FoldAR: Gesture Analytics Using Apple Vision Framework

Ian Brown
Colorado State University,
Department of Electrical and
Computer Engineering
Fort Collins, Colorado, United States
ian.brown@colostate.edu

Tani Cath
Colorado State University,
Department of Computer Science
Fort Collins, Colorado, United States
tani.cath@colostate.edu

Tom Cavey
Colorado State University,
Department of Computer Science
Fort Collins, Colorado, United States
tomcavey@colostate.edu

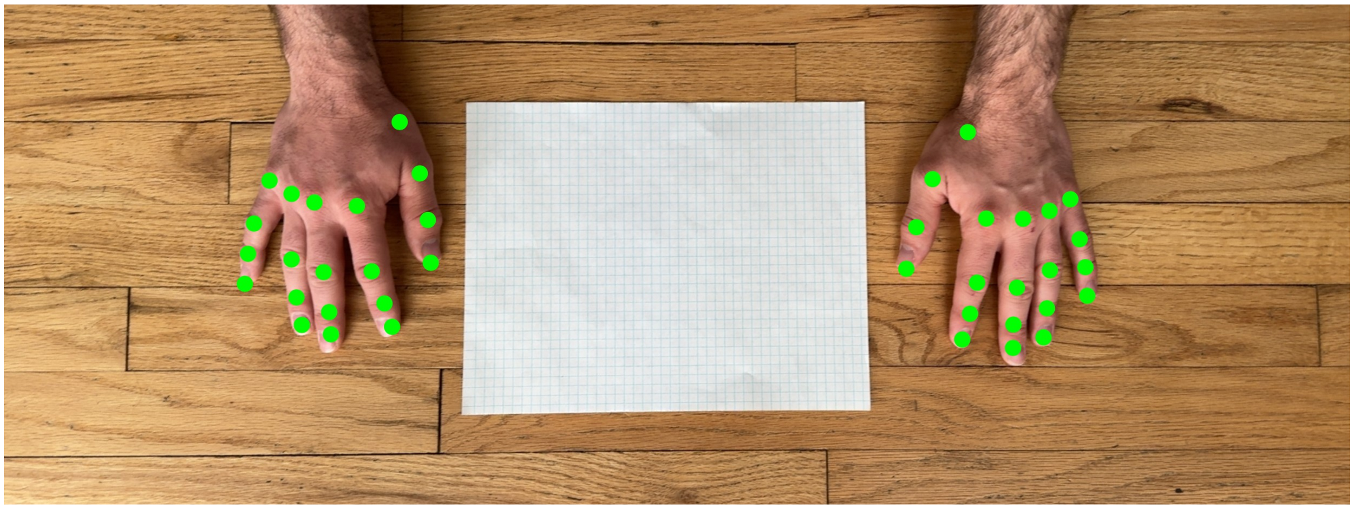


Figure 1: Hand Pose estimation application

ABSTRACT

In this experimental study, we investigate hand pose estimation and gesture analytics with paper folding exercises. Our methodology involves the utilization of a custom iOS application for human hand detection to collect hand point data during paper folding. During experimental trials, participants were instructed to fold paper under three distinct sets of constraints: with eyes open, eyes not focused on their hands, and eyes closed. Using the data collected from the experimental trials, several trends and patterns were observed that allowed the detection of distinct stages of the folding process and uniquely identified individual participants. Our application developed during the research is open-source and available for general use. We hope that our contribution will be helpful to future researchers to collect their own hand-point data.

CCS CONCEPTS

• Computing methodologies → Tracking; **• Human-centered computing → Gestural input**; **Mixed / augmented reality**; *User interface design.*

KEYWORDS

Augmented Reality, Computer Vision, Hand Tracking, Gesture Detection, Hand Gestures, Hand Pose Estimation, VisionOS, Paper Folding

ACM Reference Format:
Ian Brown, Tani Cath, and Tom Cavey. 2023. FoldAR: Gesture Analytics Using Apple Vision Framework. In *CSU, Fall '23: CS 567 – Introduction to 3D User Interfaces, August 21–December 10, 2023, Fort Collins, CO.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Gesture recognition and hand pose estimation are important components of human-computer interaction, playing a crucial role in the user experience in various applications. Interactive computing using natural hand gestures involves sophisticated tracking devices using sensors, cameras, and software infrastructures built specifically for that purpose [19].

Gesture tracking is achieved with high accuracy with the use of machine learning and high resolution hardware [30]. The objective of our research is to leverage the iOS platform and Apple's Vision

Framework, to achieve real-time tracking and analysis of hand movements during the process of folding paper.

The motivation behind this study comes from the growing world of human computer interaction and the need to continuously improve the accuracy and efficiency of hand pose estimation systems. As embedded sensors and cameras become capable of capturing increasingly complex details, requirements for an application that can accommodate dynamic hand movements grows [6]. Current platforms often require the use of multiple sensors and input devices to capture the details of intricate tasks like folding paper, where smaller movements and subtle microgestures are crucial for understanding the user's actions. Our research aims to address the challenge of collecting comparable data using only a mobile device, and to contribute to the advancement of gesture recognition analysis.

Through several experiments, we collected data procured during the act of folding paper. The choice to use paper folding as the foundation of the study is due to the inherent complexity and dexterity required to perform a combination of precise hand movements. By analyzing the hand-point data collected during the experiments, we sought to develop a method for identifying distinct folding stages and discovered a potential trend of biometric identifiers left behind by the participants. Our methodology includes the utilization of Apple's Vision Framework which provides pre-trained machine learning models for detecting hands [8], and is capable of identifying 21 individual points per hand, see Figure 2.

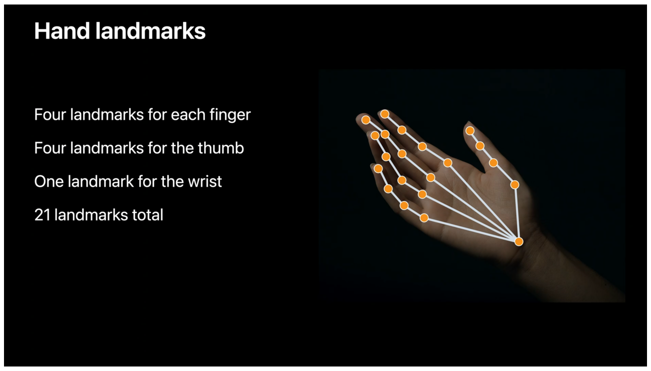

Hand landmarks
Four landmarks for each finger
Four landmarks for the thumb
One landmark for the wrist
21 landmarks total

Figure 2: Apple Vision Framework hand landmarks. Reproduced from [8]

1.1 Paper Outline

This report consists of a total of eight sections and two appendices. In Section 2, we cover many of the sources we relied on to better our understanding of the fundamentals of hand recognition and to develop our application; in Section 3 we briefly list how others can find our source code; in section 4 we cover the details of our application's design and development; in Section 5 we go into detail how we designed and executed the experimental portion of our research; in Section 6 we cover the quantitative and qualitative results of our research extracted from our collected data; in Section 7 we cover what we learned, what we could have improved, and potential avenues for future research; in Section 8 we summarize our findings and what we gained from our research. Appendix A details the process of downloading, building and using our application, FoldAR, and Appendix B contains additional tables and figures that were too large to include within the main body of the paper.

2 RELATED WORKS

Our experiment drew from a wide range of existing research, encompassing diverse methodologies and approaches. While numerous studies informed our work, specific methodologies and research questions differ notably from our own. This section aims to introduce methodologies, research, and documentation we found relatable to our experiment. Works such as hand tracking performed with the use of a sophisticated hand tracking glove [20], or dedicated sensors designed to capture hand gestures which yield high precision and accuracy. These techniques are limited in flexibility compared to using a mobile device camera system. Such as in [29], the challenges of tracking the intricate structure of the human hand are shown to be difficult due to its complexity. Hand motion is highly articulate and can be difficult for Computer Vision based models to correctly detect, even as contemporary and modern mobile phone cameras and processors are generations ahead of past available computing power and camera resolutions. Hand motion is also limited to the range of motion in human joints which reduce the amount that each finger can articulate. Hand pose estimation can also be performed by segmenting individual fingers. In [22] a method is presented for determining gestures based on the number of fingers present in the frame. This optimizes the functionality of gesture recognition but can be beneficial if only limited finger tip or finger count information is required.

In an effort to gain an understanding of how Apple may have designed their human hand pose estimation model, [1] offered an demonstration of how a Computer Vision model using a convolutional neural network detected a hand. The research also presented a few different methodologies to improve the performance and robustness of hand detection models.

A recent approach using a self-attention model for hand detection achieves consistent and more accurate detection while also managing to create a lightweight model optimized for low powered, embedded, or mobile devices [24]. The dataset used to train the model included images in-the-wild, occluded hand images, and images of hands holding objects. Other examples of 2D models trained on hand pose datasets include [31], which boasts the use of an 18,000 stereo image dataset to train the model.

Our study needed works that specifically mapped hands with their gestures, so the recognition system from SHREC'17 was just one of many helpful sources [25]. The authors focused on the development of a new three-dimensional dynamic hand gesture dataset [25]. Their models of hands, and especially the similarities in their skeleton detection, is similar to Apple's.

A study from Chonnam National University [27] was relevant to our study because it discusses the potential of hand gesture recognition for human-computer interaction (HCI). The authors of the article propose a novel method for fingertip detection and hand gesture recognition in real-time using an RGB-D camera and a 3D convolutional neural network (3DCNN) [27]. Their system achieved high accuracy and robustness in recognizing a variety of gestures, suggesting that hand gesture recognition is a promising approach for HCI.

In [4], the authors used the *COCO Whole-body dataset* [16] to train their hand detection system. Because Apple does not disclose how their hand tracking framework was trained, we can only assume that they used similar methodologies and datasets as [4] and [25] to train their hand pose estimation model.

In [2] a smartphone is used to perform remote monitoring of hand trauma via an application. The Origami Guru project was one of a few similar studies that involved paper folding [28].

Deploying a hand detection model onto a mobile device means that data can be captured about hand poses in scenes that the model may not have been trained to do. This requires a robust model that has been trained to track hand movements in various environments and backgrounds [26]. A state of the art detection model introduced in [23] shows how a real time tracking can be greatly improved by combining two algorithms: Particle Swarm Optimization [17] and Iterative Closest Point [21].

3 CONTRIBUTIONS

We present an application implementation for tracking hand-points using Apple's Vision Hand Pose framework. Our application was developed using Xcode and the source code can be viewed on our public GitHub repository under *FA23-FoldAR-Step-by-step-Instructions-for-Folding-Paper-Models-in-AR*.

4 APPLICATION DESIGN AND DEVELOPMENT

The design and development of our iOS application plays a critical role to our research, serving as a specific tool for capturing hand point data logs during experimental trials. There are other pre-existing hand point gesture datasets available for use, however our analysis specifically uses the Apple Vision framework's `HandPoseEstimationModel` library. This application can be used to capture and build datasets from trial experiments, we have developed it study paper folding gestures but it can be used to record hand point data on general tasks.

Apple's 2023 A17 Pro processor contains specifically designed hardware and firmware platform called the *Apple Neural Engine* which is capable of 35 trillion operations per second [10], making it as powerful as some desktop GPU's. Applications that use computer vision models benefit tremendously from these technical advancements, and Apple provides simple API's for developers to harness the full power of the system. The application reliably captures hand pose estimation data defined in the `HandPoseEstimationRequest`, and performs inference in real time on up to 30.04 frames per second, see Section 6.5 for details.

At a high level view the application captures live video and overlays a marker on key hand points. This involves processing video frames and dynamically adding graphics to the screen in real time. In addition to displaying the augmented scene, the hand points are recorded in a csv format in the application's memory sandbox.

As stated in the Introduction, our application is built on the base code provided by Apple as part of their 2020 World Wide Developers Conference, "Detect Body and Hand Pose with Vision" [8]. For information on how to build, run and our application, FoldAR, available at the GitHub repository listed in Section 3, see Appendix A.

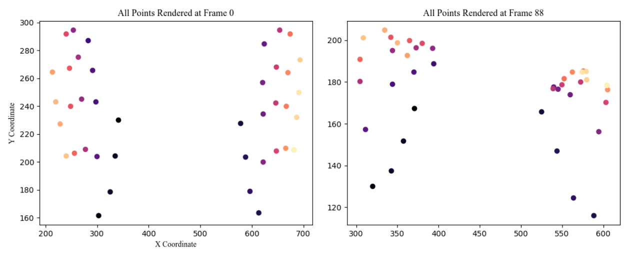


Figure 3: Points recorded during frames 0 and 88 of an experimental session

4.1 Application Architecture

The foundation of the application is built on the `UIViewController` class which manages the rendering of content on the screen [13]. This class processes a live video feed sourced from the device's back-facing camera. The `AVCaptureSession` API [12] enables the real-time capture of video frames from the camera feed. The `ViewController` and the `AVCaptureSession` together render an all the frames that visible in the application live camera view. The hand points are accessed by multiple components, including a video frame overlay which plots the location and fills in graphical information at each point. All of the hand-points are rendered in real time on the iOS device's display by the `CameraView` class. The `UIBezierPath` object is created to render the 2 dimensional points, and the `CAShapeLayer` object defines the radius of points and color for the entire overlay.

The data logger component `FileManager` API provides a method for creating files that can be stored within the application's memory sandbox. When data recording has begun the `FileManager` will create a file and being streaming the hand pose estimation data to the file. When no hand points are present in the frame during recording, the `FileManager` will not write data in order to preserve memory. The User Interface elements contain a text field and buttons for users to input the participant ID, select experimental mode, and a start/stop button to control the recording of data. UI items are rendered independently of the `AVCaptureSession`, and are not affected by the processing of video frames.

4.2 HandPoseEstimation Model

To predict the hand points for a frame, a `VNDetectHumanHandPoseRequest` is created and sent to the `VNImageRequestHandler` [14]. The handler performs inference on the frame from the input buffer. If a hand is detected, the hand pose request results are returned to the caller with `RecognizedPoints`. The data contains a prediction score, the X- and Y- coordinates of detected finger joints, and chirality among other information for hand pose estimation. See Figure 3 for a snapshot of this data in a static frame and Figure 4 for a visualization of a single point's path over an entire experiment. All frames with a prediction score of less than 30% are discarded and not rendered.

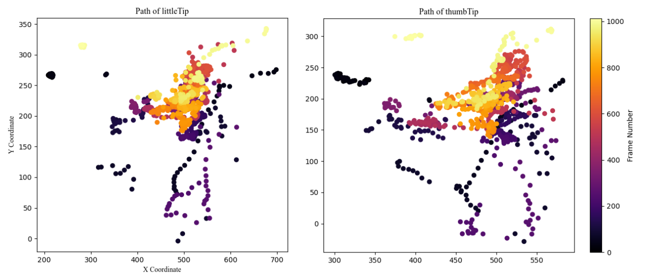
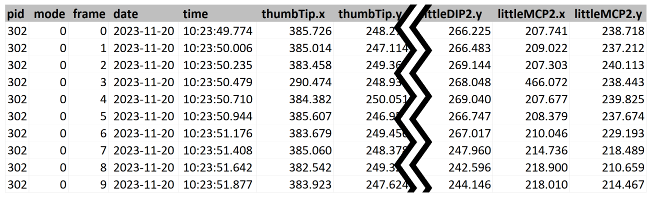


Figure 4: Temporal data for little finger and thumb tip trajectories



pid	mode	frame	date	time	thumbTip.x	thumbTip.y	ttleDIP2.y	littleMCP2.x	littleMCP2.y
302	0	0	2023-11-20	10:23:49.774	385.726	248.2	266.225	207.741	238.718
302	0	1	2023-11-20	10:23:50.006	385.014	247.114	266.483	209.022	237.212
302	0	2	2023-11-20	10:23:50.235	383.458	249.36	269.144	207.303	240.113
302	0	3	2023-11-20	10:23:50.479	290.474	248.93	268.048	466.072	238.443
302	0	4	2023-11-20	10:23:50.710	384.382	250.051	269.040	207.677	239.825
302	0	5	2023-11-20	10:23:50.944	385.607	246.9	266.747	208.379	237.674
302	0	6	2023-11-20	10:23:51.176	383.679	249.45	267.017	210.046	229.193
302	0	7	2023-11-20	10:23:51.408	385.060	248.37	247.960	214.736	218.489
302	0	8	2023-11-20	10:23:51.642	382.542	249.3	242.596	218.900	210.659
302	0	9	2023-11-20	10:23:51.877	383.923	247.624	244.146	218.010	214.467

Figure 5: Sample CSV data file

The `VNDetectHumanHandPoseRequest` returns a comprehensive list of points at the joint of each finger and up to 40 specific finger joints as well as the two wrist points can be retrieved, though only fingers were used in our research. The set of 2D points that represent coordinates for the tip and joints on all five fingers [15]. The points available on the thumb are *tip*, *interphalangeal*, *metacarpophalangeal*, *carpometacarpal*. For the index, middle, ring and little fingers the points are *tip*, *distal interphalangeal*, *proximal interphalangeal*, *metacarpophalangeal*.

The detected points are used by the `HandGestureProcessor` class which controls the processing of hand-point data, generates CSV log files, and logs the point data as entries into the files.

For a single frame, the hand detection API request can be performed on up to two hands. Apple's documentation states that only the largest hands in the camera's view will have their points detected and rendered [14]. All data parameters are recorded locally, as discussed in Section 5.3.

5 EXPERIMENTAL METHODOLOGY

The collected hand-point data is used to gain an understanding of the user's hand movement and usage while folding paper. To this end, participants were asked to perform folding actions as described in Section 5.1.

The research questions we hoped to answer with these experiments were:

(1) Whether or not individual folds can be detected using sequential frames of hand-point data, and
(2) If individuals can be uniquely identified based solely on hand-point and temporal data.

5.1 Experimental Design

In order to collect useful data, a simple experiment was performed with fourteen participants. Steps 1-5, below, where repeated a total of three times with the following constraints placed on the participants, referred to as the "experimental mode":

- 1st repetition: No constraints, "Mode 0 – Full Vision (FV)".
- 2nd repetition: The user was instructed to look away from their hands, "Mode 1 – Partial Vision (PV)".
- 3rd repetition: The participant was instructed to close their eyes, "Mode 2 – No Vision (NV)".

The sequence of steps performed during each experimental trial was as follows:

(1) The participant was asked to sit or stand in front of a flat work surface (table or counter) with a single sheet of 8.5x11 inch paper in front of them and their hands laying palms-down on either side of the paper.
(2) The iOS device running our application was positioned facing the participant, approximately 36 inches away from them and 24 inches above the work surface and angled downwards so as to center the participants' hands and arms in the camera frame.
(3) The participant was asked to fold the sheet of paper in half three times using both hands and utilizing whatever technique they chose.
(4) As the participant performed the folding action, hand-point data was collected, see Sections 5.2 and 5.3 for details.
(5) Once the third fold was completed, the participant placed the paper on the work surface in front of them and laid their hands, palms down, on the work surface on either side of the paper.

5.2 Dataset Collection and Details

As discussed in Section 4.2, Apple's Vision Framework hand pose detection library is capable of detecting up to 42 total points (21 points per hand). For our research we opted to ignore the wrist-point for a total of 40 points from both of the user's hands.

This data was recorded during the experimental process into a comma-separated values (CSV) file with each row containing a total of 85 columns, 80 for the finger point data (X- and Y-coordinates for each of our 40 detected hand-points), along with the participant's anonymized three-digit id (pid), the "mode" of the current experiment, represented as 0, 1, or 2, the frame count starting from the beginning of the session, and timestamp data including both date and time with millisecond precision, see Figure 5 for a sample data file.

This data was recorded locally in the application's "sandboxed" documents directory on the iOS device, with an individual CSV file generated for a given combination of participant id and experimental mode, with a new row of data written to the file once per frame (see Section 6.5 for further discussion on application performance related to data logging). Once experiments were concluded, the CSV session data files were offloaded from the iOS device for further data processing and analysis, see Section 6 for details of our results.

5.3 Data Analysis and Visualization

As mentioned in Section 5.2, the hand-point data recorded by the application on the iOS device was offloaded for further analysis. We utilized Python to evaluate various metrics – though due to the open format (CSV) of the data files, any analysis platform could have been used – primarily leveraging the capabilities of the widely-used Pandas [18] and Numpy [5] packages, versions 2.1.3 and 1.25.2, respectively. Visualization of our results was performed using MatPlotLib version 3.8.0 [7] for the generation and storage of bar, scatter, and box plots representing our raw and derived data.

6 RESULTS

In the findings of our results we make positive observations that support our initial hypothesis, but also create new opportunities to expand on finding a deeper meaning in the human part of the data.

In this section we provide quantitative data on how point distance calculations and other point tracking algorithms can be used to identify how many folds occur, when they occur, and the possibility of recognizing individuals by the way they fold. First to support our initial hypothesis, we find that there are a few indicators that can be used to detect distinct folding actions. Secondly, we inadvertently discovered that there may be trends in the recorded data points that can be used to distinguish subjects from one another. We find that each subject's finger trajectories follow distinct patterns that could be used as biometrically identifiable data to that subject.

6.1 Mean Euclidean Distances

The Euclidean distance formula is used to calculate the distance between two points on a 2-dimensional plane, see Eq. 1, below. This calculation is critical to our results in finding trends in folding behavior.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (1)$$

There are 3 different Euclidean distance calculations we used. The first distance calculation performed is the mean of pairwise distance between finger points for a given frame. This provides a general measurement of how close the left and right hands are. For example measuring distance between index fingers between both hands. The second distance calculation performed is the mean distance between finger tips on each individual hand. This effectively provides a general measurement for how close the fingertips are to each other. The third distance calculation performed is the mean distance between all points on each individual hand. This measurement gives a general idea of how open or closed the hand might be. A general trend in the dataset shows the mean distance between hands and fingertip points incrementally decreasing over time.

The distance between finger tips may be used to determine unique characteristics of folds, such as which stage of the fold a user has performed. In combination with fingertip points and timestamp data, it is also possible to calculate folding speed, time to finish, and finger velocity.

6.2 Fold Stage Detection

Based on the calculations described in Section 6.1, it may be possible to determine which stage of the fold a user is currently on

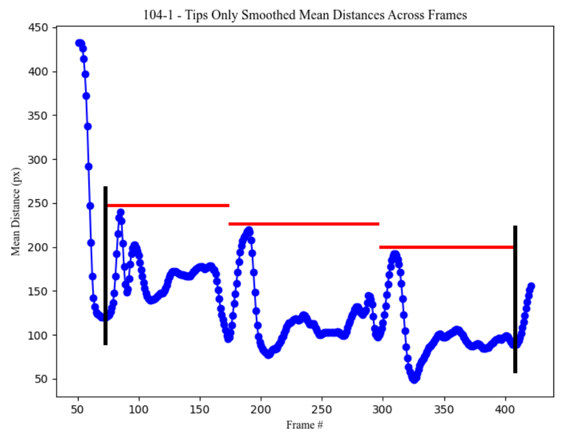


Figure 6: Mean distances of fingertips over all frames in a session. Black vertical bars represent assumed start and end points of folding sequence, red horizontal bars represent assumed distinct fold stages

based on the clustering of points. By tracking the movement of fingertips such as the thumb, index, and little finger over the entire trial the folds, the position of finger tips over the course of a folding trial reveals several distinct positions or areas within the frame. The clustering of this data shows that individual folds can be determined.

The individual fold stages within a session can be shown by calculating the Euclidean distances between selective finger or joint points. First, in Figure 6, there are distinct change points roughly at frame 80, 180, and 310 where relatively quick changes in finger tip locations are observable. An indication that the points temporal data may be the useful in identifying folds or other data. We find this trend to be accurate across all users and modes. Some examples are more difficult to interpret due to possibly noisy data or lower data capture speed. In the analysis of the mean distances data, shown in Table 1, we found that across all users as frame counts increase, the the mean distances become smaller. This gives some indication of the scale of paper folding becoming smaller with each half fold.

6.3 User Identification

Looking at the traced paths of a single fingertip over the course of a folding session, as well as over the three experimental modes, and then comparing these plots between two individual participants, one can clearly see drastic differences such as in Figures 10 and 11 in Appendix B.

As these two figures clearly show, individual participants, when provided with a sheet of paper and the open-ended instruction to "fold the paper in half three times", each perform that task in vastly differing ways. This observation leads to the intuition that using only hand-point data during a given task, individuals could not only be distinguished from each other, but potentially could also be directly identified given a large enough dataset of how they user their hands in a variety of tasks, as discussed in Section 7.1.

Table 1: Sum of mean Euclidean distances between all finger-tip points over five quantiles

		Mean Distance				
User Id	Trial #	1	2	3	4	5
102	0	83.73	50.42	37.31	32.96	43.98
102	1	133.81	97.82	59.74	37.73	60.69
102	2	65.27	45.99	42.00	37.74	38.07
103	0	107.68	54.29	55.44	52.82	71.01
103	1	83.87	62.72	52.35	54.66	81.61
103	2	86.11	50.09	60.19	51.85	65.25
104	0	98.34	61.93	54.62	55.48	69.07
104	1	104.65	62.83	50.40	52.47	70.21
104	2	83.45	56.48	60.96	58.04	63.32
		...				
302	0	50.10	36.65	41.64	39.21	46.50
302	1	47.70	28.79	26.92	26.72	28.79
302	2	60.74	31.35	28.24	22.51	38.78

The temporal data relationship between frames can help to further identify unique behaviors, from the finger trajectory path data. The trajectory is plotted using color coding that changes as the individual frame count increments. Certain participants perform patterns throughout each trial, as we can see in 4 the quick succession of vertical points in black show how a rapid and sweeping movement pattern occurs some time in the beginning of the fold trial. The data can be further analyzed to understand the time captures and the distances the finger tip points moved within a given amount time to determine spikes in finger acceleration.

When considering specific fingers in most cases we find that the index, middle, and ring finger tips generally follow the same dynamic movement paths see appendix, and the thumb tip and pinky tip have trajectories that can be particularly specific. For example, the indexTip path for user 102 during the mode 0 and mode 1 experiments have a significant amount of overlap versus mode 3. However, when comparing user 102 across all modes to user 302, they appear to be completely different patterns. 102 can be described as having an angular, stretched, or narrow arrangement where 302 has a sparse or rounded trajectory.

The uniqueness that is observed in the individual finger paths also appears to remain consistent across trial modes. For example user 205 has distinguishable vertical lines with emphasis on the left hand side of the plots. This can be seen across all the trial modes and this consistency reinforces the idea that the patterns we observe are likely an indication of habitual or muscle memory characteristics when it comes to folding paper. However, this does not hold true for all subjects that were tested. For example User 107 on mode 2 (the eyes closed method) was left justified heavily, with only a few single points on the right hand side. There are much fewer points that overlap when compared to the prior mode 0 and 1 for that user.

Finally, we calculate an overlay of points in an attempt to understand how similar a user's folding patterns are to themselves. If the distance between any two points is below a threshold, we consider it an overlaying point. For our example we used the euclidean distance between two points with a threshold of 1.0. We find that generally most users are more similar than to others, with some exceptions. This can be expanded on to attempt to use the data to try identifying and masking noisy data or removing actions that do not contribute to uniqueness.

Table 2: Participant completions times by experimental mode

Compl. Time (s)	Full Vision	Hands Obscured	No Vision
Mean	22.395	22.788	29.190
Minimum	3.559	6.675	8.804
Maximum	72.903	57.437	63.200
Median	19.608	18.015	24.488
First Quartile	12.849	12.398	17.871
Third Quartile	24.094	31.388	41.288
IQR	11.244	18.990	23.417
Std. Deviation	16.276	13.915	15.738

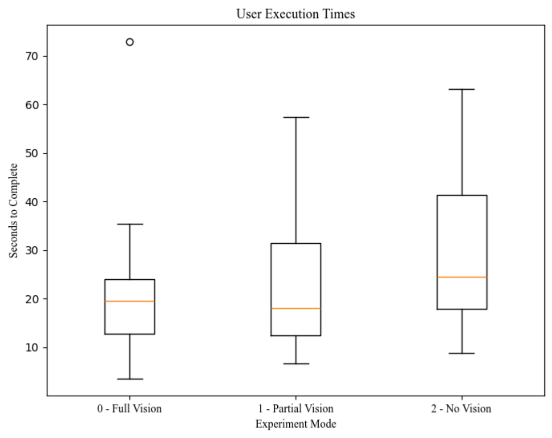


Figure 7: Participant completion time comparison by experimental mode

6.4 User Performance

One key metric that was analyzed from the experimental data was the time it took each participant to complete a fold sequence (see Section 5.1). As can be seen in Table 2 and Figure 7, while the average completion times were very similar, especially for the first and second experimental modes (full vision and hands obscured) with the third (no vision) averaging only slightly slower, with experimental mode 0 (full vision) having the tightest interquartile range.

While the statistical data shows a general pattern that progressively decreasing a user's vision capabilities directly correlates to decreased performance (longer completion times), the limited sample size of only fourteen participants most likely skews the data as one participant (108) showed almost no difference in completion time between partial vision and no vision, while three participants (106, 205, and, most noticeably, 300), roughly 21%, showed greater performance (faster completion times) no vision at all compared to limited vision, often attempting to close their eyes during this experimental mode in order to improve their efficiency, as can be

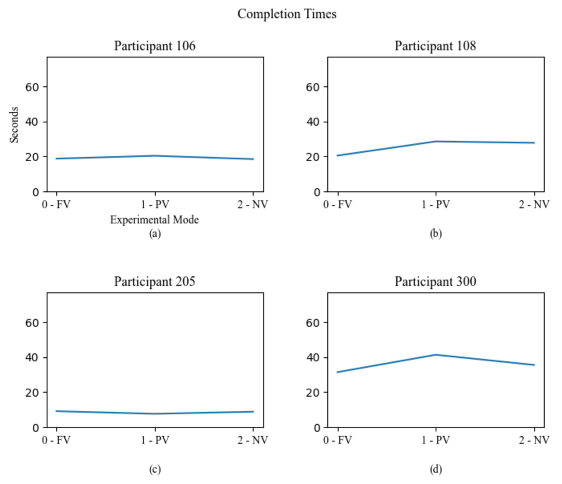


Figure 8: Participant completion times slower with partial vision than with no vision

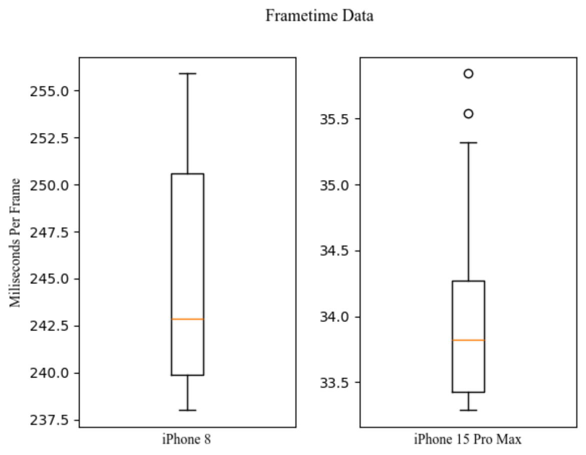


Figure 9: Frametime data comparison between iPhone 8 and iPhone 15 Pro Max

Table 3: Frametime data by iOS device, averaged over all datasets

Frametime (ms)	iPhone 8	iPhone 15 Pro Max
Mean	245.652	34.039
Minimum	237.992	33.285
Maximum	255.900	35.843
Median	242.845	33.818
First Quartile	239.872	33.426
Third Quartile	250.605	34.270
IQR	10.733	0.844
Standard Deviation	6.704	0.715
Average Framerate (FPS)	4.071	29.378

see in Figure 8. In addition, seven participants (50%) performed *slower* with full freedom of vision than with partial vision, and in some cases even slower than with no vision at all.

6.5 Application Performance

As was mentioned in Sections 5.1 and 4, experiments were conducted using two different Apple iPhone models, the iPhone 8 (2017), and the iPhone 15 Pro Max (2023). One significant performance discrepancy between the models used that was immediately noticeable in the data was the amount of time required to process a single frame of information. As can be seen in Table 3, while the iPhone 15 is able to maintain a very consistent average frametime of 34 milliseconds (just under 30 frames per second) with a standard deviation of only 0.715, the iPhone 8 struggles to maintain an average frametime of 245.6 ms (a slideshow-level four frames per second), with a significantly greater standard deviation of 6.7.

Figure 9 visualizes this discrepancy, and shows just how much processing power is required to run Apple's Vision Framework and how, despite the iPhone 8 being *technically* capable of running an augmented reality application built on the Framework, the user experience is drastically degraded due to the extremely low framerates (high frametimes) which have been found to have a significant negative impact on user quality of experience (QoE), especially at values lower than 15 FPS [32].

While the performance difference between the iPhone models is unsurprising considering that Apple officially dropped the iPhone 8's support for the most recent Apple mobile operating system (iOS 17) [9, 11], the data collected on the older hardware is still usable in our analysis, though roughly 7.2 times more sparse. Similar to the limitations imposed due to a small sample size of participants discussed in Section 6.4, these performance differences due to hardware variations are due to the scope of the study as a full-scale research project would have also allocated funds to provide all researchers with identical hardware.

7 DISCUSSION

While the findings discussed in Section 6 are promising, there remain many open avenues of research as well as, even with the limited quantity of data collected, several other metrics that could have been analyzed had additional data been collected at the time that the experiments were conducted.

For example, in addition to the complete user identification discussed in 6.3, had participants been asked to provide some personal information, such as handedness, the recorded hand-point data could have been used to potentially detect which hand is dominant for a given user, as looking at the path data of the same fingertip for two users, such as in Figures 10 and 11, one can observe a certain degree of mirroring in the two patterns, which seem to indicate in that these individuals had opposite handedness though without having collected that data we are unable to verify this hypothesis.

The current data can also provide insights into the finger tip velocity or acceleration. Using the timestamp data, it is possible to calculate the relative velocity of the finger tip points and determine spikes in finger acceleration. This could add a beneficial perspective to the data, and leverages the temporal dynamics of the experiment.

If the folding experiments were designed in a way to remove noise and a significantly larger quantity of labeled data was captured, the data could be used to train a machine learning algorithm, such as *Random Forest*, to perform pattern recognition and identify individuals based on the way they use their hands. This is similar to keystroke recognition, which is used as a method of user identification and authentication [3].

7.1 Future Work

Due to the time and data limitations of this research, our team would greatly benefit from having more time to collect data from a larger pool of participants. In addition, performing the experiments in a more controlled environment would allow for greater consistency both in how the data is collected. Improved lighting conditions and fixed angles and distances between the participants and the recording equipment, as well as access to uniform hardware for data collection would result in both higher quality and larger quantities of data.

As discussed in Section 6.3, one potential path of future research into identifying individuals based solely on hand-point data could be to prompt participants to perform series of different tasks that involve manipulating objects with their hands. In this context the folding can be used as a biometric identification technique.

8 CONCLUSION

We expected that the folding sequence could be identified using hand-point data. While we were able to detect changes in fingertip distance to count folds, we found the data revealed several other trends about hand gestures that appeared more relevant to current areas of study in the field of human-computer interaction.

During our experiments, we found that using hand pose detection on an iOS device serves as a more than adequate research tool for collecting hand-point data. We anticipated that the insights gained from our experiments would provide answers to our research questions.

Our analysis of the distances between hand points reveals a pattern consistent with the folding sequence. This pattern can be used to infer a correlation between the hand-point distances and fold count. Given further research into this area, programmatically detecting individual folds could possible.

The results of the experiments indicate it may be possible to detect distinct biometric markers and uniquely identify individuals based on the ways they use their hands while performing a given action. Given more data, individual trends may become more apparent, and we encourage further research into this topic.

We are confident that our initial research questions were answered, and that additional research questions could be asked and answered with more time and data.

ACKNOWLEDGMENTS

We would like to thank Dr. Francisco Ortega, Richi Rodriguez, and all the individuals who participated in the experimental portion of our study.

REFERENCES

[1] Mahmoud Afifi. 2018. 11K Hands: Gender recognition and biometric identification using a large dataset of hand images. arXiv:1711.04322 [cs.CV]

[2] Tania Banerjee. 2023. *Computer Vision-Based Hand Tracking and 3D Reconstruction as a Human-Computer Input Modality with Clinical Application.* Master's thesis. University of Western Ontario. https://ir.lib.uwo.ca/etd/9173

[3] Nick Bartlow. 2009. *Keystroke Recognition.* Springer US, Boston, MA, 877–882. https://doi.org/10.1007/978-0-387-73003-5_205

[4] Tuan Linh Dang, Sy Dat Tran, Thuy Hang Nguyen, Suntae Kim, and Nicolas Monet. 2022. An improved hand gesture recognition system using keypoints and hand bounding boxes. *Array* 16 (2022), 100251. https://doi.org/10.1016/j.array.2022.100251

[5] Charles R. Harris et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[6] Simon Fortin-Deschênes, Vincent Chapdelaine-Couture, Yan Côté, and Anthony Ghannoum. 2023. Patent US10838206B2 – Head-mounted display for virtual and mixed reality with inside-out positional, user body and environment tracking. https://patents.google.com/patent/US10838206B2/en

[7] J. D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/MCSE.2007.55

[8] Apple Inc. 2020. *Detect Body and Hand Pose with Vision.* Technical Report. Apple World Wide Developer Conference. https://developer.apple.com/videos/play/wwdc2020/10653/

[9] Apple Inc. 2022. *iPhone models compatible with iOS 16.* Technical Report. Apple Support Website. https://support.apple.com/guide/iphone/supported-models-iphe3fa5df43/16.0/ios/16.0

[10] Apple Inc. 2023. *iPhone 15 Pro and iPhone 15 Pro Max.* Technical Report. Apple Product Website. https://www.apple.com/iphone-15-pro/

[11] Apple Inc. 2023. *iPhone models compatible with iOS 17.* Technical Report. Apple Support Website. https://support.apple.com/guide/iphone/models-compatible-with-ios-17-iphe3fa5df43/ios

[12] Apple Inc. n.d.. *AVCaptureSession.* Technical Report. Apple Developer Documentation. https://developer.apple.com/documentation/avfoundation/avcapturesession

[13] Apple Inc. n.d.. *UIViewController.* Technical Report. Apple Developer Documentation. https://developer.apple.com/documentation/uikit/uiviewcontroller

[14] Apple Inc. n.d.. *VNDetectHumanHandPoseRequest.* Technical Report. Apple Developer Documentation. https://developer.apple.com/documentation/vision/vndetecthumanhandposerequest

[15] Apple Inc. n.d.. *VNHumanHandPoseObservation.JointName.* Technical Report. Apple Developer Documentation. https://developer.apple.com/documentation/vision/vnhumanhandposeobservation/jointname

[16] Sheng Jin, Lumin Xu, Jin Xu, Can Wang, Wentao Liu, Chen Qian, Wanli Ouyang, and Ping Luo. 2020. Whole-Body Human Pose Estimation in the Wild. arXiv:2007.11858 [cs.CV]

[17] James Kennedy. 2010. *Particle Swarm Optimization.* Springer US, Boston, MA, 760–766. https://doi.org/10.1007/978-0-387-30164-8_630

[18] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a

[19] Munir Oudah, Ali Al-Naji, and Javaan Chahl. 2020. Hand gesture recognition based on Computer Vision: A review of techniques. *Journal of Imaging* 6, 8 (2020), 73. https://doi.org/10.3390/jimaging6080073

[20] Timothy F. O'Connor, Matthew E. Fach, Rachel Miller, Samuel E. Root, Patrick P. Mercier, and Darren J. Lipomi. 2017. The Language of Glove: Wireless gesture decoder with low-power and stretchable hybrid electronics. *PLOS ONE* (2017). https://doi.org/10.1371/journal.pone.0179766

[21] Stefano Pellegrini, Konrad Schindler, and Daniele Nardi. 2008. A Generalisation of the ICP Algorithm for Articulated Bodies. In *British Machine Vision Conference.* https://api.semanticscholar.org/CorpusID:12104382

[22] M. Perimal, S.N. Basah, M.J.A. Safar, and H. Yazid. 2018. Hand-Gesture Recognition-Algorithm based on Finger Counting. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 10, 1-13 (May 2018), 19–24. https://jtec.utem.edu.my/jtec/article/view/4115

[23] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. 2014. Realtime and Robust Hand Tracking from Depth. In *2014 IEEE Conference on Computer Vision and Pattern Recognition.* 1106–1113. https://doi.org/10.1109/CVPR.2014.145

[24] Nicholas Santavas, Ioannis Kansizoglou, Loukas Bampis, Evangelos Karakasis, and Antonios Gasteratos. 2020. Attention! A Lightweight 2D Hand Pose Estimation Approach. arXiv:2001.08047 [cs.CV]

[25] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, J. Guerry, B. Le Saux, and D. Filliat. 2017. 3D Hand Gesture Recognition Using a Depth and Skeletal Dataset. In *Eurographics Workshop on 3D Object Retrieval*, Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov (Eds.). The Eurographics Association. https://doi.org/10.2312/3dor.20171049

[26] Ekaterini Stergiopoulou, Kyriakos Sgouropoulos, Nikos Nikolaou, Nikos Papamarkos, and Nikos Mitianoudis. 2014. Real time hand detection in a complex

background. *Engineering Applications of Artificial Intelligence* 35 (2014), 54–70. https://doi.org/10.1016/j.engappai.2014.06.006

[27] Dinh-Son Tran, Ngoc-Huynh Ho, Hyung-Jeong Yang, Eu-Tteum Baek, Soox-Hyung Kim, and Gueesang Lee. 2020. Real-Time Hand Gesture Spotting and Recognition Using RGB-D Camera and 3D Convolutional Neural Network. *Applied Sciences* 10, 2 (Jan. 2020), 722. https://doi.org/10.3390/app10020722

[28] Nuwee Wiwatwattana, Chayangkul Laphom, Sarocha Aggaitchaya, and Sudarat Chattanon. 2016. Origami Guru: An Augmented Reality Application to Assist Paper Folding. In *Information Technology: New Generations*, Shahram Latifi (Ed.). Springer International Publishing, Cham, 1101–1111. https://doi.org/10.1007/978-3-319-32467-8_95

[29] Ying Wu and Thomas S. Huan. 2001. Hand Modeling, Analysis, and Recognition for Vision-Based Human Computer Interaction. *IEEE Signal Processing Magazine* (2001). https://doi.org/10.1109/79.924889

[30] Yi Xiao, Tong Liu, Yu Han, Yue Liu, and Yongtian Wang. 2023. Realtime recognition of dynamic hand gestures in practical applications. *ACM Transactions on Multimedia Computing, Communications, and Applications* 20, 2 (2023), 1–17. https://doi.org/10.1145/3561822

[31] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 2016. 3D Hand Pose Tracking and Estimation Using Stereo Matching. arXiv:1610.07214 [cs.CV]

[32] Thomas Zinner, Oliver Hohlfeld, Osama Abboud, and Tobias Hossfeld. 2010. Impact of frame rate and resolution on objective QoE metrics. In *2010 Second International Workshop on Quality of Multimedia Experience (QoMEX)*. 29–34. https://doi.org/10.1109/QOMEX.2010.5518277

A APPLICATION USAGE

In order to reproduce our results or expand on our work, the source code for our application, FoldAR, is available to download from the GitHub repository listed in Section 3. System requirements include:

- An Apple mobile device running iOS 16.6 or later.
- A USB-to-Lightning or USB-to-USB-C cable rated for data transfer.
- An Apple computer running MacOS 14.0 ("Sonoma") or later.
- Xcode version 15.1 or newer.
- A development environment capable of editing and running IPython notebooks (`.ipynb` files).

A.1 Building FoldAR

To build and deploy the application onto a local iOS device, perform the following steps:

(1) Ensure that Developer Mode is enabled on your iOS device by navigating to Settings > Privacy & Security > Developer Mode, and toggling Developer Mode to "on".
(2) Download the source code repository onto an Apple computer with Xcode installed.
(3) Navigate to the `Source Code` directory and launch `FoldAR_4.xcodeproj` in Xcode.
(4) In the Xcode project navigator, select the parent `FoldAR_4` project item, go to the "Signing & Capabilities" tab, and change the following settings:
   - From the "Team" dropdown menu, select the name/Apple account that is currently signed into Xcode.
   - Rename the "Bundle Identifier" to use a unique ID, such as your GitHub username, using the following format: `userID.FoldAR_4`
(5) Connect your iOS device to the computer using a cable rated for data transfer and ensure that it is listed in the top bar of Xcode similar to `FoldAR_4 > iPhone 8`.
(6) Unlock your iOS device and click the "Build and Run" button (gray "play" triangle) in the top-left of the Xcode window. You should see a "Build Succeeded" message within Xcode and the application will automatically launch on the iOS device.
(7) To close the application either click the "Stop Execution" button (gray "stop" square) or close the application using the iOS app switcher.

A.2 Running FoldAR

To use the application, perform the following steps:

(1) With the application running in portrait mode after launching from Xcode or the home screen, rotate the phone into landscape mode.
(2) Tap the "ParticipantID" field in the top-left of the user interface and enter a name or identifying number using the on-screen keyboard.
(3) Tap or swipe the "Mode Switcher" in the top-center of the UI to select the experiment mode the participant will be performing.
(4) Tap the "START" button in the top-right of the UI to begin recording hand-point data.
(5) To stop recording, tap the "STOP" button in the top-right corner.

A.3 Accessing Data

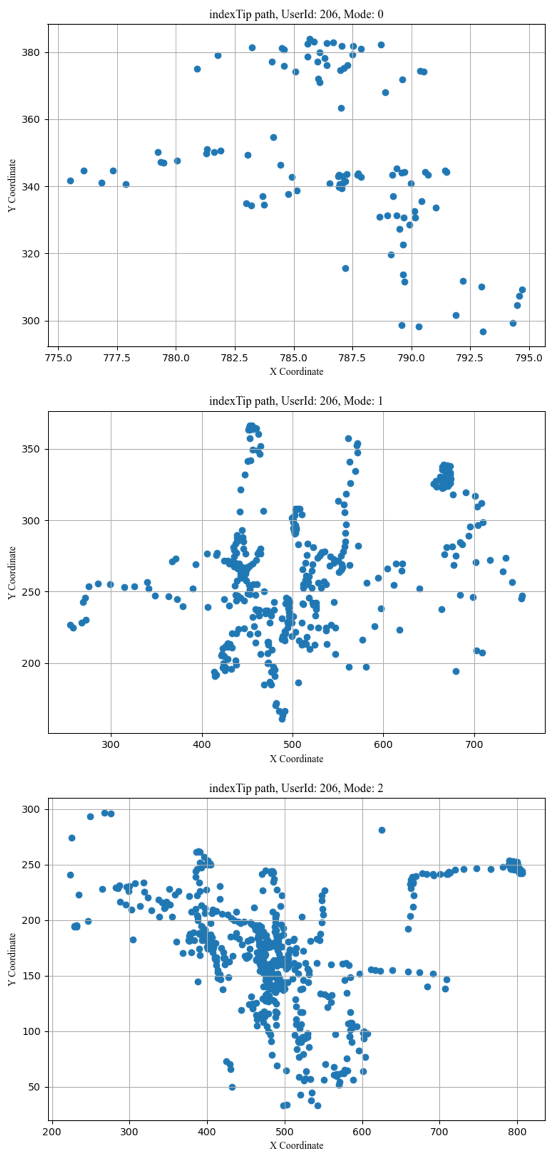
To offload the recorded session data files from the iOS device, perform the following steps:

(1) Open `FoldAR_4.xcodeproj` in Xcode, unlock your iOS device, and connect it to the computer with a cable.
(2) In Xcode's top bar, click on the iOS device model and from the dropdown menu select "Manage Run Destinations...".
(3) In the pop-up window, select `FoldAR_4` under "INSTALLED APPS", then click on the menu button below the section (a circle with three dots, next to the "+" and "-" buttons) and select "Download Container...".
(4) Choose a download location and wait for the process to complete. Once completed the destination location will pop up in the MacOS Finder with the downloaded `.xcappdata` file highlighted.
(5) Right-click the data file and select "Show Package Contents".
(6) Within the package, navigate to `AppData/Documents` and copy the desired `sessionData-x-y.csv` file(s) to another location in the operating system (`x` = `participantID`; `y` = `mode`, 0, 1, or 2).
(7) When finished, disconnect the iOS device and close Xcode, saving any changes if needed.

B SUPPLEMENTARY TABLES AND FIGURES

Figures 10 and 11 are referenced in Sections 6.3 and 7. They are presented in this appendix due to their size.

Received 9 December 2023



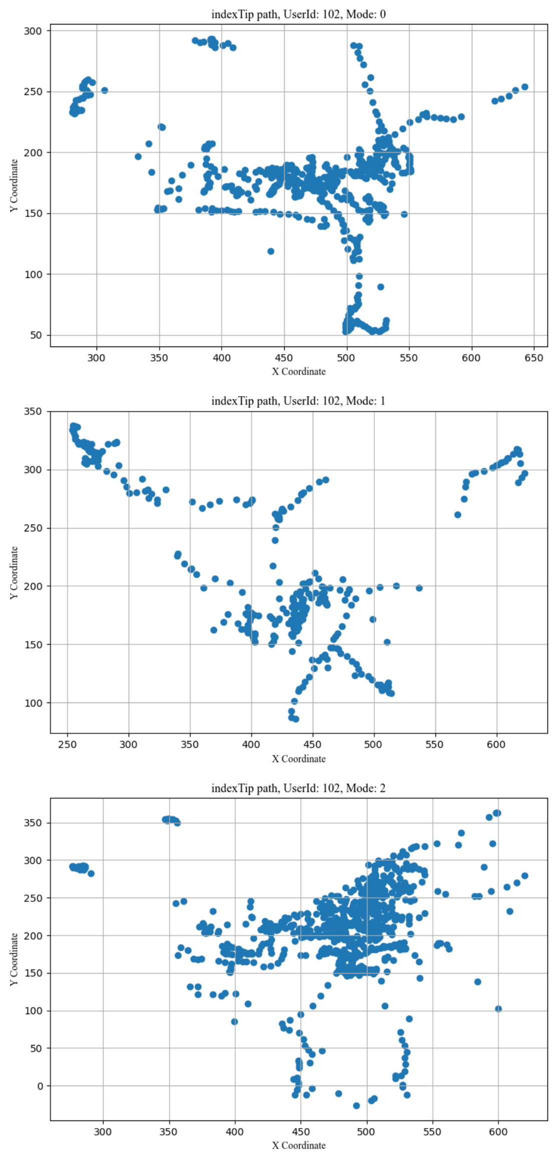


Figure 11: Path of Participant 206’s index fingertips during all three experimental modes

Figure 10: Path of Participant 102’s index fingertips during all three experimental modes