Metric Analysis in Healthcare-Based ExerGame (Virtual Gym)

Sakib Hasan
October 2022

1. Introduction to Virtual Gym

Virtual Gym is a virtual reality-based healthcare system designed and developed for the elderly. With the evolution of exergames, adapting VR games for medical purposes are also getting popular to a large extent. Keeping that in mind, we have developed this virtual healthcare game called Virtual Gym. It is specifically designed to play on *Meta Oculus* headsets, and to play this game is as simple as just install the app and enjoy.

There are currently five games that are playable in Virtual Gym which enables the player to try out different physical postures to be in a fit condition. This idea of enabling elderly people to enjoy gym-like activities in a fun and enjoyable way was key to formulating the name- Virtual Gym. The games that can be played are- *Slice Saber, Bubbles Under the Sea, Balloons Party, Archery, Rock Climbing* and *Rings*. Different physical capabilities are tested through these games on a pre-defined complexity and speed which is called a prescription. For example, in *Slice Saber* a player tries to hit incoming targets with the headset controllers as successfully as they can. Meanwhile, in *Balloons Party*, a player tries to pop randomly appearing balloons with the controllers in a virtual party room.

However, the initial system does suffer from some lacking which needed to be addressed for making the experience for the users more helpful and rewarding. And this project contributes to two aspects of the whole Virtual Gym ecosystem. Firstly, deploying the backend data streaming and collecting system on a publicly accessible server, and secondly developing techniques to analyze the collected session data from users to extract some quantifiable metrics to help understand the capability, improvement of the users over time as well as creating fruitful prescriptions for the upcoming sessions based on some previously devised metrics.

2. Virtual Gym Limitations - Project Contributions

The current system suffers the following limitations:

1. There is no automated way to collect the session data from different users to be saved in a database. The current version of the game has to be installed locally, and once a session is played, the data is saved locally in the headset memory as a *.tvg* file.

2. There is also no dedicated server for this system to store the session data in a database and also host the WebGL version of the games. So, a dedicated server would be needed ideally to host the data ingestion pipeline, saving them in a database,
3. No methods for analyzing the data for different meaningful metrics from the collected data from a user.

My contribution through this project is to address the aforementioned limitations. Firstly, I have deployed a public server machine for the whole Virtual Gym ecosystem which can be used for data ingestion, storing, analyzing, hosting the webGL version etc., and also I have adapted the **SoDa-TAP**, a data streaming and analysis system to accommodate the needs of Virtual Gym to stream and ingest data directly from the Oculus headset, storing them in CrateDB. Next, I also implemented methods to extract meaningful metrics from a user session to monitor player's performance over time and also formulate automated prescriptions for the upcoming session for a user and feed it back to the database.

3. Background and Related Work

With rapid progress in the field of Virtual Reality based applications, studies regarding the effectiveness of these VR headsets and applications in different domains are seeing a surge as well. This stretches to the entertainment sector to sports to medical sector to almost all the other sectors as well. The idea of developing different applications for healthcare adopting the idea and potential of virtual reality to support has gained a lot of interest over the past few years. Those games, known as Exergames, are developed for different platforms like- *Kinect, Oculus* etc. for different purposes. And each of those games has generated scope for different studies for the researchers as well.

Some studies applied Machine Learning techniques to recommend future physical activities for a person based on a previous set of data, while some studied how to adapt the difficulty in-game through different measurements. *Zhao et. al.* developed a machine learning approach on data collected from certain exergame players to recommend suitable games for that player [1]. They derived features for their model from a pre-defined questionnaire set which a player has to answer before starting a game. The machine learning approaches are also used to predict sickness or assess stability during a game session. Predicting cybersickness was attempted by *Jin et. al.* in [2]. Meanwhile, *Yeh et. al.* developed an assessment tool which identifies imbalance and vestibular dysfunction with real-time analysis of the data from a virtual reality based rehabilitation system [3]. *Martin et. al.* took similar approaches with machine learning to detect sickness caused by excess stress caused by the lengthy virtual reality games using features derived from various physiological signals [4].

The most common studies have seen the implementation of deriving various game-specific and posture-specific metrics to assess the performance, improvement or degradation of physical capabilities for a person. *Stranick et. al.* leveraged virtual reality and exergames to promote physical activities based on a players' previous range of motion and also visualizing the movement capability of a specific player [5]. Similar promotion of physical activities through VR games are proposed by *Campo-Prieto et. al.* in patients with Parkinson's disease by tracking their ranges of motion and also their postural balance during a game session [6].

A simple but comprehensive study was conducted by *Staiano et. al.* in [7] where the effective potential of exergames were studied by analyzing player range of motion, playing time, balance of the body structure, and also acceleration of the joints of the body which were tracked. *Yoo et. al.* developed VRmove [8], a framework designed to track actual exertion(heart rate), perceived exertion(what the player felt), and also a measure of the field of view that the player could perceive during the game to quantify the player performance. SImilar to this, Immersive FPS Games, developed for virtual reality platforms by *Lugrin et. al.* [9], collected directly the in-game accuracy of target hit, in-game score, playing field-of-view, distance covered during a game, and also range of motion for the joints that are tracked to provide feedback on a player's performance.

The studies were not only limited to these domains, rather they exceeded these boundaries. A very interesting study was proposed by *Yannakakis et. al.* in [10] where the authors proposed real-time game adaption in virtual environments to enhance user satisfaction, their performance, which eventually would also help improving the capabilities of a specific gamer. *Lee et. al.* also developed a technique based on tracked joints' data to predict the arm movement of a player during an exergame session[11]. Thus, these findings imply that the potential of studying these virtual reality based games not only enhances the game or its' usability but also helps the users to use them for other purposes of enhancing their physical capabilities or maybe predicting when the stress from these games are affecting the players' health adversely.

4. The Data-Processing Pipeline of the Virtual Gym Ecosystem

Ultimately, our target is to establish a properly working ecosystem for the Virtual Gym where we do not have to manually extract session data from different headsets. Rather from authentication of a new user to streaming real-time game data to the server and then analyzing and storing those data in databases have to be transferred into an automatic process.

As the games of Virtual Gym are developed on unity, we found that with the help of **Apache Kafka, and Apache Spark,** we can easily stream data directly from the game through the headset as long as it is connected to the internet. And then we can store that data directly to the **CrateDB** for initial purposes. For user registration and authentication, however, **Google Firebase** authentication with email address was adopted.

The CrateDB is not a secured database in some ways, and also it does not provide significant database features like- Joins, Relationship etc. Thus, albeit CrateDB is a good option to store data directly from streaming through Kafka and Spark, we needed a strong and secured database to store refined data and also the calculated meaningful metrics. The metrics on the other hand were implemented using Python directly.
The flow diagram below explains the whole pipeline of the Virtual Gym ecosystem for now.

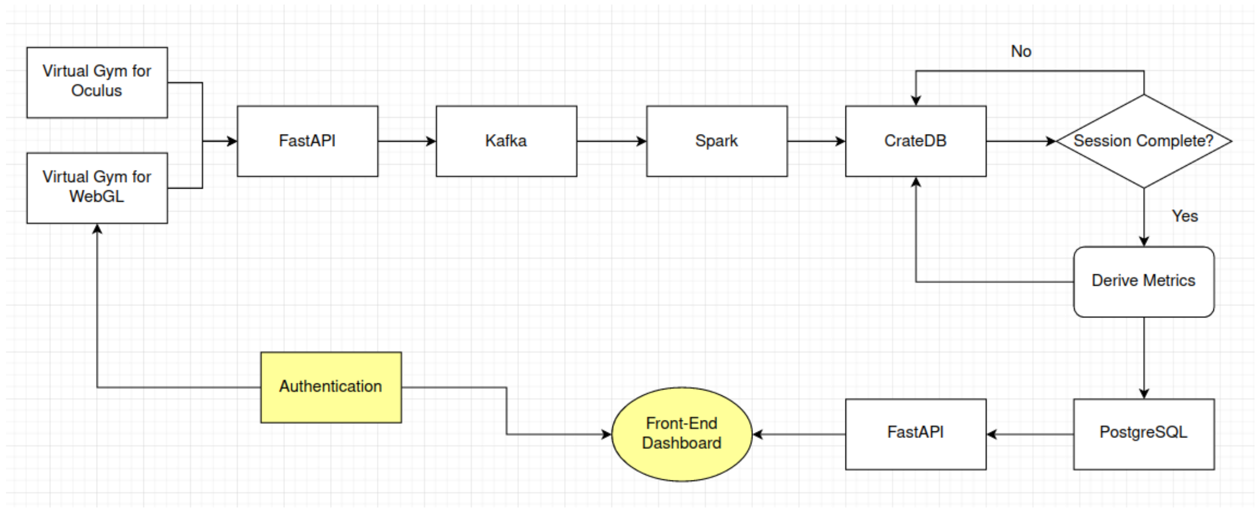

Figure 1: Pipeline of Virtual Gym

Currently, the whole pipeline is set up and it is running successfully. The whole backend system launched on a separate server machine. From Figure 1, the whole proposed pipeline for the Virtual Gym can be understood as follows-

1. Firstly, a game session has to be played which can be done either on Oculus headset directly, or a WebGL version of the game is currently under development.
2. When a user starts playing the game, the session information, frames of user movement data, and also events that the user performs are being streamed to *Kafka* producer which acts as a bridge between the database and headset for live data streaming.
3. After we have the data in Kafka, we receive them as Pandas Dataframe in different Jupyter Notebooks running inside a docker container deployed on the backend server.
4. These notebooks transfer the received data in Kafka to CrateDB with the help of deployed spark jobs.

Once the data is stored in the preliminary database, calculation of two types of metrics from those session data which have successfully ended with an end event start. A script is constantly running on the backend server developed in python, which retrieves data from the CrateDB if a session has reached its end successfully. After deriving the metrics, the metric information is put back to both the PostgreSQL and the CrateDB databases. PostgreSQL is the primary database for the Virtual Gym's backend.. The postgres database directly communicates with the dashboard where one should see the analysis and visualization of the captured data. To communicate with the database from the headset, metric calculation script, and also dashboard, an API container which is extended from the SoDa-TAP system and built upon the *FastAPI* package distributed from python is implemented. To be noted specifically, the whole pipeline is currently working successfully with only the *PartyRoom Balloons* game in the Virtual Gym platform. The *Front-end Dashboard* and *Authentication* from game and dashboard are a work in progress currently. All the other parts of the pipeline are working properly on the server for the Virtual Gym.

4.1 Database Schema and Relationships:

We have two separate databases. The first one being the CrateDB which is our open and preliminary data entry point. CrateDB is very flexible in handling data streams from *Apache Kafka* and *Apache Spark*. That is the reason we chose this database to work with in alignment with the *SoDa-TAP* platform. The other one is our primary and securely maintained PostgreSQL database where we only store our valuable information which are the session information and the corresponding metrics. The following diagram displays the relationship schema of our CrateDB. The only difference in postgres database is the lack of the ***frames & events*** table which we don't store anywhere except CrateDB.

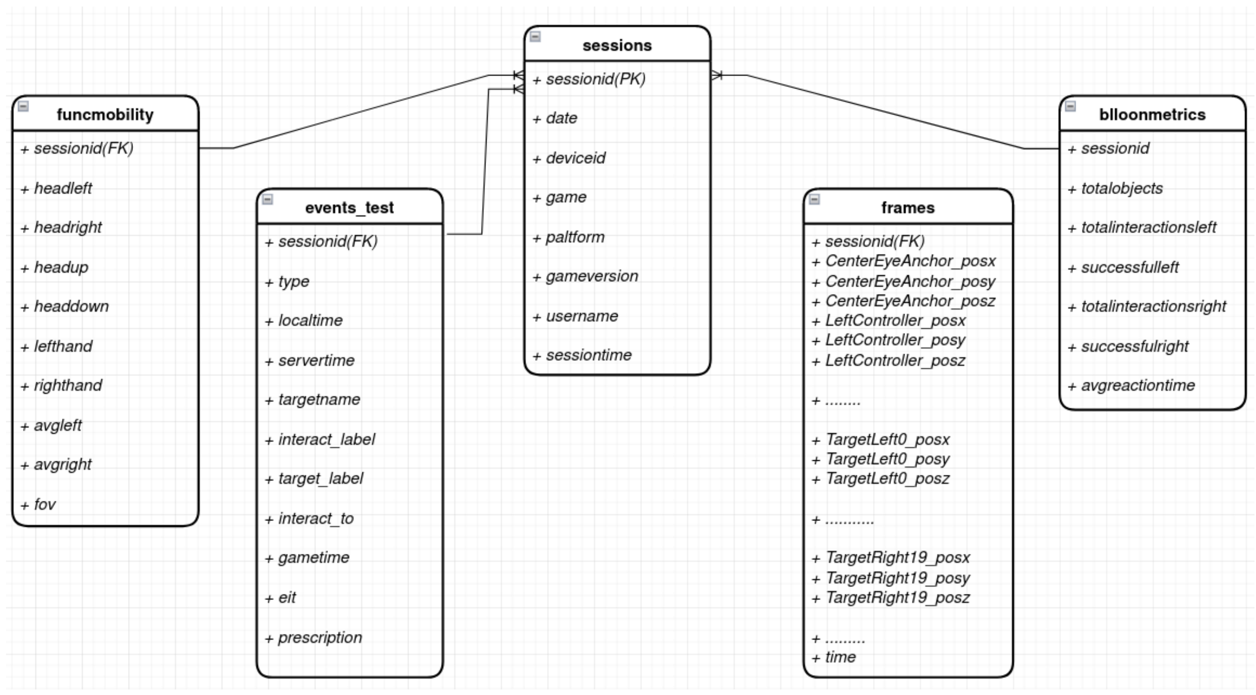

Figure 2 : Schema Diagram of CrateDB

From Figure 2, the **sessions** table is the central reference point for all the information stored in the other tables. In sessions, the information regarding a particular session like- *which device was used? Which player has played?* etc are kept. A unique *SessionId* is generated using the combination of *server time, deviceid, & username.* This very *SessionId* acts as the reference to all the other relations in the database.

The **frames** table holds the physical movement coordinates tracked with oculus during a game session, which is captured at a rate of 10 frames per second. Based on the data from this particular table, the player-specific or functional mobility metrics. Currently, 3D position coordinates in the playing field, and their relative position coordinates are also captured from Oculus. The points that are captured are- *Center Eye Anchor*(Center of the head)*, Left Hand Controller, Right Hand Controller*. Along with the position coordinates, the relative positions of these are captured with respect to the head center. Additional to these tracked points, the position coordinates of the appearing balloons(objects) are also captured.

The **events** table, on the other hand, holds all the events that have occurred during a balloons game session. There are 5 types of events that we expect from a successfully performed session. They are- *start of a session, appearance of an object, interaction with an object, disappearance of an object, & end of a session.* Based on these data, we calculate the game specific scores for the balloons game. The other two tables, **functionalMetrics & balloonMetrics,** are updated for each completed session after the metrics calculation. A successfully ending session should ideally have a *start* and an *end* event along with several *interact, appear, and disappear* events. To find events from a particular session, the unique *sessionid* is required for reference.

The two other metrics tables namely- **functionalMetrics** and **balloonMetrics** are the storages where meaningful information from a successful session are kept after through calculation from *frames* and *events* table. They are only updated after a session has been completed from the user end. If a session does not reach the end successfully, or if a session cannot transmit the events data for lack of internet or other difficulties from the Oculus, these tables for those specific sessions will not be updated.

5. Deploying the Server

The first contribution of this part of the project was to deploy a public server where we can host our desired pipeline. The current accessible server can be found deployed in one of the machines with the public address of *http://129.128.184.214* in the **SSRG** lab of the Computing Science department at the University of Alberta.

The server has been active for about two months now, it has been running on *Ubuntu 20.04 LTS* operating system as the core. The load is handled by a *Intel Core i7-6700T* CPU with 8 individual cores which supports applications from both *x86* and *x64* architectures. With built-in RAM memory of 16 gigabytes, the server also has the additional processing capacity of the integrated *NVIDIA GeForce GTX 960 OEM* graphics processing unit of 4 gigabytes capacity.

Different parts of the whole ecosystem are deployed on the server to communicate with each other simultaneously. The first container is the starting point in our pipeline *Figure 1,* and that is the **FastAPI** container built with the python FastAPI distribution. This container plays a significant role for communication among different parts in the pipeline. The next container launched is the **Apache Kafka**. This makes streaming of data more flexible to feed to the next step. To stream through *Apache Kafka*, a docker container adopted from the SoDa-TAP system has been launched on Virtual Gym's server. The Kafka is communicated through the deployed **FastAPI** container based on the JSON object sent from the game. The next container launched are the **Jupyter Notebooks** which are used to ingest the data fed from the Kafka producer. In the notebooks, the *Apache Spark* library is leveraged to manipulate the streamed data with python. The docker container is set up to continuously run spark jobs on the jupyter notebooks to accept the data from the Kafka. To absorb and store the initially sent data, the **CrateDB** docker container is launched which is communicated through the Jupyter Notebooks running spark jobs.

The next part of the ecosystem is the transfer of the calculated data to the secured *PostgreSQL* database. The PostgreSQL database is set up on the server for Virtual Gym. To keep the transferred data secure and persistent, the PostgreSQL database was installed directly into the server without using a dockerized container. The purpose of having this database is direct communication of any front-end website through different APIs. The schema of the PSQL database is as follows:

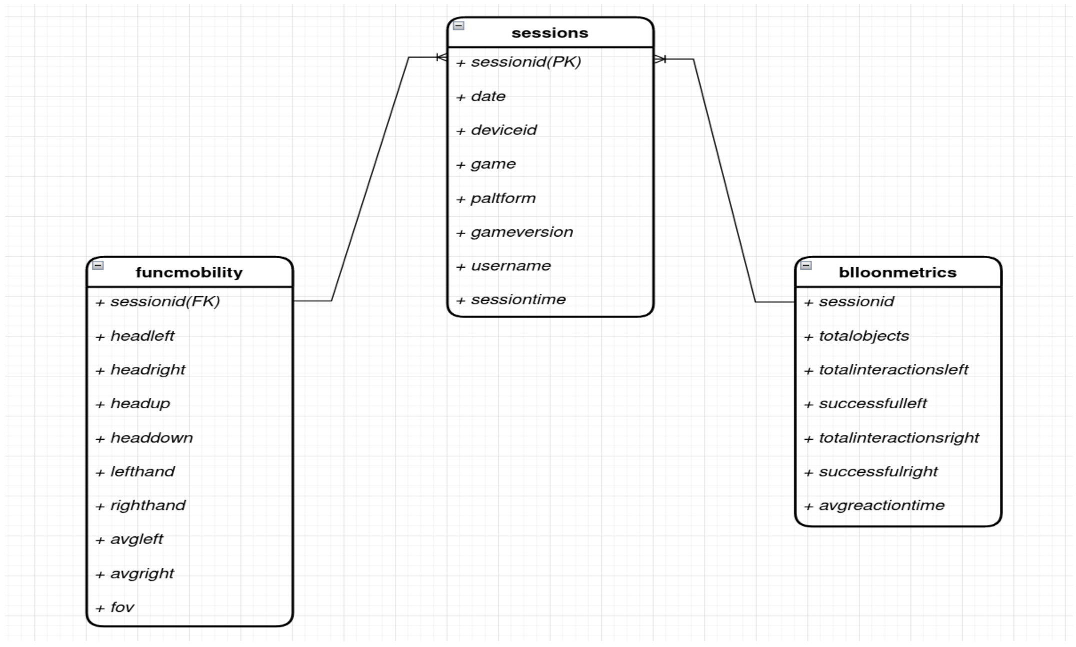

Figure 3 : PostgreSQL Schema

The schema here is pretty much alike with the CrateDB schema that we have in Section 4.1. The difference between the schemas are the lack of ***frames*** & ***events*** relations. As the PostgreSQL database is only meant to connect to the front-end for analysis works, a full snapshot of the CrateDB is not required to be stored in PSQL. Only after successful calculations of the metrics data, postgreSQL is populated with the data from newer sessions.

6. Metric Analysis

For the performance and improvement analysis of the players in the ecosystem, different metrics were adopted from the literature. There are two general types of metrics that were considered:

- *Game-specific metrics:* These are the metrics which are directly related to the performance in-game.
- *Player-specific metrics:* These are the metrics that are directly related to the postural stability or body positions that are tracked during the game sessions for a player.

6.1 Game-Specific Metrics

There are a number of game-specific metrics that are being developed. They are-

● **Session Time:** The time that a player has played a session, and the total time that a player has played one specific game is referred to as the in-game time. The calculation process is very straightforward for this. Once a *SessionId* is passed as an input to the function, it directly calculates the in-game time of that session. After calculation, this is put back in the *balloonmetrics* table in the databases. The calculation is done as follows:

SessionTime = Timestamp of last frame - Timestamp of first frame………(1)

● **Balloon Game Score:** The in-game score is calculated per session for a user for one game. It is based on the percentage of successful or unsuccessful hits to the targets in a game by a user. For example, *Slice Saber* game's score is being calculated on the number of correct and incorrect hits that the player makes during a game session with the sabers(controllers of Oculus). On the other hand, for *Balloons Party* this is calculated based on the number of correct and incorrect contacts of the hand controllers and balloons in the game. As our current standing only can deal with the Balloons game, the game specific scores that we collect are- *Total Objects Appeared, Total Objects Interacted With Left Hand, Total Successful Interactions with Left Hand, Total Objects Interacted with Right Hand, and Total Successful Interactions with Right Hand.*

To calculate the total number of objects(balloons) appearing in the playing field, the events captured from a specific session are searched through, and the number of total appearances are counted. On the other hand, to count the number of interactions along with the number of successful interactions for each hand, similar exploration through the database is done and the corresponding counts are saved. In our database schema (Section 4.1), we have *target_label* &

interact_label fields in the relation **events.** Based on the value of these two, a successful or an unsuccessful interaction can be found and counted. The same logic is being followed for calculation.

- **Average Reaction Time:** The average reaction time for a session indicates the quickness of a user to locate and react to an object appearing in the game session. The reaction time for an object is calculated based on the following simple equation:

$$\textit{Reaction Time} = \textit{Time of Interaction} - \textit{Time of appearance}\ldots\ldots\ldots(2)$$

In our method, we simply find out if an object appears on the playing field, if that was interacted with a correct/incorrect hand within 15 seconds from appearance. If yes, we find out the reaction time for that object following **equation 1.** If not, we ignore that object. Finally, if there are ***n*** interactions in total, we calculate the average reaction time of the user for that specific session as follows:

$$Average\ Reaction\ Time \ = \ \sum_{i=1}^{i=n} (ReactionTime[i])........(3)$$

6.2 Game-Independent Mobility Metrics

The player-specific metrics are more important as it will track the improvements of the player over time, and also help the system to generate future prescriptions for the game sessions. The considered metrics for now are:

- **Range of Motion:** This metric finds out how much the player could stretch his/her head movement in four directions, along with the maximum stretch of both the hands as well. This is player-specific as it does not have to be changed according to the games.

Based on the ranges of motion, the player and the system will get an idea on how capable the player is. For now this is being calculated in a window of 25 seconds each. And after a game session has been played, the player will be able to look at the maximum and average ranges of motion for head movement, left and right hand movement. Adapting window based calculations will provide support for the streaming scenario.

- **Maximum Field of View:** This is another significant metric that is being developed. As the targets in the game appear randomly at random positions in the playing field of the headset, players need to move his/head body positions and also head positions to identify a target. Based on this value we will be able to account for generation of targets in the next prescription as well for the player.

For now this is calculated as the angle between the straight lines connecting the leftmost and rightmost points of the head with the initial center position of the head. For a normal person the maximum field of view should be 180 degrees without any target on the playing field. But this would not be 180 degrees even for a normal person in our game as the appearing targets might not be stretched to 180 degrees at first. The following image will give a better idea of how it is being calculated for now.

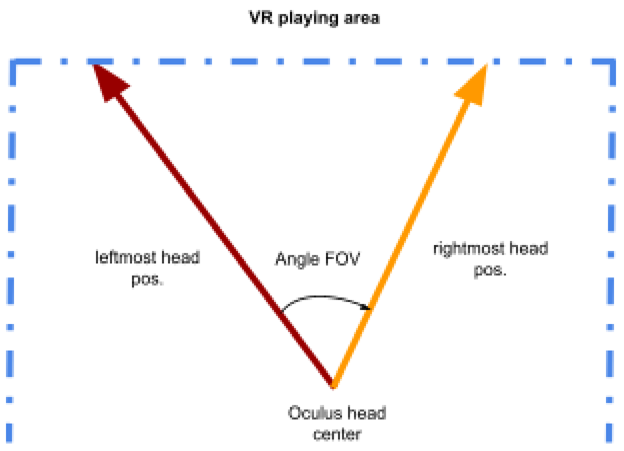

Figure 4 : Field of View for Oculus player

6.3. Metrics Calculation

For the automatic calculation of the metrics from sessions which have successfully transmitted an *end* event, there were two options. One, deploying a scheduled python script, and two, a scheduled cron-job. Both of the options work in a pretty similar way where it can be configured how often the calculation script should execute. Currently, the first option was chosen to deploy on the server for scheduling the python script execution every **5 minutes.** It uses the *scheduler* package, and keeps the python script running in the background.

7. Conclusions and Future Work

To encapsulate, this project aimed at making the Virtual Gym pipeline more well established, and has laid down the seeds for many future works to be done on the system. The project made three contributions to the whole Virtual Gym pipeline. Firstly, an automated way to transmit and save data from the VR headset Oculus was established for different games of Virtual Gym. Secondly, a public server to host the backend of the Virtual Gym was launched. This server can also be the host for the WebGL version of the game in the coming days. And lastly, methods to calculate and derive meaningful metrics from a completed session's data were implemented and integrated in the server which eventually saves the information back to both the available databases. Although the pipeline is only working for the *Ballons PartyRoom* game for now in the Virtual Gym game, a lot of potential directions of work have been laid out through this work. In the coming days, work will be needed to done in the following sections of the pipeline-

- The prescriptions (defining the targets and their position in a session) have to be generated integrated into the headset automatically for a user for a particular game. The ideal goal for this should be generating the prescriptions randomly first and then adapting session per session based on previous metrics.
- A front-end user interface to visualize the performance of users over time needs to be incorporated.
- The game-specific metrics need to be figured out for each game, and then methods to calculate them need to be implemented to work simultaneously in the pipeline.
- Users' feedback through questionnaires is a vital source of information to understand user expectation, experiences, lackings of the system, and also potential upgrade opportunities.
- The authentication system for the game and the server is also not developed in the current version, which has to be sorted out to prevent misuse of the game and restricting a user from sharing this game with whoever one likes.

Overall, with completion of this project's contribution to the Virtual Gym pipeline, new directions of work have evolved. Continuing to update the whole pipeline, and eventually launching a fully loaded version of the Virtual Gym game is the ultimate goal in the coming days.

Pointers

1) Github - pass it to Victor

The GitHub repository containing the scripts, docker containers and report can be found here: *https://github.com/sakib1486/vgDockers*

2) Soda-Tap documentation

Virtual Gym pipeline definition can be found here:
https://www.sodatap.ml/en/latest/useCases/virtualGym/

References

[1] Z. Zhao, A. Arya, R. Orji and G. Chan, "Physical Activity Recommendation for Exergame Player Modeling using Machine Learning Approach," 2020 IEEE 8th International Conference on Serious Games and Applications for Health (SeGAH), 2020, pp. 1-9, doi: 10.1109/SeGAH49190.2020.9201820.

[2] W. Jin, J. Fan, D. Gromala and P. Pasquier, "Automatic Prediction of Cybersickness for Virtual Reality Games," 2018 IEEE Games, Entertainment, Media Conference (GEM), 2018, pp. 1-9, doi: 10.1109/GEM.2018.8516469.

[3] Shih-Ching Yeh, Ming-Chun Huang, Pa-Chun Wang, Te-Yung Fang, Mu-Chun Su, Po-Yi Tsai, Albert Rizzo, "Machine learning-based assessment tool for imbalance and vestibular dysfunction with virtual reality rehabilitation system", Computer Methods and Programs in Biomedicine, Volume 116, Issue 3, 2014.

[4] N. Martin, N. Mathieu, N. Pallamin, M. Ragot and J. -M. Diverrez, "Virtual reality sickness detection: an approach based on physiological signals and machine learning," 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2020, pp. 387-399, doi: 10.1109/ISMAR50242.2020.00065.

[5] Stranick, T., Lopez, C.E. (2021). Leveraging Virtual Reality and Exergames to Promote Physical Activity. In: Stephanidis, C., Antona, M., Ntoa, S. (eds) HCI International 2021 - Posters. HCII 2021. Communications in Computer and Information Science, vol 1421. Springer, Cham. https://doi.org/10.1007/978-3-030-78645-8_50.

[6] Campo-Prieto, P.; Cancela-Carral, J.M.; Rodríguez-Fuentes, G. Wearable Immersive Virtual Reality Device for Promoting Physical Activity in Parkinson's Disease Patients. *Sensors* **2022**, *22*, 3302. https://doi.org/10.3390/s22093302.

[7] Staiano AE, Calvert SL. The promise of exergames as tools to measure physical health. Entertain Comput. 2011 Jan 1;2(1):17-21. doi: 10.1016/j.entcom.2011.03.008. PMID: 23378860; PMCID: PMC3562354.

[8] Soojeong Yoo, Marcus Carter, and Judy Kay. 2018. VRmove: Design Framework for Balancing Enjoyment, Movement and Exertion in VR Games. In Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts (CHI PLAY '18 Extended Abstracts). Association for Computing Machinery, New York, NY, USA, 295–307. https://doi.org/10.1145/3270316.3272054.

[9] Jean-Luc Lugrin, Marc Cavazza, Fred Charles, Marc Le Renard, Jonathan Freeman, and Jane Lessiter. 2013. Immersive FPS games: user experience and performance. In

Proceedings of the 2013 ACM international workshop on Immersive media experiences (ImmersiveMe '13). Association for Computing Machinery, New York, NY, USA, 7–12. https://doi.org/10.1145/2512142.2512146.

[10] G. N. Yannakakis and J. Hallam, "Real-Time Game Adaptation for Optimizing Player Satisfaction," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 2, pp. 121-133, June 2009, doi: 10.1109/TCIAIG.2009.2024533.

[11] Howon Lee, Jimmy Lee, Erik Brockbank, "Predicting Arm Movements in Virtual Environments".