Temporal Head Pose Estimation From Point Cloud in Naturalistic Driving Conditions

Tiancheng Hu, *Student Member, IEEE*, Sumit Jha, *Graduate Student Member, IEEE*, and Carlos Busso, *Senior Member, IEEE*

***Abstract*— Head pose estimation is an important problem as it facilitates tasks such as gaze estimation and attention modeling. In the automotive context, head pose provides crucial information about the driver's mental state, including drowsiness, distraction and attention. It can also be used for interaction with in-vehicle infotainment systems. While computer vision algorithms using RGB cameras are reliable in controlled environments, head pose estimation is a challenging problem in the car due to sudden illumination changes, occlusions and large head rotations that are common in a vehicle. These issues can be partially alleviated by using depth cameras. Head rotation trajectories are continuous with important temporal dependencies. Our study leverages this observation, proposing a novel temporal deep learning model for head pose estimation from point cloud. The approach extracts discriminative feature representation directly from point cloud data, leveraging the 3D spatial structure of the face. The frame-based representations are then combined with *bidirectional long short term memory* (BLSTM) layers. We train this model on the newly collected *multimodal driver monitoring* (MDM) dataset, achieving better results compared to non-temporal algorithms using point cloud data, and state-of-the-art models using RGB images. We further show quantitatively and qualitatively that incorporating temporal information provides large improvements not only in accuracy, but also in the smoothness of the predictions.**

***Index Terms*— Driver head pose estimation, deep learning, point cloud, temporal modeling.**

I. Introduction

THE era of fully autonomous vehicles is unquestionably approaching. However, according to the National Highway Traffic Safety Administration [1], the road map for developing autonomous vehicle technology before 2025 includes only "partially automated safety features" and that "[automated vehicles] are a future technology rather than one that you'll find in a dealership tomorrow or in the next few years." Many safety issues facing autonomous vehicles remain unresolved [2]. These statements emphasize the importance of developing *advanced driver assistance system* (ADAS) while the technologies for fully autonomous vehicles are still being developed. A key component of ADAS is to estimate the head pose of the driver. The driver head pose is useful for tasks such as driver attention and distraction detection [3]–[7], interaction with in-vehicle infotainment system [8] and driver drowsiness detection [9]. Even after the technology for fully automated vehicle matures, there is still a need for head pose estimation algorithms to monitor driver attention during take-over decisions, and to clarify commands using multimodal navigation or infotainment systems (e.g., "what is the address of that store?", while a passenger glances at a given building).

While research in head pose estimation using RGB data has been an active research problem for a long time, estimating head pose using depth data is relatively recent. The solutions have relied on several algorithms including *random forest* (RF) [10], *particle swarm optimization* (PSO) [11], *iterative closest point* (ICP) [12], a combination of the PSO and ICP algorithms [13], 3D *histogram of oriented gradients* (HOG) [14], *support vector machine* (SVM) [15], *triangular surface patch* (TSP) features [16], and deep learning solutions [17]–[20]. While these approaches have reliable performance in constrained domains, the challenging conditions inside a vehicle limit the use of these approaches [21]. The driver head pose is not necessarily frontal and may change quickly, where non-frontal faces are often associated with segments of interest (e.g., eye-off-the-road events). Other challenging conditions in the vehicle include frequent changes in illumination, affecting the quality of the data, and common facial occlusions.

A key observation inspiring our work is that head pose trajectories are continuous, where previous frames provide valuable information to assess the head pose of a new frame. Incorporating temporal models can attenuate some of the challenges in head pose estimation inside a vehicle. Studies have considered head pose estimation from depth data by exploring temporal modeling. These approaches usually involve tracking with algorithms such as ICP with a face model [22]–[27]. These approaches usually use the estimation of one or a limited number of previous frames as the starting points to estimate the head pose on a new frame, without using the inherent motion information present in the data [22], [23], [25], [26]. Moreover, the target applications of these approaches are usually cases where the user performs confined head rotation in a controlled laboratory setting [22], [23], [25]–[27]. These approaches also often rely on precise facial landmark detection [22]–[24], [26], [27] and require RGBD data [22]–[24], [26], [27]. Some of these methods require complex and hard-to-construct

Manuscript received August 25, 2020; revised March 3, 2021; accepted April 17, 2021. This work was supported by Semiconductor Research Corporation (SRC)/Texas Analog Center of Excellence (TxACE) under Grant 2810.014. The Associate Editor for this article was K. Yi. *(Corresponding author: Carlos Busso.)*

The authors are with the Department of Electrical and Computer Engineering, The University of Texas at Dallas, Dallas, TX 75080 USA (e-mail: tiancheng.hu@utdallas.edu; sumit.jha@utdallas.edu; busso@utdallas.edu).

Digital Object Identifier 10.1109/TITS.2021.3075350

face models as prior information [22], [24]–[27]. There are many studies showing success in temporal modeling in videos using deep learning in tasks such as video action recognition [28], [29], video summarization [30], video emotion recognition [31], [32], video object segmentation [33], point cloud refinement [34], and future frame prediction [35]. Deep learning models can leverage longer temporal dependencies, and benefit from increasing the size of the corpus, without requiring handcrafted features or face models.

This study proposes a temporal deep learning head pose estimation algorithm from point cloud designed specifically to track the orientation of the driver's head while he/she operates a vehicle. The approach has two main components: feature extraction and temporal modeling. The first component is the feature extraction module, which builds on the framework proposed in our preliminary work [20]. It extracts a feature representation directly from the point cloud data. The approach relies on a modified version of the set abstraction layer used in the PointNet++ framework [36]. The feature extractor has three main building blocks: sampling, grouping and PointNet. In sampling, we perform *iterative farthest point sampling* (IFPS) to get anchor points from the input, which will be used in the grouping step. The points are selected to reduce the dimensionality of the data while keeping their structure. In grouping, we use the ball query algorithm, in which neighboring points within a distance of each anchor point are grouped together. In PointNet, we use a parameter-shared *multilayer perceptron* (MLP) to capture local features from each point. The PointNet layer also has a max pooling operation that selects the most representative feature for each neighborhood. The sequence of sampling, grouping and PointNet is repeated five times to form deeper feature representations to predict the head pose. The second component is the temporal modeling module, which leverages the dependencies across frames, improving the accuracy and smoothness of our predictions. After extracting the feature representation of each frame, we model temporal information using *bidirectional long short term memory* (BLSTM) layers to explicitly capture the movement of the head. This approach differs from previous studies that merely use a limited number of previous frames as initialization of tracking based methods [22], [23], [25], [26]. The BLSTM layers can extract longer temporal information from previous and future frames. The contextual window is around 2.84 seconds in our implementation. While this study targets automotive applications, our proposed approach can also be applied in other domains that require head pose estimation in challenging environments such as *human-robot interaction* (HRI) in crowded spaces.

We evaluate our approach on the *multimodal driver monitoring* (MDM) database [37], which is a naturalistic driving corpus. The corpus was specifically collected to directly assign head pose labels to each frame by using the Fi-Cap device [38] (i.e., a helmet with 2D fiducial markers worn on the back of the head to avoid occlusions with frontal cameras). As a result, we have millions of frames that can be used to train our proposed models. We achieve angular errors of 5.63° or less in all three rotation axis. We show large improvements in prediction accuracy by using this model compared to the state-of-the-art RGB head pose estimation algorithms OpenFace 2.0 [39], Hopenet [40], and ZFace [41]. The improvements over these baselines are especially large in the challenging frames where the ground truth rotation angles are large, showing the benefits of estimating head pose using point cloud data. We also demonstrate the importance of modeling temporal information by comparing our proposed approach with a static framework that only has the feature extraction component. The results are consistently better by using the temporal modeling component. Furthermore, the experimental evaluations qualitatively and quantitatively show the additional benefit of improved smoothness in the predicted trajectories by using the proposed temporal model compared to the static counterpart. We also evaluate the proposed approach on the BIWI database, achieving better results than alternative approaches. The contributions of this paper are the following:

- To the best of our knowledge, we propose the first deep learning temporal head pose estimation algorithm from point cloud data, targeting automotive applications.
- The approach directly extracts discriminative head pose information from the point cloud data without handcrafted features, leveraging temporal information across frames.
- We build and evaluate the proposed approach using the MDM database, which was specifically collected to derive annotations of head pose for each frame using the Fi-Cap helmet, allowing us to train our deep learning models.

The rest of the paper is organized as follows. Section II reviews the related studies on head pose estimation from depth cameras, with and without temporal modeling. Section III describes the proposed model. Section IV describes the MDM database used for building and evaluating the models, and the inter-subject calibration algorithm used to estimate the head pose results. Section V describes the implementation of our proposed approach. Section VI presents the experimental results of the proposed model compared with various RGB and depth based baselines. Section VII concludes the paper, summarizing the contributions of this study and describing future research directions.

II. Related Work

A. Head Pose Estimation From Depth Camera

While there have been important advances in head pose estimation from RGB data [42], most studies in this area using depth cameras are relatively recent, as a consequence of the developments of depth cameras. These studies can be classified into the following two categories: template-matching based and feature based approaches. A typical template-matching based approach would require a face model that is either acquired online or offline. Then, algorithms such as ICP or PSO are used to find the transformation between the template and each frame to derive its head pose. Bär *et al.* [12] proposed a multi-template ICP approach for head pose estimation of a driver. In this work, they noted that depth cameras mounted on the dashboard of a car may only capture the driver's face partially in non-frontal poses. Therefore, they used three different face templates, performing three ICP operations per frame. Padeleris *et al.* [11] proposed a PSO based algorithm. They first collected a reference image for each user. During run time, they used the PSO algorithm to search a pose in the 6-dimensional pose space such that the frame after the transformation is the most similar to the reference. Meyer *et al.* [13] proposed an algorithm that registers a morphable face model for the depth data in question, using a hybrid framework that combines the PSO and ICP algorithms. While their method does not require a subject-specific face model, they initialized the morphable model with a meticulously created 3D Basel Face Model [43]. A somewhat different approach is proposed by Jiménez *et al.* [44]. They proposed a head pose estimation method based on a stereo camera. They construct a 3D face model with stereo images, estimating head pose by solving a nonlinear least squares problem.

The second type of approaches for head pose estimation using depth cameras is the feature-based approach, which takes the depth data, usually represented as depth maps, and extracts features at different scales. The features are then used to estimate the head pose trajectories. Fanelli *et al.* [10] proposed an algorithm using RF regression. They assumed that face detection is already done. They use random patches on the depth maps as the input. The random forest model maps the input patches to the output head pose. Saeed and Al-Hamadi [15] first performed face detection. Then, they extracted HOG features from both the RGB images and the depth maps. These features are used to train a SVM to predict the head pose. There are many recent studies utilizing deep learning for head pose estimation. For example, Venturelli *et al.* [19] and Schwarz *et al.* [17] treated depth images as RGB images, extracting feature representations with *convolution neural networks* (CNNs).

Studies have shown the benefits of extracting discriminative features directly from point cloud data [36], [45]–[47]. To the best of our knowledge, the only study using this approach for head pose estimation is our preliminary study [20], which is the building block of our proposed approach in this paper.

B. Temporal Head Pose Estimation From Depth

Head pose trajectories are continuous and smooth, which has motivated the research community to explore temporal modeling for head pose estimation using depth images. Several of these studies are simple extensions of approaches presented in Section II-B by adding a tracking component. Li *et al.* [26] used a person-specific face model to estimate the initial head pose. The next frames are established by tracking consecutive frames. They predicted the head pose by performing ICP on the current and previous frames after subtracting head motion with a Kalman filter based motion predictor. Sheng *et al.* [25] proposed a statistical 3D face model with uncertainty estimation using the *ray visibility constrain* (RVC) algorithm for tracking. Compared to ICP, RVC can identify overlapping parts of a head that are visible in both frames, making the model less vulnerable to local minima. Yu *et al.* [24] proposed to fit a *3D morphable model* (3DMM) online, reconstructing a 3D full head model. For tracking, they used the *Kanade–Lucas–Tomasi* (KLT) feature tracker on the projected 2D images. They argued that explicitly exploiting visual motion is better than simply using the estimated head pose in frame $i-1$ as initialization for frame i. Borghi *et al.* [48] proposed a deep-learning based model with CNN as a feature extractor. They used depth images, treating them as normal RGB images. The temporal modeling is captured with *long short-term memory* (LSTM) [49] from high-level features. While they claimed that this model is designed for driver head pose estimation, they only tested their model on datasets recorded in controlled laboratory settings.

C. Temporal Point Cloud Processing With Deep Learning

There are very few studies exploring temporal deep learning modeling for point cloud, which motivates this study. Liu *et al.* [50] proposed a temporal point cloud processing framework with deep learning for classification, segmentation and scene flow estimation (i.e., non-automobile applications). The approach was based on PointNet++ [36]. In this work, Liu *et al.* [50] treated a sequence of point cloud as one big point cloud. They efficiently constructed spatiotemporal neighborhoods at different scales to process point cloud sequences. Since this approach collects information from multiple point clouds in time to give one output, it is not effective in tracking temporal movement. Therefore, in our work, we take a different approach by separately processing individual point cloud frames, which are later combined by performing temporal feature refinement on higher level features.

III. Proposed Method

We propose an algorithm that directly processes temporal point cloud sequences, predicting head poses for the entire sequence. Our proposed approach for head pose estimation automatically extracts discriminative features directly from point cloud, modeling temporal information with recurrent models that have access to future and previous features. We make the assumption that the driver's head is present in every frame. Therefore, we do not use any head detection algorithm.

We consider a point cloud sequence $\mathbf{X}$ of length N. For a frame t, the point cloud data is denoted by $\mathbf{X_t}$, which consists of the 3D coordinates of n_t points (n_t,3). The model finds a mapping f such that

$$f(\mathbf{X}, t) = \mathbf{y_t}, \qquad \mathbf{y_t} \in \mathbb{R}^6 \tag{1}$$

where $\mathbf{y_t}$ is a vector representing the head pose estimation for frame t, which depends on the values of an N-length point cloud sequence ($\mathbf{X}$). In this study, we use a 6D vector to represent the head rotation, as suggested by Zhou *et al.* [51]. This 6D vector is the first two columns extracted from the full rotation matrix and can be easily converted back to a full rotation matrix via the Gram-Schmidt process. In our implementation, we downsample the point cloud data to obtain a fixed number of points per frame (i.e., n_t is a constant).

In our preliminary work, we evaluated head pose estimation by directly processing point cloud data, instead of treating point cloud data as images [20]. Our study builds upon this preliminary work, leveraging temporal information to estimate

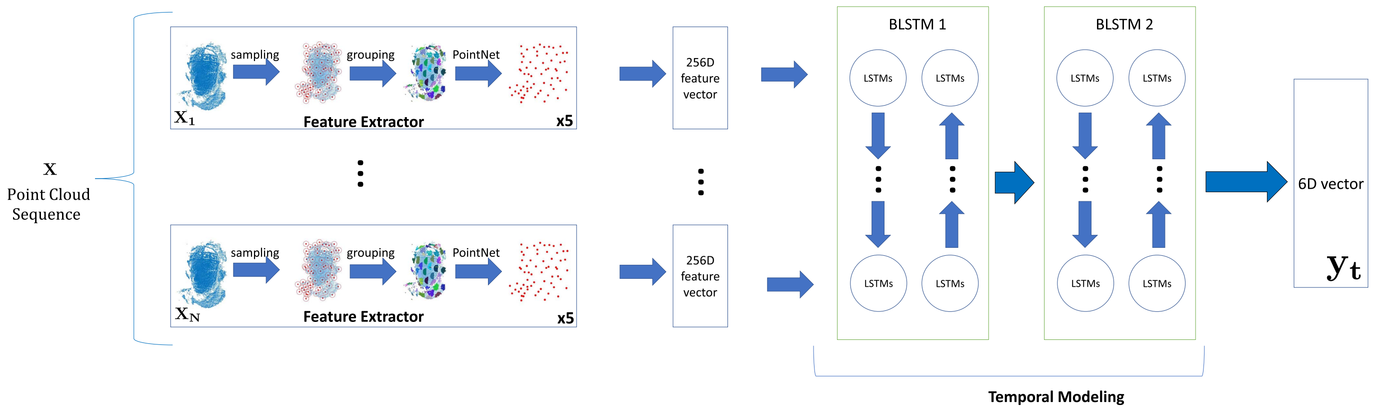


Fig. 1. Proposed temporal model for head pose estimation from point cloud data. The feature extraction module extracts feature representation for each frame, which are combined by the temporal modeling module.

head pose trajectories. Figure 1 illustrates our model, which has two main parts, namely feature extraction and temporal modeling. This section describes our proposed solution for head pose estimation.

A. Feature Extraction

The feature extraction approach is inspired by the PointNet [45], and PointNet++ [36] frameworks. Both of these approaches proposed models that target general point cloud classification and segmentation tasks (i.e., non-automobile applications). The method directly operates on point clouds, without projecting the 3D points into 2D images. Therefore, they preserve the spatial resolution of the point cloud data. In particular, we use the set abstraction layer in the PointNet++ framework.

The feature extractor has three building blocks: sampling, grouping and PointNet (left side of Fig. 1). The sampling block reduces the dimension of the point cloud data. We downsample the input point cloud data using the *iterative farthest point sampling* (IFPS) algorithm, choosing a preset number of the most representative points from the entire point cloud. $\mathbf{X_t} = \{P_1, P_2, \ldots, P_{n_t}\}$ represents the set of n_t points in the input point cloud. The goal is to obtain a reduced yet representative set with m points that describes the data, where $n_t \gg m$. First, the IFPS algorithm selects a point at random, P_{j_1}. Then, it searches for P_{j_2}, the point that is farthest away from P_{j_1}. The algorithm continues iteratively selecting points that are farthest away from the selected set of points until reaching the desired number of points m. The process selects $\mathbf{X_t^{\mathit{Anchor}}} = \{P_{j_1}, P_{j_2}, \ldots, P_{j_m}\}$. We refer to these selected points as *anchor points*. A benefit of this step is the reduction of the computational cost of the algorithm, while maintaining useful information about the structure of the point cloud data.

The grouping block aims to aggregate neighboring points to the anchor points by selecting samples within a certain radius centered at the anchor points. This step uses the set $\mathbf{X_t}$, which includes all the points in the point cloud. The size of this set is $(n_t, 3)$. For each anchor, we select l points that we want to associate with the anchor. The algorithm looks for points where the distance to the anchor is less than the radius. The search finishes when l points are found. This step defines the spatial region around anchor points, over which we extract features. The outputs of the grouping block is $\mathbf{X_t^{\mathit{Group}}}$, which consists of m sets of l points in a 3 dimensional space $(m, l, 3)$.

The PointNet block takes the points in each spatial neighborhood as the input of a weight-shared *multi-layer perceptron* (MLP) network. Mathematically, the PointNet block can be written as a mapping g,

$$g(\mathbf{X_t^{\mathit{Group}}}) = \mathbf{POOL}(h(\mathbf{X_t^{\mathit{Group}}})) \quad (2)$$

$$h : \mathbb{R}^{m\times l\times 3} \mapsto \mathbb{R}^{m\times l\times K} \quad (3)$$

$$POOL : \mathbb{R}^{m\times l\times K} \mapsto \mathbb{R}^{m\times K} \quad (4)$$

where K is the desired output feature dimension, h is a set of MLP functions, and $POOL$ is a pooling function applied to each group. This network transforms the points in the selected region into a feature representation that is trained to be discriminative for head pose estimation. There is a pooling operation after the MLP layers, which combines the feature from all the points in each spatial neighborhood such that there is one representation per spatial neighborhood. One may use multiple radii in this step to capture spatial neighborhoods of different size, achieving a better representation of the input.

The sequence of sampling, grouping and PointNet can be repeated multiple times to obtain high level features. During grouping, the feature maps of the previous PointNet layer, with dimension d, are concatenated with the representation. Therefore, the dimensions of the input and output of the grouping block are $(n_t, 3+d)$ and $(m, l, 3+d)$, respectively. We implement the sampling-grouping-PointNet blocks five times. The last block groups and unifies the information across the anchor points. The details of the implementation are given in Section V. The output vector from the feature extractor has a dimension 256D. Notice that the original PointNet and PointNet++ frameworks were designed for classification and segmentation problems. In contrast, we use this formulation to extract features from a point cloud for a regression problem.

B. Temporal Modeling

We hypothesize that capturing temporal information is important for head pose estimation in the vehicle, which can attenuate the problem associated with missing frames

(a) The UTDrive Vehicle

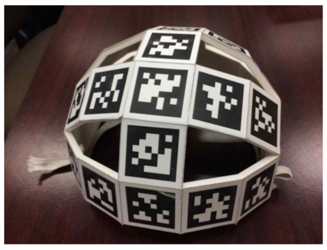

(b) Fi-Cap helmet

Fig. 2. Experimental vehicle used for collecting the multimodal naturalistic data. The figure also shows the Fi-Cap helmet, which provides frame-based ground truth information for the head pose.

due to occlusions, bad illumination or intense head rotations. As shown in Figure 1, each frame is processed by the feature extraction module. The output of this module is fed to our temporal modeling block. We aim to use *recurrent neural networks* (RNNs) to capture the temporal relationship between consecutive frames. BLSTMs are the concatenation of LSTM layers implemented in forward and backward directions, constraining the current frame on previous and future frames, respectively. BLSTMs can leverage short and long time dependencies. They also avoid the vanishing gradient problem of RNNs. Notice that the approach can be alternatively implemented with regular LSTMs if a causal model is needed. We implement the proposed approach with a different number of BLSTM layers. Section VI-D evaluates the best setting for our model. The output of the last BLSTM is connected with a fully connected layer with linear activation. The output layer has six nodes, representing the head orientation.

IV. Multimodal Driver Monitoring Database

A. Description of the Corpus

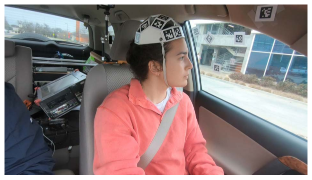
The *multimodal driver monitoring* (MDM) database [37] is a large-scale naturalistic driving corpus with various sensors, including four RGB cameras, a time-of-flight camera, and a microphone array. We also collect the *controller area network-bus* (CAN-Bus) recordings of the vehicle. Figure 3 shows examples of the images collected with the RGB and time-of-flight sensors for one frame. We use GoPro Hero 6 Black cameras to record RGB videos at 60 fps, capturing the driver's head from the back (Fig. 3(a)), the road (Fig. 3(b)), the driver's face (Fig. 3(c)), and the driver's face as seen from the rearview mirror (Fig. 3(d)). In addition, we place a CamBoard pico flex camera close to the face camera recording point cloud data at 45 fps. We acquire depth and gray-scale (Fig. 3(e)) data from this depth camera. The depth data and the gray-scale data are perfectly aligned. We use a clapping board to synchronize the sensors. Figure 2(a) shows the UTDrive (2006 Toyota RAV4) [52]–[54], which is the experimental vehicle used to collect the data.

A key feature of this corpus is the use of the Fi-Cap [38] device, which is shown in Figure 2(b). Fi-Cap is a helmet with 23 predefined 2D fiducial markers. Subjects are asked to wear the Fi-Cap helmet during the data collection. The 2D fiducial markers can be easily tracked since they have predefined patterns with fixed sizes. By tracking these markers,

(a) back camera

(b) road camera

(c) frontal camera


(d) mirror camera

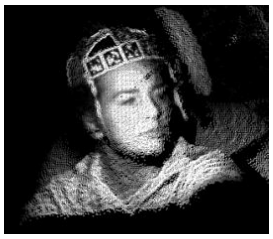

(e) ToF camera (depth and gray)

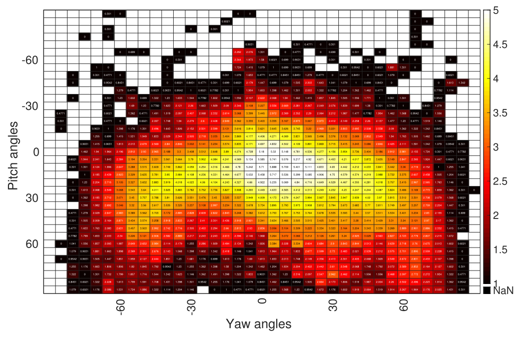
Fig. 3. Examples of images and point cloud data collected from multiple sensors in the MDM database.

we have continuous per-frame head pose annotations. This feature is very important for our study, since we can potentially train our model with each frame in the corpus. Because the Fi-Cap helmet is worn on the back of the driver's head, it does not create occlusions of the driver's face when recorded with frontal cameras. The RGB camera at the back of the vehicle is added to collect the placements of the 2D fiducial markers. The reader is referred to Jha and Busso [38] for more details about the Fi-Cap helmet, and the approach to convert all the detected 2D fiducial markers into head pose angles.

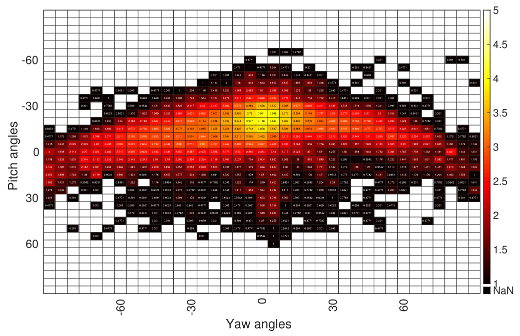
Each driver is instructed to perform the following tasks in the recordings of this corpus:

- In a parked car, the driver is asked to look at markers placed at various locations inside the car. Afterwards, the driver is asked to follow a moving target outside the car held by one of the investigators conducting the data collection.
- While driving the vehicle, the driver is asked to look at the same set of markers inside the car. This step is conducted only when it is safe.
- The driver is asked to follow a predefined route. During this process, the investigator asks the driver to look at street signals, shops, buildings and other vehicles on the road.
- The driver is asked to follow a predefined route shown in a navigation application on a smartphone. The investigator asks the driver to change the radio stations.

The MDM is a naturalistic driving dataset with a large range of head poses and gazes, given the aforementioned protocol. Figure 4(a) shows the distribution of the head orientation angles for yaw and pitch for the MDM dataset.

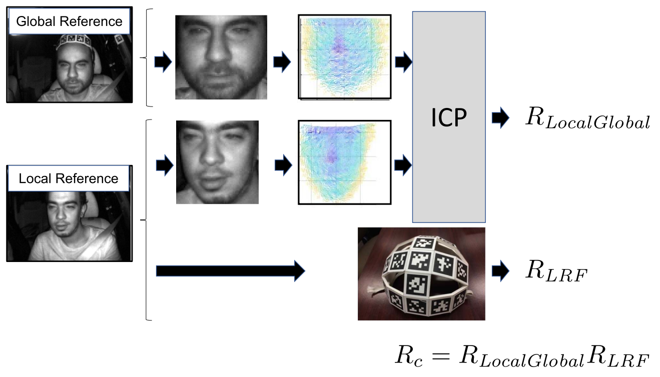


(a) MDM

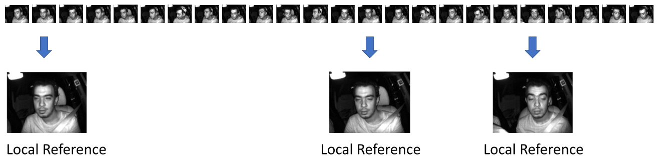


(b) DDPose

Fig. 4. The distribution of the frames as a function of yaw (horizontal) and pitch (vertical) angles of the MDM dataset compared to the DDPose dataset. The figure is in logarithmic scale for better visualization. The lighter the color is in each bin, the more frames are found in each rotation range. The bins in white color and no number means no frame is available in the specific rotation range.



(a) Transformation from local to global reference frames



(b) Selection of multiple *local reference frames* to avoid drifts during a session

Fig. 5. Calibration process to compensate for the different variations across subjects in wearing the Fi-Cap helmet. A global reference is used across subjects, where one or multiple local reference frames are used to normalize the head pose values.

As a reference, we compare the distribution of our corpus with the head orientation angles from the DDPose dataset [55], shown in Figure 4(b). Overall, the MDM corpus provides larger head rotation coverage, including an order-of-magnitude more frames in most regions. It is a relatively large database, where we currently have recordings from 59 subjects. This set consists of 48.9 hours of recordings (10,541,166 frames). The train set has recordings from 39 drivers, the development set has recordings from 10 drivers and the test set has recordings from 10 drivers. The partitions are balanced in terms of gender, and use of glasses. We used all the scenarios of the recordings, including the driver following markers in a parked car, and naturalistic driving recordings, as defined in the data collection protocol. The readers are referred to Jha *et al.* [37] for more details about this corpus. For this study, we build and evaluate the proposed model with point cloud data obtained from the depth camera. The RGB cameras are only used in this study for (1) calibrating the coordinates across drivers (Sec. IV-B), (2) extracting the frame-based head pose labels, and (3) evaluating the RGB baselines.

B. Calibration

In the recording of the MDM dataset, we asked every subject to wear the Fi-Cap helmet [38] to acquire the frame-level ground truth for his/her head pose. We conduct per-subject calibration to obtain the head pose labels, compensating for small placement variations of the Fi-Cap helmet.

We evaluate two alternative calibration approaches. The first approach is the method presented in our previous study [20]. We first identify a *global reference frame* from one of our subjects with frontal pose (frame labeled as “Global Reference” in Fig. 5(a)). This frame is used as a global reference for all the frames across the drivers. Then, we estimate a *local reference frame* (LRF) for each driver, which is the frame with the closest head rotation to the global reference frame, based on the estimations from OpenFace 2.0 [39] on the gray-scale image of the CamBoard pico flex camera. We make the assumption that the global and local reference frames have the same rotation. Let the head rotation matrix of the local reference frame given by Fi-Cap be R_{LRF}. Then, the calibration matrix for that driver is $R_c = R_{LRF}$. For a new frame belonging to this subject with a head rotation of R_{FiCap} given by Fi-Cap, the final head rotation of this frame is $R_t = R_c^{-1} R_{FiCap}$. We refer to this approach as the *single LRF* calibration, since this approach uses one local reference frame per driver.

In our analysis of models trained with this calibration approach, we noted drifts between the ground truth and the predictions using a single local reference frame per driver. These drifts are due to slight movements of the Fi-Cap helmet during the data recording. A second problem is that the head rotations of the *local reference frame* and the *global reference frame* are unavoidably different, which challenges the assumption made in our first approach. To address these limitations, we implement a second approach that relies on more than one local reference frame per driver to avoid drifts and uses the ICP algorithm to compensate for the rotation differences between the local and global reference frames.

Figure 5(a) shows the second calibration approach. This approach finds multiple frames that are closest to the *global* *reference frame* determined by OpenFace 2.0. Notice that the local and global reference frames do not have to have the exact head rotations, where the differences are compensated with the ICP algorithm. Figure 5(b) shows an example where three images are selected as local reference frames. With the local and global reference frames, we perform face detection on the gray-scale data. Because the depth and gray-scale data acquired by the CamBoard pico flex camera are perfectly aligned, we can directly crop the face area around the depth data. We also implement this approach for the global reference frame. The cropped depth images for the global and local reference frames are the input of the ICP algorithm to find a transformation between them. We refer to this transformation as $R_{LocalGlobal}$. This transformation compensates for differences between the local and global reference frames. Then, we estimate the rotation matrix of the LRF given by the Fi-Cap helmet (R_{LRF}). The final calibration matrix Rc is given by multiplying $R_{LocalGlobal}$ and R_{LRF}.

$$R_c = R_{LocalGlobal} R_{LRF} \tag{5}$$

$$R_t = R_c^{-1} R_{FiCap} \tag{6}$$

From a given frame at time t, we obtain the final rotation matrix of this frame by multiplying the inverse of the calibration matrix (R_c^{-1}) with the rotation matrix given by the Fi-Cap helmet (R_{FiCap}). To avoid selecting a noisy local reference frame, we average the calibration matrices from nearby local reference frames. We rank-order the frames based on the increasing distance to the global reference frame. We only consider frames where the angular distance to the global reference frame is less than a given threshold (i.e., the summed squared distance from all three angles should be less than 16.41° – empirically set). We select up to 250 frames from the ranked list, estimating their respective calibration matrices (R_c). We group these candidate local reference frames if they are within 10 seconds from each other. We only consider groups of three or more candidate local reference frames to attenuate the impact of a bad estimation of a single R_c. The average of R_c for each of the selected groups are used as LRF. For a given frame t, we select the calibration matrix R_c from a local reference frame that is the closest to frame t in time. We refer to this approach as the *multiple LRF* calibration, since this approach uses several local reference frames per driver.

V. Experimental Settings

This section describes the experimental settings used to train and evaluate the models. For consistency, all the models are trained on the train set, optimizing the performance on the development set. The test set is only used to evaluate the best models. We describe implementation details for the proposed approach (Sec. V-A) and the details of the baseline models (Sec. V-B).

A. Implementation of the Proposed Approach

The first step in our approach is to downsample the point cloud sequence by four for faster processing. The feature extractor approach is implemented with five repetitions of the sampling-grouping-PointNet sequence. Table I shows the details of the parameters of the proposed network. In each sampling-grouping-PointNet block, we first sample m anchor points from the input using the IFPS algorithm. Notice that the sampling is always done on the 3D coordinates (x, y, and z), as opposed to the outputs from the previous layers. This approach is implemented even in later layers. In the grouping layer, we use ball query to group the features of dimension d_f around each of the m anchor points. We group up to a maximum of l points that lie within a radius of r. We use multiple grouping radii to cover different spatial resolutions (e.g., the first grouping block has $r \in \{0.1, 0.2, 0.4\}$). We use MLP with three layers for the PointNet block. We use a MLP for each of the alternative radii configuration in the grouping block. The number of nodes per layer is given in Table I.

As an example, we further describe the details of the first sampling-grouping-PointNet block. We start by sampling 512 points from the input **S1**. Then, we group $l = 4$, 8 and 16 points, creating **G1**, **G2**, and **G3**, respectively. The outputs of **G1**, **G2**, and **G3** are processed by three MLP. For **G1**,

TABLE I

Implementation Details for the Feature Extraction Module. 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 the 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. For 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]	[512,16] (**T1**)
	[16,16,32]	[512,32] (**T2**)
	[16,24,32]	[512,32] (**T3**)
		[512,80] (**T4**)
Sampling	m=256	[m=256,d_c=3]
Grouping	[r=0.15,l=8]	[m=256,d_f=83,l=8] (**G4**)
	[r=0.3,l=16]	[m=256,d_f=83,l=16] (**G5**)
	[r=0.5,l=24]	[m=256,d_f=83,l=24] (**G6**)
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	[256,256,256]	[256] (**T5**)

the MLP has three layers with 8, 8 and 16 nodes, respectively. For **G2**, the MLP has three layers with 16, 16, and 32 nodes, respectively. For **G3**, the MLP has three layers with 16, 24 and 32 nodes, respectively. The output of the MLP are **T1**, **T2** and **T3**, which are concatenated to form **T4**. For the second sampling-grouping-PointNet block, the sampling operation selects 256 out of the 512 point clouds used in the first block. The input of the grouping operator is the 80D feature representations from **T4**, corresponding to the selected 256 point clouds. This vector is concatenated with the x, y and z coordinates of the selected points, creating a 83D representation (**G4-G6**).

The first four sampling-grouping-PointNet blocks are similar. However, we only sample one point in the last block. All the points are grouped together (i.e., infinite radius r). The output of the grouping block is the input of the three layer MLP, where each of the layers is implemented with 256 nodes. The final embedding for the feature extraction module is **T5**, which is a 256-D vector. This vector is the input for the temporal model.

The temporal model takes as input the 256D feature vector from the feature extractor. It is implemented with two BLSTM layers. Section VI-D investigates the performance of the model based on different number of BLSTMs. The BLSTMs are trained with 256 nodes per direction (512 in total). The output of the BLSTM layers is a 512D feature vector. The length of the sequence is 32-frames (i.e., $N = 32$). At 45 fps, 32 frames corresponds approximately to 2.84s after downsampling the point cloud data by four. The length of the sequence can be increased, but the complexity to train and evaluate the model will also increase.

We use the ADAM optimizer with a learning rate of 0.001 and a learning rate decay of 0.7 per 75,000 steps. The model is implemented with the L2 loss as the cost function. For the BLSTM models, we use a batch size of four with a sequence length of 32 frames. Section VI-E analyzes the performance for different sequence lengths. We train our models on a NVIDIA Quadro RTX 8000 GPU. For the BLSTM models, we first train the model with one BLSTM layer. This model is used to train the proposed model with two BLSTM layers. In Section VI-D, we evaluate the model with three BLSTM layers. Following a similar strategy, we train this model starting from the pre-trained two BLSTM model.

B. Baselines

To demonstrate the effectiveness of our proposed temporal algorithm, we compare it to its static counterpart as well as some commonly available state-of-the-art head pose estimation algorithms form RGB images.

The static version of our models corresponds to the algorithm proposed in our preliminary work [20], which relies exclusively on the feature extraction module in Fig. 1. This approach processes one frame at a time, without considering temporal information. We refer to this approach as the *static model*. We implement the feature extraction block using the same parameters described in Table I. The 256D embedding from the feature extractor is directly connected to the output layers, implemented with a linear activation function to predict the 6D head pose representation. For the static model, we use a batch size of 128. This model will illustrate the benefits of using temporal modeling to estimate head pose from point cloud data.

OpenFace 2.0 [39] is a state-of-the-art face behavior analysis toolkit. For head pose estimation, it first utilizes *convolutional experts constrained local model* (CE-CLM) [56] to detect and track facial landmarks. Then, it solves the Perspective-n-Point Problem to acquire the head pose.

Hopenet [40] is one of the state-of-the-art deep learning based methods for head pose estimation using RGB images. This model is based on the ResNet50 architecture [57] trained on the 300W-LP dataset [58]. The 300W-LP dataset is a large-scale synthetically expanded dataset. They use standardized face images from multiple databases with medium poses. They are rotated in a 3D space after being fitted to a face model to obtain large poses for yaw rotations. Hopenet is a fully end-to-end model without using facial landmark detection.

ZFace [41] is another state-of-the-art automatic face analysis tool, which provides head pose estimation. It first uses a Viola and Jones [59] face detector. Then, it estimates the location of a set of dense face landmarks, fitting a part-based 3D model. They estimate the head pose as part of the fitting process.

For OpenFace 2.0, Hopenet, and ZFace, we use the full 1920×1080 RGB video from the frontal GoPro camera as the input (Fig. 3(c)).

TABLE II
ANALYSIS OF THE CALIBRATION APPROACHES OF THE FI-CAP HELMET. THE TABLE PROVIDES THE MSE FOR THE STATIC AND PROPOSED APPROACHES USING A SINGLE OR MULTIPLE *local reference frame* (LRF)

Method	Calibration	MSE			
		Roll (°)	Yaw (°)	Pitch (°)	Mean (°)
Static Model	single LRF	**5.40**	8.27	13.92	9.20
Static Model	multiple LRF	5.77	**5.84**	**6.33**	**5.98**
Proposed Model	single LRF	**4.01**	6.16	13.41	7.86
Proposed Model	multiple LRF	5.63	**4.69**	**5.30**	**5.20**

VI. EXPERIMENTAL RESULTS

This section describes the experimental evaluation to assess the strengths of the proposed framework.

A. Effectiveness of Calibration

The first part of the evaluation aims to determine the best calibration approach. We train the proposed model and the static model using the two calibration methods described in Section IV-B (i.e., use of one or multiple local reference frames). Notice that the calibration changes the ground truth labels, and, therefore, it has an important effect on the results.

Table II lists the *mean square error* (MSE) between the prediction and the group truth values provided by the Fi-Cap helmet. For the static model, we observe important overall improvements when using the multiple LRF framework. Most of the improvements are associated with the detection of pitch rotations, where the MSE is 7.59° lower. The results for yaw

TABLE III

PERFORMANCE OF THE PROPOSED AND BASELINES SYSTEMS. THE TABLE LISTS THE MSE OF THE MODELS. FOR THE RGB BASELINES, WE COMPARE THE RESULTS ONLY ON THE FRAMES THAT THE RGB ALGORITHMS PROVIDED AN ESTIMATION

Method	MSE			
	Roll(°)	Yaw (°)	Pitch (°)	Mean (°)
	Full Test Set (0% missing frames)			
Static Model	5.77	5.84	6.33	5.98
Proposed Model	**5.63**	**4.69**	**5.30**	**5.20**
	OpenFace Set (3.08% missing frames)			
OpenFace 2.0	7.63	5.06	7.20	6.63
Proposed Model	**5.59**	**4.53**	**5.06**	**5.06**
	Hopenet Set (3.35% missing frames)			
Hopenet	6.62	6.29	8.09	7.00
Proposed Model	**5.54**	**4.52**	**5.06**	**5.04**
	ZFace Set (8.71% missing frames)			
ZFace	8.27	5.68	**4.52**	6.16
Proposed Model	**5.46**	**4.34**	5.04	**4.95**

movements also improve by 2.43° and the results for roll rotation are similar using both calibration methods. For the proposed model, we observe similar results. Using multiple LRF in the calibration provides improvement of over 8° in pitch estimation. While using a single LRF leads to slightly better performance for roll rotation, on average, using multiple LRF leads to 2.66° improvements. We notice that the Fi-Cap slipping problem happens mostly in the pitch axis. Therefore, using multiple LRF for the calibration provides much better results in the pitch axis, which demonstrates its effectiveness as a solution to the problem. Based on the results from this section, we adopt the calibration approach using multiple LRF for the rest of the paper.

B. Comparison With Baselines

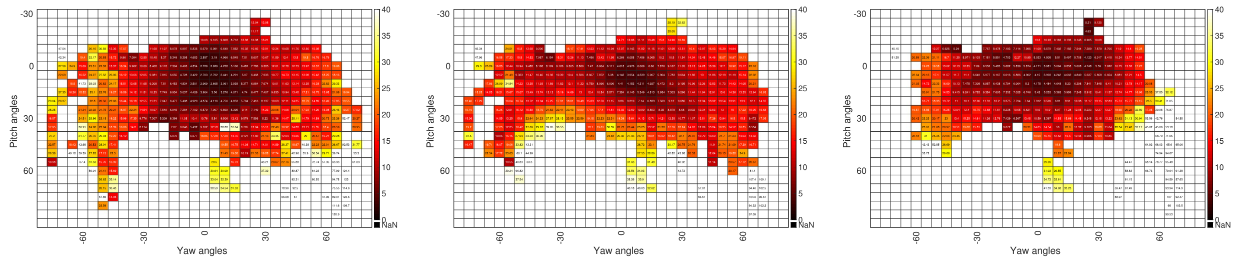
The first comparison is with the static model. The difference between the static and proposed models is the addition of the temporal information by using BLSTM layers. Notice that both of these approaches generate head pose predictions for all the frames. Therefore, the test set used for this evaluation includes all the frames in the test set. Table III shows that the addition of temporal information provides clear improvements over the static method. On average, the MSE error decreases by 0.78° by using the proposed model, which corresponds to a 13% relative improvement. Interestingly, Table II shows that the proposed model with BLSTM is better using either of the calibration methods. The improvements are consistently observed for roll, yaw and pitch rotations.

We also compare our model to the OpenFace 2.0, Hopenet and ZFace toolkits in head pose estimation. The first block in these methods is detecting the face. For some frames, these methods fail to detect the face, so they cannot provide head pose information. The percentage of missing frames are 3.08%, 3.35% and 8.71% for OpenFace 2.0, Hopenet and ZFace, respectively. In contrast, our point cloud based approach does not require face detection, so we have predictions for all frames. This is another advantage of our approach. Since the frames that are missed by the RGB based methods are often the most challenging cases, it is not fair to compare these approaches with our model using the entire testing set. Instead, we reevaluate the performance of our method using only the frames that were successfully processed by the RGB approaches. For example, the *OpenFace set* in Table III corresponds to all the frames that were successfully processed by the OpenFace 2.0 toolkit. Our proposed temporal model clearly outperforms all baselines using RGB images, especially for roll rotations. For yaw and roll rotations, we observe consistent improvements over all the baselines. For pitch rotation, ZFace is the only method that outperforms our proposed model. However, our method provides a 19.6% reduction in the average angular error across the three rotations. Another important observation is that ZFace does not provide results for as many as 8.71% of the frames, which may cause it to be unreliable in critical segments. As a result, the angular range for pitch rotation for Zface is more limited than the range provided by our model (see Figs. 6(c) and 6(e)).

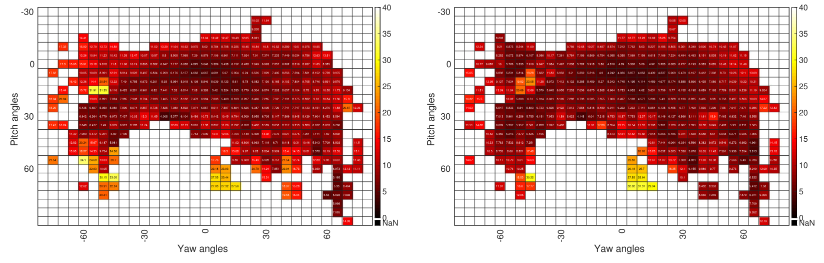
Figure 6 visualizes the results for the proposed and baseline methods. The figures plot the average error across the three angles as a function of yaw and pitch angle. The yaw-pitch space is split into bins. Each bin contains all the frames with head pose labels lying within the yaw and pitch ranges defined by the bin. Darker bins indicate a lower error in the head pose predictions. Empty bins indicate that no frames are available in the specific rotation range. This visualization helps us understand how each method performs in each angle range. Figure 6(e) shows that the proposed model provides comparable or better results across angles than the results obtained with OpenFace 2.0, Hopenet and ZFace. The benefits of using our approach is especially clear in frames with large rotations. The proposed model provides the most competitive results in all rotation ranges. This is an important strength of our model compared to state-of-the-art head pose estimation algorithms. OpenFace 2.0 is very competitive in the central region, which has frames with small head pose angles (Fig. 6(a)). However, the performance drops significantly as the head rotation becomes larger. The results for Hopenet shares a similar trend. (Fig. 6(b)). The algorithm does not provide results for many frames with pitch rotation above 60°. Figure 6(d) shows that ZFace provides competitive results in the center region of the heatmap, where the overall head rotation is mostly frontal. However, it provides results in a much smaller angle range compared to other approaches, in particular, our proposed approach. While the improvement from the static model is present across all ranges, it is especially noticeable in frames with yaw angles bigger than 50°.

C. Failure Examples

Figure 7 shows some examples where the model does not perform well. In Figure 7(a), we believe the model has large error because the face is severely occluded from the left, resulting in visibility of less than half of the face. In Figures 7(b) and 7(c) the ground truth rotation is relatively large. In the training data, frames with large rotation tend to appear less frequent than frames with smaller rotations.

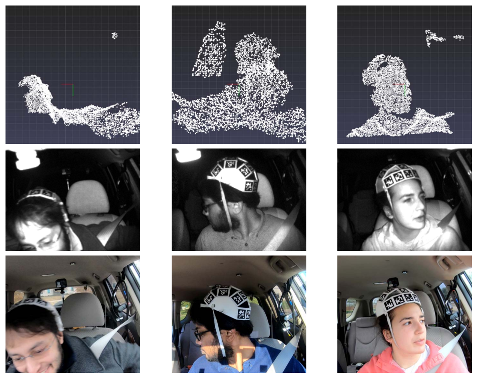


(a) OpenFace 2.0 (b) Hopenet (c) ZFace



(d) Static (e) Proposed BLSTM Model

Fig. 6. Performance of the proposed algorithm compared to the baselines as a function of yaw and pitch angles. The figure shows the per bin average geodesic distance between the true and predicted head rotation. Darker color indicates lower errors. The bins that have white color and are empty indicate that no frames are available in the specific rotation range.



	(a)	(b)	(c)
Ground Truth (°)	[14.51, -7.29, 20.40]	[74.81, 44.15, 42.98]	[-14.69, -36.43, 5.48]
Prediction (°)	[44.40, 14.26, 49.87]	[63.08, 58.21, 54.00]	[-3.36, -25.13, -0.75]

Fig. 7. Examples of frames in which the proposed model does not work well. The figure lists the Euler angles of the ground truth and predicted head orientation for roll, yaw and pitch (in that order).

TABLE IV
COMPARISON OF THE PROPOSED MODEL IMPLEMENTED WITH ONE, TWO OR THREE BLSTM LAYERS

Method	MSE			
	Roll (°)	Yaw (°)	Pitch (°)	Mean (°)
1 BLSTM	6.02	5.08	**5.28**	5.46
2 BLSTM	**5.63**	4.69	5.30	**5.20**
3 BLSTM	5.77	**4.62**	6.13	5.50

TABLE V
STUDY OF SEQUENCE LENGTH USED TO TRAIN THE BLSTM LAYERS. THE TABLE COMPARES THE PROPOSED MODEL TRAINED WITH SEQUENCES OF 16, 32 OR 48 FRAMES

# of frames	MSE			
	Roll (°)	Yaw (°)	Pitch (°)	Mean (°)
16	5.82	5.06	5.21	5.36
32	**5.63**	4.69	5.30	5.20
48	5.65	**4.61**	**5.10**	**5.12**

Therefore, it is more difficult for the model to predict large rotations.

D. Effect of the Number of BLSTM Layers

This section studies the optimal number of BLSTM layers. We implement our proposed approach with one, two and three BLSTM layers. We keep all the other hyperparameters the same, reporting the MSE results on the test set of the MDM database. Table IV shows the results. The proposed model outperforms the static model in terms of MSE even with one BLSTM layer (see Table III). This result illustrates the importance of temporal modeling for head pose estimation, especially in challenging environments such as inside a vehicle. On average, Table IV shows that we have relative improvements of 4.76% when we increase the number of BLSTM layers from one to two. The performance gains are on roll and yaw rotations. The accuracy in yaw estimation for a head pose estimation system in a vehicle application is often more crucial than the other two angles. Therefore, the 7.67% relative improvement in yaw estimation is important. On average, we observe a 5.76% relative drop in performance when we increase the number of BLSTM layers from two to three. Roll and pitch rotations are the angles that are more affected. Two BLSTM layers are enough to model the temporal relationship in the data for this problem.

E. Effect of Sequence Length

We conduct an additional experiment on the proposed approach implemented with two BLSTMs trained to evaluate the impact of the sequence length in the model. We compare the model trained with sequence length of 16, 32 and 48 frames, using a batch size of 8, 4 and 3 while keeping all other hyperparameters the same. The results are shown in the Table V. We observe that the model trained with a sequence length of 48 frames performs the best, and the model with 16 frames performs to worst. The improvement from 16 to 32 frames is slightly bigger than from 32 to 48 frames. The result from this table is in line with our expectation. Adding longer sequences to train the BLSTM leads to better results as longer contextual windows enable the model to better fit the data. Using 32 frames, as proposed in this study, is a good compromise between computational complexity and performance.

F. Smoothness of the Head Pose Trajectories

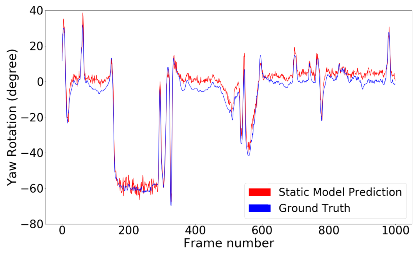
In addition to improving the accuracy in head pose estimation, modeling temporal information also provides smoother trajectories. Figure 8 compares the prediction of the static and proposed models for a segment in the test set. The figure also includes the prediction of the static model after low-pass filtering the output to smooth its trajectory. The curves provide the yaw rotations as an example. We observe that the predictions from the proposed model are much smoother than the predictions of the static model, providing a better fit for the ground truth trajectory. While using a low-pass filter can effectively smoothen the trajectories, Figure 8(b) shows that the filtered predictions are not as good as the predictions by the temporal model with BLSTM layers (Fig. 8(c)).

We present two metrics to quantify the smoothness of the trajectories in the entire test set. The first metric is the lag-1 autocorrelation of the curves. We shift the time-series by one frame and compute the Pearson correlation between the original and the shifted time series. For smooth trajectories, the correlation between both sequences will be high. The second metric is the *root mean squared of the difference* (RMSD) of the trajectories. We find the difference in the predictions of consecutive frames. Then, we estimate its root mean square values. For smooth trajectories, the difference between consecutive frames should be small. Therefore, they will have smaller RMSD value.

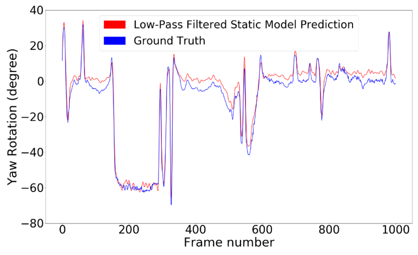
Table VI shows the results for the static and proposed model implemented with two BLSTMs evaluated on the test set. Compared to the static model, the proposed temporal model provides a large improvement in both of the smoothness metrics, which confirms the qualitative findings observed from Figure 8. As expected, the low-pass filter applied to the output of the static model clearly improve the smoothness in the predictions. However, despite the smoothness improvement, Table VI shows that using low-pass filter only provides limited improvements in terms of MSE. The model with BLSTM fits the data better than this static model.

G. Computational Resource

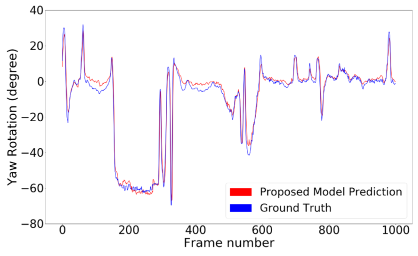
We train and test our models on an NVIDIA Quadro RTX 8000 GPU. We present the computational resource required, both in terms of time and GPU memory to train and test one batch of 32 frames of our proposed model as well as the static model without the BLSTM layers in Table VII. The proposed model with LSTMs has more parameters and requires more memory and time during both training and testing. However, the inference time is merely 52ms or equivalently 615 FPS. This speed is faster than the typical 30 or 60 FPS video streams, so real time implementation is feasible. If latency is a problem, we can also replace the BLSTMs layers with simpler



(a) Static model



(b) Static model with low-pass



(c) Proposed Model

Fig. 8. Example of a head pose trajectory predicted by the static model, static model with low-pass filter, and the proposed temporal model for yaw movements (in degrees). The trajectories are compared with the ground truth trajectories obtained with the Fi-Cap helmet.

TABLE VI

SMOOTHNESS ANALYSIS OF THE PREDICTED HEAD POSE TRAJECTORIES. THE TABLE PRESENTS THE LAG-1 AUTOCORRELATION AND RMSD FOR THE STATIC MODEL, STATIC MODEL WITH LOW-PASS FILTER AND PROPOSED TEMPORAL MODEL. THE TABLE ALSO LISTS THE PERFORMANCE OF THE MODELS IN TERMS OF MSE

Method	Lag-1 autocorrelation (smoothness)			
	Roll	Yaw	Pitch	Mean
Static Model	0.9431	0.9645	0.9311	0.9462
Static Model w. LP Filter	**0.9917**	**0.9915**	**0.9941**	**0.9925**
Proposed Model (2 BLSTM)	0.9874	0.9866	0.9888	0.9876
Method	RMSD (smoothness)			
	Roll (°)	Yaw (°)	Pitch (°)	Mean (°)
Static Model	3.11	3.78	2.76	3.22
Static Model w. LP Filter	**1.15**	**1.81**	**0.78**	**1.24**
Proposed Model (2 BLSTM)	1.48	2.29	1.03	1.60
Method	MSE (prediction accuracy)			
	Roll	Yaw	Pitch	Mean
Static Model	5.77	5.84	6.33	5.98
Static Model w. LP Filter	**5.53**	5.57	6.10	5.73
Proposed Model (2 BLSTM)	5.63	**4.69**	**5.30**	**5.20**

TABLE VII

COMPUTATIONAL RESOURCE REQUIRED FOR THE STATIC MODEL AND PROPOSED MODEL FOR A BATCH OF 32 FRAMES

Method	Par. # (M)	Training vRAM (MB)	Training Time (ms)	Inference vRAM (MB)	Inference Time (ms)
Static Model	0.31	8868	101	3702	30
Proposed Model	1.36	9322	163	4051	52

TABLE VIII

RESULT OF THE PROPOSED TEMPORAL APPROACH IN THE BIWI DATASET

# of frames	MAE, STD			
	Roll (°)	Yaw (°)	Pitch (°)	Mean (°)
POSEidon [18]	1.8±1.8	1.7±1.5	1.6±1.7	1.7±1.7
Meyer [13]	2.1	2.1	2.4	2.2
Proposed	**1.4±1.2**	1.8±**1.4**	**1.3±1.3**	**1.5±1.3**

LSTM layers. While a mobile platform currently has less computational power compared to our GPU, we are confident that by pruning and optimizing the model, our approach can run in real-time in an embedded in-vehicle system.

H. Result on Biwi

Finally, we evaluate our approach in a second database to validate the proposed approach in a different set of recordings. We consider the Biwi Kinect head pose database [60], which has been widely used for head pose estimation. The recordings are captured with a Microsoft Kinect from 20 subjects, who were asked to create 24 sequences. In total, the corpus has about 15k frames, with both RGB and depth data. The rotations are in the range $\pm$ 50° for roll, $\pm$ 75° for yaw and $\pm$ 60° for pitch. We crop out the face of the depth data using the provided bounding box and convert the depth data into point clouds. We follow the train-test split defined in Borghi *et al.* [18]. We train the proposed temporal approach with two BLSTM layers on the Biwi dataset, starting with a pre-trained model built with the MDM dataset. Table VIII presents the result of our method on the test set, and the results from other two depth-based methods. We report the results in term of the *mean absolute error* (MAE) for consistency with previous studies. Our proposed model clearly outperforms the approach from Meyer *et al.* [13] and Borghi *et al.* [18]. The only exception is in the yaw axis, where the model from Borghi *et al.* [18] has a slight margin. Another advantage of our model is the lower standard deviation in the error, indicating that our model is more consistent.

VII. CONCLUSION

This paper presented a novel temporal deep learning based head pose estimation model for point cloud data, targeting automotive applications. The approach consists of a frame-based feature extractor module that creates representations, which are later connected by a temporal modeling module. The feature extractor extracts discriminative feature representation directly from point cloud by sampling the data, grouping the points, and extracting local features that are later combined. The temporal modeling block takes the frame-based feature representations capturing the important relationship in head pose across nearby frames. To demonstrate the effectiveness of our proposed algorithm, we collected a naturalistic multimodal database from multiple drivers. This corpus provides head pose information for the driver for each frame, providing an ideal platform to train and evaluate our proposed approach. The experimental section showed that our proposed temporal model outperforms a static model using point cloud data that does not leverage temporal information. The predicted head pose trajectories are not only more accurate, but also smoother. Moreover, we compared the proposed algorithm with state-of-the-art RGB-based head pose estimation algorithms, achieving superior performances. The improvements in performance are particularly clear for frames with non-frontal head rotations. Furthermore, our proposed temporal solutions can provide reliable head pose estimations even when the head pose is extreme. These are challenging cases for RGB-based solutions that tend to work only with limited head rotations. These results are crucial for in-vehicle applications, as non-frontal head poses are particularly important for monitoring driver attention (e.g., eyes-off-the-road [61], mirror checking actions [6], visual and cognitive distractions [5]). The computational analysis demonstrated that the system can process 650FPS, which is enough for real-time implementation. Finally, the model was validated with the BIWI dataset, obtaining better performance than previous studies.

The results from this study suggest that point cloud data is an appealing alternative to RGB data in the estimation of head pose information in challenging environments. The structure of the face, as obtained from the cloud point data, provides enough information to reliably estimate head pose rotation. While the proposed temporal approach was evaluated in a vehicle environment, we expect that this approach can also be extremely useful in other applications. For example, we expect that our proposed approach can lead to better performance than RGB based solutions in applications evaluated in crowded environment.

As part of our future work, we will work on combining different sensor information for head pose estimation (RGB, infrared and depth images). We envision algorithms that can independently work with a subset of modalities, increasing the robustness as more information is available. This flexible, multimodal solution offers an ideal formulation to deal with challenging conditions for in-vehicle applications, relying only on the modalities that are available for any given frame. We will also explore alternative temporal modeling formulation, analyzing the tradeoff between accuracy in our predictions and the computational resources required to operate the system. Finally, head pose estimation constitutes an important building block for predicting gaze, given the strong correlation between the driver head pose and gaze [62]. Various studies have used this information to predict coarse visual attention from the driver head pose [63]. Building on our previous studies using RGB cameras [64]–[66], we plan to use head pose derived from depth sensors to build probabilistic visual map describing gaze regions driven by head pose.

REFERENCES

[1] (2019). *The National Highway Traffic Safety Administration (NHTSA)*. Accessed: Aug. 5, 2020. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[3] K. Liu, Y. Luo, G. Tei, and S. Yang, "Attention recognition of drivers based on head pose estimation," in *Proc. IEEE Vehicle Power Propuls. Conf.*, Harbin, China, Sep. 2008, pp. 1–5.

[4] N. Li, J. J. Jain, and C. Busso, "Modeling of driver behavior in real world scenarios using multiple noninvasive sensors," *IEEE Trans. Multimedia*, vol. 15, no. 5, pp. 1213–1225, Aug. 2013.

[5] N. Li and C. Busso, "Predicting perceived visual and cognitive distractions of drivers with multimodal features," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 1, pp. 51–65, Feb. 2015.

[6] N. Li and C. Busso, "Detecting drivers' mirror-checking actions and its application to maneuver and secondary task recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 980–992, Apr. 2016.

[7] G. L. Masala and E. Grosso, "Real time detection of driver attention: Emerging solutions based on robust iconic classifiers and dictionary of poses," *Transp. Res. C, Emerg. Technol.*, vol. 49, pp. 32–42, Dec. 2014.

[8] J. I. Árnason, J. Jepsen, A. Koudal, M. R. Schmidt, and S. Serafin, "Volvo intelligent news: A context aware multi modal proactive recommender system for in-vehicle use," *Pervas. Mobile Comput.*, vol. 14, pp. 95–111, Oct. 2014.

[9] R. O. Mbouna, S. G. Kong, and M.-G. Chun, "Visual analysis of eye state and head pose for driver alertness monitoring," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1462–1469, Sep. 2013.

[10] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Providence, RI, USA, Jun. 2011, pp. 617–624.

[11] P. Padeleris, X. Zabulis, and A. A. Argyros, "Head pose estimation on depth data based on particle swarm optimization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Providence, RI, USA, Jun. 2012, pp. 42–49.

[12] T. Bar, J. F. Reuter, and J. M. Zollner, "Driver head pose and gaze estimation based on multi-template ICP 3-D point cloud alignment," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Anchorage, AK, USA, Sep. 2012, pp. 1797–1802.

[13] G. P. Meyer, S. Gupta, I. Frosio, D. Reddy, and J. Kautz, "Robust model-based 3D head pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 3649–3657.

[14] B. Wang, W. Liang, Y. Wang, and Y. Liang, "Head pose estimation with combined 2D SIFT and 3D HOG features," in *Proc. 7th Int. Conf. Image Graph. (ICIG)*, Qingdao, China, Jul. 2013, pp. 650–655.

[15] A. Saeed and A. Al-Hamadi, "Boosted human head pose estimation using kinect camera," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Quebec City, QC, Canada, Sep. 2015, pp. 1752–1756.

[16] C. Papazov, T. K. Marks, and M. Jones, "Real-time 3D head pose and facial landmark estimation from depth images using triangular surface patch features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 4722–4730.

[17] A. Schwarz, M. Haurilet, M. Martinez, and R. Stiefelhagen, "DriveAHead—A large-scale driver head pose dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Honolulu, HI, USA, Jul. 2017, pp. 1165–1174.

[18] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara, "POSEidon: Face-from-depth for driver pose estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 5494–5503.

[19] M. Venturelli, G. Borghi, R. Vezzani, and R. Cucchiara, "Deep head pose estimation from depth data for in-car automotive applications," in *Proc. Int. Workshop Understand. Hum. Activities Through 3D Sensors (UHADS)*, in Lecture Notes in Computer Science, vol. 10188, H. Wannous, P. Pala, M. Daoudi, and F. Flórez-Revuelta, Eds., Cancun, Mexico. Berlin, Germany: Springer, Dec. 2018, pp. 74–85.

[20] T. Hu, S. Jha, and C. Busso, "Robust driver head pose estimation in naturalistic conditions from point-cloud data," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Las Vegas, NV, USA, Oct. 2020, pp. 1176–1182.

[21] S. Jha and C. Busso, "Challenges in head pose estimation of drivers in naturalistic recordings using existing tools," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Yokohama, Japan, Oct. 2017, pp. 1624–1629.

[22] S. Li, K. N. Ngan, R. Paramesran, and L. Sheng, "Real-time head pose tracking with online face template reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1922–1928, Sep. 2016.

[23] M. Martin, F. Van De Camp, and R. Stiefelhagen, "Real time head model creation and head pose estimation on consumer depth cameras," in *Proc. 2nd Int. Conf. 3D Vis. (DV)*, Tokyo, Japan, Dec. 2014, pp. 641–648.

[24] Y. Yu, K. A. F. Mora, and J.-M. Odobez, "HeadFusion: 360° head pose tracking combining 3D morphable model and 3D reconstruction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 11, pp. 2653–2667, Nov. 2018.

[25] L. Sheng, J. Cai, T.-J. Cham, V. Pavlovic, and K. N. Ngan, "A generative model for depth-based robust 3D facial pose tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 4598–4607.

[26] S. Li, K. Ngan, and L. Sheng, "A head pose tracking system using RGB-D camera," in *Proc. Int. Conf. Comput. Vis. Syst. (ICVS)*, in Lecture Notes in Computer Science, vol. 7963, M. Chen, B. Leibe, and B. Neumann, Eds., St. Petersburg, Russia. Berlin, Germany: Springer, Jul. 2013, pp. 153–162.

[27] T. Baltrušaitis, P. Robinson, and L. Morency, "3D constrained local model for rigid and non-rigid facial tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 2610–2617.

[28] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional LSTM with CNN features," *IEEE Access*, vol. 6, pp. 1155–1166, Nov. 2018.

[29] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 677–691, Apr. 2017.

[30] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science, vol. 9911, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Amsterdam, The Netherlands. Berlin, Germany: Springer, Oct. 2016, pp. 766–782.

[31] S. E. Kahou, V. Michalski, K. Konda, R. Memisevic, and C. Pal, "Recurrent neural networks for emotion recognition in video," in *Proc. ACM Int. Conf. Multimodal Interact. (ICMI)*, Seattle, WA, USA, Nov. 2015, pp. 467–474.

[32] A. N. Salman and C. Busso, "Style extractor for facial expression recognition in the presence of speech," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Abu Dhabi, United Arab Emirates, Oct. 2020, pp. 1806–1810.

[33] C. Ventura, M. Bellver, A. Girbau, A. Salvador, F. Marques, and X. Giro-i-Nieto, "RVOS: End-to-end recurrent network for video object segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 5272–5281.

[34] A. Pukach, T. Hershkovich, and P. Kozlov, "3D moving object point cloud refinement using temporal inconsistencies," U.S. Patent 16583787, Jan. 16, 2020.

[35] R. Villegas, A. Pathak, H. Kannan, D. Erhan, Q. Le, and H. Lee, "High fidelity video prediction with large stochastic recurrent neural networks," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, Vancouver, BC, Canada, Dec. 2019, pp. 81–91.

[36] C. Qi, L. Yi, H. Su, and L. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 5099–5108.

[37] S. Jha, M. F. Marzban, T. Hu, M. H. Mahmoud, N. Al-Dhahir, and C. Busso, "The multimodal driver monitoring database: A naturalistic corpus to study driver attention," Dec. 2020, *arXiv:2101.04639*. [Online]. Available: http://arxiv.org/abs/2101.04639

[38] S. Jha and C. Busso, "FI-CAP: Robust framework to benchmark head pose estimation in challenging environments," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, San Diego, CA, USA, Jul. 2018, pp. 1–6.

[39] T. Baltrušaitis, A. Zadeh, Y. C. Lim, and L.-P. Morency, "OpenFace 2.0: Facial behavior analysis toolkit," in *Proc. 13th IEEE Int. Conf. Automat. Face Gesture Recognit. (FG)*, Xi'an, China, May 2018, pp. 59–66.

[40] N. Ruiz, E. Chong, and J. M. Rehg, "Fine-grained head pose estimation without keypoints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Salt Lake City, UT, USA, Jun. 2018, pp. 2187–2196.

[41] L. A. Jeni, J. F. Cohn, and T. Kanade, "Dense 3D face alignment from 2D videos in real-time," in *Proc. 11th IEEE Int. Conf. Workshops Automat. Face Gesture Recognit. (FG)*, Ljubljana, Slovenia, May 2015, pp. 1–8.

[42] B. Ahn, D.-G. Choi, J. Park, and I. S. Kweon, “Real-time head pose estimation using multi-task deep neural network,” *Robot. Auton. Syst.*, vol. 103, pp. 1–12, May 2018.

[43] V. Blanz and T. Vetter, “A morphable model for the synthesis of 3D faces,” in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, Los Angeles, CA, USA, 1999, pp. 187–194.

[44] P. Jiménez, L. M. Bergasa, J. Nuevo, and P. F. Alcantarilla, “Face pose estimation with automatic 3D model creation in challenging scenarios,” *Image Vis. Comput.*, vol. 30, no. 9, pp. 589–602, Sep. 2012.

[45] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 77–85.

[46] Y. Wang, Y. Sun, Z. Liu, S. Sarma, M. Bronstein, and J. Solomon, “Dynamic graph CNN for learning on point clouds,” *ACM Trans. Graph.*, vol. 38, no. 5, pp. 146:1–146:12, Oct. 2019.

[47] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, “PointCNN: Convolution on X-transformed points,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Montreal, QC, Canada, Dec. 2018, pp. 820–830.

[48] G. Borghi, R. Gasparini, R. Vezzani, and R. Cucchiara, “Embedded recurrent network for head pose estimation in car,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Los Angeles, CA, USA, Jun. 2017, pp. 1503–1508.

[49] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[50] X. Liu, M. Yan, and J. Bohg, “MeteorNet: Deep learning on dynamic 3D point cloud sequences,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct./Nov. 2019, pp. 9245–9254.

[51] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, “On the continuity of rotation representations in neural networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 5738–5746.

[52] P. Angkititrakul, M. Petracca, A. Sathyanarayana, and J. H. L. Hansen, “UTDrive: Driver behavior and speech interactive systems for in-vehicle environments,” in *Proc. IEEE Intell. Vehicles Symp.*, Istanbul, Turkey, Jun. 2007, pp. 566–569.

[53] P. Angkititrakul *et al.*, “Getting start with UTDrive: Driver-behavior modeling and assessment of distraction for in-vehicle speech systems,” in *Proc. Interspeech*, Antwerp, Belgium, Aug. 2007, pp. 1334–1337.

[54] J. H. L. 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 Process. Mag.*, vol. 34, no. 4, pp. 130–142, Jul. 2017.

[55] M. Roth and D. M. Gavrila, “DD-pose—A large-scale driver head pose benchmark,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Paris, France, Jun. 2019, pp. 1176–1182.

[56] A. Zadeh, Y. C. Lim, T. Baltrusaitis, and L.-P. Morency, “Convolutional experts constrained local model for 3D facial landmark detection,” in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Venice, Italy, Oct. 2017, pp. 2519–2528.

[57] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun./Jul. 2016, pp. 770–778.

[58] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, “Face alignment across large poses: A 3D solution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 146–155.

[59] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Kauai, HI, USA, vol. 1, Dec. 2001, pp. 511–518.

[60] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, “Random forests for real time 3D face analysis,” *Int. J. Comput. Vis.*, vol. 101, no. 3, pp. 437–458, Feb. 2013.

[61] Y. Peng, L. N. Boyle, and S. L. Hallmark, “Driver’s lane keeping ability with eyes off road: Insights from a naturalistic study,” *Accident Anal. Prevention*, vol. 50, pp. 628–634, Jan. 2013.

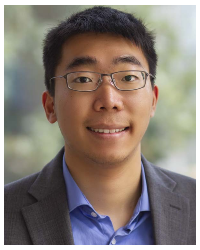
[62] S. Jha and C. Busso, “Analyzing the relationship between head pose and gaze to model driver visual attention,” in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Rio de Janeiro, Brazil, Nov. 2016, pp. 2157–2162.

[63] A. Tawari, S. Martin, and M. M. Trivedi, “Continuous head movement estimator for driver assistance: Issues, algorithms, and on-road evaluations,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 818–830, Apr. 2014.

[64] S. Jha and C. Busso, “Probabilistic estimation of the driver’s gaze from head orientation and position,” in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Yokohama, Japan, Oct. 2017, pp. 1630–1635.

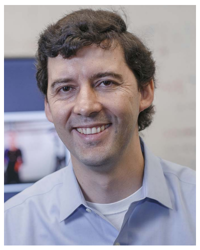
[65] S. Jha and C. Busso, “Probabilistic estimation of the gaze region of the driver using dense classification,” in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Maui, HI, USA, Nov. 2018, pp. 697–702.

[66] S. Jha and C. Busso, “Estimation of driver’s gaze region from head position and orientation using probabilistic confidence regions,” Dec. 2020, *arXiv:2012.12754*. [Online]. Available: http://arxiv.org/abs/2012.12754



Tiancheng Hu (Student Member, IEEE) received the B.Sc. degree in electrical engineering from The University of Texas at Dallas (UTD) in 2020. From 2018 to 2019 school year, he was an Assistant Director of Tutoring of the IEEE Student Branch at UTD. During his undergraduate studies, he also worked as an Undergraduate Researcher with the Multimodal Signal Processing (MSP) Laboratory. His current research interests include computer vision and perception for advanced driver-assistance systems. He was a recipient of the University-Wide Undergraduate Research Scholarship and the Jonsson School Undergraduate Research Award, in 2019 and 2020, respectively.

Sumit Jha (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from the National Institute of Technology (NIT), Trichy, India, in 2012, and the M.S. degree in electrical engineering from The University of Texas at Dallas (UTD), in 2016, where he is currently pursuing the Ph.D. degree. At UTD, he has been a part of the Multimodal Signal Processing (MSP) Laboratory, since 2015. His research interests include machine learning computer vision solutions for driver monitoring and in-vehicle safety systems.



Carlos Busso (Senior Member, IEEE) received the B.S. and M.S. degrees (Hons.) from the University of Chile, Santiago, Chile, in 2000 and 2003, respectively, and the Ph.D. degree from the University of Southern California (USC), Los Angeles, in 2008, all in electrical engineering. He is currently a Professor with the Department of Electrical Engineering, The University of Texas at Dallas (UTD). At UTD, he leads the Multimodal Signal Processing (MSP) Laboratory. His work has direct implication in many practical domains, including national security, health care, entertainment, transportation systems, and education. He has coauthored the winner paper of the Classifier Sub-Challenge Event at the Interspeech 2009 Emotion Challenge. His research interest includes human-centered multimodal machine intelligence and applications. His current research interests include the broad areas of affective computing, multimodal human–machine interfaces, nonverbal behaviors for conversational agents, in-vehicle active safety systems, and machine learning methods for multimodal processing. He is a member of ISCA and AAAC, and a Senior Member of the ACM. He was a recipient of the NSF CAREER Award. In 2014, he received the ICMI Ten-Year Technical Impact Award. In 2015, his student received the Third Prize IEEE ITSS Best Dissertation Award (N. Li). He also received the Hewlett Packard Best Paper Award at the IEEE ICME 2011 (with J. Jain), and the Best Paper Award at the AAAC ACII 2017 (with Yannakakis and Cowie). From USC, he received a Provost Doctoral Fellowship, from 2003 to 2005, and a Fellowship in Digital Scholarship, from 2007 to 2008. He was selected by the School of Engineering of Chile, as the Best Electrical Engineer, graduated in 2003, across Chilean Universities. He was the General Chair of ACII 2017 and ICMI 2021.