A Body Tracking Framework for Neuromotor Rehabilitation with Multimedia Application Support

Elvis Ribeiro[1], Alexandre Brandão[2], Marcelo Guimarães[3], Leonardo Rocha[1], José Remo Brega[4], Rodolfo Villaça[5], and Diego Dias[5(✉)]

[1] Universidade Federal de São João del-Rei (UFSJ), São João del-Rei, MG, Brazil
{elvishribeiro,lcrocha}@ufsj.edu.br
[2] Pontifícia Universidade Católica de Campinas (PUC-Campinas), Campinas, SP, Brazil
alexandre.brandao@puc-campinas.edu.br
[3] Universidade Federal de São Paulo (UNIFESP), Osasco, SP, Brazil
marcelo.paiva@unifesp.br
[4] Universidade Estadual Paulista (UNESP), Bauru, SP, Brazil
remo.brega@unesp.br
[5] Universidade Federal do Espírito Santo (UFES), Vitória, ES, Brazil
{rodolfo.villaca,diego.dias}@ufes.br

Abstract. Multimedia applications and games have become essential tools in rehabilitation and healthcare, providing an interactive and engaging approach that enhances patient motivation, enables progress tracking, and supports personalized treatment plans. These applications offer an innovative platform for both physical and cognitive therapy, allowing healthcare professionals to explore novel treatment methods while offering patients a more enjoyable and effective recovery experience. Despite recent advancements in hardware and software, there remains a need for integrated solutions that streamline the development of new applications, particularly those that leverage body-tracking sensors. This paper introduces a framework designed to simplify the creation of natural user interface (NUI) applications based on inertial sensors, supporting functionalities such as tracking, session creation, and data storage. Additionally, we present examples of applications developed using our framework, demonstrating its potential to facilitate new applications that utilize the user's body as a primary interaction medium.

Keywords: Virtual Reality · Serious Games · Multimedia · Neuromotor Rehabilitation · Sensors · Body Tracking · Software Development

1 Introduction

As people age, they may develop diseases that could be mitigated by regular physical activity, such as sarcopenia, osteoporosis, obesity, and cardiovascular

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
O. Gervasi et al. (Eds.): ICCSA 2025, LNCS 15648, pp. 345–360, 2025.
https://doi.org/10.1007/978-3-031-97000-9_22

diseases. The current landscape of physical inactivity [6,8] is linked to public health issues and the rise of non-communicable diseases (e.g., atherosclerosis, type 2 diabetes, and certain cancers), highlighting the importance of culture and lifestyle as strategies for health promotion and disease prevention. In this context, activities that engage large muscle groups—such as the muscles of the lower limbs—are particularly beneficial, as they promote intense blood circulation and, consequently, an increase in energy expenditure. For instance, body movements that simulate walking fulfill these requirements.

The search for new forms of entertainment that promote increased physical activity is of great social interest and tends to be more readily accepted when introduced in a playful manner and presented in immersive, interactive VR environments. Maloney et al. [7] suggest that interactive digital games hold potential benefits for increasing physical activity, particularly among children. Guimarães et al. [5] demonstrated that physical fatigue can limit user interaction time with gesture-based virtual technologies, implying that greater muscle strength could enhance interaction time with emerging gesture-supported technologies, such as hand-motion-controlled smart TVs, integrating these interactions naturally into daily life.

Multimedia applications play an important role today across various fields, including as rehabilitation tools [1,3,13]. Interaction is a key component of user engagement with virtual environments, achievable through controls, touch, or even gestures. The solution presented in this work leverages the user's natural interaction (NUI), captured via embedded inertial sensors. Ideally, the interaction between the user and the system should be as seamless as possible. In virtual reality (VR) systems, where immersion precludes the use of traditional input devices like keyboards, mice, and touchscreens, interaction is often achieved through specialized solutions such as joysticks, gestures, and voice. For users unable to operate a joystick, or when the goal is to encourage body movement, body-gesture interaction becomes an excellent alternative.

Body tracking technologies have been widely used in the gaming industry [4]. Optical tracking devices, such as Microsoft's *Kinect* or *Mocap* solutions,[1] utilize computer vision to track human movement [12]. However, Kinect has limitations in terms of freedom, accuracy, and occlusion, restricting user movement to the device's field of view and making it impossible to capture movements from certain angles. These issues are mitigated with *Mocap* solutions, which use multiple cameras and numerous markers on the user. However, the high cost of *Mocap* systems is a barrier to choosing a good yet reasonably priced solution. In this work, we use the *Biomechanical Sensor Node* (BSN) as an interaction device.

The BSN [2] is an innovative body-tracking device developed by Brandão and his team. It consists of a network of sensors strategically placed on the body to capture movements and postures in real time, allowing the user to control virtual environments through NUI. This data is processed to create a precise digital representation of the user. The BSN has potential applications in fields

[1] Body tracking using markers attached to the user's body and multiple high-definition cameras surrounding the environment.

such as physical rehabilitation, sports, and interactive entertainment, offering a valuable tool for professionals and researchers due to its precision and ease of use.

The main justification for this work stems from the challenge of simulating user interaction within a virtual environment in an intuitive and immersive manner. Another significant benefit of using virtual environments is that they provide a controlled and safe space for training, as the virtual setting poses no real danger compared to a physical training environment. Therefore, research focused on interaction methods for virtual environments is crucial for advancing and disseminating new applications.

In this work, we present the definition of a body tracking framework that facilitates the creation of NUI applications using the BSN device, along with the development and storage of physiotherapy sessions. This framework is demonstrated as an example for applications aimed at neuromotor rehabilitation, and additional example applications are also presented.

2 Methodology

Developing a framework involves a series of methodological steps. First, it is necessary to clearly identify and understand the problem or need that the framework aims to address. Next, an in-depth analysis of the problem domain is important to identify common patterns and abstractions. Based on this analysis, the next step is to design the framework's architecture, defining its main classes and interfaces as well as their interactions. The framework implementation should be modular and extensible, allowing developers to customize and extend its functionality as needed. Finally, it is crucial to provide detailed documentation and usage examples to help developers understand and effectively utilize the framework. It is important to note that framework development is an iterative process requiring continuous refinement and evolution based on user feedback and changes in domain needs.

The first step was to identify the common requirements of applications that would benefit from using the framework. Some functional requirements were identified, such as: multi-BSN connection – the ability to synchronize multiple BSNs to control different parts of the human body; user-friendly interface – an intuitive interface that enables the discovery and connection of BSN devices; and remote configuration – the ability to configure the framework from another device (such as a smartphone or desktop).

We adopted the Model-View-Controller (MVC) architecture and followed the event-driven programming paradigm. MVC is a widely used design pattern in software development that separates a system's components into three distinct parts. In this way, the asset is fully encapsulated, making the entire BSN configuration process transparent to the programmer/user.

2.1 Model

The Model is the module responsible for managing BSN devices, collecting data, and rotating virtual objects. It consists of the *BLE* API[2] BluetoothHardwareInterface and the BSNHardwareInterface class. This class does not store BSN data directly but instead maintains a list of a new class, BSNDevice, which abstracts a BSN device, and RotatableObject, the class that is actually available for the programmer to use.

The BSNHardwareInterface class is responsible solely for communicating with the BluetoothHardwareInterface and storing a list of BSNDevices. Upon receiving the FindBSN command via the ConfigurationAPI, the discovery process begins. Whenever a *beacon*[3] is found, BSNHardwareInterface receives a response containing the device's name and MAC address, then adds it to the BSN list and sends a response back to the ConfigurationAPI, indicating that the device has been found. This process is asynchronous and is triggered whenever a new *beacon* is detected, with all communication handled through *callbacks*.

The procedure for connecting a BSN is analogous to the discovery process. The ConfigurationAPI sends a connection command, passing the address of the device to be connected as a parameter. This connection command is forwarded to the BLE device, which eventually responds with a successful connection, providing all available services and characteristics. At this point, the *BSNDevice* object registers its *delegates* to receive data updates whenever they occur on the BSN.

The BSNDevice class represents a BSN within the application, containing its name, address, and a boolean field indicating whether it is connected. It also includes an object of the RotatableObject class, which is responsible for manipulating 3D objects in Unity[4]. This class provides methods that actually handle BSN data. Therefore, BSNDevice does not directly manage the data but rather passes it to RotatableObject. This setup allows the controlled object to be swapped by a BSN at runtime.

The RotatableObject class provides methods that receive data and are called by BSNDevice upon each *notify*[5] from the BSN, specifically the linear acceleration, raw data, and gravity vector. These methods are LinearAccHandler(Vector3 linAcc), RawDataHandler(Vector3 accelerometer, Vector3 gyroscope, Vector3 compass), and GravityVectorHandler(Vector3 gravityVector), respectively. These methods are not implemented by default and must be implemented by the user programmer. The ResetOffset() method is also available and should be called whenever BSN rotation compensation is required.

[2] *Application Programming Interface*.

[3] Signal from the BSN's BLE transmitter.

[4] Unity is a leading real-time development platform that allows the creation of 2D and 3D games and simulations for various platforms (PC, consoles, mobile, VR, and AR). It features a visual editor and scripting capabilities, providing professional tools that meet the requirements of any game.

[5] Data transmission from a BLE device.

To address orientation differences between the device and the virtual environment, the RotatableObject class has two functionalities: swapping one axis with another and inverting the direction of an axis. Both functionalities are useful for correcting axis orientation depending on the position where the BSN is attached to the user's body.

Another issue arises when we want the initial position of the BSN to differ from the default position – the *T-pose*[6]. Whenever initialized, the BSN needs to be kept at rest in a horizontal position for approximately 30 s before use to calibrate the sensors. In cases where the controlled object is the leg of an avatar, the BSN is attached vertically to the user's thigh, resulting in an unpredictable rotation of the thigh when the rotation should be zero. With this in mind, it is necessary to apply an *offset* to the BSN's rotation, so that:

$$Q_{object} = Q_{offset} * Q_{BSN}. \tag{1}$$

Q_{offset} must be calculated whenever rotation compensation is required, that is, to set the current rotation as zero rotation. The calculation is derived from Eq. (1), by multiplying the inverse of $Q_{BSN\ rotation}$ on both sides of the equation.

$$Q_{object} * {Q_{BSN}}^{-1} = Q_{offset} * Q_{BSN} * {Q_{BSN}}^{-1} \tag{2}$$

$$Q_{offset} = Q_{object} * {Q_{BSN}}^{-1} \tag{3}$$

When added to a *GameObject* in Unity, the RotatableObject class provides configuration fields in the *Inspector*. Two additional fields are created. The *Simplified Name* field is a nickname given to the object that appears in the configuration interface, while the *Precision Points* field sets the number of decimal places used for rotation. Its default value is two, based on empirical testing, which showed that more than two decimal places generally adds noise.

2.2 View

The View module consists of an interface created in Unity with the help of the *Modern UI Pack* asset for icons and animations [9]. This module has two classes: UIController, which manages the interface, and NetConfigurationServer, which handles network communication.

Since the goal was to create an integration asset capable of functioning in both new and legacy projects, the interface was designed to be as minimally invasive as possible. The interface includes a button that activates and deactivates the configuration interface, which is initially disabled.

The configuration screen consists of a search button, a panel listing the discovered BSNs, and a button to apply (assign the *offset*) rotation compensation to all BSNs. The search button remains disabled until a configuration server is found (as discussed in Sect. 2.3). Once the search is initiated, each discovered BSN is added to the panel. The item representing the discovered BSN includes

[6] In computer animation, the T-pose is a standard pose for the skeleton of a 3D model before it is animated.

the device's name, MAC address, a dropdown menu for selecting the object to control, and a connection button.

The `UIController` class is responsible for managing user input and projecting the information received from the Controller onto the user interface. It operates in conjunction with the `NetConfigurationClient`, which communicates over a network with the `NetConfigurationServer`. Initially, the `NetConfigurationServer` broadcasts messages on the network containing the application's *token* in search of a server. Upon receiving this token, the server sends a connection request (steps 2 and 3 of the sequence diagram). Subsequently, the client requests a list of available `RotatableObjects` and stores them in a list (step 4). From this point forward, all interactions are triggered by the end user through the interface (steps 5, 6, and 7).

2.3 Controller

The Controller is the layer responsible for receiving commands and applying them to the Model, as well as returning the Model's current state to the View. It consists of a configuration Application Programming Interface (API) that provides the necessary methods for a usage session of the asset.

Alongside the ConfigurationAPI, the NetConfigurationServer class acts as a listener for network commands, enabling remote configuration of BSNs by listening on a UDP socket managed by the *Ruffles* library [10]. Initially, the ConfigurationAPI listens for broadcasts on network port 5556, containing a token. If the token received in the broadcast matches the expected token, it is sent back to the originating device, and the system immediately awaits a connection request.

Once connected, the ConfigurationAPI waits for messages from the client, following the protocol outlined in Table 1. The data is separated from the command using the "/" character as a delimiter.

3 Results

The results of this work include direct implementations of the BsnAsset as an interaction tool in VR projects, demonstrated through applications such as Mazze and the Penalty Simulator. These two examples showcase the functionality of the BsnAsset for engaging both upper and lower limbs in interactive environments. Additionally, the framework supports creating and recording therapy sessions for neurorehabilitation and physiotherapy, enabling customized and trackable treatment activities.

3.1 Immersive Brain Puzzle

BLIND proposed an initial interaction model using the Gear VR controller, where the user points to the piece to select it and clicks to rotate it by -90º. However, upon further analysis, it was found that the Gear VR controller was not the most suitable for interaction, as bedridden patients would face challenges

Table 1. Messages awaited by the configuration server and client

	Message	Received Data
Server	Broadcast	Application identification token
	Connection	-
	StartDiscovering	-
	Connect	MAC address
	GetObjNames	-
	SetObj	MAC address and object name
Client	AllBsn	JSON containing a summary of all connected BSNs
	ObjNames	List of available RotatableObject names

in holding and clicking the controller. Therefore, the BSNAsset was employed in the study since the BSN is attached to the patient's wrist, relieving them from the burden of holding the device and clicking.

The functional requirements for interaction with the BSN are as follows:

- The patient must be able to move a virtual hand by pointing with their arm.
- The section under the virtual hand is selected and rotated by -90º when the patient rotates their wrist along the sagittal axis.

The virtual hand must move in only two dimensions. Therefore, we had to convert the three-dimensional rotation of the BSN into the two-dimensional position of the virtual hand. The solution was to create an empty object positioned near the camera, serving as a pivot for the rotation of an invisible cube. The position of the invisible cube was then converted to a two-dimensional position using Unity's WorldToScreenPoint(Vector3) method, which returns the position of an object relative to the camera. Figure 1 shows a simulation of the movement from the center to the right.

The selection and rotation of elements are performed using the X-axis of the gravity vector. A small threshold is set for selecting the piece, and a larger threshold is established for rotation. The piece is selected through hand rotation (according to sensitivity). The hand is locked onto the selected element, and by continuing to rotate, the piece is rotated. Movement is released once the user returns their hand to the horizontal position.

The rotation sensitivity increases as the pivot moves in the negative direction of the Z-axis, limited from 0 to -10, where 0 is the least sensitive and -10 is the most sensitive (Eq. (4)). The selection and rotation sensitivity are also adjusted, provided that the selection sensitivity is less than the rotation sensitivity, and both are below 9.8 (Eq. (5)).



(a) Pivot without rotation



(b) Pivot rotated to the right

Fig. 1. Virtual hand movement driven by pivot rotation

$$Pivot_{position} = (0, 0, -Sense), 0 < Sense < 10. \tag{4}$$

$$0 < Sense_{selection} < Sense_{rotation} < 9.8 \tag{5}$$

3.2 Mazze

Mazze is a game featuring a procedurally generated maze [14], within which a sphere and several golden coins are placed (Fig. 2). The goal is for the player to navigate the sphere through the maze to collect all coins in the shortest time possible.

During gameplay, the user rotates the BSN to control the maze's movement, allowing for precise reproduction of the user's hand movements as they guide the sphere through the corridors. The BSN is positioned horizontally, either on the back or palm of the hand, and utilizes only the device's rotation (Euler angles) to manipulate the maze. Due to its plug-and-play functionality, the *BsnAsset* only requires specifying that the maze is to be controlled by the device, making setup quick and intuitive. Mazze serves as a valuable tool in motor rehabilitation for the hands, allowing patients to perform therapeutic exercises in an engaging

and interactive environment. As they play, patients improve hand coordination and control, seamlessly integrating rehabilitation into a fun activity.

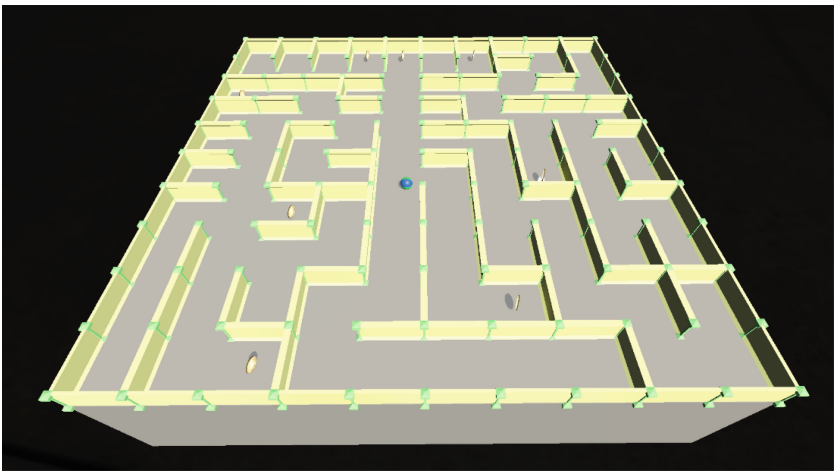


Fig. 2. Initial state of the Mazze game

3.3 Penalty Simulator

The Penalty Simulator fully leverages the capabilities of the *BsnAsset* to create an immersive and interactive rehabilitation experience. This application captures raw data from the BSN through the asset's API, enabling precise tracking of leg movements. The game takes place in a virtual soccer field, where the user faces off against a goalkeeper, as illustrated in Fig. 3. By kicking a virtual ball, the user engages in physical activity that can help improve lower limb strength and coordination, while also enjoying a motivating gameplay experience.

During the simulation, the user manipulates the avatar's leg and shin through two BSNs: one placed on the anterior thigh (medial portion) and the other on the lower leg (distal portion). These sensors capture the user's physical movements and translate them into the virtual environment. The *BsnAsset* operates in a *plug-and-play* mode, where it is easily configured in Unity to allow the BSNs to control the specified avatar segments directly. The kick force, calculated based on the accelerometer data provided by the asset's API, determines the power applied to the virtual ball. This realistic simulation aids in enhancing coordination and strength in the lower limbs, contributing to neuromotor rehabilitation efforts.

When the avatar's foot makes contact with the ball, the kick force is determined by multiplying the acceleration data obtained from the BSN's accelerometer by a predefined constant, representing the mass of the foot. This calculated force is then applied to the ball as an impulse, directed along the collision normal



Fig. 3. Penalty Simulator Scenario

and opposing the direction of impact. This approach simulates realistic physical interactions, allowing the ball's motion to respond dynamically based on the strength and direction of the kick.

The goalkeeper's save is calculated by predicting the ball's trajectory to determine where it will reach the goal, using ballistic motion equations. Based on this prediction, the animation in which the goalkeeper most closely intercepts the ball is triggered. However, as these animations were created deterministically, the goalkeeper's potential save points are limited to predefined areas. Despite this limitation, the simulator offers an engaging way for health professionals to encourage patients to perform leg exercises in a playful and interactive environment, where the physical movement required for kicking contributes to rehabilitation efforts.

3.4 Recorder and Recorder Controller

ReBase [15] was developed as a non-relational database designed specifically for storing neuromotor rehabilitation sessions, encompassing both session details and the movements performed by the patient during exercises. In [15], the storage structure is outlined, and an API is provided to facilitate data recording and retrieval.

To leverage this recording API with 3D avatar rotation data, two assets were developed: the Recorder and Recorder Controller. These assets are distributed as packages, allowing for independent use across different devices, provided they are on the same LAN network, or on the same device, as demonstrated in Sect. 3.5.

The advantage of using the Recorder Controller on a separate device is that it enables the patient to engage with VR applications through a VR headset, while the recording control is managed remotely.

The Recorder is a tool designed to recognize, interpolate, and save rotation data for a 3D avatar. It functions independently of the body tracking method, provided the avatar adheres to the standard defined by the Recorder.

The Recorder's execution flow begins when the network communication module receives a command to initiate the motion recording session. The data is then stored at a sample rate that matches the current frame rate (or Frames per Second, FPS) of the application.

3.5 Mobility Tests

[11] conducted a comparative analysis between the VICON optical body tracking system and the BSN solution. His tests were based on the execution of the Sit-to-Stand task, an exercise in which the patient sits and stands from a chair without hand support. For this comparison, hip goniometric angles (formed on the lateral axis by the abdomen and thigh) and knee angles (formed on the lateral axis by the thigh and lower leg) were measured. These measurements were obtained using a set of three BSNs positioned on the abdomen, thigh, and lower leg (frontal plane), alongside the corresponding rotations obtained from the VICON system.

Thus, a demand arose for an application to capture this data with the following requirements:

- Simple application, without VR;
- Ability to connect three BSNs simultaneously; and
- Data must be recorded in a CSV (Comma-Separated Values) file containing raw rotation information (Accelerometer and Gyroscope) and goniometric angle values for the hip and knee.

To meet these requirements, we created the BSN Mobility Tests application, utilizing all the previously developed assets: BSNAsset, Recorder, and Recorder Controller, all running on a single device, as there is no need for patient interaction with the application. In addition to the mentioned assets, a CSV file recorder was created to export session data simultaneously with the recording in ReBase.

Finally, [11] presents a moderate to strong correlation between rotations in both solutions, reaffirming the validity of using BSNs in healthcare.

3.6 Body Tracking

Following the demand from BSN Mobility Tests, the need arose to create an application capable of complete body tracking using BSNs. This application supports recording accelerometer, gyroscope, and 3D rotation data, as well as recording rotation data for each body joint (goniometric angles).

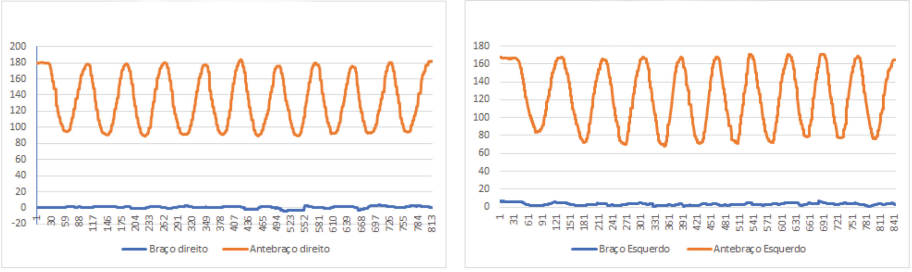
The application allows the user to freely utilize body joints, limited to groups of joints: upper right – right arm, right forearm; upper left – left arm, left forearm; lower right – right thigh, right leg; and lower left – left thigh, left leg.

The BSNs must be positioned on the body. On the arms, BSNs should be positioned with the LED facing upwards and the device face oriented outward from the body. On the legs, BSNs should be positioned with the LED facing upwards and the device face oriented forward.

Charts. The following section presents charts generated from data collected by BSN Body Tracking. The charts were created using Excel software, and the data used was collected from a user performing flexion and extension movements of the upper and lower limbs.

Bicep Curl. The bicep curl is an exercise that works the biceps brachii muscles. The movement involves flexing the elbow to bring the hand closer to the shoulder, performed with the arms extended alongside the body, elbows bent, and wrists in supination.

Data collection was conducted with the user performing a total of 10 repetitions of the exercise with the right arm (Fig. 4a) and 10 repetitions with the left arm (Fig. 4b).



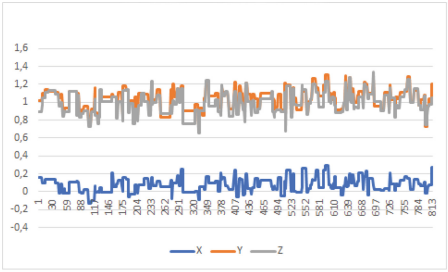
(a) Right arm bicep curl (b) Left arm bicep curl

Fig. 4. Goniometric charts of the bicep curl

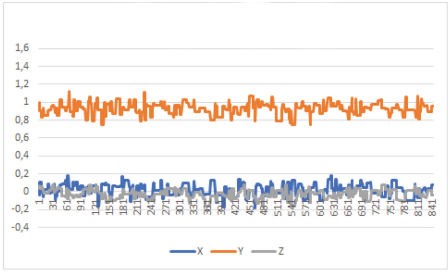
The charts in Fig. 4 show that the goniometric angles of the elbow decrease as the user flexes the elbow, and increase as the user extends the elbow, ranging between approximately 180º and 90º.

The charts in Fig. 5 represent the accelerometer data collected during the exercise. It is evident that the accelerometer data for the right and left arms are similar, as are the data for the right and left forearms. The forearm accelerometers show higher values than the arm accelerometers, as the forearms rotated during the exercise while the arms remained static.

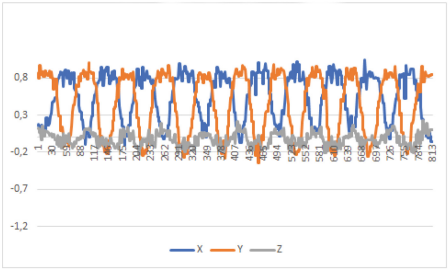
Stationary March. Stationary march is an exercise that simulates walking in place. The movement is performed by lifting the knee until it aligns with the hip, then returning to the initial position.



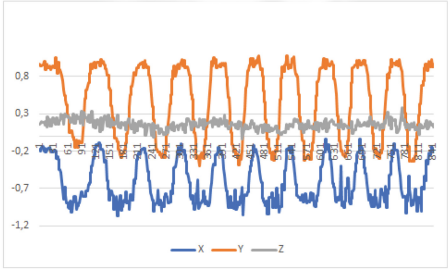
(a) Right arm accelerometer



(b) Left arm accelerometer

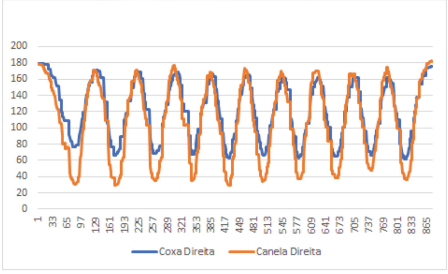


(c) Right forearm accelerometer

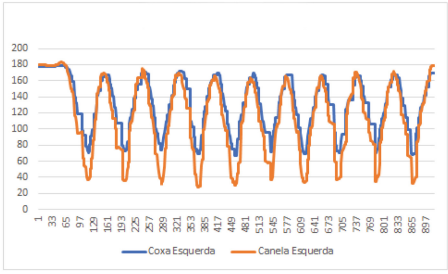


(d) Left forearm accelerometer

Fig. 5. Accelerometer charts for bicep curl



(a) Stationary march with the right leg



(b) Stationary march with the left leg

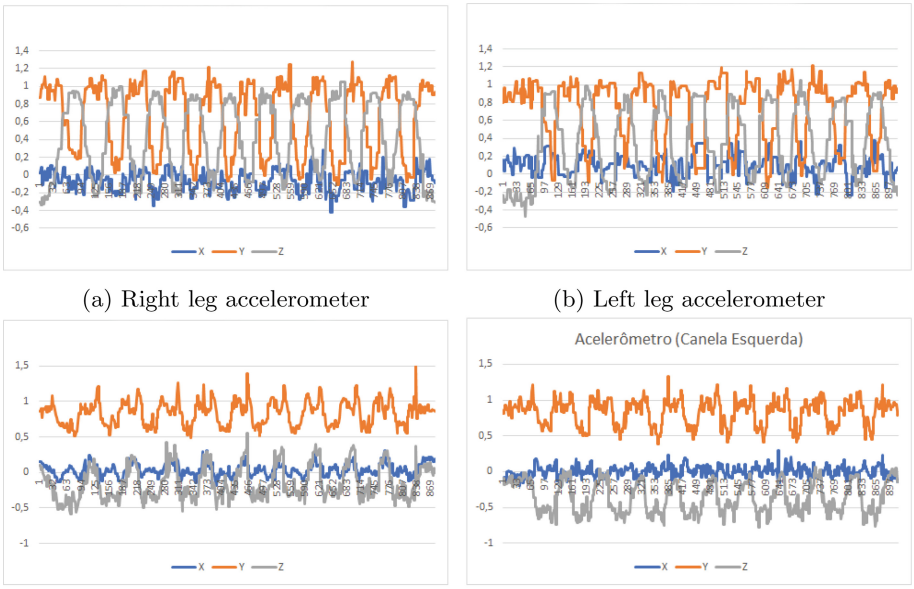
Fig. 6. Stationary march charts

Data collection was conducted with the user performing a total of 10 repetitions with the right leg Fig. 6a and 10 repetitions with the left leg 6b.

The charts in Fig. 6 show that the goniometric angles of the knee and hip decrease as the user flexes the knee, and increase as the user extends the knee, ranging between approximately 180º and 90º.

The charts in Fig. 7 represent the accelerometer data collected during the exercise. It is evident that the accelerometer data for the right and left legs are similar, as are the data for the right and left shins. The leg accelerometers display axis swapping, indicating a rotation of the device. Conversely, the shin

accelerometers show no rotation but exhibit variations in the Y and Z axes due to the upward and downward shin movement.



(a) Right leg accelerometer

(b) Left leg accelerometer

(c) Right shin accelerometer

(d) Left shin accelerometer

Fig. 7. Accelerometer charts for stationary march

4 Conclusions and Future Work

With the increasing presence of VR in homes and medical clinics, it has proven to be a highly effective tool for encouraging physical exercise and aiding in the rehabilitation of patients with neuromotor impairments due to various conditions, such as stroke. This growth presents a significant opportunity to develop a wide range of applications tailored to specific therapeutic needs. However, despite the accessibility and immersive capabilities provided by VR headsets, achieving a natural and intuitive interaction method remains a key challenge in VR application development.

The drive to create a natural interaction method that is sufficiently versatile to function across various VR applications, while allowing for customization based on the unique needs of each project, has been a guiding principle of this work and the motivation behind the creation of the BSNAsset.

In this study, we proposed and developed an asset that facilitates the connection of multiple devices, enables human body tracking, and supports the control of a virtual avatar. The solution we developed is straightforward to import and

use within any Unity project, offering a flexible tool for diverse applications. As demonstrated, the BSNAsset can be used to create comprehensive body-tracking applications, with the capability to interact seamlessly with both upper and lower limbs.

Acknowledgment. The authors acknowledge partial funding from the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Funding Code 001, the National Council for Scientific and Technological Development (CNPq), and the Research Support Foundation of the State of Minas Gerais (FAPEMIG). We also thank the Research and Innovation Support Foundation of Espírito Santo (FAPES) for the resources provided through the PROAPEM project (368/2022 - P: 2022-NGKM5) and the PDPG project (129/2021 - P: 2021-GL60J), which were fundamental to the completion of this work.

References

1. Aramaki, A.L., Sampaio, R.F., Reis, A., Cavalcanti, A., et al.: Virtual reality in the rehabilitation of patients with stroke: an integrative review. Arq. Neuropsiquiatr. **77**, 268–278 (2019)
2. Brandão, A.F., et al.: Biomechanics sensor node for virtual reality: a wearable device applied to gait recovery for neurofunctional rehabilitation. In: International Conference on Computational Science and Its Applications, pp. 757–770. Springer (2020)
3. Cano Porras, D., Siemonsma, P., Inzelberg, R., Zeilig, G., Plotnik, M.: Advantages of virtual reality in the rehabilitation of balance and gait. Neurology **90**(22), 1017–1025 (2018). https://doi.org/10.1212/WNL.0000000000005603
4. Difini, G.M., Martins, M.G., Barbosa, J.L.V.: Human pose estimation for training assistance: a systematic literature review. In: Proceedings of the Brazilian Symposium on Multimedia and the Web, pp. 189–196 (2021)
5. Guimaraes, M.P., Martins, V.F., Brasil, G.J.C., Trevelin, L.C.: Um modelo de processo de desenvolvimento de interfaces de gesto: Definição e um estudo de caso. In: XXXVII Conferencia Latinoamericana de Informática (CLEI), vol. 1, pp. 378–390 (2011)
6. Hallal, P.C., Bauman, A.E., Heath, G.W., Kohl, H.W., Lee, I.M., Pratt, M.: Physical activity: more of the same is not enough. Lancet **380**, 189–190 (2012)
7. Maloney, A.E., Threlkeld, K.A., Cook, W.L.: Comparative effectiveness of a 12-week physical activity intervention for overweight and obese youth: Exergaming with dance dance revolution. Games Health J. Res. Dev. Clin. Appl., 1–8 (2012)
8. Malta, D.C., Silva, J.B.: Policies to promote physical activity in Brazil. Lancet **380**, 195–196 (2012)
9. Michsky: Modern UI pack (2021). https://assetstore.unity.com/packages/tools/gui/modern-ui-pack-150824
10. MidLevel: Ruffles (2021). https://github.com/MidLevel/Ruffles
11. Otoboni, G.B.: Comparação entre sensores inerciais e sistema de análise de movimento VI na avaliação da tarefa de Sit-to-Stand em adultos jovens saudáveis, Master's thesis, Escola de Engenharia de São Carlos (EESC-USP) (2022)
12. Pfister, A., West, A.M., Bronner, S., Noah, J.A.: Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis. J. Med. Eng. Technol. **38**(5), 274–280 (2014)

13. Rodrigues, L.G., et al.: Upper limb motion tracking and classification: a smartphone approach. In: Proceedings of the Brazilian Symposium on Multimedia and the Web, pp. 61–64 (2021)
14. Styanton: Maze generator (2021). https://assetstore.unity.com/packages/tools/modeling/maze-generator-38689
15. Trotta Lara Barbosa, T., et al.: Rebase: data acquisition and management system for neuromotor rehabilitation supported by virtual and augmented reality. In: Symposium on Virtual and Augmented Reality, pp. 182–186. SVR'21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3488162.3488225