UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS BARBOSA DE FREITAS

EVALUATING KARATE-DO MOVEMENTS USING KINECT V2 CÂMERA AND TOOLS

CAMPINA GRANDE - PB
2020

MATHEUS BARBOSA DE FREITAS

EVALUATING KARATE-DO MOVEMENTS USING KINECT V2 CÂMERA AND TOOLS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Herman Martins Gomes.

CAMPINA GRANDE - PB

2020

F866e Freitas, Matheus Barbosa de.

Evaluating Karate-do movements using Kinect V2 camera and tools. / Matheus Barbosa de Freitas. – 2020.

13 f.

Orientador: Prof. Dr. Herman Martins Gomes.

Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Karate-do. 2. 3D motion analysis. 3. Machine learning. 4. Kinect V2 camera. 5. Algorithmic-based recognition. 6. Movimentos de Karate. 7. Ensino de karate por imagens. I. Gomes, Herman Martins. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

MATHEUS BARBOSA DE FREITAS

EVALUATING KARATE-DO MOVEMENTS USING KINECT V2 CÂMERA AND TOOLS

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Herman Martins Gomes
Orientador – UASC/CEEI/UFCG

Professor Dr. João Arthur Brunet Monteiro
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 2020.

CAMPINA GRANDE - PB

Evaluating Karate-do Movements using Kinect V2 Camera and Tools

Matheus Barbosa de Freitas
matheus.freitas@ccc.ufcg.edu.br
Universidade Federal de Campina Grande

Herman Martins Gomes
hmg@dsc.ufcg.edu.br
Universidade Federal de Campina Grande

ABSTRACT

Karate-do is a martial art of Japanese origin that has a wide variety of techniques. In Karate-do, practitioners must constantly train to perfect their movements. This can be accomplished during a karate class with the help of a teacher or through self-taught practice, done through consultation and repetition of the movements shown in books or instructional videos. However, there is some difficulty for the practitioner to define the efficiency of his/her movements in self-taught practice. In this work, we investigate the effectiveness of using a 3D capture device and machine learning algorithms aiming to detect and automatically evaluate and propose corrections to the movements of karate learners.

KEYWORDS

Karate-Do, 3D Motion Analysis, Machine Learning

1 INTRODUCTION

Karate-do is a secular martial art, whose main goals are to improve the character and practice efficient and effective self-defense techniques. With its origin traced on the Japanese island of Okinawa, karate-do is a mixture of fights native to that region with styles of Kung Fu practiced in China. Such a mixture led to the formation of various styles that were sometimes mixed in the formation of new ones in constant development, leading to some being more practiced than others [4].

For local political reasons, its practice was considered illegal until the middle of the 20th century, which also made it difficult to unify techniques in a single style of karate. Only in 1922, master Gichin Funakoshi made the first official demonstration of the art in the Japanese capital, from which modern karate was born [5]. The Shotokan style, created by Funakoshi, became popular throughout Japan and with the advent of World War II. all around the world. The importance of Shotokan in the dissemination of karate-do around the globe is notorious, being this one of the most practiced styles, we chose it to parameterize the techniques of this work.

Being a complex martial art composed of hundreds of different movements [8, 14], it is necessary an exhaustive practice on the part of its practitioners. In this way, karate practices occur through repetition of movements so that they can be improved with corrections made by a teacher.

In this sense, the self-taught practice of karate becomes difficult given that the practitioners, especially those with less experience, are unable to observe their mistakes and neither correct themselves to move correctly.

In this context, this work investigates the effectiveness of machine learning algorithms in the recognition of some movements captured through a 3D camera. However, recognition alone does not solve our problem, given that a way of correction and feedback is needed for practitioners of the movements learned by the algorithms, a concept also developed in the work.[1].

2 FUNDAMENTAL CONCEPTS

This section introduces general concepts related to the area of motion capture and recognition, its diverse approaches, and more specific concepts used directly in the development of this study. The purpose is to enable a better understanding of the later sections, in which we will use some of the concepts discussed here.

2.1 Motion Tracking Methods

The process of recognizing human movements and gestures has been the object of exploration by researchers since the 1970s [1]. Thus, the applications and methods for recognition, capture, and detection have been developed over the decades, resulting in different approaches.

In this process, the main objective is to understand what the movement represents, i.e., it's meaning. We can then classify human movements into **gestures** and **activities**. The former represents simpler movements, such as raising a hand, which has some meaning to another human being or machine. The latter consists of complex forms of movement, formed by various gestures and different body segments, as, for example, a karate technique.

Thus, to correctly identify a human movement, be it a gesture or an activity, it is necessary to extract and classify features of that movement. Currently, there are different configurations in terms of hardware or software to perform the extraction of those features [16].

Before feature extraction can happen, data acquisition must take place. To that end, optical approaches use cameras throughout the capture of the movement and may or may not use markers, active or passive, to facilitate the identification of joints. Markerless systems, however, require that the software perform all tasks of identifying and extracting the features of the movements, generally making the hardware cost considerably lower. Figure 1 contains an actor using a passive marker system for cinematic animation, while Figure 2 illustrates data acquisition using an active marker system.

On the other hand, non-optical approaches can be **mechanical**, **magnetic**, or **inertial**. They are usually constructed using, respectively, exoskeleton technology, receptors placed on the joints, and finally, sensors such as gyroscopes and accelerometers [1]. However, this type of approach will no longer be explored in this work

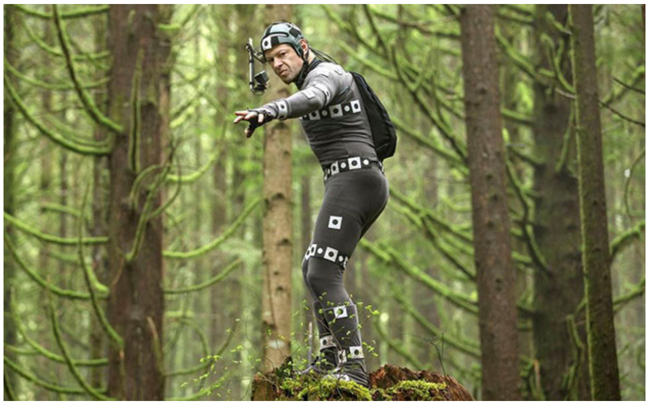


Figure 1: Passive Marker System. Source: Medium.

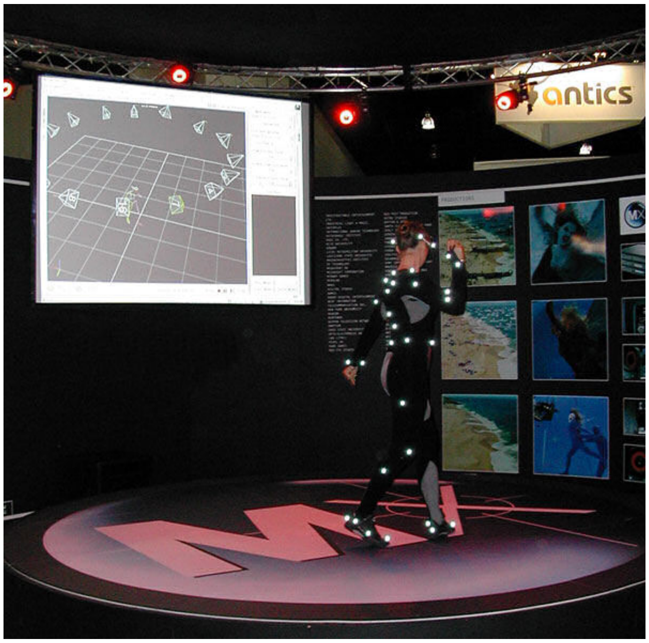


Figure 2: Active Marker System. Source: Wikipedia.

because we do not have access to this hardware. In Figure 3, there is an example of a non-optical system using a mechanical exoskeleton.

In this sense, approaches that may be optical or not, seek to distinguish the performer of the movement from the background of the image, extracting the directions and angles of the joints, which may be used to reconstruct, animate, simulate or analyze the human body movements. Figure 4 shows this study's author using a markerless system to capture movements and identify associated joints.

Considering that in this work, an equipment based on markerless technology was used, the next subsections address methods and techniques of software for recording and recognizing movements made by users while using this kind of approach.

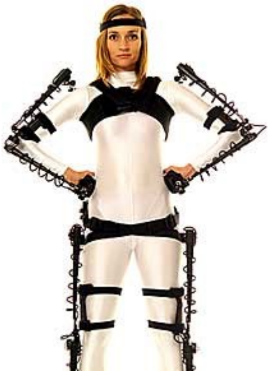
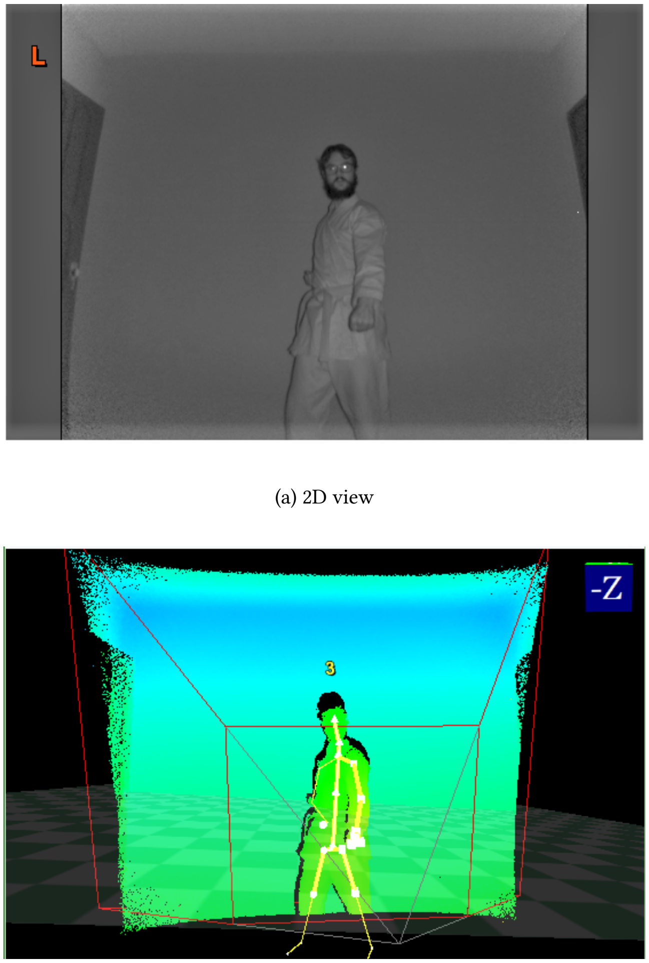


Figure 3: Exoskeleton Marker System. Source: Mocap.



(a) 2D view

(b) Joint extraction 3D view

Figure 4: Markerless System joint extraction of this work

2.2 Markerless Tracking

As previously stated, the use of markerless tracking methods while minimizing hardware costs makes the software solely responsible for extracting the main characteristics that make up the semantics of the movement. In this way, there are also different approaches in the context of software methods to be employed. Microsoft's Kinect for Windows V2 technology, the image capture equipment used in this work, is a markerless system with a 3D camera equipped with varied sensors (e.g, infrared and color), and offers us a range of options when choosing the path to follow when implementing the software responsible for the motion semantic classification [2, 11].

2.2.1 *Algorithmic-based Recognition.*

This type of approach consists of a set of rules defined directly in the code to classify and recognize movements. Although it allows an efficient (fast and accurate) classification of movements, in terms of software project development, it can difficult to reuse code between different application scenarios, since all recognition is hardcoded. In this sense, the evolution of the software with the addition of new movements becomes extremely costly, due to the need for well-defined rules to define which features are most important and their lower and upper limits for each movement [16].

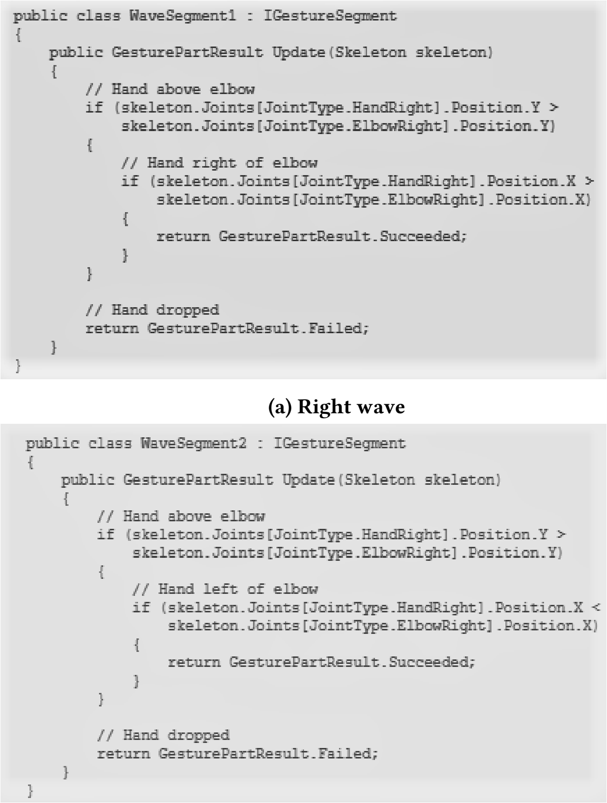
Thus, it is mainly used in applications related to rehabilitation exercises and games. In therapeutic use, for example, movements are usually performed numerous times by a patient accompanied by a professional. In this context, it is observed that the movements are well defined within a small scope, making it possible to physically replace the physical therapist with software based on rules [17, 18].

In this way, it is possible to notice the similar characteristics between the previous application and games in which a system needs to analyze and recognize a well-defined set of movements, in real-time, and to judge inferring from rules, actions to be performed. Two snippets of code in C# are presented next, where there is the identification, through the algorithmic approach, of a wave gesture. The Figure 5a identifies whether the right hand is above and to the right of the right elbow; whereas the Figure 5b identifies whether the right hand is above and to the left of the right elbow. When identified in sequence, these segments form a wave gesture using the right hand.

2.2.2 *Dynamic Time Warping Recognition.*

Dynamic time warping (DTW), is a method used to calculate the similarity between two temporal sequences that can vary in time and speed, being used for a long time for several types of applications that use speech recognition, movements, and data mining. Therefore, its main characteristic is the direct comparison between two different sets of data, usually in temporal aspects, and the consequent analysis of the similarity between them.

In general, this method converts reference captured data into temporal sequences to produce patterns that can be used to calculate the shortest distance between these reference sequences and test sequences aiming at correct classification of these. In speech recognition, for example, the sound waves captured from a continuous speech are directly compared to preprocessed word patterns. This helps to recognize words in different accents or at a different speed.



```
public class WaveSegment1 : IGestureSegment
{
    public GesturePartResult Update(Skeleton skeleton)
    {
        // Hand above elbow
        if (skeleton.Joints[JointType.HandRight].Position.Y >
            skeleton.Joints[JointType.ElbowRight].Position.Y)
        {
            // Hand right of elbow
            if (skeleton.Joints[JointType.HandRight].Position.X >
                skeleton.Joints[JointType.ElbowRight].Position.X)
            {
                return GesturePartResult.Succeeded;
            }
        }

        // Hand dropped
        return GesturePartResult.Failed;
    }
}
```

(a) Right wave

```
public class WaveSegment2 : IGestureSegment
{
    public GesturePartResult Update(Skeleton skeleton)
    {
        // Hand above elbow
        if (skeleton.Joints[JointType.HandRight].Position.Y >
            skeleton.Joints[JointType.ElbowRight].Position.Y)
        {
            // Hand left of elbow
            if (skeleton.Joints[JointType.HandRight].Position.X <
                skeleton.Joints[JointType.ElbowRight].Position.X)
            {
                return GesturePartResult.Succeeded;
            }
        }

        // Hand dropped
        return GesturePartResult.Failed;
    }
}
```

(b) Left wave

Figure 5: Algorithmic Code Snippet.

The DTW method computes the shortest distance between two sequences. Lets suppose S is the test data sequence and T is the reference data sequence, as shown in Equations 1 and 2:

$$S = s_1, s_2, ...s_n \tag{1}$$

$$T = t_1, t_2, ...t_m \tag{2}$$

These sequences can be arranged in such a way that they form a $n \times m$ plane, in which each point on the grid, (i, j), is an alignment between s_i and t_j. In this way, we will have a new sequence W (see equation 3 that maps the elements of S and T in such a way that the distance between them is minimized. That is, each w_k corresponds to a point $(i, j)_k$.

$$W = w_1, w_2, ...w_k \tag{3}$$

Finally, we define the problem as a minimization based on the cumulative distance for each path (see Equation 4), where δ is a measure of the distance between two distinct time sequences, which in turn can be calculated using the magnitude of the difference or the square of the difference.

$$DTW(S, T) = min_w \left[\sum_{k=1}^{p} \delta(w_k) \right] \tag{4}$$

Thus, in the context of movement classification, these two sequences S e T may be constructed from the orientation angles of the bones at each frame analyzed, acquired during the execution of the movement.

However, because it is based entirely on the comparison and focuses on removing temporal and speed differences, it is necessary to achieve great accuracy in the calculation of similarity when talking about motion comparison. As a result of the above, building a database becomes a complex task, given that in the real world the same actions can have slight differences between each time they are executed [9].

2.2.3 *Machine Learning.*

Machine learning is, in short, the science of making computers perform tasks without being explicitly programmed. That is, unlike traditional algorithms in which all the steps necessary for execution are given by the code, machine learning algorithms have only a part of the information they need to do, the model. The other part of the information is given by the data that are used in conjunction with the model. Thus, according to the data that is used, the machine learning model can complete the task for which it was trained [6].

Although it uses complex statistical and computational models, such as artificial neural networks and decision forests, the Kinect V2 technology allows us to use machine learning in a simple and direct way. Programming effort can be reduced through the **Kinect Studio (KS)** and **Visual Gesture Builder (VGB)** tools [15]. The first allows the use of the Kinect 3D camera to record the movements from which we want to build gestures or activities. In turn, VGB allows us to train and test movements using machine learning algorithms, such as Adaptive Boosting (AdaBoost) [10] and Random Forest Regression (RFR) [12].

Given that Machine Learning techniques were used in this work, we will go deeper into the explanation of it and of these last two algorithms available for use through the Kinect V2 tools.

Both algorithms, AdaBoost and RFR, make use of the ensemble method technique, which combines the results of multiple machine learning algorithms to make one with greater accuracy than an individual model could achieve. In this way, we can differentiate this technique in two types, boosting and bagging.

Boosting refers to the family of algorithms that improve on the results of weaker models. On the other hand, bagging or bootstrap aggregation can be defined as a random generation of samples with replacement. Typically used in decision trees, this type of technique uses the independent execution of models to ultimately aggregate the outputs of each one of them randomly.

AdaBoost is a supervised boosting algorithm that helps to capture non-linear relationships between characteristics. At each training phase, for each classifier, it determines the weight of each training item, compares it with the database, and if it is wrongly classified, increases its weight in order to ensure that items that are more difficult to classify remain in the model. After training each classifier, a weight is also assigned to it according to its accuracy in classifying training items. Consequently, a classifier with greater accuracy has greater weight and a greater impact on the final result of the model.

On the other hand, RFR is a supervised bagging algorithm in which trees are executed in parallel without any interaction between them. It works by building several trees that individually return a regression, i.e. a prediction of a certain quantity, according to the purpose for which the algorithm is being used. These results found individually by the trees are then aggregated through the model into a single ensemble model, a forest that performs better classifications than the individual models of each tree.

3 RELATED WORK

The recognition of human movements through computer systems is the object of study by researchers and companies for a long period of time, which gives us access to a large collection of articles and works related to this area. The use of Kinect, in particular, has also been the subject of study by researchers interested in this type of system, and since its launch, numerous articles have been published to measure the efficiency of this equipment in the process of capturing and recognizing movements.

This section aims to present an introduction to some of these previously published works, which were essential in supporting this undergraduate thesis. In this sense, the works reviewed here were chosen because they use the same equipment and method in the acquisition of movements as those used in this thesis. The research project entitled “Motion Comparison using Microsoft Kinect” [3] is a seminal work in the area, being referenced in most of the other works organized chronologically below [1–3, 13, 18].

Hemed Ali [3] proposes a method of comparing human movements between predetermined sequences captured in real-time using Kinect V1. For efficiency purposes, Ali argues that the predetermined movements should be serialized in a binary data structure, saving the skeleton frames in a single collection with all the information captured by Kinect, from which a new collection is created with the angles between the joints, calculated through the spatial coordinates of each of these.

The author also describes the performance of tests that demonstrate that the expected joint angles and those obtained have a small variation that must be considered with a threshold of +/- 20 degrees. In addition, other tests were carried out to check the efficiency of the algorithm with users of different bone lengths, different placements of the user in relation to the Kinect sensor, etc. These tests showed that even with the 20-degree tolerance mentioned above, it was not possible to obtain maximum accuracy due to some factors such as drop-in the acquisition frame due to host computer limitations, different durations for the same movement category, and natural body variations from person to person.

In [13], the authors propose a resolution via supervised learning to classify movements captured via Kinect. To build this system, the authors used a variation of the support vector machine (SVM) algorithm, to classify static poses that in turn compose a gesture. In this way, decision trees are built in such a way that each node represents a static position identified by the SVM and a path from a leaf to root represents a gesture. Therefore, a tree represents all the gestures that end with the static position represented by the root.

Thus, the algorithm works in such a way that when capturing an executed gesture and storing it in a buffer, all existing trees are searched for the one with the root equal to the last position stored in the buffer from which the nodes are navigated. underlying. Since a path is found to a leaf, a gesture is already trained by the algorithm, otherwise, it means that some of the executed static positions do not belong to the tree, so the gesture is not recognizable. The authors also included treatment for gestures performed at different speeds by adding a time vector to each node so that the same poses performed at different speeds are classified as different gestures.

The system experimented with eleven people, of whom one trained the algorithm and ten others tested it, with eighteen static positions that made up ten different gestures. Most gestures had a recognition rate above 80% and static positions had a hit rate almost always above 90%.

Hesham Alabbasi et al. [2] describe a system that recognizes predetermined movements in real-time, capable of providing some feedback to its users. In this work, the authors use the Unreal Engine 4 (UE4) [7], an open-source graphics engine that through a plugin provides simple and efficient integration of the Kinect V2 SDK, as well as the possibility of using all 25 joints captured by this equipment. Using this tool, the authors managed to build interactive software that allowed the recording of gestures and movements and their reproduction in 3D avatars.

The real-time feedback presented by Alabassi is limited to coloring the avatar in red when a movement is executed in the wrong way. However, this way of feedback is modified to highlight the muscle group that does not match the expected gesture in [1].

Still in [2], the authors used two methods to compare movements, one of which compares the angles between joints described in [3] and the other, simpler, is the direct comparison of joint orientation. Thus, two groups of tests are performed with two different users for each of the comparison methods, the model user, who recorded the exercises to be used as a model and a user without much sports background. It is concluded that the use of the joint angle comparison method, although it has similar results to the other method in the comparison of simple movements, in complex movements reaches a significantly greater accuracy.

Zhao et al. [18] create an XML structure based on rules, using Kinect technology, to build a system capable of guiding in real-time the execution of therapeutic movements. These rules, which make up the execution of an exercise, can be divided into three different categories, dynamic rules that define sequences of key positions for each segment of the body; static rules for defining body segments that must remain stationary; and invariant rules to describe the conditions that each body segment must meet. In turn, these rules can be formed in the most diverse ways, either by calculating the angle between the joints of two different segments of the body, in terms of orientation in relation to the anatomical planes or even the position of a certain joint in relation to another.

The rules for comparing movements used by the authors, although they have already been discussed throughout this graduation thesis, are structured in order to facilitate the extensibility and legibility of the rules created and are similar to an algorithmic approach explained previously. The feedback provided to users in this system happens during the execution of a certain movement, pointing out, if any, the violations of the rules defined for the gesture being performed. For a sequence of repetitions of a specific movement, a common situation in cases of rehabilitation exercises, it is necessary to use one or more finite state machines where each state is formed by one or more rules, which is able to identify the beginning and the end of the performed repetition.

The results obtained by the authors with the experimentation of the system using three types of simple rehabilitation exercise with eight healthy human participants demonstrates the need already observed in other studies to take into account a tolerance threshold between the values expected by the system and those obtained by the user, as well as the effectiveness of this type of system in guiding the correct execution of movements.

After this quick review, it is possible to conclude that the classification and recognition of gestures through Kinect is efficient in the most diverse contexts and methods. That said, it is clear that although the final objective of this thesis and the reviewed works are similar, that of movement recognition, the context, and method used in this thesis make experiments necessary to prove effectiveness and efficiency in this sense. Table 1 presents a summary of the papers reviewed according to the following criteria: kinect version, purpose of the applications, which kind of software approach was used, context in which it is inserted, characteristics of the movement evaluations, and dataset.

4 METHODOLOGY

This section aims to present the tools and data used to evaluate the algorithms and methods previously discussed in the identification and measurement of the quality of karate movements.

4.1 Tools

Kinect V2 was used to acquire the movements in conjunction with Kinect Studio, a tool that allows the recording, editing and reproduction of the captured data. This data can be accessed separately through what the API defines as Data Sources. Tables 2 and 3 below summarize respectively the physical characteristics of the second version of kinect and the types of data we can work with.

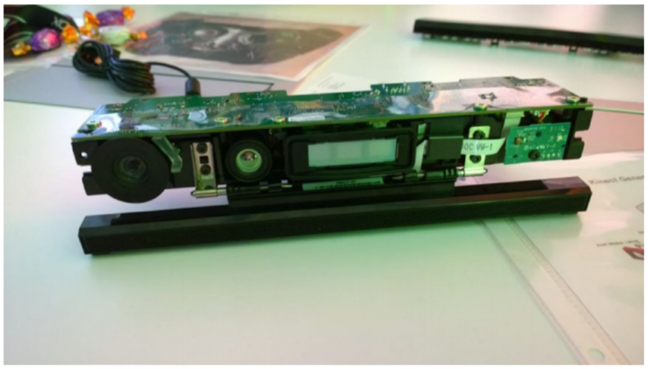


Figure 6: Kinect without plastic case. Source: [15]

In addition to these two tools, Visual Gesture Builder (VGB) was also used to create **discrete gestures**, based on the AdaBoost algorithm, a binary classifier that indicates whether the gesture is happening or not, and **continuous gestures**, which relies on RFRProgress that allows us to identify at what stage the movement is in the interval [0, 1], where 0 informs that the movement has not

Table 1: Summary of the reviewed work on the use of Kinect sensor for human motion recognition.

	Hemed Ali(2012)[3]	Miranda et al.(2012)[13]	Alabbasi et al.(2015)[2]	Zhao et al.(2017)[18]
Kinect Version	V1	V1	V2	V2
Purpose	Motion comparison	Motion comparison and user feedback	Motion comparison and user feedback	Motion comparison and user feedback
Kind of approach	Algorithmic	Machine Learning	Algorithmic	Algorithmic
Application context	Prospecting research with application in physical training or rehabilitation	Prospecting research with application in physical training or rehabilitation	Rehabilitation and physical training	Rehabilitation
Type of evaluation	movements recognized or not, direct comparison of obtained and expected angles	key poses recognized, the recognition rate of gestures, comparison with results of similar movements in other surveys	ratio between expected and obtained angles using two types of comparison algorithms (angle between joints and rotation between joints)	direct comparison of obtained and expected angles
Dataset	Proprietary, unclear how many users, simple movements (i.e.raising arm)	Proprietary, ten users plus a trainer, eighteen key poses, ten gestures.	Proprietary, two users (trainer and tester), six simple movements, five complex movements.	Proprietary, eight users with different body lengths, tree movements.

Table 2: Summary of Kinect V2 hardware characteristics. Source:[15]

Color Camera	1920 x 1080 x 16 bit per pixel 16:9 YUY2 @ 30 Hz (15 Hz in low light, HD)
Depth Camera	512 x 424 x 16 bits per pixel 16-bit ToF depth sensor
Range	0.5m to 8m (1.6 ft.–26.2 ft.) Quality degrades after 4.5m (14.7 ft.)
Angular Field View	70° Horizontal – 60° Vertical
Audio	16-bit per channel with 48 kHz sampling rate
Skeletal Joints	25 joints tracked
Skeletons Tracked	6
Vertical Adjustment	Manual, ±27 degrees of freedom
Latency	~50ms
USB	3.0

started and 1 indicates its end. In this sense, the VGB uses the data previously acquired through Kinect Studio to train a movement recognition algorithm through "tags" placed manually by the user that indicate where the movements take place.

The Figure 7 illustrates the tagging process where the red numbered circles were placed to better identify important areas in the process. Thus, we can identify in the areas 1 and 2 the views of the videos in two and three dimensions, respectively. In 3 we can see for a given frame, the values assumed by the gestures of that solution, and in 4 we can walk through the video, frame by frame, assigning these values.

This tool also allows the testing of the algorithm through a real-time view or through analysis of acquired data, generating a file that allows us to see the specific frames in which the movements were identified. This identification occurs, in discrete gestures, when there is at least one frame in which the neural network has 50% or more confidence that the movement has in fact occurred. For continuous gestures, the confidence rate is constant throughout the solution, so that we manually analyze whether the movement progress indicated by the VGB was the same as the one expected by the author.

4.2 Dataset

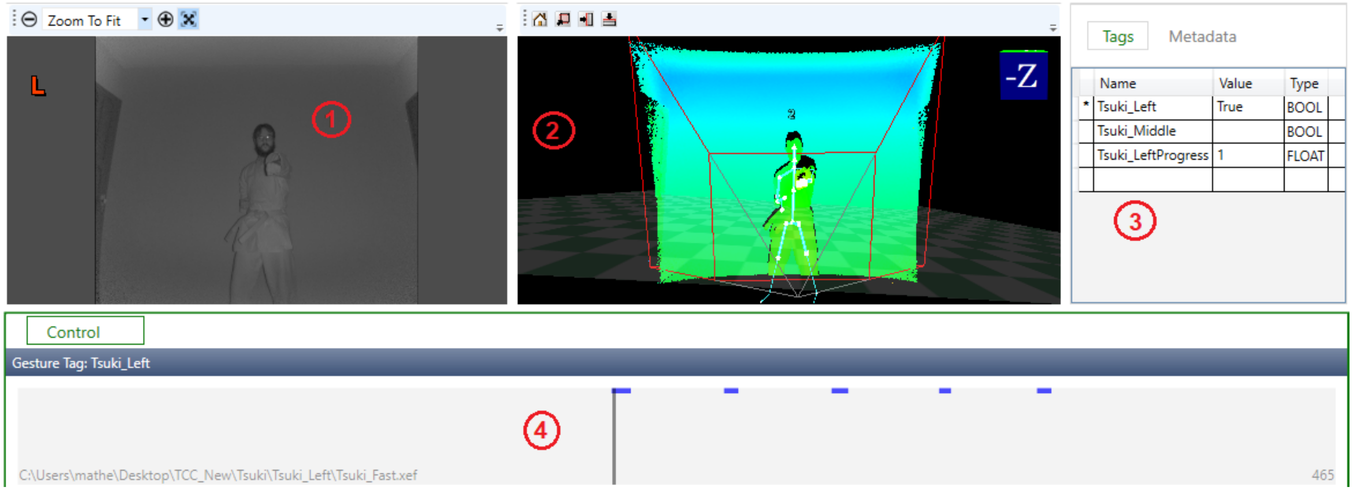
For the evaluation of the methods used in this work, we divided the data set in training and testing, in which seven different techniques of Karate's Shotokan style were analyzed, being an arm and a frontal leg strike, and five defenses. Figure 8 shows the frames of the final positions for each of the techniques in this data set. It is important to highlight that the defenses Uchi Uke and Soto Uke, have almost identical final positions but a different execution, making it necessary to test both.

These techniques were acquired at different execution speeds that we differentiate as slow and fast executions. The **training** techniques were recorded by the author of the work, a karate practitioner for a decade, and the **test** techniques were performed by black belts, being 3 adults and a 12 year old child. It is important to highlight that one of the adults in the test phase did not perform the frontal kick as he had a certain disability, having one leg bigger than the other by a few centimeters, a factor considered important to measure how the algorithm will behave when there are asymmetries in this sense.

For **training**, all techniques had ten repetitions recorded for both sides, right or left. In turn, these were divided equally between fast and slow executions. Similarly, the techniques recorded for **testing** were also acquired so that there were at about 10 repetitions, divided

Table 3: Summary of Kinect V2 API Data Sources. Source:[15]

Audio Source	Supplies multi-directional audio from the Kinect's microphone array bar.
Body Frame Source	Exposes all data about humans in view of the Kinect sensor. Provides skeletal joint coordinates and orientations for up to six individuals.
Body Index Frame Source	Yields information on whether a pixel corresponding to a depth image contains a player.
Color Frame Source	Provides image data from the Kinect's 1080p HD wide-angle camera. Can be accessed in multiple color formats, such as RGB and YUV.
Depth Frame Source	Provides depth data derived from the Kinect's depth camera. Depth distance is given in millimeters from the camera plane to the nearest object at a particular pixel coordinate.
Infrared Frame Source	Exposes an infrared image from the Kinect's 512 x 424 pixel time-of-flight(ToF) camera.
Long Exposure Frame Source	Enables long-exposure infrared photography using the same ToF infrared sensor as Infrared Frame Source.
Face Frame Source	Provides recognition of five points on a face in two dimensions(X and Y coordinates).
High Definition Face Frame Source	Provides recognition of 36 standard facial points and over 600 more vertices of non-standard facial points in three dimensions(X, Y, and Z coordinates).



	Name	Value	Type
*	Tsuki_Left	True	BOOL
	Tsuki_Middle		BOOL
	Tsuki_LeftProgress	1	FLOAT

Gesture Tag: Tsuki_Left

C:\Users\mathe\Desktop\TCC_New\Tsuki\Tsuki_Left\Tsuki_Fast.xef

Figure 7: Visual Gesture Builder Overview.

equally between the execution speeds, with slight differences in left or right executions quantities.

4.3 Building the experiment

For the construction of the experiment, each technique analyzed was built separately between the execution for the left and right side, so that it is possible to distinguish them. In this sense, each technique is trained using the two types of gestures, discrete and continuous, available at VGB software tool.

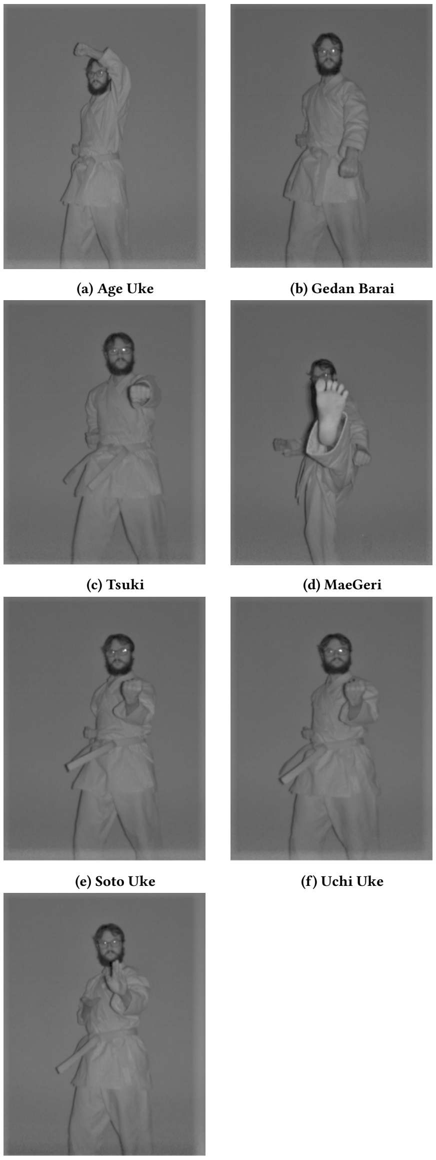
Discrete gestures are used, in each solution, to identify in a binary way if the technique is being performed or not, or if it is in the middle of its execution. These binary classifications are necessary so that in the creation of a continuous gesture, for a given technique, it is possible to identify stages of the movement, creating a path to be followed by those who perform it. In this way, through the joining of these gestures, we can identify if there were deviations in the execution of the technique and identify the quality of it.

Figure 9 is an approximate view of area 4 seen in Figure 7. While in Figure 7 we observe a discrete tagging process, which can only assume binary values, in Figure 9, we have the process of tagging a continuous gesture which is based on two discrete gestures identified by the red markings 1 and 2. In 3, we have a vertical bar that indicates which frame of the video was being viewed, at the moment the image was captured, that is, on the x-axis we can move through the frames and on the y-axis we have the progress of the gesture. Note that the progress of the gesture to the left comes to 0 when the movement is made to the right side or is not done at all.

Figure 10 is the solution project using the VGB for the identification of the left punch technique. In this, there is the presence of 6 different projects - the ones that have ended with an ".a" extension are exclusive for testing and are based on those with the same name (but without extension), which are exclusive for training and where the gesture identification is built. All other solutions in this work follow exactly the same pattern.

5 EXPERIMENTAL EVALUATION

This section aims to present the test data and conclusions. Of the three gestures used to build the neural network, in each of the



(a) Age Uke

(b) Gedan Barai

(c) Tsuki

(d) MaeGeri

(e) Soto Uke

(f) Uchi Uke

(g) Shuto Uke

Figure 8: Techniques on dataset.

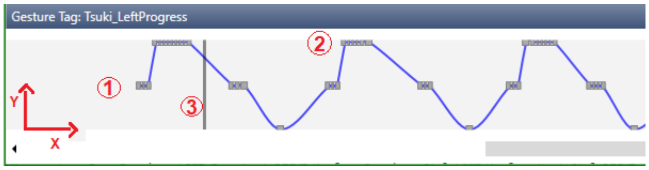
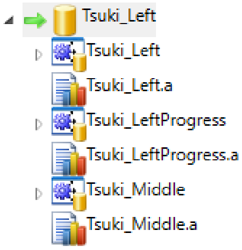


Figure 9: Continuous gesture tagging.

techniques, we had satisfactory results in only two of them, the discrete gesture that exactly defines the final position of each of the techniques, and the continuous gesture that analyzes the mapping capacity of the artificial neural network of tracking the progress of a technique.



Tsuki_Left
Tsuki_Left
Tsuki_Left.a
Tsuki_LeftProgress
Tsuki_LeftProgress.a
Tsuki_Middle
Tsuki_Middle.a

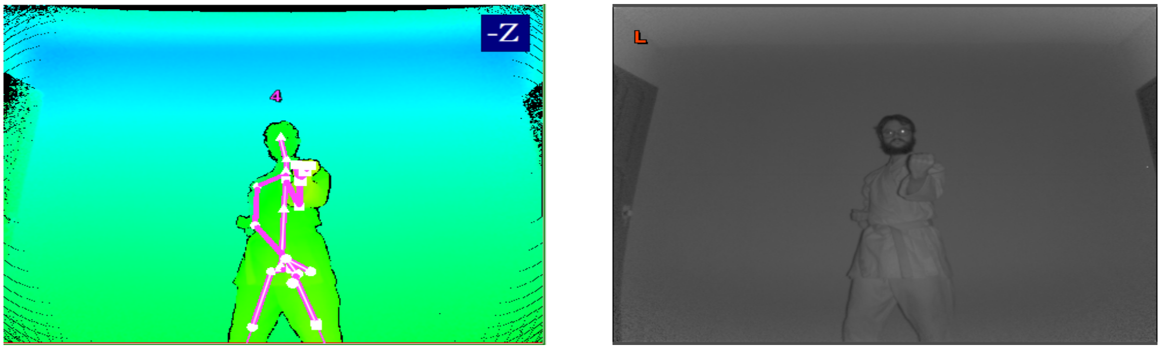
Figure 10: Solution Example.

Tables 4 and 5 below are organized in such a way that the side (left or right) and speed (fast or slow) of execution of the tests are represented in the lines. The columns contain the names of the techniques performed. The values within the intersecting cells are, in turn, the number of techniques correctly identified in the test set, meaning that the neural network had 50% or more confidence in any of the frames in which the gesture takes place, divided by the total number of executions.

In Table 4 it is observed that, with the exception of Mae Geri (frontal kick) and Uchi Uke (inside-out defense), all techniques achieved a correctness score better than or equal to 55%. We emphasize that the poor results of these two techniques follow what was observed during the acquisition of movements, where the skeleton built by kinect was sometimes incorrect as shown in Figure 11.

Table 5 presents the test results of the discrete gesture responsible for identifying the position referring to half of the technique's execution, and aims to assist the continuous gesture in the construction of the progress path of each movement. However, the individual tests of this gesture showed very low correctness scores, less than or equal to 30% in most techniques. In one of the test cases of the Uchi Uke technique, several false positives were identified in one of the slow runs to the right, something that did not occur with the other gestures.

Table 6 is organized in a similar way to Tables 4 and 5, but its cells are filled in such a way that they present the number of movements and the progress range in which they are, according to the testing results. This progress, from 0 when the movement is not happening, and 100 when the movement has reached its end, is also informed frame by frame by the VGB, and tells us how the neural network is recognizing the execution of the movement. In this way, we manually identify in which frame we consider the progress should be maximum and what was the correspondent value assigned by the network for that movement execution. For better understanding, in the "Slow Left" Age Uke tests, the algorithm identified the maximum progress of 6 executions as being 90% or more, when we considered that they should be 100% progress. In the last line is the percentage of executions of each technique in each interval identified by the VGB, which should have been considered as 100% according to our manual analysis. It is important to note that none of the tests obtained maximum results in progress, a characteristic expected since it was indicated to us by VGB that the confidence of the network in our continuous gesture is only 10%, and as already



(a) Skeleton built

(b) 2D image

Figure 11: Incorrect skeleton

Table 4: Summary of test data for the discrete gesture that identifies the final position of each technique.

	Age Uke	Gedan Barai	Mae Geri	Shuto Uke	Soto Uke	Tsuki	Uchi Uke
Slow Left	4/10	9/9	1/8	5/8	5/9	12/12	2/10
Fast Left	6/11	6/11	2/4	4/11	7/11	10/11	4/12
Slow Right	9/12	8/11	1/7	10/11	4/11	1/11	5/11
Fast Right	12/12	6/11	3/6	4/9	8/12	7/12	6/11
Total	31/45=68%	29/42=69%	7/25=28%	23/39=58%	24/43=55%	30/46=65%	17/44=38%

Table 5: Summary of test data for the discrete gesture that identifies the position that represents half of each technique.

	Age Uke	Gedan Barai	Mae Geri	Shuto Uke	Soto Uke	Tsuki	Uchi Uke
Slow Left	3/9	9/9	1/7	2/9	5/9	1/9	8/9
Fast Left	1/10	8/10	0/4	0/11	1/11	8/9	10/10
Slow Right	3/9	11/11	2/7	7/11	3/9	0/10	7/7
Fast Right	2/9	7/8	2/6	3/9	1/9	0/11	8/9
Total	9/37=24%	35/38=92%	5/24=20%	12/40=30%	10/38=26%	9/39=23%	33/35=94%

discussed, the identical copy of a movement is a near impossible task.

This continuous gesture gives us the possibility to recognize the progress of the techniques with a certain accuracy, but it is extremely efficient in detecting deviations in the movement path at any moment of its execution, allowing us to measure the quality of movement execution according to the progress level. This information could, in turn, be returned as feedback to the student so that he/she could try to improve that particular movement.

6 FINAL CONSIDERATIONS

The motion detection, tracking and classification techniques used in this work proved to be effective in exploring ways of automatically measuring the quality of karate movements.

We realized that in cases where it failed, there was a certain level of identification by the algorithm that the movement happened, but not enough that it could give a positive answer. It is possible that a larger and more variable amount of training data may considerably increase the accuracy rates of the algorithm, given that the physical differences in bone and mass can significantly influence the learned models. It is also important to note that even if executed by black belts and experienced karate practitioners, some of the acquired data could have a wrong execution of a technique.

The use of a gesture to identify half of the movement must be rethought either in the way our tags were placed or their own need. We were not able to see any significant difference in the identification of the techniques executed quickly and slowly, which leads us to believe that the training at both speeds had the expected results.

The tools available in the second version of kinect proved to be indispensable in the process, which would be much slower and more laborious if we had to develop the provided functionalities from scratch. However, as the equipment was discontinued, it became extremely rare to find any type of support or answer to questions, which ended up making it difficult at times to carry out this research. An important point that did not allow us to analyze more data was the difficulty of finding people qualified to execute the techniques due to the COVID-19 pandemic. In conclusion, with a good amount of data, using the equipment and methods analyzed here it is viable the construction of software that helps in the learning of karate, development of games that use similar concepts, and guided physical rehabilitation software.

Table 6: Summary of the test data of the continuous gesture that identifies the progress of each technique.

	Age Uke	Gedan Barai	Mae Geri	Shuto Uke	Soto Uke	Tsuki	Uchi Uke
Slow Left	6 >= 90 1 >= 70 2 >= 60	6 >= 90 3 >= 70	2 >= 90 5 >= 70	2 >= 90 4 >= 70 3 >= 50	5 >= 90 3 >= 70 1 >= 50	4 >= 90 8 >= 70	3 >= 90 3 >= 70 4 >= 50
Fast Left	6 >= 90 4 >= 70	4 >= 90 6 >= 70 1 >= 50	1 >= 90 2 >= 70 1 < 50	4 >= 90 3 >= 70 3 >= 50	2 >= 90 6 >= 70 4 >= 50	7 >= 90 5 >= 70	3 >= 90 4 >= 70 4>= 50
Slow Right	8 >= 90 3 >= 70 1 >= 50	8 >= 90 3 >= 70	1 >= 90 3 >= 70 3 >= 50	4 >= 90 4 >= 70 2 >= 50	9 >= 90 1 >= 70	2 >= 90 4 >= 70 5 >= 50	4 >= 90 2 >= 70 4>= 50
Fast Right	7 >= 90 4 >= 70	7 >= 90 4 >= 70	2 >= 90 1 >= 70 1 >= 50 2 < 50	3 >= 90 3 >= 70 3 >= 50	7 >= 90 2 >= 70 3 >= 50	1 >= 90 6 >= 70 4 >= 50	2 >= 90 4 >= 70 4>= 50
Total	64% >= 90 28% >= 70 7% >= 50	59% >= 90 38% >= 70 2% >= 50	40% >= 90 36% >= 70 13% >= 50 1% < 50	34% >= 90 36% >= 70 28% >= 50	53% >= 90 27% >= 70 18% >= 50	30% >= 90 50% >= 70 19% >= 50	29% >= 90 31% >= 70 39% >= 50

REFERENCES

[1] Hesham Alabbasi. 2016. *Contributions to the human body analysis from images.* Ph.D. Dissertation. University POLITEHNICA of Bucharest, Bucharest, Romania.

[2] H. Alabbasi, A. Gradinaru, F. Moldoveanu, and A. Moldoveanu. 2015. Human motion tracking evaluation using Kinect V2 sensor. In *2015 E-Health and Bioengineering Conference (EHB).* 1–4. https://doi.org/10.1109/EHB.2015.7391465

[3] Hemed Ali. 2012. *Motion Comparison using Microsoft Kinect.* Technical Report FIT3036. Monash University.

[4] Japan Karate Association. 2020. A Brief History of Japan Karate Association. Retrieved November 10, 2020 from https://www.jka.or.jp/en/about-jka/history/

[5] Japan Karate Association. 2020. The Father of Modern Karate. Retrieved November 10, 2020 from https://www.jka.or.jp/en/about-jka/profiles/supreme-master-funakoshi-gichin/

[6] P. Flach. 2012. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press. https://books.google.com.br/books?id=Ofp4h_oXsZ4C

[7] Epic Games. 2020. Unreal Engine. Retrieved November 10, 2020 from https://www.unrealengine.com/en-US/

[8] Hirokazu Kanazawa. 2006. *Black Belt Karate, The Intensive Course* (1st ed.). Kodansha International Ltd., Tokyo, Japan.

[9] Roanna Lun. 2018. *Human Activity Tracking and Recognition Using Kinect Sensor.* Ph.D. Dissertation. Cleveland State University, Cleveland, Ohio.

[10] Microsoft. 2014. AdaBoostTrigger. Retrieved November 10, 2020 from https://docs.microsoft.com/pt-br/previous-versions/windows/kinect/dn785522(v%3Dieb.10)

[11] Microsoft. 2014. Kinect For Windows SDK. Retrieved November 10, 2020 from https://docs.microsoft.com/pt-br/previous-versions/windows/kinect/dn799271(v=ieb.10)

[12] Microsoft. 2014. RFRProgress. Retrieved November 10, 2020 from https://docs.microsoft.com/pt-br/previous-versions/windows/kinect/dn785524(v%3Dieb.10)

[13] L. Miranda, T. Vieira, D. Martinez, T. Lewiner, A. W. Vieira, and M. F. M. Campos. 2012. Real-Time Gesture Recognition from Depth Data through Key Poses Learning and Decision Forests. In *2012 25th SIBGRAPI Conference on Graphics, Patterns and Images.* 268–275. https://doi.org/10.1109/SIBGRAPI.2012.44

[14] Masatoshi Nakayama. 2012. *Dynamic Karate* (1st ed.). Kodansha International Ltd., Tokyo, Japan.

[15] M. Rahman. 2017. *Beginning Microsoft Kinect for Windows SDK 2.0: Motion and Depth Sensing for Natural User Interfaces.* Apress. https://books.google.com.br/books?id=Q7UwDwAAQBAJ

[16] Wenbing Zhao. 2016. A concise tutorial on human motion tracking and recognition with Microsoft Kinect. *Science China Information Sciences* 59 (09 2016). https://doi.org/10.1007/s11432-016-5604-y

[17] Wenbing Zhao, Roanna Lun, Debbie Espy, and Ann Reinthal. 2014. Realtime Motion Assessment For Rehabilitation Exercises: Integration Of Kinematic Modeling With Fuzzy Inference. *Journal of Artificial Intelligence and Soft Computing Research* 4 (12 2014), 267–285. https://doi.org/10.1515/jaiscr-2015-0014

[18] W. Zhao, M. A. Reinthal, D. D. Espy, and X. Luo. 2017. Rule-Based Human Motion Tracking for Rehabilitation Exercises: Realtime Assessment, Feedback, and Guidance. *IEEE Access* 5 (2017), 21382–21394. https://doi.org/10.1109/ACCESS.2017.2759801