Visual Recognition of Gymnastic Exercise Sequences. Application to Supervision and Robot Learning by Demonstration

Jaime Duque Domingo [1], Jaime Gómez-García-Bermejo [1], Eduardo Zalama [1]

[1] *ITAP-DISA, University of Valladolid, Valladolid, Spain*
jaime.duque@uva.es

Abstract

This work presents a novel software architecture to autonomously identify and evaluate the gymnastic activity that people are carrying out. It is composed of three different interconnected layers. The first corresponds to a Multilayer Perceptron (MLP) trained from a set of angular magnitudes derived from the information provided by the OpenPose library. This library works frame by frame, so some postures may be incorrectly detected due to eventual occlusions. The MLP layer makes it possible to accurately identify the posture a person is performing. A second layer, based on a Hidden Markov Model (HMM) and the Viterbi algorithm, filters the incorrect spurious postures. Thus, the accuracy of the algorithm is improved, leading to a precise sequence of postures. A third layer identifies the current exercise and evaluates whether the person is doing it at a correct speed. This layer uses an innovative Modified Levenshtein Distance (MLD), which considers not only the number of operations to transform a given sequence, but also the nature of the elements participating in the comparison. The system works in real time with little delay, thus recognizing sequences of arbitrary length and providing continuous feedback on the exercises being performed. An experiment carried out consisted in reproducing the output of the second layer on an autonomous Pepper robot that can be used in environments where physical exercise is performed, such as a residence for the elderly or others. It has reproduced different exercises previously executed by an instructor so that people can copy the robot. The article analyzes the current situation of the automated gymnastic activities recognition, presents the architecture, the different experiments carried out and the results obtained. The integration of the three components (MLP, HMM and MLD) results in a robust system that has allowed us to improve the results of previous works.

Keywords: Automated gymnastic activity recognition, Autonomous robot, Social robotics, Multilayer perceptron, MLP, Hidden Markov model, HMM, Viterbi, Modified Levenshtein distance, MLD, OpenPose

1. Introduction

The automation of human poses and gestures recognition plays a key role in many fields. For example, there has been growing interest in the field of intelligent *Smart Living* systems as an element that facilitates the interaction of people with their environment. Moreover, the recognition of human activities can be of great interest in the field of industrial robotics, where it allows robots to be programmed by observation and demonstration, without requiring explicit programming. In the field of social robotics, the interlocutor gesture recognition provides a very important flow of information to facilitate interaction. A further interesting application field is the use of autonomous social robots to entertain and keep elderly people physically and mentally active: the robot proposes and executes a series of exercises (movements of arms, legs, hands, etc.) and the person simultaneously reproduces the proposed exercises. The challenge is that the robot can monitor the exercises performed by the person to evaluate their performance, correcting and motivating the user.

This article presents an architecture that allows the recognition of people's activities through three processes: a neural network that obtains people's posture from the joints returned by OpenPose [1]; a Hidden Markov Model that allows rectification of possible errors in the previous step; and finally, a Modified Levenshtein Distance (MLD) that allows the correct activity to be obtained. Several experiments have been carried out to validate the method, including an experiment about learning by demonstration, where a Pepper robot [2] learns different exercises from an instructor and repeats them in a residence for the elderly. In addition, our system allows the execution of the exercises to be evaluated.

The article is structured as follows: Section 2 explores the state-of-the-art of the technologies considered in this paper. Section 3 shows how the system works by integrating the Multilayer Perceptron, the Hidden Markov Model, and the Modified Levenshtein Distance. In Section 4, the different experiments carried out on people doing gymnastic activities are reported. In Section 5, the results are reported and discussed. Finally, Section 6 notes the advantages and limitations of the presented system and suggests future developments based on this method.

2. Overview of related work

The recognition of gymnastic activities was carried out using different techniques. Fasola and Mataric [3] used a standard RGB camera and visual analysis was performed against a uniform background. A robot compared the user's current arm angles to the pre-specified goal arm angles to determine whether an exercise was performed correctly. Monitoring was limited to three exercises. In [4], the robot physically demonstrated exercises for the user to be followed, and monitored the user's progress using a vision-processing unit that detected face and hand movements. This paper made an important contribution but some aspects were limited, such as the recognition of exercises restricted by the detection of hands and face. Quantitative results were not provided, demonstrations were limited to two types of gesture and corrective feedback was not shown to the elderly. Vishwakarma et al. [5] proposed an algorithm for human actions recognition using a spatial distribution of edge gradient (SDEG) for human pose and the detailed geometric orientation of a human silhouette in a video sequence. However, that approach did not rely on modern techniques such as Deep Neural Networks, leading to worse results. More recently, Gil-Martín et al. [6] have developed an outstanding system that uses several on-body sensors and Convolutional Neural Networks (CNNs) to detect the performed human activity. Also in [7], the authors use fuzzy clustering to carry out the Human Activity Recognition (HAR). However, neither work uses cameras, requiring users to wear on-body connected sensors.

The Hidden Markov Model (HMM) is one of the main temporal classification techniques used in human physical activity recognition [8]. It is one of the major trends and challenges in ubiquitous robotics research. Kwon et al. [9] proposed an improved skeleton tracker and human activity recognizer based on complexity-based motion features. They used a Kinect 3D sensor (Microsoft, USA) to obtain body joints and a Kalman filter modeled with the joints speed and integrated with a Deep Recurrent Neural Network (DRNN) to optimize the joints, even in the presence of self-occlusion. In addition, a Subsequence of time-series clustering (STSC) [10] was used to obtain a set of cluster centers of a sequence of postures that was integrated with a HMM (Hidden Markov Model) to directly obtain the activity. Other authors, such as Piyathilaka and Kodagoda [11] had previously explored the use of 3D Kinect joints with an HMM for human daily activity recognition. In our work, instead of using the HMM model to detect the class, it has been used to rectify the joints based on possible movements.

Görer et al. [12] developed a robotic fitness coach that learned a set of physical exercises from a professional trainer, and assisted elderly subjects in performing these gestures using an RGBD camera. It tried to minimize the angular difference between the robot's joint and that of the user. Tanguy et al. [13] proposed a software architecture for a robot coach, based on imitation learning techniques using Gaussian Mixture Models. It used a Kinect 3D sensor and showed the results for three joints without distinguishing different types of movement. More recently, Lotfi et al. [14] have created a *Socially Assistive Robot* (SAR) to engage, coach, assess and motivate older adults performing physical exercises. The system is composed of a vision module based on a Kinect sensor (version 2), a display module to present visual feedback to the user, the Communication/Sound Module which provides speech recognition and audio feedback, a database that stores the exercise details and the performance records, and the processing module which extracts information on joint angles and compares them with a reference. Although the system includes a robot, this robot is only used to show a feedback table of the exercises. It is worth noting that Fasola and Mataric [15] showed that people prefer to be trained by a physical object (e.g., robot) than virtual training software.

Skeleton tracking has been widely used in robotics. A remarkable work was presented by Ghandour et al. [16], who created a Human Robot Interaction system (HRI) to give indications to mobile robots to avoid collisions with people. They used the skeleton returned by the sensor Kinect 2.0, consisting of 25 joints. The works that use skeleton tracking for HAR have increased over the last few years [17]. OpenPose [1] has represented a major advance in the recognition of a person's joints. Although it has recently been developed, several authors are beginning to use it due to the robustness of the recognition. Noori et al. [18] proposed a robust human activity recognizer based on OpenPose, motion features, and DRNN. They used an RNN with LSTM cells, which tackle the long term dependencies found in data. The input of the DRNN were magnitudes between two consecutive frames, distances between the two-consecutive frames in x-axis and y-axis as well as angles, and there was one output for each activity. Their model achieved an average accuracy of 92.4% detecting 11 activities. Some authors have used HMMs [9, 11], while others have considered RNNs [18]. The comparison between both techniques has been described in [19], obtaining very similar results for a gesture recognition problem. Hidden Markov Models need some assumptions, but are simpler than Recurrent Neural Networks and work better with smaller datasets. RNNs require large datasets.

Costa et al. [20] have recently presented the PHAROS architecture, its components and the experimental results. The architecture has three main strands: A Pepper robot that interacts with the users and records their exercise performance; the Human Exercise Recognition, that uses the Pepper recorded information to classify the exercise performed using Deep Leaning methods; and the Recommender, a smart decision maker that schedules periodically personalized physical exercises in the users' agenda. The experimental results show a high accuracy in terms of detecting and classifying the physical exercises done by 7 persons. It uses a CNN for the recognition of exercises from the silhouette image obtained from OpenPose. The computational cost causes the recognition sequence to be limited to 47 frames. However, the use of a CNN to classify the postures has a big computational cost and other methods, such as the one presented in our paper, facilitates this process.

Our paper presents a novel approach to identify the gymnastic activities that people are doing and indicate how they are doing and how to improve. Among the contributions, it is worth mentioning the following ones:

- A novel architecture with three layers is presented: a first layer represented by a Multilayer Perceptron (MLP) neural network, a second layer which uses a Hidden Markov Model (HMM), and a third layer consisting of a new Modified Levenshtein Distance (MLD) algorithm.
- The human joints returned by OpenPose are used to compute the angular magnitudes as well as the length of the segments between joints, and these values are used to train an MLP network to enable the recognition of postures in any direction despite the use of 2D cameras.
- When joint occlusions occur and spurious postures are obtained by the

neural network, a Hidden Markov Model improves the recognition.

- A novel MLD algorithm detects the right exercise and allows the execution performances to be evaluated.
- The method uses common 2D cameras, which simplifies the required setup.
- The method is able to recognize exercise sequences of arbitrary length.
- It allows the recognition of exercise sequences of multiple users present in the same scene.
- The proposed methodology allows the implementation of learning sequences by demonstration of arbitrary length, taking into account different types of positions and different speeds.
- The system allows whether the user is performing the exercises in the appropriate sequence or tempo or not to be identified.
- The recognition accuracy achieved is 97.73% for the postures (after HMM), 94.6% for partial activity detection and 98.05% for complete activity recognition (after MLD).

3. Analysis of the system

An important issue in gymnastic activity recognition is the control of possible errors in human posture detection. OpenPose [1] is widely used but, because the recognition is obtained in specific frames, some errors can appear when a complete exercise is carried out by a person. Our method deals with this problem using an MLP neural network to detect the posture in specific frames and a Hidden Markov Model (HMM) to suppress spurious postures. Our system detects the sequence of people's postures, identifies the exercise and evaluates the performance of the exercise.

Figure 1 shows the scheme of the proposed architecture. When the system starts, OpenPose returns the skeletons found at $20Hz$. It may return several skeletons that are tracked to keep the sequence followed by each person. A correlation tracker based on [21] is used. This method considers the approach of Bolme et al. [22] and makes use of learning discriminative correlation filters based on a scale pyramid representation. The authors use separate filters for tracking, in real-time, objects that change in both translation and scaling. The angular magnitudes of the joints and the lengths of the segments between joints are introduced into the MLP neural network and the output is fed into the HMM. Considering the last person's postures, the HMM filters the sequence that is introduced into the Modified Levenshtein Distance module (MLD), which identifies the exercise and evaluates the results to be communicated to the user.

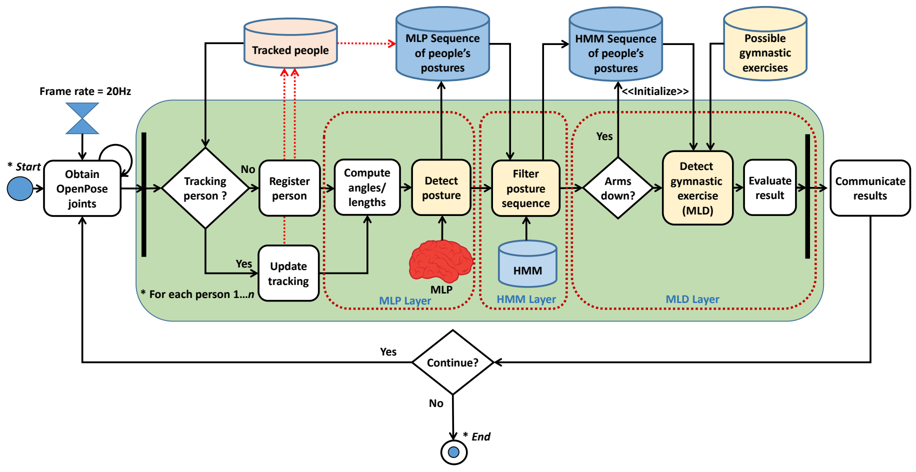


Figure 1: Scheme of the system

The OpenPose pipeline is composed of different steps. A CNN trained using the COCO 2016 dataset / MPII dataset [23] returns a heat-map and the Partial Affinity Fields (PAFs) [24]. A heat-map is a matrix that stores the confidence the network has that a certain pixel contains a certain part of the body. A PAF gives information about the position and orientation of body segments, named pairs. A pair associates couples of parts, specifically body joints. After a Non-Maximum-Suppression (NMS) algorithm, the location of the part candidates is obtained from the heat-maps. When the candidates for each one of the body parts have been found, an assignment problem selects the right candidate by means of the integration of the PAFs. Finally, the body parts of each person are connected, considering that two segments at a joint with the same coordinates are part of the same person.

Subsection 3.1 explains how the proposed system works to extract people's postures. Subsection 3.2 presents how an HMM can filter the predictions of the Neural Network in real-time. Subsection 3.3 explains how the Levenshtein distance allows what kind of exercise the person is doing to be discerned.

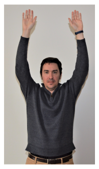
3.1. Multilayer perceptron model

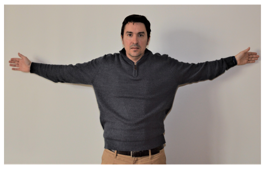
A Multilayer Perceptron model (MLP) is a standard fully connected neural network model [25]. This network is composed of several layers of nodes where each node has an input connected to all outputs from the previous layer and an output connected to all inputs for nodes in the next layer. The MLP is created with at least one dense layer. A common use of these networks are problems related to binary classification, multi-class classification, and regression. These networks are usually trained quickly and produce accurate results. The process of tuning an MLP requires knowledge about the problem, the inputs and the outputs. The training is carried out by a process of back-propagation, where the weights of the connections are changed after each element is processed, based on the amount of error in the output compared to the expected result.

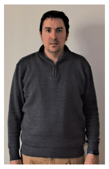
An MLP has been created in this work as the first step to obtain a person's posture, i.e., arms up. Since OpenPose [1] returns a set of 25 human joints, it is possible to process these joints to be integrable with a neural network model. Other works, such as Costa et al. [20], have considered thresholding the image produced by OpenPose to be introduced in a Convolutional Neural Network (CNN), mostly used for image recognition. However, this approach entails a high computational cost and requires a huge amount of training images. Our MLP considers only the angles and lengths of the body segments needed for the trained postures. There is an output for each possible posture, which are shown in Figure 2. To avoid the problem of occlusions, where some joints cannot be detected due to occlusions, binary indicators for each joint/link are also introduced as input into the network. These indicators show the presence (1) or absence (0) of a joint/link on the scene. Keep in mind, that a null entry of a joint reflects a value of 0° in that joint, but does not show anything about whether this joint is present or not on the scene.

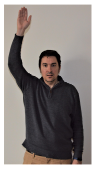
Figure 3 shows the scheme of the network. α_k represents the joint angle in k between links L_{ki} and L_{kj} (see Figure 4). These angular amplitudes are normalized between 0 and 1 (2π). The function $b(L_{ki})$ takes the value 1 if the link L_{ki} has been properly obtained, or 0 if the link has not been detected. $b(L_{ki}) \wedge b(L_{kj})$ shows whether both links have been obtained or not, which is needed to determine if α_k is properly computed. The function $n(L_{ki})$ normalizes the module of a link, L_{ki}, balanced by the module of the link between the neck and the waist that is considered to be the longest one, as seen in Equation 1. This normalization makes the system invariant to distance from the user. The links are vectors obtained with the joints returned by OpenPose, specifically J_k and J_i for a link L_{ki}. $P(c_x)$ represents the probability of being in the posture represented by class c_x, where $c_x = c_1 ... c_{19}$ for each of the 19 postures previously defined. These probabilities are also important for the next integration with the Markov model.

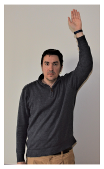
$$n(L_{ki}) = \tfrac{|L_{ki}|}{|L_{18}|} = \frac{\left|\overrightarrow{J_k J_i}\right|}{\left|\overrightarrow{J_1 J_8}\right|} \tag{1}$$

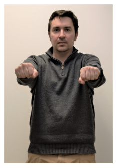

(a) Arms up (P1)

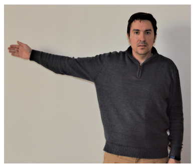

(b) Medium arms (P2)

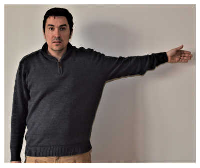

(c) Arms down (P3)

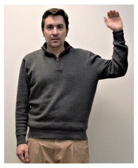

(d) Left arm up (P4)

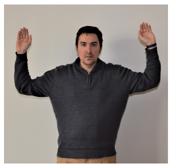

(e) Right arm up (P5)

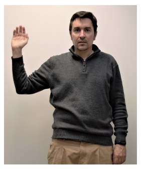

(f) Arms forward (P6)

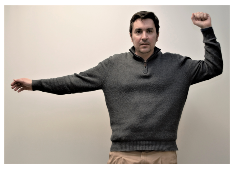

(g) Left arm in the middle (P7)

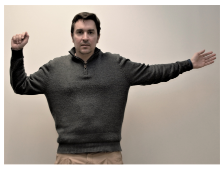

(h) Right arm in the middle (P8)

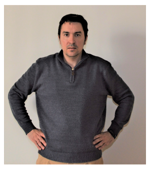

(i) Right arm half up/Left down (P9)

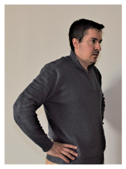

(j) Arms half up (P10)

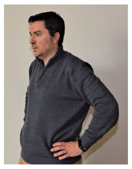

(k) Left arm half up/Right down (P11)

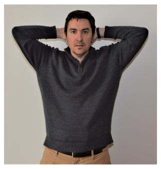

(l) Right arm half up (P12)

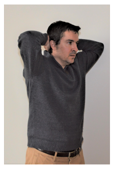

(m) Left arm half up (P13)


(n) Arms akimbo (P14)


(o) Arms akimbo to the left (P15)


(p) Arms akimbo to the right (P16)


(q) Arms behind head (P17)


(r) Arms behind head to the left (P18)

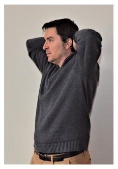

(s) Arms behind head to the right (P19)

Figure 2: Person's postures detected by the neural network

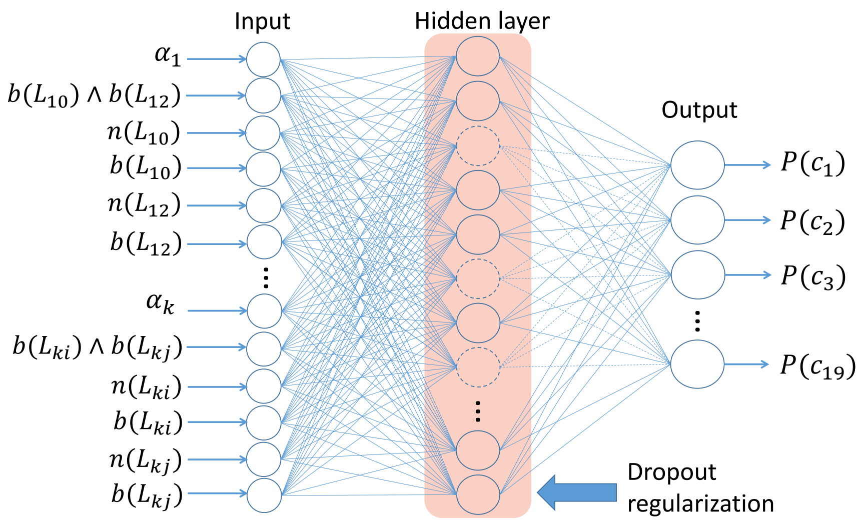


Figure 3: MLP used for posture recognition

Figure 4 shows some of the joints returned by OpenPose (J_1, J_2, etc.). The links used to compute the inputs of the neural network are obtained with the different joints. The angle α_k is obtained using the Equation 2.

$$\alpha_k = \arccos\left[\frac{L_{ki} \cdot L_{kj}}{|L_{ki}| \cdot |L_{kj}|}\right] \tag{2}$$

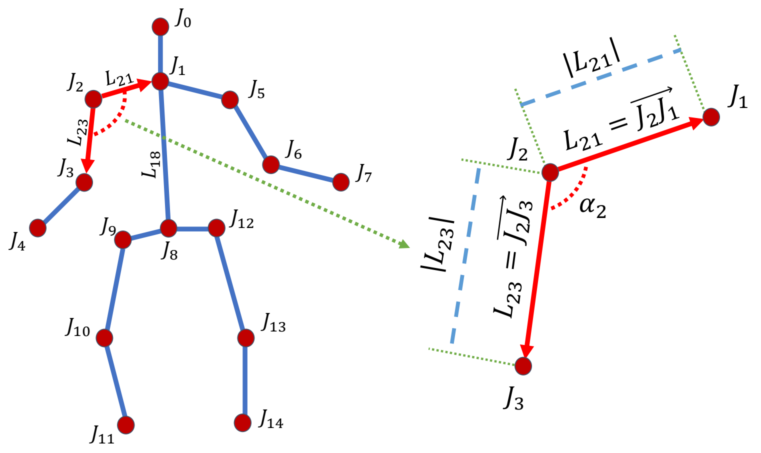


Figure 4: Skeleton returned by OpenPose and computation of MLP inputs

The angular magnitudes used are those corresponding to α_2, α_3, α_5 and α_6.

The link's modules considered are L_{21}, L_{23}, L_{34}, L_{51}, L_{65}, L_{67}. All of them are balanced by the module L_{18}.

The MLP has one hidden layer composed by λ_h neurons, obtained by Equation 3 [26].

$$\lambda_h = \frac{\lambda_s}{\alpha \cdot (\lambda_i + \lambda_o)} \tag{3}$$

where λ_s is the number of training samples, λ_i the number of inputs, λ_o the number of outputs and α an arbitrary scaling factor that takes the value 2-10 and contributes to reduce overfitting. In our system, $\lambda_i = 20$, $\lambda_o = 19$ and $\alpha = 3$. In addition, a *Dropout* regularization step with a rate of 0.2 has also been used to prevent overfitting. *Adam* algorithm [27] has been used for stochastic optimization. A $He-normal$ distribution has initialized the weights of the hidden layer and a *ReLu* function has been used as activation. Finally, a $Softmax$ function is responsible of the output classification.

3.2. Hidden Markov Model (HMM)

A Hidden Markov Model (HMM) is used to filter the sequence of person's postures over time. The Viterbi algorithm [28, 29, 30] is a dynamic programming algorithm that allows the most probable sequence of hidden states to be found, the so-called Viterbi path, which produces an observed sequence of events. An example is shown in Figure 5. A person follows a sequence of postures but one of them, marked with a red cross, is not possible because in order to go from arms behind the head to arms down, there would have to be some intermediate state like having the arms in the middle. Markov's model filters out this anomalous situation, derived for example from eventual occlusions of the arms, and makes the transition between postures to happen through possible states. A transition matrix indicates how likely it is to go from one posture to another.

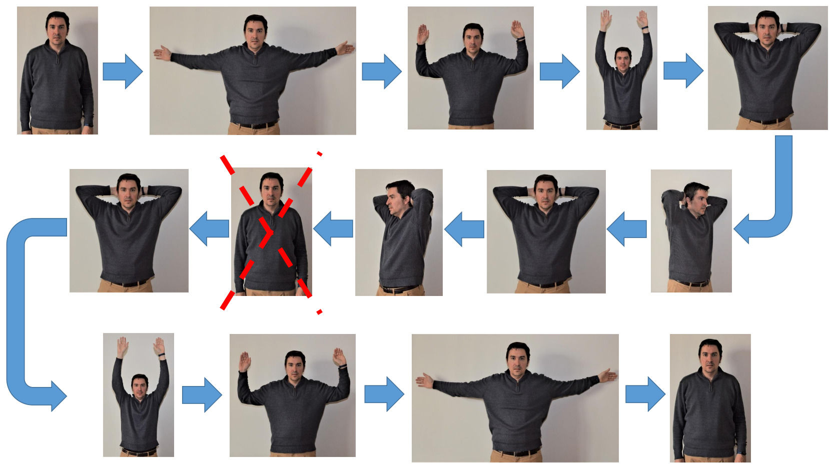


Figure 5: Example of sequence of an exercise with a posture incorrectly detected by the MLP and rectified by the HMM

The hidden Markov model (HM) is defined as $HM = \{S, V, A, B, \pi\}$, where

1. S is the finite set of states $S = S_1, S_2, \ldots, S_N$ where N is the number of unobservable states. In this case, the states corresponding to different body postures, arms up, arms down, etc. previously shown in Figure 2.
2. V is the set of M observations by state $V = V_1, V_2, \ldots V_M$.
3. A is the state transition probability distribution $A = \{a_{ij}\}$, where a_{ij} is the probability that the state at time $t+1$ is S_j, since the state at time t is S_i, that is $p(x_{t+1} = S_j | x_t = S_i)$. This stochastic matrix A defines the connection structure of the model. In our case, each matrix element represents the probability that a posture (state) at a frame will be another posture at next frame, i.e., the arms behind head, in the next frame will be the arms up. Some transitions can be zero or close to zero, as they are highly improbable; for example, it is very difficult to go directly from having your arms up to having your arms down without going through other intermediate states (e.g., arms in the middle or akimbo). If a coefficient a_{ij} is zero, it will remain zero even through the training process, so there will never be a transition from state S_i to S_j. This matrix is experimentally determined from a set of video sequences that determine, in a supervised way, the quotient between the number of transitions from one state i to another j and the number of transitions from one state i to all states including state j. It must be verified that $\sum_{j=1}^{N} a_{ij} = 1$, with $1 \leq i \leq N$. The transition matrix for the different trained postures is shown in Table 1. It has been computed from a set of recorded videos with the possible movements, recording the postures (From/To). After labeling the different transitions by the user, the different probability transitions are calculated from the video sequences (e.g., a_{ij} = number of transitions from S_i to S_j / number of transitions from S_i to any other state S_k).

From/To	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**	**P11**	**P12**	**P13**	**P14**	**P15**	**P16**	**P17**	**P18**	**P19**
P1	0.4	0.1	0	0	0	0.1	0	0	0	0.1	0	0.1	0.1	0	0	0	0.1	0	0
P2	0.03	0.49	0.03	0.03	0.03	0.03	0.03	0.03	0.01	0.2	0.01	0.02	0.02	0.01	0	0	0.03	0	0
P3	0	0.2	0.5	0	0	0.1	0.05	0.05	0	0	0	0	0	0.1	0	0	0	0	0
P4	0	0.1	0	0.5	0	0	0.2	0	0	0	0.1	0	0.1	0	0	0	0	0	0
P5	0	0.1	0	0	0.5	0	0	0.2	0.1	0	0	0.1	0	0	0	0	0	0	0
P6	0.15	0.15	0.15	0	0	0.55	0	0	0	0	0	0	0	0	0	0	0	0	0
P7	0	0.15	0.15	0.15	0	0	0.35	0	0	0	0.15	0	0.05	0	0	0	0	0	0
P8	0	0.15	0.15	0	0.15	0	0	0.35	0.15	0	0	0.05	0	0	0	0	0	0	0
P9	0	0.1	0	0	0.1	0	0	0.1	0.6	0	0	0.1	0	0	0	0	0	0	0
P10	0.1	0.1	0	0	0	0.1	0	0	0	0.4	0	0.1	0.1	0	0	0	0.1	0	0
P11	0	0.1	0	0.1	0	0	0.1	0	0	0	0.6	0	0.1	0	0	0	0	0	0
P12	0.1	0.1	0.1	0	0.1	0	0	0.1	0.1	0.1	0	0.2	0	0	0	0	0.1	0	0
P13	0.1	0.1	0.1	0.1	0	0	0.1	0	0	0.1	0.1	0	0.2	0	0	0	0.1	0	0
P14	0	0.08	0.12	0	0	0.1	0.05	0.05	0	0	0	0	0	0.3	0.15	0.15	0	0	0
P15	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	0.6	0	0	0	0
P16	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	0	0.6	0	0	0
P17	0.08	0.08	0	0	0	0	0	0	0	0.18	0	0.08	0.08	0	0	0	0.3	0.1	0.1
P18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	0.6	0
P19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	0	0.6

Table 1: Transition Matrix of the Viterbi algorithm

4. B is the probability distribution matrix of observations in each state $B = \{b_i(k)\}$, where $b_i(k)$ is the probability that the observation V_k occurs in the state S_i. Then $b_i(V_k) = p\{o_t = V_k | x_t = S_i\}$, $1 \leq i \leq N$, $1 \leq k \leq M$,

where V_k denotes the k^{th} observation, and o_t is the current distribution vector at time t. In our case, o_t is the probability distribution obtained at the output of the neural network.

5. π is the initial state distribution $\pi = \{\pi_i\}$, where π_i is the probability that the model is in state S_i at the time $t = 0$. In this case, since there is no clue about what the initial person's pose state is, we consider an initial uniform distribution $\pi_i = 1/N$.

Our problem is to find the most likely sequence of body pose states $X = (x_1, \ldots x_t, \ldots x_T)$ in the given model that produces the given observations $O = (o_1, \ldots, o_t, \ldots, o_T)$. This is known as the decoding problem that is solved by the Viterbi algorithm [29]:

$$\delta_t(j) = \max_{(x_1, x_2, \ldots x_{t-1})} p(x_1, x_2, \ldots, x_{t-1}, x_t = S_j, o_1, o_2, ..., o_t|\gamma) \tag{4}$$

where $\delta_t(j)$ represents the highest probability that the partial observation sequence and the state sequence up to $t = t$ can have, given that the current state is S_j and that $\gamma = (A, B, \pi)$ is the hidden Markov model.

The initial term is:

$$\delta_1(j) = \pi_j b_j(o_1), \quad 1 \leq j \leq N \tag{5}$$

The rest of the terms can be calculated recursively from the final estimation state

$$\delta_{t+1}(j) = b_j(o_{t+1})[\max_{1 \leq i \leq N} \delta_t(i)a_{ij}], \quad 1 \leq i \leq N, \quad 1 \leq t \leq T-1 \tag{6}$$

$$\varphi_{t+1}(j) = arg \max_{1 \leq i \leq N} \delta_t(i)a_{ij}, \quad 1 \leq j \leq N, \quad 1 \leq t \leq T-1 \tag{7}$$

$$x^*_T = arg \max_{1 \leq i \leq N} \delta_t(i) \tag{8}$$

We can backtrack to obtain the most likely sequence

$$x^*_t = \varphi_{t+1}(x^*_{t+1}), \quad 1 \leq t \leq T-1 \tag{9}$$

The main problem of the Viterbi algorithm is that processing the whole sequence of observations is required to obtain the most likely sequence of states. This can be a drawback when dealing with very long observation sequences, such as video frames that can last several minutes. Waiting to process all the frames limits the possibility of real-time recognition and makes feedback difficult, for example in the case where a robot or a monitoring system must warn the user that he/she is not performing the exercises correctly. To avoid these situations, we have defined a dynamic window of observations that moves through the entire observation sequence. Thus, given the observation sequence $O = (o_1, , o_t, \ldots, o_T)$, we define a dynamic window of observations:

$$O_w = \begin{cases} o_1, \ldots, o_{t-1}, o_t, o_{t+1}, \ldots, o_{t+w} & w > t, \quad t+w < T \\ o_{t-w}, \ldots, o_{t-1}, o_t, o_{t+1}, \ldots, o_{t+w} & w < t, \quad t+w < T \\ o_{t-w}, \ldots o_{t-1}, o_t, o_{t+1}, \ldots, o_T & w < t, \quad t+w > T \end{cases} \tag{10}$$

Applying the Viterbi algorithm at each time step t with $O_w(t)$ as input, we obtain the x_t^* likely state sequence $(x_{t-w}, \ldots, x_t, \ldots x_{t+w})$. This way, it is not necessary to process the entire sequence, it is only necessary to obtain w frames ahead. We have experimentally observed that a value of $w = 10$ (i.e., 0.5 seconds for a frame frequency of $20Hz$) leads to the same results as those obtained when processing the entire sequence.

3.3. Modified Levenshtein distance

Once the poses have been identified and the sequence determined by the HMM, it is necessary to identify the exercise performed by the person. For this, it is necessary to compare the identified sequence with the different sequences of exercises initially established. In our system, a modified Levenshtein distance has been used. During the preliminary experiments, several distances were tested, such as Levenshtein [31], Damerau-Levenshtein [32] or Jaro-Winkler [33, 34]. In our problem, they all behaved in a similar way, even though they had a common problem regarding the consideration of the nature of the postures in an exercise.

Levenshtein's distance is a magnitude which shows the minimum number of operations required to transform a character string into another [31]. It has been widely used for many different problems, most of them related to language: plagiarism detection [35], cryptanalysis [36], handwriting recognition [37] or even hieroglyph recognition [38].

To our knowledge, the Levenshtein distance has not previously been used with video pose detection. The only noteworthy works in this line are proposed by Poulisse and Moens [39], who detected scenes in Olympic videos using the audio and visual characteristics, although Levenshtein was only used to identify named entities from the audio, such as person's names; and Jarodzka et al. [40], who used it to quantify differences in perceiving and interpreting dynamic visual stimuli between experts and novices comparing their eye movements.

In our problem, an alphabet $\sum = \{a, b, c, ..., s\}$ has been defined to assign each symbol to a specific posture, as seen in Table 2.

Symbol	Posture in HMM transition matrix	Description
a	P1	Arms up
b	P2	Medium arms
c	P3	Arms down
d	P4	Left arm up
e	P5	Right arm up
f	P6	Arms forward
g	P7	Left arm in the middle
h	P8	Right arm in the middle
i	P9	Right arm half up/Left down
j	P10	Arms half up
k	P11	Left arm half up/Right down
l	P12	Right arm half up
m	P13	Left arm half up
n	P14	Arms akimbo
o	P15	Arms akimbo to the left
p	P16	Arms akimbo to the right
q	P17	Arms behind head
r	P18	Arms behind head to the left
s	P19	Arms behind head to the right

Table 2: Symbols assigned to each posture

The universal language of this alphabet, $\omega(\sum)$, represents the output of the HMM algorithm. The words of this language are sequences of postures obtained from the processed frames, i.e., $cccbbjjjjjaaaajjbbbbcc$. A simplified word $c^3b^2j^5a^4j^2b^4c^2$ represents the postures and number of repetitions of the previous example. The number of repetitions is transformed into seconds using the processing rate, this is the number of postures obtained per second. For a $20Hz$ rate, the previous word is transformed into $c^{0.15}b^{0.10}j^{0.25}a^{0.20}j^{0.10}b^{0.20}c^{0.10}$, dividing the repetitions by the rate. The word without time or repetitions is used with the modified Levenshtein distance to be compared with a predefined list of exercises. For the previous example, the word used is $cbjajbc$. The time is used as a metric in the evaluation of the exercise. If the person has to hold the arms up for 5 seconds, this will be compared with the obtained seconds. The list of 10 predefined exercises is shown in Table 3. The exercise of the preceding example corresponds to the first exercise, although the execution has been much faster than that defined in the table.

Identifier	Word ($Posture^{Seconds}$)
1	$c^2b^2j^1a^3j^1b^2c^2$
2	$c^2b^2j^1a^1q^2r^1q^1s^1q^1r^1q^1s^1q^2a^1j^1b^2c^2$
3	$c^2n^2o^1n^1p^1n^1o^1n^1p^1n^2c^2$
4	$c^2g^2k^1d^3k^1g^2c^2$
5	$c^2h^2i^1e^3i^1h^2c^2$
6	$c^2b^2f^2b^2c^2$
7	$c^2g^2k^2m^2d^3k^2g^2c^2$
8	$c^2h^2i^2l^2e^3i^2h^2c^2$
9	$c^2g^2k^2m^2j^2a^3j^2l^2i^2h^2c^2$
10	$c^2b^2j^1a^2q^2a^2q^2a^2q^2a^2j^1b^2c^2$

Table 3: Predefined exercises

The exercises have different durations; thus the Levenshtein distance has to be normalized, balancing the result of the algorithm by $max(|w_o|, |w_t|)$, where w_o is the word corresponding to the observed sequence and w_t the word of the defined exercise in the table. $|w_o|$ and $|w_t|$ are the lengths of the words w_o and w_t, respectively. This normalization leads to values within the range $[0,1]$.

As explained before, some problems were detected with the Levenshtein distance in our experiments. The problem was that a word like *cbjajqaqjajqjajq jajbc* was detected as *cbjaqrqsqrqsqajbc* instead of *cbjaqaqajbc*. The problem with this distance algorithm and others [32, 33, 34] is that they do not take into account the elements themselves; so a new modified Levenshtein distance is proposed to take into account both the Levenshtein distance and the existence of the postures that the exercise has in the table. For example, if the current exercise is *cbjaqaqajbc*, the input word should have 2 c, 2 b, 2 j, 3 a and 2 q, and if it is *cbjaqrqsqrqsqajbc*, it should have 2 c, 2 b, 2 j, 2 a, 5 q, 2 r and 2 s. In the same way, detected exercises that include postures that should not be in the recorded exercise are penalized. The new distance is calculated as shown in Equation 11, where $P_t = \{x \mid x \in w_t\}$, the set of different postures in w_t, i.e., $P_t = \{c, b, j, a\}$ for an exercise $w_t = cbjajbc$, and $\#P_t$ is the cardinality of P_t. $P_o = \{x \mid x \in w_o\}$ represents the symbols of the observed exercise. $R(x, w_t)$ is a function that returns the number of repetitions of the specified posture, x, in the word w_t. The new distance is the result of the balanced sum of the normalized Levenshtein distance and the new expression.

$$\begin{aligned} mLev(w_o, w_t) &= \tfrac{1}{2} norm(lev_{w_o,w_t}(|w_o|, |w_t|)) \\ &+ \tfrac{1}{4}\left[\frac{\sum\limits_{x \in P_t} \max\left(\left|\frac{R(x,w_o)-R(x,w_t)}{R(x,w_t)}\right|, 1\right)}{\#P_t}\right] \\ &+ \tfrac{1}{4}\left[\frac{\sum\limits_{x \notin P_t \wedge x \in P_o} [R(x,w_o)]}{|w_o|}\right] \end{aligned} \tag{11}$$

where $lev_{w_o,w_t}(|w_o|, |w_t|)$ is obtained recursively according to the expression:

$$lev_{w_o,w_t}(i,j) = \begin{cases} \max(i,j) & if \min(i,j) = 0 \\ \min \begin{cases} lev_{w_o,w_t}(i-1,j)+1 \\ lev_{w_o,w_t}(i,j-1)+1 \\ lev_{w_o,w_t}(i-1,j-1)+1_{(w_{oi} \neq w_{tj})} \end{cases} & otherwise. \end{cases} \tag{12}$$

where $1_{(w_{oi} \neq w_{tj})}$ is the indicator function with value 1 when $w_{oi} \neq w_{tj}$ and 0 otherwise. $lev_{w_o,w_t}(i,j)$ is the distance between the first i characters of w_o and the first j characters of w_t.

The activity is considered to be an exercise of the table only when the result is over a given threshold. When the identifier of the exercise has been obtained, an evaluation is carried out comparing the time the person has stayed in each posture of the exercise and the optimal time registered in the table.

Simultaneously, the complete time of the exercise is returned to the user as a feedback to show if the person should increase/decrease the speed.

Another important aspect is taken into account for this process. The system considers the posture *arms down* as an initial and final step. The recording of the exercise activity begins when the user has the *arms down* and finishes when, after other postures, the person repeats said posture. The modified Levenshtein algorithm is launched when two postures are detected. When the modified Levenshtein algorithm finishes, an evaluation of the performance of the exercise is carried out. If the time observed in performing an exercise is outside a certain tolerance around the optimal time recorded in the table, say 10%, the system will report that the execution has been too fast or slow, depending on the case. Otherwise, the system will report that the execution speed has been correct.

4. Experiments

Three experiments have been carried out to evaluate the performance of the MLP network, the HMM and the Modified Levenshtein distance. As shown in Figure 6, the MLP provides the sequence of postures, the HMM filters out erroneous postures and the MLD classifies the sequence into a specific exercise.

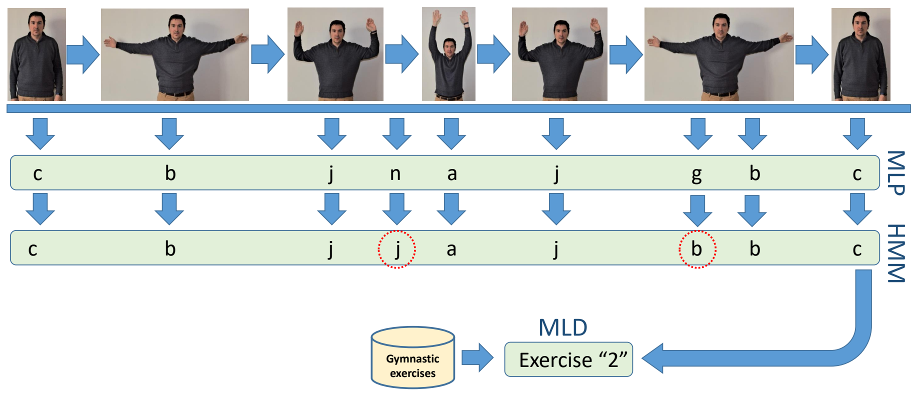


Figure 6: Steps of the overall process

The steps of the experimentation have been:

1. MLP network: 129 videos (101 minutes and 180, 958 samples/frames) have been recorded on 16 people performing the different postures (see Figure 7). 116 of these videos (66 minutes and 118, 718 samples/frames), corresponding to 3 people, have been used for training. The rest of the videos (35 minutes and 62, 240 samples/frames), recorded on the other 13 people, have been used to intensively evaluate the model.

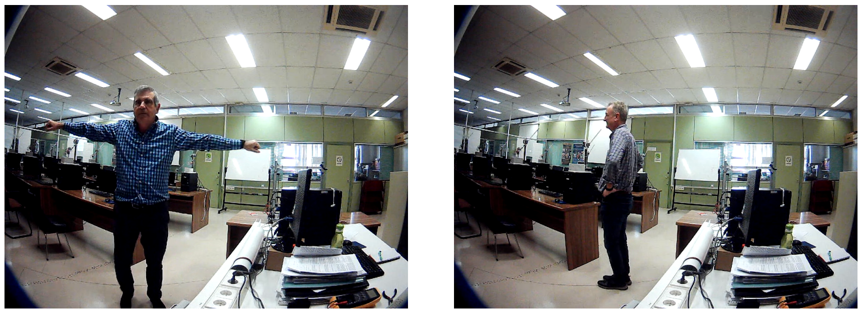

(a) Person with posture *b* (b) Person with posture *s*

Figure 7: Two people carrying out the postures

The network has been trained by splitting the previously balanced data, corresponding to the 116 training videos, into train (60%), validation (20%) and test (20%) datasets. The obtained model has taken 217 seconds to be trained in an i9-9900K/32Gb with a GPU RTX2080-TI.
In addition to our MLP training, a comparison with SVM [41] has been carried out. After the neural network training, a test with several videos recorded by 13 different people has been conducted (35 minutes and 62, 240 frames). To avoid data correlation, these people have not participated during the training phase (see Figure 8). In these videos, people have performed several complete exercises. The participants have been 2 female (39 age avg.) and 11 male (32 age avg.).

Figure 8: Different people carrying out the exercises

After recording these videos, an external observer has manually labeled the postures of each person in each frame. When this manual process was completed, the videos have passed the MLP network and the obtained results have been compared with the manual labels. Figure 9 shows a frame of one of these videos. The frame number is up on the left, followed by the manual label, the output of the MLP network and the output of the HMM model.

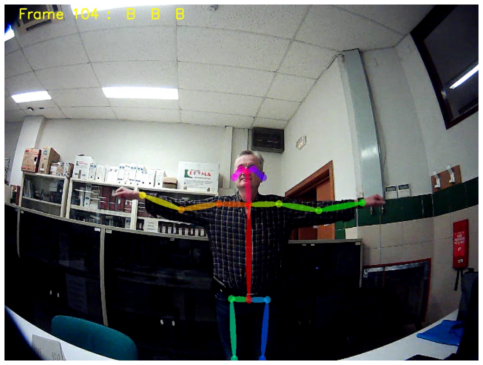


Figure 9: Labeled person with posture b

2. Hidden Markov Model: The Viterbi algorithm has been evaluated on the videos recorded after the training of the MLP network. The manual labels have been used to evaluate the right output. The Viterbi algorithm offers two main contributions: it rectifies spurious postures and removes wrongly identified movements, derived from joints incorrectly detected by OpenPose. OpenPose works frame by frame and Viterbi works with a sequence of frames to detect what was wrong. In Figure 10, a person is in the s posture, while the output of the neural network is a wrong r posture. This problem arises because certain joints have not been identified by OpenPose due to self occlusion, and happens in spite of the extensive MLP training dataset. Viterbi detects that it is pretty unlikely to go from s to r without going through q if the previous posture has been s and blocks the movement in s, having received an equally high probability of being in the posture s. As seen in the previous transition matrix, a transition between the posture s and r is 0.0, while the transition between s and s is 0.6 and between s and q is 0.4.

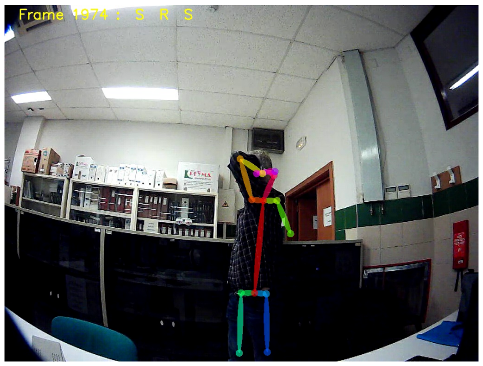


Figure 10: Incorrect posture detection by MLP rectified by HMM

An additional experiment has allowed us to reproduce the output of this layer in a Pepper robot, repeating 38 previously recorded exercises, as shown in the scheme of Figure 11. Figure 12 shows how the sequence obtained by MLP is filtered by HMM, stored and replayed on the Pepper robot.

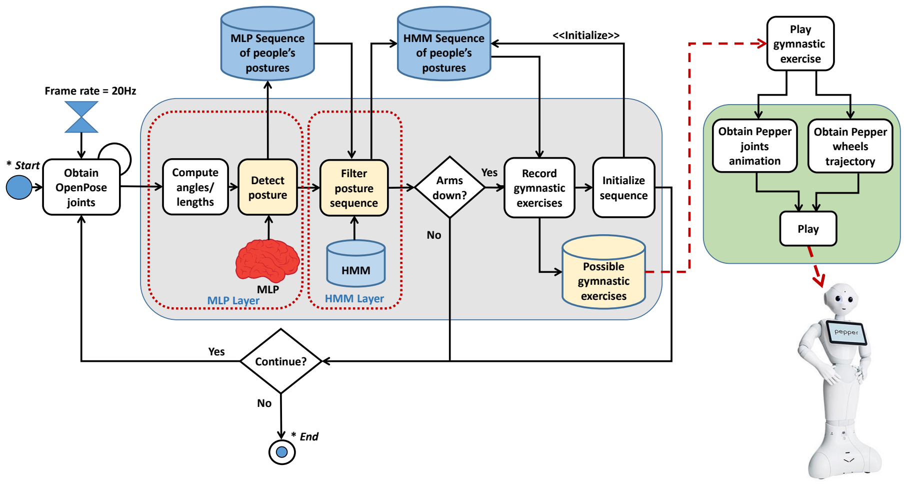


Figure 11: Scheme of the integration with Pepper robot

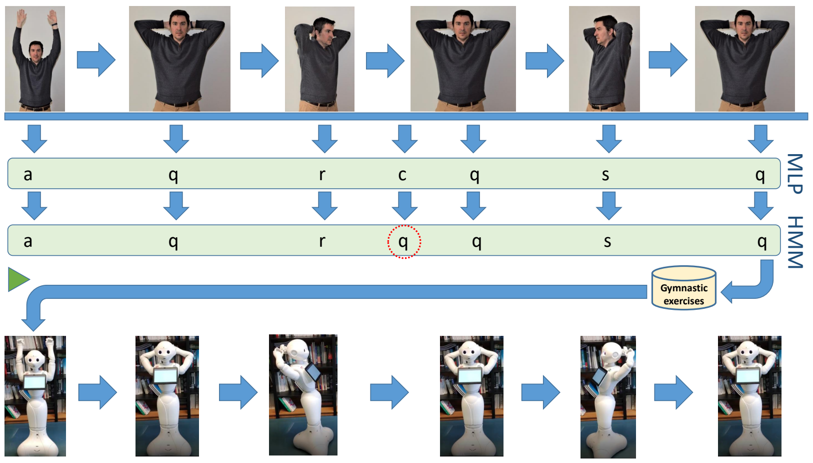
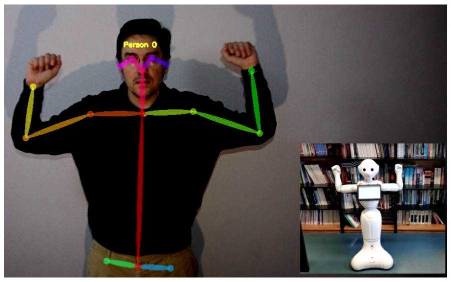


Figure 12: Steps of the process with the Pepper robot

The system considers the posture *arms down* as an initial and final step to record an exercise. These exercises are stored in the *Possible Gymnastic Exercises* database and reproduced by the Pepper robot. This reproduction obtains the joint movements and the trajectory (wheel movement) using the sequence of postures and time. The wheels are used to turn the robot left/right as it does not have a joint to carry out these turns. Figure 13(a) shows the Pepper robot performing the exercises that have previously been recorded and Figure 13(b) shows the robot performing an exercise in a residence for the elderly.



(a) Robot repeating the exercises

(b) Robot carrying out exercises in a residence for the elderly

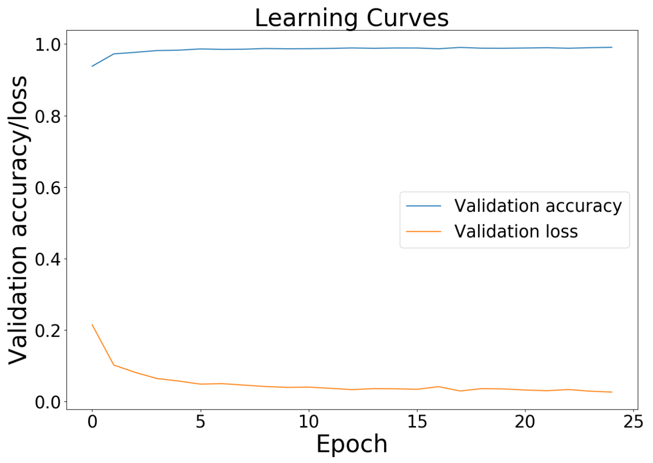
Figure 13: Experiments with Pepper robot

Pepper robot is able to play two kinds of animations. Firstly, there is a sequence of joint movements. For each step of this sequence, there is a list of 17 joints (HeadYaw, HeadPitch, LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll, LWristYaw, LHand, HipRoll, HipPitch, KneePitch, RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll, RWristYaw and RHand) for which a rotation angle is given as well as the number of frames that the joint will stay in that position. Secondly, Pepper robot has three omni-directional wheels that function together to generate motion. The wheels allow to move the robot in the plane but also to make turns. As with the joints, there is a sequence of movements that includes translations and rotations. The transformation of the exercises in smooth robot animations is carried out by obtaining the number of frames that the robot should stay in each posture and taking into account the time. Said time in each posture is transformed into a number of frames of the sequence. At the beginning of the animation, the frame rate is specified. Simultaneously, the animation corresponding to the movement of the wheels is generated in the postures that require rotations. To avoid abrupt transitions, only postures with duration over 1 second are considered. For example, a posture b is transformed into a sequence with several joint rotations: LShoulderRoll: $+90°$, LElbowYaw: $-90°$, RShoulderRoll: $-90°$ and RElbowYaw: $+90°$. The rest of joints in this example are equal to $0°$.

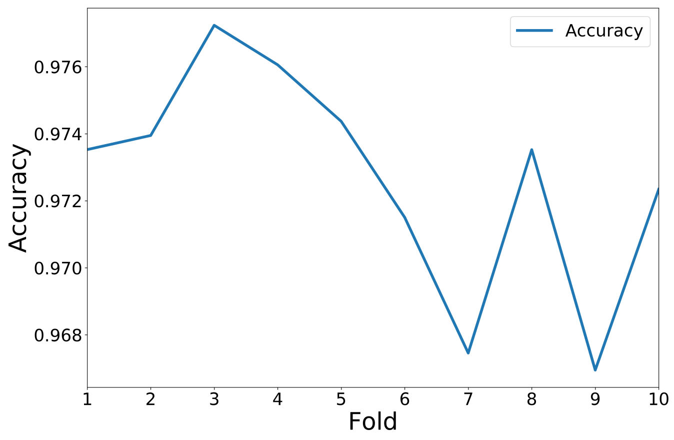
3. Modified Levenshtein distance: A quantitative analysis has been carried out to show how many times the algorithm adequately detects the right exercise from a set of videos with 154 exercises. Different distances have been evaluated to be compared with our MLD identifying the complete exercises. The accuracy results obtained have been compared with the distance algorithms proposed by Levenshtein [31], Damerau-Levenshtein [32] and Jaro-Winkler [33, 34]. The input of these four algorithms (MLD, Levenshtein, Damerau-Levenshtein and Jaro-Winkler) has been the output of the HMM.

5. Results and discussion

The performance of the MLP network on the test dataset (20% of samples) has been evaluated, leading to an accuracy figure of 96.9%. Figure 14a shows the validation accuracy and loss during the training. The number of epochs has been 25, with a batch size of 32. After the training, a k-fold cross-validation has been carried out splitting the complete dataset into $k = 10$ groups, where one of the groups is used as the test set and the rest are used as the training set. This technique has allowed the model performance to be evaluated against different data sets. As seen in Figure 14b, the proposed MLP has an accuracy about 97% for all the groups (average accuracy of $97.27\% \pm 0.32$ and a loss of 0.09). A multi-class precision-recall curve, with all classes, is displayed in Figure 15.



(a) Validation accuracy and loss



(b) k-fold cross-validation

Figure 14: Validation and evaluation of the model

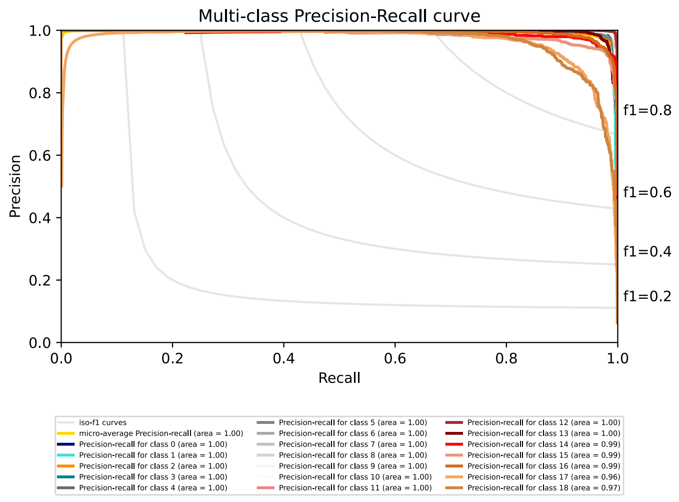


Figure 15: Multi-class precision-recall curve

In addition to our MLP training, a comparison with SVM [41] has been carried out. After an exhaustive grid search, SVM has been executed with the same inputs and outputs of our MLP and the parameters $Cost = 100$ and $gamma = 100$. SVM has been trained with 80% of the dataset while the another 20% has been used for test. The accuracy of SVM has been 96.22%. A K-fold cross-validated paired t-test has been computed, resulting in values of $t = -26.21$ and $p = 8.64 \cdot 10^{-16}$. Considering $\alpha = 0.05$, there is a significance difference between the two models. Figure 16 shows the box-plots of the obtained scores during K-fold cross-validated paired t test.

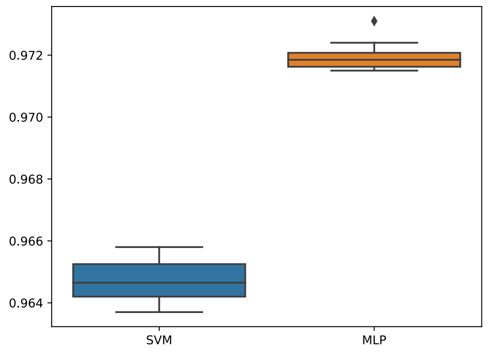


Figure 16: Box-plot of the scores obtained during K-fold cross-validated paired t test

After the neural network training, a test with several videos recorded by 13 different people has been conducted (35 minutes and $62,240$ frames). The result of the correctly classified frames has been 96.30%. Table 4 shows the confusion matrix between the estimated posture obtained by the MLP and the real one. Most detections work fine although some errors may occur due to OpenPose detection errors.

Real \ Estimated	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
a	0.98	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.00	0.96	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
e	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
f	0.01	0.00	0.01	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00
g	0.00	0.00	0.03	0.03	0.00	0.00	0.93	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00
k	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l	0.11	0.00	0.00	0.00	0.01	0.00	0.00	0.02	0.21	0.04	0.00	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.00
m	0.00	0.00	0.00	0.12	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.86	0.00	0.00	0.00	0.00	0.00	0.00
n	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.01	0.01	0.00	0.00	0.00
o	0.00	0.00	0.14	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.8	0.00	0.00	0.02	0.00
p	0.00	0.00	0.06	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.89	0.00	0.00	0.00
q	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
r	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.12	0.86	0.00
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.1	0.00	0.9

Table 4: Confusion Matrix of the postures obtained by the MLP network (%)

In our videos, we have quantified that Viterbi improves the accuracy of the detection. It is able to rectify 38% of the wrong detections, passing from an accuracy of 96.30% in MLP to 97.73% with HMM. However, in a small amount of cases, Viterbi modifies a correct value obtained by the MLP network. Table 5 shows that MLP and HMM outputs are both correct in 96.30% of cases, HMM has properly rectified the MLP output in 1.43% of cases, HMM has wrongly rectified the MLP output in 0.11% of cases and both of them have been incorrect in 2.16% of cases.

HMM \ MLP	Incorrect	Correct
Incorrect	2.16%	0.11%
Correct	1.43%	96.30%

Table 5: MLP and HMM results (%)

Table 6 shows the confusion matrix between the estimated posture obtained by Viterbi and the real one. Comparing this matrix with Table 4, it is possible to observe that most of the detections have been improved.

Real \ Estimated	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
a	0.98	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.00	0.97	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
e	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
f	0.01	0.00	0.01	0.00	0.00	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00
g	0.00	0.00	0.03	0.04	0.00	0.00	0.92	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00	0.03	0.00	0.00	0.96	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l	0.1	0.00	0.00	0.00	0.02	0.00	0.00	0.01	0.22	0.04	0.00	0.61	0.00	0.00	0.00	0.00	0.00	0.00	0.00
m	0.00	0.00	0.00	0.15	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.83	0.00	0.00	0.00	0.00	0.00	0.00
n	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.96	0.01	0.01	0.00	0.00	0.00
o	0.00	0.00	0.14	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.84	0.00	0.00	0.00	0.00
p	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.00
q	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00
r	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.95	0.00
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.91

Table 6: Confusion Matrix of the postures obtained by the HMM (%)

The experiments carried out with the Pepper robot, repeating 38 previously recorded exercises, have shown that the robot reproduces the postures correctly 93.3% of the time. The synchronization is not complete due to certain limitations of the Pepper's kinematics, especially in turns.

The quantitative evaluation of the Modified Levenshtein distance (MLD) identifying the complete exercises and the comparison with the distance algorithms proposed by Levenshtein [31], Damerau-Levenshtein [32] or Jaro-Winkler [33, 34] are shown in Table 7. The modified Levenshtein distance properly identified 98.05% of the complete exercises.

Distance	**Accuracy**
MLD (Modified Levenshtein distance)	98.05%
Levenshtein distance [31]	94.15%
Damerau-Levenshtein distance [32]	94.15%
Jaro-Winkler distance [33, 34]	91.56%

Table 7: Comparison of distances in the classification of exercises (%)

The result of identifying partial exercises is presented in the confusion matrix of Table 8. Most errors were produced when too few postures had been obtained, say 2 or 3. The cells with a zero value represent exercises which have not been incorrectly detected. The overall accuracy identifying exercises which have not been completed is 94.6%. To evaluate incomplete exercises, only words with a minimum of 5 postures have been considered. When the person performs 5 postures or more, the system begins to identify the exercise.

Estimated / Real	1	2	3	4	5	6	7	8	9	10
1	0.91	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09
2	0.07	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.26
3	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00	0.00
8	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00	0.00
9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.0	0.00
10	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.88

Table 8: Confusion Matrix of the modified Levenshtein algorithm, recognizing partial exercises (%)

When the quantitative analysis was finished, a qualitative analysis showed that the system works well in a real-time scenario, receiving the feedback of the system, as seen in Figure 17, where all the information is shown: posture, final/partial exercise and the recommendation for the final and partial result. As previously stated, when the performance of an exercise is outside a 10% tolerance around the optimal time recorded in the table, the system reports that the execution has been too fast or slow, depending on the case.

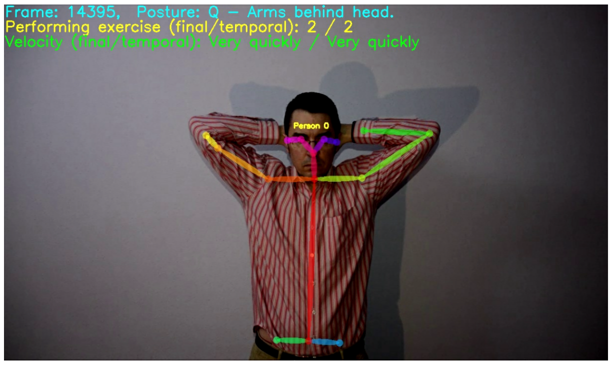

Frame: 14395, Posture: Q – Arms behind head.
Performing exercise (final/temporal): 2 / 2
Velocity (final/temporal): Very quickly / Very quickly
Person 0

Figure 17: Final result showing the posture, final/partial exercise and the recommendation

Our results are better than other known works [20] and with lower computational cost. Costa et al. [20] showed a high accuracy in terms of detecting and classifying the physical exercises (97.35%), but in our work, 98.05% of complete exercises have been properly detected. Regarding other works that use RNNs, our work obtains better results than [18], whose model obtained an average accuracy of 92.4%.

Our approach computes the angular amplitudes and modules of the links to identify the postures. This reduces the computational cost compared to other alternatives such as end-to-end applications, which pass raw data from one layer to another without processing (i.e., Convolutional Neural Networks (CNN)). When the arithmetic complexity is evaluated, Costa et al. [20] use a C2R model, composed of a CNN (ResNet-50 [42]) and a Gated Recurrent Unit

(GRU) [43]. ResNet-50 has 23.5 million parameters, result of several consecutive convolution layers. GRU has 32 units connected to a dense layer with 100 outputs. They process 47 frames to detect each exercise. In total, considering a minimum of one arithmetic operation per parameter, the arithmetic complexity is about $O(47 \cdot 24million) = O(1,128million)$. In that case, GPU parallel computing is required. In our system, the MLP has $39,546$ parameters (20 inputs, $1,014$ hidden neurons and 19 outputs). HMM is an iterative process that considers 10 steps with 19 possible postures. We have approximately estimated 741 parameters. Finally, although MLD is recursive, it has a small number of operations since the strings to be compared are short. Considering a maximum number of 300 operations per comparison, MLD would have $3,000$ operations to compare a string with the 10 possible exercises. When 47 frames are processed, our arithmetic complexity is about $O(47 \cdot (39,546+741+3,000)) = O(2million)$. Both Costa et al. [20] and our work use OpenPose, so the above calculations refers to the part of the system once the skeletons have been obtained. OpenPose uses a convolutional network composed of 25 million parameters to obtain the separate body parts that are are linked by solving an optimization problem on Part Affinity Fields (PAFs). The arithmetic cost of the optimization algorithm is given by the number of operations required to perform K iterations of the algorithm and also depends on the number of detected body parts and people. The cost is variable depending on the convergence of the problem. Our system needs to learn less parameters than other end-to-end approaches and is able to generalize with the learning of only three people. Costa et al. [20] trained their C2R model with $159,990$ samples of 10 images each, and it was validated against a validation dataset of $79,995$ samples. 7 people participated in their experiments. Our system does not require as much data as them or as heavy in size because it uses the angular amplitudes and modules of the links to identify the postures. In contrast, Costa et al. [20] directly uses skeleton images, which results in much more data at the input of the model.

The software, models and videos with the gymnastic activity recognition and robot movements are available on the Internet at the URL:
https://github.com/jaiduqdom/gymnasticactivity.git.

6. Conclusions

This work presents a novel software architecture to autonomously identify and evaluate the gymnastic activity that people are carrying out.

The architecture is composed of three different and interconnected layers, a Multilayer Perceptron (MLP) neural network, a Hidden Markov Model (HMM) layer and an innovative Modified Levenshtein Distance (MLD) layer. The MLP layer detects the posture of people according to the joints provided by the OpenPose library. However, this library works frame by frame, which can result in detection errors due to occasional occlusions. The HMM and the Viterbi algorithm allow incorrectly obtained postures to be rectified. Finally, different distance algorithms have been considered for comparing the detected sequences to reference ones [31, 32, 33, 34]. However, they do not consider the nature of the symbols, so a novel Modified Levenshtein Distance has been presented. This new distance allows the detection of the exercises to be improved and provides continuous feedback to the users. The integration of the three components results in a robust system that has allowed us to improve the results of previous works. Our MLP has an accuracy of 96.30%, which is improved by the HMM to 97.73%. The MLD has been able to correctly identify all completed exercises in 98.05%. The integration of the three components allows us to obtain a robust system that has allowed us to improve the results of previous works.

The proposed system can be useful for evaluating users' performance while they are physically exercising. In this way, it is possible to give users feedback in real time, so they can do the exercise correctly (e.g., raise their arms higher, do the exercise faster, or more slowly, etc.). Another of the functionalities is the programming of robots by demonstration. This article has shown how, once a sequence of exercises has been learned, it can be reproduced by an autonomous Pepper robot without explicit programming. This allows considerable time savings in addition to achieving greater synchronization of movements between the exercise proposed by the master and the robot. In this case, the proposed exercises have been implemented on the Pepper robot in a nursing home.

Until now, the proposed system has allowed us to quickly program by demonstration the exercises to be performed by the robot in the nursing home. This avoids the need for explicit Pepper programming. As a future work, it is proposed to deploy the novel architecture in a social robotics environment with the elderly, so that the robot provides feedback to the users during the exercises.

CRediT author statement

J.D.D., J.G.G.B. and E.Z. have conceived and designed this research, carried out the experiments, analyzed the data and written the paper.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this manuscript.

Acknowledgments

The present research has been partially financed by "Programa Retos Investigación del Ministerio de Ciencia, Innovación y Universidades (Ref. RTI2018-096652-B-I00)" and by "Programa de Apoyo a Proyectos de Investigación de la Junta de Castilla y León (Ref. VA233P18)", cofinancied with FEDER funds.

References

[1] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, Y. Sheikh, Openpose: realtime multi-person 2d pose estimation using part affinity fields, arXiv preprint arXiv:1812.08008 (2018).

[2] A. K. Pandey, R. Gelin, A mass-produced sociable humanoid robot: Pepper: The first machine of its kind, IEEE Robotics & Automation Magazine 25 (2018) 40–48.

[3] J. Fasola, M. J. Mataric, Robot exercise instructor: A socially assistive robot system to monitor and encourage physical exercise for the elderly, in: 19th International Symposium in Robot and Human Interactive Communication, IEEE, 2010, pp. 416–421.

[4] P. Gadde, H. Kharrazi, H. Patel, K. F. MacDorman, Toward monitoring and increasing exercise adherence in older adults by robotic intervention: a proof of concept study, Journal of Robotics 2011 (2011).

[5] D. K. Vishwakarma, R. Kapoor, A. Dhiman, A proposed unified framework for the recognition of human activity by exploiting the characteristics of action dynamics, Robotics and Autonomous Systems 77 (2016) 25–38.

[6] M. Gil-Martín, R. San-Segundo, F. Fernández-Martínez, J. Ferreiros-López, Improving physical activity recognition using a new deep learning architecture and post-processing techniques, Engineering Applications of Artificial Intelligence 92 (2020) 103679.

[7] H. He, Y. Tan, W. Zhang, A wavelet tensor fuzzy clustering scheme for multi-sensor human activity recognition, Engineering Applications of Artificial Intelligence 70 (2018) 109–122.

[8] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, M. Barreto, Ubiquitous robotics: Recent challenges and future trends, Robotics and Autonomous Systems 61 (2013) 1162–1172.

[9] W. Y. Kwon, Y. Park, S. H. Lee, I. H. Suh, Human activity recognition using deep recurrent neural networks and complexity-based motion features, Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep (2015) 1–7.

[10] W. Y. Kwon, I. H. Suh, Complexity-based motion features and their applications to action recognition by hierarchical spatio-temporal naive bayes classifier, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 3141–3148.

[11] L. Piyathilaka, S. Kodagoda, Gaussian mixture based hmm for human daily activity recognition using 3d skeleton features, in: 2013 IEEE 8th conference on industrial electronics and applications (ICIEA), IEEE, 2013, pp. 567–572.

[12] B. Görer, A. A. Salah, H. L. Akın, A robotic fitness coach for the elderly, in: International Joint Conference on Ambient Intelligence, Springer, 2013, pp. 124–139.

[13] P. Tanguy, O. Rémy-Néris, et al., Computational architecture of a robot coach for physical exercises in kinaesthetic rehabilitation, in: 2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), IEEE, 2016, pp. 1138–1143.

[14] A. Lotfi, C. Langensiepen, S. W. Yahaya, Socially assistive robotics: Robot exercise trainer for older adults, Technologies 6 (2018) 32.

[15] J. Fasola, M. J. Mataric, Using socially assistive human–robot interaction to motivate physical exercise for older adults, Proceedings of the IEEE 100 (2012) 2512–2526.

[16] M. Ghandour, H. Liu, N. Stoll, K. Thurow, Human robot interaction for hybrid collision avoidance system for indoor mobile robots, Advances in Science, Technology and Engineering Systems (ASTES) Journal 2 (2017) 650–657.

[17] C. Dhiman, D. K. Vishwakarma, A review of state-of-the-art techniques for abnormal human activity recognition, Engineering Applications of Artificial Intelligence 77 (2019) 21–45.

[18] F. M. Noori, B. Wallace, M. Z. Uddin, J. Torresen, A robust human activity recognition approach using openpose, motion features, and deep recurrent neural network, in: Scandinavian Conference on Image Analysis, Springer, 2019, pp. 299–310.

[19] N. Granger, M. A. el Yacoubi, Comparing hybrid nn-hmm and rnn for temporal modeling in gesture recognition, in: International Conference on Neural Information Processing, Springer, 2017, pp. 147–156.

[20] A. Costa, E. Martinez-Martin, M. Cazorla, V. Julian, Pharos-physical assistant robot system, Sensors 18 (2018) 2633.

[21] M. Danelljan, G. Häger, F. Khan, M. Felsberg, Accurate scale estimation for robust visual tracking, in: British Machine Vision Conference, Nottingham, September 1-5, 2014, BMVA Press, 2014.

[22] D. S. Bolme, J. R. Beveridge, B. A. Draper, Y. M. Lui, Visual object tracking using adaptive correlation filters, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 2544–2550.

[23] M. Andriluka, L. Pishchulin, P. Gehler, B. Schiele, 2d human pose estimation: New benchmark and state of the art analysis, in: Proceedings of the IEEE Conference on computer Vision and Pattern Recognition, 2014, pp. 3686–3693.

[24] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7291–7299.

[25] S. Haykin, Neural networks: a comprehensive foundation, Prentice Hall PTR, 1994.

[26] L.-T. Ophélie, P. Charton, X. F. Cadet, B. Grondin-Perez, E. Saavedra, D. Cédric, C. Frédéric, Identification of flux checkpoints in a metabolic pathway through white-box, grey-box and black-box modeling approaches, Scientific Reports (Nature Publisher Group) 10 (2020).

[27] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[28] L. E. Baum, T. Petrie, Statistical inference for probabilistic functions of finite state markov chains, The annals of mathematical statistics 37 (1966) 1554–1563.

[29] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE transactions on Information Theory 13 (1967) 260–269.

[30] H.-L. Lou, Implementing the viterbi algorithm, IEEE Signal processing magazine 12 (1995) 42–52.

[31] V. Levenshtein, Leveinshtein distance, 1965.

[32] F. J. Damerau, A technique for computer detection and correction of spelling errors, Communications of the ACM 7 (1964) 171–176.

[33] W. E. Winkler, String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. (1990).

[34] W. W. Cohen, P. Ravikumar, S. E. Fienberg, et al., A comparison of string distance metrics for name-matching tasks., in: IIWeb, volume 2003, 2003, pp. 73–78.

[35] Z. Su, B.-R. Ahn, K.-Y. Eom, M.-K. Kang, J.-P. Kim, M.-K. Kim, Plagiarism detection using the levenshtein distance and smith-waterman algorithm, in: 2008 3rd International Conference on Innovative Computing Information and Control, IEEE, 2008, pp. 569–569.

[36] J. D. Golić, M. J. Mihaljević, A generalized correlation attack on a class of stream ciphers based on the levenshtein distance, Journal of Cryptology 3 (1991) 201–212.

[37] S. D. Chowdhury, U. Bhattacharya, S. K. Parui, Online handwriting recognition using levenshtein distance metric, in: 2013 12th International Conference on Document Analysis and Recognition, IEEE, 2013, pp. 79–83.

[38] J. Duque-Domingo, P. J. Herrera, E. Valero, C. Cerrada, Deciphering egyptian hieroglyphs: Towards a new strategy for navigation in museums, Sensors 17 (2017) 589.

[39] G.-J. Poulisse, M.-F. Moens, Unsupervised scene detection in olympic video using multi-modal chains, in: 2011 9th International Workshop on Content-Based Multimedia Indexing (CBMI), IEEE, 2011, pp. 103–108.

[40] H. Jarodzka, K. Scheiter, P. Gerjets, T. Van Gog, In the eyes of the beholder: How experts and novices interpret dynamic stimuli, Learning and Instruction 20 (2010) 146–154.

[41] M. E. Mavroforakis, S. Theodoridis, A geometric approach to support vector machine (svm) classification, IEEE transactions on neural networks 17 (2006) 671–682.

[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).