Closed-Eye Gaze Gestures: Detection and Recognition of Closed-Eye Movements with Cameras in Smart Glasses

Rainhard Dieter Findling[0000−0002−6985−1298], Le Ngu Nguyen[0000−0001−7765−1483], and Stephan Sigg[0000−0001−6118−3355]

Ambient Intelligence Group, Department of Communications and Networking, Aalto University, Maarintie 8, 02150 Espoo, Finland
rainhard.findling@aalto.fi
http://ambientintelligence.aalto.fi/

Abstract. Gaze gestures bear potential for user input with mobile devices, especially smart glasses, due to being always available and hands-free. So far, gaze gesture recognition approaches have utilized open-eye movements only and disregarded closed-eye movements. This paper is a first investigation of the feasibility of detecting and recognizing closed-eye gaze gestures from close-up optical sources, e.g. eye-facing cameras embedded in smart glasses. We propose four different closed-eye gaze gesture protocols, which extend the alphabet of existing open-eye gaze gesture approaches. We further propose a methodology for detecting and extracting the corresponding closed-eye movements with full optical flow, time series processing, and machine learning. In the evaluation of the four protocols we find closed-eye gaze gestures to be detected 82.8%-91.6% of the time, and extracted gestures to be recognized correctly with an accuracy of 92.9%-99.2%.

Keywords: Closed eyes · Gaze gestures · Machine learning · Mobile computing · Recognition · Smart glasses · Time series analysis.

1 Introduction

Users interact with their mobile devices frequently and throughout their daily routines. Smart phones and tablets have an average of 60 and 23 interactions per day, respectively, with a total usage duration of on average 221 minutes per day [7]. For this reason, user input across mobile devices should be fast, easy, reliable, and convenient. Gaze gestures have been demonstrated feasible for input to mobile devices [4]. They bear potential as being conceptually both hands free and allow to perform quick input. However, gaze gestures with smart phones and tablets are usually done when users are holding them/looking at them. In contrast, smart glasses with embedded eye-facing cameras allow hands-free gaze gesture recognition, which does not require additional preparation time (i.e. taking devices out of a trousers pocket) or users to look at a device screen.

So far, mobile gaze gesture sensing has utilized movements from opened eyes only and disregarded closed eyes. With gaze gesture alphabets in general being limited (e.g. 4 to 8 easily performable gaze gestures [4, 6]), sensing and recognizing also closed-eye gaze gestures would allow for an extended alphabet that combines movements of both opened and closed eyes. Eyes being open or closed could thereby be distinguished by the presence or absence of pupils in recordings. The major challenge with recognizing closed-eye gaze gestures from optical sources is that prior work has mostly utilized pupil movements, which renders them inapplicable for detecting closed-eye movements. In this paper we therefore investigate whether detection and recognition of closed-eye gaze gestures from close-up optical sensors, e.g. from eye-facing cameras embedded in smart glasses, is feasible. Our contributions are:

- We propose a processing methodology to detect and recognize closed-eye gaze gestures from recordings of cameras embedded in smart glasses.
- We propose three basic closed-eye gaze gesture protocols for smart glasses, which contain different sets of gaze gestures, and which all effectively extend existing open-eye gaze gesture alphabets with closed-eye movements.
- We comparatively evaluate the proposed protocols for their gesture detection and confusion rates using different machine learning approaches.

2 Related Work

While there has been numerous prior work on open-eye gaze gestures (cf. [6]), closed-eye gaze gestures have so far not been investigated. EyeWrite [13] uses gaze gesture input for text composition. The concept is based on EdgeWrite [12], in which each character is replaced by a gesture. The alphabet therefore contains 26+10 gestures for numeric characters and further gestures for punctuation and text control. The approach is designed to work with a stationary Tobii ET-1750 eye tracker in the form factor of a computer monitor. A set of 12 gaze gestures is used in [10]. Each gaze gesture in their alphabet consists of left, right, up, down, and optional diagonal movements. They evaluate their approach with a PC setup, and compare it to dwell-based gaze input in subsequent work [8]. In [5] an alphabet of 8 gaze gestures is used. Each gesture is an unidirectional stroke into a certain direction (left, right, up, down, and the four diagonals in between). The authors utilize a setup with a computer monitor and a camera with attached infrared light to perform pupil tracking and extract gaze gestures. The same alphabet is used in [4] for gaze gesture input to mobile devices. All those approaches have in common that they rely on optical pupil tracking, hence on opened eyes, for gaze gesture extraction. While cameras can be built into smart glasses frames [11], pupils are not visible with closed eyes, which prevents those approaches from being applied to closed-eye gaze gestures.

Electrooculography (EOG) recognition, while technically different in the sensing, is conceptually able to overcome this limitation, as it does not rely on pupil tracking. However, employing EOG gaze gestures with closed eyes too has

not been investigated yet. Related work on EOG based gaze gestures with opened eyes [2] used a basic alphabet related to the one in [5], consisting of 8 gestures (left, right, up, down, and four diagonals). In subsequent work they expand user input to have either small or big eye movements in left, right, up, and down [1]. By combining two movements they thereby encode a total of 16 different characters for user input. When considering smart glasses, EOG sensors bring a significant drawback. While EOG should in general allow for sensing closed-eye movements, embedding those sensors into smart glasses impacts the usability of the devices as glasses. Firstly, in contrast optical sensors, EOG sensors need to touch the skin, which limits possible sensor positions to around the nose and close to the ears (similar e.g. to the Jins Meme device [9]). Those locations cause sensors to be positioned horizontally, but not vertically. While this allows for horizontal eye movement sensing, vertical eye movement sensing would require additional sensors above and below the eye (similar to the goggles utilized in [3]). Adding such sensor positions would make the device significantly more cumbersome due to increased size and weight. Secondly, for good EOG signal quality, sensors need to be connected well to the skin. This would require either wet electrodes, arguably reducing comfort and usability, or firm skin contact. The latter seems possible when utilizing the weight of common glasses, but only on positions where glasses abut on skin (nose, ears), which again disregards vertical sensing.

To summarize, while there is prior work on optical sensing for gaze gestures, the employed approaches are not applicable with closed eyes due to pupils being hidden. EOG sensors should conceptually allow for sensing closed-eye movements, but embedding them in smart glasses seems cumbersome. It would either limit vertical sensing capabilities or cause the device to become goggles instead of glasses, with drawbacks in size and weight. This paper addresses the remaining gap: it provides a methodology to detect and recognize closed-eye gaze gestures with close-up cameras, and thereby is a first step towards utilizing both open- and closed-eye gaze gestures with smart glasses, thereby allowing and extension of the restricted alphabet of gaze-interfaces.

3 Our Approach to Closed-Eye Gaze Gestures

Our approach enables recognition of closed-eye gaze gestures from cameras embedded in smart glasses. In this section we propose four closed-eye gaze gesture protocols, together with a technical methodology to detect and extract those from sensed data. From a technical perspective, our approach relies on machine learning for recognizing optically sensed closed-eye gaze gestures. Our recognition consists of a training part to enroll individual users, and a recognition part to utilize users' eye movements as input. Both training and recognition internally perform optical closed-eye gaze gesture detection and extraction from video data (Fig. 1). The details for the individual steps in those parts are discussed in the sections below.

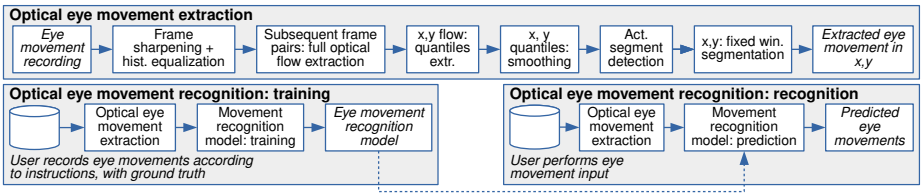
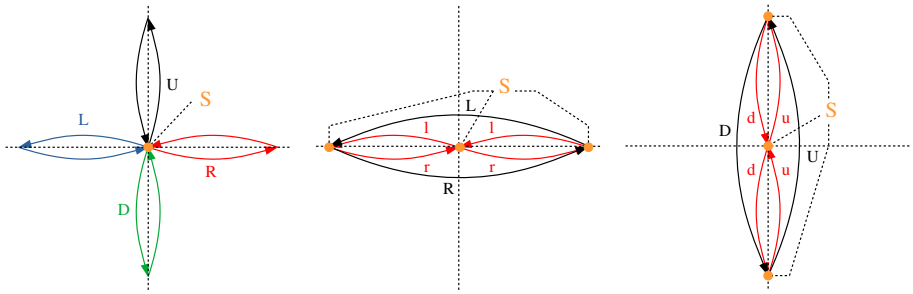


Fig. 1. Overview of processing in our approach to recognize closed-eye gaze gestures.



(a) LRUDS (b) horizontal (c) vertical

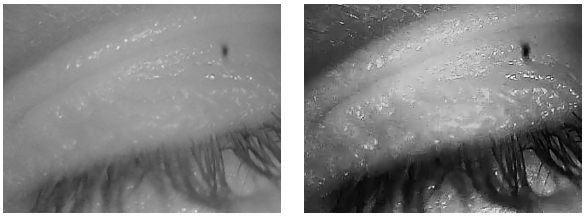
Fig. 2. Graphical depiction of possible gaze gestures with different protocols. While (a) the LRUDS protocol contains bidirectional gaze gestures only, all other protocols contain a set of small (e.g. "r") and/or big (e.g. "R") unidirectional gaze gestures, in both (b) horizontal and/or (c) vertical direction.

3.1 Closed-Eye Gaze Gesture Protocols

We propose 4 closed-eye gaze gesture protocols, which are related to protocols from prior work on open-eye gaze gestures (cf. [4, 5, 8, 10]). With all protocols, users look straight on and close their eyes to begin user input, and they open their eyes to end user input. Each user input can contain multiple individual gaze gestures (Fig. 2). The alphabet of possible gaze gestures is defined by the respective protocol.

The alphabet of the LRUDS protocol contains a total of 5 possible gaze gestures: 4 horizontal or vertical bidirectional gaze gestures (LRUD) and a squint movement (S). All gaze gestures start and end in the center. The bidirectional gaze gestures go either left, right, up, or down, and backwards in the same gesture. The squint movement results from shortly squinting eyes. It therefore is equal to blinking except that users do not open their eyes during the process.

The alphabet of the LlRrUuDdS protocol contains a total of 9 possible gaze gestures: 8 horizontal or vertical unidirectional gaze gestures (LlRrUuDd) and a squint movement (S). The gaze gestures are either small (e.g. "r") or big (e.g. "R") eye movements, and they do not go back to the start of the gesture after that movement. Hence, they do not need to start or end in the center.



(a) Unprocessed (b) Preprocessed

Fig. 3. Sample frame of a recording of a user performing a closed-eye gaze gesture with (a) the unprocessed and (b) the preprocessed frame. The black spot in the upper right is a dust particle on the camera lens, which does not negatively effect our approach.

The alphabet of the lrudS protocol contains a total of 5 possible gaze gestures: 4 horizontal or vertical unidirectional gaze gestures (lrud) and a squint movement (S). This protocols only allows for small eye movements, and gaze gestures do not go back to the start of the gesture after that movement. Hence, they do not need to start or end in the center.

The alphabet of the LlRrS protocol contains a total of 5 possible gaze gestures: 4 horizontal unidirectional gaze gestures (LlRr) and a squint movement (S). This protocol utilizes only horizontal eye movements. The gaze gestures are either small (e.g. "r") or big (e.g. "R") eye movements, and they do not go back to the start of the gesture after that movement. Hence, they do not need to start or end in the center.

3.2 Data Recording and Preprocessing

Data is recorded with eye-facing cameras embedded in smart glasses. If open-eye gaze gestures should be utilized alongside closed-eye gaze gestures, approaches from related work can be used. We therefore declare open-eye gaze gestures out of scope for our approach. With all closed-eye gaze gesture protocols, users close their eyes to start closed-eye user input, and open them to end the input. The system detects that eyes are closed by pupils not being visible anymore. If the duration of eyes being closed is longer than a predefined threshold (1.5 s in our configuration) the system treats eye movements in between closing and opening eyes as potential closed-eye gaze gestures. The video captured from this input is then processed further once eyes are opened. For preprocessing, we apply frame-wise histogram equalization and image sharpening (Fig. 3). For the latter, with the 320x240 pixel resolution we employed in our evaluation, a 5x5 pixel luma matrix with a luma effect strength of 1.5 pixel was utilized.

3.3 Closed-Eye Gaze Gesture Detection and Extraction

On a preprocessed closed-eye movement video we employ frame-wise full optical flow to extract movements. For a pair of two subsequent MxN pixel frames, the

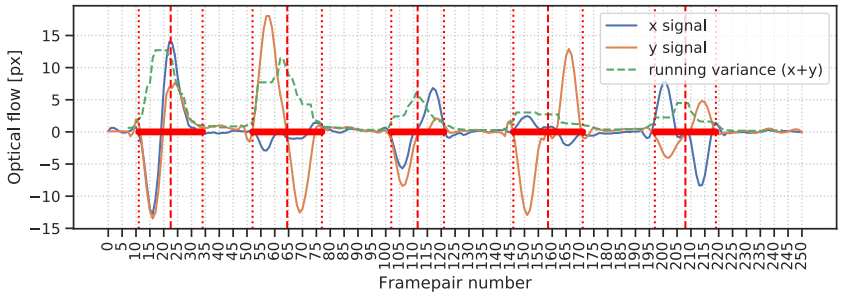


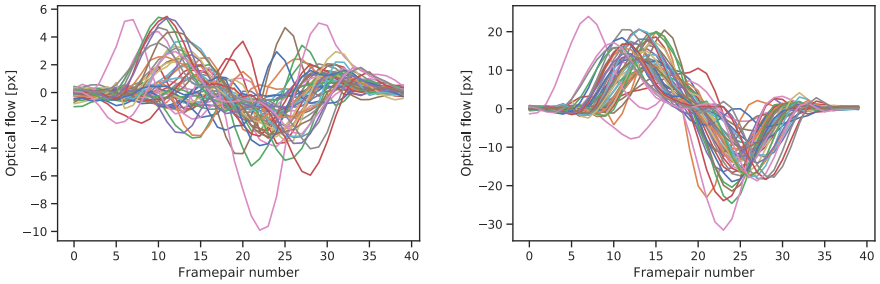
Fig. 4. Filtered optical flow in pixels between frame pairs, for an LRUDS video sample, containing "SDLRU" closed-eye gaze gestures, with detected gestures marked red.

optical flow yields two MxN matrices containing the horizontal and the vertical optical flow, respectively. Hence, a video consisting of F frames results in F-1 pairs of optical flow matrices. With that series of matrices, we compute the 10%, 25%, 50%, 75%, and 90% quantiles of optical flow per matrix. Those effectively capture both major positive and major negative closed-eye movements in optical flow. Each of them can therefore be understood as a one-dimensional time series, capturing closed-eye movements over time in either horizontal or vertical direction. To reduce noise in each of those time series, we employ a Savitzky–Golay filter. For subsequent closed-eye gaze gesture detection, the filter configuration is a window length of 0.5 s and polynomial grade of 4. For feature extraction used in subsequent model training and prediction, we further utilize all permutations of window length $\{0.3\,\text{s}, 0.4\,\text{s}, 0.5\,\text{s}\}$ and polynomial grade $\{2, 3, 4\}$.

For detecting segments of active eye movements in the filtered optical flow time series, we at first compute the sum of the filtered 10% and 90% quantile time series per axis, then the piecewise $L2$-norm over both axis. The variance of the resulting time series shows closed-eye movements in the video as non-zero periods. To automatically detect those periods we utilize a Schmitt-Trigger with a window length and a high and low trigger of $\frac{1}{2}$ s, 1.5, and 1 for the LRUDS protocol, and $\frac{1}{3}$ s, 2, and 1.5 for other protocols (Fig. 4). Each detected period of activity is segmented at its center with a fixed size window. The window length is 1.3 s for the LRDUS protocol and 1 s for other protocols. The reason for the LRUDS protocol utilizing different setting and longer windows is that its movements are bidirectional, hence take slightly longer. The segmented periods of activity become closed-eye gaze gestures (see example in Fig. 5), which we subsequently utilize for both training and user input recognition.

3.4 Closed-Eye Gaze Gesture Recognition: Training and Prediction

Users enroll by performing closed-eye gaze gestures according to instructions of the selected protocol. All possible gaze gestures of the corresponding protocol are thereby recorded multiple times together with ground truth. Gaze gestures



(a) horizontal (b) vertical

Fig. 5. Human understandable representation of multiple automatically extracted closed-eye gaze gesture samples, for the "U" movement from the LRUDS protocol. (a) and (b) show closed-eye movement in pixels in between frame pairs, in horizontal and vertical direction. While each sample contains multiple time series, this representation only shown the sum of the 10% and 90% quantile of the extracted 2D optical flow over time.

are detected and segmented, and together with the ground truth form the basis for training a user specific closed-eye gaze gesture recognition model.

As with our approach gaze gestures manifest as one positive or negative peak in optical flow over time in horizontal and/or vertical direction, we believe that those peaks can be represented with a few main components using Principal Component Analysis (PCA). To avoid individual features dominating PCA, we scale them to mean=0 and std=1 before applying PCA. The strongest principal components (PCs), which together explain 80% of the variance of the training data, will be used for subsequent training and prediction with the model. Note that the parameters for centering, scaling, and PCA transformation are derived from training data only and applied to recognition-case data likewise (cf. figure 1).

4 Evaluation

To evaluate our approach we compare the gesture detection and classification rates of all proposed closed-eye gaze gesture protocols. For this we record test data, perform gaze gesture detection and extraction, and use extracted gaze gestures to comparatively evaluate the performance of different machine learning approaches on recognizing gaze gestures.

4.1 Evaluation Dataset

Data recording was done with a first generation Pupil Eye Tracker [11], which has the form factor of smart glasses. The device has a right-eye-facing camera to record videos of gaze gesture input with 60 Hz and a resolution of 320x240 pixels. We recorded data of one subject over a total of 8 sessions (2 per protocol),

Table 1. The utilized dataset: amount of recordings per protocol with their total contained closed-eye gaze gestures. The last two columns depict results of applying our gaze gesture extraction approach to this dataset (Sec. 5.1).

Protocol	Recordings	Gaze gestures	Gaze gestures extracted	Extraction rate
LRUDS	60	290	240	82.8%
LlRrUuDdS	16	204	182	89.2%
lrudS	16	148	126	85.1%
LlRrS	89	382	350	91.6%

indoors, in office spaces. Over all protocols, the dataset thereby contains a total of 181 closed-eye gaze gesture video recordings, which together contain a total of 1024 closed-eye gaze gestures (Tab. 1).

4.2 Evaluation Procedure

Gaze gesture detection rates are quantified from the amount of correctly detected and extracted gaze gestures. With extracted gaze gestures, we then analyze components contained in gaze gestures. Further, we train and apply different machine learning models to perform gaze gesture classification, which will be quantified based on the amount of (in-)correctly classified gaze gestures.

For model tuning, selection, and reporting final performances, we utilize nested cross validation for data partitioning. We select models using the highest accuracies from the inner loop and report final results with gaze gesture confusion matrices for each protocol from the outer loop. The hyperparameter search for model tuning relies on logarithmic parameter grids. Data specific preprocessing (such as PCA) is done inside cross validation to not bias results.

5 Results

5.1 Closed-Eye Gaze Gesture Detection and Extraction Results

We found our closed-eye gaze gesture extraction to correctly detect and extract gestures for 82.8%-91.6% of all samples (Tab. 1). The threshold to accept a movement as gaze gesture has been configured to avoid false positives, which would be gaze gestures extracted from noise or small, unintended eye movements. This causes a trade-off, enabling low false positives by accepting certain false negatives. While with our data all intended gaze gestures, together with false positives, would initially have been recognized, the threshold was set too high for certain movements, which, as a consequence, were not recognized.

5.2 Closed-Eye Gaze Gesture Decomposition Results

Applying PCA with extracted closed-eye gaze gestures indicates dominance of few PCs, which aligns with our expectations. To explain 80%, 90%, and 95%

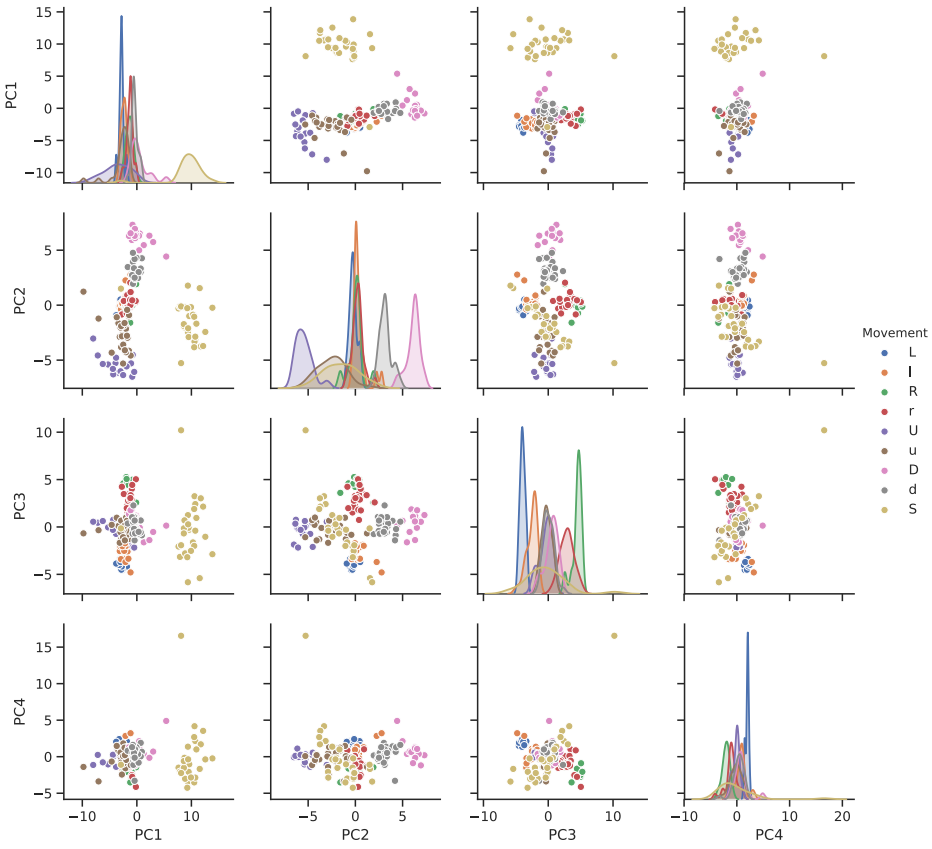


Fig. 6. Distribution of samples in PC space for the LlRrUuDdS protocol. While there is some overlap visible with certain classes, other classes seem clearly distinguishable.

of the variance in the data, 9, 13, and 17 components are required for the LRUDS protocol, 7, 11, and 15 for the LlRrUuDdS protocol, 7, 11, and 14 for the lrudS protocol, and 5, 7, and 11 for the LlRrS protocol. An interesting detail thereby is that the protocol LRUDS, in which all gaze gestures contain bidirectional movements, requires the most components (9). In contrast, protocol LlRrUuDdS, which consists of more gaze gestures, but in which each gaze gesture only contains unidirectional movements, requires fewer components (7). Protocol LlRrS requiring the fewest components (5), as all of its gaze gestures only contain horizontal movements. Further, with all protocols, the distribution of samples in PC space indicates that some classes are easily separable, while others show stronger amounts of overlap (see LlRrUuDdS protocol example, Fig. 6).

Table 2. Mean/standard deviation of gaze recognition accuracy per protocol and model, over all gaze gestures in that protocol and all inner CV repetitions. For each protocol, the result is emphasized for the model that was selected for the outer CV loop evaluation from those results, and its hyperparameters are stated in the bottom row.

Model\Protocol	LRUDS	LlRrUuDdS	lrudS	LlRrS
KNN	0.988/0.020	0.945/0.067	0.992/0.023	0.926/0.020
LDA	0.979/0.021	0.940/0.073	0.984/0.031	0.900/0.024
CART	0.958/0.037	0.857/0.089	0.944/0.049	0.906/0.044
SVM Linear	0.983/0.021	0.940/0.049	**0.992/0.023**	**0.937/0.020**
SVM Radial	0.988/0.020	**0.961/0.049**	0.992/0.023	0.937/0.034
ANN	**0.992/0.017**	0.912/0.064	0.992/0.023	0.920/0.033
Hyperparam.	hidden=(50), $\alpha = 3^{-10}$	$C = 3^2$, $\gamma = 3^{-6}$	$C = 3^{-4}$	$C = 3^4$

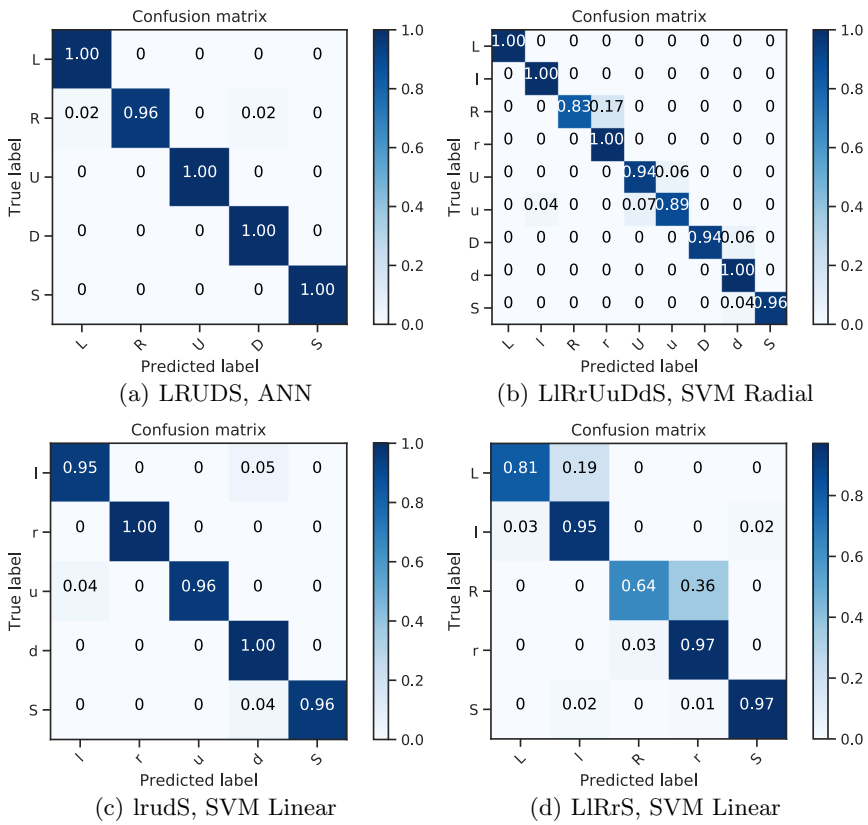
5.3 Closed-Eye Gaze Gesture Recognition: Model Tuning Results

Evaluation results from tuning different models in the inner CV loop of our evaluation in general indicate good closed-eye gaze gesture recognition rates for all protocols (Tab. 2). Even simple models, such as a linear SVM, are able to achieve good results. Nevertheless, recognition accuracies over protocols vary in between 99.2%-93.7%, which indicates that gaze gestures of different protocols might be differently hard to distinguish.

5.4 Closed-Eye Gaze Gesture Recognition: Protocol Results

From applying the selected model types and hyperparameter configurations (Tab. 2) to the corresponding protocols in the outer CV loop of our evaluation, we obtain final gaze gesture confusion matrices (Fig. 7).
Gaze gestures in all except the LlRrS protocols seem well distinguishable. Overall closed-eye gaze gesture recognition accuracy is 99.2% with the LRUDS protocol, 95.6% with the LlRrUuDdS protocol, 97.6% with the lrudS protocol, and 92.9% with the LlRrS protocol. Most gaze gesture prediction errors confuse movements which are into the same direction but of different size. This is strongly visible with the LlRrS protocol, where 36% of "R" movements are wrongly predicted to be "r", and 19% of "L" as "l". It is also visible with the LlRrUuDdS protocol, which confuses 17% of "R" movements as "r", 6% of "D" as "d", 6% of "U" as "u", and 7% of "u" as "U".

To summarize: while the LlRrUuDdS and LlRrS protocols yield higher closed-eye gaze gesture detection and extraction rates (89.2% and 91.6%), gaze gesture recognition results seem to be in favor of the LRUDS and lrudS protocols (99.2% and 97.6%). While the LlRrU protocol does not require vertical and relies purely on horizontal eye movements, the LlRrUuDdS protocol provides the biggest alphabet of gaze gestures. In combination with the noticeably better gaze gesture recognition results, the LlRrUuDdS protocol (89.2% gesture detection rate, 95.6% gesture classification accuracy) hence seems to be the most robust option within the evaluated protocols for closed-eye gaze gestures with cameras in smart glasses.



(a) LRUDS, ANN (b) LlRrUuDdS, SVM Radial

(c) lrudS, SVM Linear (d) LlRrS, SVM Linear

Fig. 7. Symbol confusion per protocol from the outer CV evaluation.

6 Conclusion

In this paper we investigated the feasibility of detecting and recognizing closed-eye gaze gestures from close-up optical sources, such as cameras embedded in smart glasses. We proposed four basic closed-eye gaze gesture protocols, consisting of different horizontal and vertical eye movements. We further proposed a methodology to detect, extract, and classify the corresponding gaze gestures, based on full optical flow extraction, time series processing, and machine learning. For our evaluation we utilized data from an eye tracker in the form factor of smart glasses, which records closed-eye movement videos with an embedded camera. For each closed-eye gaze gesture protocol, in the evaluation we investigated the detection rate of gestures, the amount of components required to represent extracted gestures, as well as gaze gesture confusion during classification. Results indicate gaze gesture detection rates of 82.8%-91.6% and average gaze gesture classification accuracies of 92.9%-99.2%. Further, with PCA, 80%, 90%, and 95% of variance in extracted closed-eye gaze gestures can be explained with a

maximum of 9, 13, and 17 principal components, respectively, for all protocols. While those numeric results are based on limited amounts of data, our work has demonstrated the technical feasibility of detecting and recognizing closed-eye movements from optical sensors. It therefore is a first step towards utilizing closed-eye user input with cameras embedded in smart glasses, and extending gaze gesture alphabets with closed-eye gestures. Future work could investigate closed-eye gaze gestures across users with and without enrollment phases. Further, it could investigate the feasibility of combining open- and closed-eye gaze gestures in a single protocol. The feasibility and applicability of employing other sensors for closed-eye gaze gestures, such as EOG sensors embedded in the frame of smart glasses, would too be an interesting field of investigation for future work.

References

1. Bulling, A., Ward, J.A., Gellersen, H., Troster, G.: Eye movement analysis for activity recognition using electrooculography. IEEE Transactions on Pattern Analysis and Machine Intelligence **33**(4), 741–753 (April 2011)
2. Bulling, A., Roggen, D., Tröster, G.: It's in your eyes - towards context-awareness and mobile HCI using wearable EOG goggles. In: Proc. UbiComp 2008. vol. 344, pp. 84–93 (Sep 2008)
3. Bulling, A., Roggen, D., Tröster, G.: Wearable EOG goggles: Eye-based interaction in everyday environments. In: CHI'09 Extended Abstracts on Human Factors in Computing Systems. pp. 3259–3264. ACM, New York, NY, USA (2009)
4. Drewes, H., De Luca, A., Schmidt, A.: Eye-gaze interaction for mobile phones. In: Proc. Mobility'07. pp. 364–371. ACM, New York, NY, USA (2007)
5. Drewes, H., Schmidt, A.: Interacting with the computer using gaze gestures. In: Human-Computer Interaction (INTERACT). pp. 475–488. Springer (2007)
6. Heikkilä, H., Räihä, K.J.: Speed and accuracy of gaze gestures. Journal of Eye Movement Research **3**(2) (Nov 2009)
7. Hintze, D., Hintze, P., Findling, R.D., Mayrhofer, R.: A large-scale, long-term analysis of mobile device usage characteristics. Proc. ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **1**(2) (Jun 2017)
8. Hyrskykari, A., Istance, H., Vickers, S.: Gaze gestures or dwell-based interaction? In: Proc. ETRA 2012. pp. 229–232. ETRA '12, ACM, New York, NY, USA (2012)
9. Ishimaru, S., Kunze, K., Tanaka, K., Uema, Y., Kise, K., Inami, M.: Smart eyewear for interaction and activity recognition. In: Proc. Conference Extended Abstracts on Human Factors in Computing Systems. pp. 307–310. ACM (2015)
10. Istance, H., Hyrskykari, A., Immonen, L., Mansikkamaa, S., Vickers, S.: Designing gaze gestures for gaming: An investigation of performance. In: Proc. ETRA 2010. pp. 323–330. ETRA '10, ACM, New York, NY, USA (2010)
11. Kassner, M., Patera, W., Bulling, A.: Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. In: Proc. UbiComp 2014, Adjunct Publication. pp. 1151–1160. ACM (2014)
12. Wobbrock, J.O., Myers, B.A., Kembel, J.A.: Edgewrite: A stylus-based text entry method designed for high accuracy and stability of motion. In: Proc. UIST 2003. pp. 61–70. ACM, New York, NY, USA (2003)
13. Wobbrock, J.O., Rubinstein, J., Sawyer, M.W., Duchowski, A.T.: Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In: Proc. ETRA 2008. pp. 11–18. ACM, New York, NY, USA (2008)