A Step Toward Dynamic Displays and Ecological Data Collection In Cognitive Testing

by

Karunya Sethuraman

S.B., Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author . . . .
Department of Electrical Engineering and Computer Science
August 21, 2020

Certified by. . . . .
Randall Davis
Professor
Thesis Supervisor

Certified by. . . . .
Dr. Dana L. Penney
Director of Neuropsychology, Lahey Hospital & Medical Center
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chair, Department Committee on Graduate Theses

A Step Toward Dynamic Displays and Ecological Data Collection In Cognitive Testing

by

Karunya Sethuraman

Abstract

We have a developed a proof-of-concept for a testing ecosystem for cognitive disorders. This system assists in detection of cognitive disorders by re-implementing existing neurocognitive tests as applications on devices with stylus-sensitive screens. We believe this will allow digital displays to assist in administering the test in a way that steers the subject away from behaviors that would invalidate the test, such as non-sequential test completion. Using the sensors in the stylus and other devices enables collecting data about the subject's incidental behavior, e.g. how long it took to read and respond to directions. This info may aid in diagnosis. With automated data collection and a secure central server, this proof-of-concept also aims to reduce the administrative workload on clinicians.

Thesis Supervisor: Randall Davis
Title: Professor

Thesis Supervisor: Dr. Dana L. Penney
Title: Director of Neuropsychology, Lahey Hospital & Medical Center

Acknowledgments

Contents

Chapter 1

Introduction

1.1 Problem Statement

Neurocognitive testing has been used to assess the cognitive health of subjects in a variety of circumstances, such as head injuries, strokes, and dementia. Increased attention is being paid to neurological testing currently because of the increase in conditions such as Alzheimer's disease. Several tests developed by neuropsychologists, such as the Timed Up and Go (TUG) Test, Tremor Test, Maze Test, and Symbol Digit Modalities Test (SDMT), measure subjects' ability to perform physical and mental tasks.

Advances in digitization allow for these tests to be offered on mobile devices or tablets. Studies show that such digitized tests can be comparable to their pen and paper counterparts. For example, scores from digitized and traditional offerings of the Minnemara battery of tests showed strong correlations across a wide range of ages as well as gender [2]. Björngrim et al. [2] digitized exams with good psychometric properties – the Rey Auditory Verbal Learning Test, the Corsi Block-tapping test, and the Trail Making Test (TMT) – and observed statistically equivalent scores on both the traditional and digital versions of the exams. Fellows et al. [5] noted that the digitized TMT, when offered on a tablet with stylus sensitive screen, could be adapted to gather extra information on timing and pauses. Davis et al. [4] migrated the Clock Drawing Test, traditionally offered with pen and paper, to a digital form using a

digitizing pen and collected data about timing and pauses that are instrumental in evaluating the Clock Drawing Test.

1.1.1 Dynamic Displays

Dynamic displays, displays that can actively change in real time, provide two opportunities for innovation in neurocognitive testing. First, they provide new ways to direct and aid in testing. Second, they aid in the collection of ecological data during a test. Digitally enabled devices can improve the testing process by adapting in response to subject behaviors and can thereby reduce the administrative load on the clinician. That is, re-implementing tests that are traditionally offered on paper adds the ability for the test to adapt in the moment and respond to the subjects as they engage with the test.

For example, a digital Maze solving test on a tablet allows for the opportunity to have the test prevent unwanted subject behavior. When offered on paper, if the subject picks up their pen and starts solving the maze with their eyes, the administrator of the test will stop them and ask them to put the pen back on the paper.

With a tablet, we have the opportunity to engage the tablet in helping to direct the user's behavior. In our system, the application covers up all of the test except for a small square around the pen whenever the pen is picked up off the surface. By drawing the subject's focus back to where their pen is located and limiting the observable area, the subject is encouraged to continue solving the maze using the pen.

With this and similar work, the Davis Lab has found dynamic displays open the opportunity for assessment of a variety of behaviors. For example, in the SDMT test, the subject is asked to fill the exam sequentially, from left to right, and top to bottom. In an electronic medium, this behavior can be enforced by the system itself. Additionally, the frequency with which they move in a non-sequential manner and the total number of times they violate this property can be determined automatically and may provide insight into their overall cognitive function.

Our goal is to design a test that can react just as a test administrator would for a variety of subject actions, as well as collect data that a clinician might not normally

be able to collect, such as the duration of pauses during the exam.

1.1.2 Ecological Data

Traditionally, there is a limited amount of data clinicians can collect using pen and paper and their own senses. In our application, we define ecological data as data about subject behavior that is not directly collected as part of the test; such behavior is not something the subject expects to be monitored during the examination. Examples of ecological data include eye tracking or the time they take to click the "start" button, as well as any number of other measurable actions. Some of these behaviors, such as their reading speed or where they are looking, are instinctive, immediate actions that the subject cannot or would not usually control. We believe that ecological data collection can help aid diagnosis and provide insight into real world functioning. With this innovative approach, we may be better able to predict cognitive state.

1.2 Preview of Results

In this work, we developed adaptable applications that create a testing framework for neurocognitive testing. Digitization of the TUG, Tremor, and Sit/Stand Tests supports the test administrators by collecting and saving the data automatically while administering the test. The measurement of subject behavior on important metrics is almost invisible. For example, in the digitized TUG Test, the test collects information about the user's posture and movement, while the user focuses only on walking. Initial user feedback has shown that the TTS application creates a positive experience for the subject and administrator.

A tablet based platform allows us to create a dynamic and adaptive display that helps administer the test in a way that steers the user away from behaviors that would invalidate the test. For subjects, the application walks them through all the steps and has an easy to use interface. Digitization of the dSDT means the application can reinforce proper testing procedure for the subject as well as collect the data and save it automatically while administering the test.

Chapter 2

Background

2.1   The Symbol Digit Modalities Test

We developed an application that extended the SDMT, using an Apple iPad and an Apple Pencil. The SDMT was developed in 1973 by Dr. Aaron Smith [9]. In the SDMT, there are 9 geometric figures, mapped to the numbers 1-9. The test is given on paper, where the top of the paper has the answer key mapping geometric figures to numbers, while the rest of the paper is a randomized presentation of those nine geometric figures, with blank spaces above for the subject to fill. The subject is asked to write the correct number on the grid above the corresponding geometric figure. The task is timed, with subjects limited to 90 seconds. The subject responds verbally or in writing and their responses are evaluated.

2.2   Digital Symbol-Digit Test

The Digital Symbol Digit Test (dSDT), a novel version of the Symbol Digit Modalities Test, was developed by Dr. Dana Penney at Lahey Clinic and Professor Randall Davis at MIT. It is designed to detect changes in cognitive function in patients who are progressing from normal function to mild cognitive impairment. There are 6 symbols in the dSDT, mapped to 6 numbers, and subjects are asked to match symbols to digits. The test is in turn composed of three tasks, the translation task, seen in

Figure 2-1: Translation Task

Figure 2-2: Digit Copy & Delayed Recall Tasks

Figure 2-1, and the digit copy and delayed recall tasks, seen in Figure 2-2. The translation and copy tasks are designed so that they have identical solutions, which means the physical motion is identical. However, an important testing principle is that the cognitive load differs across tasks; the translation task has a higher cognitive load. In this case, the lower load consists of matching a digit to a digit, while the higher load exam requires matching a symbol to a digit located in a central key. When the test is administered, the subject is shown the translation task first, and then shown the digit copy task and delayed recall task.

2.2.1 Testing Instructions

There is no time limit for the dSDT; the subject is requested to work as quickly and as accurately as possible. The test administrator guides the subject through the sample section, the first 6 cells before the bold line in the first task, but the subject fills in the rest of the numbers independently. In current use, the dSDT is administered on paper, using the Anoto pen, a digital pen which records the subject's pen strokes.

2.2.2 Translation Task

In the translation task, seen in Figure 2-1, subjects are asked to match six abstract symbols with six numerical digits. The symbols and numbers are related by an answer key at the top of the page, which subjects can refer to. The subject fills in the numbers until they reach the stop sign.

2.2.3 Digit Copy Task

Next, the digit copy task, seen in Figure 2-2, is displayed to the subject. The subject simply fills in the numerical digits as seen, and then reaches the stop sign. As noted, the digit copy task uses the same six numbers as in the translation task and has the same solution. This allows the test to differentiate between behavior change arising due to cognitive and physical reasons.

2.2.4 Delayed Recall

After the digit copy task, there has been a delay from when the subject was last matching symbols to numbers in the translation task. The delayed recall task is seen in the bottom right corner of Figure 2-2. The subject is once again asked to fill in the numbers for the six symbols from the translation task, this time without access to the answer key. This allows us to determine the extent to which the subjects have learned the association between any of the symbols and the corresponding digits (this is called incidental learning).

The entire test is then re-administered in exactly the same way. This enables the measurement of change in subject behavior and performance due to prior experience with the test and enables the measurement of implied learning.

2.3 Dynamic Testing in the Maze Test

The Maze test, another assessment commonly used to track the progression of cognitive disorders, was successfully digitized by Jack Cook when he worked in the Davis Lab. The maze test traditionally is administered with pen and paper, and the subject is asked to solve the maze as quickly as they can, without lifting the pen up from the paper. Figure 2-3 shows an example of a maze, unsolved, and Figure 2-4 shows the same maze with the solved path highlighted in blue pen. The application, designed for iPad and Apple Pencil, helps focus the subject on the goal, which is completing the maze correctly, while not lifting up the pencil. If the subject lifts their pencil up, all of the maze test is obscured except for a small circle around where the pen was last drawing. This way, the subject is reminded to place their pen down again on the paper, and once they do so, the rest of the exam is uncovered and they can continue with it. This enables the software to help guide the testing in a way that mimics what a human examiner is required to do.
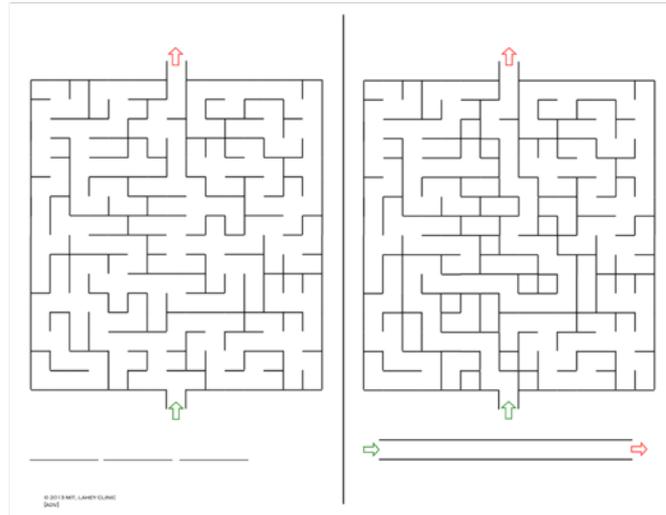
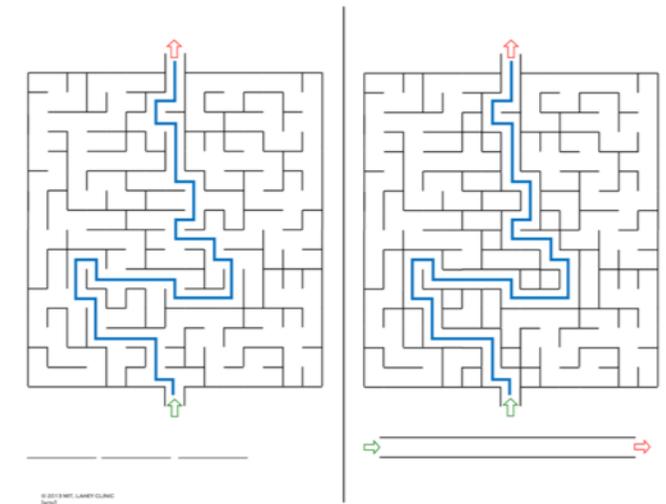Figure 2-3: Unsolved Digital Maze Test

Figure 2-4: Solved Maze Test

2.4 Research On Making Testing Engaging

A team at the Open University made neurocognitive testing engaging by making them into games [10]. This allowed them to gather data on the player's cognitive abilities and state, across multiple factors: working memory, sustained attention, focus, and split attention [10]. They observed how the users' cognition varied by time of day, and extrapolated how an individual's sleep-wake cycle affected their cognitive ability [10]. An important takeaway from this study is that the neurocognitive testing was administered in a mini game format, rather than being communicated as a test, yet they were able to gather a substantial and valid dataset on subject behaviors. This might lead to more research on the effectiveness of neurocognitive testing when it is not formatted or presented like a test.

2.5 Ecological Testing

In cognitive testing, there are two ways to determine how a test relates to everyday life. We can categorize this difference by using verisimilitude and veridicality [3]. Verisimilitude, or in this case how closely the exams resemble the cognitive demands of everyday activities, must be aimed for during the design of the test. Veridicality is the degree to which existing tests are empirically related to measures of everyday functioning [6]. It is demonstrated by using statistical techniques to relate performance on traditional neuropsychological tests to measures of real-world function [3].

It is important to gauge how the subject can best perform and most tests are designed to assess this. However, this does not provide much information about the subject's everyday performance, and reduces the overall ecological validity. In a review of many neuropsychological assessments, Chaytor found that many neuropsychological tests are only moderately related to everyday subject performance [3].

In our work, ecological testing is defined as collecting all the data about the subject's behavior that we can. This may include the time to process instructions, or the delay between button taps. This approach has become more common as emerging

technologies allow for more exploration of this method.

By collecting data about the subject's behavior, such as the speed at which they read, we may be more informed about the subject's everyday performance. We believe that collecting ecological data can help bridge the gap between cognitive testing and everyday performance by assisting in tracking everyday subject behaviors as well when assessing subjects. Generally, ecological assessment can be applied to the entire process of taking a neurocognitive test, and here, we expand the meaning of the test to be the entire time the subject is interacting with the application. Combining the results of the actual assessment with the ecological data collected from different sources allows for a broader view of the subject's overall performance on the same test.

For example, information processing times, which get longer with age, have been shown to be disproportionately slower in those with certain cognitive disorders such as Alzheimer's. With ecological testing, it is possible to note how long a subject looks at the instructions, by simply recording the time when they press the button to advance to the next screen. When combined with the data collected from the assessment itself, this timing data may help the clinician learn more about the state of the subject's cognition. Additionally, since the subject is not aware that the button presses are being recorded, they would not be motivated to rush through in order to get better results, possibly making the data more consistent with the subject's everyday cognitive capabilities.

2.6   Server

The data collected by the applications detailed in Chapter 3 and 4 are sent to a common server, hosted by MIT's Computer Science Artificial Intelligence Laboratory (CSAIL). Originally, the server was written in PHP, as CSAIL legacy servers run Apache. However, this PHP code was not able to handle the data, which was a CSV file that needed to be sent as multipart form data. An existing endpoint on the server was able to support uploading of CSV files by the user clicking on a button, choosing

the file, and submitting it. But, this did not easily lend itself to HTTP requests, as there was no straightforward way to trigger the button submit action code when the HTTP request containing the file reached the server. However, CSAIL also has newer servers that support configurations such as Node, and we chose to use that instead.

The data file collected by the applications is sent to a custom URL hosted on the CSAIL server. The URL is powered by a Node server, which uses a variety of Node modules. Using HTTP, we set up a function to be called every time an HTTP request, such as a push request with the above file, is made to the server. We used a Node module called Express to set up the router, and set up two different URL endpoints on the server that can receive files, one for each application. We send the file from the TTS application to one URL endpoint, where it will be saved according to a process specific to its structure, and we send the file from the Symbol-Digit application to another URL endpoint, where it is processed as needed.

The code for each URL endpoint saves files to a separate folder in the server, so the data from each app is siloed. Additionally, to be safe, each received file is stored twice on the server, as well as once locally on the device before it is sent. As soon as the datafile, which is already free of any identifying data, arrives at the server, it is encrypted using the AES-256 version in OpenSSL, and the unencrypted file is immediately removed. We initially explored using Apple specific encryption libraries on the application end to encrypt the file before sending it to the server. This did not function well, as the many differences in the implementations of AES in Apple frameworks and OpenSSL resulted in errors when trying to decrypt the file on the server. Unfortunately, OpenSSL is not supported on Apple, and the Apple supported encryption libraries do not run on Linux, so there is not a simple way to use the same encryption library on both ends.

Once the file is saved to the server, the development team is sent an email confirming that the file was saved and encrypted in the server. To make sure the server is always running, we built it as part of a Linux service that runs as root. The service is created with systemd, and is used to run a script that starts the Node server. The service is given the path to the server code, as well as the configurations necessary

20

to start the server. This service, which runs as root, needs to have the correct permissions to both access the Node server code and run it. In order that root be able to access it, the server code resides in the home ubuntu folder of the CSAIL server. Additionally, with systemd we are able to have the service restart the Node server automatically on failure, ensuring that the server and therefore both endpoints will always be available to receive files from the application. To further avoid data loss, we have set up a daily Chron job at midnight that copies all the files from that day, now encrypted, over to a secondary secure server, which allows us to have one more distributed backup of the data.

Chapter 3

TTS Application

3.1 Overview of the Tests

TTS stands for a battery of tests meant to assess current performance and predict future fitness: the TUG, Tremor, and Sit/Stand tests.

A variety of research has shown that the TUG test can accurately assess mobility, balance, walking ability, and risk of falling in older adults [7]. Depending on the subject's age, gender, and presence of other disorders, such as Alzheimer's, Arthritis, Cerebral Palsy, or Frail Elderly, the score they earn on the TUG test can be used to predict their future health [7].

The Tremor test is a performance-based way to assess the functional impairment caused by a tremor; it involves observing the subject's hand trembling, instead of using a questionnaire. In a study by Louis et al. [8], a different performance-based assessment of function in essential tremor was found to be internally consistent and valid across cases. It is important for the clinician to have an objective means to quantitatively measure the tremor, which this Tremor test provides.

The Sit/Stand test, where the subject is asked to stand up and sit down repeatedly in a set time period, relates to the subject's overall functional independence, which is a good indicator of overall life function. Traditionally, the subject's movement is recorded manually. In a study from the Netherlands, a team found that an instrumented version of the Sit/Stand test is more clinically relevant than the manu-

ally recorded version, as the results were more "strongly associated with participant health status, functional status and physical activity ... in older adults" and the instrumented format allowed for "assessment of the dynamic phases of the test, which were likely more informative than the static sitting and standing phases". [11]

3.2 Background

The TTS application is designed for use on the iPhone, and collects motion data on the test subject as they proceed through the steps of each test. It was built partially by William A. Rodriguez Jimenez in the Davis Lab. This application aids the clinician, who watches the subject and scores them during the exam. The application measures posture and movement at the rate of 100 times per millisecond. This data can be visualized as a graph or analyzed for further quantitative metrics.

3.2.1 Tremor Test

When the subject takes the Tremor Test, the iPhone is held in the hand, face up, in the hand they are testing (first dominant, then non-dominant). Depending on their handedness, the subject will test either the right or left hand first. The Tremor Test uses the gyroscope and specialized sensors available in the iPhone to measure motion data in detail. In Figure 3-1, the diagram demonstrates the three axes on which the gyroscope collects data. Movement in the z-axis in Figure 3-1 corresponds to moving the hand clockwise or counterclockwise, movement in the x-axis corresponds to flexing the wrist up and down, and movement in the y-axis corresponds to twisting the whole forearm.

3.2.2 Sit/Stand Test

During the Sit/Stand Test the subject wears the phone on their chest in a halter. Once they start the Sit/Stand test, they stand up and sit down as many times as they can in 30 seconds. Their motion data, collected by the iPhone internal apparatus,
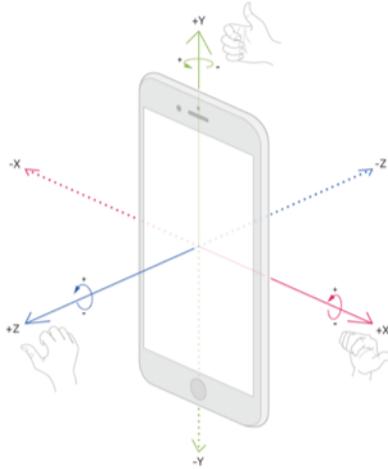
Figure 3-1: Gyroscope Motion Data On Apple iPhone

is recorded and displayed on the screen. The z-axis or the roll axis corresponds to moving left or right as you are standing. The x-axis is pitch which corresponds to leaning forward or back. The y-axis, or yaw, corresponds to turning the body left or right.

3.2.3 TUG Test

Similarly, during the TUG Test, the subject wears the phone on their chest in a halter. The app records the motion of the subject, in all directions, allowing for an objective evaluation of their movement along with a display of their motion data.

3.3 System Architecture

The basic architecture for the Sit/Stand and TUG Tests was kept as designed by Rodriguez.

3.3.1 Tremor Test

We duplicated the Tremor test for each hand, so that we could gather handedness specific tremor data. To this end, the application was also augmented to collect handedness as part of the subject information dialog, as it is important when analyzing

the Tremor data to be able to know whether it was the subject's dominant hand, non-dominant hand, or if they are ambidextrous. Additionally, we augmented the demographic data collected, so that the handedness information was included in the file.

3.3.2   Data Collection

Initially, the TTS app had a race condition in the recording of the data. The app would begin to graph the data before the application was fully finished recording, resulting in undefined behavior and errors. This was solved by delaying the start of data visualization by a few milliseconds.

The TTS app sends the data from the iPhone as soon as the test is done, without requiring any further action from the user. This was designed to remove the need for interacting with the share button dialog at the upper right hand side of the screen. The dialog could be confusing to administrators or users, who might forget to send the data or not realize the share button dialog was the way to do it. The subject demographic data is collected once, and added to the header of each file sent to the server. Once the graphs of the data appear on the iPhone screen, and the data from that test has been sent to the secure central server, described in Chapter 2, the same file is also saved locally. Saving the data in multiple locations automatically helps avoid data loss.

The figures below are the screenshots of the data visualizations created by the TTS app after the Tremor, Tug, and Sit/Stand Tests. The pitch, roll, and yaw can be understood using the diagram in Figure 3-2. The pitch, roll, and yaw data is in radians. The accelerometer data is in units of gravitational force, and the gyroscope data is in radians per second. In Figure 3-3, we can see the normal data from someone with no tremor. As expected, it is mostly constant, with the sensitive gyroscope picking up natural movement. In Figure 3-4, we can see the tremor test data for someone whose hand is shaking a lot. We see that the roll has the biggest amplitude on the graph, signifying that their hand is shaking a lot side to side, if we imagine the yaw as perpendicular and upwards from their hand with the phone. In Figure
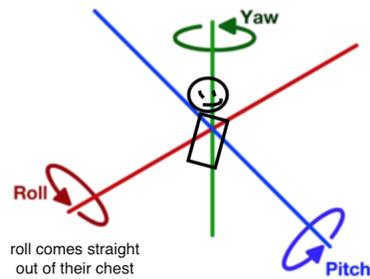
Figure 3-2: Pitch, Roll, Yaw

3-5, we can see the TUG test data for a normal subject. Since they turn the same way twice during the test, the yaw is almost 6.28 radians, or two full rotations. In Figure 3-6, we can see the TUG test data for someone who's impaired. It takes them a lot longer to get up, as seen in the accelerometer data – there are two peaks before the large peak where they get up, and the peaks in the accelerometer are further apart than in a normal test, as they are walking more slowly. In Figure 3-7, we can see the Sit/Stand test data for a normal subject, who is able to get up and sit down frequently in the allotted time period. In Figure 3-8, we see sample Sit/Stand test data for a subject with underlying disorders, who takes a long time to initially get up, and then is only able to complete the test a couple times.
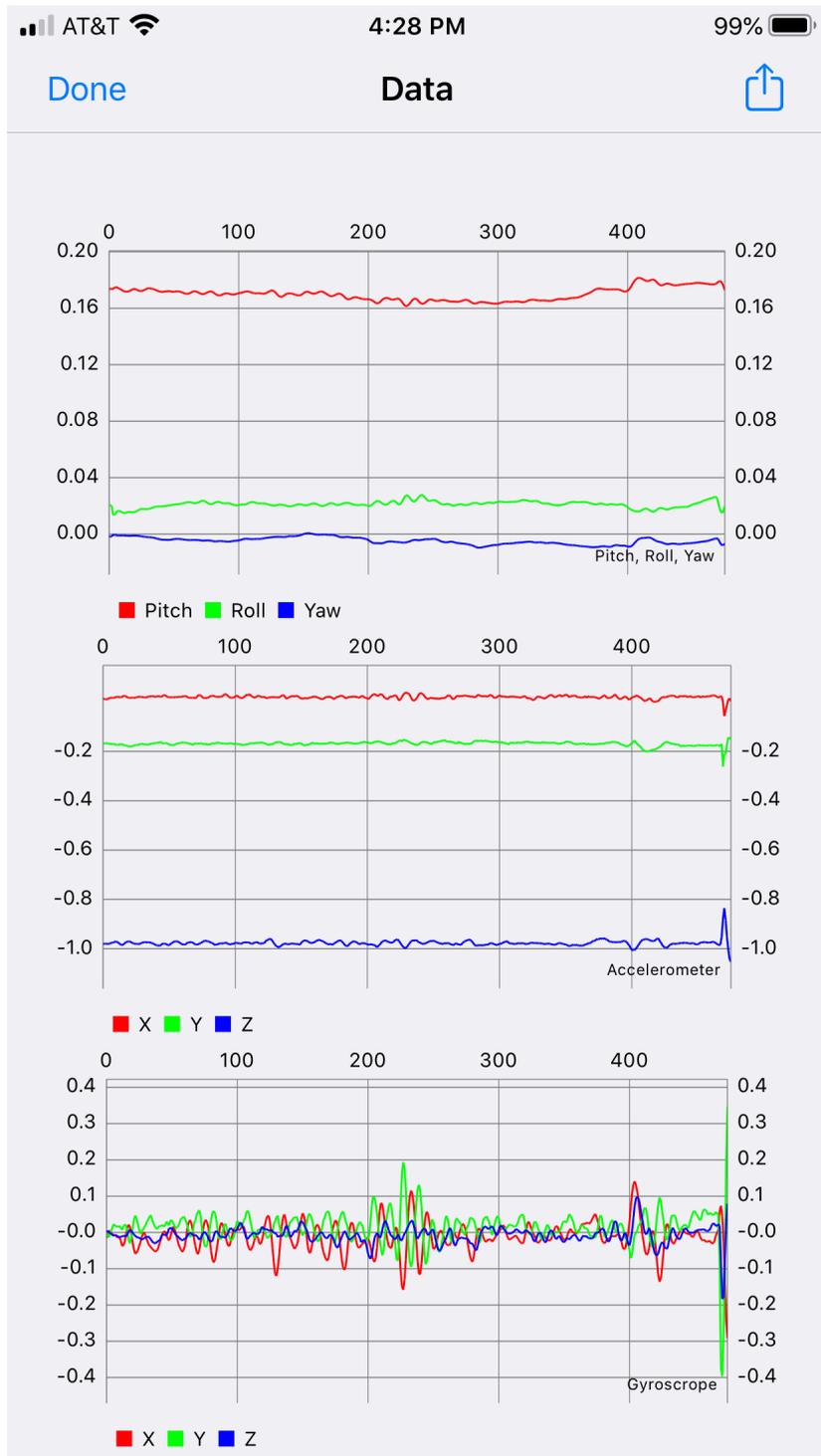
Figure 3-3: Tremor Test Data: Normal Subject
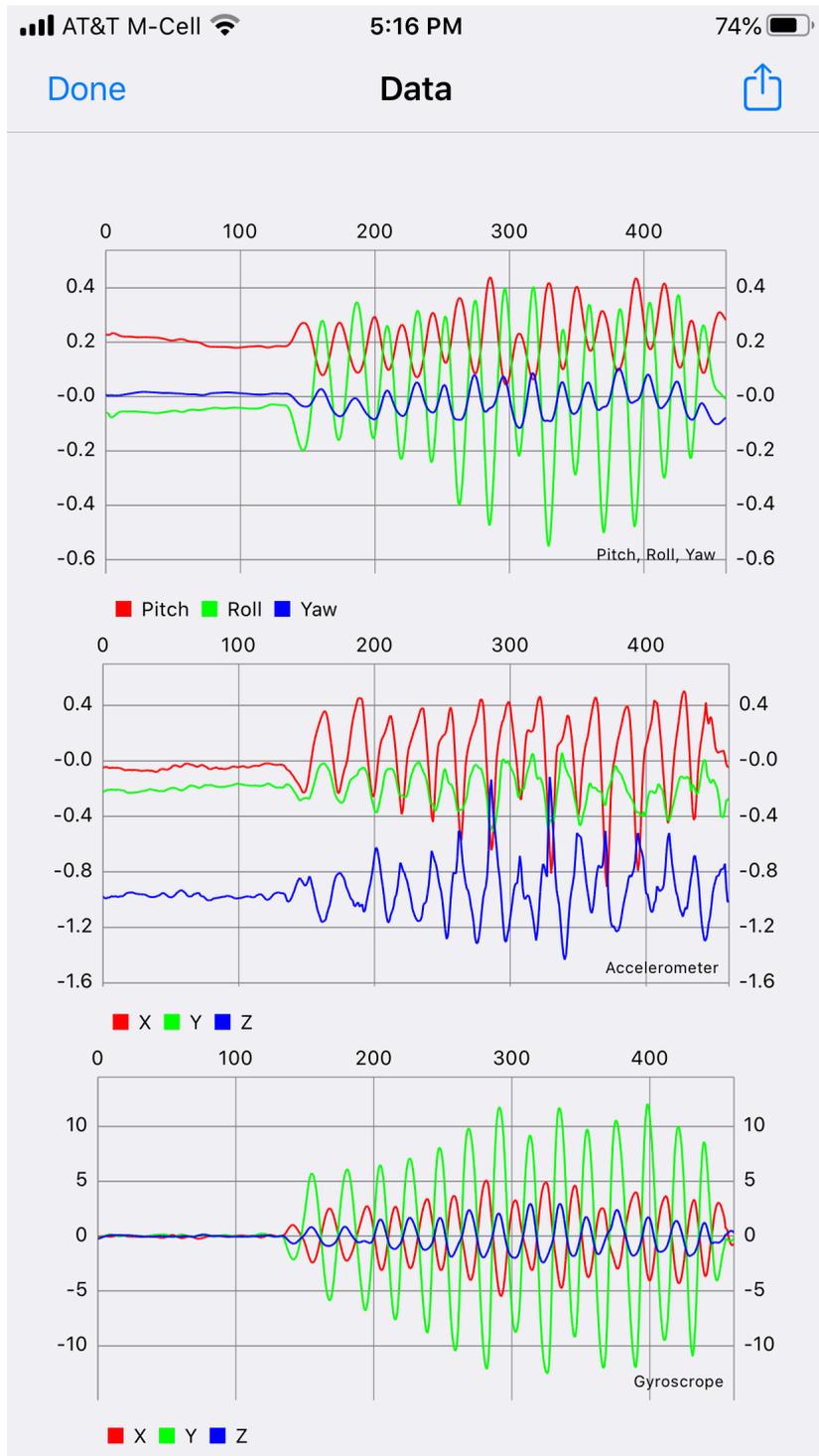
Figure 3-4: Tremor Test Data: Subject with Essential Tremor
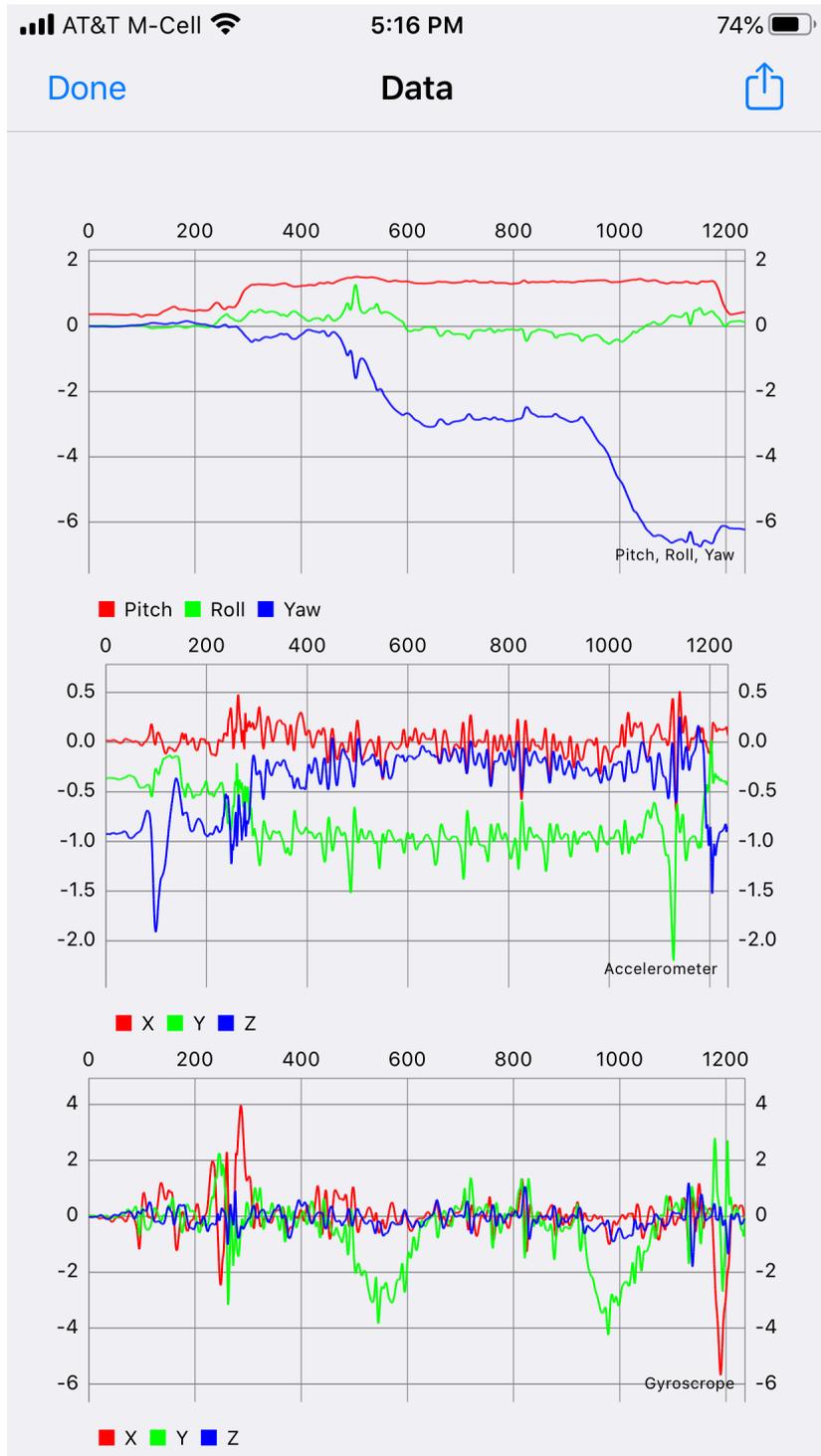
Figure 3-5: Timed Up and Go (TUG) Test Data: Normal Subject
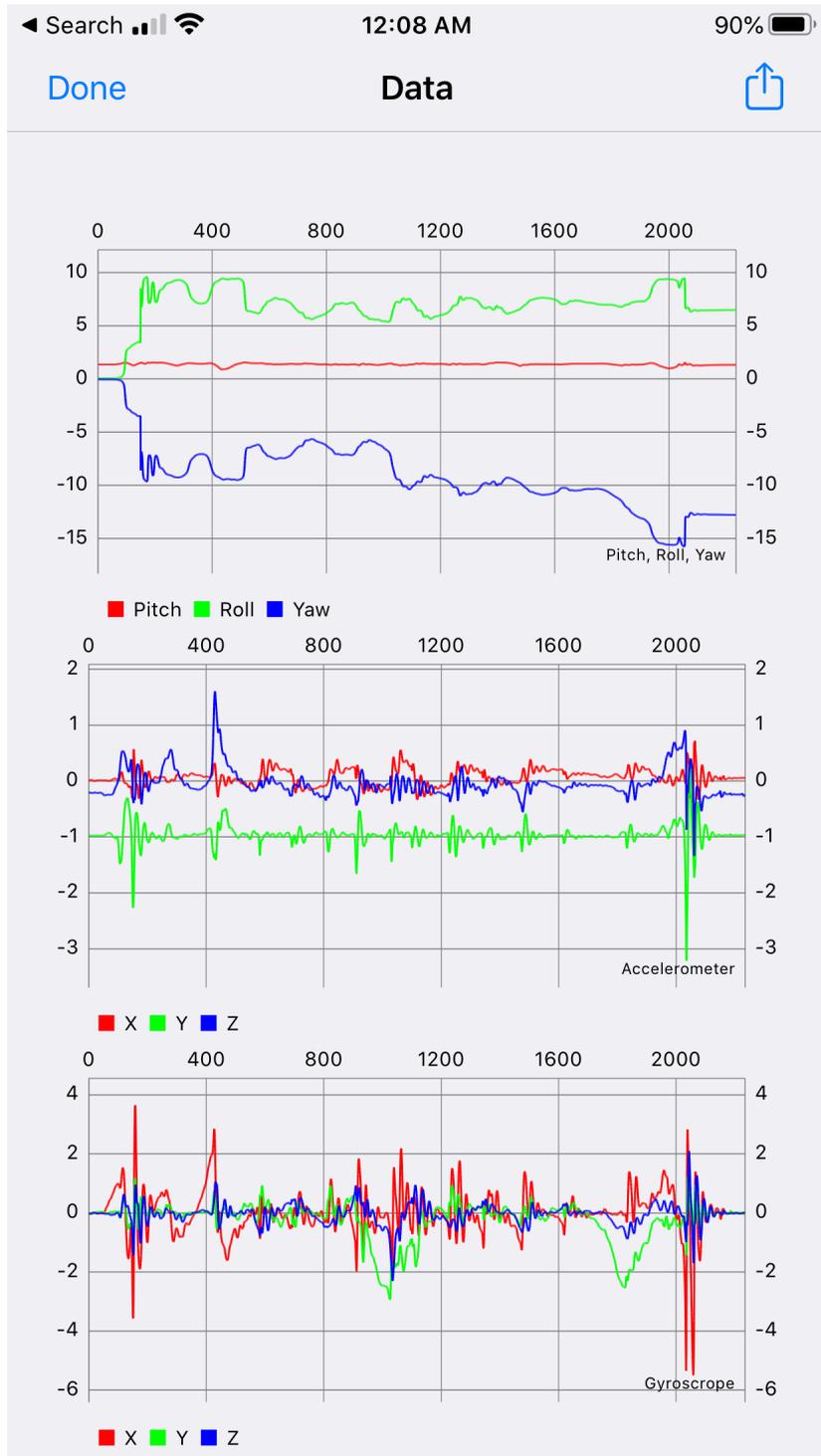
Done   **Data**   ⬆️

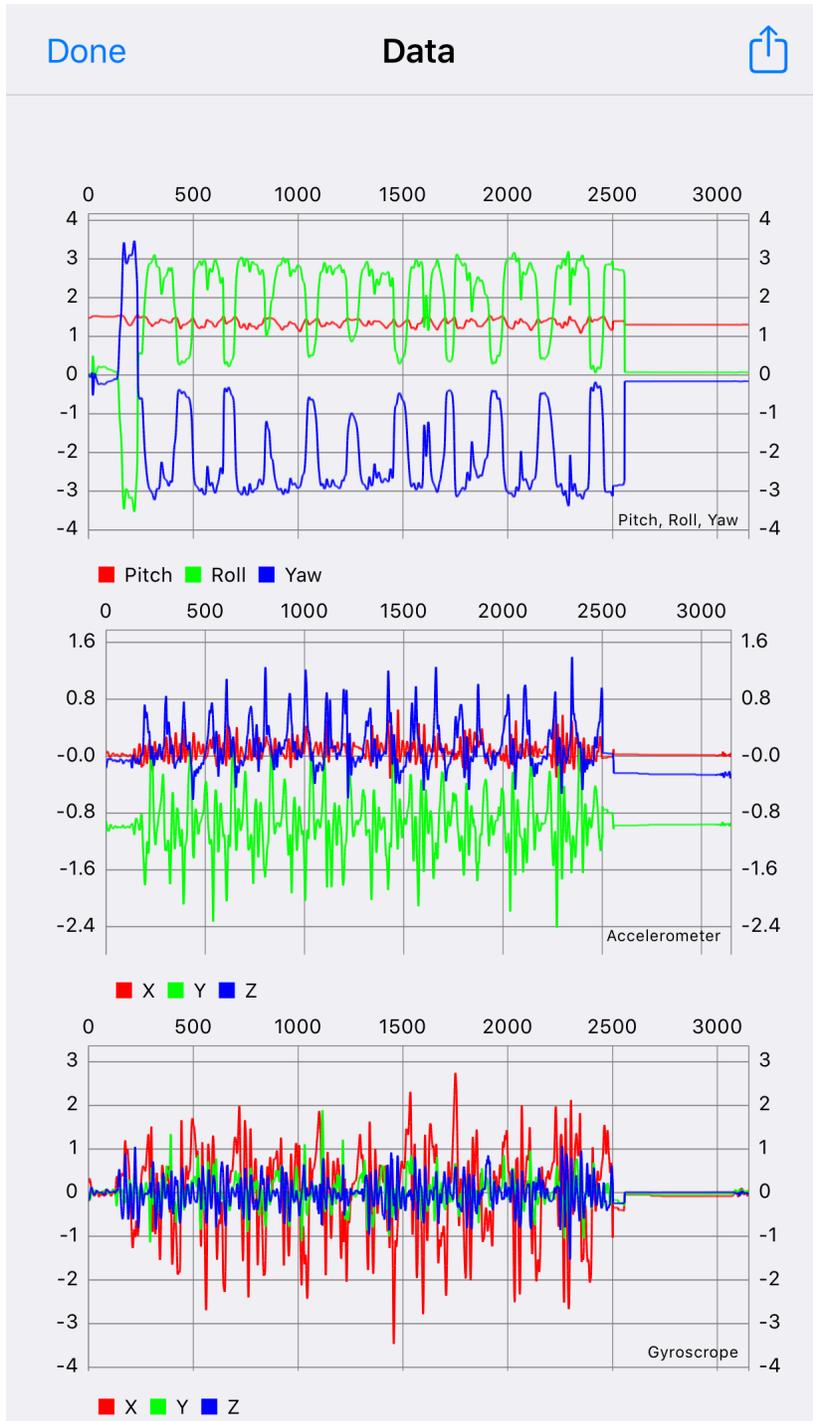Figure 3-6: Timed Up and Go (TUG) Test Data: Subject with Cognitive Impairment

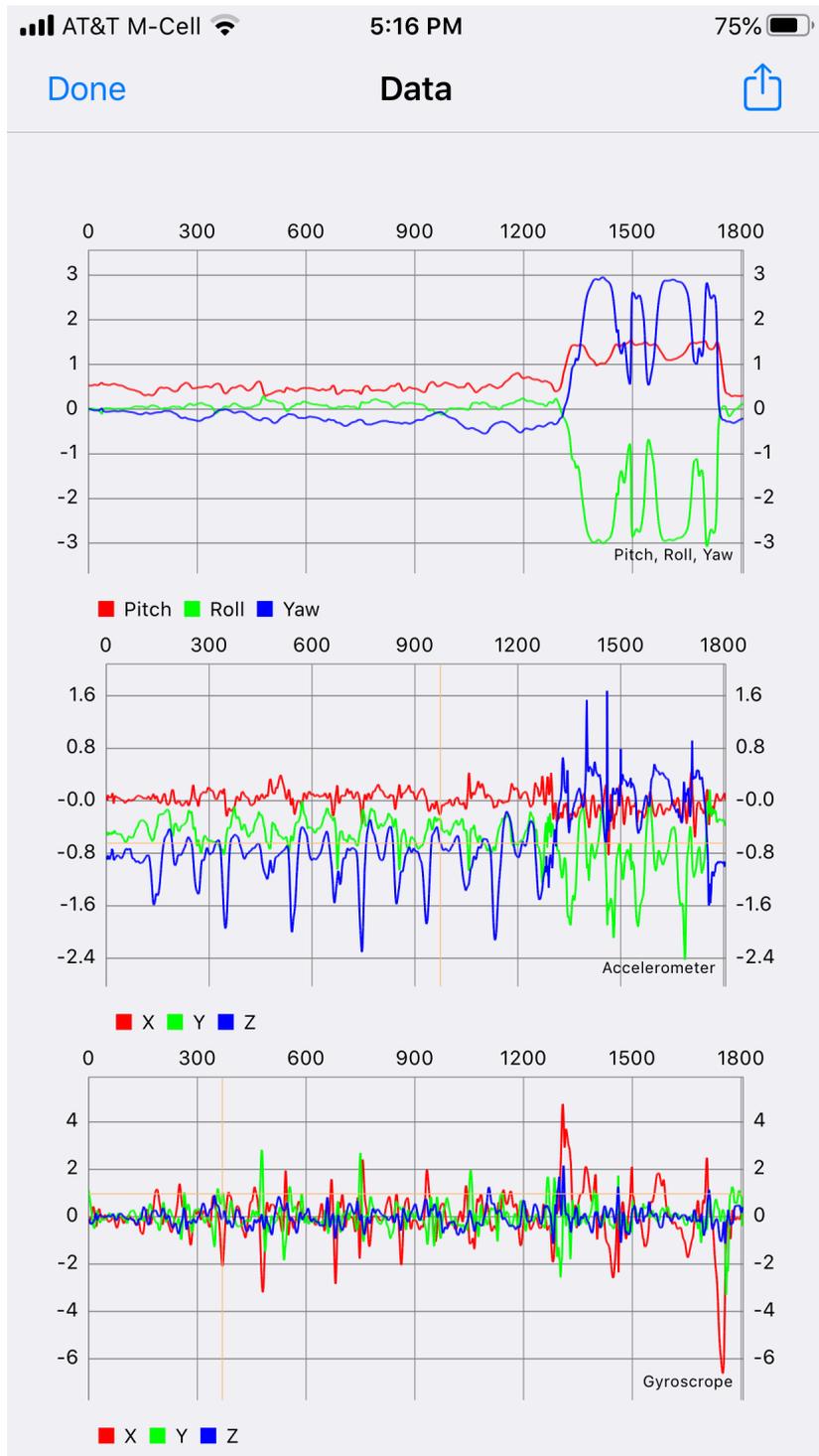Figure 3-7: Sit Stand Test Data: Normal Subject

Figure 3-8: Sit Stand Test Data: Subject with Cognitive Impairment

Chapter 4

Symbol-Digit Application

4.1 Hardware

The Symbol-Digit application is a re-implementation of the dSDT, designed to run on an Apple iPad. To provide the same experience as a paper test, the application maintains the test grid at a constant size, but can scale the white space around it on the iPad display to accommodate larger iPad screens. Thus, the application can adjust to larger iPads. The other component of the pen and paper test is provided by an Apple Pencil. Using the sensors in the Apple Pencil, which sample data multiple times per second, we are able to collect the location where the pencil touches down, the azimuth angle, the altitude that it is held at, and the current time. The pencil has several sensors in its tip as well as its case that detect and send the pencil's pressure and positioning. Figure 4-1 outlines the meaning of these measurements in a 3D context.

4.2 Application Overview

The application's initial screen is simple, with the title and the name of the lab. This screen leads to the subject information page, which has a variety of input fields so that the test can also track the assistive devices, gender, handedness, and location information of the subject. The information collected here can be augmented or
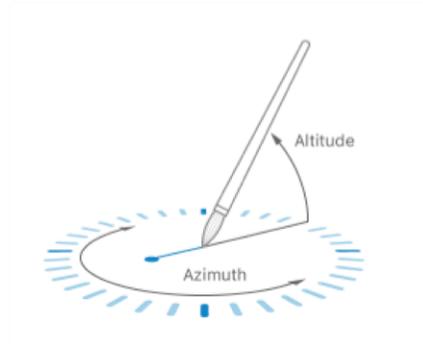
Figure 4-1: Apple Pencil Data Angles

reduced with ease. This information is incorporated into the data file sent to the server. At any point, the subject can press the back button and revisit a previous screen, in case they have any doubts about the screen they are on, or need to change something. The subject information can be edited and re-saved by the user. This way, it is maintained but can be changed if needed before it is sent as part of the test results at the end. Please refer to the system design diagram in Figure 4-2 for a visual representation.

Upon saving the subject information, the application displays the test selection page. Test selection allows the user to select the test variant. However, the administrator would have to advise the subject which test variant to select based on their needs. It would of course be very easy to hide some of the options on the test selection page if the application were to be used for just one variant. Moreover, the process to increase the variants offered on this page or vary the tests themselves is straightforward. This is because the testing interface, detailed below, allows for rapid generation of tests with any combination of six symbols and six digits.

4.3   System Architecture

The test itself is implemented using a stack of four layers: the base view, the arrow view, the overlay view, and the draw view, as seen in Figure 4-3. Both tasks of the Symbol-Digit have the same fundamental structure, but the datafile structure varies, as does the grid layout, to accommodate the format of each task. The code controls
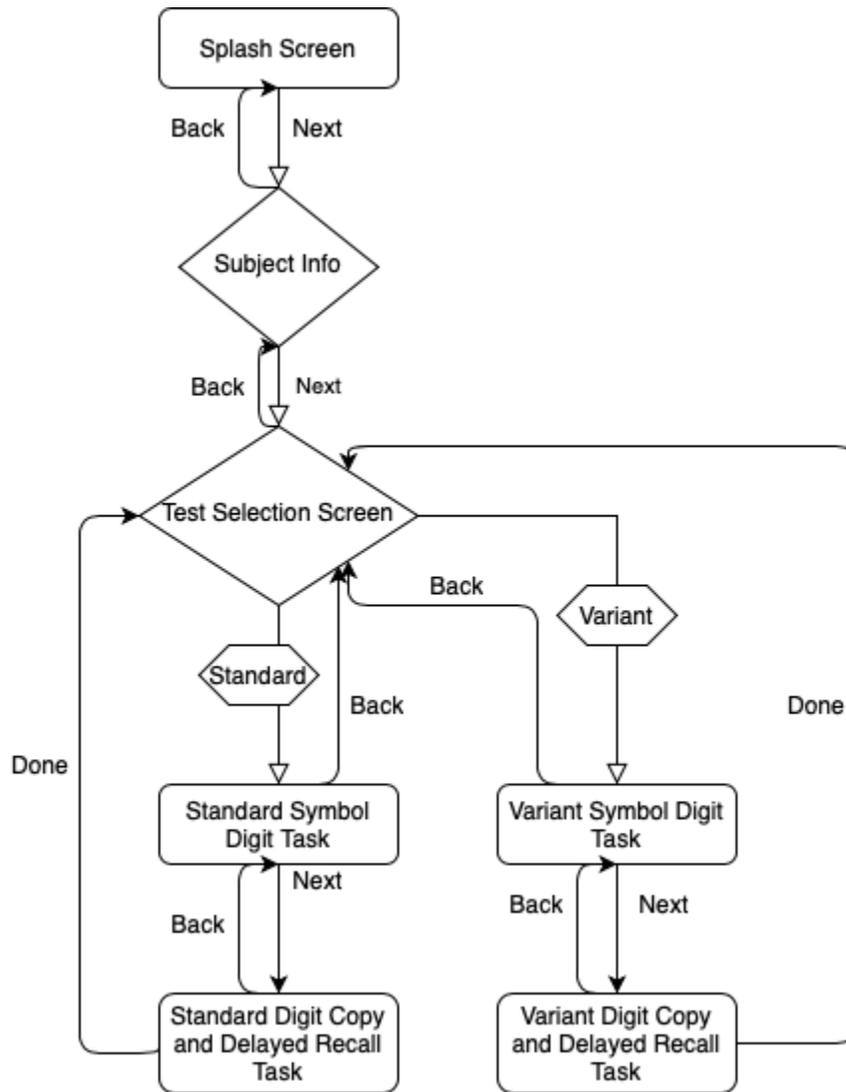
Figure 4-2: System Design of Symbol Digit Application
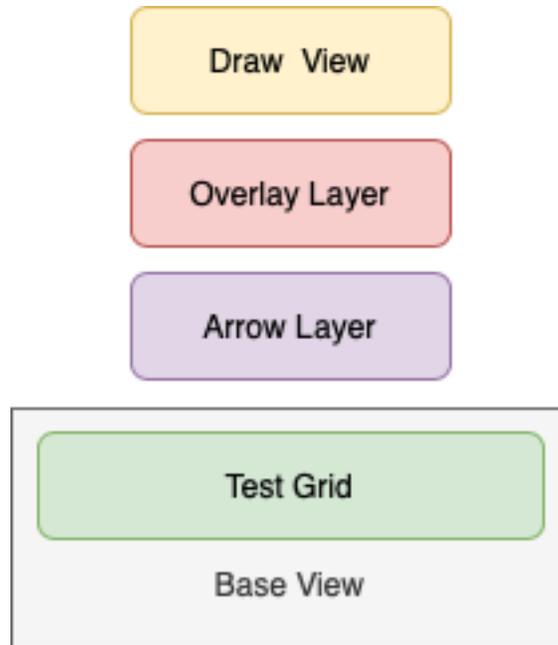
Figure 4-3: Layer Structure of Test Task Pages

the color and width of the pen strokes drawn on the draw view, so the digitized test can be made to look similar to the paper and pen version.

The layers are stacked atop a base background, which is simply white. The base view holds the test itself, which is constructed as a grid into which the geometric or numerical symbols are placed. This placement is determined by the datafile that the page is based on.

The datafiles are used by the code in the base view to specify the symbol/number to display in each cell, and the pattern in which they are placed. The test is designed with 6 symbols that repeat in groups of 6, with the same symbols never appearing next to each other on the grid, and the datafile enforces this requirement. The datafile first specifies the names of the symbol and numerical symbol images. The next six rows correspond to the positions of each of the first six symbols. A sample datafile can be seen below. In the example, the "circle" symbol, on row 1, will be filled into cells 46, 49, 86, etc, according to the list in row 13. Row 2 indicates that the next symbol is the "arrow", and row 14 indicates the cells where it should be placed. Simply by creating a new datafile and adding the necessary symbol images, it is easy to rapidly generate a new version of the exam. Given the similarity in the layout of both of the

tasks of the dSDT, we were able to implement both tasks with the same software, just using a datafile to specify exactly which symbols or digits should be presented.

```
1   circle
2   arrow
3   clover
4   hourglass
5   trapezoid
6   star
7   one
8   two
9   three
10  four
11  five
12  six
13  46 49 86 89 126 128 134 172 179
14  42 48 85 88 97 133 138 168 175
15  45 52 84 93 96 132 135 171 174
16  43 53 55 92 94 131 137 170 178
17  47 50 54 90 95 129 139 169 176
18  44 51 87 91 127 130 136 173 177
19  180
```

To add the images to the grid, the application leverages a custom cell class that embeds a UIImageView, the Apple image display default class, in each cell. The UIImageView is resized and constrained to the dimensions of the grid cell, and then this custom cell class is used to create each cell of the grid. As the cell now has the correctly constrained UIImageView in it, the next step is to simply set the image of the cell to the one indicated by the datafile.

This grid, which we can think of as the test view, is then wrapped in a container before being added as a subview to the base view. This allows us to resize the grid easily using the container, as well as abstracts the test grid itself into an object. This container view is then added to the base layer view displayed on the screen. The container view provides a structure that accommodates changes to the design of the test itself without affecting any of the layers that are placed above it. Once constructed, it looks like Figure 4-4.

Once the base test is set up, there can be multiple layers above the test grid layer which enable the test to help administer itself and dynamically display responses to subject behavior. The set of layers between the test grid and the draw view is where the implementation of the dynamic aspects of the display occurs. Examples of these layers are the arrow and overlay layers, described in detail below.

◯	⇨	✪	⊠	▽	✧
1	2	3	4	5	6

Sample

⇨	⊠	✧	✪	◯	▽	⇨	◯	▽	✧	✪	⊠	▽	⊠

✪	⇨	◯	✧	⇨	◯	▽	✧	⊠	✪	⊠	▽	✪	⇨

◯	✧	◯	▽	✧	⊠	✪	⇨	◯	✪	✧	⊠	⇨	▽

⇨	▽	⊠	✪	◯	✧	✪	⇨	▽	✧	⊠	◯	STOP

Figure 4-4: Translation Task without Overlay

40

The arrow layer lies atop the base view. The purpose of this layer is to prompt the subjects to move on the next row if they stop at the last cell of the row they have just filled out. Although the subjects are told in the instructions how to fill out the exam, for a variety of reasons, some subjects may need prompting at the end of the first row. If they pause for more than 4 seconds after filling in the last cell of any row, this layer appears, as seen in Figures 4-5 and 4-6. This layer works by checking 4 seconds after each stroke ends to see if anything has been written since. If not, the subject has paused. If they are at the end of the row, the dynamic display responds. A partially transparent white rectangle covers all rows of the test completed so far, but not the key at the top of the test, and a red arrow points to the first cell of the next row. Simultaneously, the iPad announces in English, "Please continue." When they start writing again, the blur and arrow are removed. This layer appears whenever the subject pauses for more than the 4 second time limit at the end of rows, but will not appear if they pause anywhere else in the exam.

The overlay layer, which lies atop the base view containing the test, is another example of the kind of dynamic behavior this system offers. The purpose of this layer is to keep the subject filling in their answers to the test sequentially. We check this by maintaining the position of the last stroke that was filled in sequentially and the corresponding cell index. If the next stroke's position falls in a cell that is not the next sequentially, they have skipped forward. Then, this layer is enabled, as seen in Figure 4-7. A black rectangle covers the test, but not the key at the top of the test, and a cutout circle focuses on the square they should be filling out and the symbol above that square. Until they start filling in the test at the correct spot, i.e., the next empty box, the layer will not disappear. In addition to blocking out the rest of the task, the iPad also announces in English a reminder to fill the test out sequentially. This layer appears whenever the subject skips cells, but will not appear if they revisit a cell they have already written in, as that is allowed by the structure of the test.

The draw view allows for data collection when it comes to the numbers that subjects are writing in the answer slot. We record every touch of the Apple Pencil to the screen, as well as its timestamped location and position. The draw view uses

Figure 4-5: Translation Task with Arrow Layer After First Row Pause

Figure 4-6: Translation Task with Arrow Layer After Second Row Pause

Figure 4-7: Translation Task with Overlay

a drawing framework written in Swift called SwiftyDraw, written by Andrew Walz and maintained by Linus Geffarth, found at https://github.com/Awalz/SwiftyDraw. SwiftyDraw keeps track of touch events and creates and renders penstrokes from those. SwiftyDraw calls a method every time the Apple Pencil touches or continues touching the view. Each part of a stroke thus recorded is saved as an instance of the Symbol-Touch class, which holds the time, x and y location, altitude, azimuth, and force of the touch. When the stroke is concluded, SwiftyDraw once more calls a method, and the entire stroke, consisting of the multiple SymbolTouch objects created earlier, is saved as the next element in another custom data structure: SymbolTestFinal, which holds all the strokes for that administration of the test. The use of SwiftyDraw means that the app knows and therefore can dynamically respond whenever the subject starts, stops, or is drawing. This provides a strong foundation upon which new features can be added to facilitate and measure the subject's activity. At the end of the exam, the strokes in the SymbolTestFinal object are packaged into a data file before being sent to the server.

While the digital pen uses a coordinate system that has its (0, 0) at the top left of the paper, the Apple Pencil has its (0, 0) as the top left of the screen, and it took some basic bookkeeping to ensure that the Symbol Digit Application reported positions that used the same coordinate system as in the paper and pencil version of the test.

In the dSDT, the subject uses an Anoto pen, a digitized pen that digitizes and records all their penstrokes. When the subject is writing in the top left cell of the digit copy task, it looks the same as when they are writing at the top left cell of the translation task, but these are at different places on the physical paper. When the test was administered with paper and the Anoto pen, the pen picked up on this difference and recorded it accordingly. For the sake of consistency in data collection, I needed to adjust the iPad measurements to match this. The raw data from the iPad test needed to be offset in the x, or horizontal direction, by the distance from the left side of the paper to the leftmost side of the grid in the paper test. The y-direction offset for the first task, Symbol-Digit, is the distance from the top of the paper to

the top of the answer key in the paper version of the test. For the second and third tasks, the y-direction offset is the distance from the top of the paper all the way to the top of the answer key that is under the translation task.

The process of creating the digitized version of the dSDT required converting the dimensions of the paper version into the units of the iOS application, so that the digitally rendered test matched the paper test exactly in appearance. This was done for multiple reasons. First, the test experience must be exactly the same in all possible ways so that the test can be shown to be comparable to the Anoto pen and paper version. Additionally, the Apple Pencil stroke data needed to be converted into the same units as the data collected when using paper and Anoto pen, so that the data could be processed using the pipelines set up for the paper and Anoto pen version. In order to convert the Apple pencil data, the application code first converts the raw pencil data, which is in Apple CGFloat units, to millimeters (131.95 CGFloat = 1 mm), and then converts the millimeter values into Anoto units (0.3 mm = 1 Anoto unit), hence 39.585 CGFloat = 1 Anoto unit.

This data, now converted into Anoto pen units, is processed by the application into a data file with the same format as the data that was collected previously when the test was given with a digitizing pen. Therefore, all of the pen stroke and data analysis tools that have been previously built can be applied to data from the iPad test.

The subject information is added to the header of the file. Then, each stroke is given a stroke ID, and the specific touch data is combined into the strokes. Each stroke has the number of samples, or individual touches, that compose the stroke. The timestamp and color of each stroke is recorded as well, so that the THink software can use that information. The samples are each of the data points collected from the Apple Pencil. The data, which is in Anoto units, can now be sent to the server and then imported into the THink software. This allows for further analysis, and creates a replay of the pen strokes so a clinician can watch the subject taking the test afterwards.

An example data file can be seen below. The first stroke has 18 samples, and the

start time is the UNIX start time. The first two values in each sample row are the x and y position. The third value is the delta time, calculated as the milliseconds since the last sample was recorded. The next three values are the force, altitude, and azimuth of the Apple Pencil. The force is measured as the force relative to an average force of 1.0, which is set by Apple. The first stroke is the digit 1, naturally a vertical straight line, so my pencil position stayed fairly constant while writing, accounting for the lack of change in altitude and azimuth over the time period. The lack of change in x and y in rows 15 through 21 signify a slight pause in writing.

```
1   <ClockSketch version="2.0">
2   <PenData>
3   Pen id: C4MRDB67GWTJ
4   Number of pages: 1
5   Page address: 21.1351.20.118
6   Page bounds: 0 0 930 719
7   Number of strokes: 4
8   StrokeID: 1
9   Number of samples: 105
10  Color: 0 0 255
11  StartTime: 1591914263.648
12  89.7467 151.7773 0 0.3333 1.3160 0.3154
13  89.7467 151.7773 27 0.3333 1.3160 0.3154
14  89.7467 151.7773 0 0.0417 1.3160 0.3154
15  89.7467 151.7773 16 0.0000 1.3160 0.3154
16  89.7467 151.7773 16 0.0000 1.3160 0.3154
17  89.7467 151.7773 17 0.0000 1.3160 0.3154
18  89.7467 151.7773 16 0.0000 1.3160 0.3154
19  89.7467 151.7773 17 0.0000 1.3160 0.3154
20  89.7467 151.7773 16 0.0000 1.3160 0.3154
21  89.7467 151.7773 15 0.0000 1.3160 0.3154
22  89.7467 151.7773 15 0.0401 1.3160 0.3154
23  ... 80 more lines
24  108.0380 151.7773 15 0.3698 1.3160 0.3154
25  108.3589 151.7773 16 0.3722 1.3160 0.3154
26  108.3589 151.7773 16 0.3888 1.3160 0.3154
27  108.6798 151.7773 17 0.3637 1.3160 0.3154
28  109.0007 151.7773 16 0.3586 1.3160 0.3154
29  109.0007 151.4565 16 0.3538 1.3160 0.3154
30  109.3217 151.4565 17 0.3530 1.3160 0.3154
31  109.3217 151.4565 18 0.3642 1.3160 0.3154
32  109.3217 151.4565 16 0.3690 1.3160 0.3154
33  109.3217 151.4565 17 0.3323 1.3160 0.3154
34  109.3217 151.4565 14 0.2472 1.3160 0.3154
35  109.3217 151.4565 16 0.1400 1.3160 0.3154
36  109.0007 151.4565 0 0.0000 1.3160 0.3154
37  StrokeID: 2
38  Number of samples: 51
39  Color: 0 0 255
40  StartTime: 1591914265.726
41  108.6798 151.1357 0 0.3333 1.3823 0.3592
42  108.6798 151.1357 0 0.3333 1.3823 0.3592
43  ... 20 lines
44  108.0380 162.3647 17 0.7841 1.3823 0.3592
```

```
45  108.0380 163.0063 16 0.8175 1.3823 0.3592
46  108.0380 163.3271 17 0.8299 1.3823 0.3592
47  108.0380 163.9688 16 0.8542 1.3823 0.3592
48  108.0380 164.2896 16 0.8690 1.3823 0.3592
49  108.0380 164.9313 16 0.8960 1.3823 0.3592
50  108.0380 165.2521 16 0.9140 1.3823 0.3592
51  108.0380 165.8938 19 0.9382 1.3823 0.3592
52  108.0380 165.8938 13 0.9343 1.3823 0.3592
53  ... 10 lines
54  108.0380 169.4229 14 0.9612 1.3823 0.3592
55  108.0380 169.7437 16 0.9605 1.3823 0.3592
56  108.0380 169.7437 17 0.9858 1.3823 0.3592
57  108.0380 170.0645 18 0.9571 1.3823 0.3592
58  108.0380 170.0645 14 0.8336 1.3823 0.3592
59  108.0380 170.0645 16 0.5425 1.3823 0.3592
60  108.0380 169.7437 8 0.4966 1.3823 0.3592
61  108.0380 169.4229 0 0.0000 1.3823 0.3592
62  StrokeID: 3
63  Number of samples: 104
64  Color: 0 0 255
65  StartTime: 1591914268.547
66  ... two more strokes
67  </PenData>
68  </ClockSketch>
```

The application leverages the Alamofire framework to send the data file to the server. Alamofire is a Swift HTTP networking library that leverages the functionality innate to Swift, to create a simple way to send HTTP requests along with callbacks, instead of the delegate methods favored by Swift, such as NSURLConnection. Once the test is concluded, the data, stored as a SymbolTestFinal object, is first processed using newlines to separate and format the strokes one by one, with the numerical values truncated to four decimal places so that it is in a uniform and readable format. Then, the data is made into one continuous string. This string is then sent in the body of a POST request to the custom endpoint set up on the server for this app. It is sent as a CSV from the application, using a multipart form data format in the POST request, as that is how CSVs are sent over HTTP. Rather than sending the file all at once, the multipart data formatting sends it in many individual parts, providing a level of robustness that helps to ensure no data will be lost no matter how long the test is.

Overall, the application harnesses the power of data collection using the iPad and Apple Pencil, by accessing a variety of data values, such as the force of the pencil stroke, from sensors. The workflow is designed to prevent confusion and allows

subjects to go back if needed but not skip a necessary step.

4.4   Dynamic Testing

Dynamic testing is illustrated by the overlay that prevents non sequential test taking, as well as the voice-over that explains the mistake and reminds the subject to proceed to the next row of the exam. If the subject pauses for more than four seconds after writing in the last cell of a row, the application will blur the test completed already and point an arrow to the the first cell in the next row. It will also announce in English that the subject should continue filling out the next row of the test. Once the subject starts writing again, the arrow will disappear and the completed part of the exam will be viewable again.

Throughout the app, all buttons have both text and symbols. This is based on the recommendation of de Barros, et al. [1] which concluded that "older adults prefer to tap the icon even when both the icon and text work as a button." Additionally, each page has a back button, as the same user studies led them to "believe that the back button is important as a "fail safe" mechanism, that older adults rely on when they do not know how to solve a given problem." Additionally, they found that only "after being taught how to do it, older adults [were] able to perform a swipe gesture to navigate an interface." The apps are all free of pages that scroll using swipe gestures, so there is no advance instruction needed in that respect for older subjects to use the app.

4.5   Ecological Testing

The application records data about every button press. This is one of the ways to collect all the information we can about subject behavior. This data can be later analyzed along with the test results to inform diagnosis or explain symptoms. The data recorded is in the format seen in Figure 4-8.

Each button press is recorded as a ButtonTouch data object, which holds the x, y,

x	y	timestamp	altitude	azimuth	title
350.2696	244.7937	1596137114	0.6404	0.976	Tap to Proceed
384.5909	463.7019	1596137119	1.0547	0.4993	Save Subject Information
342.2506	109.9335	1596137122	0.9509	0.5907	SD Standard
630.6136	15.02	1596137123	0.955	0.5989	Next
35.2836	26.2051	1596137127	1.034	0.5886	Back
38.1704	13.1025	1596137129	0.9633	0.6451	Back
346.7413	179.6006	1596137133	0.7534	0.8344	SD Variant
42.661	13.1025	1596137134	0.5851	0.6299	Back
35.6043	30.6791	1596137134	0.5851	0.6299	Back

Figure 4-8: Button Timestamp Data Collected In Symbol Digit Application (x, y, timestamp, altitude, azimuth, title)

timestamp, altitude, and azimuth of the pencil touch, as well as the Apple UIButton object reference, so that the code can easily be augmented to collect other qualities of the button which was pressed. The x values and y values are measured from the top and left, respectively, of the iPad when in landscape orientation. This is a different coordinate system than that of the test, as this one covers the entire display screen. The timestamp is the UNIX timestamp. Currently, the data records only the title of the button, so that we know the order of the buttons that were clicked. Simultaneously, the applications maintains a running list of all the ButtonTouch objects created, which is called ButtonTimeFinal. In this way, all the button presses from one subject's use of the application are grouped together and can easily be saved for future analysis alongside the test results. The button timestamp data is added to the header of the data file, so that it is accessible with the rest of the test results.

Chapter 5

Contributions

With the TTS and Symbol Digit applications, we have a built a proof-of-concept for a testing ecosystem for cognitive disorders. This system aids with detection of cognitive disorders by creating another way to easily administer neurocognitive tests. It is a flexible platform that is designed to be easily augmented to study other facets of subject-test interaction as well as developed to reduce the administrative workload on clinicians.

Within the Symbol Digit application itself, it is possible to dynamically hide or call attention to certain parts of the exam in order to focus the subject's attention. To support this, the application is constructed with layered views, building from the base background view to the topmost draw view, with views in between aiding in test administration and ecological data collection. To create specialized features, the application supports adding a view between those layers to enact that action. Currently, layers such as the overlay layer help the subject focus on the correct cell to be writing in if they start filling out the test non-sequentially. In a similar vein, the blur and arrow layer helps guide the subject through the exam if they happen to pause at the end of a row. The collection of all button presses and their associated data using a custom class that records each instance provides a model for observing and documenting subject behavior. Furthermore, the applications record the anonymous subject information and test results. The data is automatically packaged up and sent to the server, where it is encrypted and stored for further analysis.

5.1 Future Work

5.1.1 Dynamic Displays

The test requires the administrator to guide the subject in filling out the first six cells of the dSDT. We believe that this introductory task could be made a part of the application. It could then be used to demonstrate the overlay and arrow functionality, in order to teach subjects how the exam reacts before they take the exam. We could purposely instruct them to fill out a cell, skip the next one, and then continue filling out the exam, which could trigger the overlay. Then, we could explain that they would need to fill the cell they skipped, and proceed in order, or the overlay would appear. Similarly, we could instruct the subjects to fill out the last cell in a row and pause for 5 seconds. The arrow would appear, and we could then instruct them to start writing in the next row. We could observe how the subjects learn in this stage as well and collect ecological data. The overlay concept could also be applied to the situation where the subject writes on top of the symbol, instead of in the blank cell underneath. In this case, the overlay could help draw their attention to the correct spot for their answer.

5.1.2 Ecological Data

Building on the implementations here, we believe there is more data that can be collected about subject behavior. For example, we can use the pen stroke data to recognize when a subject writes a non-sequential stroke that jumps rows, and collect aggregate statistics on that phenomenon. We could also observe both the timestamp and the frequency of non-sequential strokes. We could individualize the overlay that guides subjects when they pause at the end of a row by using their own pen stroke data. For the first 10 cells, we could measure the average time delay between the subject writing in each cell, and use that in real time to inform what length of time signifies a pause of confusion at the end of a row. In this case, the test would adapt to those taking it more slowly or more quickly. We believe there are many creative ways

to use dynamic displays to prevent unwanted subject behavior and varied methods to record subject behavior outside of what the test itself calls for.

Bibliography

[1] Ana Barros, Roxanne Leitão, and Jorge Ribeiro. Design and evaluation of a mobile user interface for older adults: Navigation, interaction and visual design recommendations. *Procedia Computer Science*, 27:369–378, 02 2014.

[2] Stina Björngrim, Wobbie van den Hurk, Moises Betancort, Alejandra Machado, and Maria Lindau. Comparing traditional and digitized cognitive tests used in standard clinical evaluation - a study of the digital application minnemera. *Frontiers in psychology*, 10:2327, 2019.

[3] Naomi Chaytor and Maureen Schmitter-Edgecombe. The ecological validity of neuropsychological tests: A review of the literature on everyday cognitive skills. *Neuropsychology review*, 13:181–97, 01 2004.

[4] Randall Davis, David Libon, Rhoda Au, David Pitman, and Dana L. Penney. Think: Inferring cognitive status from subtle behaviors. pages 2898–2905, Québec City, Québec, Canada, January 2014. 26th Innovative Applications of Artificial Intelligence Conference, IAAI 2014.

[5] Robert Fellows, Jessamyn Dahmen, Diane Cook, and Maureen Schmitter-Edgecombe. Multicomponent analysis of a digital trail making test. *The Clinical Neuropsychologist*, 31:1–14, 10 2016.

[6] M. D. Franzen and K. L. Wilhelm. Conceptual foundations of ecological validity in neuropsychology. 1996.

[7] Talia Herman, Nir Giladi, and Jeffrey Hausdorff. Properties of the 'timed up and go' test: More than meets the eye. *Gerontology*, 57:203–10, 04 2011.

[8] Elan D. Louis, Kristin J. Wendt, Steven M. Albert, Seth L. Pullman, Qiping Yu, and Howard Andrews. Validity of a Performance-Based Test of Function in Essential Tremor. *Archives of Neurology*, 56(7):841–846, 07 1999.

[9] Aaron Smith. Symbol digit modalities test: Manual. *Western Psychological Services, Los Angeles*, 1982.

[10] Martin Thirkettle, Jen Lewis, Darren Langdridge, and Graham Pike. Ou brainwave: A mobile app delivering a gamified battery of cognitive tests designed for repeated play (preprint). *JMIR Serious Games*, 6, 03 2018.

[11] Rob C. van Lummel, Stefan Walgaard, Andrea B. Maier, Erik Ainsworth, Peter J. Beek, and Jaap H. van Dieën. The instrumented sit-to-stand test (iSTS) has greater clinical relevance than the manually recorded sit-to-stand test in older adults. *PLOS ONE*, 11(7):1–16, 07 2016.