*Journal on Interactive Systems, 2024, 15:1,*doi: 10.5753/jis.2024.3516

TeamBridge 2.0: an extensible and non-invasive middleware for control and adapting games for motor rehabilitation

Rummenigge Dantas [**Universidade Federal do Rio Grande do Norte** | ***rudson@ect.ufrn.br***]
Alan K. S. Alves [**Universidade Federal do Rio Grande do Norte** | ***alan.klinger@ifrn.edu.br***]
André V. dos Santos [**Universidade Federal do Rio Grande do Norte** | ***andre.vieira@ifrn.edu.br***]

Escola de Ciências e Tecnologia, Universidade Federal do Rio Grande do Norte, Campus Universitário, s/n, Lagoa Nova, Natal, RN, 59078-970, Brazil.

Received: 06 July 2023 • **Accepted:** 05 January 2024 • **Published:** 22 January 2024

Abstract: This work present the TeamBridge 2.0, a middleware able to perform the communication between digital games and hardware devices, like joysticks, mice and cameras (both rgbd and monocular). That communication does not require any modification to the game source code, allowing an old game to be adapted to work with a new hardware device. To prove this, tests were carried out with several games, including one of them being a commercial game. This middleware also allows the use of more than one device at the same time, so we can obtain more accurate information, one device can supply the deficiencies of the other. Finally, we performed tests to make sure that the middleware would not interfere with the user experience. Tests have shown that TeamBridge 2.0 can receive, interpret and send information quickly, the time varies according to the device used, getting 33ms when used with Kinect, 40ms with Leap Motion and 255ms with a DIY Data-Glove.

Keywords: Exergames, Virtual Reality, Middleware, Gesture Interaction, Motor Rehabilitation

1 Introduction

The serious games market reached US$ 5.94 billion[1] (3.7% of the games market) in 2020. A part of these serious games is formed by games created for health care (Drummond et al. [2017]). A systematic review developed by Lu and Kharrazi [2018] analyzed 1553 health games and organized them by primary health topics. Among the games found by Lu and Kharrazi [2018], 5.47% are games aimed at physical activities. These games for physical activities require movement from the user and are also known as exergames (Oh and Yang [2010]). The Nintendo Wii leveraged the popularity of exergames since it allowed the control of the game through real movements (Jones and Thiruvathukal [2012]). Hence, other devices appeared, like Kinect, Leap Motion, etc. Many researchers already used exergames to increase the level of physical activity, motivation, and participation of players (Mugueta-Aguinaga and Garcia-Zapirain [2017]).

These kinds of games and their (console or hardware) platforms are used in physical therapy and virtual reality rehabilitation (Weiss et al. [2014]) to increase the engagement of the participants (Rose et al. [2018]). VR (Virtual Reality) devices have been used along with exergames to motivate patients to attend therapy and engage them during the repetitive tasks. The ability of Games to cause an immersion and thus make people engage in exercise acknowledging their creation for therapeutic purposes since (Bianor et al. [2017b]; Silva et al. [2016]; Silva et al. [2013]; Iqbal et al. [2010]; Takaiwa et al. [2011]; Borja et al. [2018]).

Besides the engagement, some studies demonstrate the benefits of rehabilitation with virtual reality to improve balance (Park et al. [2017]; Corbetta et al. [2015]), gait (Cano Porras et al. [2019]), motor function (Matamala-Gomez et al. [2022]) and other body movements. However, VR devices are constantly evolving, bringing more data, more precision, or lower cost. With that, they lose compatibility with existing games due to the strong coupling (Fregnan et al. [2019]) between the hardware and the game. Because of this constant evolution, a new study could use completely different hardware from another just one year apart (Garrett et al. [2018]). A new VR software can be easier to use than the old ones, but the preparation time of a single VR experiment with an exergame still takes a lot of time (Tieri et al. [2018]).

These problems are associated with the fact that there is no standardization for VR device integration with other software (Lee and Shin [2021]) and exergames. This lack of standardization is one of the challenges for the use of VR in Clinical Research (Garrett et al. [2018]). Part of the software that will be prepared for a new VR experiment will use code snippets already used in previous experiments, and because of this, it would be practical to reuse code or adopt a framework (Wang et al. [2020]). Suppose this experiment uses hardware never explored for VR applications, a new type of hardware, or a fusion of different hardware. In that case, we'll have a heterogeneous environment requiring a middleware[2](Milazzo et al. [2018]; Sartori [2020]) instead a framework.

Based on this set of heterogeneous devices and the lack of standardization found in VR for rehabilitation, we propose the Teambrige 2.0 middleware that can perform the communication between digital games and hardware devices in a non-invasive way so it does not require modification of the game source code, this allows the use of the devices in already developed games. With this solution, it is possible to reuse exergames that are already built, in addition to not

[1]https://www.alliedmarketresearch.com/serious-games-market

[2]Middleware - Software that mediates between software.

needing developers to worry about the particularities of each new device. If we had to create a game to work only on the pinch movement, which is an essential movement for therapy (Mathiowetz et al. [1985]), we could program the game to work with Kinect v2, Leap Motion, and other devices, however, for each device, we would need a Software Development Kit (SDK), plugin or some driver. Furthermore, the information generated by each device would be heterogeneous. Kinect v2, for example, generates a yes or no value for the hand-close, while Leap Motion generates a value between 0.0 and 1.0. Our middleware transforms these values into standard data input to software. In addition to solving the strong coupling, this middleware allows the use of one or more devices at the same time, allowing one device to supplement the deficiencies of the other one.

We organized this work into seven sessions: Introduction, where the problem had been reported; Related work, where works that treated the same situation were listed; Specifications and Architecture; we talked about specifications, structure, and how to configure it; Results, reporting experience to make games compatible with the devices through the middleware; Performance tests, describing the tests performed to ensure that the middleware does not spoil the player experience; Conclusion, where we present final considerations and future works.

2 Related work

Developed by Suma et al. [2011], the Flexible Action and Articulated Skeleton Toolkit (FAAST) is a middleware that allows the user to perform gestures to trigger certain keyboard or mouse commands. Thus, FAAST manages to adapt games to receive, indirectly, sensor inputs, such as Kinect, without the need to modify the game source code.

Movements, such as moving arms or legs, stationary walk and jumping, among others, are computed by FAAST, converted to keyboard or mouse commands and sent to the operating system. When compared to the TeamBridge, FAAST uses an own VRPN (Virtual Reality Peripheral Network) (Taylor II et al. [2001]), different from the open VRPN middleware. In addition, FAAST has no open source code, it is only compatible with Kinect and PrimeSensor, being not applicable to motor rehabilitation. Thus, the creation of a gesture setting is very comprehensive and also more complex. The TeamBridge goes a little further, the programmer can record all the gestures performed by the patient for later playback and, depending on the configuration, it can generate danger alerts.

A Flexible Input Mapping System (FIMS) proposed by Lee and Shin [2021] and is capable of receiving input from new input devices, including new data structures and interpreting and converting them to structures already known by game engines. The objective of this project is similar to ours, the main difference is that it needs access to the game source code, in addition to not working via the network and not having any data storage method, an important feature when it comes to therapy.

Santos [2016] designs another tool that uses a similar approach called uOS Plugin. However, according to the developer it would still need the development of a wrapper that allows compatibility with games that are external to the project. The work of Shen and Pantic [2009] (named HCI) uses the same idea, converting data from common web cameras, Kinect, Tobii Eye Tracker[3] into user-configurable keyboard or mouse outputs. With the GEMINI (Teófilo et al. [2013]), it was possible to match the game The Elder Scrolls V: Skyrim, with Kinect. According to the author, the choice of this game was because of its complex interaction system. Based on this premise, the authors decided to use the game Mount and Blade as a test target for the project, since it has a complex interaction system.

The other solution for the compatibility issue would be to create a framework or game engine. In the case of Intelligent Game Engine for Rehabilitation (IGER), a game engine focused on motor therapy was created by Pirovano et al. [2013]. Among the attributes of IGER, some are essential: monitoring, evaluating, and abstraction of input devices. The game engine manages to monitor the patient, being capable to detect whether the gestures are being performed properly, and the therapist is able to do evaluation of all movements performed by the patient after the section. In addition, the tool is compatible with Sony PlayStation Eye, Microsoft Kinect, Wii Balance Board, Omni Phantom, Novint Falcon [4], Tyromotion Tymo [5] e o Moticon OpenGo Insoles [6].

The IGER is based on the game engine Panda 3D that uses C++. The negative point of this approach is the obligation for games to be developed in this game engine. The aim of the middleware present in this paper is that any game can be used and its functionalities as cross device, incorrect motion alerts and the therapy session recording been automatically incorporated. Other works have developed game engines or frameworks to solve the problems as RehabConnex and PlayMancer by Conconi et al. [2008], the VR-Rides by Wang et al. [2020], also propose a framework for Unity [7]. In addition, the ARTiFICe by Mossel et al. [2013] followed a similar logic, however instead of developing a complete game engine, they build a plugin for the Unity.

In the work of Pandit et al. [2019] a web platform was developed with machine learning techniques for remote monitoring and evaluation of physiotherapy. This system is composed of a Kinect, capable of specifically detecting the human skeleton in 16 key points through the joints. In addition, it has the integration of a web server, a front-end web application and a desktop application. Although the study by Pandit et al. [2019] is not entirely focused on games, they make it clear that there are benefits to including games and hardware devices as a motivating object for clinical purposes in rehabilitation.

Rybarczyk et al. [2019] proposed a web-based platform for telemotor rehabilitation applied to patients after hip arthro-

[3]Tobii Eye Tracker - Hardware that reads the position on the monitor where the user's eyes are focusing

[4]Omni Phantom e Novint Falcon - Haptic devices that enable the simulation of object manipulation on the computer. The user moves a part of the device which sends the movement information to the computer.

[5]Tyromotion Tymo - Device with sensors for capturing posture, focused on therapy applications

[6]Moticon OpenGo Insoles - It has sensors to capture pressure, weight, balance and movement and is used as the sole of the shoe.

[7]Unity - One of the most popular game engines on the market.

plasty surgery. These patients need a postoperative functional rehabilitation program to regain strength and joint mobility. The study justifies that many of the individuals after this surgical procedure have difficulty getting to the rehabilitation center. Therefore, it is necessary to use a technology that enables rehabilitation at home. The architecture of this solution is based on modules composed of a Kinect v2 device that is used to capture the patient's movements and to extract the coordinates of the body's joints. In another work on Telerehabilitation, Souza et al. [2022] developed an exergame that is controlled by the pedaling movement. The authors used an Arduino board with sensors coupled to a cycle ergometer. The Arduino sends data via the serial port to the game (using JSON format).

According to Siddiqui et al. [2015], human action recognition has many aspects: full body movement recognition, facial movement, expression recognition and hand gestures. In the aspect of hand gesture recognition, Siddiqui et al. [2015] used Kinect to extract the skeletal tracks from the dataset to recognize hand gestures in a vehicular driving approach in videogames. The study proposes recognizing acceleration gestures, turning right with acceleration and turning left with acceleration.

Luo et al. [2019] developed a low-cost 3D motion capture system of human skeletal joints. They use a normal monocular RGB camera as a sensor to capture the images and feed a neural network that estimates the pose in 2D. After processing, the neural network returns the 3D estimation. The system is even capable of detecting multiple people in real time with a single image as input.

Mehta et al. [2017] presents a method of capturing the 3D pose of the entire human skeleton in a stable and temporally consistent manner using a single RGB camera. The method combines a convolutional neural network based on a pose regression with a kinematic skeletal fit, making it feasible to estimate the human skeleton pose as a control in real-time applications with an RGB camera. The study demonstrates that the solution achieved significant accuracy with the use of sensors when compared to RGB-D sensors.

In the Konstantinidis et al. [2015] solution, the CAC Framework captures the Kinect data, Wii controls, NeuroSky MindWave, Android devices and disclose it through a Web Service RESTFUL [8]. So any application can connect with the Web Service and extract data. Although the CAC Framework does not allow compatibility with closed source games, the idea of providing the data as a service can be useful for other purposes. It is possible to implement this approach in the middleware used in this paper by creating a module for that purpose. The code was already organized to implement easily these adaptations.

The Table 1 presents a summary of the characteristics of each project compared to our middleware. A differential of our proposal is that it is the only one that reports being able to work with more than one different device at the same time.

Our approach does not require the modification of the game's source code since the main problem to be solved was the need to modify the games whenever a new device was created. It is possible to use Kinect, Leap Motion, and other devices within the game engines. However, for any update, it would be necessary to modify the source code of each of the games and recompile everything. So, more than the game engine is needed to solve this problem. That's why we say our approach is non-invasive.

Table 1. Comparison between tools.

Project	Non invasive	Works on network	Has persistence
FAAST	Yes	VRPN	-
IGER	-	-	Yes
uOS Plugin	-	uOS	Yes
RehabConnex	Yes	-	Yes
HCI	Yes	-	-
PlayMancer	-	-	Yes
VR-Rides	-	-	-
GeMiNI	Yes	-	-
ARTiFICe	-	VRPN	-
FIMS	-	-	-
CAC Framework	-	Web Service	Yes
TeamBridge	Yes	VRPN	Yes

It is important to remember that TeamBridge 2.0 is not intended to adapt VR games but games for motor rehabilitation. However, we use VR tools, so our middleware has a data structure called Tracker compatible with VR environments, but our focus was to make any type of computer game compatible with any input device, so in most cases, the input data will be converted to mouse or keyboard commands. For this reason, TeamBridge will not be compatible with cardboard, as it runs on Android; it will only be compatible with a VR environment that runs on Windows and accepts keyboard or mouse input.

3 Specifications and architecture

Since this is a device for rehabilitation, a support structure to therapy was adopted. Methods were created to record all the data generated during the therapy and mechanisms capable to identify and alert the patient about an exaggerated speed during the gesture and a possible loss of balance. Although it is focused on therapeutic games, there is no limitation from it being used for exergames in general.

The programming language C++ was chosen for development because the VRPN was already developed in this language and for better performance issues.

This is a tool that will be used in games, so it is important that it does not hinder the player's experience to response time. In this way, a performance test was structured, being it validated and compared with FAAST developed by Suma et al. [2011], that is a similar tool.

3.1 Input devices

This project required compatibility with some devices, such as Leap Motion (Bianor et al. [2017a]) or Kinect (Lun and Zhao [2015]).

Leap Motion uses infrared cameras to observe an area one meter around, connects to the computer via USB, and provides an Application Programming Interface (API) for data

[8]RESTFUL - RESTful is a system that uses the REST `https://www.w3.org/2001/sw/wiki/REST` standard, which are services that respond to common HTTP POST and GET calls, among others.

Table 2. Overview of the devices used in this project

Device	Connection	SDK	Type of data
NED Glove	USB	No	Analog
Physio Happy	USB	No	Button
Leap Motion	USB	Yes	Tracker/Analog
Kinect v1	USB	Yes	Tracker
Kinect v2	USB	Yes	Tracker/Analog
Monocular Cam	USB	Yes	Tracker/Analog

access (Weichert et al. [2013]). Developed for hand and finger capture, the SDK itself already performs the recognition of the most important gestures: close and open the hand and the tweezers gesture. Therefore, this information was reused in this project.

Kinect, which acts as a depth sensor, has a normal camera, an IR emitter and receiver. There are three different versions of the hardware, each one working with their specific SDK. Kinect v2 already has close/open hand gesture recognition (Lun and Zhao [2015]).

In addition to the commercial devices, there was also a need for compatibility with NED Glove developed by Silva et al. [2013] and Physio Happy (Bianor et al. [2017b]), devices developed by our research group. The NED Glove uses JS and C# and it is connected via USB. The output generated is something similar to "760,730,831,720,710", where each number represents a finger and how much it is flexed.

In relation to Physio Happy, the focus was to get a very low cost. It is a data glove built only with cloth, elastic, velcro, plastic bottle and a mouse. The data generated by the device are just the mouse clicks, limiting the amount of existing buttons, so the natural connection of the mouse via USB is maintained. The Physio Happy interprets when the user flexes to one side or to the other, and can be adapted to parts of the body such as knee, elbow, wrist and ankle.

The Table 2 summarizes the type of connection for each device, amount of protocols, and heterogeneity of the data. The Button is just pressed or not; Analog is a range of values, and the Tracker is a position in a three-dimensional space.

3.2 VRPN

The VRPN [9] is a network protocol for VR input devices that has been developed to facilitate the use and assembly of a VR lab (Taylor II et al. [2001]). It already has a well-defined client and server architecture, also it has several mechanisms for transmitting data across the network.

3.3 Architecture

The TeamBridge architecture was separated into client and server. It follows the same structure of the VRPN, making the input devices connected to the server side of the application.

The Figure 1 presents the middleware architecture. Our middleware is only compatible with Windows, which must have all the necessary software to communicate with the input devices. VRPN is the transport layer itself. The VRPN also does the communication between the server and the client. The Data Capture layer is responsible for capturing the data from the input devices and sending it to the client through VRPN. In this layer, it is possible to create drivers to adapt new hardware devices. The server side does not perform any processing of this data. It only collects them and makes them available to the client. Since this work used the open VRPN, some drivers were already ready, such as the mouse and the keyboard. The Physio Happy uses the mouse driver, but we created new drivers for the others.

The client-side has most of the code. The VRPN will receive the data from the server side and forward it to the Interpretation layer. In this layer, we have the gesture recognition package; each device can provide several gestures, so it is possible to create classes to interpret more gestures. Still, in this layer, we have the Actions package; an action is a code that executes in the operating system or in the middleware itself. For example, the KeyPress sends to the operating system the signal that simulates a key being pressed. The middleware receives the ShowMessage and displays a message on the screen.

New actions can be created, for example, an effort to run an application, trigger a command over the network, or provide a Web Service. The operating system makes the communication between the middleware and the game. In addition to these, there is also the Therapy module, which has the unique functionality of persisting the interpreted data. In addition, some gesture captures aimed at therapy were created, such as detecting stationary walking, stepping up and down, wrist flexion, raising the hand or legs, and body angle about the floor.

3.4 The controll module

The Teambridge version 2.0 architecture features a module called C0NTR0LL. This module uses machine learning and computer vision techniques as a way to manage actions in games. The use of these techniques allows the capture, training, and classification of image models for three functionalities: face, hands, and body capture. The C0NTR0LL uses a monocular camera to make these three types of capture.

The use of monocular cameras allows the system in its 2.0 version to no longer need Kinect v1 or v2. Studies still need to be done with Kinect Azure integration with Teambridge 2.0. However, the objective of enabling a monocular camera is to reduce the cost of using our middleware.

The system captures the image using an RGB camera for recognition. The image acquisition serves as input to feed the feed-forward convolutional neural networks of the system. The trained models are used to analyze and process the images. These training models are returned as the output of the neural network in the form of a matrix of coordinates.

In addition to the coordinate arrays, the trained and pretrained models return coordinate confidence scores, which correspond respectively to the key points of the poses. The structures generated by neural networks are not anatomically correct structures. They are just an arbitrary set of points identified by the models, and the number of epochs is defined in the model training process by the rehabilitation professional.

The C0NTR0LL system interacts through movements (of

[9]An existing VRPN that has open source available at `https://github.com/vrpn/vrpn` was opted to be used.

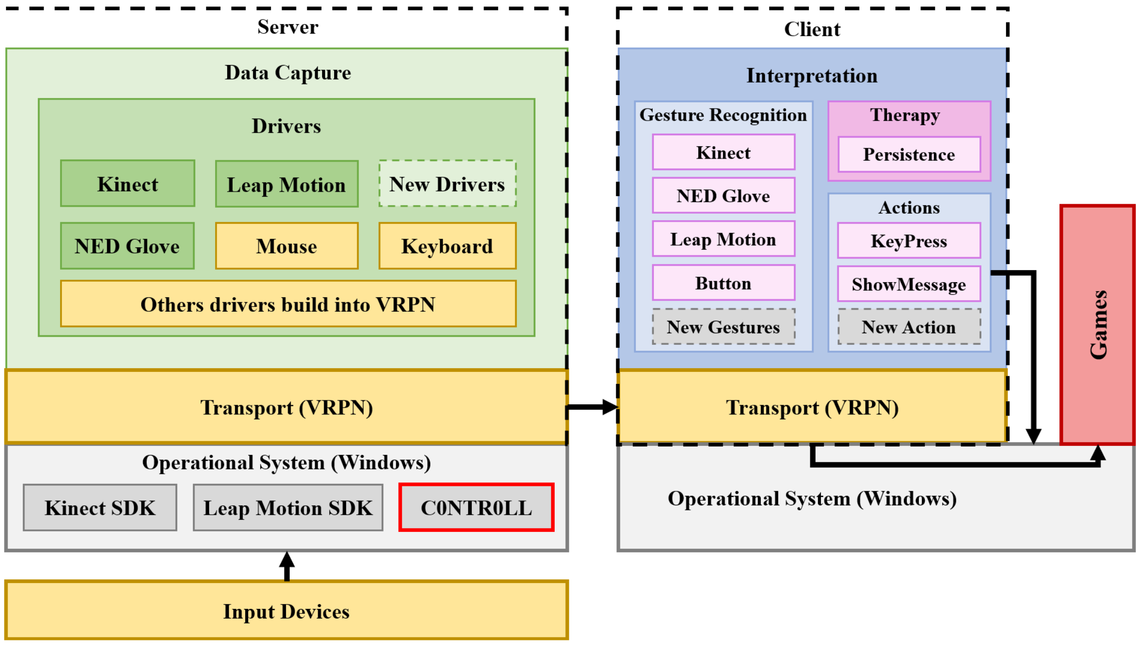

Figure 1. Architecture of the middleware.

the hands, face, and human skeleton) and the selected game. The basic operating flow of the system has its starting point with the choice of the model or neural network (that will be used during the experiment). The user informs this choice. After the user informs which pose he wants to manipulate, the system makes it possible to train a new model or load a pre-trained model, as shown in Figure 2.

The image acquisition phase is related to the production of the digital image by the camera. The pixel values of the images produced are generally related to the intensity of the light reflected from the objects. However, they can also quantified in another way related to different mathematical measures, depending on the sensor used.

The pre-processing phase is responsible for executing the transformations in order to standardize and facilitate the way the image is recognized. In this phase, the C0NTROLL corrects the real-world coordinate system to the sensor coordinate system. The proposed process makes other corrections in the pre-processing phase: the image's projection in the sensor's coordinates, and the reduction of noise introduced during image acquisition, among others.

The high-level processing phase executes the verification of the received data. This phase also realizes the estimation of the parameters for carrying out the verification of correspondences between the known features and the features present in the image.

The phase responsible for obtaining the response from the system is the decision and classification phase. This response can be represented by a position in the image of a pose or a set of data to make automatic decisions. Finally, the last phase of the C0NTR0LL system is responsible for assigning the output of the neural network to a given action in a digital game. This process converts a network pose to a keyboard key. The system's interface allows rehabilitation professionals to customize and assign poses to specific keys. Through the C0NTR0ll, it is possible to assign poses for specific actions in digital games. This attribution process is done exclusively by the rehabilitation professional and the development team, allowing the creation of poses to be specific according to the treatment needs of each patient.

3.5 Human body pose estimation with the C0NTR0LL

The generic functioning of access to the neural networks of the C0NTR0LL system occurs through image acquisition, pre-processing, feature extraction, high-level processing, decision, and classification.

Implementing the neural network responsible for estimating the human body pose is done through the PoseNet model with the ml5.js library. PoseNet is a machine learning model that detects key points of joints in the structure of the human skeleton in real time and of several people. However, C0NTR0LL works to monitor only one person per image.

In this work, the PoseNet neural network recognizes the 17 key points of a human and classifies the pose defined by the rehabilitation professional. These points come from a matrix of coordinates that the model processes.

The implementation of human pose capture has two neural networks: a PoseNet type neural network to detect body key points and another neural network of ml5 type. This ml5 network receives the output of the first network, allowing it to create and train a new model.

The PoseNet neural network sends the 17 combinations of numbers that represent the human pose. Each combination has two coordinates in the matrix: an x coordinate and a y coordinate, resulting in 34 entries. The neural network of the ml5.js library classifies these 34 inputs. So, It makes appropriate output assignments for the keyboard keys. The implementation adds the 17 key point pairs into a simple array. In addition to identifying the key points, through a state machine, it is possible to create, save, and load new poses

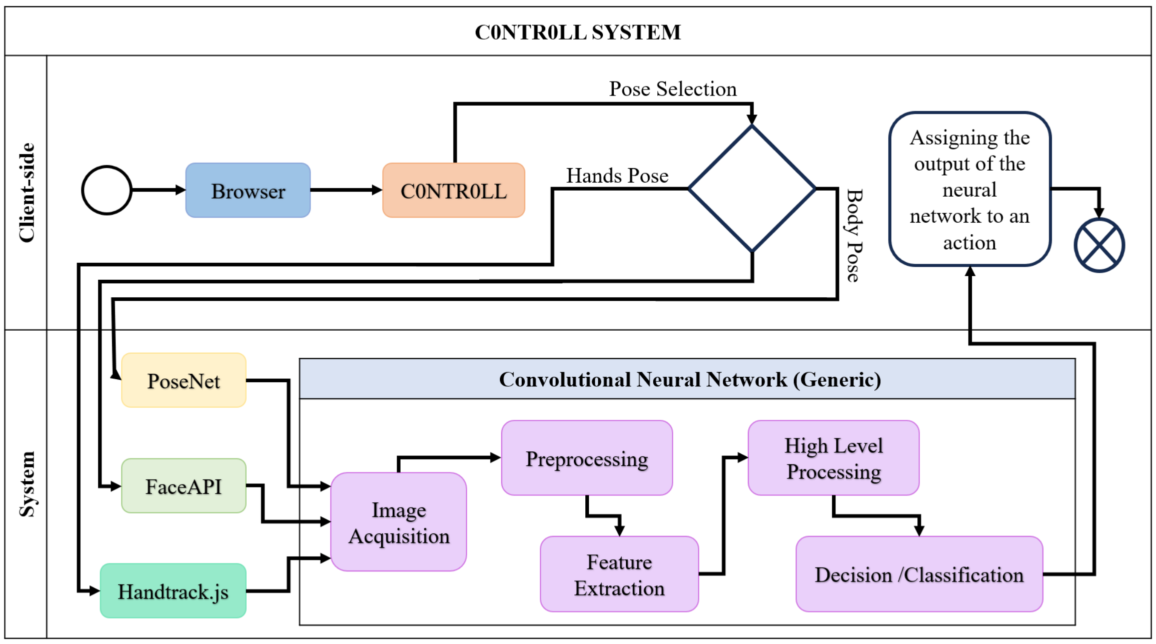

Figure 2. Basic functioning diagram of C0NTR0LL neural networks.

through the compact .JSON file (JavaScript Object Notation). For each group, there are a series of functions with different functionalities. For example, in the function *imageClassifier*, through neural networks, we can recognize the image's content, returning an object with information about the image coming from a pre-trained model. With the *imageClassifier* function, we can train our model.

3.6 Hand pose estimation

The Handtrack.js library is used to make interactions based on web games through hand gestures. C0NTROLL uses handtrack.js to capture an html image element and returns an array of bounding boxes, class names and confidence scores. The user does not need to connect any sensors or additional hardware, but can immediately reap the engagement benefits that result from gesture-based interactions and body input.

The data used by this neural network are mainly from the Egohands dataset as defined by Bambach et al. [2015]. It consists of 4800 images of the human hand with bounding box annotations in various configurations, captured using a Google glass 1 device. A model is trained to detect hands using Tensorflow's object detection API. For this project, an SSD was used with the MobileNetV2 Architecture. The results of the trained model were exported as a saved model. Handtrack.js represents really early steps towards the overall potential of enabling new forms of human-computer interaction with AI in the browser. Therefore, this technology allows mouse movement and keystrokes to be mapped to hand movement for control purposes in digital games.

3.7 Data collect

C0NTR0LL comprises two dedicated blocks: the main block and the Real-Time Interaction block (as shown in Figure 3). The main block is an environment dedicated to the professional therapist who receives all the results from the patient. The Real-Time Interaction block is where all patient data will be acquired.

The C0NTR0LL allows rehabilitation professionals to assign poses, train models, and use these models as a form of control in digital games. The system follows these steps:

1. Data collection: where the professional will inform the poses he wants his patients to perform;
2. Training the model: in this step, the professional retrieves the data collected from the previous step and introduces the model with the number of epochs that suits him;
3. Validate model: where the professional can verify that the models are behaving as expected, that is, if they generate the correct output when classifying a pose.

The data collection step works through the identification of key points by PoseNet. After detecting these points, the game programmer references the key points with geometric figures in the final version of the game. At the end of data collection, the programmer assigns poses to keyboard (or joystick) inputs corresponding to game controls available in the game. Finally, the system generates an extension file, .json, which has the coordinate matrices of all the collected poses.

With the collection stage completed, the rehabilitation professional must train the data so that it is possible to generate the model that patients will use in their sessions. Thus, the step involves informing the .json file generated by data collection and the number of epochs the algorithm will run.

In training the models, the ml5.js library acts to receive the input and output parameters, the task it will perform (which, in the case of the C0NTR0LL system, is the classification algorithm), and the normalization of the data. After the ml5.js library receives all the necessary parameters, model training starts.

If the input variables are linearly combined, as in the case of the multilayer perceptron, it is necessary to standardize the

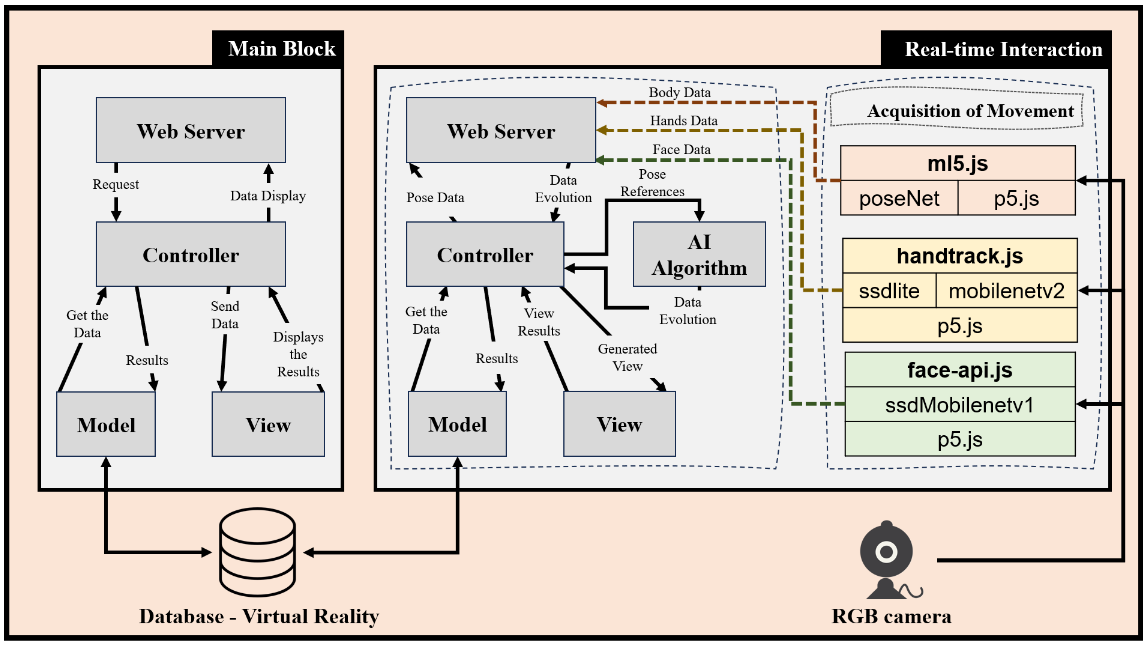

Figure 3. C0NTR0LL module architecture.

inputs. The reason is that any resizing of an input array can be effectively undone by changing the corresponding weights and offsets, leaving the same outputs as before. However, there are several practical reasons why standardizing entries can make training faster and reduce failures.

4 Results

The main result of this research is the Teambridge 2.0 middleware, which is available in this repository: https://github.com/Natalnet/teambridge2. To demonstrate the effectiveness of Teambridge 2.0, we carried out tests which we will present in this section. The first is a load test was carried out with the following objectives:

- Compare the middleware with other similar software. The FAAST was chosen because it's the only one available to download.
- Validate the TeamBridge 2.0 for use, in relation to the response time required to read a device gesture and translate it to the form of a keyboard or mouse command.
- Compare the performance of the devices in various scenarios.
- Identify bottlenecks in reading or interpreting devices.

When conducting the tests, the authors followed this standard performance tests defined by Meier et al. [2007]:

1. **Identify the test environment.** - two computers were used in this test. The first one has an i5-5200 processor with 8 GB of RAM and Windows 10 as operating system. The second computer had a 3.4 Ghz processor (AMD Athlon II X2 B28) with 4 GB of RAM and Windows 7 as operating system.
2. **Identify performance acceptance criteria.** - events that have a maximum delay of 56ms are recognized by humans as a single event, according to Michotte [2017], in a study was published in 1946 and became known as ”launching effect”. Therefore, the delay below the 56ms is the criterion of acceptance for this test, since it is the ideal time to not feel the delay in the commands' execution.
3. **Plan and draw the tests.** - The TeamBridge has been tested with each of the compatible devices separately; With devices combined locally and networked; And even two servers on the same physical machine.
4. **Setting up the test environment.** - Unnecessary applications were removed. The memory used, adding the client and server application, came to occupy 15 MB of RAM. For this reason, the goal is not to verify the processing or use of memory, but the software performance.
5. **Implementation of the test design.** - In order to perform these tests, a small software was developed.
6. **Run the test.** - Finally, the tests were executed and monitored.
7. **Analyze the results, report and redo the test.** - To ensure greater accuracy in the average population, the tests, which the standard score was below 68%, were repeated.

For the verification of significance between the comparisons, a paired T-test was used to compare two data sets, such as verifying the difference between the running time of one program and another. Also, a paired T-test is stronger than the common T-test, justifying the choice of it in this work (Wainer et al. [2007]).

4.1 Software for performance testing

In order to do the performance test, a project was created and had the function of receiving several keyboard inputs and counting the time between each. The goal is to identify how much time will be spent for the software to read the device data, interpret it, and deliver a keyboard input. In order to obtain a statistical significance, a sample of at least 100 measurements is recommended in two independent tests (Meier et al. [2007]). In the case, 300 iterations were performed. This amount of iterations has been chosen to take around 10 seconds via Kinect (v1 and v2), allowing to observed network oscillations. The application calculates the average, median, and standard deviation of the time, in addition to storing the minimum and maximum interval between one input and another. This application was built in C++ and is also available on GitHub to be used for future tests.

4.2 Running the performance Test

The first test was the validation of the optimal time of 56ms, according to the acceptance criteria, in which it was decided to test a common USB keyboard.

The following tests involved comparing the FAAST with the TeamBridge. For this, it was configured in both that the gesture of raising the hand above the head triggers a key for the test software. The tests with FAAST were performed in two scenarios: on a single machine and networked. In this case, a 100mbps-head network was used. The Figure 4 presents the arrangement performed in the test.

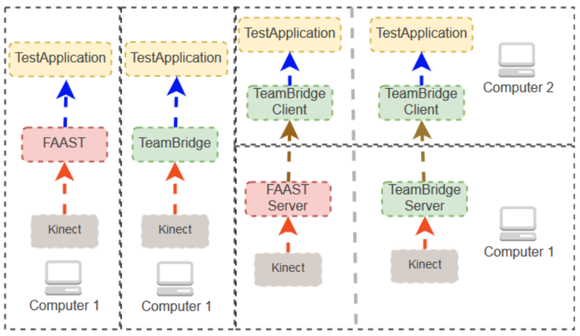

Figure 4. Organization of the comparative test with the FAAST.

First, we ran the test with the FAAST and then with the TeamBridge, because they can not be tested at the same time. Since FAAST cannot normally be used on a network, the client of the TeamBridge was used with the FAAST server, so that the network test was performed.

After being compared with the FAAST, other tests were carried out — local and on the network — combining devices. However, these tests could not be performed with the FAAST, since it does not have compatibility with other devices besides the Kinect. The Figure 5 presents a representation of the arrangement for these tests.

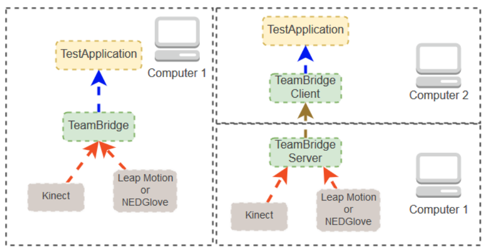

Figure 5. Arrangement of the combination test via Kinect and other devices.

In the end, the middleware developed in this work was also tested on a single machine, however with a server application for each device, in which a server was configured on the original port of VRPN, to the Kinect (v1 and v2), and a server for Leap Motion, on another port.

During the tests, it was observed that errors occurred intermittently, closing the application. After verification, it was found that the error occurred when the two devices tried to send messages at the same time. In order to resolve this issue, the VRPN code was modified by implementing a mutex or lock, which is used to prevent two processes or threads from simultaneously accessing a critical sector of the code (Marshall [1999]), the latest version of VRPN already has the fix for this problem. This technique can affect the performance of the application, as part of the code should wait for the other to continue, so tests were done to check the performance with and without this modification.

4.3 Tests results

The time measured by the keyboard was below the 56ms, with $32 \pm 3,97$ms, validating the criterion of acceptance. So most people will not notice a time difference between pressing a key and the software answer.

The Kinect v2 with SDK 2.0 was also tested. However, its performance was similar to the Kinect v1, with no statistical significance (p=0.9831), therefore, the authors decided that both devices will be referenced in this work only as “Kinect”.

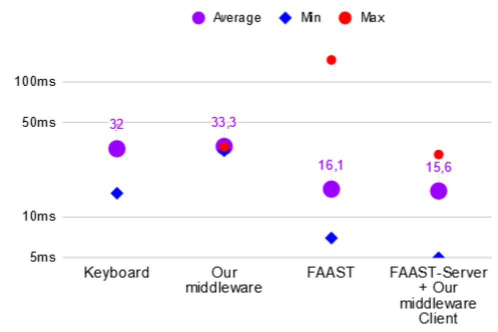

Figure 6. Load test comparing a keyboard with FAAST and with the TeamBridge.

In figure 6, it is possible to notice that the TeaTeamBridge 2.0 has an acceptable performance, providing inputs with a range of $33.3 \pm 5,37$ms, close to a keyboard and below 56ms. On the other hand, the FAAST showed an unexpected result, being able to deliver inputs twice faster than the USB keyboard. Due to this result, the test was repeated – maintaining the FAAST server – but with the interpreter (client) from the TeamBridge. According to the result, it was still possible to maintain the interval smaller than that of the keyboard, with no statistical significance (p=0,4802) in the scenario of pure FAAST against the FAAST server with the client of TeamBridge.

The Figure 7 shows the network test, and it is possible to observe that the standard deviation of the remote FAAST increased from $15.6 \pm 4,9$ms to $17.1 \pm 7,21$ms, with a statistical significance (p=0,0041). On the other hand, the developed middleware went from 33.3 $\pm 5, 37$ms to 33.4 $\pm 8, 29$ms, with no statistical significance (p=0,097).

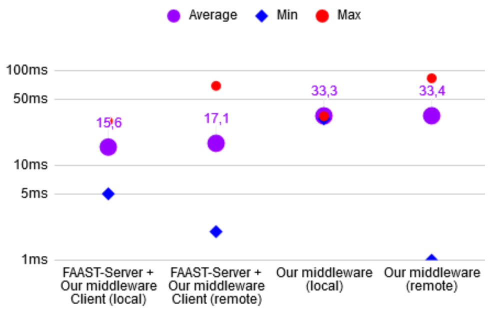

Figure 7. Load test comparing local and remote modes.

The Figure 8 shows the results of the Leap Motion and its combination with Kinect. It is possible to observe that the close hand gesture showed a much lower performance than the wrist flexion. The difference between these gestures is because the close hand gesture was reused from the Leap Motion SDK, however the wrist flexion was implemented in the TeamBridge.

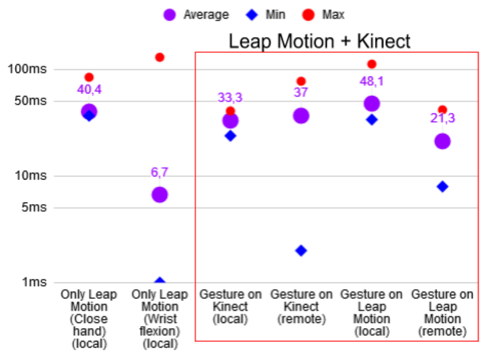

Figure 8. Load test using Leap Motion with Kinect.

Although the close hand gesture is slower, it is still within the 56ms acceptance criterion. The Figure 9 presents the results of the association between Kinect and NED Glove.

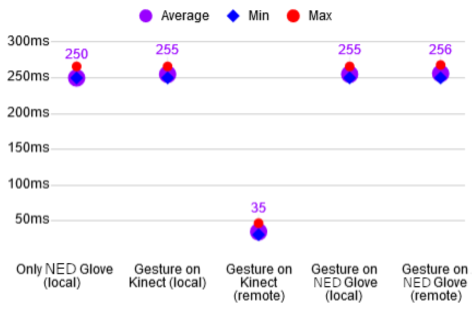

Figure 9. Load test using NED Glove with Kinect.

The NED Glove presented the time of 255 $\pm 3, 45$ms, well above of the acceptable. Thus, this test allowed the observation of this fact and the possibility of a future work in order to improve the performance of this device. However, it also brings a greater concern in the development of future devices for this requirement. It is important to notice that the identification of these problems was part of the objectives of this test.

In results of the test performed using Leap Motion with Kinect, it was observed that only the slowest device, Leap Motion, lost performance, shifting from 40.4 $\pm 3, 59$ms to 48.1 $\pm 10, 62$ms, with statistical significance (p=0,0001). The Kinect kept its time with no statistical significance (p=0,93). While, in the test with NED Glove there was a drastic decrease in the performance of Kinect, going from 33.3 $\pm 5, 37$ms to 255 $\pm 4, 88$ms. In this case, the slower device influenced considerably in the performance of the faster device, being the Response time determined by the slowest device.

The Figure 10 presents the tests carried out in relation to the mutex, in order to verify that it brings a considerable difference to the performance loss. Then, it can be inferred that the mutex does not impair the performance significantly. The problem of performance loss when using more than one device can be circumvented by using more than one server, both local or network. When networked, it was possible to observe that the TeamBridge operates with stability, so that in some cases it performs better than locally. Therefore, it is possible to connect a device on each computer.

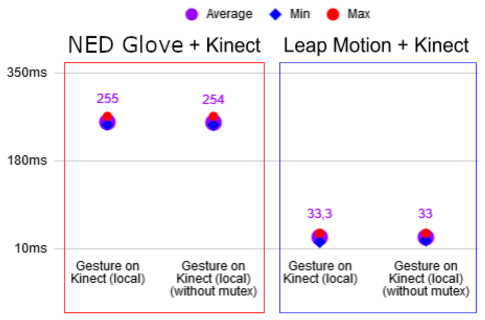

Figure 10. Load test of the devices with and without mutex.

However, it will not always be possible to have multiple computers available. In this case, the VRPN also allows the creation of more than one server on the same machine, just by modifying the port where the server will operate. The Figure **??** shows the same tests performed on the Kinect with Leap Motion, however with two server applications on the same machine. It can be observed that the Leap Motion was did not suffer any interference, since its average operating alone was 40.4 $\pm 3, 59$ms and passed to 41 $\pm 4, 26$ms, including no statistical significance (p=0,0502). The average Kinect remained in 33.3 $\pm 2, 2$ms, also without statistical significance (p=0,925).

4.4 Teambridge 2.0 integration with commercial games

To prove that the TeamBridge 2.0 developed in this work can adapt games without modification of source code, we perform a test with the commercial game, the Mount and Blade. We chose this game because it is incompatible with Kinect or any other VR device and allows the execution of several different commands, such as attack by the right, attack by the left, attack from above, defense with shield, walk, and others. With these commands, it was possible to make an analogy to the gestures performed via Kinect. In this experiment, we use both Kinect v1 and v2.

Since TeamBridge 2.0 captures the stationary walk, this gesture made the character walk. When the user rotates the body, the player's character also rotates. For the command defense with shield, the user needs to raise the left hand at chest height. The attack commands require the player to place the right hand on the side – left or right – of the body or on top of the head to define the direction of the hit. Then, the player must take his right hand in front of him at the chest height to do the hit. A demo is available at (https://www.youtube.com/watch?v=hQgXh6EPxDw).

4.5 Integration with exergames

Our research group developed some games for motor rehabilitation. Other team developers created these games (not the team who created Teambridge 2.0). These games were considered legacy games but were still functional.

One of the games is the exergame Fat Bird, an imitation of Flappy Bird. This game has compatibility only with Leap Motion and Physio Happy developed by Silva et al. [2020]. Its development took place before our middleware. Physio Happy captures wrist flexion. So, the player uses the wrist down or up while flexing to control the game. Leap Motion has native detection of opening and closing the hand, so with Leap Motion, the user needs to perform opening and closing movements. However, through TeamBridge 2.0, it was also possible to reproduce it via Kinect and Leap Motion using the wrist flexion gesture. It is important to note that wrist flexion is usually not identified by either the Leap Motion SDK or Kinect, and it is possible to use this configuration only through TeamBridge 2.0. We did not make changes to the game's source code.

The other game, "The Kayak Supremo" (Cassiano [2013]), was the only game with its source code modified. This modification was necessary because it was only possible to play it with NED Glove (Silva et al. [2013]). Therefore, the adaptation only allowed keyboard inputs, creating four commands with the keys Q, A, E, and D, where the keys Q and A take the Kayak to the left and the keys E and D take it to the right. However, the A and D keys make the Kayak move faster. This speed difference helps interpret when the glove is slightly or significantly closed.

The TeamBridge 2.0 captures gestures and configures them in a range of values. Therefore, the NED Glove was set to trigger the key A if the hand closing force is between 30 and 50. That is, when the hand is slightly closed, the Kayak goes to the left, but if the hand is completely closed, it moves faster. The configuration editor also allows testing the force so the person who is setting it up knows which track should insert. Just plug in the glove and click the "force test" button. With this, the glove captures the force and displays it on the screen.

Another project used the TeamBridge 2.0 successfully. This project involved 50 elderly patients and 11 experts for game validation. The project involved an exergame called Virtualter created for a study of Pacheco [2020]. The game uses Kinect (v1 and v2), but the game also accepts a keyboard as input (without any concern for Kinect).

The study was carried out within the guidelines of the Research Ethics Committee of the Federal University of Rio Grande do Norte (Opinion number 2.628.749) and following resolution 466/12 of the National Health Council. All participants involved in the research signed a Term of Free and Informed Consent - TCLE, authorizing their voluntary participation. Figure 11 shows a volunteer performing a test.

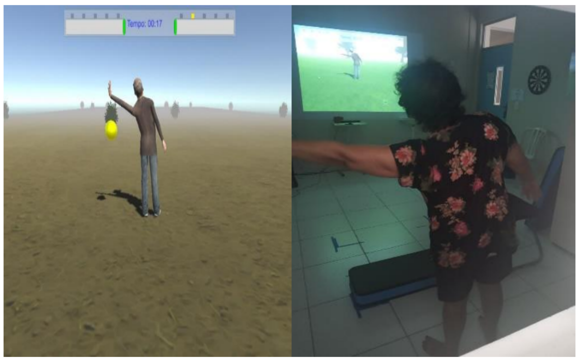

Figure 11. Patients using the Virtualter. Fonte: Pacheco [2020].

This game has a unique feature compared to the others. In it, the player controls his character, walking in stationary motion. Stationary walking is flexing and stretching the knees alternately while standing and without bending the trunk forward. In other words, it's simulating walking without leaving where it is. This type of movement is helpful for a semi-immersive Virtual Reality simulation, which is the case of the Virtualter game. The Kinect v1 and v2 models can perceive this stationary motion. So, it is acceptable for the project or TeamBridge 2.0.

The game had a second requirement: the player would find steps in the game, and the character should be able to climb those steps. A camera must capture the player's movement in the real world. The game environment needs a gym step (1 or 2 equipment of this type). The player must climb and descend the steps in the real world for his character to replicate this movement in the game. The game was proposed for balance training in older adults, and this movement of climbing and descending the steps is essential for analyzing postural balance.

During testing with Kinect, some limitations were observed. The device did not capture the data correctly when the player descended or climbed the step. When walking in front of the Kinect, we identified that the user's height changed as he approached or distanced from Kinect, and this would hinder TeamBridge 2.0 from placing when the user climbed or descended the ladder. Because of this problem, our goal has become to find a region where this distortion is minimal.

4.6 Device merging

Unlike FAAST, our middleware can work with more than one input device at the same time. The other studies did not observe this possibility.

It is interesting working with two devices simultaneously. One can supplement the deficiencies of the other. Kinect has already been used in conjunction with a smart glove (Huang et al. [2012]), precisely because the Kinect v1 cannot capture hands, so the smart glove meets this need.

Figure 12 is part of a video showing Kinect being used with Leap Motion. The TeamBridge identifies when the user closes the hand to hold the object and sends the signals to the application that cause the object to be carried by the user. This is also an example of how you could use TeamBridge with VR, as this application created in Unity receives the Tracker data structure through a VRPN plugin for Unity. So, Kinect sends the data to Unity through VRPN via the TeamBridge server. Unity updates the skeleton pose according to the data, in addition to receiving mouse inputs in place of Leap Motion inputs.

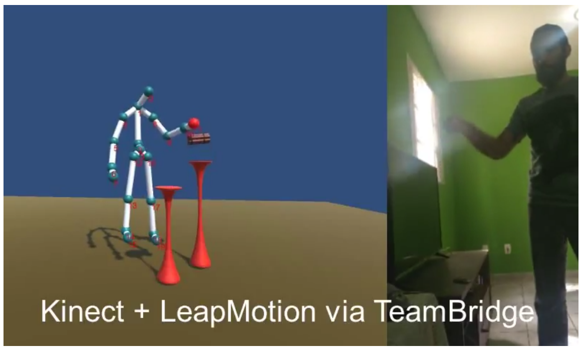

Figure 12. Kinect union with Leap Motion via TeamBridge.

4.7 Control of robotic equipment

In another experiment, we evaluate the possible use of TeamBridge to allow the control of robotic equipment through sensors such as Kinect; with this, the project would enter the area of telerobotics, also known as teleoperation, which occurs when the user is capable of remotely controlling robotic equipment according to Davies et al. [1995].

The main objective of this experiment is the conversion of Kinect data to Arduino. For this conversion, the communication channel, the interpretation of the data, the code, and the physical equipment for tests are necessary.

First, we define how the communication between the devices would be. TeamBridge collects and converts the information before sending it through the serial port to the Arduino. Such information understood which angle the engine should perform.

We developed a piece of equipment with Arduino Mega to execute the tests. The equipment received the angles and had two servo motors with a capacity of 15 kg, model msg99, and two voltage regulators. We use one servo to move the base joint with the arm. The other servo moves the wrist joint. Figure 13 represents the constructed device.

The architecture of TeamBridge made it necessary to create only a single class to communicate with the Arduino, the *ArduinoAction* shown in Figure 14. This class was created in the client part of the application and contains the method for connecting and sending messages to the Arduino.

We already implemented all the code for interpreting the

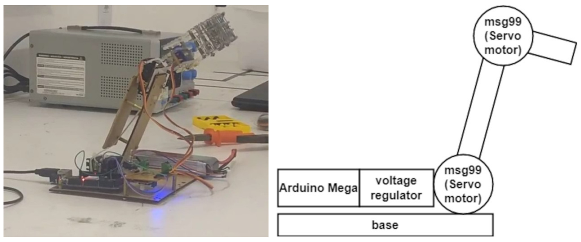

Figure 13. Arduino robotic arm

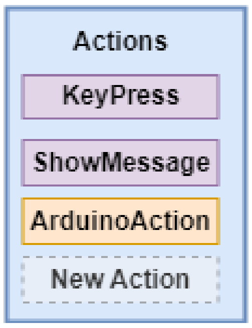

Figure 14. Representation of the place in the architecture where the class was created.

gestures in Kinect, and it was possible to reuse it. However, it was also necessary to create the implementation of capturing the angle between the points.

During the final test, we observed an inconsistency in the connection with the Arduino. This inconsistency blocked the Arduino after a few seconds of sending information. Supposedly, this crash is due to the amount of information transmitted. To solve this crash, we implemented a delay of 40ms in sending data to the Arduino. With this delay, the Arduino can operate for up to 7 minutes before crashing. When the crash occurs, it is necessary to reconnect the Arduino. Delays from 500ms to 10ms were tested, but with values below 40ms, the robotic arm behaved unexpectedly, executing sudden commands, and with very high values, the delay in activating the commands was noticeable.

5 Conclusion

With the TeamBridge, it was possible to obtain a mediation between the input devices and the games in a non-invasive way, taking into consideration that it was possible to reconcile the devices that were approached in this project, even with commercial games, there was no need to modify the source code. This also allowed that the production of exergames not require direct contact with such devices, so a programmer interested in building an exergame will not need to learn the functioning of the SDK. He will program so that the game works with keyboard or mouse, being only necessary that the person responsible for applying the exergames knows how to create a configuration file. The programmer will need to modify the source code of the TeamBridge only if he wants to include compatibility with a new device or to include a new feature or gesture.

With this work, it was also possible to generate an acceptance criterion, as well as performance tests for the input devices, an issue that was not taken into consideration, but that could hinder the player's experience. With such tests, it was possible to identify that NED Glove needs performance improvements. Also, it was identified that, although Leap Motion complies with the acceptance criterion, the modification of certain algorithms of this device may greatly increase its performance. Finally, the comparative test performed with FAAST identified that the TeamBridge has bottlenecks when operated with Kinect, but even with such performance loss, it is still within the acceptance criterion.

There were also two problems in the usage of two or more input devices, along with a need for a message flow control and a performance problem. However, the possible solution to both problems was also identified.

5.1 Future work

This work opens several opportunities for future work, such as:

- The use of machine learning, so the TeamBridge recognizes a new gesture.
- Creations of a smartphone driver and application in order to allow the use of the data generated on the device, making it a joystick.
- Creation of a speech interpreter to be able to use words like input.
- Provide input device data via a Web Service instead of triggering keyboard commands.
- Allow control of drones through TeamBridge.
- Implementation of a reverse data stream in order to enable the development of haptic devices or devices that generate some kind of feedback to the user. Currently, the data departs from the device to the TeamBridge, toward the application.

References

Bambach, S., Lee, S., Crandall, D. J., and Yu, C. (2015). Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *Proceedings of the IEEE international conference on computer vision*, pages 1949–1957.

Bianor, F., Cavalcanti, A., and Dantas, R. R. (2017a). Evaluate leap motion control for multiple hand posture recognition. *19th Symposium on Virtual and Augmented Reality*.

Bianor, F., Cavalcanti, A., and Dantas, R. R. (2017b). Physiohappy: A low cost device for virtual rehabilitation. *19th Symposium on Virtual and Augmented Reality*.

Borja, E. F., Lara, D. A., Quevedo, W. X., and Andaluz, V. H. (2018). Haptic stimulation glove for fine motor rehabilitation in virtual reality environments. In *International conference on augmented reality, Virtual Reality and Computer Graphics*, pages 211–229. Springer.

Cano Porras, D., Sharon, H., Inzelberg, R., Ziv-Ner, Y., Zeilig, G., and Plotnik, M. (2019). Advanced virtual reality-based rehabilitation of balance and gait in clinical practice. *Therapeutic advances in chronic disease*, 10:2040622319868379.

Cassiano, L. (2013). Kayak supremo. *Demos do SBGames 2013*. `http://www.sbgames.org/sbgames2013/festival/post.php?tag=game114` Accessed: 13 January 2024.

Conconi, A., Ganchev, T., Kocsis, O., Papadopoulos, G., Fernández-Aranda, F., and Jiménez-Murcia, S. (2008). Playmancer: A serious gaming 3d environment. In *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, pages 111–117. IEEE.

Corbetta, D., Imeri, F., and Gatti, R. (2015). Rehabilitation that incorporates virtual reality is more effective than standard rehabilitation for improving walking speed, balance and mobility after stroke: a systematic review. *Journal of physiotherapy*, 61(3):117–124.

Davies, B. R., McDonald Jr, M. J., and Harrigan, R. W. (1995). Virtual collaborative environments: programming and controlling robotic devices remotely. In *Telemanipulator and Telepresence Technologies*, volume 2351, pages 34–43. SPIE.

Drummond, D., Hadchouel, A., and Tesnière, A. (2017). Serious games for health: three steps forwards. *Advances in Simulation*, 2(1):1–8.

Fregnan, E., Baum, T., Palomba, F., and Bacchelli, A. (2019). A survey on software coupling relations and tools. *Information and Software Technology*, 107:159–178.

Garrett, B., Taverner, T., Gromala, D., Tao, G., Cordingley, E., Sun, C., et al. (2018). Virtual reality clinical research: promises and challenges. *JMIR serious games*, 6(4):e10839.

Huang, M.-C., Xu, W., Su, Y., Lange, B., Chang, C.-Y., and Sarrafzadeh, M. (2012). Smartglove for upper extremities rehabilitative gaming assessment. In *Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments*, page 20. ACM.

Iqbal, J., Tsagarakis, N. G., Fiorilla, A. E., and Caldwell, D. G. (2010). A portable rehabilitation device for the hand. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 3694–3697. IEEE.

Jones, S. E. and Thiruvathukal, G. K. (2012). *Codename revolution: the Nintendo Wii platform*. MIT Press.

Konstantinidis, E. I., Antoniou, P. E., Bamparopoulos, G., and Bamidis, P. D. (2015). A lightweight framework for transparent cross platform communication of controller data in ambient assisted living environments. *Information Sciences*, 300:124–139.

Lee, E.-S. and Shin, B.-S. (2021). A flexible input mapping system for next-generation virtual reality controllers. *Electronics*, 10(17):11.

Lu, A. S. and Kharrazi, H. (2018). A state-of-the-art systematic content analysis of games for health. *Games for health journal*, 7(1):1–15.

Lun, R. and Zhao, W. (2015). A survey of applications and human motion recognition with microsoft kinect. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(05):1555008.

Luo, D., Du, S., and Ikenaga, T. (2019). Multi-task and multi-level detection neural network based real-time 3d pose estimation. In *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1427–1434. IEEE.

Marshall, D. (1999). Further threads programming:synchronization. `https://users.cs.cf.ac.uk/dave/C/node31.html` Accessed: 13 January 2024.

Matamala-Gomez, M., Slater, M., and Sanchez-Vives, M. V. (2022). Impact of virtual embodiment and exercises on functional ability and range of motion in orthopedic rehabilitation. *Scientific reports*, 12(1):1–10.

Mathiowetz, V., Volland, G., Kashman, N., and Weber, K. (1985). Adult norms for the box and block test of manual dexterity. *American Journal of Occupational Therapy*, 39(6):386–391.

Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D., and Theobalt, C. (2017). Vnect: Real-time 3d human pose estimation with a single rgb camera. *Acm transactions on graphics (tog)*, 36(4):1–14.

Meier, J., Farre, C., Bansode, P., Barber, S., and Rea, D. (2007). *Performance Testing Guidance for Web Applications: Patterns & Practices*. Microsoft Press, Redmond, WA, USA.

Michotte, A. (2017). *The perception of causality*, volume 21. Routledge.

Milazzo, F., Gentile, V., Gentile, A., and Sorce, S. (2018). Kind-dama: A modular middleware for kinect-like device data management. *Software: Practice and Experience*, 48(1):141–160.

Mossel, A., Schönauer, C., Gerstweiler, G., and Kaufmann, H. (2013). Artifice-augmented reality framework for distributed collaboration. *International Journal of Virtual Reality*.

Mugueta-Aguinaga, I. and Garcia-Zapirain, B. (2017). Fred: Exergame to prevent dependence and functional deterioration associated with ageing. a pilot three-week randomized controlled clinical trial. *International journal of environmental research and public health*, 14(12):1439.

Oh, Y. and Yang, S. (2010). Defining exergames & exergaming. *Proceedings of Meaningful Play*, pages 1–17.

Pacheco, T. B. F. (2020). Desenvolvimento e usabilidade de um jogo digital para reabilitação do equilíbrio postural de idosos. In *Tese de Doutorado em Fisioterapia*, page 38. Universidade Federal do Rio Grande do Norte (UFRN).

Pandit, S., Tran, S., Gu, Y., Saraee, E., Jansen, F., Singh, S., Cao, S., Sadeghi, A., Shandelman, E., Ellis, T., et al. (2019). Exercisecheck: A scalable platform for remote physical therapy deployed as a hybrid desktop and web application. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, pages 101–109.

Park, D.-S., Lee, D.-G., Lee, K., and Lee, G. (2017). Effects of virtual reality training using xbox kinect on motor function in stroke survivors: a preliminary study. *Journal of Stroke and Cerebrovascular Diseases*, 26(10):2313–2319.

Pirovano, M., Lanzi, P. L., Mainetti, R., and Borghese, N. A. (2013). The design of a comprehensive game engine for rehabilitation. In *Games Innovation Conference (IGIC), 2013 IEEE International*, pages 209–215. IEEE.

Rose, T., Nam, C. S., and Chen, K. B. (2018). Immersion of virtual reality for rehabilitation-review. *Applied ergonomics*, 69:153–161.

Rybarczyk, Y., Luis Pérez Medina, J., Leconte, L., Jimenes, K., González, M., and Esparza, D. (2019). Implementation and assessment of an intelligent motor tele-rehabilitation platform. *Electronics*, 8(1):58.

Santos, L. H. d. O. (2016). Arcabouço para construção de jogos ubíquos com foco em reabilitação. In *Dissertação de Mestrado em Informática*, page 60. Universidade de Brasília (UnB).

Sartori, F. (2020). An api for wearable environments development and its application to mhealth field. *Sensors*, 20(21):5970.

Shen, J. and Pantic, M. (2009). A software framework for multimodal humancomputer interaction systems. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 2038–2045. IEEE.

Siddiqui, S. A., Snober, Y., Raza, S., Khan, F. M., and Syed, T. Q. (2015). Arm gesture recognition on microsoft kinectusinga hidden markov model-based representations of poses. In *2015 International Conference on Information and Communication Technologies (ICICT)*, pages 1–6. IEEE.

Silva, F. G. M., Bessa, N. P. O. S., Amaral, R. H. O., Pacheco, T. B. F., Medeiros, F. B. S., Nagem, D. A. P., Dantas, R. R., and Cavalcanti, F. A. d. C. (2020). Physiohappy interface in the treatment of wrist and hand of a child with cerebral palsy: case repot. *Research, Society and Development*, 9.

Silva, L., Dantas, R., Diniz, P., Jeronimo, V., Bueno, L., and Dutra, T. (2016). Phys. io: Wearable hand tracking device. In *Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), 2016 IEEE International Conference on*, pages 1–6. IEEE.

Silva, L., Dantas, R., Pantoja, A., and Pereira, A. (2013). Development of a low cost dataglove based on arduino for virtual reality applications. In *2013 IEEE International conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA)*, pages 55–59. IEEE.

Souza, C. H. R., de Oliveira, D. M., do Nascimento, D. F., de Oliveira Berretta, L., and de Carvalho, S. T. (2022). A serious games and game elements based approach for patient telerehabilitation contexts. *Journal on Interactive Systems*, 13(1):179–191.

Suma, E. A., Lange, B., Rizzo, A. S., Krum, D. M., and Bolas, M. (2011). Faast: The flexible action and articulated skeleton toolkit. In *Virtual Reality Conference (VR), 2011 IEEE*, pages 247–248. IEEE.

Takaiwa, M., Noritsugu, T., Ito, N., and Sasaki, D. (2011). Wrist rehabilitation device using pneumatic parallel manipulator based on emg signal. *Int. J. Autom. Technol.*, 5(4):472–477.

Taylor II, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. (2001). Vrpn: a device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61. ACM.

Teófilo, L. F., Nogueira, P. A., and Silva, P. B. (2013). Gemini: A generic multi-modal natural interface framework for videogames. In *Advances in Information Systems and Technologies*, pages 873–884. Springer.

Tieri, G., Morone, G., Paolucci, S., and Iosa, M. (2018). Virtual reality in cognitive and motor rehabilitation: facts, fiction and fallacies. *Expert review of medical devices*, 15(2):107–117.

Wainer, J. et al. (2007). Métodos de pesquisa quantitativa e qualitativa para a ciência da computação. *Atualização em informática*, 1:221–262.

Wang, Y., Ijaz, K., Yuan, D., and Calvo, R. A. (2020). Vr-rides: An object-oriented application framework for immersive virtual reality exergames. *Software: Practice and Experience*, 50(7):1305–1324.

Weichert, F., Bachmann, D., Rudak, B., and Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393.

Weiss, P. L., Keshner, E. A., and Levin, M. F. (2014). *Virtual reality for physical and motor rehabilitation*. Springer.