EyeMU Interactions: Gaze + IMU Gestures on Mobile Devices

Andy Kong
Carnegie Mellon University
Pittsburgh, PA, USA
akong@andrew.cmu.edu

Karan Ahuja
Carnegie Mellon University
Pittsburgh, PA, USA
kahuja@cs.cmu.edu

Mayank Goel
Carnegie Mellon University
Pittsburgh, PA, USA
mayank@cs.cmu.edu

Chris Harrison
Carnegie Mellon University
Pittsburgh, PA, USA
chris.harrison@cs.cmu.edu

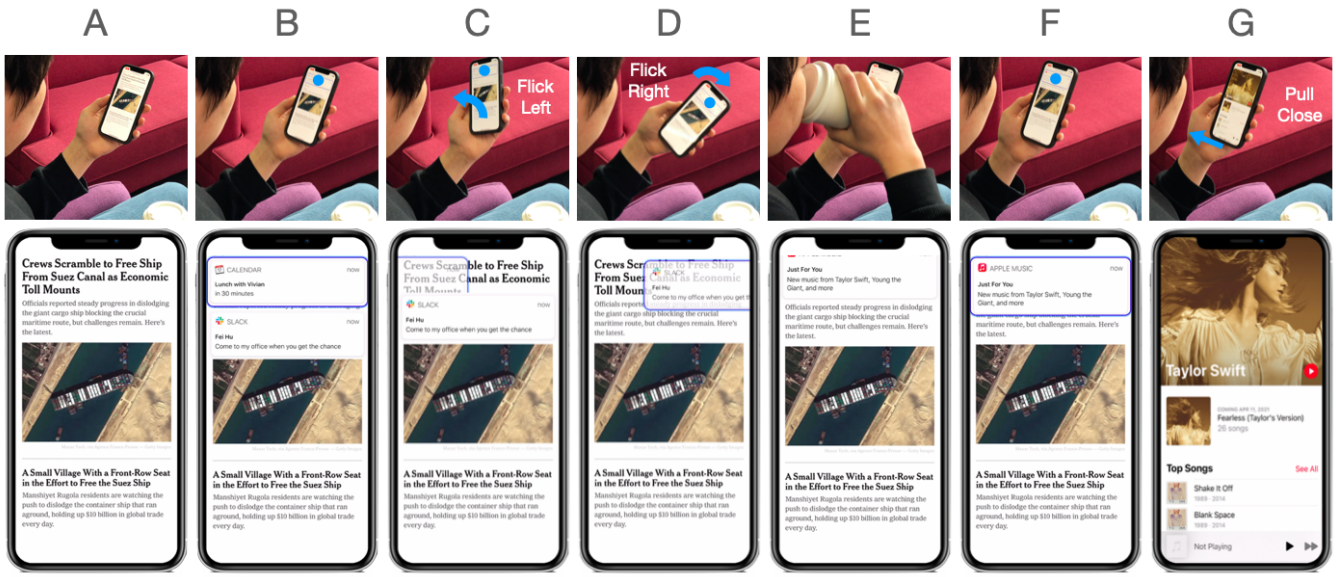

Figure 1: A user is reading the news (A) when two notifications arrive (B). They read the top one and flick left to dismiss it (C). The user then reads the other notification and snoozes it with a flick right (D). The user takes a sip of their coffee, just as a third notification arrives (E). Upon reading it and wanting to see more (F), the user pulls their phone closer (G) to launch the associated app. Throughout, the user has never had to touch the screen.

ABSTRACT

As smartphone screens have grown in size, single-handed use has become more cumbersome. Interactive targets that are easily seen can be hard to reach, particularly notifications and upper menu bar items. Users must either adjust their grip to reach distant targets, or use their other hand. In this research, we show how gaze estimation using a phone's user-facing camera can be paired with IMU-tracked motion gestures to enable a new, intuitive, and rapid interaction technique on handheld phones. We describe our proof-of-concept implementation and gesture set, built on state-of-the-art techniques and capable of self-contained execution on a smartphone. In our user study, we found a mean euclidean gaze error of 1.7 cm and a seven-class motion gesture classification accuracy of 97.3%.

ICMI '21, October 18–22, 2021, Montréal, QC, Canada

ACM ISBN 978-1-4503-8481-0/21/10.
https://doi.org/10.1145/3462244.3479938

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**; **Interaction techniques**; **Smartphones**; *Human computer interaction (HCI).*

KEYWORDS

Smartphone camera; computer vision; inertial measurement unit; accelerometer; gyroscope; gesture recognition; eye tracking.

ACM Reference Format:
Andy Kong, Karan Ahuja, Mayank Goel, and Chris Harrison. 2021. EyeMU Interactions: Gaze + IMU Gestures on Mobile Devices. In *Proceedings of the 2021 International Conference on Multimodal Interaction (ICMI '21), October 18–22, 2021, Montréal, QC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3462244.3479938

1 INTRODUCTION

As smartphone screens have grown in size, single-handed use has become more cumbersome. It is not uncommon to have interactive targets that are impossible to reach unless the user adjusts their grip. No doubt many millions of smartphones have been dropped while performing such a grip change, especially if the user is walking or performing another action. For this reason, many users have added aftermarket rings and grips to the rear of their phones, adding weight and bulk to devices that we wish to be as thin as possible.

We present *EyeMU interactions*, a set of intuitive and rapid gestural actions that can be used on mobile phones, powered by a combination of state-of-the-art gaze estimation and IMU-tracked motion gestures. Importantly, our interactions require no grip change or touch input. To avoid false positive and accidental activations, as well as eliminate unnecessary computation, our system only activates when a series of conditions are met. First, a user must be present in the camera's view, then attend to the screen, then fixate on a widget, and finally, while maintaining that fixation, perform a motion gesture. We note our technique is highly complementary with conventional touch input, and can serve to alleviate reach issues, as well as expose advanced functionality typically buried in long presses and menus.

Consider the following interaction sequence: while scrolling with a thumb through an online news article (Figure 1A), a user receives a calendar and slack notification. Rather than attempting to reach these distant interactors at the top of the screen, the user simply stares at the calendar notification (Figure 1B) and performs a left flick of the phone to dismiss it (Figure 1C). For the slack notification, the user performs a right flick to snooze the alert (Figure 1D). After sometime (Figure 1E), a music notification appears. The user stares at this target (Figure 1F) and pulls the device closer, launching the full screen application to get more details (Figure 1G). When finished, the user can push the phone back to return to reading their webpage, having never touched the screen. In this way, gaze + motion gestures provide a convenient and single-handed way to trigger events, especially for out-of-reach targets. Even when the interactor is within finger reach, users could opt to use gaze + motion gestures as a way to access more complex functionality vs a long press + context menu.

After reviewing related work, we describe our proof-of-concept implementation and gesture set, built on state-of-the-art techniques drawn from the literature. Unlike most prior smartphone gaze tracking systems, we do not calibrate per user and our process runs entirely self-contained on a phone with real-time performance. We use our pipeline to power a series of example demos, and conclude with a user study that found a mean euclidean gaze estimation error of 1.7 cm and motion gesture recognition accuracy of 97.3%.

2 RELATED WORK

Our work broadly intersects with research on gaze estimation on smartphones, as well as IMU-based gesture recognition on mobile devices. Most related are multimodal systems that combine gaze with another input channel, which we review in greater detail.

2.1 Gaze Estimation on Smartphones

Gaze estimation is a long-standing research topic spanning many domains, such as computer vision [2, 28, 31], graphics [57, 64] and human-computer interaction [3, 40, 41]. Apparatus for tracking gaze can either be worn, as is the case with a head-mounted eye tracker [3, 54, 56], or external, such as cameras in the environment [2, 28]. With the advent of deep learning and widespread prevalence of high-quality cameras, image-based gaze estimation approaches are becoming increasingly popular.

Gaze estimation using cameras span many approaches, such as dedicated IR eye-trackers [54], depth cameras [17], and RGB cameras [2, 28, 59]. Most heuristic [15, 37] and machine learning [30, 46] approaches utilizing cameras can be separated into two categories: model-based [51] and appearance-based [29, 46]. While model-based approaches build geometric models to exploit the inherent structure of the eyes (such as tracking the contours of the iris), appearance based methods rely on a supervised training approach ingesting raw images of the eyes as input.

Due to their ubiquity and high-resolution, modern smartphone cameras have also been a recent testing ground for both model- and appearance-based gaze estimation approaches [19, 28]. Prior work in smartphone model-based approaches has utilized infrared cameras, which offer images with better contrast to achieve error rates as low as 1° [8, 31]. However, these require an instrumented device with IR LEDs and a special wide-angle IR camera, thus making them less practical and limited in adoption [9].

In contrast, appearance-based models have had more success, spurred in part by the release of large datasets such as the GazeCapture dataset [28], a 2.4 million image dataset for eye-tracking on mobile devices. The authors of this dataset also created iTracker, a CNN-based approach that achieved 1.74 cm without calibration (1.34 cm with calibration). Attempts to surpass iTracker accuracy have utilized training on multiple eye-tracking datasets [25], trained only on task-relevant data [53], or used knowledge distillation frameworks to better generalize [19]. A recent breakthrough is the architecture of Valliappan et al. [55], which achieves an uncalibrated accuracy of 1.92 cm, which drops to 0.5 cm with a per-user calibration session. This calibrated model's error was comparable with a commercial head-worn eye tracker, and represents the current state-of-the-art result in smartphone eye tracking. We adapt this architecture in our approach, adding information about the user's head orientation to create a generalizable model that does not require each user to calibrate before use.

2.2 Mobile IMU-Based Gesture Recognition

IMU-based sensing on smartphones has spawned a plethora of applications ranging from activity recognition [16, 44], inertial odometry [12, 49], gestural input [20, 32, 58], behavioral authentication [42, 60] and accessible user interfaces [63]. As the complete review of all these applications is outside the scope of this paper, we focus on approaches employing motion-based gesture recognition for interactive applications.

Motion gesture segmentation and recognition is now a well understood problem in HCI. In their seminal work, Hinckley et al. explored applications for on-device accelerometers such as tilt to change orientation and tilt based scrolling [22]. Since then, a variety of other mobile device gestures have been explored, including taps [23, 47], squeezes [21], tilts [13], and in-air gestures [38, 50]. These approaches share a similar workflow – there is a simple gesture trigger detector, followed by a gesture classifier that is either based on heuristics [39] or machine learning [26]. We follow a similar approach and draw our gesture set from this literature.

2.3 Combining Gaze With Other Modalities

Given the natural utility of gaze to specify a target, researchers have looked at combining it with various sensing modalities to enable novel interactions. This includes gaze coupled with speech

[34], touch [43], hand gestures [10], electromyography [33], facial gestures [5], and foot contact [27], among many more. Past work has also separately compared gaze interaction against other techniques including motion and touchscreen gestures, but not together [14]. This has enabled a variety of applications such as modelling human communication dynamics [36, 52], remote target selection [48, 62], and human-robot interaction [1, 35].

While many approaches exist, most common is the combination of gaze with touch for enhanced selection and spatial manipulation [43, 45, 61]. Works by Chatterjee and Carter have expanded the interactivity from touch based events to free space gestures, combining gaze with hand gesture recognition [10, 11]. Another sensing modality pair that stands out is the use of gaze to enable targeted speech interfaces. These include systems such as Put-That-There [7], WorldGaze [34], DoV [4], and [48].

While combinations of gaze and other modalities have been explored, to the best of our knowledge, the interactions afforded by combining gaze and IMU-based gestures on smartphones remains unexplored. This can be attributed to low-gaze accuracy on smartphones until very recently. Our implementation combines gaze-tracking with motion gestures to produce an interaction that is both reactive and discreet, recovering finer-grained device control while retaining the usability of the device.

3 SYSTEM AND IMPLEMENTATION

We built our proof-of-concept implementation on an Apple iPhone 12 Pro (screen size is 12.8 × 6.4 cm), which features a front-facing camera resolution of 12 MP (4000 × 3000 px). For rapid prototyping, EyeMU is a javascript application that runs in real-time in the Safari browser on Apple iOS devices. As we prototyped EyeMU, we validated its performance across multiple iPhone models. Since our implementation is built in Javascript and runs entirely on the client side, EyeMU can theoretically run on any smartphone with a Javascript-enabled browser. However, we envision a commercial implementation of EyeMU as a compiled OS-level feature available through an API to any application wishing to incorporate gaze + motion interactions.

3.1 User Presence & Screen Attention Detection

EyeMU takes a multi-step approach to enable gaze + motion interactions. As a first step, it detects whether a user's face is present in the camera view, and then whether the user's visual attention is on the screen. We use MediaPipe's Face Mesh [18] model to calculate face orientation. The model gives a face detection score for each frame, and further, is able to output predictions even when user's face is partially off-camera. Once a user is detected, we use the output facial landmarks from Face Mesh (in 3D phone coordinates) to extract head size, images of both eyes, and eye location within the frame. EyeMU also uses the facial landmarks to calculate the yaw, pitch, and roll off the head. If head yaw and pitch are within 30 and 20 degrees, respectively, we assume the user is attending to the screen and continue to the next step of our pipeline.

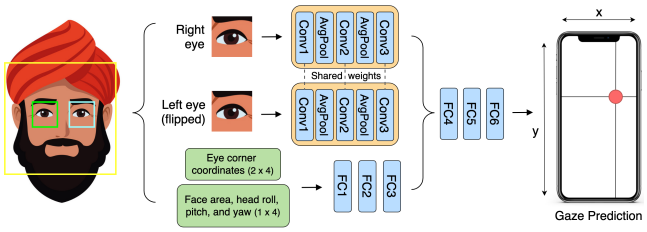

Figure 2: Architecture of EyeMU's gaze estimation CNN.

3.2 Gaze Estimation

The next step is to estimate where the user's eyes are looking on the screen. The core of our gaze estimation pipeline is a Convolutional Neural Network (CNN) adapted from the state-of-the-art eye tracking model presented by Valliappan et al. [55], chosen for achieving the lowest error after calibration. The model also achieves a low error without any per-user training, making it ideal for our use. We modified their architecture, adding information received from MediaPipe[18] about the user's head orientation and size (Figure 2). We used TensorFlow to train this modified CNN model from scratch using data from the GazeCapture dataset[28], and achieved accuracies comparable to those reported by Valliappan et al.

Once trained, the CNN takes the left and right eye crops as input and estimates the 2D estimate of the gaze in screen coordinate space. The normalized coordinates of users' eye corners and their head angles are then added before the fully connected layers of the CNN. To run our base model on-device, we converted the CNN into a TensorFlow JS model, allowing us to prototype interactions in a browser. On our iPhone 12 Pro running iOS v14.4, the base model run alone processes video frames at 20 Hz.

Even though our model should work across users, there are nonetheless device-specific camera intrinsics and a diversity of screen sizes we must account for. To calibrate our model, we added a small, in-browser model training routine. Data to perform this fitting is captured from the front-facing camera while the phone is in use. We first create a gaze feature vector representing each frame. Each feature vector includes the outputs of the final three layers of our CNN, to which we append the following facial features extracted from the face mesh: head yaw, pitch, and roll, the area of the face with respect to the frame, and the on-screen coordinates of the left and right eye corners. As we describe further in our user study section, the resulting 26-feature vector is constantly collected during our user study in order to train a device-calibrated Support Vector Regressor.

3.3 Gaze Target Fixation Detection

We use a straightforward heuristic for determining target fixation. We maintain a 500 ms rolling window of gaze location estimates; if all points reside within a 2.5 cm diameter circular region, we consider the user to be fixated on that small region of the screen. In cases where we have access to the UI widget hierarchy (such as in our demo apps), we use the physical bounds of the widget itself (*i.e.*, as the "hit box"). These widgets are usually larger in surface area than the 2.5 cm diameter circle. For instance, the notifications bar on top of the iPhone 12 Pro in iOS v14.4 is 1.9 × 6.0 cm (Figure 1).

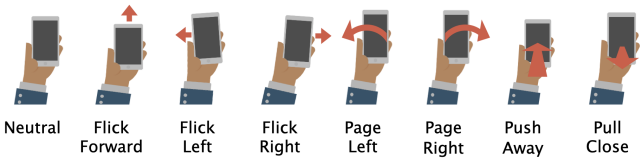

Figure 3: The IMU-driven motion gesture set we employed.

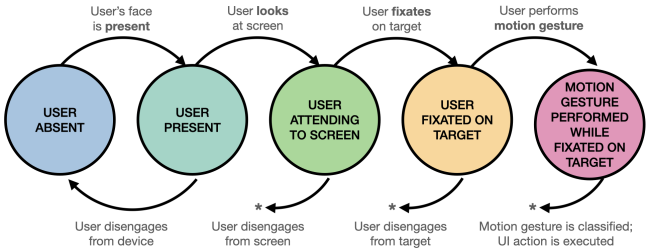

Figure 4: To mitigate the Midas Touch problem, we use a highly gated approach, illustrated above, to determine when to trigger UI events.

3.4 Motion Gesture Detection & Classification

Once EyeMU detects a gaze fixation, it waits and listens for a motion gesture. We observe a noticeable shift in the RMS values of the IMU when the smartphone transitions from no motion to the start of a motion gesture, and conversely transitions from the end of a motion gesture to no motion. We find that a simple threshold over the RMS of IMU values (raw linear acceleration and angular velocity values) over a window of 200 ms is sufficient to segment the start and end of a motion gesture. We use the IMU's output to train a motion gesture classifier.

We extract features from the IMU values per-axis (the accelerometer and gyroscope raw values sampled at 60 Hz) and feed it as training data to a Support Vector Classifier. We treat each IMU axis history over one trial as an independent time series and compute the following features: the minimum, mean, maximum, range, and standard deviation of the time series and the first two coefficients of a quadratic fit to the time series. Features are calculated for each axis are then concatenated into a common feature vector and flattened, resulting in a feature vector of 72 elements for each gesture trial. The set of seven motion gestures we use is shown in Figure 3 (plus a neutral pose).

3.5 Midas Touch & False Positive Mitigation

Gaze interactions can suffer from the Midas Touch problem if not carefully designed [24]. To mitigate this significant issue, our approach relies on a series of cascading condition checks (Figure 4) such that our classifiers only run when needed. This not only reduces false positives, but also power consumption. Our first check is to determine if there is even a user present in the view of the user-facing camera. If no face is detected, our entire pipeline sleeps. If there is a face present, we next determine if the user is attending to on-screen content or looking elsewhere. For this, we run our gaze estimation pipeline. If the user is determined to be looking at the screen, we enter the "user attending to screen" state. We now maintain a rolling history of live gaze estimations (in our study, we found an ideal window is around 500ms), on which our gaze fixation detector runs. If target fixation is detected, we move into the "user fixated on target" state. Only now does our motion gesture classifier begin to run using live IMU data. If the user disengages from the target, screen or device, the pipeline immediately falls back to an earlier state and motion gesture detection is turned off. However, if the user performs a motion gesture while also fixated on a target, the corresponding event handler will be triggered.

We found this highly-gated and double-trigger approach to be an effective means of reducing false positive triggers, which as noted in much prior work generally carries a higher user cost than missing true positive events [6]. The simple fact is that users are constantly gazing at their screens and constantly moving as well (especially while e.g., walking), and so the mere presence of screen gaze and hand motion is insufficient for robust operation. Moreover, face and gaze tracking is not 100% accurate. While reading news stories or scrolling through photos on a smartphone, it should be expected there will be periodic false positive fixation events, even when using a long window of gaze history. However, in order for there to be a falsely triggered UI event, the user would have to also perform a motion-gesture-like movement while such an error was occurring. The likelihood of co-occurring errors is relatively low.

Put simply, if we can keep the false positive trigger rate of our gaze-fixation detection and motion-gesture detection down to say 5% each (i.e., 95% true positive accuracy, similar to our study results), we can expect a system false-positive trigger rate of just 0.25%, as both pipelines have to fail in order to incorrectly trigger an event. Of course, the downside of this double-trigger arrangement is that our true positive accuracy suffers - two 95%-accurate detectors will compound their errors, resulting in 90.25% accuracy. However, as noted previously, there is an asymmetric user cost in errors – false triggers are generally perceived as more costly than true misses – and so we believe this arrangement to be worthwhile in practice.

3.6 Latency & Frame Rate

Our end-to-end pipeline runs at a framerate of 19.1 Hz and has a mean latency of 43.03 ms (SD = 3.1 ms) from captured photo to output gaze prediction. The bulk of the processing time is spent on computing the face mesh and the gaze estimation neural network sequentially, which run independently at 25 Hz and 20 Hz respectively. Our gesture classification module is comparatively faster, running at above the 60Hz data output rate of the smartphone IMU. Even though our gaze estimation model is optimized for smartphones, it does incur a cost on the battery. We find that our full stack can run continuously for around three hours on the iPhone 12 Pro (battery capacity of 10.8 Wh). In practice, our false positive mitigation process (described above) would further boosts battery life since the gaze model need not always be running.

3.7 Open Source Code

To help other researchers explore this domain, the source code for our live pipeline is freely available on Github here.

Figure 5: Email Client: A user is browsing their inbox (A). Spotting an interesting email, the phone is pulled closer to take a peek(B), and then pushed back to close it (C). An email thread's details are unrolled with a right flick (D), and after reading (E), rolled back up with a flick left (F). Finally, an unimportant email is deleted with a page left gesture (H).

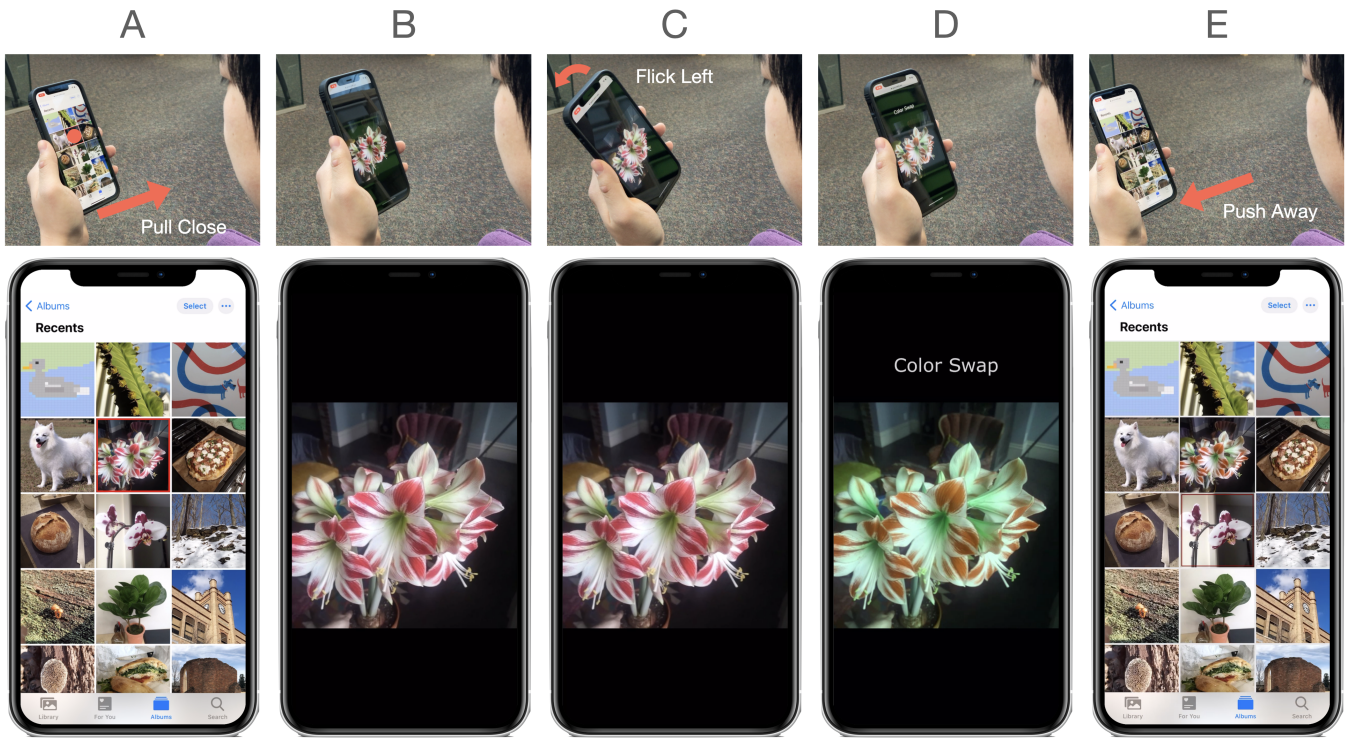

Figure 6: Photo Gallery: while browsing their library, a user decides to look more closely at some flowers (A & B). The user then flicks left to cycle through different filters (C & D). The user then returns to their library by pushing the phone away (E).

4 EXAMPLE APPLICATIONS

To illustrate real-world uses of EyeMU interactions, we built a series of demos. In our introduction, we described one example sequence involving notifications (Figure 1). In this section, we describe three more use cases. In all of these demos, interactions were powered by our live pipeline. Please also see our Video Figure.

New emails most often arrive at the top of the screen, which is the most difficult region to reach in single-handed use. Moreover, a

Figure 7: Home Screen: In this demo, a user looks at the calendar icon on their home screen (A) and pulls the phone closer (B) to open it (C). The calendar is minimized with a push action (D). The user then fixates on the weather icon and pulls the phone closer (E) to open the app (F).

user browsing their inbox can often only open or close an email, with no option to glance at the contents of an email thread. Many of the archive and dismiss options can only be triggered with a swipe that extends across the screen. In our EyeMU-augmented mailbox demo app (Figure 5), emails can be quickly peeked at when pulled closer, or dismissed just as quickly. A thread can also be opened and closed using a flick to the right or left, respectively. Finally, emails can be dismissed and archived using a page left gesture.

In our EyeMU-augmented photo gallery app (Figure 6), a user can fixate on a thumbnail to select it. The photo can then be maximized by pulling their phone closer, or minimized with a push gesture. Filters can be applied to the full-screened photo or fixated thumbnail with left or right flicks.

Finally, peeking into an application from the home screen usually requires a long or forceful press, negating the speedup that motivates peeking. With our EyeMU-augmented home screen (Figure 7), users can peek into applications by pulling their device closer while fixated on the app icon.

5 USER STUDY

We designed our user study to test the three variables that would affect the accuracy of EyeMU interactions, namely: our gaze estimation pipeline, gesture trigger detector and gesture classifier.

We recruited 10 participants (4 male, 6 female, mean age of 21.3) from a local university population. The study lasted approximately one hour and paid $20 in compensation. At the start of the study, the experimenter walked the participants over the study procedure and asked them to perform the various motion gestures once to get them acquainted with the gesture set. The participants were asked to hold the phone in whichever hand felt natural (nine were right-handed) in the portrait orientation, and sit in a comfortable position.

Each session started with the system randomly loading a list or grid view with the corresponding gaze target marked with a red dot. The candidate placement positions for each view can be seen in Figure 8A. These layouts were motivated by our applications (Section 4), which consisted of either a grid view (*e.g.*, photo gallery and home screen) or list view (*e.g.*, email clients and notifications). The grid view exhibited eight potential targets, while the list had six.

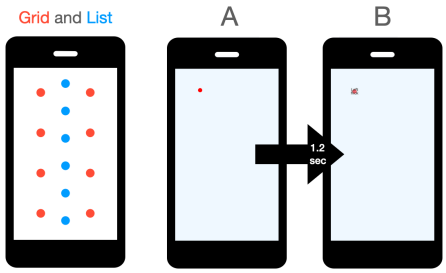

Figure 8: Left: Locations of the target dots for the grid and list rounds of the user study. Right: Every trial of the evaluation featured a gaze fixation phase (A) and motion gesture phase (B). After performing the motion gesture, the participant ended the trial by tapping the screen.

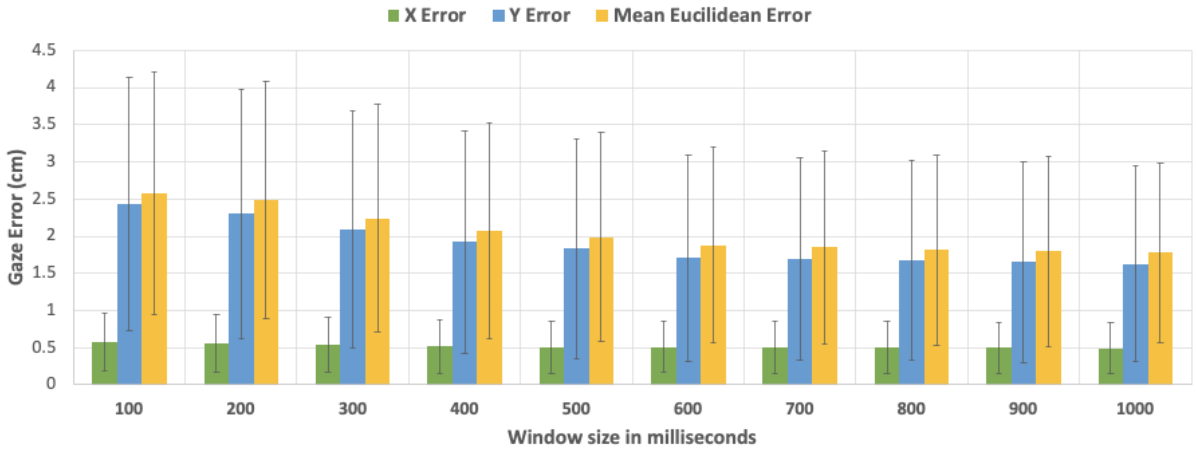

Figure 9: Gaze estimation accuracy of EyeMU. The gaze error is inversely proportional to the window length.

The participants were asked to gaze at a single red dot (Figure 8A). After 1.2 seconds (enough time for the user to fixate), the screen advanced to capture a motion gesture (Figure 8B). Specifically, the user was asked to perform one of the seven gestures at random (Figure 3) using small text overlaid on the dot. After the user performed the gesture, they were asked to tap the screen to end the trial. The experimenter observed users throughout the study and any trials in which a user performed an erroneous gesture was repeated.

Throughout the study, our system logged a timestamped feature vector consisting of raw IMU output, base model embeddings, ground truth gaze (X/Y) location, and the target gesture class. Each participant performed three sets of both grid and list layouts in a randomized order, back to back. Each participant performed every combination of gesture and screen target for each set, logging 294 trials over the course of the study, yielding 2940 EyeMU interactions in total.

6 RESULTS

We evaluated the performance of our gaze estimation model on the user study data we captured. Specifically, we performed a leave-one-user out cross validation. In this process, we train a Support Vector Regressor on the 26 gaze features (Sec 3.2) from nine of our participants and test on the tenth (all combinations, results averaged). Given each gaze trial lasts for 1.2 seconds, we removed the first 200 ms to account for the user's reaction time to fixate on the dot. The remaining 1.0 seconds of data was chunked into 100 ms pieces and concatenated together to create windows of varying duration from 100 to 1000 ms. Shorter windows (*i.e.*, less data) mean less latency, but also generally less accuracy — an trade-off we wished to quantify. Figure 9 shows the accuracy of our gaze model across these different window sizes.

In summary, for our shortest window length of 100 ms, our model has a mean euclidean error of 2.58 cm (SD = 1.6). The mean absolute error along the X direction is 0.58 cm (SD = 0.4) and that along the Y direction is 2.43 cm (SD = 1.7). The error decreases proportional to window size, dropping to 1.78 cm (SD = 1.2) when using 1000 ms of data. For comparison, if we use the Google CNN model out-of-the-box and do not train it for our target device, it has a mean euclidean error of 2.65 cm (SD = 1.6 cm).

The higher error along the Y-direction relative to the X can be attributed to two reasons. The first is that the users hold the phone in portrait orientation, where the screen size along Y (12.8 cm) is twice that of X (6.4 cm), leading to a higher margin for error. Second, the range of pitch for both the head and eye pose is much smaller than their yaw, thus making the changes harder to discern. At a high level, we believe our gaze pipeline is very much usable to identify user fixations and enable gaze + motion based gestures.

To evaluate the accuracy of our model across different motion gestures, we first segmented them using a threshold based heuristic (see Section 3.4) across all of our participants. We found this accuracy to be 100%. Similar to our gaze evaluation, we employ a leave-one-user out cross-validation, training on the data from nine users and testing on the tenth (all combinations, results averaged). We found our mean gesture classification accuracy to be 97.2% (SD=0.04%); the confusion matrix can be seen in Figure 10. We found that almost all gestures worked equally well and there was

	Flick Forward	Flick Right	Flick Left	Pull Close	Push Away	Page Right	Page Left
Flick Forward	1.00	0.01	0.00	0.01	0.01	0.00	0.00
Flick Right	0.00	0.99	0.00	0.00	0.00	0.03	0.00
Flick Left	0.00	0.01	0.99	0.00	0.00	0.00	0.03
Pull Close	0.00	0.00	0.00	0.99	0.00	0.00	0.00
Push Away	0.00	0.00	0.00	0.01	0.99	0.00	0.00
Page Right	0.00	0.00	0.00	0.00	0.00	0.90	0.00
Page Left	0.00	0.00	0.01	0.00	0.00	0.08	0.97

Figure 10: Normalized gesture confusion matrix.

no significant difference in performance based on the handedness of the participants.

7 LIMITATIONS & FUTURE WORK

While the results of EyeMU are promising, there are several key limitations that will need to be overcome before it is ready for consumer use. The first is the accuracy of our gaze module. Even with a state-of-the art mean error of 1.74 cm, it still falls short of the sub-centimeter accuracy offered by dedicated eye trackers for fine grained selection and control. In the future, advances in computer vision and higher-resolution smartphone cameras will only serve to make EyeMU interactions more robust and practical. We also note the current implementation of EyeMU is tested on one device. While its web-app form factor does allow it to run on any device with an IMU and front-facing camera, more testing is required. Additionally, the potential applications of EyeMU on larger devices such as tablets remains to be explored. We believe that on such devices the gaze estimation task may be easier due to increased screen real estate, leading to more substantial eye movements.

Our current prototype's impact on a phone's battery life is significant. EyeMU's execution loop requires the evaluation of three sequential neural networks (for face detection, coarse gaze prediction, and fine-tuned gaze output). Even so, EyeMU is capable of running for three hours continuously on an iPhone 12 Pro. Our proof-of-concept implementation was written in JavaScript, a language not known for its efficiency. With tighter hardware integration (*e.g.*, running the camera at 10 FPS instead of 30) and better hardware acceleration, no doubt significant strides could be made. A commercial implementation could be dramatically more efficient, in the same way "Hey Siri" / "Hey Google" always-on audio detection has been highly optimized. Likewise, intensive computer vision SDKs (e.g., Apple's ARKit) have been made practical through extensive use of special processing hardware, which our approach could also leverage.

Finally, in the future we hope to explore sensor fusion based approaches to bolster the accuracy of EyeMU. For example, using both the RGB and front facing depth camera feeds for gaze prediction. Furthermore, it can also be coupled with adaptive dwell-based interfaces to add a temporal aspect to our gaze + motion gesture interaction paradigms.

8 CONCLUSION

Adding to the literature on multimodal gaze, we have presented our work on EyeMU, a new gaze- and IMU-driven interaction technique and implementation. Our pipeline is built on state-of-the-art techniques, such that we achieve parity to the best prior work in terms of gaze accuracy. In our user study, we found a mean euclidean gaze error of 1.7 cm, sufficiently accurate to allow users to fixate on larger widgets, such as notifications. In the same study, our IMU-driven motion gesture classifier was 97.3% accurate. Our entire software pipeline runs in real time on an iPhone 12 Pro. With future optimizations, we envision our technique running as a background process, allowing for new avenues of gaze-driven interactions that work synergistically with traditional motion input.

REFERENCES

[1] Henny Admoni and Brian Scassellati. 2017. Social eye gaze in human-robot interaction: a review. *Journal of Human-Robot Interaction* 6, 1 (2017), 25–63.

[2] Karan Ahuja, Ruchika Banerjee, Seema Nagar, Kuntal Dey, and Ferdous Barbhuiya. 2016. Eye center localization and detection using radial mapping. In *2016 IEEE International Conference on image processing (ICIP)*. IEEE, Washington D.C., 3121–3125.

[3] Karan Ahuja, Rahul Islam, Varun Parashar, Kuntal Dey, Chris Harrison, and Mayank Goel. 2018. Eyespyvr: Interactive eye sensing using off-the-shelf, smartphone-based vr headsets. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 2 (2018), 1–10.

[4] Karan Ahuja, Andy Kong, Mayank Goel, and Chris Harrison. 2020. Direction-of-Voice (DoV) Estimation for Intuitive Speech Interaction with Smart Device Ecosystems. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 1121–1131.

[5] Anwar Al-Haddad, Rubita Sudirman, Camallil Omar, Koo Yin Hui, and Muhammad Rashid Jimin. 2012. Wheelchair motion control guide using eye gaze and blinks based on pointbug algorithm. In *2012 Third International Conference on Intelligent Systems Modelling and Simulation*. IEEE, Washington D.C., 37–42.

[6] Steve Benford, Holger Schnädelbach, Boriana Koleva, Rob Anastasi, Chris Greenhalgh, Tom Rodden, Jonathan Green, Ahmed Ghali, Tony Pridmore, Bill Gaver, et al. 2005. Expected, sensed, and desired: A framework for designing sensing-based interaction. *ACM Transactions on Computer-Human Interaction (TOCHI)* 12, 1 (2005), 3–30.

[7] Richard A. Bolt. 1980. "Put-That-There": Voice and Gesture at the Graphics Interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*. ACM, New York, NY, USA, 262–270. https://doi.org/10.1145/800250.807503

[8] Braiden Brousseau, Jonathan Rose, and Moshe Eizenman. 2018. Accurate Model-Based Point of Gaze Estimation on Mobile Devices. *Vision* 2, 3 (2018). https://doi.org/10.3390/vision2030035

[9] B. Brousseau, J. Rose, and M. Eizenman. 2018. SmartEye: An Accurate Infrared Eye Tracking System for Smartphones. In *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*. 951–959. https://doi.org/10.1109/UEMCON.2018.8796799

[10] Marcus Carter, Joshua Newn, Eduardo Velloso, and Frank Vetere. 2015. Remote Gaze and Gesture Tracking on the Microsoft Kinect: Investigating the Role of Feedback. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction (OzCHI '15)*. ACM, New York, NY, USA, 167–176. https://doi.org/10.1145/2838739.2838778

[11] Ishan Chatterjee, Robert Xiao, and Chris Harrison. 2015. Gaze+ gesture: Expressive, precise and targeted free-space interactions. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, New York, NY, USA, 131–138.

[12] Changhao Chen, Peijun Zhao, Chris Xiaoxuan Lu, Wei Wang, Andrew Markham, and Niki Trigoni. 2018. Oxiod: The dataset for deep inertial odometry. *arXiv preprint arXiv:1809.07491* (2018).

[13] Raimund Dachselt and Robert Buchholz. 2008. Throw and tilt–seamless interaction across devices using mobile phone gestures. *INFORMATIK 2008. Beherrschbare Systeme–dank Informatik. Band 1* (2008).

[14] Morten Lund Dybdal, Javier San Agustin, and John Paulin Hansen. 2012. Gaze Input for Mobile Devices by Dwell and Gestures. In *Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12)*. ACM, New York, NY, USA, 225–228. https://doi.org/10.1145/2168556.2168601

[15] Y. Ebisawa. 1998. Improved video-based eye-gaze detection method. *IEEE Transactions on Instrumentation and Measurement* 47, 4 (1998), 948–955. https://doi.org/10.1109/19.744648

[16] Shurui Fan, Yating Jia, and Congyue Jia. 2019. A feature selection and classification method for activity recognition based on an inertial sensing unit. *Information* 10, 10 (2019), 290.

[17] Kenneth Alberto Funes Mora, Florent Monay, and Jean-Marc Odobez. 2014. EYEDIAP: A Database for the Development and Evaluation of Gaze Estimation Algorithms from RGB and RGB-D Cameras. In *Proceedings of the ACM Symposium on Eye Tracking Research and Applications* (Safety Harbor, Florida, United States of America). ACM. https://doi.org/10.1145/2578153.2578190

[18] Google. 2021. *MediaPipe Face Mesh*. https://google.github.io/mediapipe/solutions/face_mesh.html Accessed: 2021-04-04.

[19] Tianchu Guo, Yongchao Liu, Hui Zhang, Xiabing Liu, Youngjun Kwak, ByungIn Yoo, Jae-Joon Han, and Changkyu Choi. 2019. A Generalized and Robust Method Towards Practical Gaze Estimation on Smart Phone. *CoRR* abs/1910.07331 (2019). arXiv:1910.07331 http://arxiv.org/abs/1910.07331

[20] Hari Prabhat Gupta, Haresh S Chudgar, Siddhartha Mukherjee, Tanima Dutta, and Kulwant Sharma. 2016. A continuous hand gestures recognition technique for human-machine interaction using accelerometer and gyroscope sensors. *IEEE Sensors Journal* 16, 16 (2016), 6425–6432.

[21] Beverly L Harrison, Kenneth P Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. 1998. Squeeze me, hold me, tilt me! An exploration of manipulative user

interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, New York, NY, USA, 17–24.

[22] Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. 2000. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 91–100.

[23] Scott E Hudson, Chris Harrison, Beverly L Harrison, and Anthony LaMarca. 2010. Whack gestures: inexact and inattentive interaction with mobile devices. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*. ACM, New York, NY, USA, 109–112.

[24] Robert J. K. 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 (CHI '90)*. ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/97243.97246

[25] L. Jigang, B. S. L. Francis, and D. Rajan. 2019. Free-Head Appearance-Based Eye Gaze Estimation on Mobile Devices. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 232–237. https://doi.org/10.1109/ICAIIC.2019.8669057

[26] Minwoo Kim, Jaechan Cho, Seongjoo Lee, and Yunho Jung. 2019. IMU Sensor-Based Hand Gesture Recognition for Human-Machine Interfaces. *Sensors* 19, 18 (2019). https://doi.org/10.3390/s19183827

[27] Konstantin Klamka, Andreas Siegel, Stefan Vogt, Fabian Göbel, Sophie Stellmach, and Raimund Dachselt. 2015. Look & pedal: Hands-free navigation in zoomable information spaces through gaze-supported foot input. In *Proceedings of the 2015 ACM on international conference on multimodal interaction*. ACM, New York, NY, USA, 123–130.

[28] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. 2016. Eye tracking for everyone. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, Washington D.C., 2176–2184.

[29] C. Lai, Y. Chen, K. Chen, S. Chen, S. Shih, and Y. Hung. 2014. Appearance-Based Gaze Tracking with Free Head Movement. In *2014 22nd International Conference on Pattern Recognition*. 1869–1873. https://doi.org/10.1109/ICPR.2014.327

[30] Joseph Lemley, Anuradha Kar, Alexandru Drimbarean, and Peter Corcoran. 2018. Efficient CNN Implementation for Eye-Gaze Estimation on Low-Power/Low-Quality Consumer Imaging Systems. *CoRR* abs/1806.10890 (2018). arXiv:1806.10890 http://arxiv.org/abs/1806.10890

[31] J. Lemley, A. Kar, A. Drimbarean, and P. Corcoran. 2019. Convolutional Neural Network Implementation for Eye-Gaze Estimation on Low-Quality Consumer Imaging Systems. *IEEE Transactions on Consumer Electronics* 65, 2 (2019), 179–187. https://doi.org/10.1109/TCE.2019.2899869

[32] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675.

[33] Julio C. Mateo, Javier San Agustin, and John Paulin Hansen. 2008. Gaze Beats Mouse: Hands-Free Selection by Combining Gaze and Emg. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems (CHI EA '08)*. ACM, New York, NY, USA, 3039–3044. https://doi.org/10.1145/1358628.1358804

[34] Sven Mayer, Gierad Laput, and Chris Harrison. 2020. Enhancing Mobile Voice Assistants with WorldGaze. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/3313831.3376479

[35] Noriaki Mitsunaga, Christian Smith, Takayuki Kanda, Hiroshi Ishiguro, and Norihiro Hagita. 2008. Adapting robot behavior for human–robot interaction. *IEEE Transactions on Robotics* 24, 4 (2008), 911–916.

[36] Louis-Philippe Morency. 2010. Modeling human communication dynamics [social sciences]. *IEEE Signal Processing Magazine* 27, 5 (2010), 112–116.

[37] Carlos H. Morimoto and Marcio R.M. Mimica. 2005. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding* 98, 1 (2005), 4–24. https://doi.org/10.1016/j.cviu.2004.07.010 Special Issue on Eye Detection and Tracking.

[38] Tao Ni, Doug A Bowman, Chris North, and Ryan P McMahan. 2011. Design and evaluation of freehand menu selection interfaces using tilt and pinch gestures. *International Journal of Human-Computer Studies* 69, 9 (2011), 551–562.

[39] K. Noh, D. Lee, and H. Jeong. 2015. Description and recognition based on directional motion vector for spatial hand gestures. In *2015 IEEE SENSORS*. 1–4. https://doi.org/10.1109/ICSENS.2015.7370260

[40] Alexandra Papoutsaki, Aaron Gokaslan, James Tompkin, Yuze He, and Jeff Huang. 2018. The eye of the typer: a benchmark and analysis of gaze behavior during typing. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, New York, NY, USA, 1–9.

[41] Alexandra Papoutsaki, James Laskey, and Jeff Huang. 2017. Searchgazer: Webcam eye tracking for remote studies of web search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*. 17–26.

[42] JunGyu Park, TaeGuen Kim, and Eul Gyu Im. 2016. Touch Gesture Data based Authentication Method for Smartphone Users. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*. 136–141.

[43] 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 (UIST '14)*. ACM, New York, NY, USA, 509–518. https://doi.org/10.1145/2642918.2647397

[44] Wen Qi, Hang Su, Chenguang Yang, Giancarlo Ferrigno, Elena De Momi, and Andrea Aliverti. 2019. A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone. *Sensors* 19, 17 (2019), 3731.

[45] Vijay Rajanna. 2016. Gaze Typing Through Foot-Operated Wearable Device. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '16)*. ACM, New York, NY, USA, 345–346. https://doi.org/10.1145/2982142.2982145

[46] Rajeev Ranjan, Shalini De Mello, and Jan Kautz. 2018. Light-Weight Head Pose Invariant Gaze Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE, Washington D.C.

[47] Sami Ronkainen, Jonna Häkkilä, Saana Kaleva, Ashley Colley, and Jukka Linjama. 2007. Tap input as an embedded interaction method for mobile devices. In *Proceedings of the 1st international conference on Tangible and embedded interaction*. ACM, New York, NY, USA, 263–270.

[48] David Rozado, Louis Stephen, and Navinda Kottege. 2014. Interacting with Objects in the Environment Using Gaze Tracking Glasses and Speech. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design (OzCHI '14)*. ACM, New York, NY, USA, 414–417. https://doi.org/10.1145/2686612.2686676

[49] Arno Solin, Santiago Cortes, Esa Rahtu, and Juho Kannala. 2018. Inertial odometry on handheld smartphones. In *2018 21st International Conference on Information Fusion (FUSION)*. IEEE, Washington D.C., 1–5.

[50] Jie Song, Gábor Sörös, Fabrizio Pece, Sean Ryan Fanello, Shahram Izadi, Cem Keskin, and Otmar Hilliges. 2014. In-air gestures around unmodified mobile devices. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 319–329.

[51] Rainer Stiefelhagen, Jie Yang, and Alex Waibel. 1997. A model-based gaze tracking system. *International Journal on Artificial Intelligence Tools* 6, 02 (1997), 193–209.

[52] Jürgen Streeck. 1993. Gesture as communication I: Its coordination with gaze and speech. *Communications Monographs* 60, 4 (1993), 275–299.

[53] W. Sunhem and K. Pasupa. 2020. A Scenario-based Analysis of Front-facing Camera Eye Tracker for UX-UI Survey on Mobile Banking App. In *2020 12th International Conference on Knowledge and Smart Technology (KST)*. 80–85. https://doi.org/10.1109/KST48564.2020.9059376

[54] Tobii. 2014. *Tobii eye tracker for HTC Vive*. https://blog.tobii.com/eye-tracking-vr-devkit-for-htc-vive-311cbca952df

[55] Nachiappan Valliappan, Na Dai, Ethan Steinberg, Junfeng He, Kantwon Rogers, Venky Ramachandran, Pingmei Xu, Mina Shojaeizadeh, Li Guo, Kai Kohlhoff, et al. 2020. Accelerating eye movement research via accurate and affordable smartphone eye tracking. *Nature communications* 11, 1 (2020), 1–12.

[56] Eric Whitmire, Laura Trutoiu, Robert Cavin, David Perek, Brian Scally, James Phillips, and Shwetak Patel. 2016. EyeContact: scleral coil eye tracking for virtual reality. In *ISWC*. ACM, New York, NY, USA, 184–191.

[57] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. ACM, New York, NY, USA, 131–138.

[58] Jian Wu and Roozbeh Jafari. 2018. Orientation independent activity/gesture recognition using wearable motion sensors. *IEEE Internet of Things Journal* 6, 2 (2018), 1427–1437.

[59] Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. 2015. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755* (2015).

[60] Yulong Yang, Gradeigh D Clark, Janne Lindqvist, and Antti Oulasvirta. 2016. Free-form gesture authentication in the wild. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 3722–3735.

[61] ByungIn Yoo, Jae-Joon Han, Changkyu Choi, Kwonju Yi, Sungjoo Suh, Dusik Park, and Changyeong Kim. 2010. 3D User Interface Combining Gaze and Hand Gestures for Large-Scale Display. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems (CHI EA '10)*. ACM, New York, NY, USA, 3709–3714. https://doi.org/10.1145/1753846.1754043

[62] 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*. ACM, New York, NY, USA, 246–253.

[63] 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*. ACM, New York, NY, USA, 2878–2889.

[64] Xucong Zhang, Seonwook Park, Thabo Beeler, Derek Bradley, Siyu Tang, and Otmar Hilliges. 2020. ETH-XGaze: A Large Scale Dataset for Gaze Estimation under Extreme Head Pose and Gaze Variation. In *European Conference on Computer Vision*. Springer, 365–381.