and the size of an input image, and the number of convolutional layers) on estimating head orientation. Note that the image size decreases when it passes each layer. Therefore, the number of layer and size of input have dependency. Our selected configurations are summarized in Table 1. N2 containing four convolutional layers is the proposed DNN structure illustrated in Figure 2. The networks N1–N4 include four convolutional layers, and perform with the input images of 32×32 pixels. The networks N5–N8 contain five convolutional layers, with the input

Table 1. Summary of DNN structures. $I(s, s)$ denotes a square input image of s pixels on a side. $C(k, n)$ means convolutional layer with square filters of k pixels on a side, where n is the number of filters. Pooling layer is denoted by $P(p)$, where p is the size of the square pooling regions. $F(e)$ indicates fully connected layer, where e is the number of neurons.

	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
N1	I(32,32)	C(5,30)	P(2)	C(5,30)	P(2)	C(3,30)	C(3,120)	F(84)	F(3)		
N2	**I(32,32)**	**C(5,16)**	**P(2)**	**C(5,20)**	**P(2)**	**C(3,20)**	**C(3,120)**	**F(84)**	**F(3)**		
N3	I(32,32)	C(5,10)	P(2)	C(5,20)	P(2)	C(3,20)	C(3,120)	F(84)	F(3)		
N4	I(32,32)	C(5,10)	P(2)	C(5,10)	P(2)	C(3,10)	C(3,120)	F(84)	F(3)		
N5	I(64,64)	C(5,30)	P(2)	C(5,30)	P(2)	C(4,30)	P(2)	C(3,30)	C(3,120)	F(84)	F(3)
N6	I(64,64)	C(5,16)	P(2)	C(5,20)	P(2)	C(4,20)	P(2)	C(3,20)	C(3,120)	F(84)	F(3)
N7	I(64,64)	C(5,10)	P(2)	C(5,20)	P(2)	C(4,20)	P(2)	C(3,20)	C(3,120)	F(84)	F(3)
N8	I(64,64)	C(5,10)	P(2)	C(5,10)	P(2)	C(4,10)	P(2)	C(3,10)	C(3,120)	F(84)	F(3)

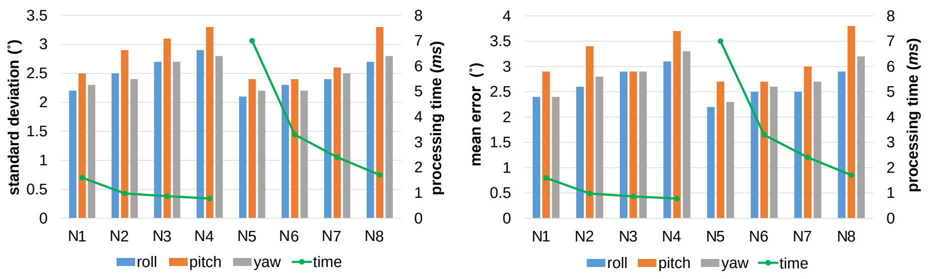

Fig. 4. Mean and standard deviation of the errors, and processing speed of various networks defined in Table 1.

images of 64 × 64 pixels. Figure 4 and Table 2 show the performance of the networks in Table 1.

Figure 4 shows the comparison results on mean and standard deviation of the errors. Processing time shown in Fig. 4 over the eight DNN structures are tested on Nvidia^{TM} GTX Titan Black 6GB GPU. Results show that the performance can be slightly improved when the networks have more than four convolutional layers and use high quality input images of 64 × 64 pixels. However, in these networks, processing is much slower. It seems that four convolutional layers with low quality images of 32 × 32 pixels are satisfied for accurate head orientation estimation. 32×32 resolution is approximately two times smaller than [29] which is designed for recovering canonical view with important parts of face images. In face orientation problem, we believe relative location of the chin, nose and eyes regardless of the individual person still works as a useful cue in 32×32 resolution, even though they are not shown obviously. In addition, due to the reduced dimensions, we could achieved impressive computational time. When comparing the networks having the same depth, as the number of feature maps increases, the result tends to be improved. However, since the processing time is increased as well, deciding the number of feature maps depends on its applications.

Table 2. Mean and standard deviation of the errors, and processing time of various networks. N1-N4 structures are composed of four convolutional layers, and N5-N8 structures consists of five convolutional layers.

	Mean error ± standard deviation (°)			time (ms)
	Roll	Pitch	Yaw	
N1	2.4±2.2	2.9±2.5	2.4±2.3	1.60
N2	**2.6±2.5**	**3.4±2.9**	**2.8±2.4**	**0.98**
N3	2.9±2.7	2.9±3.1	2.9±2.7	0.87
N4	3.1±2.9	3.7±3.3	3.3±2.8	0.78
N5	2.2±2.1	2.7±2.4	2.3±2.2	7.00
N6	2.5±2.3	2.7±2.4	2.6±2.2	3.30
N7	2.5±2.4	3.0±2.6	2.7±2.5	2.41
N8	2.9±2.7	3.8±3.3	3.2±2.8	1.71

5.3 Temporally Stable Head Orientation Estimation

We validate our particle filter based module described in Sec. 4.2 on the RobeSafe [33] dataset. The video contains the driver who moves his/her head smoothly during driving. Note that the driver data is not used for training in our pipeline. Figure 5 shows estimated head orientation over some periods. The estimated head orientation however, shows inconsistent orientation over adjacent time due to abrupt change of the appearance and occlusion (around 15th frame in the Fig. 5) not by the physical head movement.

5.4 Comparison with Fanelli *et al.* [12]

We compare with the state-of-the-art approach for real time head pose estimation, which uses random forest regression with depth sensor, Kinect. They provide and use the same dataset in our experiments. Table 2 shows comparison results on mean and standard deviation of the errors, and processing time of both [12] and ours. Note that all the results on accuracy and precision from the networks we design (Table 2) significantly outperform those of the state-of-the-art approach. While the method in [12] compares internal depth values from extracted random patches for voting head poses, our DNN based approach uses filters automatically learned from many training images without handcraft low level features (intensity difference, edge etc.). Our approach results in implicitly extracting important high level information (relative position of eyes, nose, chin etc.). In addition, the approach using depth values from the Kinect sensor may be affected by noise and low-resolution of depth maps. In contrast, the level of noise in grey scale image is lesser than that of depth map, which is another benefit. Some examples of the estimation are shown in Figure 6, where the center of the white circle with radius $120mm$ is on the head center. It demonstrates that our method is reasonable even though the person has various facial expressions and poses. We believe that it is a result from training a large database which consists of various facial expressions and poses. Given roughly localized head

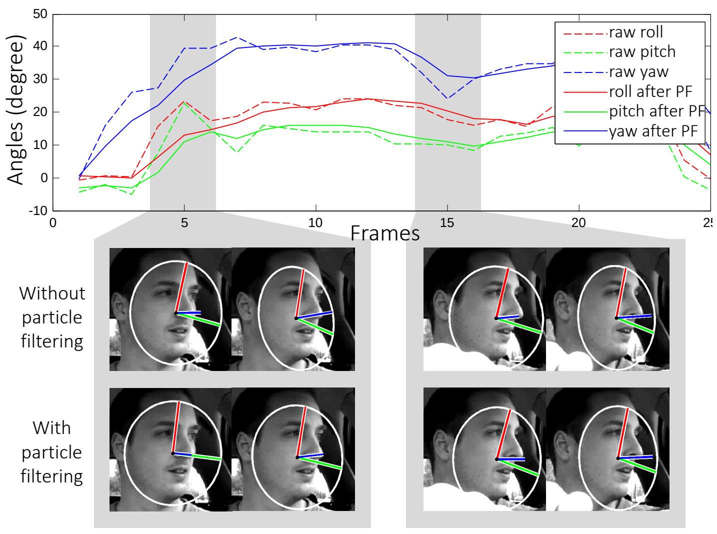

Fig. 5. Validation on RobeSafe driver mornitoring dataset [33]. The method in Sec. 4.2 stabilizes abrupt changes of head orientation which is occurred by shadow and occlusion not by physical movement of the head. The comparison of the head orientations without and with particle filtering is displayed.

position, our approach requires less than $1ms$ to estimate head orientation. For comparing computational time, we analyze with the reported time in [12], although [12] performs face detection and head orientation simultaneously. Since [12] finds the head region abruptly by thresholding depth values, their reported time is mainly for processing depth values for head orientation.

Figure 7 illustrates the normalized success rates of the estimations on the validation set for each 15×15 degrees. Angular error below $15°$ is regarded as a success, and the background color of the the heat map reflects the number of images present in each region. In almost all regions, the estimates show the results of 100% or close to 100% success rates, which outperform the equivalent plot in [12]. Also, it shows that the algorithm works well over large variations of head orientations.

6 Conclusion

In this work, we introduce an efficient and accurate method for estimating head orientation. Inspired by the remarkable success of deep neural network which automatically learns desirable features, we design network structure which achieves notable performance and speed comparable to the state-of-the-art algorithm. We tested our algorithm on various types of video and photos. Possible application scenarios include measuring driver's attention, robust face recognition and saliency estimation. Our future work is designing a general-purpose head detec-

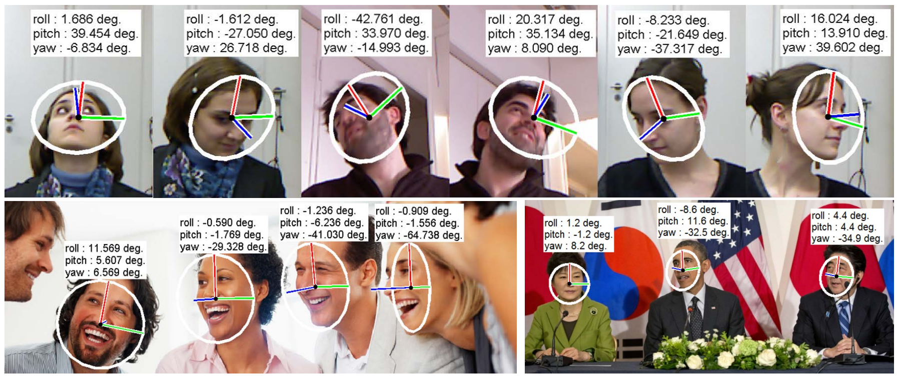

Fig. 6. Our head orientation estimation on validation set [12] (first row) and the web photos (second row).

Table 3. Comparison with Fanelli *et al.* [12] on mean and standard deviation of the errors, and processing time. The test environment used in [12] is a 2.67GHz Intel Core i7 CPU, and ours is the same level of CPU with Nvidia^{TM} GTX Titan Black 6GB GPU. Our time is only for estimating head orientation whereas [12] includes time for abrupt face detection using depth map.

	Mean error ± standard deviation (°)			Time (ms)
	Roll	Pitch	Yaw	
Fanelli stride 5	5.4±6.0	3.5±5.8	3.8±6.5	44.7
Fanelli stride 10	5.5±6.2	3.6±6.0	4.0±7.1	17.8
Fanelli stride 15	5.5±6.2	3.8±6.4	4.2±7.8	10.7
Ours N2	**2.6±2.5**	**3.4±2.9**	**2.8±2.4**	**0.98**

tion algorithm as well to make a comprehensive deep neural network for 5D head pose estimation. We expect that the complete system will boost up the accuracy and usefulness in practice.

Acknowledgement We appreciate constructive comments from anonymous reviewers. This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. 2010-0028680).

References

1. Murphy-Chutorian, E., Trivedi, M.M.: Head pose estimation in computer vision: A survey. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **31** (2009) 607–626
2. Foytik, J., Asari, V.K.: A two-layer framework for piecewise linear manifold-based head pose estimation. Int'l Journal of Computer Vision (IJCV) **101** (2013) 270–287
3. Zhu, X., Ramanan, D.: Face detection, pose estimation and landmark localization in the wild. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2012) 2879 –2886

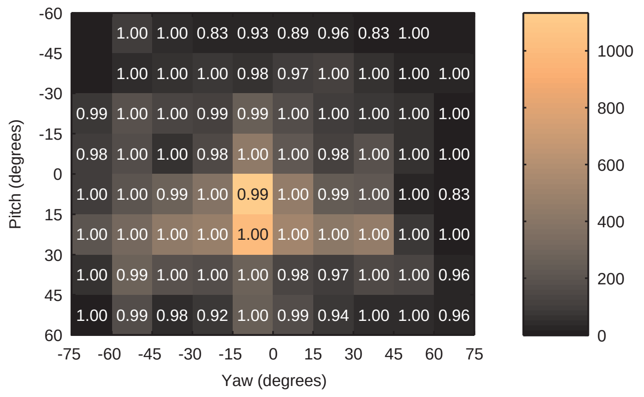

Fig. 7. Normalized success rates. Angular error below 15° is regraded as success. The background color represents the number of images in each bin, as illustrated by the side bar.

4. Ji, H., Liu, R., Su, F., Su, Z., Tian, Y.: Robust head pose estimation via convex regularized sparse regression. In: Proc. of Int'l Conf. on Image Processing (ICIP). (2011) 3617 –3620
5. Huang, C., Ding, X., Fang, C.: Head pose estimation based on random forests for multiclass classification. In: Proc. of Int'l Conf. on Pattern Recognition (ICPR). (2010) 934–937
6. BenAbdelkader, C.: Robust head pose estimation using supervised manifold learning. In: Proc. of European Conf. on Computer Vision (ECCV). (2010) 518–531
7. Aghajanian, J., Prince, S.J.: Face pose estimation in uncontrolled environments. In: Proc. of British Machine Vision Conf. (BMVC). (2009) 1–11
8. Gruji, N., Ili, S., Lepetit, V., Fua, P.: 3d facial pose estimation by image retrieval. In: 8th IEEE Intl Conference on Automatic Face and Gesture Recognition. (2008)
9. Balasubramanian, V.N., Ye, J., Panchanathan, S.: Biased manifold embedding: A framework for person-independent head pose estimation. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2007) 1–7
10. Breitenstein, M.D., Kuettel, D., Weise, T., van Gool, L.: Real-time face pose estimation from single range images. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2008) 1–8
11. Padeleris, P., Zabulis, X., Argyros, A.A.: Head pose estimation on depth data based on particle swarm optimization. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). (2012) 42–49
12. Fanelli, G., Dantone, M., Gall, J., Fossati, A., Gool, L.V.: Random forests for real time 3d face analysis. Int'l Journal of Computer Vision (IJCV) **101** (2013) 437–458
13. Hug, Y., Chen, L., Zhoug, Y., Zhang, H.: Estimating face pose by facial asymmetry and geometry. In: FG '04. 6th IEEE International Conference on Automatic Face and Gesture Recognition. (2004) 651–656
14. Pathangay, V., Das, S., Greiner, T.: Symmetry-based face pose estimation from a single uncalibrated view. In: FG '08. 8th IEEE International Conference on Automatic Face and Gesture Recognition. (2008) 1–8

15. Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J.: Active shape models-their training and application. Computer Vision and Image Understanding (CVIU) **61** (1995) 38–59
16. Cootes, T.F., Edwards, G., Taylor, C.: Active appearance models. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI) **23** (2001) 681–685
17. Martins, P., Batista, J.: Accurate single view model-based head pose estimation. In: FG '08. 8th IEEE International Conference on Automatic Face and Gesture Recognition. (2008) 1–6
18. Morency, L.P., Whitehill, J., Movellan, J.: Monocular head pose estimation using generalized adaptive view-based appearance model. Image and Vision Computing **28** (2009) 754–761
19. Gourier, N., Hall, D., Crowley, J.L.: Estimating face orientation from robust detection of salient facial features. In: Proceedings of Pointing 2004, ICPR, International Workshop on Visual Observation of Deictic Gestures. (2004)
20. Lecun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1** (1989) 541–551
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). (2012)
22. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2013) 3626 –3633
23. Burger, H.C., Schuler, C.J., Harmeling, S.: Image denoising: Can plain neural networks compete with bm3d? In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2012)
24. Sun, Y., Wang, X., Tang, X.: Deep convolutional network cascade for facial point detection. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2013) 3476–3483
25. Zhou, E., Fan, H., Cao, Z., Jiang, Y., Yin, Q.: Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In: IEEE International Conference on Computer Vision Workshops (ICCVW). (2013) 386–391
26. Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proc. of Computer Vision and Pattern Recognition (CVPR). (2014)
27. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324
28. Coates, A., Lee, H., Ng, A.Y.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics (AISTATS). (2011) 215–233
29. Zhu, Z., Luo, P., Wang, X., Tang, X.: Recover canonical-view faces in the wild with deep neural networks. Computing Research Repository (CoRR), arXiv (2014)
30. Doucet, A., Freitas, N.D., Gorden, N.: Sequential monte carlo methods in practice. Springer (2001)
31. Gordon, N., Salmond, D., Smith, A.: Novel approach to nonlinear/nongaussian bayesian state estimation. IEE P. Radar Signal Processing **140** (1993) 107–113
32. Weise, T., Bouaziz, S., Li, H., Pauly, M.: Realtime performance-based facial animation. In: Proc. of SIGGRAPH. (2011)
33. Nuevo, J., Bergasa, L.M., Jiménez, P.: Rsmat: Robust simultaneous modeling and tracking. Pattern Recognition Letters **31** (2010) 2455–2463