REAL-TIME MOVEMENT CLASSIFICATION AND ANALYSIS

FROM RGB – D VIDEO DATA

by

Mahya Saeednejad, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
May 2021

Committee Members:

Vangelis Metsis, Chair

Anne Hee Hiong Ngu, Co-Chair

Dan Tamir

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

DEDICATION

I dedicate this project to my family and many friends. A special feeling of gratitude to my loving parents, Amir and Mahnaz whose exceptional support and encouragement made it possible for me to pursue my master's degree in computer science at Texas State University. My sisters Mina and Mahshid have never left my side and are very special. I will always appreciate what they have done for me. I live a truly amazing life and it is all because of them.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest appreciation for my patient and supportive supervisor, Dr. Vangelis Metsis, for his dedicated support and guidance both as a professor and an advisor during my studies.

I would also like to extend my sincere thanks to Dr. Ngu and Dr. Tamir for deciding to join my commission and contributing their time and their expertise to this research. In addition, thanks to all faculty and staff members of the Department of Computer Science and the College of Engineering for providing an outstanding learning and research atmosphere with excellent facilities for teaching, learning and research.

I thank my friends and colleagues who helped me through this process and made this path easier for me.

Finally, I am extremely grateful to my family, whose love and support are always with me. My success would not have been possible without them.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviation	Description
NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
DTW	Dynamic Time Warping
WPF	Windows Presentation Foundation

ABSTRACT

Physical therapy is a form of rehabilitative treatment that includes specially designed and prescribed exercises to help patients recover from diseases that disturb their movements of daily life or enhance their physical abilities. In physical therapy, tracking the movements of different body parts and classifying the movements during the exercises has great importance. The goal of this thesis is to create an interactive system able to recognize physical therapy exercises from patients' movements and count the repetitions of them in real-time.

To achieve this objective, we used a combination of a Long Short-Term Memory (LSTM) model and Dynamic Time Warping (DTW) algorithm for the classification and counting the repetitions of exercises using 3D skeleton tracking data captured by Microsoft Kinect. First, we developed methods for preprocessing and normalization of a multi-dimensional skeleton sequence. Second, we explored ways of increasing the precision of offline human movement classification methods; finally, we created a real-time system for exercise recording and classification. In addition, we proposed a 3D DTW algorithm for 3D skeleton sequence pattern matching and counting the repetition of movements, and we designed and implemented an interactive application that tracks the actions of an individual, transfers information for processing, and displays the results.

I. INTRODUCTION

In a world where computers have a huge role in our lives, we always look for a way to make computers more intelligent and smarter. If computers had our five senses, they would be more aware of their environment and as a result they would improve our quality of lives even more. Since vision is one of our most important senses, the idea of making computers understand the visual clues is crucial to have more intelligent computers.

Computer vision wants to discover techniques to help computers 'see' and understand digital image content, such as photographs and videos. It is one of the hottest areas of computer science and artificial intelligence research due to its critical challenges and the range of its applications including virtual reality, surveillance systems, advanced user interfaces, and motion analysis.

Computer vision applications, acting as an additional pair of eyes, has had revolutionary impacts on healthcare sector by improving the speed and accuracy of medical diagnoses and treatments. One of the newly emerging computer vision application particularly human movement classification application is in physical therapy field. Physical therapy helps patients with injuries, disabilities, or other health conditions to improve their ability to move, reduce or manage pain, restore function, and prevent disability with prescribed exercises, hands-on care, and patient education.

Usually, patients need to visit physiotherapists regularly to repeat certain exercises under supervision of the therapist. Monitoring and evaluating the patients' performance is crucial for developing the treatment plan and to make sure that the exercises are practiced correctly and accurately as they are prescribed, since practicing

wrong exercises could trigger more pain and injury. Therefore, physical therapy treatment highly depends on the physician's interpretations and needs regular in-person visits that can be inconvenient and expensive for many patients. In this case, an automated system that can evaluate the patients' performance in real time can bring mobility for patients as they can practice the exercises at their preferred places, and they can constantly monitor their performance. In addition, using new and powerful hardware like Kinect sensors to capture the performances can remarkably reduce the costs of the treatment.

To achieve this automated system, we need to tackle two problems. First, the system should be able to identify the movement type, and second, it should determine how accurate the practiced exercise is compared to the prescribed exercise in real-time.

Statement of the Problem

In this research, we are concentrating on identifying the movement type in real-time. The purpose of this research is to detect and interpret human movements automatically in real-time from the information acquired from the Kinect sensor.

In real-time classification, the main problem is that we only have information from a few previous movements at a given time, and if we do not process or store them, we lose them. Therefore, the model needs to identify the exercise based on the few past bits of information and without any knowledge of the next contents. Also, it is quite a difficult task to identify the exercise in real time since certain movements are common between different exercises, and by knowing just few movements, we may not be able to pick only one label, because those movements might have multiple labels.

The other challenge is how data should be stored and processed. The data for human movement classification could be so large that makes the data transferring and processing procedure slow. Also, the data might need a lot of memory to store. Therefore, to handle this challenge, we might need to transfer and process the data portion by portion. An additional challenge is the procedure of recording, transferring the data and action recognition all should happen in reasonable timing. The speed of motion detection is even more important in high-risk situations, such as crime scenes where we need to act immediately. (e.g., when a gun is pointed at someone). Deep convolutional [1] and Long-Short Term Memory (LSTM) [2] neural networks have shown outstanding performance for classifying sequence data such as human movements.

However, the major problem with many of the existing methods is that they do not apply to real-time classifications [3][4][5][6]. In other words, these algorithms work efficiently when data is segmented, but in real-time movement, the data is unsegmented since the beginning and end of the action is unknown. There are a few kinds of research for real-time action recognition. However, they still lack accuracy and efficiency. Besides, they are designed for specific tasks and data type [7][8][9][10].

Data collection is done with the Microsoft Kinect motion sensor. 16 people were asked to practice 13 different physiotherapy exercises in front of the Kinect with a specific order of exercises and random orders. Their actions were recorded and saved in depth images, RGB images, and skeleton sequences format.

In the paper [1], Johansson experiments suggested that humans could recognize activity with only seeing the light spots attached to the person's major joints. Based on this experiment, other researchers have explored the computer vision field and suggested

that by extracting joints and body parts, we can recognize activities. Also, with the help of color-depth cameras such as Microsoft Kinect, it is easier and cheaper to get depth data and 3D skeletons of the human body [11]. Therefore, in this research, we use skeleton sequences data, which consist of 3D coordination of 25 main body joints for classification.

Contributions

This thesis:

- Presents an application that records data from the Kinect sensor, processes it, and shows a report of movements classifications and their repetitions in real time.
- Demonstrates a method for normalizing and preprocessing the multi-dimensional feature vector.
- Develops a Long Short-Term Memory model for online classification using 3D skeleton sequence data with high accuracy.
- Introduces a 3D Dynamic Time Warping algorithm for pattern matching and count the repetitions of the movements.
- Evaluates the real-time accuracy of the system and compares it to the offline classification.

The organization of the rest of the thesis is as follows: In chapter II, we will provide background information about Neural Networks, Recurrent Neural Networks, Long Short-Term Memory, and Dynamic Time Warping algorithm. In chapter III, we briefly summarize the related work. In chapter IV, we will elaborate on details of our methodology. In chapter V, we will present the results and analyze them. Finally, chapter

VI will conclude this thesis and discuss possible future work based on the results.

II. REVIEW OF THE LITERATURE

Action recognition and classification have attracted a lot of research interests due to their rapid growth in number of applications in the past decade. Although studies have done satisfying job in covering action recognition with broad data types such as accelerometer data, images, videos, and 3d skeleton configurations, they mostly focused on pre-segmented sequences which means their methods work well when the start and the end of the movement is pre-defined [12][13][14][15][16]. However, in many real-life applications we need to recognize the action before completion of the action. There are few approaches that can classify actions within unsegmented data [3][4][5][6]. In this section, we review recent related work, particularly, real-time skeleton-based action detection and recognition.

Using 3D skeleton data in action recognition has become popular when Shotton et al. [18] introduced a real-time approach to extract 3D positions of the body joints from depth images. Later, how the position, motion, and orientation of joints could be an indicator of a movement was proven in many studies [17].

In the past few years, many traditional machine learning approaches such as support vector machines (SVM), artificial neural networks (ANN), decision trees (DT) and K-nearest neighbor classifiers (KNN) have been widely used for action recognition with 3D skeleton data [19]. Paraskevopoulos et.al (2019) experimented with SVM, KNN, naïve Bayes, KNN and random forest for real-time arm movement recognition using 3D skeleton joint data extracted from Microsoft Kinect sensor. To use mentioned algorithms, they calculated a set of statistical features on the dataset. Although the mentioned approaches proved to be successful regard to the action recognition problem, they are not

the best approaches as deep learning approaches such as recurrent neural network (RNN) and Long Short-Term Memory (LSTM) have shown their strength in classification of a sequence data such as movements data [6].

The paper [7], Carrara et.al (2019) introduced a LSTM model for real-time movement classification using a sequence of 3D skeleton configurations, which consist of 3D coordinates of 31 tracked joints. The proposed unidirectional LSTM model's architecture is based on the recurrent neural network that is already proven to work efficiently for short dependencies sequences. It can effectively encode the skeleton frames within the hidden network states. The LSTM model can detect the beginning of the movement immediately, without termination of the action. In this paper, they focused on multi-label detection of user-specified actions in unsegmented sequences as well as continuous streams. They utilize a bidirectional LSTM model to estimate class probabilities by considering the future frames as well as the past ones.

In article [11], Li et.al (2016) proposed a classification-regression recurrent neural network model to detect the actions on the fly from the untrimmed stream data. This model can find the start and endpoints of actions automatically and accurately. Specifically, by using LSTM as a recurrent layer, they could achieve high computational efficiency. It can automate the feature-learning and handle the long sequence dependencies. Furthermore, based on the regression curve, the model can predict the action prior to its occurrence.

Liu et.al (2018) applied LSTM on 3D skeleton sequences for human action recognition due to the LSTM strength to solve dependencies and dynamics in sequential data. To improve LSTM's attention performance, they used Global Context-Aware

Attention LSTM (GCA-LSTM) that can choose the joints that are more informative and focus on them [6].

In paper [20], Amin Ullah et.al used deep bidirectional LSTM with CNN features for action recognition in video sequences. First, they extract features from every sixth frames to reduce the redundancy and complexity. In the next step, they used bidirectional LSTM to learn the dependency and sequential information among frames. The proposed method is capable of learning sequences and frame to frame change in features due to small change in visual data of videos.

III. BACKGROUND

Sequence or time series forecasting refers to the process of predicting the future by using current and previous data. It has been one of the hardest problems in data science and this domain includes a wide range of problems. Typical examples of sequence prediction applications are tracking daily, hourly, or weekly weather temperature, tracking changes in stock market prices, and medical devices to visualize vitals in real time. The major challenges in sequence forecasting are understanding the patterns in the sequence and using this pattern to analyze the future sequences. Figure 1 shows a body movements pattern while jumping from one point to another point. Obviously, if we only look at the body movement at one instance (initiation, start, middle or landing), it is not possible to recognize the activity, since a movement is a sequence of different movements at different time. So, we need to detect the entire movement pattern in a sequence to be able to predict it in the future.

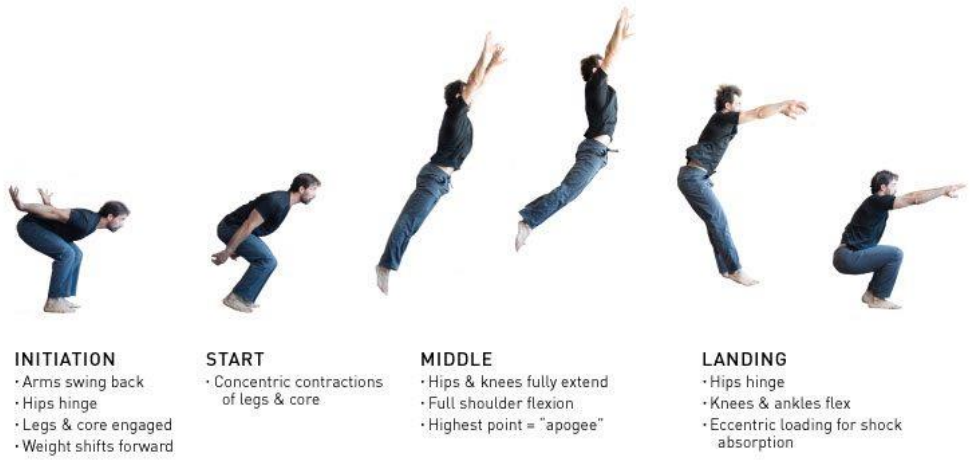


Figure 1: body movements patterns in jumping action [21]

Long Short-Term Memory networks (LSTMs) have been proven to efficiently solve sequence problems. LSTMs are a special kind of Recurrent Neural Networks (RNNs) and RNNs are a class of Neural Networks (NNs) that are powerful for modeling sequence data [22]. To understand LSTMs, we must understand NNs and RNNs first. Therefore, in the following paragraphs we explain how NNs and RNNs work and then we explain LSTMs.

Neural Network

Traditional neural networks, also known as feed-forward neural networks, are a subset of machine learning algorithms and at the heart of deep learning algorithms that endeavor to recognize relationships in a set of data. Their name and structure are inspired by the human brain, simulating the way biological neurons signal to each other to process data. NNs are used in variety of applications such as pattern recognition and data classification.

A neural network is made up of layers of neurons. The neurons are the core processing units of the network. First, there is an input layer which receives the input data. Second, there are hidden layers which performs most of the computation required by the network. Finally, there is an output layer which predicts the final outputs [23].

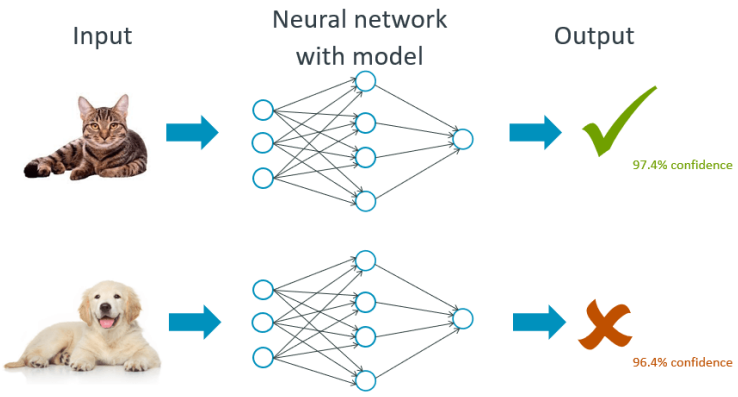


Figure 2: image classification using neural networks [24]

In conventional feed-forward neural networks, all test cases are independent. For example, the trained feedforward NN can recognize a cat or a dog in an image. In this training process, the classification process of the second image is independent from the classification process of the first image. Therefore, the output of cat does not relate to the output of dog. When the previous understanding of data is necessary neural networks do not work well since these networks do not have memory to relate the previous events to predict the next one. Therefore, NNs cannot process sequential data like the data from language translation, music recognition and movement classification.

Recurrent Neural Network

A recurrent neural network is designed to make up the shortcoming of the traditional neural networks. It has shown great success in problems such as speech recognition, translation, and more time-series problems.

A RNN looks like Figure 3. In RNNs, output of each process passes to the next processes. So, output at time $h_{(t)}$ depends on the new input, the outputs of all the previous computations, and all information learned from previous processes which are stored in

the “memory”.

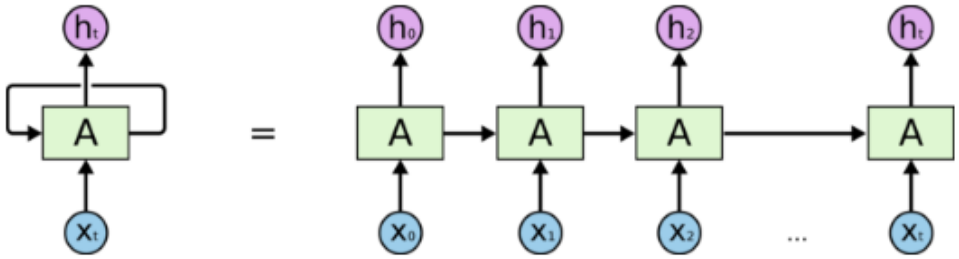


Figure 3: a RNN architecture [25]

However, RNNs can solve sequence problems to a great extent but not entirely. Hochreiter (1991) [German] and Bengio, et al (1994) [26] proved that when the sequence is long enough, RNNs have a difficult time carrying the information from earlier time steps to later ones and may leave out important information from the beginning steps. For instance, consider the case when an RNN needs the input X_0 to predict the output of h_t. If the distance of [0, t] is too long, the RNN may lose some information from early steps. The long-term dependency problem in RNNs is solved in the Long Short-Term Memory network.

Long Short-Term Memory

In 1997, Hochreiter & Schmidhuber proposed the idea of the LSTM, which can learn long term dependencies in a sequence. Special structure inside LSTM cells, makes it able to selectively keep or forget information. LSTMs can learn which information is important and which is not [27]. Figure 4 shows the architecture of an LSTM.

A memory cell is the core component of the LSTM. It consists of a cell state vector and gating units. The cell state vector reflects the LSTM memory and modifies information by forgetting old memory (through the forget gate) and inserting new

memory (through the input gate). A forget gate decides how much the unit should recall from the past. An input gate decides how much new information should be added to the cell state. Finally, an output gate decides which part of the current cell is the output [25].

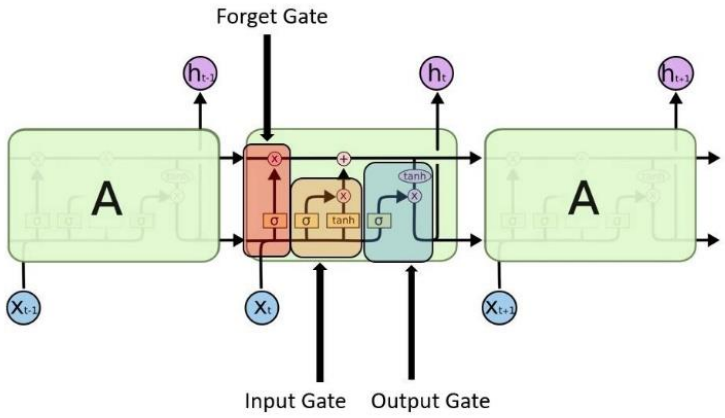


Figure 4: a LSTM architecture with 3 modules: input, output, and forget gate [25]

Dynamic Time Warping

Dynamic Time Warping (DTW) is a well-known algorithm for measuring the similarity between two time series, which may vary in time or have different lengths. The DTW measures the similarity by providing non-linear alignments which minimizes the effects of shifting and distortion in time between two sequences and computing a distance function between them [28]. It is a robust technique that has been widely used in handwriting and online signature matching [30] [31], movements recognition [32] [33], database mining [29], computer vision and computer animation [34], music and signal processing [34] [35] [36].

In an ideal situation where time series sequences are perfectly sync up and they move at the same speed and time, finding Euclidean distance is useful. The Euclidean

metric calculates the shortest path between two points on both sequences that occur at the same time. However, in most cases, sequences are out of sync. For example, in running from one point to another point, people run at different speed, but they all have similar pattern of running. In these cases, the DTW method can be helpful to find the similarity of the two time-series [16].

Take $X = (x_1, x_2, ...x_n)$, $n \in N$ and $Y = (y_1, y_2, ...y_m)$, $m \in N$ as examples two time series. The DTW algorithm starts by building the distance matrix $C \in R^{N \times M}$ representing all pairwise distances between X and Y. Since the Dynamic Programming algorithm lies in the core of the DTW it is common to call this distance function the "cost function" [37]. Although the DTW provides non-linear and flexible alignments, it must satisfy the following criteria:

- The starting and ending point of one sequence should match to the starting point and the ending points in the other sequence.
- To preserve the time-ordering of the points, we should consider $n_1 \leq n_2 \leq ... \leq n_k$ and $m_1 \leq m_2 \leq ... \leq m_K$ conditions.

The DTW works as follows:

Algorithm 1 AccumulatedCostMatrix(X, Y, C)

```
1: n ← |X|
2: m ← |Y|
3: dtw[] ← new [n × m]
4: dtw(0, 0) ← 0
5: for i = 1; i ≤ n; j++ do
6:     dtw(i, 1) ← dtw(i − 1, 1) + c(i, 1)
7: end for
8: for j = 1; j ≤ m; j++ do
9:     dtw(1, j) ← dtw(1, j − 1) + c(1, j)
10: end for
11: for i = 1; i ≤ n; j++ do
12:     for j = 1; j ≤ m; j++ do
13:         dtw(i, j) ← c(i, j)+min {dtw(i − 1, j); dtw(i, j − 1); dtw(i − 1, j − 1)}
14:     end for
15: end for
16: return dtw
```

Figure 5: the DTW algorithm [47]

The matrix value at index C [n, m] represents the distance between sequence X and sequence Y.

To improve the DTW performance for pattern matching, the constrained DTW has been introduced to limit the length of jumps (shifts in time) while aligning sequences and provides a smoother match [38]. We will discuss this condition in detail in the chapter IV and how we used it in our problem.

IV. DESIGN AND METHODOLOGY

Overview

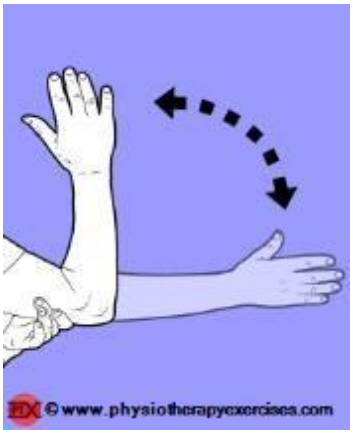
In this research, we introduce a real-time LSTM model to recognize physical therapy movements in a stream using Kinect RGB-D video data. Finally, we introduce 3D Dynamic Time Warping algorithm to count the repetition of movements in a stream.

For this study, 12 physical therapy exercises with standard set of instructions were selected. Some exercises use same joints in a different way and some exercises use completely different joints. Numbers 1 through 12 are assigned to each exercise.

Adductor stretch.

Hamstring strengthening in standing.

Hip abductor strengthening in standing.

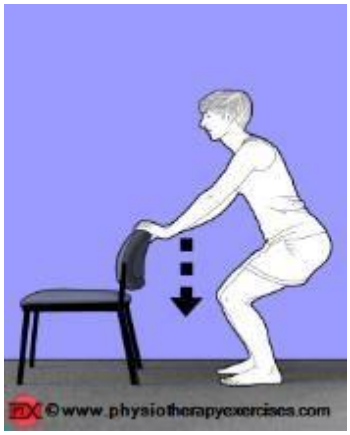


Elbow flexion.

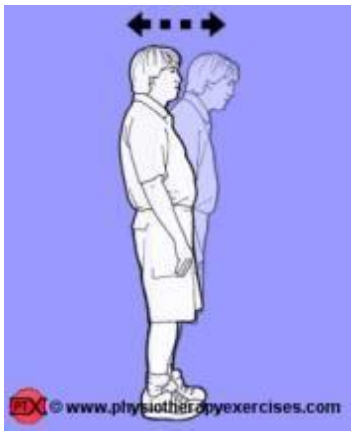
Elbow flexion and extension skimming body.

Hip flexor strengthening in standing.

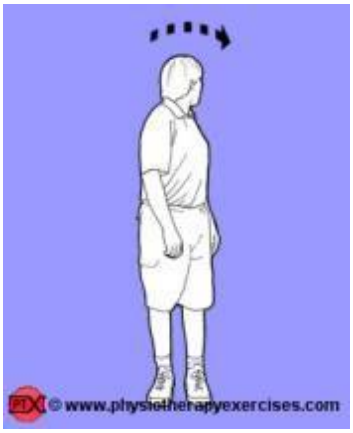
Side-stepping.



Squatting.



Stand and shift weight forwards and backwards.



Stand and look behind.

Stand on one leg and move the other leg.

Standing up and sitting down

Figure 6: exercises instructions [39]

System Design

The diagram below displays the application cycle from the time when it receives a frame to the moment when it shows the results. Once the application starts, it keeps receiving a new frame from the Kinect sensor with 30 frame per seconds speed (Stage 1). Since movement recognition is a time series problem, we need to have a sequence of frames to be able to predict the movement. Therefore, we wait for the application to

receive 120 frames and save their skeleton joints 3D coordinates (Stage 2) before proceeding to the next steps. At Stage 3, we save the data in two forms for the movement recognition purpose with the LSTM and counting the repetitions of movements purpose with the DTW. The data that is used for the LSTM should have a sequence behavior. To do so, when we receive a new frame, we remove the first frame in the list and add the newest frame to the end of the list and this process happens for every new frame. However, one new frame cannot fully represent a change in a movement, so we make a prediction with the LSTM every 50 frames. Therefore, for every two runs of the LSTM, there is an overlap of 70 frames between each of the two separate batches of data, with 50 unique frames per batch, for a total of 120 frames.

The data that is used for the DTW should not have the overlap that we create in data for the LSTM. So, we used another list that save the frames information in a list as they arrive. After the normalization and predicting a label with the LSTM for them, we compare the skeleton sequences with the standard movements. This process happens after receiving 200 new frames. After counting the repetitions of the movements, we clear the list and wait for a new batch of 200 frames.

Finally, the results which includes the label and number of repetitions are displayed on the screen. Also, raw skeleton sequence and the frame labels, the duration of each exercise, and the repetitions of each exercise are saved in text files.

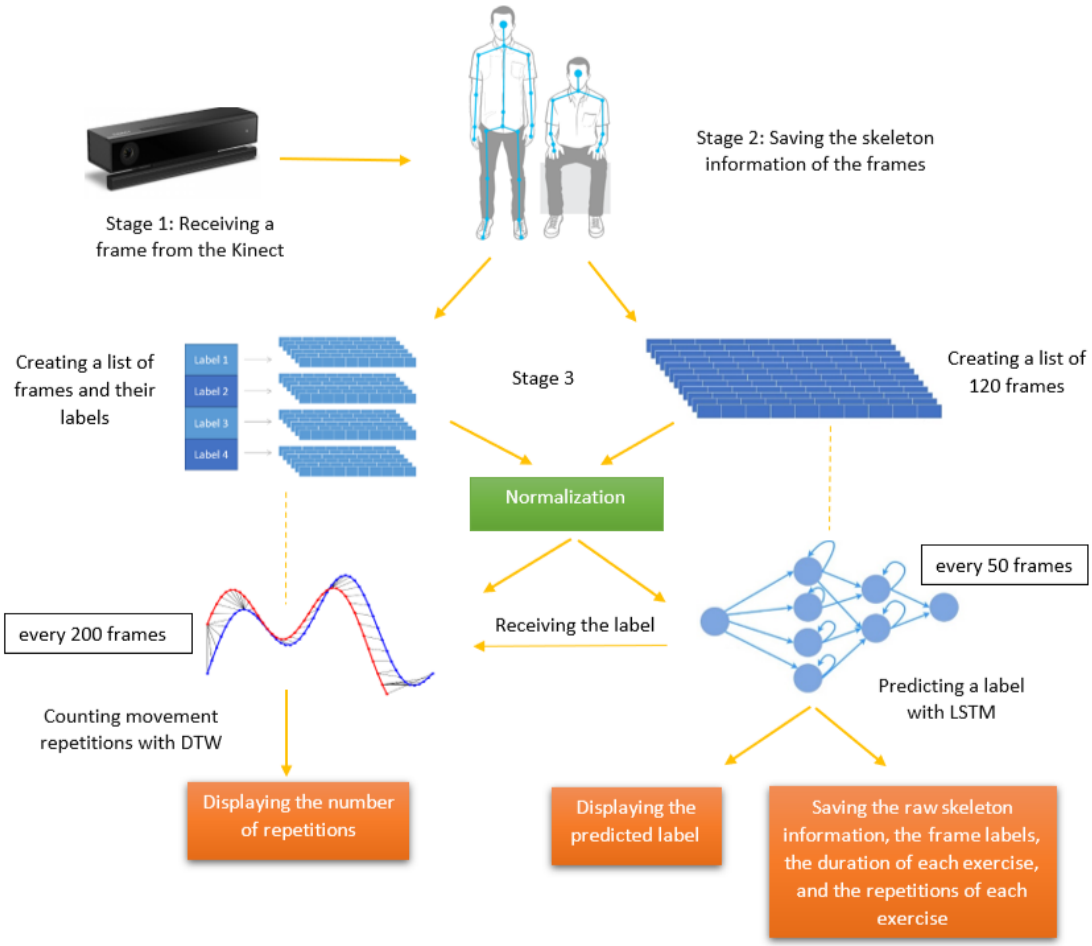


Figure 7: system diagram

The picture below is a screenshot of the application when a user is performing Elbow Flexion.

Figure 8: application interface while a user practicing Elbow Flexion movement

Data Collection

The data collection involves 12 people performing sequences of physical therapy exercises in front of Microsoft Kinect sensor to the best of their abilities. Each subject performed two sequence of exercises. The first sequence which was the same for all subjects included five repetitions of each exercises with a specific order. The second sequence was a set of exercises with randomized order and random number of repetitions between two and five.

Table 1: exercises orders and number of repetitions in predefined set

Order	Exercise name	Repetition
1	Adductor stretch	5
2	Hamstring strengthening in standing	5
3	Hip abductor strengthening in standing	5
4	Elbow flexion	5
5	Elbow flexion and extension skimming body	5
6	Hip flexor strengthening in standing,	5
7	Side-stepping	5
8	Squatting	5
9	Stand and shift weight forwards and backwards	5
10	Stand and look behind	5
11	Stand on one leg and move the other leg	5
12	Standing up and sitting down	5

The data are stored in RGB and depth images, and skeleton data which includes 3D coordination (x, y, z) of twenty-five key body joints which is the number of joints the Microsoft Kinect can track. Since RGB, depth and skeleton streams often do not have the same frame rate, time stamps are also recorded for the purpose of aligning frames from one stream to the corresponding frames in parallel streams.

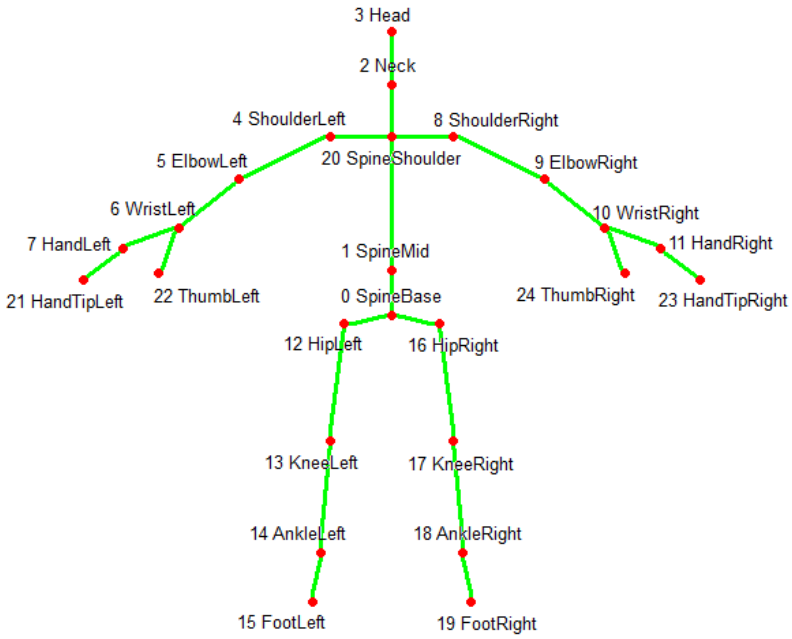


Figure 9: twenty-five joints detected in Kinect sensor and their names [40]

For finding the labels, start and end of the exercises and their time stamp are manually recorded. Then, a script segmented the skeleton information based on time stamps and assigned them a label.

```
start 1
Time 09-27-2018 13.20 38.418   Hash 23749772rgb
Time 09-27-2018 13.20 39.809   Hash 61312212rgb
stop 1
```

Figure 10: start, name, and the end of the exercise along with their timestamps

Due to the missing files in the database for some people, we were not able to retrieve the labels for their skeleton sequence. So, for this research we used skeleton sequence data of 10 people which are available in the database.

Data Preprocessing and Normalization

Initially, data for each person were stored in separate .mat files. To increase the speed of loading and processing the data, we converted all the data in one .csv.

To normalize the data, we followed Wei, T & Qiao, Y & Lee, B. (2014) method. The goal of this method is to have a skeleton in a standard orientation. To do so, we rotate the spin and the shoulders in a way that the base of the spin be at the origin and the shoulders be at an equal depth. This normalization method compensates angles of body in relation to the Kinect sensor. In addition, the whole skeleton is scaled based on the distance between base of the spin and the top of the spin. So, we can achieve scaling that match the standard spin length.

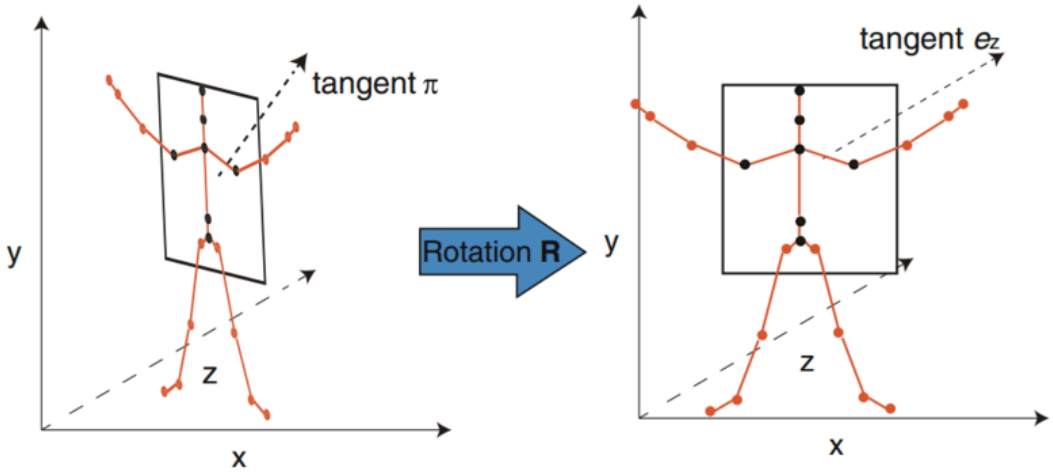


Figure 11: changing human skeleton orientation [41]

In the Wei, T & Qiao, Y & Lee, B. (2014), normalization method, they used average of 120 frames as representation of the skeleton to rotate and scale the body. In real time LSTM, we also normalize the data in groups of 120 frames.

Since the LSTM required 3D format input, after normalization, we did another data preprocessing to convert sequence data to 3D format before fitting the data to the model. The three dimensions of the input are:

- Samples: the total number of sequences in the dataset.
- Time steps: the size of the sequences.
- Features: the number of features describing each of your timesteps.

We converted the data to 51040 * 50 * 75 dimensions where 51040 is the number of samples, 50 is the number of time steps, and 75 is the number of features. In our research, we define a sequence from beginning of a movement to the end of a movement. Fifty is selected as the time-steps since the average length of 13 exercises is 50 frames with the fastest exercise with 19 frames and the slowest with 90 frames. So, the chance that entire exercise captured in 50 frames is high. Also, because Kinect sensor detects 25

joints and each joint has X, Y, and Z coordination, the number of features is 25 * 3.

Offline LSTM

As the LSTM showed promising results in solving long dependency sequences, we built a LSTM model for physical therapy exercises classification. Our goal was to train a LSTM model in an offline mode first, and then use the trained network for the real time classification. The model is developed with the Keras and the LSTM architecture is defined as follows:

```
model = Sequential()
model.add(LSTM(50, input_shape=(num_timesteps,num_features), return_sequences= False))
model.add(Dropout(0.2))
model.add(Dense(12, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 12: the LSTM architecture

In this LSTM architecture, number of hidden units, timesteps and features are respectively 50,50 and 75. The number of hidden units defines the number of variables that are calibrated for a classification to be made. The optimal number of hidden units could be smaller than the number of features (75 features) and more than number of classes (12 classes). By experimenting various range of hidden units between 12 and 75, we found out that 50 works the best. The more hidden units made the iteration slower for many hours without increasing the accuracy. Also, we used simple LSTM over bidirectional LSTM, as bidirectional LSTM made the training process slower and did not increase the accuracy. Number of outputs is 12 since we have 12 physical therapy exercises.

```
history = model.fit(train_X, train_Y, epochs=5, batch_size=10, verbose=1, validation_data=(test_X, test_Y))
train_losses.append(history.history["accuracy"])
train_accuracies.append(history.history["loss"])

#evaluate model
loss, accuracy = model.evaluate(test_X, test_Y, batch_size=10, verbose=1)
test_losses.append(loss)
test_accuracies.append(accuracy)
```

Figure 13: training parameters

Furthermore, we used a layer of dropout to avoid over fitting and improve generalization error in the model. Also, Softmax layer is used to calculate the probability of each class per training point. Adam optimizer which is a gradient decent method is used to improve learning the dependencies in each iteration and increase the accuracy. Batch size is a hyperparameter that define number of samples to work through the network before updating internal model parameters. The epoch number is the number of times that entire dataset is passed forward and backward through the neural network exactly one time. If the entire dataset cannot be passed into the model at once, it must be divided into batches. A small batch size requires less memory, and it makes the training procedure faster. The number of epochs and batches are chosen by several experiments.

Real-time Classification System

For real-time classification, we developed an application that connects to the Kinect sensor to capture and record the body movements. It uses the LSTM model to predict the movement and uses the 3D DTW algorithm to count the repetition of movements. Finally, it shows us the results on application interface. Also, it stores the data with the predicted label and their repetitions in text files. This desktop application is

developed with Windows Presentation Foundation (WPF) app with C#.

Application Connection with Kinect

The application needed to have a user-friendly interface to have a pleasant display of a person who stands in front of the camera and the results. More importantly, it needed to connect to the Kinect sensor to capture RGB images and track the body joints. These connections were made by using “Color Frame” and “Body Frame” reader libraries, provided by Microsoft [42]. As soon as a person stands in a certain distance from the camera, the sensor detects the body, draws body and joints on the skeleton for better display and gets the 3D coordination of joints. The Kinect has an eight-meter depth range, but its skeleton tracking range is 0.5 to 4.5 meters, and it fails to locate a skeleton closer than 1.5 meters due to the camera's field of view. Therefore, the cameraZ values will normally be between 1.5 and 4.5 meters [43].

Application Connection with LSTM

Since training a LSTM takes a long time, it is not possible to train a model in real time format. So, we need to train the model first, and then save and load the trained model in the application. To do so, we used Keras libraries in Python to save the model architecture in json file and save model’s weights in .h5 file. We also used Keras.Net library in C# to rebuild the model and use it for prediction [44]. Once the application is opened, it starts loading these files. After, few seconds the loading is complete, and we can use the model prediction function in the application.

LSTM Prediction

As we discussed in chapter II and chapter III, the LSTM model needs a sequence of data points for classification. In our case, data points are frames. So, it is not possible to predict a label based on just one frame or two frames. We need to wait to have long enough sequence to call the prediction function. To do so, we wait until the application receives 120 frames (about 4 seconds) before normalizing the data and predicting the label for the first time. We keep all the frames in a list. To keep the sequence behavior of the data by arriving the next frame, we remove the frame at the beginning of the list and add the new frame to end of the list and this procedure continues for every new frame. After receiving approximately 50 new frames (about one 1.6 seconds), we normalize the data in the list and then predict the class of exercise based on the updated list.

We chose 120 frames as the size of the list and 1.6 seconds (50 frames) delayed for prediction based on the average length of the exercises (60 frames) and sensor frame rates (30 fps). It is likely that a movement sequence finishes in 50 new frames. So, there is a higher chance for the LSTM to predict a right label. Also, we found out the application performance is in the highest level with these numbers. Any number lower or higher these numbers made the program too slow to show the result or less accurate in the prediction as the displayed labels and the performed exercises were asynchronous.

3D Dynamic Time Warping

Dynamic Time Warping algorithm was initially designed for comparing the similarity between two 2D sequences. Since our data is 3D, we used Wöllmer, Al-Hames, Eyben, Schuller, and Rigoll (2009) [45] study to introduce our 3D DTW algorithm that

works for the skeleton data. In the movement classification, we also had the challenge of not knowing the beginning and end of the sequence. So, we made the algorithm compatible for unknown start and unknown end.

In this algorithm, we segmented the 3D sequence with 25 * 120 * 3 dimensions where 25 is the number of joints, 120 is the number of frames, and 3 is the 3D coordination of the joints into 25 segments of 120 *3 dimensions. In this way, we reduce one dimension, so we can use the basic 2D DTW for computing the distance. By segmenting the joints sequences, every joint in the first sequence is compared with corresponding joint in the second sequence and this procedure repeated for all 25 joints. In this method, the final distance between two sequences is the average of 25 distances.

For handling the beginning and the end of the movement in the entire sequence, we consider the beginning of the sequence where the LSTM predicts a new label. However, we still need to figure out the end of the movement.

In the DTW algorithm when the end of the sequence is known, we consider the last element of the cost matrix as the minimum distance between two sequences. We use this idea to find the end of the sequence. In the unbound DTW, we calculate the distance between two sequences with the regular DTW, but instead of using the last element as the distance, we search through the last row of cost matrix for the minimum number. The end of the sequence is where the minimum number is, and that is where one repetition of the movement ends. The start of the second repetition is minimum number index in the matrix plus one. The algorithm repeats this procedure to counts the repetition of the movement in a long sequence until the min index reaches to the end of long sequence minus the half length of the short sequence.

In this algorithm, we use a set of movements sequences as the standard sequences, and we compare every input sequence with that standard sequence for pattern matching.

V. RESULTS

In this section, we evaluate our LSTM model and DTW algorithm in two formats of offline and online, and we compare their results. For offline evaluation, we used the data set that we used for training and testing the model. The online evaluation is done by asking an individual to practice the exercises in front of the Kinect sensor for several times.

Offline LSTM Results

In the following table, evaluations parameters are shown for each class. These parameters are calculated based on number of frames for each class, not the whole movement. The overall accuracy of the LSTM model in offline mode is 0.8626%. However, since the length of exercises are different, one exercise can be finished in 30 frames and another exercise in 130 frames, our data based on number of frames per exercise is unbalanced. In this case, it is better to consider other evaluation parameters such as precision and true positive rate along with accuracy.

Table 2: number of frames per exercise in the data set

Exercise	Number of frames
1	4471
2	3583
3	3555
4	2592
5	2381
6	4138
7	6280
8	4647
9	4760

10	4021
11	2855
12	7807

Evaluation parameters such as precision, accuracy, false positive, false negative are shown in the table below.

Table 3: frame classification by the LSTM

class	True positive	False positive	True negative	False negative	precision	True positive rate	accuracy
1	714	71	9930	333	0.90955	0.68194	0.96343
2	625	137	10143	143	0.82020	0.81380	0.97465
3	546	137	10227	138	0.79941	0.79824	0.97510
4	566	7	10470	5	0.98778	0.99124	0.99893
5	335	12	10688	13	0.96541	0.96264	0.99773
6	706	117	10187	38	0.85783	0.94892	0.98597
7	1350	358	9267	73	0.79039	0.94869	0.96098
8	872	26	9934	216	0.97104	0.80147	0.97809
9	750	106	9935	257	0.87616	0.74478	0.96714
10	710	67	10110	161	0.91377	0.81515	0.97936
11	392	6	10592	58	0.98492	0.87111	0.99420
12	1965	473	8528	82	0.80598	0.95994	0.94976

The following graphs show the accuracy, validation accuracy, training loss and validation loss over the 5 epochs. Based on these graphs, training set has a good performance and validation set has a desirable performance.

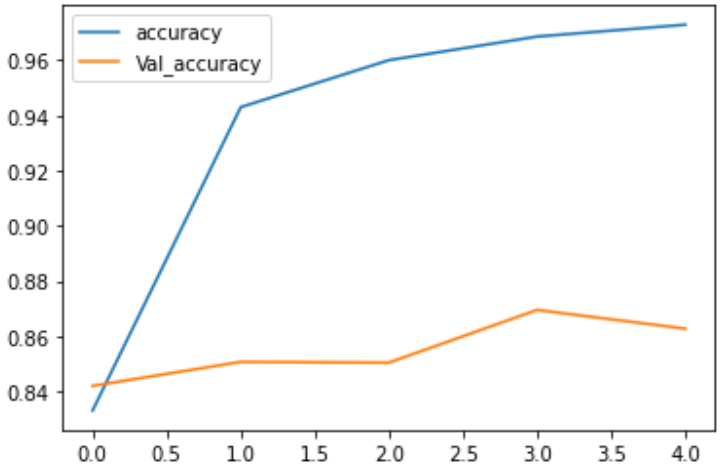


Figure 14: the LSTM accuracy in 5 epochs

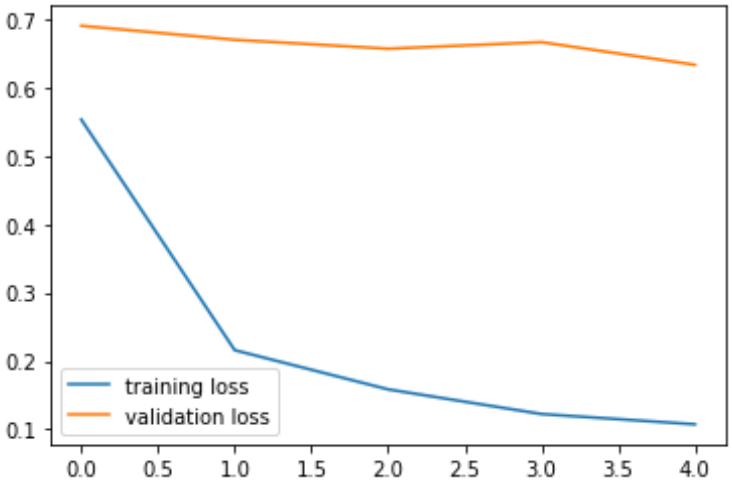


Figure 15: training loss and validation loss in 5 epochs

In the confusion matrix below, we see the model mislabeled an exercise with another exercise which uses the same body joints and has the most similar body movements. For example, exercise number 10 in which a person stands and look behind is mostly confused with exercise number 9 in which the person stands and shifts weight forwards and backwards. In the first part of both exercises, the person stands in front of the camera. So, it makes these two exercises very similar.

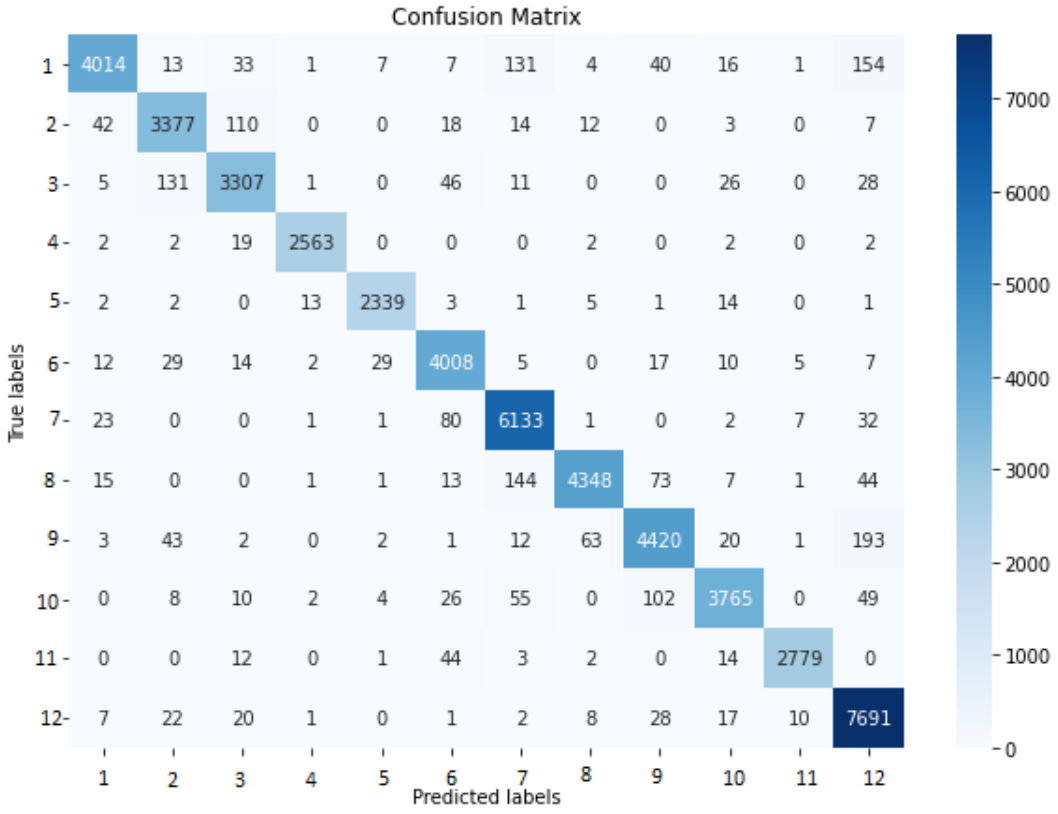


Figure 16: the LSTM model confusion matrix

Real-time LSTM Results

To evaluate the real-time classification, we asked individuals to follow a list of exercises and practice them in front of the camera. They used a timer to practice one exercise for specific amount of time and switched to the next exercise in the list when timer was up. We recorded the predicted labels and their timestamps and matched them to the exercise in the designed list that supposed to be done in that timestamps. Then, we compare the real label and the predicted label.

The order of the exercises in the list was random and everyone had a different list. Since the height of the camera and the distance from the camera effect the application performance, the height and the distance were the same for all individuals.

The overall accuracy of the real-time classification based on 6 sessions is 88.66% which is close to the accuracy that we got for the offline LSTM. The most wrong labels happened in transition time from one exercise to the next one. In some situation, wrong labels happen when two exercises are like each other as we expected based on the confusion matrix of offline model.

The table below is an example of mislabeling two similar exercises:

Table 4: example of mislabeling two similar exercises

Sequence number	**Real Label**	**Predicted Label**
1	Stand and shift weight forwards and backwards	Stand and shift weight forwards and backwards
2	Stand and shift weight forwards and backwards	Stand and shift weight forwards and backwards
3	Stand and shift weight forwards and backwards	Stand and shift weight forwards and backwards
4	Stand and shift weight forwards and backwards	Stand and shift weight forwards and backwards
5	Stand and shift weight forwards and backwards	Stand and shift weight forwards and backwards
6	Stand and shift weight forwards and backwards	Stand and Look Behind
7	Stand and shift weight forwards and backwards	Stand and shift weight forwards and backwards
8	Stand and Look Behind	Stand and Look Behind
9	Stand and Look Behind	Stand and Look Behind
10	Stand and Look Behind	Stand and Look Behind
11	Stand and Look Behind	Stand and Look Behind
12	Stand and Look Behind	Stand and shift weight forwards and backwards
13	Stand and Look Behind	Stand and Look Behind
14	Stand and Look Behind	Stand and Look Behind

The table below is an example of mislabeling in transition between two exercises:

Table 5: example of mislabeling in transition between two exercises

Sequence number	**Real Label**	**Predicted Label**
1	Elbow Flexion	Elbow Flexion
2	Elbow Flexion	Elbow Flexion
3	Elbow Flexion	Elbow Flexion
4	Elbow Flexion	Elbow Flexion
5	Elbow Flexion	Elbow Flexion
6	Adductor Stretch	Side-Stepping
7	Adductor Stretch	Stand and Look Behind
8	Adductor Stretch	Adductor Stretch
9	Adductor Stretch	Adductor Stretch
10	Adductor Stretch	Adductor Stretch
11	Adductor Stretch	Adductor Stretch

Offline DTW Results

The dynamic time warping experiments were carried out using a single user as a baseline for the ideal movement. The experimental comparisons were made using the remaining data that did not contain that user.

Table 6: the DTW results in offline mode.

Exercise Number	Real Repetition	DTW Results	Root Square Error Rate
1	40	38	0.05
2	40	43	0.075
3	40	48	0.2
4	40	55	0.37
5	40	39	0.025
6	40	32	0.2
7	40	31	0.22
8	40	33	0.175
9	40	49	0.22
10	40	46	0.15
11	40	45	0.125
12	40	35	0.125
Average Error rate = 0.18			

As seen in the table above, the DTW results are accurate, with an average of just 0.18 error rate when counting repetitions. The class number in the left column corresponds to the same exercise number. The number of repetitions performed for each exercise is stated in the "Real Repetition" column. The number of repetitions for each exercise determined using the DTW are described in the " DTW Results" column.

Real-time DTW Results

To evaluate the real-time DTW algorithm, we asked individuals to practice exercises in front of the Kinect sensor for specific number of repetitions. After the LSTM prediction, we compare the recorded sequence with corresponding standard sequence to count the repetitions.

Table 7: the DTW results in real-time mode

Exercise Number	Real Repetition	DTW Results	Root Square Error Rate
1	100	104	0.04
2	100	108	0.08
3	100	114	0.14
4	100	104	0.04
5	100	81	0.19
6	100	100	0
7	100	82	0.18
8	100	98	0.02
9	100	133	0.33
10	100	120	0.2
11	100	81	0.19
12	100	124	0.24
Average Error rate = 0.137			

As seen in the table above, the DTW is also accurate, with an average of just 0.137 error rate which is close the 0.18 error rate that we had in offline format. The class number in the left column corresponds to the same exercise number. The number of

repetitions performed for each exercise is stated in the "Real Repetition" column. The number of repetitions for each exercise determined using the DTW are described in the " DTW Results" column. We observed that when the LSTM predicts a wrong label, it causes increment in repetition for a wrong label and it mostly happens in transitions between two exercises or when two exercise use the same body joints and body movements. As we can see in the table above, the LSTM mislabels exercise 9 with exercise 11 as a result the repetitions for exercise 9 is slightly higher than 100 and for exercise 11 is less than 100.

VI. CONCLUSION

In this work, we proposed a system for real-time physical therapy exercises classification by utilizing 3D skeleton sequences extracted from every frame of Microsoft Kinect sensor and processing them through the LSTM. The model was able to learn long term and complex sequential patterns in the features.

The 3D DTW showed that it can be used for skeleton data and it works well for counting the repetitions of an exercise in a sequence. Both methods effectively work in terms of speed and accuracy.

In future, we have intention to extend this research to recognize more class of exercises and create an interface for non-expert person to add a new class of exercise. Furthermore, we aim to combine this work with techniques to intelligently compares the practiced exercise with a standard exercise and shows the similarity of the performed action with standard action in a real-time.

REFERENCES

[1] AGGARWAL, L. X. J. Human activity recognition from 3d data: A review.2014 Elsevier B.V., May 2014.

[2] DRUMOND BRUNO A. DORTA MARQUES, C. N. V. R. R.; CLUA, E. Peek - an lstm recurrent network for motion classification from sparse data.VISIGRAPP 2018, 2018.

[3] Field M, Stirling D, Pan Z, Ros M, Naghdy F (2015) Recognizing human motions through mixture modeling of inertial data. Pattern Recognit 48(8):2394–2406.

[4] Kr¨uger B, V¨ogele A, Willig T, Yao A, Klein R, Weber A (2017) Efficient unsupervised temporal segmentation of motion data. IEEE Trans Multimedia 19(4):797–812.

[5] Yu X, Liu W, Xing W (2017) Behavioral segmentation for human motion capture data based on graph cut method. J Vis Lang Comput 43:50–59.

[6] Liu J, Wang G, Duan L, Hu P, Kot AC (2018) Skeleton based human action recognition with global context-aware attention LSTM networks. IEEE Trans Image Process 27(4):1586–1599.

[7] CARRARA PETR ELIAS, J. S. P. Z. F. Lstm-based real-time action detection and prediction in human motion streams. Springer Science+Business Media, LLC, part of Springer Nature 2019, June 2019.

[8] Zhao X, Li X, Pang C, Sheng QZ, Wang S, Ye M (2014) Structured streaming skeleton—a new feature for online human gesture recognition. ACM Trans Multimedia Comput Commun Appl 11(1s): 22:1–22:18.

[9] Xu Y, Shen Z, Zhang X, Gao Y, Deng S,Wang Y, Fan Y, Chang EC (2017) Learning multi-level features for sensor-based human action recognition. Pervasive Mob Comput 40:324–338.

[10] Wu D, Shao L (2014) Leveraging hierarchical parametric networks for skeletal joints-based action segmentation and recognition. In: 2014 IEEE conference on computer vision and pattern recognition, pp 724–731.

[11] Li Y, Lan C, Xing J, Zeng W, Yuan C, Liu J (2016) Online human action detection using joint classification-regression recurrent neural networks. In: Leibe B, Matas J, Sebe N,Welling M (eds) Computer vision—ECCV 2016. Springer International Publishing, Cham, pp 203 220.

[12] Du Y, Wang W, Wang L (2015) Hierarchical recurrent neural network for skeleton based action recognition. In: 2015 IEEE conference on computer vision and pattern recognition, pp 1110–1118.

[13] Evangelidis G, Singh G, Horaud R (2014) Skeletal quads: human action recognition using joint quadruples. In: 22nd International conference on pattern recognition (ICPR 2014), pp 4513–4518.

[14] Nunez JC, Cabido R, Pantrigo JJ, Montemayor AS, Velez JF (2018) Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. Pattern Recogn76:80–94.

[15] Zhu W, Lan C, Xing J, Zeng W, Li Y, Shen L, Xie X (2016) Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In: 30th AAAI conference on artificial intelligence, AAAI 2016. AAAI Press, pp 3697–3703.

[16] Kyaagba, S. (2018, September 7). Dynamic Time Warping with Time Series. Medium. https://medium.com/@shachiakyaagba_41915/dynamic-time-warping-with-time-series-1f5c05fb8950.

[17] A. Sharaf, M. Torki, M. E. Hussein and M. El-Saban, "Real-Time Multi-scale Action Detection from 3D Skeleton Data," 2015 IEEE Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 2015, pp. 998-1005, doi: 10.1109/WACV.2015.138.

[18] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. Communications of the ACM, 56(1):116–124, 2013.

[19] Paraskevopoulos, G.; Spyrou, E.; Sgouropoulos, D.; Giannakopoulos, T.; Mylonas, P. Real-Time Arm Gesture Recognition Using 3D Skeleton Joint Data. Algorithms 2019, 12, 108. https://doi.org/10.3390/a12050108.

[20] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad and S. W. Baik, "Action Recognition in Video Sequences using Deep Bi-Directional LSTM With CNN Features," in *IEEE Access*, vol. 6, pp. 1155-1166, 2018, doi: 10.1109/ACCESS.2017.2778011.

[21] Manasurangkul, W. (2018, March 21). Movement as a Language. Medium. https://medium.com/@KaneTrains/movement-as-a-language-2741c3558af3

[22] Recurrent Neural Networks (RNN) with Keras. (2021). TensorFlow. https://www.tensorflow.org/guide/keras/rnn

[23] Education, I. C. (2021, January 6). Neural Networks. IBM. https://www.ibm.com/cloud/learn/neural-networks

[24] Özlü, A. (2020, June 13). Long Short-Term Memory (LSTM) Networks in a nutshell. Medium. https://ahmetozlu93.medium.com/long-short-term-memory-lstm-networks-in-a-nutshell-363cd470ccac

[25] Srivastava, P. (2020b, May 18). Essentials of Deep Learning: Introduction to Long Short-Term Memory. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/

[26] Hochreiter, Sepp. (1991). Untersuchungen zu dynamischen neuronalen Netzen.

[27] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8). https://doi.org/10.1162/neco.1997.9.8.1735.

[28] Adwan, S., & Arof, H. (2012). On improving Dynamic Time Warping for pattern matching. Measurement, 45(6), 1. https://doi.org/10.1016/j.measurement.2012.02.015.

[29] A. Johannes, B. Thomas, K. Hans-Peter, K. Peer, R. Matthias, Periodicpattern analysis in time series databases, In Database Systems ForAdvanced Applications, Lecture Notes in Computer Science 5463(2009) 354–368.

[30] A. Efrat, Q. Fan, and S. Venkatasubramanian, "Curve matching, time warping, and light fields: New algorithms for computing similarity between curves," J. Math. Imaging Vis., vol. 27, no. 3, pp. 203–216, April 2007.

[31] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 12, no. 8, pp. 787–808, 1990.

[32] A. Kuzmanic and V. Zanchi, "Hand shape classification using dtw and lcss as similarity measures for vision-based gesture recognition system," in EUROCON, 2007. The International Conference on "Computer as a Tool", 2007, pp. 264–269.

[33] A. Corradini, "Dynamic time warping for off-line recognition of a small gesture vocabulary," in RATFG-RTS '01: Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01). Washington, DC, USA: IEEE Computer Society, 2001.

[34] "Dtw-based motion comparison and retrieval," 2007, pp. 211–226.

[34] M. Muller, H. Mattes, and F. Kurth, "An efficient multiscale approach to audio synchronization," pp. 192–197, 2006.

[36] "Dynamic time warping," 2007, pp. 69–84.

[37] Senin, Pavel. (2009). Dynamic Time Warping Algorithm Review.

[38] Heloir, A., Courty, N., Gibet, S., & Multon, F. (2006). Temporal alignment of communicative gesture sequences. Computer Animation and Virtual Worlds, 17(3–4), 1. https://doi.org/10.1002/cav.138.

[39] G. (2021). Physiotherapy Exercises for people with injuries and disabilities. Physiotherapy Exercises. https://www.physiotherapyexercises.com/

[40] Acosta-Vargas, Patricia & Jadan-Guerrero, Janio & Guevara, Cesar & Sanchez-Gordon, Sandra & Calle-Jimenez, Tania. (2019). Technical Contributions to the Quality of Telerehabilitation Platforms: Case Study—ePHoRt Project. 10.5772/intechopen.83686.

[41] Mirinezhad, S.Younes. (2016). Title: Human activity recognition using image-processing techniques in RGBD video. 10.13140/RG.2.2.13024.17927.

[42] Leightley, D. (2016, December 17). Recording Kinect One Streams using C#. Daniel Leightley. https://leightley.com/recording-kinect-one-streams-using-c/

[43] Jamhoury, L. (2020, April 18). Understanding Kinect V2 Joints and Coordinate System. Medium. https://medium.com/@lisajamhoury/understanding-kinect-v2-joints-and-coordinate-system-4f4b90b9df16

[44] S. (2020). SciSharp/Keras.NET. GitHub. https://github.com/SciSharp/Keras.NET

[45] Wöllmer, M., Al-Hames, M., Eyben, F., Schuller, B., & Rigoll, G. (2009). A multidimensional dynamic time warping algorithm for efficient multimodal fusion of asynchronous data streams. Neurocomputing, 73(1-3), 366-380. doi:10.1016/j.neucom.2009.08.005

[46] Wei, T & Qiao, Y & Lee, B. (2014). Kinect skeleton coordinate calibration for remote physical training. MMEDIA - International Conferences on Advances in Multimedia. 66-71.