

**Navigating Neurological Realms,**

**Virtual Reality Diagnosis for Early Parkinson's Detection**

**Capstone Project – Phase B** -61999

**24-1-D-28**

**Writers:**

Michael Malka **311446231** [mickael266@gmail.com](mailto:mickael266@gmail.com)

Nir Asulin **305221483** [nir\_aso@hotmail.com](mailto:nir_aso@hotmail.com)

**Supervisors**:

Dr. Anat Dahan ( [anatdhn@braude.ac.il](mailto:anatdhn@braude.ac.il) )

Associate Professor Marcela Viviana Karpuj ( [mvkarpuj@braude.ac.il](mailto:mvkarpuj@braude.ac.il) **)**

**A logo with a picture of a car

Description automatically generated**

***Table of Contents***

1. [***Abstract 3***](#Abstract)
2. [***Introduction 4***](#Introduction)
   1. [***Parkinson Disease 4***](#ParkinsonsDisease)
   2. [***Asymmetry in PD Patients 4***](#AsymmetryinPDPatients)
   3. [***Our solution 5***](#Oursolution)
3. [***Development Process 5***](#DevelopmentProcess)
   1. [***General Description 5***](#GeneralDescription)

***3.1.1*** [***How Oculus Quest 2 Tracks the Movements 5***](#HowOculusQuest2TrackstheMov)

***3.1.2*** [***How Unity accesses the movement data recorded 5***](#HowUnityAccessesTheMovement)

* 1. [***Software Engineering Process 6***](#SoftwareEngineeringProcess)

***3.2.1*** [***Process 6***](#Process)

***3.2.2*** [***Product 7***](#Product)

***3.2.2.1*** [***CSV File Example 8***](#CSVFileExample)

***3.2.2.2*** [***Algorithm - Analyze Results 9***](#AlgorithmAnalyzeResults)

* 1. [***Diagrams 10***](#Diagrams)

***3.3.1*** [***Architecture diagram 10***](#ArchitectureDiagram)

***3.3.2*** [***Deployment Diagram 11***](#DeploymentDiagram)

***3.3.3*** [***Activity diagram 12***](#ActivityDiagram)

1. [***Evaluation 14***](#Evaluation)
   1. [***Testing Process 14***](#TestingProcess)
   2. [***Tests Table 14***](#TestsTable)
2. [***Challenges During the Development Phase 15***](#ChallengesDuringTheDevelopmentPhase)
   1. [***Hand Movement Tracking 15***](#HandMovementTracking)
   2. [***VR Immersion 15***](#VRImmersion)
   3. [***Synchronization issues 15***](#SynchronizationIssues)
   4. [***Data collection integration 15***](#DataCollectionIntegration)
   5. [***Colliders and scene interaction 15***](#CollidersAndSceneInteraction)
   6. [***Optimization for Oculus Quest 2 15***](#OptimizationDorOculusQuest2)
   7. [***Challenges during the project 16***](#ChallengesDuringTheProject)
3. [***Results and conclusions 16***](#ResultsAndConclusions)
   1. [***CSV results 16***](#CSVResults)
   2. [***Future development 17***](#FutureDevelopment)

***6.2.1*** [***Enhanced Simulation Environment 18***](#EnhancedSimulationEnvironment)

***6.2.2*** [***Transition to Full Hand-Tracking 18***](#TransitionToFullHandTracking)

***6.2.3*** [***Real-Time Feedback and Data Visualization 18***](#RealTimeFeedbackAndDataVisualization)

***6.2.4*** [***Integration with Machine Learning Algorithms 18***](#IntegrationWithMachineLearningAlgorithms)

***6.2.5*** [***Expanded Research Scope 18***](#ExpandedResearchScope)

***6.2.6*** [***Wearable Integration 18***](#WearableIntegration)

***6.2.7*** [***Improved Asymmetry Detection Algorithm 18***](#ImprovedAsymmetryDetectionAlgorithm)

***6.2.8*** [***Cloud-Based Data Storage and Analysis 18***](#CloudBasedDataStorageAndAnalysis)

1. [***Our Achievements 19***](#OurAchievements)
2. [***References 28***](#References)
3. ***Abstract***

Parkinson's disease (PD) is a neurodegenerative disorder that affects millions worldwide, characterized by symptoms such as tremors, bradykinesia, rigidity, and postural instability. Early diagnosis remains a significant challenge due to the subtlety and variability of early-stage symptoms, which can overlap with those of other neurodegenerative diseases. Accurate early detection is crucial for differentiating PD from other conditions to ensure appropriate care plans and timely intervention.

Leveraging advancements in virtual reality (VR) technology, this project proposes a novel approach for early PD detection, aiming to improve diagnostic accuracy and enable timely intervention. The proposed system, dubbed Navigating Neurological Realms, utilizes VR environments to detect subtle motor and cognitive impairments associated with early-stage Parkinson's. By identifying these specific symptoms, the system enhances diagnostic precision, facilitating the initiation of targeted therapies and potential preventative strategies. This project outlines the conceptual framework, technological components, and potential impact of the project, by offering a non-invasive, engaging, and highly sensitive diagnostic tool, Navigating Neurological Realms paves the way for more effective diagnostic methods in neurological healthcare. Early and accurate diagnosis can significantly influence the management and treatment of PD, enabling the use of emerging drugs and therapies that slow disease progression, and assist in planning for future care needs, thereby improving patient outcomes and quality of life.

Code Repository:<https://github.com/NanoShark/Navigating-Neurological-Realms>

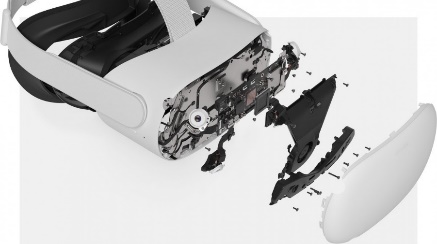
1. **Introduction** 
   1. **Parkinson’s Disease** (PD), named after James Parkinson, is a progressive, neurodegenerative disease. Neurodegenerative refers to the progressive breakdown of neurons in the brain and nervous system, leading to impairment of cognitive function, motor skills, or both. The brain cells in the brain's substantia nigra gradually deteriorate and lose their function over time. PD is characterised by motor symptoms of tremor, bradykinesia (slowness of movement and speed), and postural instability [[1]](#א). PD impacts people in different ways. Not everyone will experience all the symptoms of PD, even if people do, they won’t necessarily experience the symptoms in quite the same order or at the same intensity. Some people experience the changes from the disease's stages progressing over 20 years or more. Others find the disease progresses more quickly. PD is the second-most common neurodegenerative disorder (NDS) in the United States. Most people diagnosed with PD are 60 years or older, however, an estimated 5 to 10 percent of people with PD are diagnosed before the age of 50. Approximately 500,000 Americans are diagnosed with PD but given that many individuals go undiagnosed or are misdiagnosed the actual number is likely much higher. Given the progressive nature of the disabilities associated with PD, the disease affects thousands more wives, husbands, children, and other caregivers. [[4]](#ד)  
       Neuronal loss in the substantia nigra, [[6]](#ו) a critical brain region that produces dopamine, affects many systems of the central nervous system, including movement control, cognitive executive functions, and emotional limbic activity. This neuronal loss causes striatal dopamine deficiency and leads to the formation of intracellular inclusions containing aggregates of α-synuclein. a brain protein involved in nerve cell communication. When excessively accumulated, α-synuclein can form clumps which disrupt normal cellular function and contribute to the progressive loss of neurons. These are neuropathological hallmarks of Parkinson disease [[5]](#ה).
   2. **Asymmetry in PD Patients:** Motor asymmetry is a defining characteristic of Parkinson's disease, typically evident from the disease's onset. Patients often experience more severe symptoms on one side of the body, which can significantly impact their daily functioning and quality of life. This asymmetry is believed to result from uneven degeneration of dopaminergic neurons in the substantia nigra, a key brain area affected by PD. The reasons behind this uneven neuronal loss remain a subject of investigation, with hypotheses including genetic factors, environmental exposures, and variations in neuronal vulnerability. Understanding and monitoring motor asymmetry in PD is crucial for several reasons. Clinically, it assists in the accurate diagnosis and differentiation of PD from other neurological conditions that might present with more symmetrical motor impairments. Furthermore, the degree and progression of asymmetry can inform the tailoring of treatment strategies. Motor symptoms in PD patients remain asymmetric over time, without a significant shift toward symmetry, which highlights the persistent and unilateral nature of PD pathology. This supports the use of motor asymmetry as a diagnostic and prognostic marker in PD [[3]](#ג).
   3. **Our solution** proposes a virtual reality (VR)-based diagnostic tool designed to detect motor asymmetry in early-stage PD patients. The tool utilizes a virtual environment to simulate swimming motions, where hand movements are closely monitored to capture differences between the left and right hands. This method emphasizes detecting subtle motor impairments, which can be crucial indicators of early-stage PD.
2. **Development Process**
   1. **General Description**

The target users for this project are individuals diagnosed with PD or those suspected to be in its early stages. The primary objective was to develop a system that supports early diagnosis by offering a comfortable and patient-friendly testing method. Our approach leverages cutting-edge VR technology to enhance the diagnostic experience, ensuring it is both effective and accessible while utilizing the latest advancements in VR.

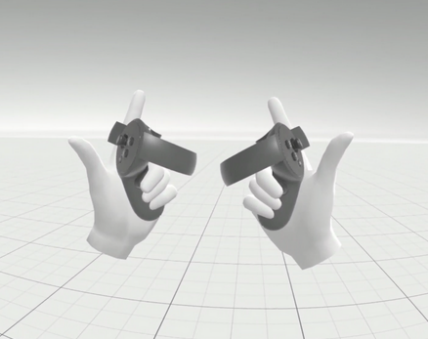
We decided to use the Unity engine, coupled with the Oculus Quest 2 VR headset, to deliver a detailed analysis of movement in a simulated swimming environment.  
Unity is popular for its user-friendly interface and ability to support a wide range of media assets. Unity also provides extensive tools for VR development and extensive support for Oculus devices and allows to integrate a sophisticated motion tracking system that records participants' movements accurately.

* + 1. **How Oculus Quest 2 Tracks the Movements**

The Oculus Quest 2 uses a combination of sensors and advanced algorithms for accurate motion tracking. At its core is an Inertial Measurement Unit (IMU) that includes accelerometers and gyroscopes, providing data on acceleration and rotational movement. The headset's inside-out tracking system, which relies on four ultra-wide-angle cameras, eliminates the need for external sensors by capturing the surrounding environment and tracking infrared (IR) LED patterns on the controllers and headset.

Computer vision algorithms and Simultaneous Localization and Mapping (SLAM) scan and map the environment, enabling accurate positional tracking. The Quest 2 also features a depth sensor for more precise environmental mapping. Hand tracking is achieved through cameras and machine learning, allowing users to interact without controllers. A proximity sensor adjusts the display when the headset is worn or removed. This combination of technologies delivers a highly immersive and responsive virtual reality experience.

* + 1. **How Unity accesses the movement data recorded by the Oculus Quest 2** Unity accesses Oculus Quest 2 movement data through the Oculus Integration package and the XR Input Subsystem. The Oculus Integration package provides tools like **OVRCameraRig** and **OVRInput** to retrieve position, rotation, and tracking data for the headset and controllers. It also handles differences between Oculus headsets, simplifying development.

Unity’s XR Input Subsystem offers a unified interface for accessing movement data from various XR devices, including the Quest 2, via consistent APIs. Hand tracking data can also be accessed through both the Oculus Integration package and XR Input Subsystem. Additionally, environmental data such as occlusion and spatial anchoring can be used to enhance the immersive experience.   
Developers can write custom scripts in Unity to process Quest 2 movement, hand tracking, and environmental data, enabling interactive VR experiences[[7]](#ז).

* 1. **Software Engineering Process** 
     1. **Process**

We worked in an Agile development process, dividing the project into several main sprints. Each sprint focused on specific aspects of the system:

* First Sprint: Concentrated on the user interface (UI) and enhancing the immersion of the virtual environment, improving the experience with new hand models and sound effects.
* Second Sprint: Focused on developing the core hand-tracking functionality and ensuring basic interaction with the virtual environment.
* Third Sprint: Extended the functionality to include automated data collection, refining how the movement data was recorded and structured.

This software development process begins with an extensive search for an asset that could deliver a highly realistic diving experience. After evaluating various options, we selected the **DeepUnderwater** asset from the Unity asset store and integrated it into our Unity workspace.  
Next, we incorporated a VR player using the **OpenXR** framework to enable virtual reality functionality. We then enhanced the experience by adding virtual hands to the VR player. However, the default “glove” style hands provided insufficient realism, as they lacked individual finger control. Finger movement is affective for an immersive user experience. To resolve this, we downloaded a more realistic 3D hand model from an external website, allowing for precise finger manipulation.

We implemented a custom swimming script, adjusting the gravity settings to simulate an underwater environment. **Colliders (**Components in Unity that define the shape of an object for physical interactions. They ensure that objects don't pass through each other by detecting collisions between game objects**)** were applied to all elements in the scene, including the terrain (A Unity feature used to create large, detailed outdoor environments, such as landscapes with hills, mountains, or flat surfaces. It's often used for creating the ground or the floor of a scene), to prevent the player from passing through objects.  
For an added sense of realism, we incorporated sound effects. Upon starting the simulation, users hear an authentic water entry sound. Throughout the experience, ambient underwater noises and the sound of water being cut by the player’s rowing motion enhance immersion.

The swimming simulation is capped at two minutes, so we added an in-game timer visible regardless of where the user looks during the simulation. The timer begins when the user starts swimming, and the simulation automatically ends and locks after the two-minute period.

To track user movement, we developed a script to record hand movements along the x, y, and z axes, hand rotation, velocity, and the number of instances where the hands are within 10 centimeters of each other. This script focuses solely on recording swimming motions to minimize unnecessary data collection. The recorded data is exported to a CSV file with the naming convention: username\_yyyymmdd\_hhmm. Here, the **username** is entered in the main menu, followed by the **year(yyyy)**, **month(mm)**, **day(dd)**, and **time (hour = hh , minutes = dd)** the simulation took place.  
Lastly, we added a login screen with a main menu that allows the user to input their username, start the simulation, and access instructions on how to perform the simulation correctly.

* + 1. **Product**

Our project delivers a VR environment, aimed to find asymmetric movements through simulated swimming activities. The system tracks the user's movements during the exercise and records all the movements of the patient into a CSV file, which provides a detailed record of the hand movements in the exercise, after the CSV file will be transferred into an algorithm that will diagnose the data.

The VR environment is built using the Unity game engine. To track the user's hand movements accurately, we utilize the Oculus Hand Tracking feature, which allows for precise tracking of hand positions and gestures. The Oculus Hand Tracking system uses advanced computer vision algorithms and depth sensors to detect and track the user's hands in real-time, providing a natural and intuitive way to interact with the virtual environment.

As the user performs the simulated swimming exercises within the VR environment, the Oculus Hand Tracking system captures the position, rotation, and velocity data of their hands. This data is then processed and recorded into a CSV file, providing a comprehensive record of the exercise session. The CSV file includes measurements as a comprehensive set of data. The measurements should include the x, y, z coordinates of each hand (left and right) to track their positions and trajectories during the swimming motions. Additionally, we should record the x\_rotation, y\_rotation, and z\_rotation of each hand to capture the orientation and twist of the hands during the swimming strokes. Optional hand animation data, such as animation curves or keyframes for hand and finger movements, can provide more detailed information about specific hand and finger articulations if needed.

Timing data, including timestamps for each recorded data point, is essential to analyze the timing and synchronization of hand movements between the left and right sides. Event data, such as flags or markers for specific events during the swimming exercise (e.g. stroke start, stroke end, stroke type), can help segment the data and analyze specific phases of the swimming motion. Optionally, we may also consider capturing position and rotation data for other body parts (e.g. arms, shoulders, torso) to provide context and help analyze the overall body coordination during the swimming exercise.

By capturing these comprehensive measurements, we will have detailed information about the hand movements, including their positions, rotations, timings, velocity and relevant events. This data can then be structured into a CSV file, with columns representing each measurement and rows representing the data points at each timestamp or frame of the exercise. This structured data format will make it easier to feed into the algorithm for analysis and diagnosis, allowing us to identify any asymmetries or discrepancies between the left- and right-hand movements, which could potentially be indicative of early-stage PD.

* + - 1. **CSV File Example**

This CSV example contain data from the hand movement analysis captured during the VR swimming simulation.

* Timestamp: Records the time at which each data point was captured.
* LeftPosX, LeftPosY, LeftPosZ: The X, Y, and Z coordinates of the left hand's position in the virtual environment at each timestamp.
* LeftRotX, LeftRotY, LeftRotZ, LeftRotW: The rotation of the left hand, likely stored as a quaternion (X, Y, Z, W components), to capture the orientation in 3D space.
* RightPosX, RightPosY, RightPosZ: The X, Y, and Z coordinates of the right hand's position.
* RightRotX, RightRotY, RightRotZ, RightRotW: Similar to the left hand, these represent the rotation quaternion values for the right hand.
* **A white background with many squares

  Description automatically generated with medium confidence**HandLossCount: This may represent how many times the hands were close to each other (10 cm apart ).
  + - 1. **Algorithm - Analyze Results**

To determine if a patient exhibits asymmetry in their hand movements, we analyze the differences between the left- and right-hand measurements for various parameters such as positions, rotations, and timings, as well as the timing differences between the start and end stroke events. The analysis involves calculating the difference between the left- and right-hand measurements for each data point using the Index of Asymmetry (IA) formula:

where L and R are the measurements of the left and right hands, respectively. This calculation is applied to each relevant measurement, such as positions (x, y, z), rotations (x\_rotation, y\_rotation, z\_rotation), for each timestamp.  
Next, we calculate the timing difference between the start and end stroke events for the left and right hands. This is done by finding the timestamps associated with these events for each hand and calculating the duration of the stroke for each side using the formula:

.

Then we calculate the difference in stroke duration between the left and right hands using the IA formula:  
.

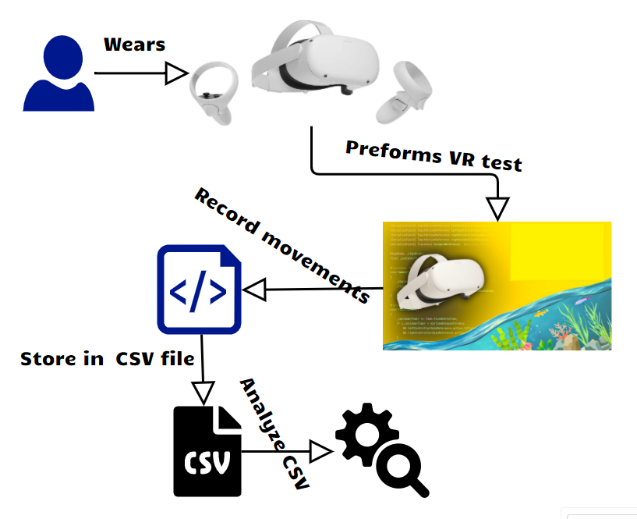
We then calculate the mean Index of Asymmetry across all data points for each measurement type, including the stroke duration difference, by using the formula:

, which gives the average asymmetry between the left and right hand measurements for each measurement type (position, rotation, timing, and stroke duration).  
To determine a threshold value for asymmetry, we use the formula:

(for healthy individuals), where α is a constant factor that defines the acceptable level of asymmetry for healthy individuals. This threshold can be determined based on a dataset of healthy individuals or empirical data. We compare the mean IA for the patient with the determined threshold for each measurement type.

, then the patient is considered to have asymmetry in that measurement.  
Finally, we combine the results for different measurements. If the patient is flagged as having asymmetry in multiple measurements (positions, rotations, timings, and stroke duration), it increases the likelihood of asymmetric hand movements, which could be indicative of Parkinson's disease. By incorporating the stroke duration difference calculation and including it in the analysis, we can potentially identify asymmetries in the timing and coordination of hand movements during specific strokes, which could be an important indicator of Parkinson's disease [[2]](#ב).

* 1. **Diagrams**
     1. **Architecture Diagram**

****

* A user wears the VR headset and controllers.
* They perform a VR test, engaging in a simulated swimming activity.
* The system records hand movements during the test.
* These movements are stored in a CSV file.
* The CSV data is exported to computer and analyzed to identify asymmetry movement patterns for diagnosing early-stage Parkinson's disease.
  + 1. **Deployment Diagram**

**Development Environment:**

* Unity Editor: Where the project is developed using C# scripts.
* Oculus Integration SDK: Enables VR features like hand tracking.
* Build Scripts: Create the APK file to deploy on the Oculus Quest 2.
* Git Version Control: Tracks code changes, managed via GitHub Desktop.

**Android Build System:**

* Builds the project into an APK file, necessary for Oculus Quest 2, which runs on an Android-based OS.

**Oculus Quest 2:**

* Runs the Unity Application with features like hand tracking and data recording.
* CSV Data Recorder logs hand movement data into CSV files for analysis.
* GameTimer and Main Scene Timer manage time tracking in the simulation.

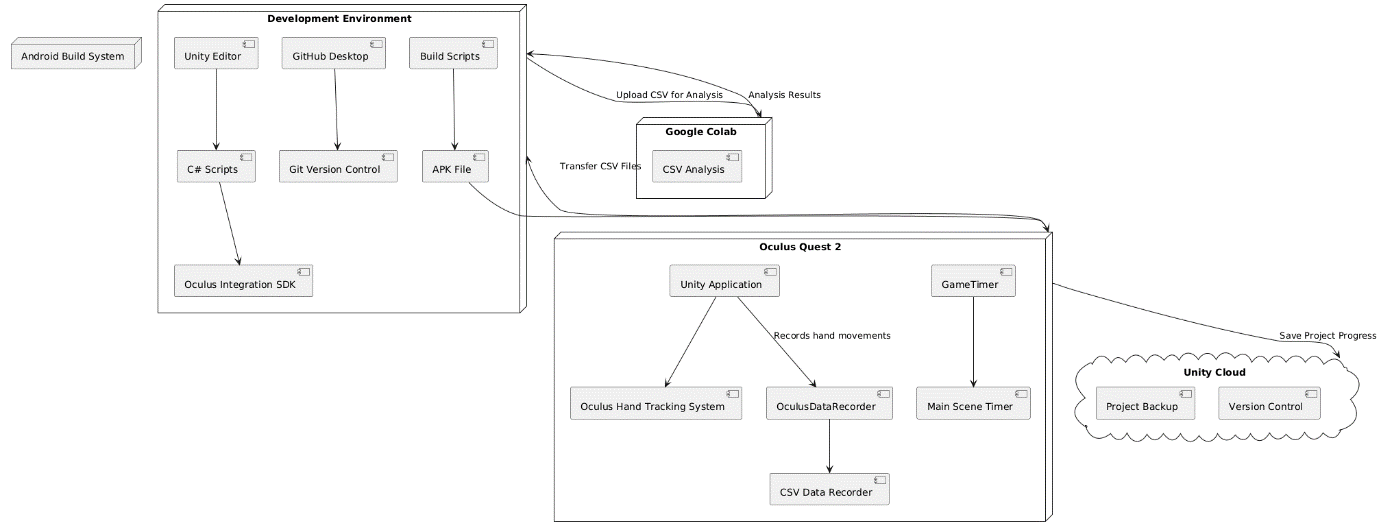
**Google Colab:**

* Analyzes the CSV files collected from Oculus using scripts, returning results to the development environment.

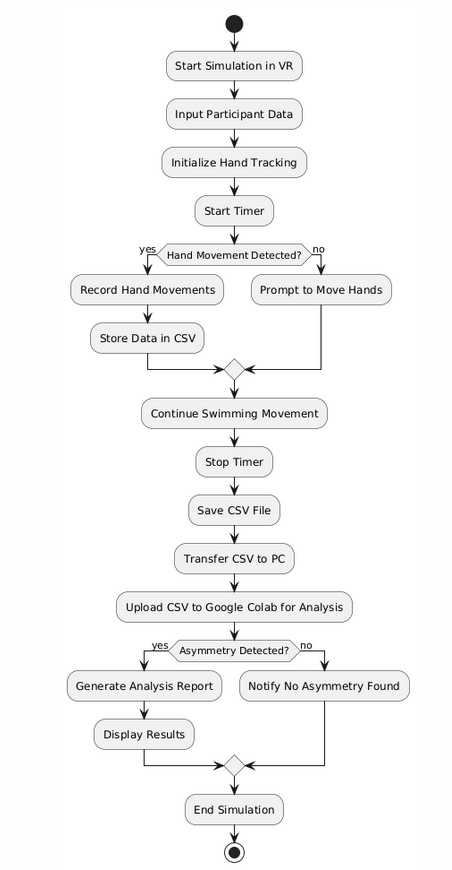
**Unity Cloud:**

* Provides Project Backup and Version Control, storing project data and progress.

**Data Flow:**

Oculus Quest 2 sends CSV data to the Development Environment for transfer to Google Colab for analysis, and project progress is backed up to Unity Cloud.

* + 1. **Activity Diagram**

****

**Start Simulation**:  
The process begins when the participant starts the VR simulation. The simulation focuses on detecting early signs of Parkinson's disease by tracking hand movements.

**Input Participant Data**:  
The participant’s information, such as their name or ID, is entered into the system to track who the data belongs to. This step ensures personalized data collection and accuracy.

**Initialize Hand Tracking**:  
The Oculus Quest 2’s hand tracking system is initialized, preparing to track hand movements throughout the simulation. This involves activating the system’s sensors and algorithms for movement detection.

**Start Timer**  
Once the hand tracking is set, a timer is started to measure the duration of the simulation. Timing is crucial for assessing how quickly and accurately the participant completes tasks or performs movements.

**Detect Hand Movement**:  
The system continuously monitors hand movements through the VR interface. If any movement is detected, the process proceeds to the next step. If no movement is detected, the system may loop back to reinitialize the hand tracking.

**Record Movement Data**:  
As the participant moves their hands, the OculusDataRecorder records all the motion data, including positions, rotations, and timings, into a CSV file. This raw data will later be used for analysis.

**End Timer**:  
When the simulation concludes (or after a set time limit), the timer is stopped. This ensures that the movement data is properly time-stamped and measured.

**Save CSV File**:  
The recorded hand movement data is saved in a CSV file format on the Oculus Quest 2. This file contains all necessary details for analyzing movement patterns.

**Transfer CSV to PC**:  
The CSV file is transferred from the Oculus Quest 2 to a computer for detailed analysis, typically done using Google Colab.

**Analyze Data in Google Colab**:  
The hand movement data is analyzed in Google Colab, where algorithms calculate the Index of Asymmetry (IA) and other metrics to detect potential Parkinson's disease indicators.

**Asymmetry Found?**:  
After analysis, the system checks if asymmetry in hand movements is detected, which can indicate early-stage Parkinson's.

**Report Asymmetry**:  
If asymmetry is found, a report is generated, notifying the medical team or researchers about the irregularity in the participant's hand movements.

**No Asymmetry Detected**:  
If no asymmetry is found, the system confirms that the participant's hand movements were symmetrical, and no early signs of Parkinson’s were detected.

**End Simulation**:  
After reporting the results (whether asymmetry was found or not), the simulation ends, and the participant can exit the VR environment.

1. **Evaluation**
   1. **Testing Process**

We needed to build an orderly testing plan that encompasses all the functional and non-functional operations in the system.

The testing program is designed to detect weaknesses, and problematic processes and is designed to verify the accuracy of the various actions contained in the system.

As mentioned in the article, the VR works with several sensors, and we need to make sure that they work as required and that the information extracted from them enables us to perform the appropriate analyses to obtain the desired results.

We built the test plan with the help of the Activity diagram which contains all the main scenarios in each complex process in the system which we must verify are correct.

* 1. **Tests Table**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Test case | Expected results | Status |
| 1 | VR environment launches | The VR environment should open successfully and be visible through the VR headset. | Pass |
| 2 | User input (name) | User enters their name, and the “Start” button is enabled. | Pass |
| 3 | Start simulation | Upon clicking “Start,” the swimming simulation begins, and the timer shows 02:00 minutes, but it is not running. | Pass |
| 4 | Display timer during simulation | The timer is visible and counts down from 2 minutes during the simulation and start only when the first swimming motion is performed. | Pass |
| 5 | Hand tracking during swimming | Hand movements are tracked in real-time without delays. | Pass |
| 6 | Simulation time limit (02:00 minutes) | The simulation ends automatically after 02:00 minutes, and swimming is no longer possible. | Pass |
| 7 | System performance | VR system operates smoothly with no frame drops or glitches. | Pass |
| 8 | User feedback (sound effects) | Users hear appropriate sound effects when interacting with the simulation (e.g., water entry and swimming motions). | Pass |
| 9 | Save data to CSV | The movement data (x y z positions, rotations etc) is successfully saved in a CSV file. | Pass |
| 10 | CSV file import | The saved CSV file can be successfully imported and used for analysis. | Pass |

1. **Challenges During the Development Phase**

The development of the NNR simulation was guided by the need for accurate and reliable tracking of user movements in the virtual environment. We focused heavily on designing an intuitive interface and ensuring that the simulation functioned seamlessly. Several challenges emerged during the process, but we managed to overcome them and deliver a working prototype.

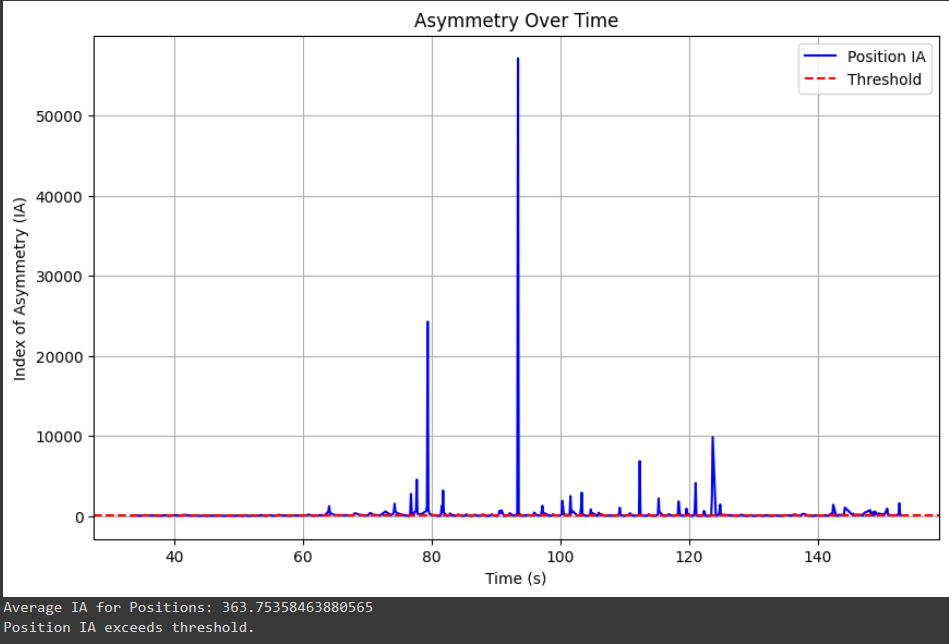
* + 1. Hand Movement Tracking: One of the primary challenges was ensuring that the system accurately tracked the movements of both hands. In the initial phases, we struggled with detecting the subtle differences between left- and right-hand movements. We conducted extensive testing, making small adjustments to the tracking system, which finally led to achieving the desired precision.
    2. VR Immersion: Achieving the right level of immersion was key to making the user experience engaging and comfortable. Initially, the default hand models provided by Unity felt too rigid and unrealistic. After much trial and error, we replaced the default models with more natural hand models and improved the overall experience by integrating subtle sound effects that simulate underwater movement, which added a layer of realism.
    3. Synchronization issues: During our testing phase, we noticed discrepancies in how the system handled hand movement synchronization. Some users experienced slight lag between their physical movements and the VR hand model’s response, which impacted data accuracy. After optimizing the Oculus Quest 2's hand-tracking capabilities, we managed to minimize this lag and ensure smooth synchronization across different testing environments.
    4. Data collection integration: We also had to ensure that the data collected during the simulation was captured correctly in the CSV files. Initially, we faced challenges with formatting the data and ensuring that it was being saved at the right moments. Through multiple iterations and debugging, we were able to fine-tune the system to capture hand movements with precise timestamps, positions, velocity and rotations.
    5. Colliders and scene interaction: Another challenge was ensuring that the users didn’t pass through objects in the environment while swimming. To resolve this, we implemented colliders on all scene elements, preventing users from accidentally passing through the virtual terrain or obstacles. This added both realism and structure to the interaction. In addition, we added boundaries to the terrain so that the user does not leave the borders of the terrain and go to the void. In order for the user not to be able to go out of the established limits or pass through them, we defined it as colliders.
    6. Optimization for Oculus Quest 2: To optimize the NNR simulation for the Oculus Quest 2, several key changes were made to ensure smooth performance without compromising functionality. First, complex materials were simplified by replacing high-polygon textures with basic shaders, significantly reducing the rendering load. Additionally, overall graphics quality was lowered, adjusting texture resolutions and visual effects to suit the Quest 2’s hardware limitations while maintaining the immersive underwater experience. Unused components, such as obsolete scripts and assets, were identified and removed from the Unity project to reduce memory usage and improve load times. Scene optimization techniques like frustum culling were also implemented, rendering only the objects within the user’s view to conserve processing power. These combined optimizations allowed the system to run efficiently on the Quest 2, providing a smooth, responsive, and portable VR diagnostic tool.
    7. During the project, amidst the deteriorating security situation in the country, one of the team members, Nir, was deployed for reserve duty in Gaza with no defined return date. This created significant challenges in scheduling joint work sessions and coordinating individual efforts throughout the project’s development. Additionally, the heavy mental strain and drop in national and personal morale, along with environmental pressures stemming from the overall situation, likely impacted both motivation and performance levels.

1. **Results and conclusions**

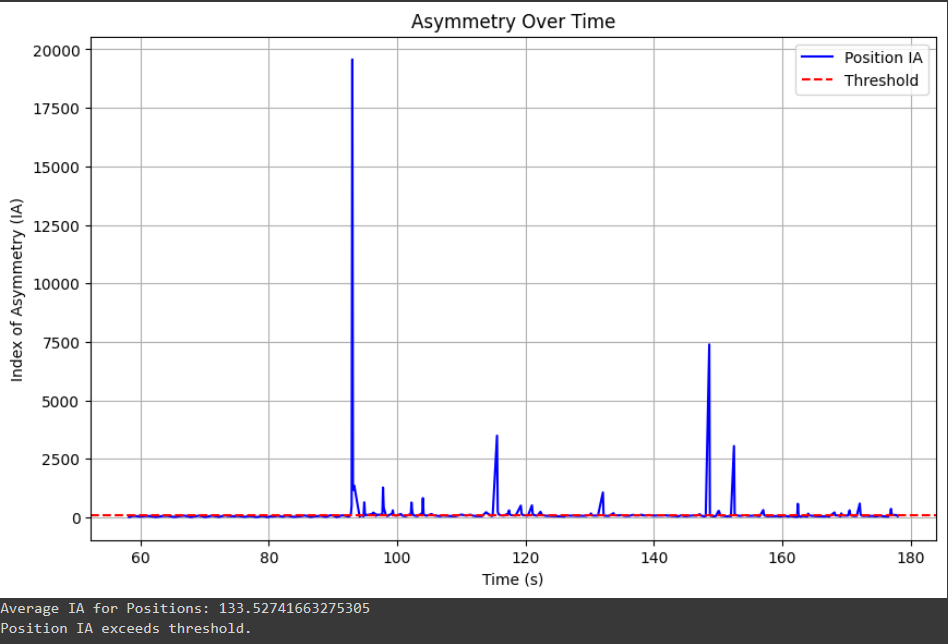
Despite the challenges, we managed to develop a functional and user-friendly diagnostic tool. Once testing was completed, we were able to confirm that our VR environment accurately detected movement asymmetries, which is critical for early detection of Parkinson's disease. This success gave us the confidence to explore further applications of this system and consider adapting it to other neurological disorders in future developments.

* **IA Calculation:** The IA is calculated for each time point by comparing the positions of the left and right hands in 3D space (X, Y, Z dimensions). The IA for each axis is averaged to obtain a single value representing the overall asymmetry for that point in time.
* **Threshold Comparison:** The red dashed line in the plot represents the mean IA value from healthy subjects, adjusted with a safety margin. The subject's IA values consistently exceed this threshold, reinforcing the finding of significant hand movement asymmetry.
* **Trend Over Time:** The IA remains elevated throughout the simulation, indicating that the asymmetry is persistent rather than momentary, which can be a critical factor when analyzing motor impairment.
* **Diagnostic Insight:** The sustained high IA values suggest a deviation from normal hand movement coordination, which is an important feature in diagnosing motor-related diseases. The test confirms that the subject's movements are significantly more asymmetric compared to healthy participants.
  + 1. **CSV results**We conducted four tests: two healthy people conducted the tests, and we also created two that simulated asymmetry. In the first two tests, where we performed regular swimming motions, the results were calculated into an initial threshold. In the asymmetry simulating tests we conducted asymmetry swimming on purpose. As observed, the threshold in both was exceeded, indicating a significant asymmetry issue, suggesting that the patients may be at risk for PD.

**First Asymmetry** **Test**



**Second Asymmetry Test**

****

The **Average IA for Positions** is the mean asymmetry between the left- and right-hand positions (X, Y, Z) over time.

From the generated charts, we observe the asymmetry values over time for the subject’s hand positions. The Index of Asymmetry (IA) for both hands are plotted across all time points of the simulation. The blue line represents the calculated IA values for the subject, while the red dashed line indicates the threshold derived from healthy participants' simulations.

It is clear that the average IA values for the subject's hand positions throughout the simulation are significantly higher than the healthy threshold. This suggests that the subject exhibits a notable level of asymmetry in hand movements compared to healthy individuals, which may be indicative of a motor function disorder, such as early-stage Parkinson's disease.

* 1. **Future development**

Despite the challenging security situation, we successfully achieved our objectives. However, with additional time and resources, several key improvements and expansions could further elevate the project.

* + 1. **Enhanced Simulation Environment**: We plan to introduce dynamic sea animals, varying water conditions (such as waves and currents), and interactive tasks like object retrieval or obstacle avoidance. These elements would provide a more immersive and realistic swimming experience, improving user engagement and ensuring more accurate data collection by replicating real-world conditions.
    2. **Transition to Full Hand-Tracking**: Moving away from controllers to full hand-tracking would allow us to capture finer details, including finger movements. This would enable more precise measurements of hand movement asymmetry and offer deeper insights into motor control issues in early Parkinson’s patients. Incorporating gestures like finger pinching or specific hand movements would also broaden the diagnostic range.
    3. **Real-Time Feedback and Data Visualization**: Implementing real-time feedback mechanisms, such as visual overlays or haptic responses, could provide immediate insights into the user’s performance. Additionally, real-time data visualization could allow clinicians to observe the movement asymmetry during the simulation, making it easier to detect abnormalities at earlier stages.
    4. **Integration with Machine Learning Algorithms**: Incorporating machine learning models to analyze hand movement patterns could enhance the system’s ability to predict the progression of Parkinson’s disease. These models would continuously learn from new patient data, improving detection accuracy and offering more personalized diagnostics.
    5. **Expanded Research Scope**: Conducting experiments with a larger, more diverse sample of Parkinson’s patients at different stages of the disease (including early, mid, and advanced stages) would generate more comprehensive data. We would also explore using the system to study other neurodegenerative disorders that affect motor functions, such as Huntington’s disease or multiple sclerosis.
    6. **Wearable Integration**: Introducing external sensors, such as wearable devices or motion trackers, could allow for the collection of additional physiological data, including muscle tension or heart rate variability, which might further correlate with motor function deterioration.
    7. **Improved Asymmetry Detection Algorithm**: With additional patient data and technological upgrades, we aim to refine the asymmetry detection algorithm to improve accuracy. The algorithm would evolve to identify more subtle signs of movement discrepancies, making it an even more reliable tool for early diagnosis.
    8. **Cloud-Based Data Storage and Analysis**: Transitioning to a cloud-based storage system would allow for secure, remote access to patient data, enabling clinicians to track progress over time and collaborate with other researchers. The cloud integration would also support more sophisticated analysis by leveraging larger datasets across multiple research centers.

1. **Our Achievements**

In our project, we have successfully integrated advanced VR technology to create a dynamic environment for the early detection of PD through the analysis of movement asymmetry. The key achievements of our initiative include:

* + Development of a customized VR diagnostic tool: We have developed a VR system that simulates swimming actions to detect subtle motor asymmetries indicative of early PD. This tool uses the Oculus Quest 2 and Unity game engine to create a responsive and immersive diagnostic environment.
  + Movement analysis algorithm: At the heart of our VR tool is an algorithm that tracks and analyzes hand movements in real time. It calculates the Index of Asymmetry (IA) from the movement data, giving a clear measure of motor asymmetry.
  + Automated data capture and analysis: Our system automatically captures detailed movement data, including position, orientation, velocity and timing of each hand movement during the VR simulation. This data is processed to detect inconsistencies between the left- and right-hand movements, which are critical markers for PD.
  + User-centric design: The VR environment is designed to be intuitive and user-friendly, allowing users to interact naturally with the virtual environment. This focus on user experience is essential for ensuring accurate data collection and patient comfort during the diagnostic process.
  + Empowering early diagnosis: The most significant achievement is the potential impact of our tool in the early diagnosis of PD. By enabling early and accurate detection, our project opens new pathways for timely intervention and treatment, significantly altering the patient care journey for individuals at risk of PD.  
    Our project offers new hope for those seeking early detection methods for Parkinson's Disease.

**User Manual**

**General description**

The **NNR** simulation is a VR-based diagnostic tool designed to detect early-stage PD through the analysis of hand movement asymmetry. By leveraging VR technology, the system immerses users in a virtual environment where they perform simulated swimming motions, allowing for real-time tracking and analysis of their hand movements. This approach offers a non-invasive, engaging method for early detection of PD, which is crucial for timely intervention and treatment.

The system is built using the **Oculus Quest 2** and the **Unity** game engine, enabling accurate tracking of hand movements without requiring additional sensors or equipment. It is designed to be user-friendly, allowing individuals of varying technical proficiency to perform the diagnostic tasks with ease. The VR environment is intuitive and includes visual and auditory feedback to guide users through the simulation.

The **NNR** simulation is intended to be used in clinical or research settings to assist in the early detection of PD. Its primary user base includes individuals suspected of early-stage PD, healthcare professionals, and researchers in the neurological field. The system runs on **Oculus Quest 2** hardware, requiring no external cameras or complex setups, making it a portable and scalable solution for neurological diagnostics.

By integrating advanced movement tracking and data analysis capabilities, this system represents a significant step forward in using VR for medical diagnostics, providing a new tool for detecting the subtle motor impairments associated with Parkinson’s disease.

**System Acknowledgement**

1. The user should be standing in an open space that allows the user to move freely with open arms.
2. The assessor gives the user the controllers.
3. First, secure the ropes around the user’s hands, then have them hold the controllers.
4. Show the user how to preform swimming rowing motion.
5. Only after these steps, place the headset on the subject’s head and help them adjust it to the right size.

**User’s Guide**

Opening the simulation, the user is greeted with a screen featuring text box into which they would enter their name and two main selection buttons: “Start” and “Instructions”, accompanied by the sound of submerging underwater. The “Start” button is disabled until the user inserts their name. At this point, the user enters the simulator phase, where they are introduced to the diagnostic game and learn how to interact during the simulation. Red laser beams, which extend from the user's virtual hands, allow for button selection by pointing at and interacting with the screen elements. This phase helps familiarize the user with the controls and flow of the simulation.

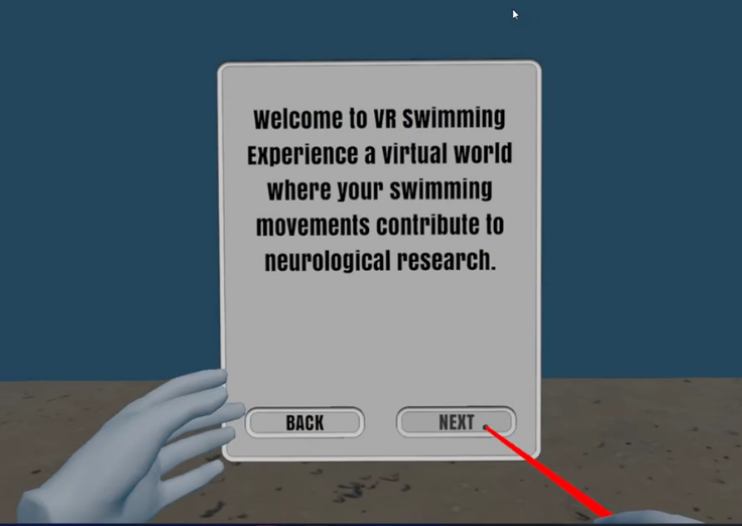
**Opening Window**

**A screenshot of a computer screen

Description automatically generated**

This screen displays opening words welcoming the future user to the simulation. On top of the screen there are two navigation buttons, one "Back" to return to the main screen, and the other "Next" to go to the instructions regarding the use of the simulation.

**Instructions Window 1**

****

This screen provides essential instructions for performing the swimming motion that is central to the diagnostic task.

**Instructions Window 2**

**A screen shot of a video game

Description automatically generated**

When pressing the text field, a virtual keyboard will appear.

**Keyboard Opening**

****

The start button will be unable when the user will enter his name.

**Username input**

****

This screen is where the actual diagnostic task takes place. It's important to maintain focus and perform the swimming motion as accurately as possible to ensure that the collected data is valid for early Parkinson's detection analysis. The clock starts at 2 minutes and goes down.  
it won’t activate until the user starts the actual swimming motion and it would follow the swimming hands all through the simulation.

**Simulation starts**

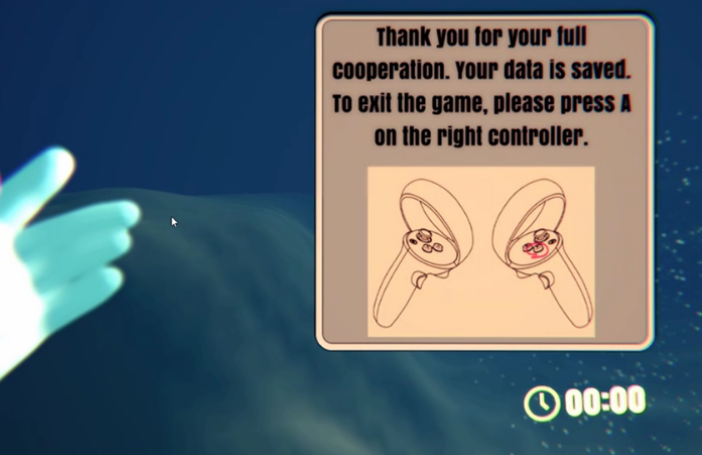
A screenshot of a video game

Description automatically generated

The timer will start count down when the user starts his swimming motion.  
**Timer on**

****

When the timer hit 00:00 the simulation stops and the user can’t swim anymore, and he have the option to press A button on the right controller to exit the simulation **End simulation**

****

**Maintenance Guide**

**Unity Editor Requirements**

* Unity Version: Ensure you have Unity installed (version 2020.3 or later recommended).
* GitHub Repository: Download the project ZIP file from the official NNR GitHub Repository. Extract the project files to your desired directory.
* Opening the Project: Use Unity Hub to open the extracted project. Navigate to the project directory and select it from Unity Hub.

A screenshot of a computer program

Description automatically generated

**Unity Project Setup**

**XR Plugin Management**

Navigate to Edit > Project Settings > XR Plugin Management.

Install the XR Plugin Manager package if it's not already installed.

Enable Oculus in both the Android and PC, Mac & Linux tabs. This integrates Oculus-specific VR functionality required for the NNR project.

**Configuring Player Settings**

Go to Edit > Project Settings > Player and configure the following:

XR Settings (Android):

Ensure Virtual Reality Supported is checked.

Target API Level: Set the Target API Level to 29 or above. This ensures compatibility with Oculus Quest 2.

Minimum API Level: Set the minimum API to Android 10.0 or higher to prevent compatibility issues.

**Switch Platform to Android**

Navigate to File > Build Settings.

In the Platforms section, select Android, then click on Switch Platform to set Android as the target build platform for the Oculus Quest 2.

Verify that the Run Device option lists your Oculus device when connected.

**Oculus Quest 2 Setup**

**Developer Mode**

Enable Developer Mode on your Oculus Quest 2 device:

Open the Oculus app on your phone.

Navigate to Settings > Developer Mode and enable it.

**Connecting Oculus Quest 2 to Unity**

**Connect your Oculus Quest 2 to your PC using a USB cable.**

In Unity, your Oculus Quest 2 should now appear in File > Build Settings > Run Device. Ensure it's selected to build and run your project directly onto the headset.

**Input Setup**

**Input Mapping for Oculus Controllers**

Go to Edit > Project Settings > Input.

The Oculus Integration package provides pre-configured mappings for Oculus controllers:

Ensure these mappings are applied to interact with the project correctly.

Modify specific input settings as needed for the NNR project, such as setting thumb triggers for hand movements and controller-based actions.

**Additional Settings for NNR**

**Hand Movement Tracking**

Ensure that hand movement tracking is set up to capture asymmetry in the user's movements for the swimming simulations.

**Data Capture**

Confirm that the scripts for recording CSV data (positions, rotations, and timing) are functioning as expected, especially during simulations that track hand movement asymmetry.

**Profiling and Testing**

Use Unity Profiler for monitoring performance while running the app on your Oculus device. This will help ensure that the simulations run smoothly and efficiently.

**Maintenance Guide Asymmetry Analysis Script**

This guide explains how to use and maintain the updated combined script for asymmetry analysis. The script calculates the Index of Asymmetry (IA) based on positional data from CSV files containing hand movement data. It also checks IA against thresholds defined from healthy individuals' data and generates visual plots for analysis.

**Prerequisites**

* Ensure that you have the Colab environment set up with Google Drive integration:

from google.colab import drive

drive.mount('/content/gdrive/')

Install required libraries if not already installed:

!pip install pandas matplotlib numpy

Running the Combined Script

Step 1: Preparing the CSV Files

Place the CSV files containing data for both healthy individuals and patients in the appropriate directory on Google Drive.

Healthy individuals’ CSV files will be used to calculate IA thresholds.

Patient data will be used to calculate the IA and compare it against the predefined thresholds.

Step 2: Modifying File Paths

Ensure that the correct file paths for your CSV data are specified in the main() function:

healthy\_data = pd.read\_csv("/path/to/your/healthy\_data.csv") test\_data = pd.read\_csv("/path/to/your/patient\_data.csv")

Step 3: Running the Script

The script runs both the threshold calculation and the test data analysis in one flow. To run the script:

Execute the main() function, which will:

Calculate IA thresholds from healthy data.

Plot the IA values over time for patient data.

Check if the calculated IA exceeds the thresholds.

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Functions Overview**

* **calculate\_ia(L, R)**:
  + Calculates the Index of Asymmetry (IA) between the left (L) and right (R) values.
* **analyze\_asymmetry(data)**:
  + Extracts positional data from the CSV file and calculates the IA values for positions over time.
* **calculate\_healthy\_threshold(healthy\_data)**:
  + Analyzes asymmetry for healthy individuals and calculates the average IA to be used as the threshold.
* **plot\_asymmetry\_over\_time(test\_data, healthy\_thresholds)**:
  + Plots the IA values over time for the patient’s test data and compares them against the healthy thresholds.
* **determine\_if\_results\_are\_ok(asymmetry\_results, healthy\_thresholds)**:
  + Checks if the IA results for the patient exceed the predefined thresholds.

**4. Visualizing the Results**

After running the script, you’ll see a plot showing the IA values over time for the patient’s data. A red dashed line indicates the threshold derived from the healthy data. The script will also print out whether the patient’s IA exceeds the threshold.

* **Example output**:
  + Average IA for positions is displayed.
  + A decision (whether the IA exceeds the threshold or not) is printed.

**5. Maintenance Tips**

* **Updating File Paths**: Always ensure that the file paths for the CSV files are updated when new data is added.
* **Adjusting Thresholds**: If you have new healthy data, re-run the threshold calculation by updating the path to the new healthy CSV files.
* **Plot Customization**: You can modify the plot\_asymmetry\_over\_time() function to adjust the plot’s appearance, such as color or size, depending on the required analysis.

***8.*** ***References:***

[1] Capriotti, T., & Terzakis, K. (2016). Parkinson disease. *Home healthcare now*, *34*(6),   
300-307   
<https://pubmed.ncbi.nlm.nih.gov/27243427/>

[2] Kuhtz-Buschbeck, J. P., Brockmann, K., Gilster, R., Koch, A., & Stolze, H. (2008). Asymmetry of arm-swing not related to handedness. *Gait & posture*, *27*(3), 447-454.‏

<https://www.sciencedirect.com/science/article/pii/S0966636207001361>

[3] Miller-Patterson, C., Buesa, R., McLaughlin, N., Jones, R., Akbar, U., & Friedman, J. H. (2018). Motor asymmetry over time in Parkinson’s disease. *Journal of the neurological sciences*, *393*, 14-17.‏

<https://www.sciencedirect.com/science/article/pii/S0022510X18303125>

[4] National Institute of Neurological Disorders, & Stroke (US). (2004). *Parkinson's disease: Challenges, progress, and promise*. National Institute of Neurological Disorders and Stroke, National Institutes of Health  
<https://www.ninds.nih.gov/current-research/focus-disorders/parkinsons-disease-research/parkinsons-disease-challenges-progress-and-promise>

[5] Poewe, W., Seppi, K., Tanner, C. M., Halliday, G. M., Brundin, P., Volkmann, J., ... & Lang, A. E. (2017). Parkinson disease (Primer). *Nature Reviews: Disease Primers*, *3*(1).‏  
<https://sci-hub.se/https://www.nature.com/articles/nrdp201713>

[6] Sonne, J., Reddy, V., & Beato, M. R. (2022). Neuroanatomy, substantia nigra. In *StatPearls [Internet]*. StatPearls Publishing.‏  
<https://www.ncbi.nlm.nih.gov/books/NBK536995/>

[7] **oculus integration package documentation** <https://developer.oculus.com/documentation/unity/>