Comparing Head and AR Glasses Pose Estimation

Ahmet Firintepe*
BMW Group Research, New Technologies, Innovations, Garching (Munich)
TU Kaiserslautern, Germany

Oussema Dhaouadi†
BMW Group Research, New Technologies, Innovations, Garching (Munich), Germany

Alain Pagani‡
German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany

Didier Stricker§
German Research Center for Artificial Intelligence (DFKI)
TU Kaiserslautern, Germany

ABSTRACT

In this paper, we compare AR glasses and head pose estimation performance. We train different pose estimation approaches for head pose estimation with the generated head pose labels to compare them to their AR glasses estimation accuracy. These include the state-of-art GlassPoseRN and P2P networks, as well as our novel CapsPose algorithm. We show that estimating the AR glasses pose is more accurate than the head pose in general. In a first analysis, we show the general regression performance of the models when the AR glasses and faces are both known to the network during training. We then analyze the driver generalization performance, where all glasses are known, but part of the drivers are unknown to the Neural Networks. There, the estimation of AR glasses pose again exceeds the head pose. Only in our third analysis, head pose estimation performs better than AR glasses pose estimation. In this case, a new glasses model is added, which was unknown to the Neural Network yet. In addition, we introduce a novel pose estimation network called CapsPose, which is the first network deploying Capsule Networks for 6-DoF pose estimation. We outperform the current state-of-the-art method GlassPoseRN on the HMDPose dataset by reducing the error by 46% for orientation and 51% for translation.

Index Terms: Computing methodologies—Artificial intelligence—Computer vision—Tracking; Computing methodologies—Machine learning—Machine learning approaches—Neural networks

1 INTRODUCTION

AR glasses steadily matured over the years and are currently being used in the industry. In the future, consumer adaptation of AR glasses is highly likely. This comprises the usage while driving a car to enable use cases like AR navigation or Point of Interest highlighting. For this purpose, it is necessary to predict the 6-DoF pose of the AR glasses inside a moving vehicle for an accurate superimposition of virtual elements on top of the real world. Various AR glasses conduct inside-out tracking in static environments, which is not scalable for the in-car use case. This would require identical sets of sensors and computation capacities for all AR glasses models of different manufacturers. In addition, inside-out tracking is challenging in the car, as the optical sensors inside Head-Mounted Displays register parts of the outside world and deliver a limited set of features of the car interior for tracking. The problem is solvable by outside-in tracking, which enables generalization and ensures the capturing of features of the car interior only.

Head pose estimation has been in the focus of the Computer Vision community for many years. Different data sources and methods for this purpose exist. The data sources range from RGB [13, 15] to infrared (IR) images [7] to depth images [2], as well as combinations like RGB-D information [19]. Before the Deep Learning era, this was done mainly based on feature extraction from the images, matching them with predefined features, and conducting pose estimation of the head. Recent methods incorporate Deep Learning, using Deep Neural Networks to estimate the 6-DoF head pose.

Similar algorithms can be deployed for AR glasses pose estimation as done for head pose estimation. In general, the pose of the AR glasses can be obtained using different approaches: first, one can consider the AR glasses as one rigid object and use object pose estimation algorithms to compute the pose. Second, we can consider the head of the person wearing the glasses as the entity to track, and apply head pose estimation algorithms. Theoretically, the head pose is different from the AR glasses, as the relation between head and glasses differs from user to user. This relation can change with slight glasses movement on the wearer's head. Still, for the real-world deployment, the question remains if head pose estimation can deliver more accurate pose results, making it a more stable alternative than AR glasses pose estimation. To answer this question, a comparison of both estimation methods on the same data source is required.

Thus, in this paper, we first generate head pose ground truth from an AR glasses dataset called HMDPose. Then, we benchmark two existing algorithms on the head pose, which were previously trained on the same dataset with the target of AR glasses pose estimation. We introduce a novel pose estimation method called CapsPose, which is based on Capsules. We benchmark this on head and AR glasses pose estimation. We then compare and evaluate the performance of the approaches, showing that the estimation of AR glasses results in a mostly better performance.
In detail, our contributions are:

- We train and benchmark two existing state-of-the-art AR glasses pose estimators on generated head pose labels for further head and AR glasses comparison, one being a point cloud-based CNN called P2P [8] and the other an image-based CNN named GlassPoseRN [6].
- We introduce a pose estimation network called CapsPose relying on Capsules. To the best of our knowledge, Capsule Networks haven't been used for 6-DoF pose estimation before. We outperform the current state-of-art method GlassPoseRN on the HMDPose dataset by reducing the error by 46% for orientation and 51% for position.
- We show that AR glasses pose estimation exceeds head pose estimation performance in most cases. Pose estimation of AR glasses outperforms the head if the head and glasses are both known, or the glasses are known, but the head is unknown to the trained network. The head pose is only more accurate if the head is known, but the AR glasses are unknown to the networks.

In the remainder of the paper, we present our head pose computation pipeline based on the HMDPose dataset and the Head and AR glasses pose estimation algorithms in Section 2. In Section 3, we discuss

*e-mail: Ahmet.Firintepe@bmwgroup.com
†e-mail: Oussema.Dhaouadi@bmwgroup.com
‡e-mail: Alain.Pagani@dfki.de
§e-mail: Didier.Stricker@dfki.de

our evaluation on the HMDPose dataset, comparing the Head and AR glasses pose performance of the methods.

2 Methods

2.1 AR Glasses Dataset

We use the HMDPose dataset [5] to generate our head pose label to train and test the two existing methods. In addition, we train and evaluate our CapsPose algorithm with the AR glasses labels given by the HMDPose dataset and head pose labels we generate. HMDPose is a large-scale data glasses dataset, containing 3 million 1280×752 pixel images. It consists of IR images from 3 different perspectives of 4 different AR glasses, worn by 14 different subjects.

2.2 Head Pose Label Generation

Since the head pose labels are not included in the HMDPose dataset, we propose a pipeline for measuring head orientation and position using the triple images of HMDPose to further train head pose estimation methods. The extrinsic and intrinsic parameters of the cameras are known. In the head pose annotation pipeline, we extract facial landmarks and compute their 3D positions in the reference frame. Subsequently, we define the head coordinate system. Finally, we post-process the measured head poses.

2.2.1 Facial Landmarks Extraction

In this step, we first detect the head and localize its 2D bounding box in all cameras. We use the BlazeFace model [1], which extracts image features. Then, a regressor is used to obtain the final bounding box coordinates. After cropping the head using the predicted bounding box, we use the Face Alignment Network (FAN) [4] to extract 69 facial landmarks in total (Figure 1).

(a) Left View (b) Middle View (c) Right View

Figure 1: Visualization of the extracted facial landmarks. The blue box depicts the predicted bounding box by the head detector.

2.2.2 Landmarks Reconstruction

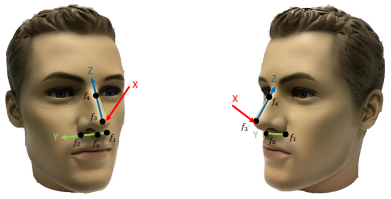
After extracting a set of facial landmarks per image in the image triples, we determine the coordinates of 2D landmarks in the 3D space. As the projection of a 3D point onto the pixel plane P_i of the camera i is non-linear, we solve the triangulation problem using the Bundle Adjustment (BA) technique. We re-project the estimated 3D points onto the pixel coordinates of all cameras and minimize the re-projection error. A Levenberg–Marquardt non-linear optimizer [12] estimates the 3D point $\boldsymbol{P_i}$ by minimizing a non-linear cost function:

$$\hat{\boldsymbol{P}}_i = \underset{\boldsymbol{P}_i}{\operatorname{argmin}} \sum_{k=1}^{3} \left\| \boldsymbol{p}_i^k - \boldsymbol{f}_k(\boldsymbol{P}_i) \right\|_2^2, \tag{1}$$

$\boldsymbol{p}_i^k$ are the 2D coordinates of the landmark i in the pixel frame k and $\boldsymbol{f}_k$ is the non-linear projection of 3D points onto the pixel frame k. We stack all N landmark points and all 3 cameras into a single vector to perform a joint optimization, where the re-projection errors are minimized in all pixel frames. The bundled error vector norm is minimized by a non-linear least square estimation. The solver iteratively obtains the optimal 3D facial landmarks.

2.2.3 Head Pose Computation

We define the head coordinate system center and its orthonormal basis based on the 3D facial landmarks as shown in Figure 2.



(a) Side View (b) Front View

Figure 2: Definition of the head coordinate system shown from two different views.

We use points of a rigid region of the head. Inspired by Breitenstein et al. [3], we choose the nose as our region of interest. We define the subnasale key-point f_0 as the center of the head. We define the Y-axis as the normalized vector pointing from the left alare f_1 to the right alare f_2 feature. The Z-axis is defined by the normalized vector linking the nose tip f_3 and the mid-endocanthion point f_4 of the face. The X-axis is given by the cross product of the two other axes.

2.2.4 Post Processing

We eliminate outliers using the Median Absolute Deviation based outlier detector [10] by computing the MAD scores for every single pose in all pose dimensions. We replace anomalies by averaging the pose of the previous and next frame. Subsequently, we use a first order Butterworth filter to smooth the poses. We give more insight on the rotation values of the HMDPose dataset and our HeadPose annotations in the supplementary material.

2.3 Data Split Strategies

The HMDPose and HeadPose datasets are acquired from various subjects wearing different glasses yielding a large inter-class variation. Therefore, we define different types of data splits (Figure 3).

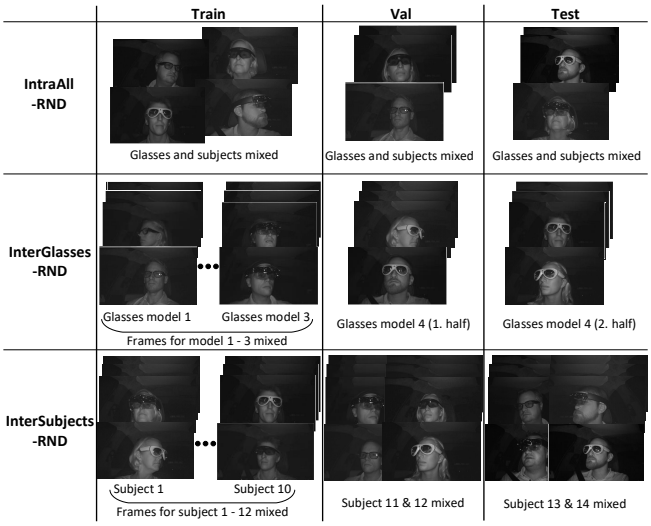


Figure 3: An overview of the different split strategies IntraAll-RND, InterGlasses-RND and InterSubjects-RND.

IntraALL-RND shuffles all acquired frames and randomly selects portions of 94%, 3% and 3% for training, validation and testing sets. In that strategy, all types of glasses and all subjects are present in all sets, showing the general pose prediction performance of a NN when trained on this split. This split has been used by other publications on the HMDPose dataset [6, 8].

InterGlasses-RND uses three types of glasses for the training set and one type of glasses for the validation and testing sets. This split is interesting to test the ability of NNs in generalizing glasses. This

type of evaluation is needed when a new type of glasses is used, on which the NN was not trained.

InterSubjects-RND split technique is similar to InterGlasses-RND, where the split is according to the class of subjects. Only 10 subjects are included in the training set and two other different subjects for each of the validation and test sets. The evaluation results show the performance of the NNs in generalizing the subjects, and thus, its performance on an arbitrary person.

2.4 Feed-Forward Pose Estimation Architectures

We benchmark two existing networks and our new network called CapsPose on the HMDPose and HeadPose datasets to evaluate the difference between glasses and head pose estimation. This results in six models in total. The networks are called "GlassPoseRN-HMD", "GlassPoseRN-HEAD", "P2P-HMD", "P2P-HEAD", "CapsPose-HMD" and "CapsPose-Head". The additions "-HMD" and "-HEAD" refer to the used dataset, being either the HMDPose or our generated HeadPose based on the HMDPose dataset.

2.4.1 GlassPoseRN-HMD & GlassPoseRN-HEAD

Firintepe et al. presented in [7] an approach for glasses pose estimation called GlassPoseRN, which is based on a Residual Network. In this work, we call this network *GlassPoseRN-HMD*, while we name the retrained version on the head pose *GlassPoseRN-HEAD*.

Both networks use ResNet-18 [9] as backbone for feature extraction. The backbone is followed by three fully connected layers with 256, 64 and 7 neurons, respectively. While rotations are represented by 4D quaternions, translations are represented by Euclidean 3D positions of the AR glasses.

2.4.2 P2P-HMD & P2P-HEAD

We additionally utilize the Points to Pose (P2P) estimation approach benchmarked on the HMDPose dataset [8]. A point cloud estimator first generates point clouds from the triple infrared images. A PointNet++ [14] backbone sub-samples the full point cloud and learns point cloud features. The selected points are called seed points, where each point provides a vote for predefined key points. A proposal module aggregates the key points to regress the pose.
For the sake of comparison, we select cloud points of either the HMD or the head. The point selection is a deterministic process that follows predefined steps. We assume that the represented target point cloud $PC_T = \{{}^1\mathbf{c}^T, \ldots, {}^N\mathbf{c}^T\}$ is the result of an Euclidean transformation $\boldsymbol{T}_S^T$ embodying the rotation $\mathbf{R}$ and the translation $\mathbf{t}$ applied on a source point cloud $PC_S = \{{}^1\mathbf{c}^S, \ldots, {}^N\mathbf{c}^S\}$.

A source point cloud PC_S represents a centered head wearing the glasses. Hence, the translation $\mathbf{t}$ is given by the mean of the target point cloud. By subtracting the measured translation $\mathbf{t}$ from the target point cloud and multiplying the result by the inverse rotation $\mathbf{R}^T$ obtained from the HMDPose dataset, we get the source point cloud. We divide the cloud points into points of the head and points of the glasses. The first 20% of the source point cloud starting by counting from the top are elements of the head. We label the next 450 points elements of the glasses. The rest of the points are assigned to the head class, which results in 1550 head points. Then, we back-transform the selected point clouds to the target space. As a consequence, we obtain two disjoint point cloud datasets PC^{HMD} and PC^{Head} respectively corresponding to the glasses and the head, where $PC^T = PC^{HMD} \cup PC^{Head}$.

We adapt the network *P2P* proposed in [8] by changing the input shape. We call the original model P2P-F-HMD as the network using all points of the cloud for glasses pose estimation. The corresponding model for head pose estimation is called P2P-F-HEAD. We denote the models that use a subset of the cloud by P2P-P-HMD and P2P-P-HEAD for glasses and head pose estimation, respectively.

2.5 Capsule Pose Network

2.5.1 Capsule Networks

Despite their success in various domains, CNNs suffer from certain limitations. One key property of CNNs is the translation invariance, which is achieved by dropping the location information, negatively affecting the generalization performance of pose estimators. Capsule Networks (CapsNet) [16] were designed to address the weaknesses of traditional CNNs. The elemental entities of CapsNet are called Capsules, which represent a small group of neurons that encapsulate their outputs in an activity vector. This vector represents the instantiation parameters of specific entity types, such as objects or object parts. Instantiation parameters include properties such as pose or texture. The entity's existence is either learned as an instantiation parameter or encoded by the magnitude of the activity vector as done in CapsNet. A visual system should learn the existence of an object based on its relevant features and the relationship between them and the high-level features. The association between the simple entity represented by a low-level Capsule, the complex entity represented by a high-level Capsule, and the instantiation parameters of the Capsules are obtained through the routing algorithm. CapsNet [16] consists of a convolutional layer used to extract very low-level features. Multiple convolutional operators are used in parallel to construct the first Capsule layer named Primary Capsule layer (PrimCaps), followed by the Digit Capsule layer (DigitCaps) consisting of 10 Digit Capsules, each representing the existence of a single digit. The digit existence probability is encoded in the magnitude of the activity vector of the Digit Capsule.

We use the CapsNet architecture as a baseline for our network called Capsule Pose (CapsPose), performing 6-DoF pose estimation. Our network is the first to solve the 6-DoF pose estimation problem using Capsules to the best of our knowledge.

2.5.2 CapsPose Network Architecture

CapsPose consists of 4 layers: two convolutional layers *ConvReLU* and *ConvReLU2*, *PrimaryCaps* and an output Capsule layer *PoseCaps*. Figure 4 shows an overview of the CapsPose architecture.

ConvReLU layer detects basic features in the input. It inputs down-sampled triple images in form of inputs tensor of size $320 \times 188 \times 3$ pixels. The given pixel data is converted to the activity of local feature detectors, extracting low-level feature maps. The layer *ConvReLU* generates 64 feature maps over 32×32 filters with a stride of 8×8 and no padding, containing of a *ReLU* activation. The second layer is also a convolutional layer with 64 filters of size 8×16 and a unit stride. The third layer is a convolutional Capsule layer, which can be viewed as a convolutional layer with 256 filters of size 16×16, 8×8 stride and no padding, including a squashing activation function [16]. Rearranging the result produces 96 8-dimensional primary Capsules. The next layer is the PoseCaps layer, consisting of 7 Capsules including 16 instantiation parameters. Each pose Capsule receives the 8-D input of all 96 primary Capsules. Each of these input vectors receives its own 8×16 weight matrix, which maps the 8-D input space to the 16-D Capsule output space. The routing-by-agreement takes place only between the *PrimaryCaps* and *PoseCaps* layers. The activity vector of the corresponding Capsules describe the local position of these features in the PrimaryCaps. The dynamic routing and thus, the routing-by-agreement between the PrimaryCaps and PoseCaps defines a trainable transformation matrix. It transforms the orientation and position of the features from their local coordinate systems to the coordinate system of the higher-level pose Capsule. According to the coupling coefficients, its activity vector is computed. It embodies the pose of

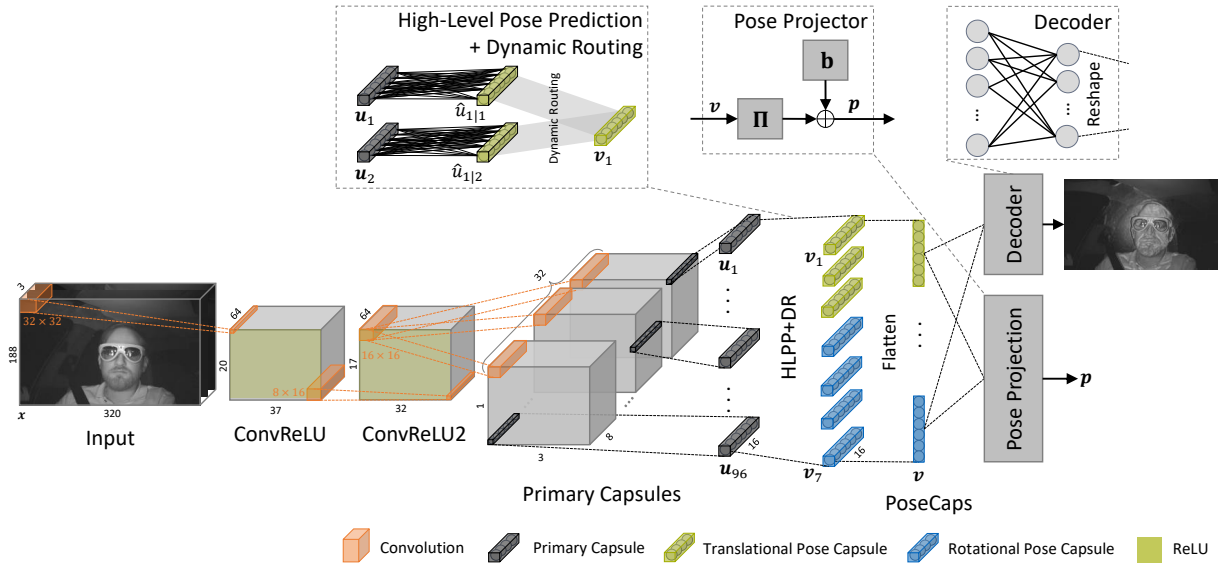


Figure 4: Our CapsPose architecture. The ConvReLU layer extracts low-level features. The parallel convolutional layers form the primary Capsules represent low-level entities. A high-level pose prediction step and the Dynamic Routing algorithm transmit the information to the pose Capsules. A pose projector projects the activity vector of pose Capsules onto the pose space. A Decoder reconstructs one image.

the target object represented in its own Capsule space.
We add a projector module, which maps elements from the instantiation parameter space of the Capsules to the 7-dimensional pose space. Three values represent the position Euclidean space, while four values regress the quaternion for the orientation. The following equation describes the projection operation:

$$\boldsymbol{p} = \boldsymbol{\Pi}\boldsymbol{v} + \boldsymbol{b}, \tag{2}$$

where $\mathbf{v}$ is a vector stacking the activity vectors of all Capsules $\mathbf{v}_j$ with $j = 1, \ldots, 7$. The pose vector $\boldsymbol{p}$ contains predicted pose values consisting of the translational and rotational poses.
To enhance the feature learning, we build a decoding module, which reproduces one input image given the values of all activity vectors stacked in $\mathbf{v}$. The vector $\mathbf{v}$ encapsulates the information for the pose prediction and reconstruction. This addition is inspired by [16]. The decoder consists of two fully connected *ReLU* layers with 512 and 1024 neurons and one fully connected *Sigmoid* output layer.

We use the Adam optimizer with the learning rate $\alpha = 0.0001$ to train the CapsPose networks for 180 epochs. We train the network using the following loss function introduced by Kedall et al. [11]:

$$Loss = \|\boldsymbol{t} - \hat{\boldsymbol{t}}\|_2 + \left\| \boldsymbol{q} - \frac{\hat{\boldsymbol{q}}}{\|\hat{\boldsymbol{q}}\|_2} \right\|_2. \tag{3}$$

$\boldsymbol{q}$ is the ground truth label, while $\hat{\boldsymbol{q}}$ is the estimated quaternion. Equally, $\boldsymbol{t}$ and $\hat{\boldsymbol{t}}$ define the translation. The predicted quaternion is normalized and the Euclidean distance to the ground truth quaternion is computed. Regression of unit quaternions on the positive w scale ensures unambiguous estimations for the orientation.

3 Evaluation

3.1 Evaluation metrics

To compare the head and AR glasses models of the GlassPoseRN and P2P methods as well as our CapsPose algorithm, we use the metrics as already used on previous HMDPose benchmarks [6, 8]. Thus, we use the Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Balanced Mean Angular Error (BMAE). The BMAE introduces the definition of sections to consider the unbalanced amount of different head orientations [7, 17, 18]:

$$BMAE := \frac{d}{k} \sum_{i=1} \phi_{i,i+d}, i \in d\mathbb{N} \cap [0,k], \tag{4}$$

The metric divides the range of movement k into sections i with sizes d. By this, extreme and rare poses are being weighted equally to frequent and regular poses. $\phi_{i,i+d}$ is defined as the average angular error. The section size d is set to 5 degrees and the range size k to 180 degrees as done in previous benchmarks. For the position estimation evaluation, we use the MAE for the individual axes and the L_2 error for the axes combined for the position error.

3.2 Results

To compare the head and glasses pose estimation performance, we first investigate the estimation accuracy of networks trained on the HMD and head labels. For the sake of fairness, the same input frames are fed to the head and glasses pose estimators in the training and the evaluation phases. Moreover, we evaluate a network using the splits on samples of a seen driver and seen glasses named InterSubjects-RND and InterGlasses-RND to investigate the generalization performance regarding the driver and the glasses.

3.2.1 Pose Estimation Analysis

Table 1 summarizes the models' orientation estimation results trained on HMD and head labels, respectively. Table 2 2 illustrates the position estimation results.

GlassPoseRN-HMD improves over GlassPoseRN-HEAD by reducing the rotational MAE of up to 65% and the rotational RMSE of up to 55% (Table 1). The translational error also improves by up to 48%. There are multiple explanations for this. The pixels brightness of the glasses in an IR image is significantly higher than that of the head. This facilitates the search of the target object by the network. Furthermore, the shape of the glasses is not uniform. Many edges could be extracted from the glasses, which may improve the quality of the extracted feature map. Moreover, glasses have a roughly rectangular form, and extracting the orientation is thus simpler. On the contrary, the head is oval-shaped, making estimating its pose based on 2D images challenging. Therefore, the network relies on the head's general form and other parts such as the nose and the mouth. The BMAE of GlassPoseRN-HMD is lower than that of

Model	MAE				RMSE				BMAE			
	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg
GlassPoseRN-HMD [6]	**0.06**	**0.10**	**0.14**	**0.10**	**0.09**	**0.25**	**0.22**	**0.19**	**0.29**	**0.12**	**0.31**	**0.24**
GlassPoseRN-HEAD	0.20	0.20	0.47	0.29	0.28	0.29	0.73	0.43	0.71	0.43	0.50	0.55
P2P-P-HMD	**0.39**	**0.43**	0.75	**0.52**	**0.55**	**0.57**	1.09	**0.74**	**0.72**	**0.55**	2.14	**1.14**
P2P-P-HEAD	0.44	0.51	1.58	0.84	0.60	0.67	2.14	1.14	1.07	1.27	**1.80**	1.38
P2P-F-HMD [8]	0.53	0.47	**0.61**	0.53	0.73	0.68	**0.88**	0.76	1.81	0.56	**1.26**	1.21
P2P-F-HEAD	2.70	5.32	4.93	4.31	3.66	6.54	6.53	5.58	5.42	8.46	7.48	7.12
CapsPose-HMD (ours)	**0.05**	**0.09**	**0.12**	**0.09**	**0.08**	0.24	**0.21**	**0.18**	**0.09**	**0.09**	**0.21**	**0.13**
CapsPose-HEAD (ours)	0.22	0.23	0.54	0.33	0.22	**0.23**	0.54	0.33	0.35	0.28	0.50	0.38

Table 1: Orientation results of all models for head and glasses pose estimation trained and evaluated on HMD and head labels, respectively. Feed-forward models follow the IntraALL-RND data split. The evaluation results are conducted in the Euler space and expressed in degrees.

Model	MAE			L_2
	X	Y	Z	
GlassPoseRN-HMD [6]	**0.49**	**0.50**	**0.44**	**0.90**
GlassPoseRN-HEAD	0.86	1.12	0.57	1.74
P2P-P-HMD	10.98	6.55	17.99	25.72
P2P-P-HEAD	**4.21**	**4.26**	**4.90**	**9.01**
P2P-F-HMD [8]	**3.24**	**2.39**	**2.96**	**5.75**
P2P-F-HEAD	33.39	31.02	65.87	89.02
CapsPose-HMD (ours)	**0.25**	**0.26**	**0.13**	**0.44**
CapsPose-HEAD (ours)	0.68	0.99	0.34	1.40

Table 2: Position results of all models for head and glasses pose estimation trained and evaluated on the HMD and head labels, respectively. Feed-forward models follow the IntraALL-RND data split. The evaluation results are expressed in millimeters.

GlassPoseRN-HEAD. GlassPoseRN-HMD produces better translation estimation performance compared to GlassPoseRN-HEAD with an error reduction of 48%. We especially observe a high error of the GlassPoseRN-HEAD on the Y-axis.

The P2P results for head and glasses pose estimation depend on the type of point clouds fed to the network. When using partial point clouds, either including the points of the head or the glasses, P2P-P-HEAD reduces the translation error by around 65% over P2P-P-HMD. Meanwhile, the rotation performance of the head pose estimator is worse than the glasses pose estimator. One reason for this inconsistency is the use of fewer points for the glasses pose estimation. These few points of the glasses are enough to result in lower rotation error. When using the whole point cloud, the findings align with the results reported by the previous methods. On the one hand, P2P-F-HEAD shows a low translation performance, which may be related to the points on the face being very similar. On the other hand, P2P-F-HMD reduces the translation error by 93% compared to P2P-F-HEAD, as the few points of the glasses are distinguishable from the rest of the cloud.

We make a similar observation evaluating for CapsPose. The CapsPose-HMD rotation and translation performance improve compared to CapsPose-HEAD with an error reduction of up to 73% and 69%, respectively. The MAE and the BMAE for yaw of CapsPose-HEAD are high and close, showing low performance for yaw for extreme and normal poses. The drop of performance is only noticeable for CapsPose-HMD in the case of extreme poses.

Generally, the glasses pose estimation methods outperform the head pose estimation methods in the case of known glasses and drivers.

When comparing glasses pose estimators trained on all types of glasses, we observe worse performance of P2P-P-HMD and P2P-F-HMD models than the other models. Furthermore, the reason for the high RMSE may be caused by outliers. The high MAE error shows that predictions are not close to the ground truth, which may cause unstable and non-smooth predictions. Also, the effectiveness of CapsPose-HMD is clear. It outperforms all other methods and reduces the GlassPoseRN-HMD errors by up to 46% and 51% for rotation and translation, respectively. The BMAE along the roll axis is also considerably reduced by 68%. As the roll range in the dataset is small, it is a notable advantage to use this model. This finding tallies with our expectations that Capsules are better in capturing variations in a relatively narrow range. The same pattern was observed for the head pose estimators when comparing the models. Overall, our CapsPose method has demonstrated a marked improvement in the quality of pose estimation.

3.2.2 Generalization Analysis

We use the InterSubjects-RND data split strategy to inspect the driver generalization performance, as it includes two unseen subjects in the validation set and two persons in the test set. This gives us an insight into the network performances in estimating the pose of the glasses and the head of an unknown person. We further analyze the glasses generalization performance. Tables 3 and 4 list the results of CapsPose trained for head and glasses pose estimation using InterSubjects-RND and InterGlasses-RND data split strategies.

On the InterSubjects-RND split, CapsPose-HMD outperforms CapsPose-HEAD for the orientation estimation with an error reduction of 56%. Although the glasses are known, the position estimation performance of CapsPose-HMD is still low. This indicates that the network relies on some head features for the glasses position estimation. Also, CapsPose-HEAD shows high errors as the network regresses the pose from unseen heads. CapsPose-HEAD predictions seem to be unstable, as shown by the RMSE. Relatively high errors are also observed for extreme poses compared to CapsPose-HMD. This is caused by the network strongly relying on head features for the head pose estimation. CapsPose-HMD and CapsPose-HEAD result in comparable performance for head and glasses translation regression i.e. the networks use a similar pool of features mainly consisting of head features due to the observed high errors.

Concerning the InterGlasses-RND split, CapsPose-HMD and CapsPose-HEAD show a difference in the orientation performance. CapsPose-HMD produces less accurate results since the network uses unseen features of the glasses to predict the pose. CapsPose-HEAD also shows relatively high errors, although the driver's head is known. This indicates that the head pose estimator relies on the head features and some glasses features. However, CapsPose-HMD strongly uses glasses features and some head features, yielding acceptable rotation results. As the glasses are unknown, the RMSE demonstrates unstable predictions, especially for extreme poses, as proven by the high BMAE. In total, CapsPose-HEAD is more unstable in general and more stable in extreme poses than CapsPose-HMD. The translation errors drop for head pose estimation by 74%.

4 Conclusion

In this paper, we compared AR glasses pose and head pose estimation performance based on the same dataset and algorithms. We

Data Split	InterSubjects-RND												InterGlasses-RND											
Model	MAE				RMSE				BMAE				MAE				RMSE				BMAE			
	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg	Roll	Pitch	Yaw	Avg
CapsPose-HMD	**1.76**	**2.70**	**3.38**	**2.61**	**1.76**	**2.70**	**3.38**	**2.61**	**2.72**	**2.85**	**4.16**	**3.24**	**1.60**	4.66	**3.41**	3.22	**2.11**	5.28	**4.02**	**3.80**	3.43	6.53	**3.77**	4.58
CapsPose-HEAD	2.69	8.93	6.41	6.01	3.42	9.94	8.88	7.41	3.22	7.85	8.53	6.53	2.14	**3.02**	3.67	**2.94**	2.75	**4.21**	4.82	3.93	**2.59**	**4.17**	4.36	**3.70**

Table 3: Orientation results of CapsPose model for head and glasses pose estimation trained on HMD and Head labels, respectively, following the InterSubjects-RND and InterGlasses-RND data splits. The evaluation results are conducted in the Euler space and expressed in degrees.

Data Split	InterSubjects-RND				InterGlasses-RND			
Model	X	Y	Z	L_2	X	Y	Z	L_2
CapsPose-HMD	10.93	9.14	**3.69**	**16.88**	15.03	37.54	8.59	42.75
CapsPose-HEAD	**3.22**	**7.85**	8.53	16.94	**7.16**	**5.33**	**3.56**	**10.89**

Table 4: Position results of CapsPose for head and glasses pose estimation trained on HMD and head labels, respectively, following the InterSubjects-RND and InterGlasses-RND data splits. The evaluation results are conducted in the Euclidean space and expressed in millimeters.

presented a pipeline for head pose annotation generation using the triple images given in the HMDPose dataset. Then, we benchmarked pose estimation approaches like the state-of-art GlassPoseRN and P2P networks, as well as our novel CapsPose algorithm on head pose labels. In a comparison with three analysis levels, we showed that estimating the HMD pose is generally more accurate than the head pose. The first analysis showed the general regression performance of the models when AR glasses and faces are both known to the networks during training, in which case HMD pose estimation performs better than head pose estimation. In a second analysis for testing the driver generalization, AR glasses pose estimation exceeded head pose estimation. The driver was unknown, but AR glasses were known to the networks. The third analysis focused on the glasses' generalization ability. Only in this case, the head pose estimation performance is better than the HMD pose. Therefore, AR glasses pose estimation is more accurate but can be supported by head pose estimation. One case would be introducing new AR glasses in a pose estimation system when the networks have not been trained with the new glasses model yet. In this case, directly estimating the head pose and combining the poses can be beneficial. Finally, our novel pose estimation network called CapsPose is the first network to deploy Capsule Networks for 6-DoF pose estimation. It outperforms the state-of-the-art method GlassPoseRN on the HMDPose dataset by reducing the error by 46% for orientation and 51% for translation. Future work will consist of occlusion analysis, where the HMDs or heads are occluded to further analyze their differences for more difficult cases.

REFERENCES

[1] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. 2019.

[2] G. Borghi, M. Venturelli, R. Vezzani, and R. Cucchiara. POSEidon: Face-From-Depth for Driver Pose Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4661–4670, July 2017.

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

[4] 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 *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1021–1030, 2017.

[5] A. Firintepe, A. Pagani, and D. Stricker. HMDPose: A large-scale trinocular IR Augmented Reality Glasses Pose Dataset. In *26th ACM Symposium on Virtual Reality Software and Technology*. ACM, 2020.

[6] A. Firintepe, A. Pagani, and D. Stricker. A Comparison of Single and Multi-View IR image-based AR Glasses Pose Estimation Approaches. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 571–572, 2021.

[7] A. Firintepe, M. Selim, A. Pagani, and D. Stricker. The More, the Merrier? A Study on In-Car IR-based Head Pose Estimation. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1060–1065. IEEE, 2020.

[8] A. Firintepe, C. Vey, S. Asteriadis, A. Pagani, and D. Stricker. From IR Images to Point Clouds to Pose: Point Cloud-Based AR Glasses Pose Estimation. *Journal of Imaging*, 7(5), 2021.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016.

[10] B. Iglewicz and D. C. Hoaglin. *How to detect and handle outliers*, vol. 16. Asq Press, 1993.

[11] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2938–2946, 2015.

[12] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[13] M. Patacchiola and A. Cangelosi. Head pose estimation in the wild using convolutional neural networks and adaptive gradient methods. *Pattern Recognition*, 71:132–143, 2017.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 5105–5114. Curran Associates Inc., Red Hook, NY, USA, 2017.

[15] N. Ruiz, E. Chong, and J. M. Rehg. Fine-grained head pose estimation without keypoints. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 2074–2083, 2018.

[16] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic Routing between Capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 3859–3869. Curran Associates Inc., Red Hook, NY, USA, 2017.

[17] A. Schwarz, M. Haurilet, M. Martinez, and R. Stiefelhagen. Driveahead-a large-scale driver head pose dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–10, 2017.

[18] M. Selim, A. Firintepe, A. Pagani, and D. Stricker. AutoPOSE: Large-scale Automotive Driver Head Pose and Gaze Dataset with Deep Head Orientation Baseline. In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP,*, pp. 599–606, 2020.

[19] G. Zhang, J. Liu, H. Li, Y. Q. Chen, and L. S. Davis. Joint Human Detection and Head Pose Estimation via Multistream Networks for RGB-D Videos. *IEEE Signal Processing Letters*, 24(11):1666–1670, 2017.