iWink: Exploring Eyelid Gestures on Mobile Devices

Zhen Li
University of Toronto
zhen@dgp.toronto.edu

Mingming Fan
Rochester Institute of Technology
mingming.fan@rit.edu

Ying Han
University of Toronto
yingh.han@mail.utoronto.ca

Khai N. Truong
University of Toronto
khai@cs.toronto.edu

ABSTRACT

Although gaze has been widely studied for mobile interactions, eyelid-based gestures are relatively understudied and limited to few *basic* gestures (e.g., blink). In this work, we propose a gesture grammar to construct both *basic* and *compound* eyelid gestures. We present an algorithm to detect nine eyelid gestures in real-time on mobile devices and evaluate its performance with 12 participants. Results show that our algorithm is able to recognize nine eyelid gestures with 83% and 78% average accuracy using *user-dependent* and *user-independent* models respectively. Further, we design a *gesture mapping scheme* to allow for navigating between and within mobile apps only using eyelid gestures. Moreover, we show how eyelid gestures can be used to enable cross-application and sensitive interactions. Finally, we highlight future research directions.

CCS CONCEPTS

• **Human-centered computing → Interaction techniques**; **Ubiquitous and mobile computing**.

KEYWORDS

Eyelid gestures; hands-free interaction; mobile interaction

ACM Reference Format:
Zhen Li, Mingming Fan, Ying Han, and Khai N. Truong. 2020. iWink: Exploring Eyelid Gestures on Mobile Devices. In *1st International Workshop on Human-centric Multimedia Analysis (HuMA'20), October 12, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3422852.3423479

1 INTRODUCTION

Interacting with mobile devices can be cumbersome or distracting when users' hands are occupied or temporarily unavailable [4, 13]. Although gaze has been widely studied to enhance hands-free interactions (e.g., [5, 15, 33, 39]), eyelid-based gestures are relatively under-explored. Previous research on eye-based interactions has mostly focused on gestures with eyelids in one state (e.g., close or open) briefly, such as blinking and winking [2, 11, 12, 14, 16, 17, 19–22, 24, 29, 30, 36, 38, 40, 44], or on gestures involving eyebrow movements [7, 9, 26], for hands-free interaction. However, two eyelids can be in different states (e.g., close or open) for different *duration* (e.g., short or long) in different *sequences* (e.g., concurrently or sequentially) to form a rich set of eyelid gestures to enhance hands-free interactions. As a result, we extend this line of research, as shown in our related work Table 1, by presenting a real-time algorithm, *iWink*, to detect a more comprehensive set of eyelid gestures on smartphones and evaluating its performance with users. iWink is beneficial in scenarios where it is not easy to use both hands to interact with smartphones, such as in a tight crowd of people (e.g., a busy subway). Further, iWink could also benefit people with physical impairments to interact with their smartphones without using hands.

Specifically, we identified four *primitive* eyelid states and constructed nine eyelid gestures. We then designed a real-time algorithm to recognize these states and gestures on smartphones, which handles the variations of eyelid gestures within and across users. We conducted a user study with 12 participants, and the results show that it can recognize 4 eyelid states with .97 overall accuracy and 9 eyelid gestures with .83 overall accuracy. We further presented applications to demonstrate its usage scenarios. Specifically, we designed an *eyelid gesture mapping scheme*, which allows users to navigate within and across mobile apps only using eyelid gestures. We also designed an eyelid-state based *cross-application interaction* method that allows users to switch two apps simply by closing or opening one eyelid, which reduces the need for costly manual app

Table 1: The Related Work Table. *iWink* detects six *basic* gestures with short- and long-duration and three *compound* gestures on smartphones and evaluates the performance with users.

Work	The set of eyelid gestures recognized				Evaluation
	Basic gestures			*Compound* gestures	
	Blink	Wink (left eyelid)	Wink (right eyelid)		
iWink	✓(2)	✓(2)	✓(2)	**3**	✓
Kim et al. [19]	✓(1)	✓(1)	✓(1)	1	✗
Zhang et al. [44]	✓(1)	✓(1)	✓(1)	✗	✓
Shaw et al. [36]	✓(1)	✓(1)	✓(1)	✗	✗
Ku et al. [23]	✓(1)	✓(1)	✓(1)	✗	✓
Jota & Wigdor [16]	✗	✓(1)	✓(1)	✗	✗
Hemmert et al. [12]	✗	✓(1)	✓(1)	✗	✗
Kaufman et al. [17]	✓(1)	✗	✗	✗	✓
Kwon et al. [24]	✓(1)	✗	✗	✗	✗
Miluzzo et al. [30]	✓(1)	✗	✗	✗	✓
Ishimaru et al. [14]	✓(1)	✗	✗	✗	✓
Heikkilä & Räihä [11]	✓(1)	✗	✗	✗	✓
Wang & Grossman [40]	✓(1)	✗	✗	✗	✓

HuMA'20, October 12, 2020, Seattle, WA, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8151-2/20/10…$15.00
https://doi.org/10.1145/3422852.3423479

switching [3, 25]. We also designed an eyelid-state based *sensitive interaction* for users to spot sensitive on-screen content quickly by closing any eyelid to avoid shoulder-surfing attacks.

2 RELATED WORK

While gaze-based interaction has been widely studied, *eyelid-based interaction*, which involves actively opening and closing eyelids, has relatively under-explored. The most commonly studied eyelid gesture is *Blink*. Blink has been explored for people with disabilities. For example, Kaufman et al. used EOG sensors to detect Blink following an eye movement to trigger computer commands [17]. Similarly, Kwon and Kim detected Blink and eye movements based on the EOG signals received from the electrodes positioned on an eyeglass frame and used the blinks to trigger mouse click [24]. Further, the Blink gesture has also been explored for general users. For example, Miluzzo et al. used Blink to activate an application for mobile phone users [30]. Ishimaru et al. demonstrated that the frequency of Blink combined with head motion can be used to detect five activities (i.e., reading, talking, watching TV, math problem solving, and sawing) [14], and their method was able to detect Blink with 0.80 precision and 0.78 recall [14]. Wang and Grossman leveraged the synchronicity of touch and blink events to augment the input vocabulary of smartwatches [40].

Variations of blinks have further been studied. For example, Heikkilä and Räihä used an eye tracker to detect long Blink and used it to stop a moving onscreen target [11]. Kim et al. proposed one sequential eyelid gesture, which connects two Blink gestures together (i.e., double blink) [19].

Wink is another eyelid gesture that has been studied for people with disabilities. For example, Shaw et al. constructed a prototype by mounting IR emitters and receivers on a pair of eyeglasses [36]. The prototype detects the open and close states for each eye and uses such information to infer three atomic eyelid gestures: Blink, Wink left eye, and Wink right eye. Similarly, Zhang et al. designed a smartphone application that detects a long Blink gesture and two Wink gestures and used them together with gaze direction information to type characters [44]. The proposed method was evaluated with 12 users and was able to detect the long Blink gesture with an average accuracy of 77.5% [44].

Wink has also been used for general users. Ku et al. used an eye tracker to detect Wink left eye, Wink right eye, and Blink gestures [23]. While their initial evaluation showed that the method was able to detect these three gestures with more than 90% accuracy, they did not reveal details about the method [23]. Instead of using an eye tracker, Hemmert et al. used a common webcamera to detect two Wink gestures and used them to alter onscreen contents [12]. Specifically, with one eye closed, users can temporarily see a zoomed-in view through a sniper scope, a zoomed-out view of a webpage, or the most recently edited documents in a file finder app. Lastly, Jota and Wigdor took a step further and proposed potential usage of two Wink gestures for mobile, desktop, and tabletop devices [16]. For example, when using a mobile phone to browse a webpage, a user closes an eye to invoke the URL bar or the controls (e.g., backward, forward) and opens both eyes to see the webpage content without any controls or the URL bar; when working with two applications on a desktop, users can close any of the two eyes to bring a different application to the foreground [16].

In addition to Blinks and Winks, our two eyelids can be in different states (e.g., close or open) for different *duration* (e.g., short or long) in different *sequences* (e.g., concurrently or sequentially) to form a rich set of eyelid gestures to enhance hands-free interactions. As a result, there is an opportunity to explore other potential eyelid gestures and their usage for interactions.

We create a related work table (see Table 1) to better position our research and contrast it with the related work. To sum it up, there are three limitations with the previous eyelid gesture research. First, prior work has mostly focused on Blinks and Winks and their usage in interactions. Consequently, eyelid gestures with a sequence of eyelid states have been largely left unexplored (with the exception of the double Blink gesture [19]). Second, even within Blinks and Winks, previous work has primarily focused on short-duration versions of these gestures [14, 17, 23, 24, 30, 36] with the exception of the long Blink gesture [11, 19, 44]. In contrast, long-duration eyelid gestures, such as long Wink gestures, and variations of such gestures could potentially enable new interactions. Third, previous research often focused on proposing proof-of-concept and did not provide implementation details of their recognition method. Consequently, prior work often did not have an formal evaluation [12, 16, 19, 24, 36].

To address these limitations, we propose taxonomies to construct both *basic* eyelid gestures and *compound* eyelid gestures, which will be described in the next section. We propose a real-time algorithm to address the challenges by detecting nine eyelid gestures on a smartphone. We then evaluate its performance and robustness with 12 users in a user study. Finally, we present potential usage of eyelid gestures.

3 EYELID GESTURE DESIGN

Two eyelids can be in four *primitive* states: *both eyelids **O**pen*, ***B**oth eyelids closed*, ***R**ight eyelid closed*, and ***L**eft eyelid closed*. These states are denoted as O, B, R, and L. Technically, eyelids could also be in *half-closed* states. However, as previous research suggests that keeping eyelids in a half-close state (e.g., squinting) for a sustained period places stress on eye muscle and can cause it to twitch or cramp [16], we focus on the close and open eyelid states in this work as an initial exploration to this vast design space. "Both eyelids open" (O) is chosen to be the *gesture delimiter* because it is the default state in which our eyelids are when we are awake. Thus, all eyelid gestures start and end with the O state. We represent an eyelid gesture using the eyelid states between the start and end states in the gesture.

Since users can sustain their eyelids in an open or close state for different duration and it can be hard for them to remember an exact duration, we divide the duration of an eyelid state into two discrete levels: *short* and *long*. *Short* duration refers to the time it takes for a user to *intentionally* close an eyelid (e.g., longer than a spontaneous blink (50-145ms [37])) and open it immediately afterward. For example, the eyelid gesture that "intentionally closes the right eyelid and opens it immediately afterward" is denoted as **R**. In contrast, *Long* duration is closing an eyelid, holding it for some time, and then opening it. As users may prefer to hold for

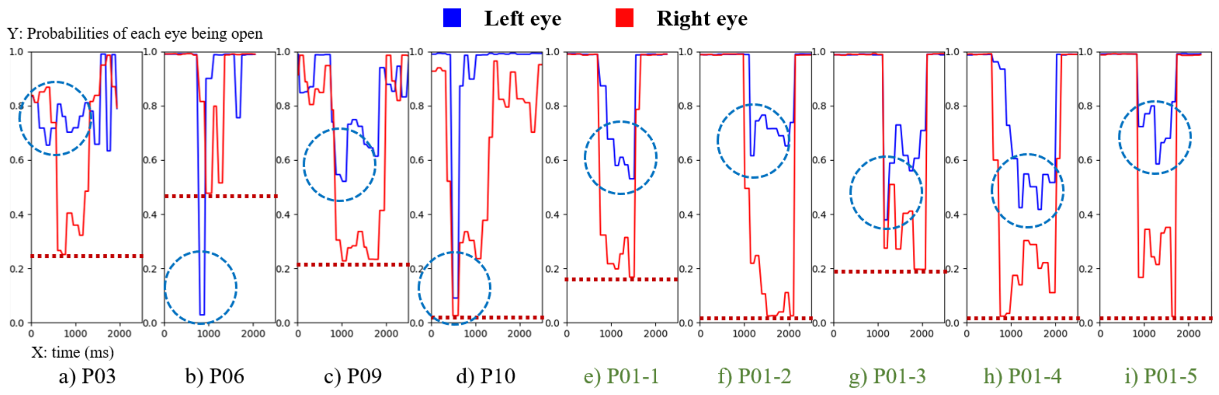


Figure 1: The red and blue lines show the probability estimations of the right and left eye being open respectively. (a–d): Four different users performing the same gesture *R*. (e–i): The same user performing the gesture *R* five times.

different time, it is important to allow them to decide their preferred holding time as long as they keep it consistent. For simplicity and consistency, in this work, users count a fixed number of numbers (e.g., three) by heart for holding. *Holding* an eyelid state is denoted by the symbol '-'. For example, "intentionally close the right eyelid, hold it for some time, and then open it" gesture is denoted as **R-**. Following this definition, we could construct six **basic** eyelid gestures that contain only *one* key eyelid state between the common start and end states (O): B, R, L, B-, R-, and L-.

We could add another eyelid state between the common start and end states to construct **compound** eyelid gestures. For example, *'B-R-'* represents the gesture that starts from the common start state "both eyelids open", transitions to "Both eyelids close", holds in the state for some time (-), transitions to "only the Right eyelid close", holds in the state for some time (-), and finally ends at the common end state "both eyelids open". Similarly, we could construct even more complex eyelid gestures with three or more states between the start and end states using this mechanism. As an initial exploration of the vast eyelid gestures space, we focus on three simple compound eyelid gestures: B-R-, B-L-, and BOB, which represents intentionally blinking two eyelids twice. Next, we present our algorithm, iWink, to recognize these 9 gestures: B, R, L, B-, R-, L-, B-R-, B-L-, and BOB.

4 EYELID GESTURE DETECTION

4.1 Implementation Challenges

Although there are methods to estimate the *moment-to-moment* probability of an eye being open (e.g., Google Mobile Vision API [27]), sever challenges must be overcome to recognize *eyelid gestures* from the stream of moment-to-moment probability estimations. First, the start and end of an eyelid gesture must be identified from the noisy input stream. This is challenging because there are variations in the probability estimations of an eye being open for different users and even the same user (Fig. 1). First, different users may perform the same gesture differently. As highlighted by the blue circles in Fig. 1a-d, when performing the same gesture *R*, users unintentionally close their left eye as they are closing their right eye. What's more, such effect varies from being minor (P03, Fig. 1a) to being severe (P06 and P10 in Fig. 1b, d). Second, a user can perform the same eyelid gesture with variations too. As shown in Fig. 1e-i, while the user performs the gesture *R* for multiple times, the probability estimations of the right eye being open as well as that of the left eye being 'dragged down' also vary significantly. These challenges make it impractical to use a threshold-based approach. Instead, we create a machine learning algorithm to handle variations within and across users.

4.2 Implementation Details

The pipeline of the algorithm is shown in Fig. 2. We implemented our algorithm on Samsung S7 running Android OS 8.0. The algorithm first obtains an image frame from the front-camera at a speed of 30 frames per second with 640 x 480 resolution. It then identifies the eyes, computes the probability of each eye being open using Mobile Vision API [27] (Fig. 2a) and outputs the probabilities as a pair (P_L, P_R). Next, the *eyelid-state SVM classifier* takes the pair (P_L, P_R) as input and classifies the eyelids into two states: *open* (O) if both eyes are open and *close* (C) if any eye is closed. We design the classifier to recognize only two states because eyelid gestures always start and end with the *O* state – the gesture delimiter. To increase the robustness of the gesture recognition, we filter out extremely short duration gestures, which are likely introduced by *spontaneous blinks* (e.g., 50-145ms [37]) or unstable estimations of the eye open state. Next, the algorithm takes (P_L, P_R) in a segment as input and further detects whether the segment is a short-duration gesture or a long-duration gesture (Fig. 2c). Then, the algorithm resamples the probability pairs in the segment and converts them into the fixed-length vector (Fig. 2d). Finally, the resampled fixed-length vector is further distinguished within each group of eyelid gestures using SVM classifiers. Specifically, the short-duration gesture SVM classifier detects whether the segment is *R*, *L*, *B* or *BOB*; similarly, a long-duration gesture SVM classifier detects whether the segment is *R-*, *L-*, *B-*, *B-R-*, or *B-L-* (Fig. 2e). All SVM classifiers use the RBF kernel with default parameters in the scikit-learn library [32]. Our code is available here[1].

[1]https://github.com/mingming-fan/EyelidGesturesDetection

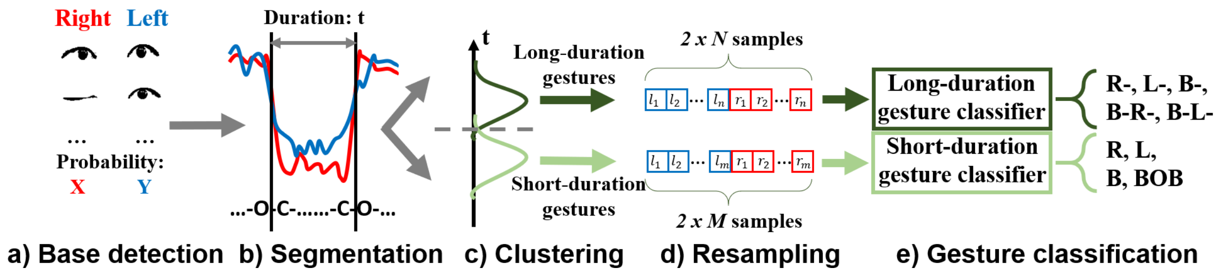


Figure 2: The algorithm flowchart: a) it estimates the probabilities of two eyelids being open (P_L, P_R) using Mobile Vision API [27]; b) the *eyelid-state classifier* classifies the eyelids into two states (*open* if both are open and *close* otherwise) and detects the start and end of a gesture; c) it distinguishes long gestures from short ones; d) it resamples the data points to get a fixed-length vector; e) the corresponding long- or short- gesture classifier detects the eyelid gesture.

5 USER STUDY

5.1 Participants

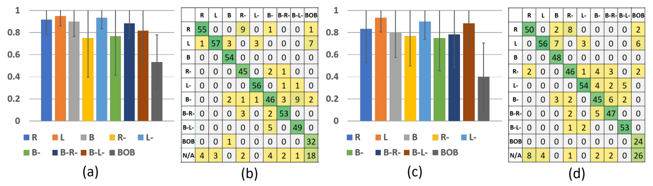
Twelve volunteers (seven females and five males), between the ages of 20 and 27 (*Mean* = 23, *SD* = 3), participated in the study. Their eye colors included dark brown (9), light brown (1), amber (1), and hazel (1). In the study, seven wore glasses, two wore contact lenses and three did not wear glasses or contact lenses.

5.2 Procedure

The study was conducted in a quiet office room with each participant sitting at a desk. A Samsung S7 Android phone was placed in a dock on the desk and used as the testing device. The phone was positioned to capture the participant's face with its front camera. The evaluation app presented the textual description of a target eyelid state on the top side of the screen. The participant was asked to first prepare her eyes in the target state and then press the 'START' button to start data collection at a speed of 30 frames per second. The app beeped after collecting 200 frames. The app presented another eyelid state and repeated the procedure until data samples for all four eyelid states were collected. The evaluation app used the data samples to train an eyelid state classifier on the phone in real-time, which took on average 558 milliseconds to complete. The evaluation app next collected five samples for each of the nine eyelid gestures. Similarly, the app presented the textual description of a target gesture on the top side of the screen. The participant pressed a button on the UI and then performed the gesture. Upon finishing, she pressed the button again. The app recorded and stored a sequence of eyelid states during this period. The app collected the samples for all eyelid gestures and used them for training the classifier. After a break, the app asked the participant to perform each eyelid gesture in a random order five times again, and these data were used for *user-dependent* testing.

5.3 Results

5.3.1 Primitive Eyelid States Recognition. We performed a 10-fold cross validation to evaluate the performance of the eyelid-state classifier for each participant and averaged the results across all participants. The overall accuracy for each eyelid state was as follows: *both eyelids open* (.99), *right eyelid close* (.95), *left eyelid close* (.96), and *both eyelids close* (.96).



(b)

	R	L	B	R-	L-	B-	B-R-	B-L-	BOB
R	55	0	0	9	0	1	0	0	1
L	1	57	3	0	3	0	0	0	7
B	0	0	54	0	0	0	0	0	0
R-	0	0	0	45	0	2	1	0	0
L-	0	0	0	0	56	0	1	1	0
B-	0	0	2	1	1	46	3	9	2
B-R-	0	0	0	3	0	2	53	0	0
B-L-	0	0	0	0	0	5	0	49	0
BOB	0	0	1	0	0	0	0	0	32
N/A	4	3	0	2	0	4	2	1	18

(d)

	R	L	B	R-	L-	B-	B-R-	B-L-	BOB
R	50	0	2	8	0	0	0	0	2
L	0	56	7	0	3	0	0	0	6
B	0	0	48	0	0	0	0	0	0
R-	2	0	0	46	1	4	3	0	2
L-	0	0	0	0	54	4	2	5	0
B-	0	0	3	2	0	45	6	2	0
B-R-	0	0	0	2	0	5	47	0	0
B-L-	0	0	0	1	2	0	0	53	0
BOB	0	0	0	0	0	0	0	0	24
N/A	8	4	0	1	0	2	2	0	26

Figure 3: The accuracy and confusion matrix (columns: ground truth; rows: predictions; N/A: not recognized) of (a, b) user-dependent and (c, d) user-independent models.

5.3.2 User-dependent Eyelid Gestures Recognition. As is described in the previous section, for each participant, we trained a user-dependent classifier with five samples for each gesture and tested with another five samples from the same user. We then averaged the performance of each gesture across all participants. The average accuracy of all gestures was .83 (*SD* = .11). The accuracy for each gesture was as follows (Fig. 3a): *R* (.92), *L* (.95), *B* (.90), *R-* (.75), *L-* (.93), *B-* (.77), *B-R-* (.88), *B-L-* (.82), and *BOB* (.53). Fig. 3b also shows the confusion matrix for each gesture.

5.3.3 User-Independent Eyelid Gestures Recognition. To understand how well our algorithm could recognize the eyelid gestures of a new user whose data the algorithm is not trained on, we conducted a *user-independent* evaluation by adopting *leave-one-participant-out* scheme. We held a participant's data for testing and trained the classifier on the rest of the participants' data. We performed this evaluation for each participant and averaged the performance across all participants. The accuracy for each gesture was as follows (Fig. 3c): *R* (.83), *L* (.93), *B* (.80), *R-* (.77), *L-* (.90), *B-* (.75), *B-R-* (.78), *B-L-* (.88), BOB (.40). The confusion matrix results are available in Fig. 3d. The overall accuracy was .78 (*SD* = .13), which was slightly lower than that of the user-dependent classifier; this was expected because there were more variations across users than within a user. The more unrecognized cases for *BOB* also indicate larger variations in ways how they performed *BOB*, such as the duration between two *B* gesture.

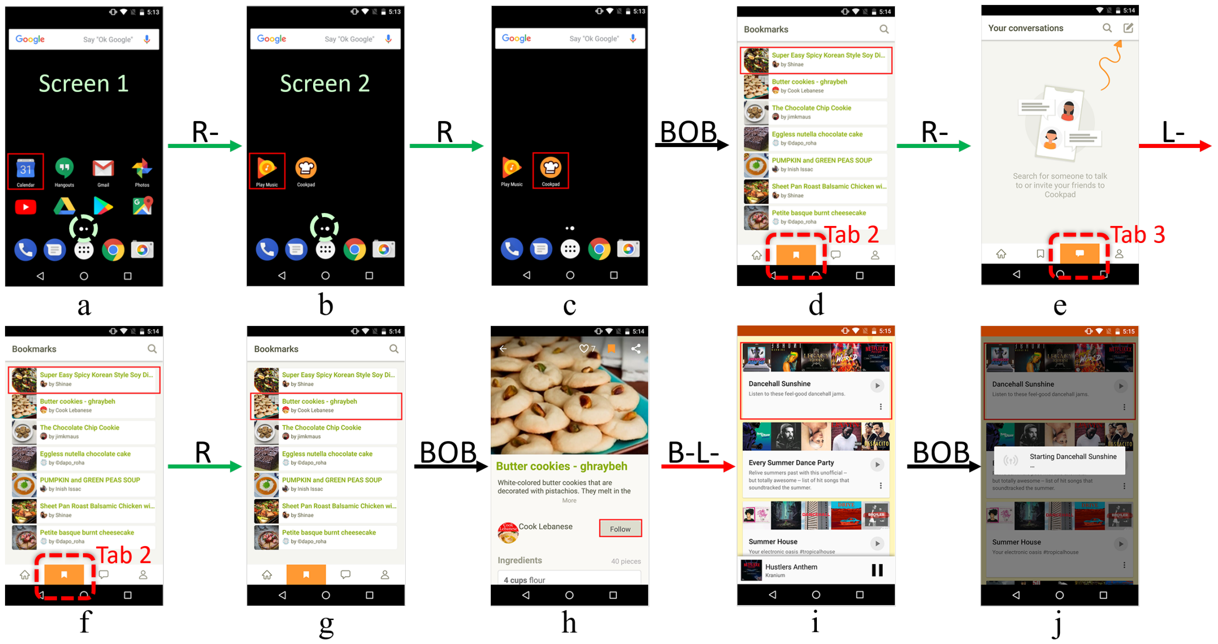


Figure 4: (a, b) *R-* moves the current screen to the next. (c) *R* moves the focus to the next app and (d) *BOB* opens the target app. (e, f) *R-*/*L-* moves forward/backward between tabs. (g) *R*/*L* moves forward/backward between containers. (h) *BOB* opens the target recipe. (i) *B-L-*/*B-R-* moves backward/forward between the running apps. (j) *BOB* selects the target music.

6 POTENTIAL USAGE

We present three applications to demonstrate the usage of eyelid gestures (see the supplemental video for more details).

6.1 App Navigation

Mobile app navigation happens at three levels: between *apps*, between *tabs/screens*, and between *containers*. *Tab* is a common approach to organize content in an app (e.g., Fig. 4 d and e are two tabs of an app). *Screen* is another way of organizing content in an app, usually in the App launcher (e.g., Fig. 4 a and b are two screens). Within a *tab*, content is further organized by *containers*, often visually presented as cards (e.g., each row in Fig. 4 d is a container). We design a *gesture mapping scheme* between eyelid gestures and three levels of navigation (Fig. 4). Atomic gestures *R* and *L* are used to navigate *forward* and *backward* between *containers* in a *tab* or a *screen*, which is the lowest-level navigation. Atomic gestures *R-* and *L-* are used to navigate forward and backward between tabs or screens within an app, which is the middle-level navigation. Compound gestures *B-R-* and *B-L-* are used to navigate forward and backward between apps, which is the highest-level navigation. We followed two design guidelines: actions on the right/left eyelid navigate forward/backward, following English reading convention, and the complexity of the eyelid gestures increases from the lowest-level to the highest-level navigation as navigating between *apps* causes the most significant overhead [25].

6.2 Cross-application Interactions

Mobile users often use more than one app simultaneously and need to switch repeatedly between these apps [3]. Such back and forth app switching may introduce a significant overhead [25]. Inspired by porous interfaces that enable partially transparent app windows overlaid on top of each other and allow for interacting with each app with a designated finger [10], we propose *iWink porous interfaces* that allows users to quickly switch the order of two apps by closing or opening an eyelid. As Fig. 5a-b shows, the user brings the background app (i.e., map) to the foreground by closing any eyelid, interacts with it with any finger and resumes the order by opening two eyelids.

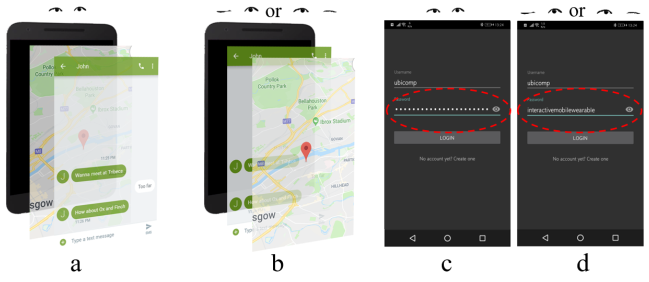


Figure 5: (a) *Opening both eyelids* allows for interacting with the foreground app with all fingers; (b) *closing any eyelid* allows for interacting with the background app with all fingers; (c) opening all eyelids allows for typing with the password hidden; (d) closing any eyelid temporarily reveals sensitive content to minimize shoulder-surfing attack.

6.3 Sensitive Interactions

People may need to perform sensitive interactions, such as typing passwords, in public settings (e.g., a crowded subway), and thus they may suffer from should-surfing attacks. Common approaches, such as hiding passwords while typing, take the visual feedback away. Consequently, users may commit typos and only realize it after submitting the wrong password. This may risk them from being locked out with too many failed attempts. With *iWink*, users can temporally reveal the password while typing anytime by simply closing an eye (Fig. 5d). Opening both eyes resumes the normal mode to keep the password secure and minimize should-surfing attacks (Fig. 5c).

7 LIMITATIONS AND FUTURE WORK

While our work shows the promise of eyelid gestures for scenarios where using both hands is inconvenient, our user study was conducted in one indoor environment with a limited number of participants (N=12) for a short period. Future work should examine the robustness of the algorithm in different environments for longer periods, such as via an in-the-wild deployment study, with more participants. As not all people can control their two eyelids at the same level of ease [31], it is important to design personalized recognition models. Similar to previous research that combines gaze with other modalities (e.g., [18, 33–35, 42, 43]), future research should combine eyelid gestures with gaze or other modalities (e.g., mouth-, cheek-, and ear-based inputs [1, 8, 28, 41]). Further, our participants did not report they felt fatigued as the study was relatively short. However, it is worth exploring the issues associated with long-term use of eyelid gestures (e.g., fatigue). Moreover, it is also worth exploring the applications of eyelid gestures for people with motor-impairments, such as those with limited arm or hand dexterity. Indeed, we recently started to explore how well the algorithm could detect eyelid gestures performed by people with motor impairments and how they feel about using eyelid gestures [6]. Finally, in addition to supporting hands-free interactions for mobile devices, iWink could also be integrated into smart TV to allow for remotely controlling it.

8 CONCLUSION

We have presented the taxonomies to construct eyelid gestures based on four primitive eyelid states, designed an algorithm to detect nine eyelid gestures on a smartphone in real-time, and evaluated the algorithm with twelve able-bodied users in *user-dependent* and *user-independent* studies. Results show that *user-independent* classifier can detect nine eyelid gestures for a *new* user with 78% overall accuracy and that *user-dependent* classifier could achieve 83% overall accuracy with only five samples per gesture from a user. Moreover, we have demonstrated the usage of eyelid gestures in app navigation, cross-app interactions, and sensitive interactions.

ACKNOWLEDGMENTS

The authors would like to acknowledge support from members of DGP lab. In particular, we would like to thank John Hancock for his help in preparing the demo video.

REFERENCES

[1] Toshiyuki Ando, Yuki Kubo, Buntarou Shizuki, and Shin Takahashi. 2017. Canalsense: Face-related movement recognition system based on sensing air pressure in ear canals. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 679–689.

[2] Behrooz Ashtiani and I Scott MacKenzie. 2010. BlinkWrite2: an improved text entry method using eye blinks. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*. 339–345.

[3] Matthias Böhmer, Brent Hecht, Johannes Schöning, Antonio Krüger, and Gernot Bauer. 2011. Falling asleep with Angry Birds, Facebook and Kindle. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services - MobileHCI '11*. ACM Press, New York, New York, USA, 47. https://doi.org/10.1145/2037373.2037383

[4] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze Interaction for Smart Watches using Smooth Pursuit Eye Movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*. ACM Press, New York, New York, USA, 457–466. https://doi.org/10.1145/2807442.2807499

[5] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 457–466.

[6] Mingming Fan, Zhen Li, and Franklin Mingzhe Li. 2020. Eyelid Gestures on Mobile Devices for People with Motor Impairments. In *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'20)*. ACM, In Press. https://doi.org/10.1145/3373625.3416987

[7] Nathan Gitter. 2017. Rainbrow: an eyebrow-controlled game. https://apps.apple.com/us/app/rainbrow/id1312458558. (Accessed on 08/04/2020).

[8] Mayank Goel, Chen Zhao, Ruth Vinisha, and Shwetak N Patel. 2015. Tongue-in-cheek: Using wireless signals to enable non-intrusive and flexible facial gestures detection. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 255–258.

[9] Kristen Grauman, Margrit Betke, Jonathan Lombardi, James Gips, and Gary R Bradski. 2003. Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces. *Universal Access in the Information Society* 2, 4 (2003), 359–373.

[10] Aakar Gupta, Muhammed Anwar, and Ravin Balakrishnan. 2016. Porous Interfaces for Small Screen Multitasking using Finger Identification. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. ACM Press, New York, New York, USA, 145–156. https://doi.org/10.1145/2984511.2984557

[11] Henna Heikkilä and Kari-Jouko Räihä. 2012. Simple gaze gestures and the closure of the eyes as an interaction technique. In *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA '12*. ACM Press, New York, New York, USA, 147. https://doi.org/10.1145/2168556.2168579

[12] Fabian Hemmert, Danijela Djokic, and Reto Wettach. 2008. Perspective change: a system for switching between on-screen views by closing one eye. In *Proceedings of the working conference on Advanced visual interfaces - AVI '08*. ACM Press, New York, New York, USA, 484. https://doi.org/10.1145/1385569.1385668

[13] Seongkook Heo, Michelle Annett, Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2017. No Need to Stop What You're Doing: Exploring No-Handed Smartwatch Interaction. In *Graphics Interface*. Canadian Human-Computer Communications Society, Waterloo, Ontario, Canada, 107–114. https://doi.org/10.20380/gi2017.14

[14] Shoya Ishimaru, Kai Kunze, Koichi Kise, Jens Weppner, Andreas Dengel, Paul Lukowicz, and Andreas Bulling. 2014. In the Blink of an Eye: Combining Head Motion and Eye Blink Frequency for Activity Recognition with Google Glass. In *Proceedings of the 5th Augmented Human International Conference* (Kobe, Japan) *(AH '14)*. ACM, New York, NY, USA, Article 15, 4 pages. https://doi.org/10.1145/2582051.2582066

[15] Robert JK Jacob. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 11–18.

[16] Ricardo Jota and Daniel Wigdor. 2015. Palpebrae Superioris: Exploring the Design Space of Eyelid Gestures. In *Proceedings of the 41st Graphics Interface Conference*. Canadian Human-Computer Communications Society, Toronto, Ontario, Canada, 3–5. https://doi.org/10.20380/GI2015.35

[17] Arie E Kaufman, Amit Bandopadhay, and Bernard D Shaviv. 1993. An eye tracking computer user interface. In *Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium*. IEEE, 120–121.

[18] Mohamed Khamis, Mariam Hassib, Emanuel von Zezschwitz, Andreas Bulling, and Florian Alt. 2017. GazeTouchPIN: protecting sensitive data on mobile devices using secure multimodal authentication. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. 446–450.

[19] Jinhyuk Kim, Jaekwang Cha, Hojun Lee, and Shiho Kim. 2017. Hand-free Natural User Interface for VR HMD with IR Based Facial Gesture Tracking Sensor. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*

(Gothenburg, Sweden) *(VRST '17)*. ACM, New York, NY, USA, Article 62, 2 pages. https://doi.org/10.1145/3139131.3143420

[20] Piotr Kowalczyk and Dariusz Sawicki. 2019. Blink and wink detection as a control tool in multimodal interaction. *Multimedia Tools and Applications* 78, 10 (2019), 13749–13765.

[21] Aleksandra Królak and Paweł Strumiłło. 2012. Eye-blink detection system for human–computer interaction. *Universal Access in the Information Society* 11, 4 (2012), 409–419.

[22] Pin-Sung Ku, Te-Yen Wu, Ericka Andrea Valladares Bastias, and Mike Y. Chen. 2018. Wink It: Investigating Wink-based Interactions for Smartphones. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct* (Barcelona, Spain) *(MobileHCI '18)*. ACM, New York, NY, USA, 146–150. https://doi.org/10.1145/3236112.3236133

[23] Pin-Sung Ku, Te-Yan Wu, and Mike Y. Chen. 2017. EyeExpression: Exploring the Use of Eye Expressions As Hands-free Input for Virtual and Augmented Reality Devices. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology* (Gothenburg, Sweden) *(VRST '17)*. ACM, New York, NY, USA, Article 60, 2 pages. https://doi.org/10.1145/3139131.3141206

[24] S.H. Kwon and H.C. Kim. 1999. EOG-based glasses-type wireless mouse for the disabled. *Proceedings of the First Joint BMES/EMBS Conference. 1999 IEEE Engineering in Medicine and Biology 21st Annual Conference and the 1999 Annual Fall Meeting of the Biomedical Engineering Society (Cat. No.99CH37015)* 1 (1999), 592. https://doi.org/10.1109/IEMBS.1999.802670

[25] Luis Leiva, Matthias Böhmer, Sven Gehring, and Antonio Krüger. 2012. Back to the app: the costs of mobile application interruptions. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services - MobileHCI '12*. ACM Press, New York, New York, USA, 291. https://doi.org/10.1145/2371574.2371617

[26] Jingjing Liu, Bo Liu, Shaoting Zhang, Fei Yang, Peng Yang, Dimitris N Metaxas, and Carol Neidle. 2013. Recognizing eyebrow and periodic head gestures using CRFs for non-manual grammatical marker detection in ASL. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*. IEEE, 1–6.

[27] Google LLC. 2019. Mobile Vision | Google Developers. https://developers.google.com/vision/

[28] Michael J Lyons, Chi-Ho Chan, and Nobuji Tetsutani. 2004. Mouthtype: Text entry by hand and mouth. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*. 1383–1386.

[29] I Scott MacKenzie and Behrooz Ashtiani. 2011. BlinkWrite: efficient text entry using eye blinks. *Universal Access in the Information Society* 10, 1 (2011), 69–80.

[30] Emiliano Miluzzo, Tianyu Wang, and Andrew T. Campbell. 2010. EyePhone: activating mobile phones with your eyes. In *Proceedings of the second ACM SIGCOMM workshop on Networking, systems, and applications on mobile handhelds - MobiHeld '10*. ACM Press, New York, New York, USA, 15. https://doi.org/10.1145/1851322.1851328

[31] Lyelle L Palmer. 1976. Inability to wink an eye and eye dominance. *Perceptual and motor skills* 42, 3 (1976), 825–826.

[32] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[33] Ken Pfeuffer, Jason Alexander, Ming Ki Chong, and Hans Gellersen. 2014. Gaze-touch: combining gaze with multi-touch for interaction on the same surface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 509–518.

[34] Ken Pfeuffer, Jason Alexander, Ming Ki Chong, Yanxia Zhang, and Hans Gellersen. 2015. Gaze-shifting: Direct-indirect input with pen and touch modulated by gaze. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 373–383.

[35] Marcos Serrano, Barrett M Ens, and Pourang P Irani. 2014. Exploring the use of hand-to-face input for interacting with head-worn displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 3181–3190.

[36] Robin Shaw, Everett Crisman, Anne Loomis, and Zofia Laszewski. 1990. The eye wink control interface: using the computer to provide the severely disabled with increased flexibility and comfort. In *Proc. of the Third Annual IEEE Symposium on Computer-Based Medical Systems*. IEEE, 105–111. https://doi.org/10.1109/CBMSYS.1990.109386

[37] John A Stern, Larry C Walrath, and Robert Goldstein. 1984. The endogenous eyeblink. *Psychophysiology* 21, 1 (1984), 22–33.

[38] Roel Vertegaal, Aadil Mamuji, Changuk Sohn, and Daniel Cheng. 2005. Media eyepliances: using eye tracking for remote control focus selection of appliances. In *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. 1861–1864.

[39] Mélodie Vidal, Andreas Bulling, and Hans Gellersen. 2013. Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. 439–448.

[40] Bryan Wang and Tovi Grossman. 2020. BlyncSync: Enabling Multimodal Smartwatch Gestures with Synchronous Touch and Blink. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–14.

[41] Xuhai Xu, Chun Yu, Anind K Dey, and Jennifer Mankoff. 2019. Clench Interface: Novel Biting Input Techniques. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[42] Koki Yamashita, Takashi Kikuchi, Katsutoshi Masai, Maki Sugimoto, Bruce H Thomas, and Yuta Sugiura. 2017. CheekInput: turning your cheek into an input surface by embedded optical sensors on a head-mounted display. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. 1–8.

[43] Shumin Zhai, Carlos Morimoto, and Steven Ihde. 1999. Manual and gaze input cascaded (MAGIC) pointing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 246–253.

[44] Xiaoyi Zhang, Harish Kulkarni, and Meredith Ringel Morris. 2017. Smartphone-Based Gaze Gesture Communication for People with Motor Disabilities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 2878–2889. https://doi.org/10.1145/3025453.3025790