Robust Driver Head Pose Estimation in Naturalistic Conditions from Point-Cloud Data

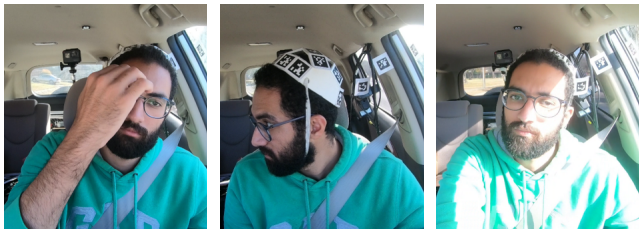
Tiancheng Hu, Sumit Jha and Carlos Busso

***Abstract*— Head pose estimation has been a key task in computer vision since a broad range of applications often requires accurate information about the orientation of the head. Achieving this goal with regular RGB cameras faces challenges in automotive applications due to occlusions, extreme head poses and sudden changes in illumination. Most of these challenges can be attenuated with algorithms relying on depth cameras. This paper proposes a novel point-cloud based deep learning approach to estimate the driver's head pose from depth camera data, addressing these challenges. The proposed algorithm is inspired by the PointNet++ framework, where points are sampled and grouped before extracting discriminative features. We demonstrate the effectiveness of our algorithm by evaluating our approach on a naturalistic driving database consisting of 22 drivers, where the benchmark for the orientation of the driver's head is obtained with the Fi-Cap device. The experimental evaluation demonstrates that our proposed approach relying on point-cloud data achieves predictions that are almost always more reliable than state-of-the-art head pose estimation methods based on regular cameras. Furthermore, our approach provides predictions even for extreme rotations, which is not the case for the baseline methods. To the best of our knowledge, this is the first study to propose head pose estimation using deep learning on point-cloud data.**

I. INTRODUCTION

Head pose estimation (HPE) is an important task in computer vision. It has a wide range of applications in areas such as advanced driver assistance system [1], visual attention modeling [2] and gaze estimation systems [3]–[6]. In automotive applications, which is the focus of this study, head pose estimation can be very valuable. Head pose estimation can be used to monitor driver's awareness [7], [8], assist interaction with in-car entertainment system [9], understand driver's intention and predict future driver actions [10]. Therefore, having a reliable head pose estimation system is crucial for advances in in-vehicle safety systems.

The topic of HPE has been extensively studied using regular RGB cameras. The solutions include nonlinear manifold learning [11], *random forest* (RF) [12], *support vector machine* (SVM) [13], and most recently, deep learning [9], [10], [14], [15]. However, progress on developing robust head pose estimation for automotive applications is limited [1], [10], [15], [16]. Existing research has shown promising results, but there is still room for improvement as existing HPE algorithms commonly suffer from high errors when there are occlusion of the driver, extreme head rotations, and sudden changes in illumination in the vehicle [17]. (Fig. 1). A time-of-flight camera based solution can, to a certain degree, alleviate some of the issues affecting head pose estimation in the vehicle. Because time-of-flight cameras use active infrared lighting to acquire images, they are immune to sudden illumination changes, which commonly occur in a moving vehicle. The most common way to represent depth camera recording is through depth maps, which project depth information into a single view [9], [10], [12], [15]. This approach ignores important 3D spatial information. Instead, we represent depth camera recording using point-clouds. A point-cloud contains 3D positions of each point relative to the center of the camera. Figure 2(a) shows an example of a point-cloud frame, describing the face of the driver.



(a) Occlusion (b) Extreme head pose (c) Illumination

Fig. 1: Examples of three types of challenges for HPE in a car. The frames are from recordings from the MVAD corpus.

Inspired by the success of point-cloud approaches such as PointNet++ [18], we present a deep learning based end-to-end regression algorithm to estimate the driver's head pose that directly operates on point-cloud data. We modified the set abstraction layer from PointNet++ as the backbone of our model. In each set abstraction layer, we have three components: sampling, grouping and PointNet. For sampling, we perform *iterative farthest point sampling* (IFPS) to choose anchor points from the input. For grouping, we use ball query to group neighboring points of each anchor point together. Then, we perform PointNet, which consists of a parameter shared *multilayer perceptron* (MLP) to capture local features, and a maxpooling operation after the MLP. We repeat the set abstraction layers five times, where each of them is implemented with three different scales. Finally, we generate a 6D vector as our prediction from linear combinations of the output from the last set abstraction layer from which we infer the orientation of the head. To the best of our knowledge, this is the first work on HPE using deep learning-based algorithm on point-cloud.

We use a multimodal database to demonstrate the ef-

*This work was supported by Semiconductor Research Corporation (SRC) / Texas Analog Center of Excellence (TxACE), under task 2810.014.

Tiancheng Hu, Sumit Jha and Carlos Busso are with the Department of Electrical Engineering, University of Texas at Dallas, Richardson, Texas, USA. {Tiancheng.Hu@utdallas.edu, Sumit.Jha@utdallas.edu, busso@utdallas.edu}

(a) point-cloud data (b) Grayscale image

Fig. 2: Examples of images from the depth camera. (a) point-cloud data from one frame, (b) grayscale image from the depth camera used as a reference for calibration (Sec. IV-B).

fectiveness of our algorithm. The model is trained and evaluated with data from 22 subjects, who participate in naturalistic driving recordings with ground truth for head pose information provided by the Fi-Cap device [19]. We compare this algorithm with two automatic HPE algorithms using regular RGB cameras: OpenFace 2.0 [20] and FAN [21]. The results show that using our point-cloud algorithm leads to lower *mean square errors* (MSE) over the baselines for most conditions. Likewise, we obtain predictions for a wider range of HPE compared to the baselines, which is another important benefit of our approach. These results demonstrate the benefits of our proposed solution.

II. RELATED WORK

A. General Head Pose Estimation

The computer vision community has extensively studied the problem of head pose estimation. Murphy-Chutorian and Trivedi [22] and Czuprynski and Strupczewski [23] surveyed studies on head pose estimation. Methods using appearance-based models have recently received increased attention due to the popularity of *convolutional neural network* (CNN). Liu *et al.* [24] proposed a regression model implemented with CNN trained with synthesized RGB images. They demonstrated significant improvement in HPE compared to previous works. Most existing works have focused on HPE from RGB images. Recently, commercial depth cameras have been increasingly accessible, which has led to a growing interest in HPE from depth images and point-clouds. Meyer *et al.* [25] localized 3D head in a depth image, fitting the head portion of the depth image onto a 3D morphable face model by combining the particle swarm optimization algorithm and the iterative closest point algorithm.

B. Driver Head Pose Studies

HPE is a difficult task inside vehicles using regular cameras. Jha and Busso [17] evaluated state-of-the-art HPE algorithms concluding that illumination, extreme head rotation and occlusions significantly impact the results. In fact, the selected algorithms were not able to provide an output in 8.9% to 21.8% of the frames.

Murphy-Chutorian *et al.* [1] used two *support vector regressors* (SVRs) to estimate head rotations (pitch and yaw). They detected the face with a cascaded-Adaboost face detector. They extracted the *localized gradient orientation* (LGO) histogram for each detected head, which were used as features to train the SVRs. LGO histograms were more robust to light conditions, as tested on a real vehicle. Bär *et al.* [26] designed an approach for head pose and gaze estimation intended for in-vehicle applications. The method uses multiple 3D templates that are aligned with point-cloud data using the *iterative closest point* (ICP) algorithm. To address the problem of temporal partial occlusions of the driver's face, they adopted multiple different face templates, taking the mean value of the different estimations. The approach was evaluated with laboratory recordings. Borghi et. al [9] utilizes a multi-modal approach based on several CNNs. They trained one CNN for head localization. They used three CNNs independently trained with either RGB images, depth data or optical flow information. The three CNNs were then fused to consolidate the HPE results. The method was evaluated with data from the POSEidon corpus, which was collected in a car simulator. Schwarz *et al.* [10] proposed a CNN based model that fuses information from infrared and depth images for HPE, evaluating their models on the DriveAHead corpus.

C. Processing Depth Data Using Deep Learning

The most common approach to represent data from depth cameras is to create 2D depth maps, where a value in the array provides its depth information at that specific x and y location. This representation has been used widely used in HPE [12], [25]. For example, studies have used CNN-based approaches to process depth maps for HPE [10], [15].

An alternative way to represent depth data is with point-cloud, which is a two dimensional array containing the x, y and z locations of each detected point in the coordinate system of the camera. For point-cloud classification tasks, Wu *et al.* [27] proposed a 3D CNN approach that represent 3D point-cloud as a binary probabilistic distribution on a 3D voxel grid. They used a 3D CNN to process the grid. This approach is not very effective because converting a 3D point-cloud into a 3D voxel grid inevitably leads to information loss. This method also requires a large model size and, therefore, it is harder to train, especially given that point-cloud datasets are usually much smaller than RGB image datasets. Another approach proposed by Su *et al.* [28] was to generate 2D image renderings of a 3D point-cloud, and used these renderings to train a CNN-based algorithm with view polling for 3D object classification. This approach gives more compelling results compared to the 3D voxel grid CNN approach. More recently studies include PointNet [29] and PointNet++ [18]. These two approaches directly process 3D point-cloud data without converting to any other intermediate representation, showing very competitive results on classification and segmentation tasks.

III. PROPOSED APPROACH

Inspired by the success of PointNet [29] and PointNet++ [18], we propose a model that utilizes the set abstraction layers used in PointNet++ as a feature extractor for HPE. The feature representation is the input of a fully connected layer with linear activation to estimate the rotation of the head.

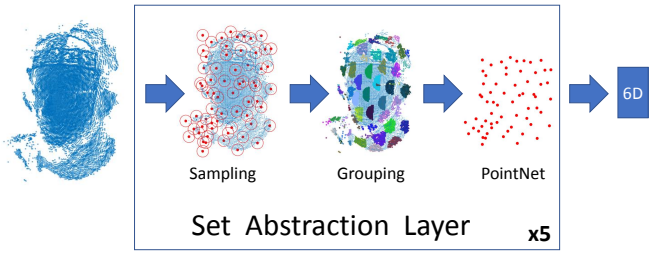


Fig. 3: Proposed point-cloud approach for HPE. The model creates a 6D vector from which we can estimate the head rotation of the driver.

Zhou *et al.* [30] show the benefits of representing orientation with a 6D vector extracted from the rotation matrix. We follow this approach by training our model to estimate this 6D vector. Notice that it is easy to convert the 6D vector into a full rotation matrix by using the Gram-Schmidt process.

Consider the point-cloud from frame t, consisting of n_t points. We create a $n_t \times 3$ vector $\mathbf{X_t}$. The proposed deep learning model creates a mapping $\mathbf{f}$ such that

$$f(\mathbf{X_t}) = \mathbf{y}, \qquad \mathbf{y} \in \mathbb{R}^6 \tag{1}$$

where $\mathbf{y}$ is the 6D vector representing the head orientation. This section explains our proposed approach.

A. Model Structure

Figure 3 shows a diagram of our proposed point-cloud model. Table I shows the implementation details of our model using five set abstraction layers, where the number of sampled points in each set is sequentially reduced.

The main blocks of the set abstraction layer are sampling, grouping, and PointNet. The first step is sampling points from the point-cloud data, where we reduce the level of redundancy in the raw data, and increase the computational efficiency of the algorithm. We use the *iterative farthest point sampling* (IFPS) algorithm, choosing a predefined number of anchor points. For example, the first set abstraction layer samples 512 points, denoted by **S1** in Table I.

The second step is grouping the anchor points. Similar to the way CNN works, the goal is to capture the relationship between the anchor points and their neighboring points. The algorithm uses ball query to group nearby points around an anchor point. The grouping is implemented at three different resolutions. For example, the first set abstraction layer simultaneously groups $l = 4$ points within $r = 0.1$ units of each anchor point (**G1**), $l = 8$ points within $r = 0.2$ units of each anchor point (**G2**), and $l = 16$ points within $r = 0.4$ units of each anchor point (**G3**).

The third step is PointNet [29], which aims to find a discriminative feature representation. This goal is achieved by transforming the points into a higher dimension, where we can achieve better discrimination. We can view these transformations as extra layers that capture the features of each anchor point as well as its neighboring areas. PointNet is implemented with three-layer *multilayer perceptrons* (MLPs) with shared weights. Every point is passed through the same set of filters. The MLP projects the point into a different feature representation. There are polling operations after each MLP transformation. For example, PointNet on the first set abstraction layer operates on the groupings **G1**, **G2** and **G3**. For the grouping **G1**, the three layers of the MLP have 8, 8 and 16 neurons, respectively. We use the maxpooling operation to create the 512×3 output (**T1**). For the grouping **G2**, the three layers of the MLP have 16, 16 and 32 neurons, respectively. After the maxpooling operation, we create a 512×3 output (**T2**). For the grouping **G3**, the three layers of the MLP have 16 , 24 and 32 neurons, respectively. We also obtain a 512×3 output (**T3**) after the maxpooling operation. PointNet provides a 512×9 output (**T4**) by concatenating **T1**, **T2**, and **T3**.

Our proposed point-cloud solution uses five set abstraction layers. The second set abstraction layer is exactly the same as the first set, except that the input is no longer the raw point-cloud data. Instead, we perform sampling from **S1**, and grouping from **T4** (e.g., sampling from the anchor points selected in the previous set, and grouping from the transformed anchor points from the previous set). From the second set onward, the PointNet layer will take as input the concatenation of the grouping step output and the anchor points. The third and fourth set abstraction layers are similar to the second set. The fifth set abstraction layer only has one anchor point. In the grouping layer, we group all points together by using an infinity radius. Then, we perform the usual PointNet layer, but with weighted average pooling at the end (**T5**).

PointNet++ was designed for classification and segmentation problems. Instead, we are using the set abstraction layer to obtain a discriminative feature representation for a regression problem. The final step is to connect **T5** to a fully connected layer with linear output units, creating a 6D vector.

IV. Multimodal Driver Monitoring Dataset

This study uses recording from the *Multimodal Driver Monitoring* (MDM) Dataset, which we are currently collecting at *The University of Texas at Dallas* (UT Dallas). The corpus consists of naturalistic driving recordings on the UTDrive vehicle [31]–[33], which is a 2006 Toyota RAV4 equipped with multiple sensors. This data collection is an ongoing effort, where the set used for this study contains recordings from 22 subjects (18 males, 4 females), with a total duration of 17 hours and 39 minutes. The subjects are mostly college students attending UT Dallas.

In this data collection, we use four GoPro HERO6 Black cameras installed to collect (1) frontal views of the driver's face, (2) semi-profile views of the driver's face as viewed from the rear mirror, (3) the back side of the driver's head to record the Fi-Cap device (see Sec. IV-B), and (4) the road. We set the frame rate to be 60 fps with a resolution of 1920×1080 pixels. Likewise, we use a CamBoard pico flexx for depth data recording. The camera is capable of capturing both depth data and grayscale images. Pico flexx supports a resolution of 224×171 and we set the frame rate to the maximum, which is 45 fps. The measuring range is between 0.1m and 4m which suits the requirement for in-vehicle recordings of a driver. We use a clapping board to

TABLE I: Model specifications for the proposed approach. m is the number of sample chosen as anchor points. $d_c = 3$ is the dimension of the input coordinate system (i.e., x, y and z). r represents the radius of the ball query for grouping operation. l represents the number of points within r to be chosen in the grouping step. d_f is the feature dimension of the previous set. In the first set, d_f=d_c=3.

Layer	Specification	Output Dimension
Sampling	m=512	[m=512,d_c=3] (**S1**)
Grouping	[r=0.1,l=4]	[m=512,d_f=3,l=4] (**G1**)
	[r=0.2,l=8]	[m=512,d_f=3,l=8] (**G2**)
	[r=0.4,l=16]	[m=512,d_f=3,l=16] (**G3**)
PointNet	[8,8,16]	[512,3] (**T1**)
	[16,16,32]	[512,3] (**T2**)
	[16,24,32]	[512,3] (**T3**)
		[512,9] (**T4**)
Sampling	m=256	[m=256,d_c=3]
Grouping	[r=0.15,l=8]	[m=256,d_f=12,l=8]
	[r=0.3,l=16]	[m=256,d_f=12,l=16]
	[r=0.5,l=24]	[m=256,d_f=12,l=24]
PointNet	[16,16,64]	[256,64]
	[32,32,64]	[256,64]
	[48,48,96]	[256,96]
		[256,224]
Sampling	m=128	[m=128,d_c=3]
Grouping	[r=0.15,l=8]	[m=128,d_f=227,l=8]
	[r=0.3,l=16]	[m=128,d_f=227,l=16]
	[r=0.5,l=24]	[m=128,d_f=227,l=24]
PointNet	[16,16,64]	[128,64]
	[32,32,64]	[128,64]
	[48,48,96]	[128,96]
		[128,224]
Sampling	m=64	[m=64,d_c=3]
Grouping	[r=0.2,l=16]	[m=64,d_f=227,l=16]
	[r=0.4,l=32]	[m=64,d_f=227,l=32]
	[r=0.8,l=48]	[m=64,d_f=227,l=48]
PointNet	[32,32,64]	[64,64]
	[48,48,64]	[64,64]
	[64,64,128]	[64,128]
		[64,256]
Sampling	m=1	[m=1,d_c=3]
Grouping	[r=inf]	[64,259]
PointNet	[256,256,384]	[384] (**T5**)
Fully Connected Layer	6	[6]

synchronize the two types of cameras at the beginning of each recording. We align intermediate frames based on the time elapsed from the clapping frame to the current frame.

A. Data Collection Protocol

We ask the driver to perform the following protocol:

- **Phase I**: Drivers are asked to look at markers inside and outside the car at different locations while the car is parked.
- **Phase II**: Drivers are asked to operate the vehicle following a predefined route. While driving, they are asked to identify landmarks on the route and cars satisfying certain conditions (e.g., red cars and specific car make). We also asked them to look at markers inside the vehicles.
- **Phase III**: Drivers are asked to conduct secondary tasks following the instructions given by the researcher leading the data collection. The tasks include changing the channel on a radio, and following GPS directions.

The subjects are told to only follow our instructions when they believe it is safe to conduct these tasks. They have the right to withdraw from the experiment at any time during the recording.

B. Ground Truth labels for HPE using Fi-Cap

A key feature of the corpus is that robust head pose labels can be easily assigned to each frame, with the use of the Fi-Cap device [19]. Fi-Cap is a 3D helmet with 23 AprilTags [34]. AprilTags are 2D fiducial markers with predefined patterns that can be robustly detected with regular cameras. It is easy to estimate the orientation and position of each AprilTag with regular cameras. The Fi-Cap is worn on the back side of the driver's head, without creating occlusions for cameras recording the frontal views of the face. The purpose of the camera behind the driver is to record as many AprilTags as possible to derive reliable labels for HPE. We use the Kabsch algorithm and the ICP algorithm to obtain the ground truth for the head pose from all the visible AprilTags (details are explained in Jha and Busso [19]). We have reported a detailed analysis on the performance of Fi-Cap, finding that the median angular error was only 2.30° [19].

Fi-Cap gives the head pose labels in terms of the Fi-Cap coordinate system. We need to calibrate the coordinate across subjects, because the participants are different, and the actual placement of the Fi-Cap varies across recordings. We manually select one grayscale frame from the Pico Flexx camera as a global reference (Fig. 2(b)). All the angles are expressed with respect to this reference. Then, we use OpenFace 2.0 [20] (Sec. V-B) on the grayscale recordings of each subject to find the frame for each subject that has the most similar head pose orientation as the reference frame. Finally, we correct the ground truth rotation of other frames from that subject by multiplying the inverse of the rotation matrix associated with the reference frame.

V. EXPERIMENTAL EVALUATION

A. Implementation

We use ADAM optimizer, with an initial learning rate of 0.001. We use a learning rate decay of rate 0.7 every 2 million steps. We use a batch size of 16. The model is implemented with a L2 loss function comparing the predicted and actual 6D vectors representing the head rotation. We use distance-based and statistical filters to remove points that are clearly part of the background. Then, we apply voxel grid downsampling to sample 5000 points from each of our point-clouds. Each point-cloud is normalized such that the centroid is at the origin, and all the points lie inside a unit sphere.

We split the corpus using participant-independent partitions, where data from 14 subjects are used for the train set, data from four subjects are used for the development set, and data from four subjects are used for the test set.

B. Baselines

The baselines in this study are two state-of-the-art HPE using regular cameras. While our approach processes data from the CamBoard pico flexx, the baselines are evaluated using the high-resolution frames from the GoPro HERO6 Black camera facing the driver.

The first approach is OpenFace 2.0 [20], which is a real-time toolkit that supports many facial behavioral analysis tasks, including HPE. It utilizes *convolutional experts constrained local model* (CE-CLM) [35] to detect facial landmark. The head rotation is obtained by using the 3D facial landmarks to solve the Perspective-n-Point Problem.

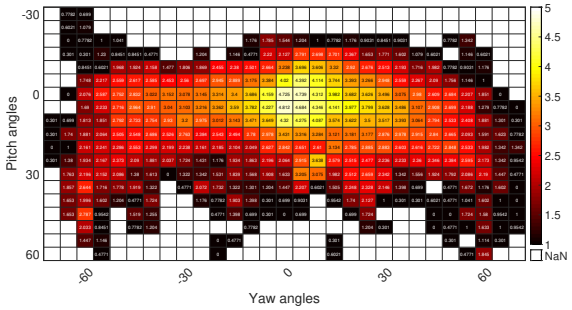


Fig. 4: The distribution of the frames in the test set as a function of yaw (horizontal) and pitch (vertical) angles. The figure is in logarithmic scale for better visualization. Bins with lighter the colors indicate more frames are found in the corpus. The bins with white color and no number indicate that no frames are available in the specific rotation range.

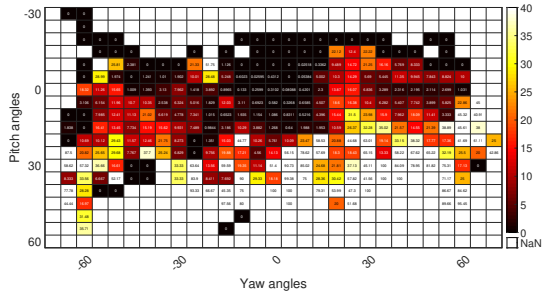
The second baseline is the *face alignment network* (FAN) [21], which is a state-of-the-art deep learning based facial landmark estimation algorithm. We use the RGB images from the GoPro camera that match the reference frames found in the calibration step for each of our subjects. Then, we extract 68 landmarks with FAN for each of these reference frames. We estimate the head pose of each frame by comparing the positions of its 68 facial landmarks with the corresponding landmarks from the reference frame. We use *singular value decomposition* (SVD) to calculate the rotation.

The predictions from OpenFace 2.0 and FAN are with respect to their own coordinate systems. We apply a subject-wise transformation so that the coordinate system matches the coordinate system of the pico fexx camera. For a given driver, we find transformations between the ground truth labels and the predictions of a baseline for all the frames. Then, we find the average transformation, which is used to compensate for the differences in the coordinate systems. The ground truth has been filtered to consider only frames with roll, yaw and pitch rotations below 45°, 80°and 70°, respectively. These values are considered following the head rotation limits of an average adult [36].

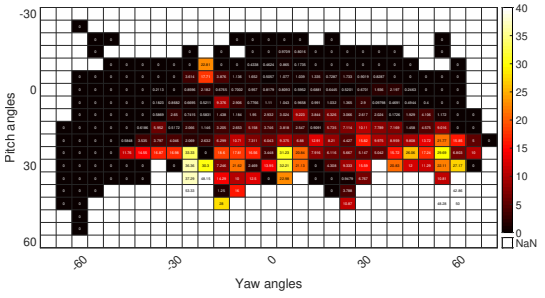
C. Results

Figure 4 shows the head pose histogram for the test set as a function of the yaw and pitch angles estimated with Fi-Cap. The number in each bin indicates the number of frames in that specific rotation range, where we use a logarithmic scale (base 10) for better visualization.The majority of frames have near frontal pose, which is expected in naturalistic driving. Although there is comparatively less number of frames with non-frontal poses, these frames are crucial when studying driver distraction or maneuvering behavior.

Most HPE methods based on regular RGB images tend to fail in challenging conditions in naturalistic driving scenarios [17]. In these cases, the face may not be even detected. A benefit of our proposed point-cloud approach is that we can provide an estimation for all the frames. This is not the case for our baselines methods. In fact, OpenFace 2.0 and FAN do not provide results for 4.85% and 2.10% of the frames, respectively. Figure 5 shows the percentage of



(a) Detection failure rate for OpenFace 2.0



(b) Detection failure rate for FAN

Fig. 5: Detection failure rate of OpenFace 2.0 and FAN as a function of yaw and pitch rotation angles. In contrast, the proposed approach provides estimation for all the frames. The detection failure increases in bins with lighter colors. The bins with white color and no number indicate that no frames are available in the specific rotation range.

frames in each bin for which the baseline methods failed to provide a prediction. The figure ignores bins with fewer than 10 frames. We observe that OpenFace 2.0 has a higher failure rate at large yaw and pitch angles. For some angle combinations, OpenFace 2.0 fails for all frames. While FAN has a lower detection failure rate overall, it also suffers in extreme head poses. Instead, our approach takes the raw data and directly predicts HPE, obtaining an estimation for all the frames. This approach is practical in current naturalistic driving scenarios where the presence of a person in the driver seat is expected when the car is moving.

Table II shows the *mean square error* (MSE) for our model and the baselines. The first row of the table provides the results for our proposed approach on all the frames in the test set. The MSEs for all the angular rotations are between 5°and 8°. These values show we can construct a robust HPE model using point-cloud data, which is particularly useful in challenging environments such as inside a vehicle.

We also compare the MSE of our method with OpenFace2.0 and FAN. Since these algorithms do not provide predictions for all the frames, we define two subsets for fair comparisons with our method. We define the *OpenFace set* as the subset where OpenFace 2.0 provides a prediction. Likewise, we define the *FAN set* as the subset where FAN provides an estimate. Table II shows that our method provides competitive result compared to both baselines. In the *OpenFace set*, our approach reaches lower MSE for roll and pitch rotations. The proposed method is slightly worse for yaw rotations. It is important to highlight that OpenFace

TABLE II: MSE of the proposed method using point-cloud data. The table also compares the HPE results with the results from OpenFace 2.0 and FAN using subsets of frames for which the baselines provided a prediction.

Method	Errors		
	Roll(°)	Yaw (°)	Pitch (°)
Proposed Point-Cloud Solution	5.91	7.32	6.68
	OpenFace Set		
OpenFace 2.0	9.32	**6.21**	8.42
Proposed Point-Cloud Solution	**5.48**	7.15	**6.39**
	FAN Set		
FAN	11.30	19.28	8.47
Proposed Point-Cloud Solution	**5.66**	**7.24**	**6.53**

2.0 does not provide estimation for extreme rotations, so the benefits of using our approach is not only better performance for most of the angular rotations, but also predictions for all the frames. In the *FAN set*, Table II clearly demonstrates the superior performance of our algorithm over the FAN algorithm. Our model shows much lower MSE in all the three rotation axes.

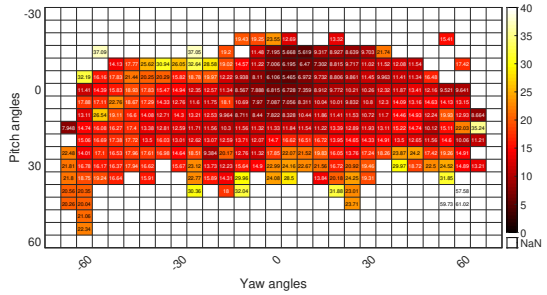
To understand better the performance of our proposed point-cloud approach and the baselines, we show the performance of each of these methods as a function of yaw (horizontal) and pitch (vertical) angles. Figure 6 shows the errors compared to the ground truth head pose labels obtained with Fi-Cap. The error is represented by the geodesic distance in the rotation matrix space [37].

$$\Delta(R_1, R_2) = \| \log(R_1 R_2^T) \| \tag{2}$$

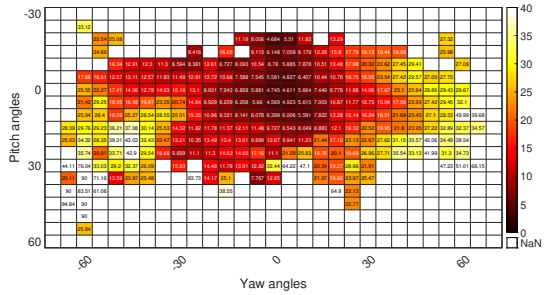
The color of the bin indicates the error, where darker colors correspond to smaller errors. We ignore bins with fewer than ten frames in this analysis. Bins that are completely white indicate that there are not enough frames for these bins. Note that for our proposed method, the histogram includes frames where OpenFace 2.0 and FAN do not have predictions. Our method provides far superior performance than the baselines in all frames, especially for frames with large rotations. While the baselines provide reasonably good performance for small rotations, the error grows rapidly as the ground truth rotation increases. This particularly clear with FAN. OpenFace 2.0 provides very good rotation estimation when the overall rotation is small, but its performance drops significantly as the rotation increases. In contrast, the error in our approach grows much slower as the ground truth rotation increases, allowing a wider range of rotations. Overall, the proposed point-cloud framework provides far superior performance in frames with large rotations while maintaining competitive performances at small rotations.

VI. CONCLUSIONS

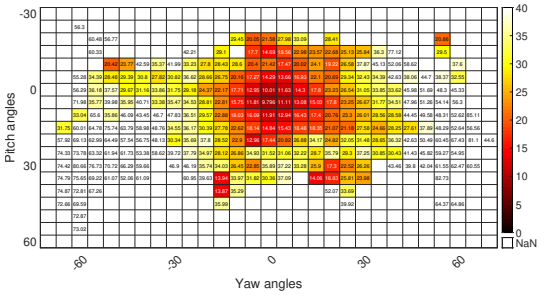
This paper presented a novel point-cloud based algorithm to estimate the head pose of a driver from a single depth camera. The proposed algorithms achieved promising performances, outperforming the state-of-the-art baselines OpenFace2.0 and FAN, which rely on regular RGB images. The improvement in performance is especially clear on frames with large rotations. For automotive applications such as detecting driver distraction [7], [38], it is important to have accurate HPE on non-frontal poses. Our method is far more



(a) Proposed point-cloud approach



(b) OpenFace 2.0



(c) FAN

Fig. 6: Performance of HPE algorithm as a function of yaw (horizontal) and pitch (vertical) angles. The graph shows the average geodesic distance between the true and predicted angles per bin. Bins with darker color indicates lower errors. The bins with white color and no number indicate that no frames are available in the specific rotation range.

suitable for this purpose than the baselines. Furthermore, the proposed algorithm can also work in cases with very low illumination or even during the night. In these scenarios, RGB based approaches cannot be used.

In the future, we would like to add temporal modeling to our framework to leverage the strong correlation between consecutive head poses. Our multimodal data is suitable for this task, since we have head rotation labels for all the frames in the corpus. We are also working to increase the size of the corpus by recording new subjects. We expect more robust and accurate models by increasing our training set, which will enable our model to better capture more facial variability across drivers.

REFERENCES

[1] E. Murphy-Chutorian, A. Doshi, and M. Trivedi, "Head pose estimation for driver assistance systems: A robust algorithm and experimental evaluation," in *IEEE Intelligent Transportation Systems Conference (ITSC 2007)*, Seattle, WA, USA, September-October 2007, pp. 709–714.

[2] S. Ba and J.-M. Odobez, "Recognizing visual focus of attention from head pose in natural meetings," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 1, pp. 16–33, February 2009.

[3] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, Boston, MA, USA, June 2015, pp. 4511–4520.

[4] S. Jha and C. Busso, "Analyzing the relationship between head pose and gaze to model driver visual attention," in *IEEE International Conference on Intelligent Transportation Systems (ITSC 2016)*, Rio de Janeiro, Brazil, November 2016, pp. 2157–2162.

[5] ——, "Probabilistic estimation of the driver's gaze from head orientation and position," in *IEEE International Conference on Intelligent Transportation (ITSC)*, Yokohama, Japan, October 2017, pp. 1630–1635.

[6] ——, "Probabilistic estimation of the gaze region of the driver using dense classification," in *IEEE International Conference on Intelligent Transportation (ITSC 2018)*, Maui, HI, USA, November 2018, pp. 697–702.

[7] N. Li and C. Busso, "Predicting perceived visual and cognitive distractions of drivers with multimodal features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 51–65, February 2015.

[8] ——, "Detecting drivers' mirror-checking actions and its application to maneuver and secondary task recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 980–992, April 2016.

[9] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara, "POSEidon: Face-from-depth for driver pose estimation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, Honolulu, HI, USA, July 2017, pp. 5494–5503.

[10] A. Schwarz, M. Haurilet, M. Martinez, and R. Stiefelhagen, "DriveAHead - a large-scale driver head pose dataset," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW 2017)*, Honolulu, HI, USA, July 2017, pp. 1165–1174.

[11] B. Raytchev, I. Yoda, and K. Sakaue, "Head pose estimation by nonlinear manifold learning," in *IEEE International Conference on Pattern Recognition (ICPR 2004)*, vol. 4, Cambridge, UK, August 2004, pp. 462–466.

[12] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011)*, Providence, RI, USA, June 2011, pp. 617–624.

[13] Y. Yan, E. Ricci, R. Subramanian, O. Lanz, and N. Sebe, "No matter where you are: Flexible graph-guided multi-task learning for multi-view head pose classification under target motion," in *IEEE International Conference on Computer Vision (ICCV 2013)*, Sydney, NSW, Australia, December 2013, pp. 1177–1184.

[14] S. S. Mukherjee and N. M. Robertson, "Deep head pose: Gaze-direction estimation in multimodal video," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2094–2107, November 2015.

[15] M. Venturelli, G. Borghi, R. Vezzani, and R. Cucchiara, "Deep head pose estimation from depth data for in-car automotive applications," in *International Workshop on Understanding Human Activities through 3D Sensors (UHA3DS 2016)*, ser. Lecture Notes in Computer Science, H. Wannous, P. Pala, M. Daoudi, and F. Flórez-Revuelta, Eds. Cancun, Mexico: Springer Berlin Heidelberg, December 2018, vol. 10188, pp. 74–85.

[16] Q. Ji and X. Yang, "Real-time eye, gaze, and face pose tracking for monitoring driver vigilance," *Real-Time Imaging*, vol. 8, no. 5, pp. 357–377, October 2002.

[17] S. Jha and C. Busso, "Challenges in head pose estimation of drivers in naturalistic recordings using existing tools," in *IEEE International Conference on Intelligent Transportation (ITSC)*, Yokohama, Japan, October 2017, pp. 1624–1629.

[18] C. Qi, L. Yi, H. Su, and L. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *In Advances in Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, December 2017, pp. 5099–5108.

[19] S. Jha and C. Busso, "Fi-Cap: Robust framework to benchmark head pose estimation in challenging environments," in *IEEE International Conference on Multimedia and Expo (ICME 2018)*, San Diego, CA, USA, July 2018, pp. 1–6.

[20] T. Baltrušaitis, A. Zadeh, Y. C. Lim, and L. Morency, "OpenFace 2.0: Facial behavior analysis toolkit," in *IEEE Conference on Automatic Face and Gesture Recognition (FG 2018)*, Xi'an, China, May 2018, pp. 59–66.

[21] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2D & 3D face alignment problem? (and a dataset of 230,000 3D facial landmarks)," in *IEEE International Conference on Computer Vision (ICCV 2017)*, Venice, Italy, October 2017, pp. 1021–1030.

[22] E. Murphy-Chutorian and M. M. Trivedi, "Head pose estimation in computer vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 607–626, April 2009.

[23] B. Czupryński and A. Strupczewski, "High accuracy head pose tracking survey," in *International Conference on Active Media Technology (AMT 2014)*, ser. Lecture Notes in Computer Science, D. Ślezak, G. Schaefer, S. Vuong, and Y. Kim, Eds. Warsaw, Poland: Springer Berlin Heidelberg, August 2014, vol. 8610, pp. 407–420.

[24] X. Liu, W. Liang, Y. Wang, S. Li, and M. Pei, "3D head pose estimation with convolutional neural network trained on synthetic images," in *IEEE International Conference on Image Processing (ICIP 2016)*, Phoenix, AZ, USA, September 2016, pp. 1289–1293.

[25] G. P. Meyer, S. Gupta, I. Frosio, D. Reddy, and J. Kautz, "Robust model-based 3D head pose estimation," in *IEEE International Conference on Computer Vision (ICCV 2015)*, Santiago, Chile, December 2015, pp. 3649–3657.

[26] T. Bär, J. Reuter, and J. Zöllner, "Driver head pose and gaze estimation based on multi-template ICP 3-D point cloud alignment," in *International IEEE Conference on Intelligent Transportation Systems (ITSC 2012)*, Anchorage, AK, USA, September 2012, pp. 1797–1802.

[27] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, Boston, MA, June 2015, pp. 1912–1920.

[28] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *IEEE International Conference on Computer Vision (ICCV 2015)*, Santiago, Chile, December 2015, pp. 945–953.

[29] C. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, Honolulu, HI, USA, July 2017, pp. 77–85.

[30] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, June 2019, pp. 5738–5746.

[31] P. Angkititrakul, M. Petracca, A. Sathyanarayana, and J. Hansen, "UTDrive: Driver behavior and speech interactive systems for in-vehicle environments," in *IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, June 2007, pp. 566–569.

[32] P. Angkititrakul, D. Kwak, S. Choi, J. Kim, A. Phucphan, A. Sathyanarayana, and J. Hansen, "Getting start with UTDrive: Driver-behavior modeling and assessment of distraction for in-vehicle speech systems," in *Interspeech 2007*, Antwerp, Belgium, August 2007, pp. 1334–1337.

[33] J. Hansen, C. Busso, Y. Zheng, and A. Sathyanarayana, "Driver modeling for detection and assessment of driver distraction: Examples from the UTDrive test bed," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 130–142, July 2017.

[34] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *IEEE International Conference on Robotics and Automation (ICRA 2011)*, Shanghai, China, May 2011, pp. 3400–3407.

[35] T. B. A. Zadeh, Y. C. Lim and L. Morency, "Convolutional experts constrained local model for 3D facial landmark detection," in *IEEE International Conference on Computer Vision Workshops (ICCVW 2017)*, Venice, Italy, October 2017, pp. 2519–2528.

[36] V. Ferrario, C. Sforza, G. Serrao, G. Grassi, and E. Mossi, "Active range of motion of the head and cervical spine: a three?dimensional investigation in healthy young adults," *Journal of Orthopaedic Research*, vol. 20, no. 1, pp. 122–129, January 2002.

[37] D. Huynh, "Metrics for 3D rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, June 2009.

[38] J. Jain and C. Busso, "Analysis of driver behaviors during common tasks using frontal video camera and CAN-Bus information," in *IEEE International Conference on Multimedia and Expo (ICME 2011)*, Barcelona, Spain, July 2011.