Distinguishing Between Head and Phone Gestures On a Smartphone With Front-Facing Camera and IMU

James Whiffing
jw204@bath.ac.uk
University of Bath - Department of Computer Science
Bath, England

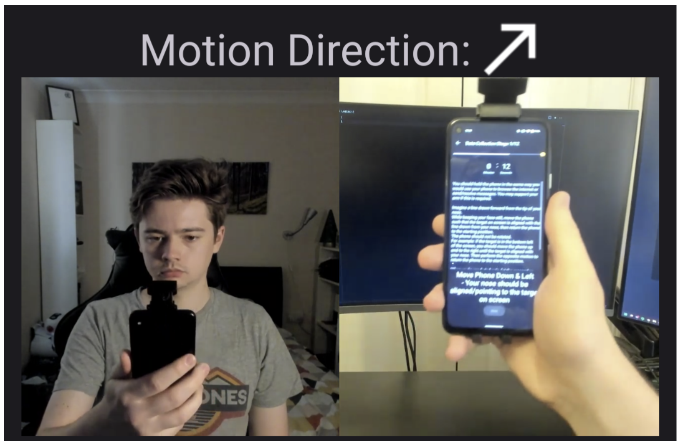

Figure 1: ***TODO: Replace with Image showing diff between head vs phone moving resulting in similar photo (at least head pose)***

ABSTRACT

TODO

ACM Reference Format:
James Whiffing. 2022. Distinguishing Between Head and Phone Gestures On a Smartphone With Front-Facing Camera and IMU. In *Proceedings of (Conference acronym 'XX).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Touchscreens have been the primary interface with which users interact with modern smartphones, either through directly touching UI elements (such as buttons or an on-screen keyboard) or through the use of touch-gestures. Over recent years there has been a trend of smartphone touchscreens increasing in size[33], which does not afford optimal reachability of the full screen for the thumb for interactions[20]. This reduces usability for one-handed interaction, which is a common mode of use[12]

An emerging solution to this is to include head gestures as an additional mode of input, by tracking the head via the smartphone's front-facing camera[5, 7, 9, 11, 27, 32]. A problem with tracking something via a camera that can have a moving Point-of-View (PoV) is that changing the camera's PoV can *look* like movement of the object being tracked. For example, the user moving their phone to the left, will be treated the same as the user moving their head to the right, since the positioning and movement of the head from the front-facing camera's point of view will look the same, see Figure 1. Some papers knew of this issue and accept it as a feature[11], others note it as a known fault to be aware of[7, 30], while others don't indicate whether this has been accounted for[5, 9, 27, 32].

In this paper we look to propose a system that can distinguish between the head or smartphone being moved. In being able to differentiate the two, it should also be able to recognise gestures based on the smartphone movement.

In order to develop a proof-of-concept, a data driven approach was taken. As such a study was performed to collect data: image sequences from the front-facing camera of the smartphone, IMU data from the smartphone, and 3D positioning of the user's head and the smart device via a motion capture system (to provide a ground truth).

With the motion capture data being synced to the IMU data and images, a system could be trained to recognise several gestures and learn to distinguish between whether an observed gesture was due to the smartphone or the user's head moving.

Pending confirmation of system's performance (what was actually delivered/produced) and how it could be extended upon.

Conference acronym 'XX, N/A, N/A

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 LITERATURE REVIEW

Do I need to explain the tech, or just the name is enough, e.g. Haar cascades, CNNs, etc... In this section we will review existing literature to build an understanding of: the gestures we can expect to process, how they may be used and what they mean; Methods with which to obtain data pertaining to the pose of a head; and finally the means with which we can track movement and determine the gesture being performed.

2.1 Gesture Classifications and Usage

Given our goal is to develop a means to distinguish head and phone gestures on smartphone devices, we first need to understand the gesture's we want to recognise and distinguish. Here we will look at existing literature that outline the head and phone gestures you would expect to use while interfacing with a smartphone.

After a review of gestures utilised within Human Computer Interaction literature Karam et al. define five distinct classes with which we can differentiate between types of gestures utilised by the systems proposed in the literature[16]:

Deictic Gestures that involve pointing, and mapping this to either a specific object or location within the interface.
Manipulative A gesture which indicates intent to manipulate some object.
Semaphoric Gestures which map to a specific action or intent.
Gesticulative Typically accompany speech, but not required to transfer meaning.
Language A substitute to written or verbal language.

In our review of head/phone gesture systems we found that none utilised the Language or Gesticulative gesture styles, which is to be expected as we were focusing on gestures for control and interaction rather than for communication. Of the 3 remaining gesture styles, we noted that systems rarely utilised a single gesture style. Either due to the gestures themselves being viewable as multiple gesture styles, being both semaphoric and manipulative, or by actively including different styles of gestures, such as pointing to a region, then using a sempahoric gesture to trigger an action.

An example of this can be seen in the work of Yan et al.[35] who propose nine head gestures (Figure 2), the majority of which are purely Semaphoric, however several (such as scrolling, dragging, and zooming) could also be seen as Manipulative through the mapped action physically moving the content on screen. Yan et al. derived these gestures through a study wherein participants where asked to proposed a set of head gestures, without being given an associated action. These gestures were then collated manually into a set of 80 gestures, which were then effectively voted upon by the participants for their respective actions. The gestures with the most votes for a given action were selected, with some minor adjustments to ensure there were no clashes between actions.

Another system that utilised multiple gestures was EyeMu[18], which outlines several gestures that are performed by physically moving the smartphone, to improve user interaction when the user is forced to interact with phone single handed. As with Yan et al.'s gestures, most are Semaphoric, but can be viewed as manipulative. Some map to actual actions, e.g. flicking between items, others are less derivative and have less of a connection to the desired effect, e.g. moving phone closer/further from face to select an item / open a page.

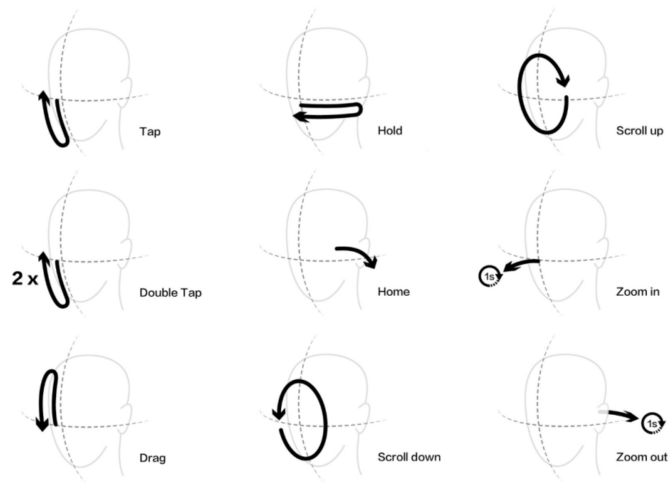

Figure 2: Proposed head gestures and their corresponding actions[35].

Two systems that were purely Deictic are Nouse[9] and a system developed by Varona et al.[30], both of which map the position of the user's nose, within images from the front-facing camera, to a location on the screen. Neither system recognises sequences of motions of the head as gestures, other than recognising blinking and winking, which were recognised as an action to select what was under the cursor. You could also argue that these systems are also manipulative given they show a cursor and as such the head gestures are in an attempt to move the cursor to the relevant location.

One system which could be said to combine all three gesture styles would be the virtual 3D display proposed by López et al.[21]. Their system treats the smartphone screen as a window into a 3D box, where the region of the interior rendered is controlled via the user adjusting the position of their head with respect to the screen. ***Want to use figure from their paper, assume I can include screenshot as long as ref the page num and fig num?*** The tracking of relative positioning of the head to adjust the perspective of what is rendered can be seen as a form of both Deictic and Manipulative gesturing, as the user is looking at different regions within the interior of the virtual box by simply changing where they are looking, but with the intent to adjust the visible interior of the virtual box. The system also provides Semaphoric gestures when interacting with specific programs; in one example they use a browser, with which they can look to the edge of the page to reveal the bookmarks bar.

2.2 Head Localisation

Before being able to distinguish between head and phone gestures, we first need to extract them. To start with we will be reviewing the methods in surrounding literature to extract relevant data required to track head gestures through the use of a smartphone.

The naive approach often taken for finding a face, or more generally a person, within an image is to perform colour segmentation[2, 4, 13], which involves taking an image and filtering the pixels based on a range of colour values that have been chosen as representing skin-tones.

While simple, and given favourable conditions, effective, this approach has several drawbacks:

(1) Detection of objects which have colours that have similar colour and chrominance levels, as noted bybin Abdul Rahman et al.[2].
(2) Determining the values with which to segment the image, i.e. what colours will we accept as skin-tone? During their system evaluation Chan and Abu-Bakar[4] used participants with similar skin-tones to improve the system's robustness.
(3) Dependence on environment lighting.
All three of the papers above[2, 4, 13] do make use of the HSV/yCbCr colour spaces, which make them more robust to changes in lighting intensity, however these systems can still be susceptible to changes in lighting temperature, colour, or shadows.

How would I cite myself for unpublished work? This is what I did in my previous dissertation

A less naive approach is the Viola-Jones algorithm proposed in 2004[31], used in digital cameras, smartphone camera apps, and several head gesture systems[7, 17, 25]. Rather than looking for skin-tone to find faces, it uses the difference of intensity between regions of pixels, and checks if they match a set of templates, Haar-features. These features compare the relative intensity of 2, 3, or 4 neighbouring regions, e.g. is the centre of a region brighter than the regions to the left and right. The algorithm proposed by Viola and Jones uses a degrading cascading classifier[1] to apply these Haar-features on an integral image[2] and will return a bounding box for each face found.

Kim et al. build upon the Viola-Jones algorithm, still utilising Haar features but building their classifier to return the locations of four facial features: left eye, right eye, nose-tip, and mouth, in place of bounding boxes[17].

A more typical approach however is to use the Viola-Jones algorithm to retrieve the bounding box of faces within the image, and then perform further processing to extract facial features[7, 17, 25]. One downside with the algorithm that Viola and Jones note is that it cannot reliably detect faces that are rotated ±15°, while the person is still facing the camera, or ±45°, where the person is facing off to the side of the camera.

Another solution present in the literature, that is invariant to face pose, is the use of depth cameras. In particular we found the use of Apple's ARKit framework used alongside the front-facing depth camera (on supported iOS devices)[5, 14, 32]. ARKit provides a reliable 3D representation/positioning of the face and its features which could be used for tracking the head's movement. The only downside of this approach seems to be the requirement of the hardware and OS in order to use the ARKit framework.

The final solution we reviewed was the use of Convolutional Neural Networks (CNNs). YuNet[36] for example outputs the bounding box along with the positions of the eyes, nose tip, and the corners of the mouth. YuNet is in fact the replacement suggested by OpenCV for detecting faces, which previously used and recommended the Viola-Jones algorithm.

Yan et al. propose a CNN which instead predicts the roll, pitch, and yaw of a face provided within an input image[34]. With their CNN they were able to observe reduced error in their predictions compared to existing tools.

The benefit of a CNN is that they should be invariant of head rotation, given the training data includes samples of heads at different rotations. A potential downside is the need for sufficient processing power. However there do now exist mobile variants of popular Neural Network frameworks, such as TensorFlow, with TFLite, and PyTorch, with PyTorch Mobile, which make running CNNs feasible on mobile devices.

2.3 Phone Localisation

During our review of related works we came across several means of localising a smartphone / tracking a smartphone's movement, each with varying degrees of feasibility.

The least reasonable methods do not bare reviewing due to unrealistic expectations of the population of smartphone users, such as the need for a Motion Capture (MoCap) system[3] or the use of a Head Mounted Display with a mounted tracking marker[23]. These may be suitable in specific environments, but are not reasonable in meeting our goal.

A more reasonable set of methods involve localising the smartphone's position relative to its environment. ***Picture of features used for camera tracking***
One method is 'camera tracking', wherein the movement of the camera is estimated through analysis of an image stream from the rear-facing camera. This is a technique common-place in VFX to recreate the path taken by a camera in 3D[1], but has also been extended to use in Augmented Reality applications[15]. Unfortunately this isn't reasonable to use on current modern mobile phones as they don't all support the ability to capture images from multiple cameras (some via software, others due to hardware limitations). As such we will not be able to utilise the rear camera as the front-camera will be required to track the user's face in our proposed system.

Another solution is the use of either Depth-Cameras or LiDaR and tracking the smartphone's movement through the observed 3D space. Unfortunately these require special hardware that isn't available on most smartphones; most depth cameras that exist on modern smartphones are front-facing and the only current mainstream phones to provide a LiDaR on the rear of the phone are the iPhone 12 and 13 Pro series.

[1]Where a traditional cascading classifier will have possibly have 2 branches at each node, a degrading one will always exit, returning nothing, on one of the branches of each node.
[2]A representation of the input image that permits an efficient means to calculate the sum of a rectangular region of the image with just the for corner points.

The only method we found to be reasonable and feasible was to record the linear and angular acceleration of a smartphone's Inertial Measurement Unit (IMU)[8, 19, 22, 24]. An IMU provides the acceleration experienced in the 6 Degrees of Freedom (DoF)[3] the smartphone can be manoeuvred through. A common issue however with processing IMU output is noise, as noted by Neelasagar and Suresh. To address potential noise they utilised low and high pass filters on the acceleration data.

2.4 Gesture Recognition

Knowing how we can obtain facial features and the 'pose' of the user's head through a front-facing camera, and the localisation of the smartphone itself, the next step is to be able to recognise gestures performed by the user with either their head or the smartphone.

One solution employed by papers proposing systems that tracked Deictic pointing gestures (and possibly Manipulative pointing gestures), was to simply use the raw data, or a function of the data, to map detected facial features to a location within the UI.

A common approach was to take the position of the nose and map it to a point on the screen. This could either be used to manipulate a cursor[9, 26, 30], allowing the user to move their head to highlight specific places on the screen, or to highlight the region of the phone the user is looking[14, 27, 32].

For semaphoric gestures you need to be able to be able to identify the gesture within a sequence of input. An RNN is a Neural Network that takes a sequence of elements and has an internal state that is updated by some function of the current element being processed and the current state. Sharma et al. proposed the use of an RNN in order to recognise head gestures, wherein the RNN input was a sequence of facial landmarks extracted from a sequence of images[28]. An advantage of using an RNN is that you don't strictly need to know exactly when the in the sequence the gesture was performed, just that it is present within the sequence. A downside however is that internal state isn't maintained between predictions, as such you *must* provide a sequence and the input sequence *must* always be of a fixed length[4]. Input must there for be broken-up to fixed lengths, either requiring padding prior to/after the gesture recorded (if you do not have enough elements for the required sequence). To break-up the input you need to either run the model each time-step, providing a rolling window representing the last x frames of state, or to have another means to segment your recorded input to then pass into the RNN.

To predict Semaphoric gestures using a continuous input, without a need for a rolling window, we found a couple of systems which proposed the use of Hidden Markov Models (HMMs)[6, 29]. A HMM describes the possible hidden states a system can be in, the probabilities/rules for transitioning from one state to another, the sates that can be observed, and the probability that a given observation arises from each hidden state. For example, the HMM employed by Elmezain et al.[6] has the gestures as its hidden states, in this case arabic numbers, with the possible observations being a quantised direction[5] that the user's hand travelled, captured via a camera. The probabilities of the sequence of observations would be observed for a given number drawn by the user can then be trained.

3 METHODOLOGY

This section details the process undertaken to develop the system which can meet the goal (outlined in the Introduction): distinguishing between head and phone based gestures on a smartphone.

3.1 Taking A Data Driven Approach

In order to develop the aforementioned system we opted to take a data-driven approach. The benefit of taking a data-driven approach is that we can leverage Machine Learning, and train the system with exemplar data, rather than needing to manually determine the features and derive the algorithm needed to accomplish our goal.

For a data-driven approach to work we need to first determine what data we need to collect in order to train our system. We have identified the following types of data:

Images From The Front Facing Camera
: Given the majority of papers we reviewed ***Citations again?*** utilise a camera to track the user's face, from which they can derive the gestures, we feel it necessary to do the same.

Smartphone Acceleration Data
: To understand whether the smartphone's PoV is changing we need to know how it is moving.

Head Acceleration Data
: If we can determine the movement of the smartphone via acceleration data, it is reasonable to see if we can also do the same with the user's head.

Actual Head and Phone Pose (Ground-Truth)
: In order to accurately train one models we propose below, we will need some Ground-Truth data.

With these data-types we can then build-up a dataset by recording each of the data-types during the performance of a series of gestures. Knowing the gesture associated to the recorded data will allow us to then train our system to recognise the gestures based on the data.

To create the required dataset we decided upon 11 gestures, each with 2-8 variations (effectively directions the gesture could be performed in), resulting in a total of 44 distinct motions to obtain samples of. A table of the gestures and variations can be found under Appendix A.

3.2 Data Collection Study

3.2.1 *Apparatus and Techniques.*

Given the data types listed above, we decided to use the following tools:

Smartphone
: Pixel 4a An Android Smartphone with Bluetooth, a front-facing camera, and an IMU

eSense Earable - A Bluetooth Earbud with an IMU

[3] 3 Linear Axis: X, Y, Z, 3 Angular Axis: Yaw, Pitch, Roll (though this can also be expressed as a Quaternion to avoid gimble lock***ref here, or just drop Quaternion note?***)

[4] ***metion that best to our knowledge this isn't supported***

[5] To reduce the possible observation space, the angles from 0°to 360°are bucketed into a range of 0 to 18

CAMERA Motion Capture Studio
- A Motion Capture (MoCap) studio found on campus within the University of Bath

The smartphone and earbud (when paired via bluetooth) will be able to provide the first three data-types defined above. While the Ground-Truth data can then be supplied by the MoCap studio.

To collect the data we developed an application to run on the smartphone. This was developed in Kotlin[6] and the Android SDK. The application was designed to show participants a motion (a gesture and direction/variation) to perform. This is detailed in text, images, and a video. ***Include figures to show this (spread accross columns at top of page?)*** The participant would then be asked to perform this motion after pressing a record button. While recording the app would do the following:

- Capture images as frequently as possible from the front-facing camera, saving them as raw YUV bytes, with the UTC timestamp as the name.
- Record the smartphone IMU data (linear and angular accelerations), saving them to a csv with the UTC timestamp.
- Record the earbud IMU data (linear and angular accelerations), saving them to a csv with the UTC timestamp.

Once the participant has finished with the motion they could press the same button to stop the recording. Otherwise the recording will automatically terminate after 10 seconds, since the gestures shouldn't take more than a couple seconds to perform and the phone has limited RAM and storage with which to save data. To prevent accidentally stopping the recording too soon, say by accidentally double-tapping the screen, we disable the button for 2 seconds.

Once a motion has been recorded, the app shows the participant the next motion to perform. When the participant completes the final motion to perform, the app returns to the first motion. This repeats two times, such that each motion is captured 3 times. This is to collect variance in each motion for each participant.

In order to collect the Ground-Truth data, the study was performed within the MoCap studio. The participant was asked to wear hat that had a motion-tracker attached, such that the tracker was placed around the middle of the back of their head. An exact position wasn't important as we only needed to determine the relative movement of their head, rather than the exact position. The smartphone was then tracked via a motion tracker attached to a 3D-printed mount, such that the tracker would not affect the participant's grip on the phone, or interfere with the images captured from the front-facing camera.***Include figure to show this*** Each tracker was composed of 5 points. 3 were positioned such that they formed a right-angle triangle, allowing the orientation of the tracker to be derived. The other 2 points were there to improve tracking accuracy, and help make the trackers unique and distinguishable. The MoCap system would track each of the 10 points at 60 fps and export the data as an fbx file.

In order to later synchronise the data collected we required the user to shake the smartphone, with a force of at least 2G, prior to beginning each round of 44 motions. The app would record the shake magnitude (in the X, Y, and Z axis) and the UTC timestamp of when it happened.

The full study protocol can be found under Appendix C

3.2.2 Study Results. Our study was run with 8 participants.

Unfortunately due to an issue with the application, the earbud IMU data was not recorded, despite the earbud being on and paired with the phone. This was not caught until after the study was completed.

Due to a late start, and overrunning into the next participant's slot, participant 0 was unable to complete their 3$^{\text{rd}}$ round of motions.

Some participants didn't initially stop recording upon completing a motion, as such their initial motions have superfluous frames that don't contain data relevant to the motion they're recorded for.

Stats from the data, including figs and tables? Range of motion per gesture. Time taken by gesture and participant, Average sample rate for IMU and Images, Average Number of frames containing face by participant and gesture

3.3 Data Post-Processing

Before being able to use the data for training, we needed to synchronise the data recorded from the smartphone, and the fbx data from the MoCap studio. To do this we derived the acceleration of the phone based on the MoCap data to find where it meets/exceeds the magnitude of the shake recorded by the app. From this we can determine the frame of the fbx data that corresponds to the recorded timestamp. We can determine the frame for any subsequent timestamp based on the known frame-rate of 60fps. To verify the data didn't drift we resync the data based on the other 2 recorded shakes, verifying that they're within 10 frames of the expected frame. In doing this verification, we did not come across any recording wherein subsequent shakes were not found to be at the expected frame.

Synchronisation and post-processing of the fbx data was performed with Blender and Python. Blender was used as permitted viewing the fbx data and verify derived location, roll, pitch, and yaw were correct. Also only way I was able to access the fbx data programmatically. Synchronised data was exported to CSVs for each motion recording, containing a path to the image, the raw IMU data, and the derived MoCap data.

To increase the amount of effective data we have for training we shall do some fps scaling, such that we copy the data, but assuming we're only capturing images every Xfps. We will find the photo closest to the new frame where it would have been captured, and average appropriate data (such as acceleration).

We will also slice the data in overlapping chunks (at least for the RNN model).

3.4 The Proposed Model

To achieve our goal we opted to train 2 models with which we could evaluate and compare performance. The first is a cascading classifier which predicts the motion being performed given a sequence of data. It first identifies if a face is present in an image via a CNN which returns a bounding box of the face[36]. If a bounding box is present the pixels within the box are passed to another CNN

[6]A programming language that runs on the JVM and is used to develop applications for Android.

which extracts the position of 68 landmarks of the face[10]. These landmarks, the bounding box location, the average IMU data since the last image, and the last X frames are then passed into an RNN we trained to classify the gesture. If no bounding box or landmarks are found for a given image, zeros are provided ***or previous data?***. ***is input going to be padded with zeros for first frames / last frames, or require certain number of frames before attempting classification?***

The second model is 2 models which will be trained to predict the direction of movement in each of the 6 DoF for the head and phone (the head model will also take the landmarks and bounding box as input). It will output as a 2d one-hot encoded array, each row being the Degree of Freedom, the column being the direction (0 = stationary, -1 = negative, +1 = positive). The output of the 2 can then be fed into a HMM trained to predict the gesture performed based on the derived motion. (possibly an RNN if easier)

3.4.1 *Training.*

3.4.2 *Model Evaluation.*

4 RESULTS

5 DISCUSSION

6 LIMITATIONS AND FURTHER WORK

Collect more data (to improve accuracy)

Collect the earbud data (if model unsuccessful)

Depending on the model limited by only recognising gestures, not pointing / cursor manipulation

Try transfer learning, had issues loading yunet as made for pytorch

7 CONCLUSION

REFERENCES

[1] Alastair Barber, Darren Cosker, Oliver James, Ted Waine, and Radhika Patel. 2016. Camera tracking in visual effects an industry perspective of structure from motion. In *Proceedings of the 2016 Symposium on Digital Production*. 45–54.

[2] Nusirwan Anwar bin Abdul Rahman, Kit Chong Wei, and John See. 2007. Rgb-h-cbcr skin colour model for human face detection. *Faculty of Information Technology, Multimedia University* 4 (2007).

[3] Wolfgang Büschel, Patrick Reipschläger, Ricardo Langner, and Raimund Dachselt. 2017. Investigating the use of spatial interaction for 3D data visualization on mobile devices. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 62–71.

[4] YH Chan and SAR Abu-Bakar. 2004. Face detection system based on feature-based chrominance colour information. In *Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004.* IEEE, 153–158.

[5] Chatsopon Deepateep and Pongsagon Vichitvejpaisal. 2020. Facial Movement Interface for Mobile Devices Using Depth-sensing Camera. In *2020 12th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 115–120.

[6] Mahmoud Elmezain, Ayoub Al-Hamadi, Jorg Appenrodt, and Bernd Michaelis. 2008. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *2008 19th international conference on pattern recognition*. IEEE, 1–4.

[7] Jérémie Francone and Laurence Nigay. 2011. Using the user's point of view for interaction on mobile devices. In *Proceedings of the 23rd Conference on l'Interaction Homme-Machine*. 1–8.

[8] Enrique Garcia-Ceja, Ramon Brena, and Carlos E Galván-Tejada. 2014. Contextualized hand gesture recognition with smartphones. In *Mexican Conference on Pattern Recognition*. Springer, 122–131.

[9] Dmitry O Gorodnichy and Gerhard Roth. 2004. Nouse 'use your nose as a mouse' perceptual vision technology for hands-free games and interfaces. *Image and Vision Computing* 22, 12 (2004), 931–942.

[10] Yin Guobing. 2021. Head Pose Estimation By TensorFlow and OpenCV. https://github.com/yinguobing/head-pose-estimation

[11] Thomas Riisgaard Hansen, Eva Eriksson, and Andreas Lykke-Olesen. 2006. Use your head: exploring face tracking for mobile interaction. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*. 845–850.

[12] Steven Hoober. 2013. How do users really hold mobile devices. *Uxmatters (http://www. uxmatter. com). Published: Feburary* 18 (2013), 2327–4662.

[13] Kohsia S Huang and Mohan M Trivedi. 2004. Robust real-time detection, tracking, and pose estimation of faces in video streams. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 3. IEEE, 965–968.

[14] Sebastian Hueber, Christian Cherek, Philipp Wacker, Jan Borchers, and Simon Voelker. 2020. Headbang: Using head gestures to trigger discrete actions on mobile devices. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–10.

[15] Bolan Jiang, Suya You, and Ulrich Neumann. 2000. Camera tracking for augmented reality media. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532)*, Vol. 3. IEEE, 1637–1640.

[16] Maria Karam et al. 2005. A taxonomy of gestures in human computer interactions. (2005).

[17] Jin Kim, Gyun Hyuk Lee, Jason J Jung, and Kwang Nam Choi. 2017. Real-time head pose estimation framework for mobile devices. *Mobile Networks and Applications* 22, 4 (2017), 634–641.

[18] 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*. 577–585.

[19] Sven Kratz, Michael Rohs, and Georg Essl. 2013. Combining acceleration and gyroscope data for motion gesture recognition using classifiers with dimensionality constraints. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. 173–178.

[20] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2018. Fingers' Range and Comfortable Area for One-Handed Smartphone Interaction Beyond the Touchscreen. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[21] Miguel Bordallo López, Jari Hannuksela, Olli Silvén, and Lixin Fan. 2012. Head-tracking virtual 3-D display for mobile devices. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 27–34.

[22] V-M Mantyla, J Mantyjarvi, T Seppanen, and Esa Tuulari. 2000. Hand gesture recognition of a mobile device user. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532)*, Vol. 1. IEEE, 281–284.

[23] Peter Mohr, Markus Tatzgern, Tobias Langlotz, Andreas Lang, Dieter Schmalstieg, and Denis Kalkofen. 2019. Trackcap: Enabling smartphones for 3d interaction on mobile head-mounted displays. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–11.

[24] K Neelasagar and K Suresh. 2015. Real time 3D-handwritten character and gesture recognition for smartphone. *International Journal of Computer Applications* 123, 13 (2015).

[25] Euclides N Arcoverde Neto, Rafael M Barreto, Rafael M Duarte, Joao Paulo Magalhaes, Carlos ACM Bastos, Tsang Ing Ren, and George DC Cavalcanti. 2012. Real-time head pose estimation for mobile devices. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 467–474.

[26] Yoshikazu Onuki and Itsuo Kumazawa. 2016. Combined use of rear touch gestures and facial feature detection to achieve single-handed navigation of mobile devices. *IEEE Transactions on Human-Machine Systems* 46, 5 (2016), 684–693.

[27] Maria Francesca Roig-Maimó, Javier Varona Gómez, and Cristina Manresa-Yee. 2015. Face Me! Head-tracker interface evaluation on mobile devices. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 1573–1578.

[28] Mohit Sharma, Dragan Ahmetovic, László A Jeni, and Kris M Kitani. 2018. Recognizing visual signatures of spontaneous head gestures. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 400–408.

[29] Juan R Terven, Joaquin Salas, and Bogdan Raducanu. 2014. Robust head gestures recognition for assistive technology. In *Mexican Conference on Pattern Recognition*. Springer, 152–161.

[30] Javier Varona, Cristina Manresa-Yee, and Francisco J Perales. 2008. Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications* 31, 4 (2008), 357–374.

[31] Paul Viola and Michael J Jones. 2004. Robust real-time face detection. *International journal of computer vision* 57, 2 (2004), 137–154.

[32] Simon Voelker, Sebastian Hueber, Christian Corsten, and Christian Remy. 2020. HeadReach: Using Head Tracking to Increase Reachability on Mobile Touch Devices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[33] Pei Xuesheng and Wang Yang. 2018. Research on the Development Law of Smart Phone Screen based on User Experience. In *MATEC Web of Conferences*, Vol. 176. EDP Sciences, 04006.

[34] Peng Yan, Binbin Wu, Xiaoqian Nie, Bingqi Wang, Xianliang Wang, and Ziwei Liu. 2021. A Fast and Efficient Face Pose Estimation Algorithm for Mobile Device.

In *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. IEEE, 970–973.

[35] Yukang Yan, Chun Yu, Xin Yi, and Yuanchun Shi. 2018. Headgesture: hands-free input approach leveraging head movements for hmd devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–23.

[36] Shiqi Yu and Yuantao Feng. 2022. YuNet. https://github.com/ShiqiYu/libfacedetection

A TABLE OF GESTURES AND VARIATIONS

B DATA COLLECTION APP CLASS DIAGRAM

C STUDY PROTOCOL DIAGRAM

D STUDY DATA ANALYSIS