Body gesture validation using multi-dimensional dynamic time warping on Kinect data

Lorenzo Patras, Ion Giosan, Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca, Romania
Lorenzo.Patras@gmail.com, {Ion.Giosan, Sergiu.Nedevschi}@cs.utcluj.ro

***Abstract*—This paper presents a system capable of identifying and validating various human body gestures. Body data is acquired from a Kinect sensor and consist in a set of bones represented by their rotation in 3D space. The main goal is to correctly identify the user performed gesture and return feedback related to its performance accuracy. A database with several samples for each gesture is built and used as ground truth in the gesture validation process. A novel approach of dynamic time warping algorithm is proposed for synchronizing the performed gesture with the corresponding ground truth dataset. If the sequences are not synchronized, feedback is returned to user as a comparison between its performance and the closest sample in the database. Experimental results show high accuracy at about 90% success rate. The system performs better for gestures in which the users' body is fully exposed to the sensor.**

Keywords—body gestures; recognition; validation; quaternion; dynamic time warping; Kinect

I. INTRODUCTION

Human gesture recognition is a domain with a large range of application starting from controlling machines, controlling 3D objects in a virtual environment or characters in an animation, analysis of patterns in human movement etc. Our system's aim is to use human gesture validation in order to help users perform physical exercises, at home, without need of a qualified trainer. The system replaces the work that a supervisor should do by showing to the user how an exercise should be performed, tracking the way a user performs a specific exercise and returning feedback about correctness of user performance.

To achieve all these goals we will need a robust body posture recognizer and an algorithm to translate a series of body postures into a gesture. Body pose recognition is beyond the scope of this paper but a short introduction is worth mention.

Parvini et al. proposes in [1] a method to track hand gestures using a CyberGlove which is a glove having sensors on different key points. Schwarz et al. propose in [2] a similar method (extended to body pose estimation) using sensors attached to the extremities of an actor. Song et al. propose in [4] a method to estimate upper body pose and hands position based on depth data taken from a depth camera. Rehm et al. use in [3] acceleration data acquired from a Wiimote controller in order to estimate body activity. Shotton et al. [5] developed a method to estimate joint position using the depth data from a Microsoft Kinect sensor. The studies made by Shotton and its colleagues helped at the improvement of Kinect SDK which offers now a robust body tracking system. Due to its robustness and reliability this sensor was chosen for body data acquisition.

Body gesture recognition and/or validation is performed using a classic algorithm from speech recognition field, called dynamic time warping. This algorithm is often used in order to align two time-dependent sequences. Muller describes in its book [6] how this algorithm can be applied in body motion and music information retrieval. Our system considers a sequence of body poses as a multi-dimensional signal, each dimension being the rotation amount of each bone. Holt et al. propose in [7] a way to compute such multi-dimensional dynamic time warping, in which considers the distance between two points in a multi-dimensional space as the sum of the absolute differences in all dimensions. The proposed system is intended to be able to give user concise feedback about its errors. If we were to use the method proposed by Holt, specific information related to each bone would have been suppressed. As a result, the returned feedback would have contained only general information, whether the gesture was validated or not without specific information about which part of the gesture was not synchronized with the sample in the gesture database.

In this paper we propose a novel approach of dynamic time warping algorithm that is applied on body data, at each bone level, for synchronizing the user performed gesture with the ground-truth database recorded gestures and provide feedback to the user about the correctness of the performed gesture.

II. RELATED WORK

Body gesture recognition is a domain that was intensively studied in the last years. Although it was a field of interest for many engineers, the lack of hardware support and efficient algorithms for body tracking was a major drawback for studying gesture patterns.

In [8], Song studies different gestures and determined characteristics of body and hand posture sequence during different gestures. The main drawback of this method is that for each new gesture, a complex analysis should be made. Park and Lee proposed in [9] a more generic method using Hidden Markov Models (HMM). In their algorithm, each gesture is composed of a sequence of body frames while the temporal

relation between these frames is enforced by HMM. In [10], Forrer proposed a complete system for gesture recognition, also using HMM. Its system consisted of four main steps depicted by figure 1.

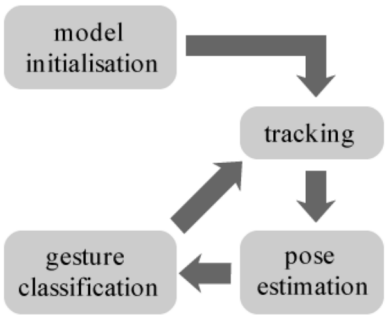

Fig. 1: Gesture recognition steps [11]

Blackburn and Ribeiro describe in [11] a method using dynamic time warping on isometric feature mapping. In [12], Ravi used the Kinect sensor for developing a gesture recognition system that could be used in physiotherapy. A partnership between Microsoft and Nike resulted in one of the best known products involving Kinect and gesture recognition, Nike+Kinect. This is a fitness application that can be used by anyone having an Xbox and a Kinect sensor.

III. System Architecture

A. Hardware components

In order to use the system, one needs a Kinect sensor for body tracking, and a personal computer for Kinect data interpretation and gesture computation.

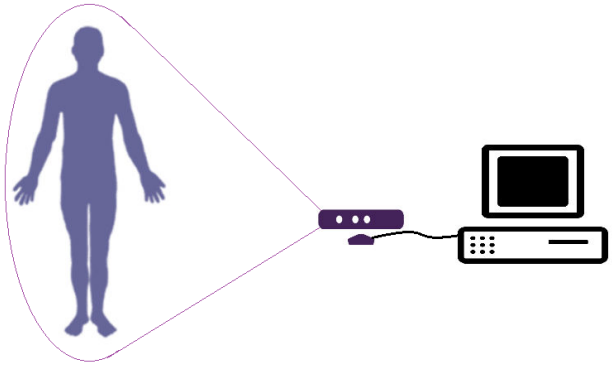

Fig. 2: Main hardware components:
1 - personal computer; 2 - Kinect sensor

B. Software components

In terms of software components a component to continuously track the body is needed. It will be called Kinect manager and will have the responsibility to get data from the Kinect sensor. Recorded samples for each gesture will be stored in a database. A component to compare the records from Kinect sensor with the recorded gestures in the database is also needed.

Since the stream of information coming from the sensor is continuous, a segmentation of it into gestures is necessary. For this, a component which records the initial-final position of the user is necessary. Initial-final position is defined as the position from which the user starts performing the gesture and to which the user returns when gesture is performed. In most of the cases, a gesture starts and ends in the same position, but for more complex gesture in which start position differs from end position, this component should be split in two components.

A Gesture Manager will manage the components previously listed. The conceptual software architecture is presented in figure 3.

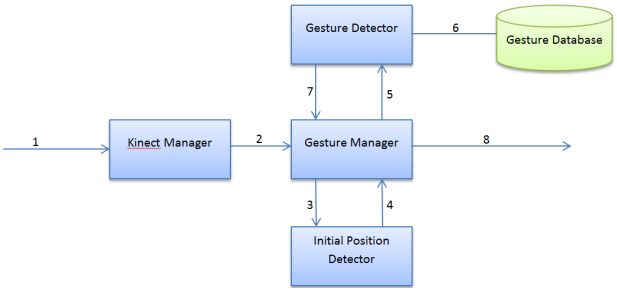

Fig. 3: Conceptual software architecture. 1 - data acquisition from Kinect; 2 - after data is transformed in a format suitable for the system it is sent to Gesture Manager; 3 – data is sent to Initial Position Detector; 4 - status of initial position is sent back to Gesture Manager; 5 - if body is not in the initial position data is sent to gesture detector; 6 - comparison between recorded gesture and samples in the database; 7- result is returned to Gesture Manager; 8 - display result

IV. Kinect Data

As stated before, due to its robustness, Microsoft Kinect sensor was chosen for body data acquisition. It consists of an infrared emitter, RGB camera, infrared camera and a microphone array, as shown in figure 4. Based on the infrared emitter and the infrared camera, it is able to obtain depth information about the surrounding environment. Studies performed by Shotton et al. [5] lead to a development of an algorithm for body tracking which was introduced in Kinect SDK offered by Microsoft for development [13].

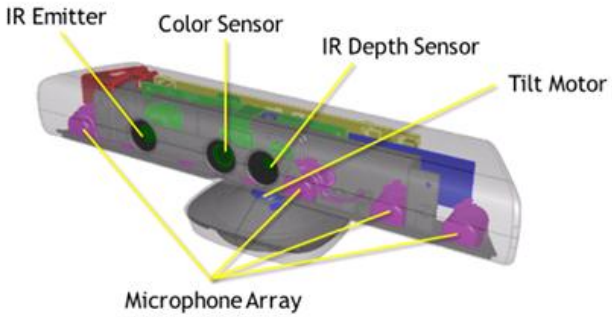

Fig. 4: Kinect sensor main components

Body data can be obtained as 3D position of joints relative to the sensor or as bones rotation. Using Kinect SDK one can obtain bone rotation either as rotation matrices or as quaternions. The rotations can be obtained as an absolute rotation or as relative rotation to parent bone. In both cases, rotations can be obtained as rotation matrices or quaternions. Bones are presented in figure 5 where parent of a bone is the bone to which it is attached (e.g.: parent of left forearm(8) is left arm(6)).

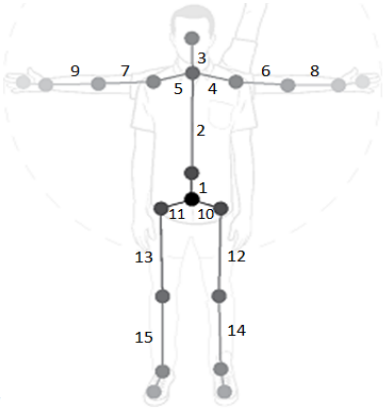

Fig. 5: Bones considered in the application: 1 - lower spine, 2 - upper spine, 3 - neck, 4 - left clavicle, 5 - right clavicle, 6 - left arm, 7 - right arm, 8 - left forearm, 9 - right forearm, 10 - left hip, 11 - right hip, 12 - left femurs, 13 - right femurs, 14 - left tibia, 15 - right tibia (base image taken from Microsoft Kinect SDK documentation)

For simplicity and robustness, the system uses the hierarchical rotations of bones expressed as quaternions. Quaternions are an extension of complex numbers, a four-dimensional number capable to uniquely express the rotation of an object (in this case a bone) in 3D space relative to another object. The general formula for a quaternion is

$$Q = w + xi + yj + zk,$$
$$\text{where } w, x, y, z \text{ are real numbers} \quad (1)$$
$$\text{and } i, j, k \text{ are imaginary units}$$

While performing an exercise, a series of body postures characterized by bone rotations, as quaternions, are recorded using Kinect. Since quaternions are used to uniquely describe a bone rotation in 3D space, we can assume that during two occurrences of the same gesture, the sequence of body postures are almost similar, thus the sequence of quaternions is also, almost similar.

V. Dynamic Time Warping

Dynamic Time Warping (DTW) is an efficient algorithm for expressing the similarity between two sequences regardless of their variation in time and speed. It was efficiently used in speech recognition for a long time as shown in [14][15]. They were also efficiently used in genetics for aligning gene expression time series [16]. In figure 5 it is demonstrated the difference between time alignment of two sequences and the alignment after applying DTW.

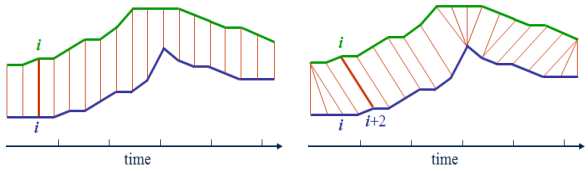

Fig. 5: left – time alignment, right – DTW alignment [16]

First step in applying the DTW algorithm on two sequences $X = (x_1, x_2, \dots, x_n)$, $n \in \mathbb{N}$ and $Y = (y_1, y_2, \dots, y_m)$, $n \in \mathbb{N}$ is to define a cost function

$$c : \mathbb{N} \times \mathbb{N} \to \mathbb{R} \quad (2)$$

Although many implementations of DTW recommend using as cost function the Euclidean distance of two samples,

$$cost(x_i, y_j) = \sqrt{x_i^2 + y_j^2} \quad (3)$$

for simplicity and computational reasons we choose as cost function

$$cost(x_i, y_j) = x_i - y_j \quad (4)$$

Next step is to construct the cost matrix of the two sequences. The cost matrix is obtained by applying the cost function on each combination of pairs from the sequences X and Y, being defined as

$$C \in \mathrm{R}^{N \times \mathrm{M}}, C(i,j) = cost(x_i, y_j) \quad (5)$$

The goal now is to find an optimal warping path between sequences X and Y. A warping path is defined in [16] as "a sequence $p = (p_1, p_2, \dots, p_L)$, where $p_l = (n_l, m_l) \in [1{:}N] \times [1 : \mathrm{M}]\ for\ l \in [1{:}L]$." The warping path starts from C(0,0) and ends at C(n,m) (see figure 6).

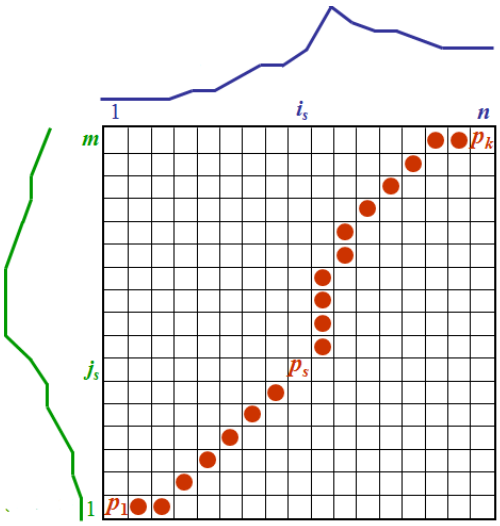

Fig. 6: Warping path example [16]

In order to compute a correct warping path, some constraints should be satisfied:

a) Monotonicity – The path does not go back in time

$$i_{t-1} \le i_t \text{ and } j_{t-1} \le j_t \tag{6}$$

This condition guarantees that features are not repeated in the alignment.

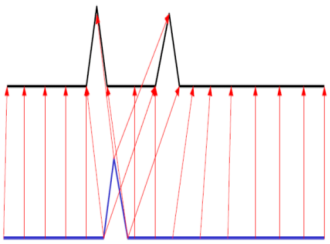

Fig. 7: Monotonicity condition violation [16]

b) Continuity – The path can pass only through adjacent cells.

$$i_t - i_{t-1} \le 1 \text{ and } j_t - j_{t-1} \le 1 \tag{7}$$

This condition guarantees that no important features are omitted.

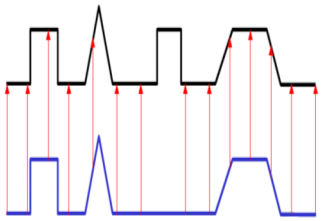

Fig. 8: Continuity condition violation [16]

c) Boundary conditions – The path must start at C(0,0) and end at C(n,m).

$$i_0 = 0, i_k = n \text{ and } j_0 = 0, j_k = m \tag{8}$$

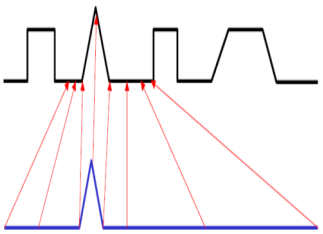

Fig. 9: Boundary condition violation [16]

This condition guarantees that the alignment is performed on the entire sequences, and does not consider only a part of one of the sequences.

d) Warping window – The path should be bounded in the proximity of the diagonal.

$$|i_t - j_t| \le r \text{ where } r > 0 \text{ is the window size} \tag{9}$$

This condition guarantees that the alignment will not skip different features and get stuck at similar features.

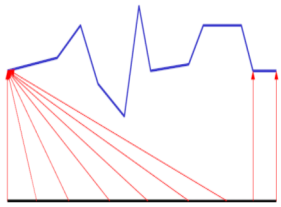

Fig. 10: Warping window condition violation [16]

e) Slope constraint – The path should not be too steep or shoo shallow. This constraint prevents short parts of the sequence to be matched to very long ones.

$$\frac{j_{t_p} - j_{t_0}}{i_{t_p} - i_{t_0}} \le p \text{ and } \frac{i_{t_p} - i_{t_0}}{j_{t_p} - j_{t_0}} \le q \tag{10}$$

where $q > 0$ is the number of steps in the x direction and $p > 0$ is the number of steps in the y direction.

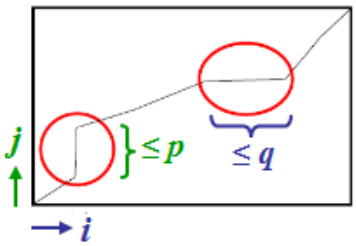

Fig. 11: Slope constraint violation [16]

Even though all these constraints are imposed, there will be more than a single warping path, each of them having its own cost defined as

$$pathCost_p(X,Y) = \sum_{l=1}^{L} cost\left(x_{n_l}, y_{m_l}\right) \tag{11}$$

The goal is to find the optimum warping path which is the path having the minimal cost. This is the end result of the DTW algorithm and it is defined as

$$DTWcost(X,Y) = \min\left\{ pathCost_p(X,Y) \,\middle|\, p \text{ is a warping path} \right\} \quad (12)$$

For two sequences that are almost similar the cost will be low while for two sequences that don't resemble many characteristics the cost will be high.

One of the main advantages of DTW is the fact that the cost matrix can be computed in parallel. What is more, in our particular case, the cost for each bone rotation can also be computed in parallel thus improving the computation time.

VI. GESTURE VALIDATION

Since Kinect body tracking is a marker less body tracking, data might appear with some noise. In order to remove the noise one can apply a low-pass filter on the sequence of poses to smooth the signal.

Fig. 12: Signal plot for x component of quaternion corresponding to upper left arm, in a sequence of body poses representing raising the left arm; black dots represent the raw signal taken from Kinect sensor while orange dots represent the filtered signal

Afterwards, DTW cost must be computed between each quaternion component of each bone. Computing the cost on the entire matrix, using a greedy algorithm might result in a high cost, both for DTW and computational time. Using the window constraint, the DTW cost is greatly reduced. As a result of window constraint, the computational time is also reduced. The experimental results are shown in figures 13 and 14. The square represents the cost matrix. A low cost between two samples yields a white area while a higher cost yields a darker red area. Green was used to display the computed warping path while gray area represents the warping constraint.

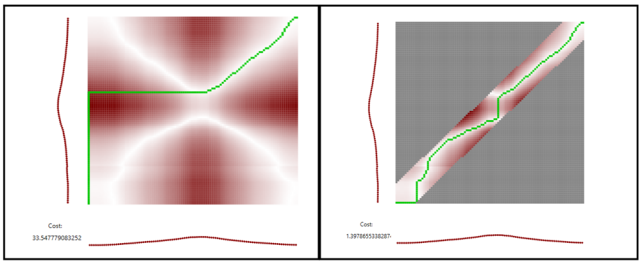

Fig. 13: DTW matrix and warping path computed between two instances of left arm raising gesture; DTW matrix in the pictures represent the matrix for *x* component of quaternion representing left arm; left - simple DTW matrix (cost = 33.5477); right - DTW matrix and warping path constraint (cost = 1.39704)

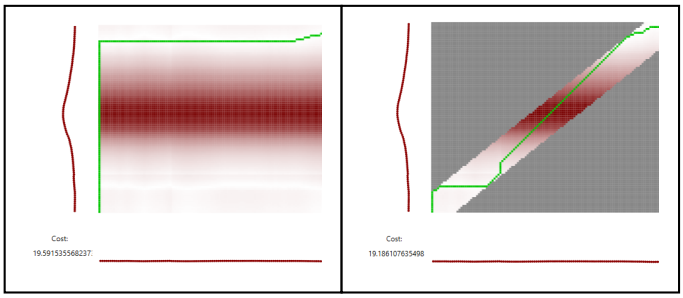

Fig. 14: DTW matrix and warping path computed between an instance of left arm raising gesture and an instance of right arm raising gesture; DTW matrix in the pictures represent the matrix for *x* component of quaternion representing left arm; left - simple DTW matrix (cost = 19.59153); right - DTW matrix and warping path constraint (cost = 19.1861)

In order to perform body gesture validation we define our own cost function between two series of body poses as

$$bodyCost(body1, body2) = \sum_{bones} \sum_{qComp} DTWcost\begin{pmatrix} body1.bones.qComp, \\ body2.bones.qComp \end{pmatrix} \quad (13)$$

where bones ∈ {*lower spine, upper spine, neck, left clavicle, right clavicle, left arm, right arm, left forearm, right forearm, left hip, right hip, left femurs, right femurs, left tibia, right tibia*} and qComp ∈ {*wComponent, xComponent, yComponent, zComponent*}, the components of a quaternion. The overall body cost is obtained as the sum of all DTW costs computed at the level of quaternion component.

One can observe from figures 13 and 14 that the computing DTW without window constraint may not lead to the optimal warping path, and the DTW cost may be high even for the same gesture. What is more, the DTW cost for two instances of "raise left hand" which are similar, is higher than the cost computed for an instance of "raise left hand" and "raise right hand". However, adding the window constraint and bounding the warping path to diagonal, one can observe that for similar gestures the cost is really small while for different gestures, we still obtain a high cost.

Using this approach, both gesture recognition and gesture validation can be performed. Gesture recognition may be slower depending on the size of the database, due to the fact that the system should compare each gesture performed by the user, to each sample in the database and the sample with the lowest cost would be considered to be the one matching the gesture. Gesture validation supposes that the user already knows what gesture wants to perform and its performance will be compared against those samples resembling the selected gesture. The result would be either validation of the gesture, i.e. selected gesture is performed correctly, or invalidation, i.e. feedback about performance correctness is returned.

Before using the system, it has to be trained in order to construct a database of gestures. For achieving this, an individual must repeat the gesture for a number of times in order to construct the set of sample gestures

$$gestureSet = \{sample_0, sample_1, \ldots, sample_n\} \quad \text{where } n \in \mathbb{N} \tag{14}$$

In our case n = 5 proved to be enough for training the system. After this a cost threshold is computed as

$$threshold = \max\{bodyCost(sample_i, sample_j)\} \quad \text{where } sample_i, sample_j \in gestureSet \tag{15}$$

The threshold value must be computed for each gesture when it is added to the database. Given the fact that the *n* samples resemble the same gesture, one can assume that the gesture performed by the user must be in the same range. As a result, the computed cost between the recorded gesture and at least one of the samples in the database must be lower than the threshold to validate the gesture:

$$isValidGesture = bodyCost(record, sample_i) < threshold, \quad \text{where } sample_i \in gestureSet \tag{16}$$

In order to use the system, user must select the gesture it wants to perform. The system will require the user to stay in the initial position for 5 seconds. This is necessary in order to train the system with user initial position, which will be the marker by which the data stream will be fragmented into gestures.

The conceptual diagram presented in figure 3 shows that the data taken from Kinect sensor is handled by a component called Kinect Manager. This transforms data into a suitable format to be further sent to Gesture Manager. Gesture manager is the main component of the system, which receives data as a sequence of body poses and delegates the work to Initial Position Detector or Gesture Detector.

First time, Initial Position Detector receives the data and sends back a message to Gesture Manager regarding the posture. If the body posture is in initial position the system stays in idle state. As soon as a body pose is not in initial position, Gesture Manager sends the samples to Gesture Detector. This component accumulates samples in a queue of poses. When the body returns to initial position, Gesture Manager sends a message to Gesture Detector to start computing the gesture validation.

Based on the result of gesture validation, the user will either receive as feedback the fact that the gesture was correctly performed or a replay will be shown, where both the gesture performed by the user and the closest sample in the database, based on *bodyCost*, will be plotted in parallel (see figure 15).

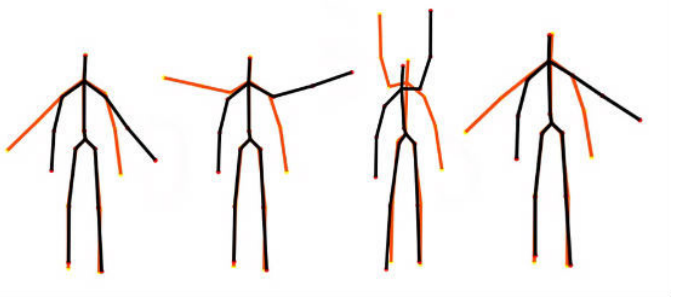

Fig. 15: Example of feedback sequence for performing a lateral right hand raise while the expected gesture was lateral left hand raise – black skeleton represents the gesture performed by the user while red skeleton represents the expected gesture

It is important to mention the fact that for the same user, exercises present similar characteristics. However, during testing of the system, we observed that different user perform the same exercise slightly different, but still in a correct form. In order to be able to train the system with one user and validate exercises performed by another user a confidence threshold should be considered. As a result the real threshold for the exercise performed by the new user will be

$$realThreshold = threshold * confidenceThreshold \tag{17}$$

Confidence threshold is computed experimentally, by observing a number *n* of persons performing a gesture *m* times. For this, $recordSet_i$ is defined as the set of records for person *i*

$$recordSet = \{rec_0, rec_1, \ldots, rec_m\} \text{ where } m \in \mathbb{N} \tag{18}$$

Then the cost for each person is computed as

$$personCost = \min\{bodyCost(sample_i, rec_j)\} \quad \text{where } sample_i \in gestureSet \text{ and } rec_j \in recordSet \tag{19}$$

This cost shows which is the minimum cost for each gesture performance recognized by the system. Taking the maximum of these minimum costs and dividing it by threshold we obtain the level of confidence threshold

$$confidenceThreshold = \max\{personCost_i | 0 \le i \le m, m \in \mathbb{N}\} / threshold \tag{20}$$

VII. Experimental Results

In order to prove the usefulness of the system, the database was trained with 10 gestures: 7 simple gestures (first 7 rows of the result table) and 3 slightly more complex ones (last 3 rows of the result table). Afterwards, a user performed each gesture from the database 10 times. The results are presented in the Table I.

TABLE I. Performances of the Body Gesture Validation System

Gesture	Threshold cost	Performed gesture			
		Avg. cost	Min. cost	Max. cost	Success rate
Left arm lateral raise	70.7250443	75.79240	62.65014	83.20147	10/10
Right arm lateral raise	62.28287	70.34488	61.07209	78.16035	9/10
Both arms lateral raise	79.43044	91.44340	73.06082	108.9929	9/10
Right arm front raise	82.9584351	116.52569	95.00497	145.6474	9/10
Both arms curls	38.3784676	51.37135	46.89584	55.37613	10/10
Right knee raise	50.32552	54.12544	47.90615	60.96284	10/10
Right foot forward	69.57885	55.51976	31.3826	52.2625	8/10
Squat	155.280457	144.3788	101.613	212.1844	7/10
Right foot reverse lunge	112.773056	116.90734	74.78407	162.7144	9/10
Left foot lateral lunge	97.99207	86.224108	76.19372	92.3608	9/10

As one can observe from the Table I, for the simpler gestures, the validation worked better than for the complex ones. One of the main reasons for this is the fact that in simpler gestures, all bones were exposed to Kinect sensor, so the approximation of them was accurate enough. The complex gestures contain body poses in which some of the bones are hidden by others. For example, when performing a squat, the hip center and hip bones are hidden by tibia bones in the lowest position of the squat. This results in an approximation made by the sensor which is not always accurate enough. Tests were performed with more complex gestures but due to the limitation of the sensor, the approximation of the bones resulted in costs so big between samples in the database, that a correct gesture validation would have not been so reliable.

VIII. Conclusions

Following the experimental results, the remark is that the proposed system is more suitable for gestures in which the whole body is exposed to the sensor. This is due to the fact that the sensor performs an approximation of the hidden bones. The approximation can lead to some inaccuracies in the synchronization of the body pose sequences.

In order to train the system with one user and validate the gesture for another user, a confidence threshold should be computed. This is due to the fact that the Kinect sensor records slightly different bones data for different users, even though they perform similar gestures.

As a future work, an improved system for body data acquisition could be used in order to increase the accuracy of the recorded data, which will lead to improved results of the currently proposed method.

Acknowledgment

This work has been supported by UEFISCDI (Romanian National Research Agency) in the national research project Cooperative Advanced Driving Assistance System Based on Smart Mobile Platforms and Road Side Units (SmartCoDrive), project no. PNII-PCCA 18/2012.

References

[1] F. Parvini, D. McLeod, C. Shahabi, B. Navai, B. Zali, S. Ghandeharizadeh, "An Approach to Glove-Based Gesture Recognition", in *Human-Computer Interaction. Novel Interaction Methods and Techniques*, pp. 236-245, July 2009.

[2] L. A. Schwarz, D. Mateus, N. Navab, "Discriminative Human Full-Body Pose Estimation from Wearable Intertial Sensor Data", in *Modeling the Physiological Human*, pp. 159-172, November 2009.

[3] M. Rehm, N. Bee, E. André, "Acceletometer Based Gesture Recognition for Culture Specific Interactions", in *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers*, pp. 13-22, 2008.

[4] Y. Song, D. Demirdjian, R. Davis, "Tracking Body and Hands for Gesture Recognition: NATOPS Aircraft Handling Signals Database", in *Proceedings of the 9th IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 500-506, March 2011.

[5] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, "Real-Time Human Pose Recognition in Parts from Single Depth Images", in *Machine Learning for Computer Vision*, pp. 119-135, June 2011.

[6] M. Müller, "Dynamic Time Warping" in *Information Retrieval for Music and Motion*, pp. 69-84, 2007.

[7] G.A. ten Holt, M. J. T. Reinders, E. A. Hendriks, "Multi-Dimensional Dynamic Time Warping for Gesture Recognition", in *Proceedings of Conference of the Advanced School for Computing and Imaging*, June 2007.

[8] Y. Song, "Multi-Signal Gesture Recognition Using Body and Hand Poses", *Master Thesis*, September 2010.

[9] A-Y. Park, S. W. Lee, "Gesture Spotting in Continuous Whole Body Action Sequences Using Discrete Hidden Markov Models", in *Proceedings of the 6th international conference on Gesture in Human-Computer Interaction and Simulation*, pp. 100-111, 2005.

[10] T. Forrer, "Arm and Body Gesture Recognition", online: https://diuf.unifr.ch/main/diva/sites/diuf.unifr.ch.main.diva/files/joomla_reportForrer.pdft

[11] J. Blackburn, E. Ribeiro, "Human Motion Recognition Using Isomap and Dynamic Time Warping", in *Human Motion - Understanding, Modeling, Capture and Animation*, pp. 285-298, 2007.

[12] A. Ravi, "Automatic Gesture Recognition and Tracking System for Physiotherapy", *Technical Report at Electrical Engineering and Computer Sciences University of California at Berkley*, May 2013, online: http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-112.pdf

[13] Microsoft Kinect for Windows SDK documentation, online: https://msdn.microsoft.com/en-us/library/hh855347.aspxt

[14] T. Bin Amin, "Speech Recognition using Dynamic Time Warping", in *Advances in Space Technologies*, pp. 74-79, November 2008.

[15] L. Muda, M. Begam, I. Elamvazuthi, "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient and Dynamic Time Warping Techniques", in *Computing Research Repository*, March 2010.

[16] J. Criel, E. Tsiporkova, "Gene Time Expression Warper: a tool for alignment, template matching and visualization of gene expression time series", in *Bioinformatics Oxford Journal* vol. 22, issue 2, pp. 251-252, November 2005.