Using LSTM for Automatic Classification of Human Motion Capture Data

Rogério E. da Silva[1,2], Jan Ondřej[1,3] and Aljosa Smolic[1]
[1]*V-SENSE, School of Computer Science and Statistics, Trinity College Dublin, Ireland*
[2]*Department of Computer Science, Santa Catarina State University, Brazil*
[3]*Vologrames, Dublin, Ireland*
{*dasilvro, smolica*}*@scss.tcd.ie, jan@vologrames.com*

Keywords: human motion classification, motion capture, content analysis, deep learning, artificial intelligence

Abstract: Creative studios tend to produce an overwhelming amount of content everyday and being able to manage these data and reuse it in new productions represent a way for reducing costs and increasing productivity and profit. This work is part of a project aiming to develop reusable assets in creative productions. This paper describes our first attempt using deep learning to classify human motion from motion capture files. It relies on a long short-term memory network (LSTM) trained to recognize action on a simplified ontology of basic actions like walking, running or jumping. Our solution was able of recognizing several actions with an accuracy over 95% in the best cases.

1 INTRODUCTION

Creative industry is a broad term generally used to refer to any company devoted to create content for games, animations, AR/VR, VFX, etc. Over time, the amount of creative content that can be produced by these companies can easily become overwhelming; how big? it will depend on the size of the company itself. Studios often rely on their own production pipeline to reduce time and effort spent producing new content, therefore reducing their production costs in order to increase profit. One effective way of reducing costs is by reusing content from older productions, adapting them into new contexts, thus speeding up the production. Despite that this notion sounds reasonable, achieving it in a production setting is not easy, because, as mentioned before, being able to retrieve the one specific desired content among the millions of "*assets*" being produced every year (e.g. 3D models, textures, sound effects, soundtracks, animations, scripts, etc.) has proven to be a high demanding task.

One particular asset that has great interest for the creative industry is motion captured data (mocap) (Menache, 2011; Delbridge, 2015). Consider the following scenario: an animator needs to animate a sequence where a character walks limping from its left leg due to an injury; instead of having the motion capture team recording a new sequence in the mocap lab, the animator remembers that a few years back he already animated a similar sequence of a limping walk which means he could adapt this sequence from that old one (if he manages to remember which file contains that desired animation). So, he makes a quick search in the backup databases only to learn that the studio has, in fact, thousands of motion captured files in storage! Now, how to find the one he's looking for? Re-recording the sequence in the mocap lab. might be, in this case, a faster alternative to pursue (although not cheaper).

This hypothetical situation is frequently observed in a production pipeline, and finding ways to automatize the process for documenting creative contents being produced (so to facilitate content retrieval) would have a big impact in reducing time and effort a production team would have to spend searching through a database of older projects.

This work focuses on studying ways for automatizing the motion capture tagging process. To tag a content means to label it according to a given ontology, so that later it could more easily be found by tag-based searches, thus facilitating its retrieval and use. Our approach relies on deep learning and long short-term memory neural networks (LSTM) to analyze a series of mocap data files, classifying them accordingly. This work is being developed under the context of the SAUCE project (Smart Assets for re-Use in Creative Environments) that is a three-year EU Research and Innovation project between several compa-

nies and research institutions to create a step-change in allowing creative industry companies to re-use existing digital assets for future productions.

The goal of SAUCE project is to produce, pilot and demonstrate a set of professional tools and techniques that reduce the costs for the production of enhanced digital content for the creative industries by increasing the potential for re-purposing and re-use of content as well as providing significantly improved technologies for digital content production and management (SAUCE Project, 2018).

In this work, we studied ways for using long short-term memory networks to automatically classify content from motion captured data according to a given ontology. We have designed a LSTM architecture that accepts motion captured data and determines the actions being portrayed in each file. This work represents a first step for SAUCE project in developing a general-purpose media classifier system that could help speeding up a creative pipeline process.

In the next section we discuss the problem of human motion classification then, on Section 3 a brief literature review on motion capture, long short-term memory neural networks (LSTM) and a few related works are presented and discussed. Following, Section 4 explains our experiments and results we obtained from applying our method. Finally, we present our conclusions and final remarks in Section 5.

2 HUMAN MOTION CLASSIFICATION PROBLEM

Classifying motion means determining what kind of action (e.g. walking, running, jumping, fighting, dancing, etc.) is being portrayed by any given human motion in space. Tracking motion in space is usually achieved by *motion capture* that involves decomposing each motion as a series of three-dimensional poses (called a *skeleton*) in time (see Section 3.1).

Therefore, being able to understand motion means determining temporal relations among changes occurring to specific body parts over time while performing each given movement. In this work, we are interested in studying ways for automatic labelling a series of motion captured data to facilitate its reuse in future productions by a creative studio.

According to (Martinez et al., 2017), "*learning statistical models of human motion is a difficult task due to the high-dimensionality, non-linear dynamics and stochastic nature of human movement*". Classifying data involves analyzing each candidate extracting a series of properties from it trying to match them to a specific class in a set of known classes (an ontology). Several issues need to be tackled while working on this kind of problem:

1. finding a suitable **ontology** description that is an enumeration of possible classes and their attributes (the criteria used to identify an element of that class);
2. an adequate **knowledge representation** on how to represent ontological attributes in a way that artificial intelligence can work with;
3. a **prediction criteria** to describe how to determine if a given candidate belongs to a certain class.

In the following sections we describe how to track and represent spatial motion of humans, how long short-term memory networks work and why we choose to use them in this research and finally, a few related works aiming human motion classification problem are presented.

3 STATE OF THE ART

3.1 Motion Capture

Motion capture (or **mocap**) is the process of recording the movements of objects or people via special hardware setups . There are several possible technologies that can be applied to capture movement in space:

Optical Systems utilize data captured from image sensors to triangulate the 3D position of a subject between two or more cameras calibrated to provide overlapping projections. This can be achieved using special markers that can be passive (reflective) or active (synchronized flashing LEDs); they can also use markerless tracking systems that relies on computer vision to recognize human parts from the set of cameras;

Non-optical Systems any other technology that allows motion tracking. The most common ones are: **inertial systems** use inertial measurement units (IMUs) containing a combination of gyroscopes, magnetometers, and accelerometers, to measure rotational rates; **mechanical motion** are often referred to as exoskeleton motion capture systems, due to the way the sensors are (directly) attached to the body to perform the tracking; and **magnetic systems** calculate position and orientation by the relative magnetic flux of three orthogonal coils on both the transmitter and each receiver.

One approach that is very popular these days and is traditionally employed by animation studios involves an actor wearing a suit covered with optical

markers that can then be tracked by an optical system of infrared cameras.

From these recordings results a series of 3D coordinates for each marker (tracked several times per second) that can later be mapped to a character as an animation (this technique is called *retargeting*). Figure 1 shows a few frames of a skeletal animation obtained via a motion capture session of a walk (CMU Graphics Lab, 2018).

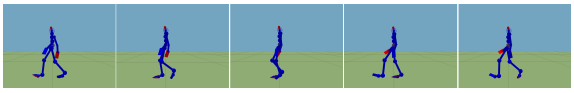

Figure 1: Motion captured sequence of a walk

Regarding file formats for storing motion capture data one popular choice is the BVH format. The *Biovision Hierarchical data* (BVH) file format was originally developed by Biovision (a motion capture service company) to distribute mocap data to their customers. Later, it became a very popular format for storing mocap.

The reason why we decided to adopt this format is because while other motion capture formats, like the C3D format (`https://www.c3d.org/`), store only the coordinates for each tracked markers (Figure 2 on the left), the BVH format also represents hierarchical relations between joints, i.e., their physical relations called a *skeleton* (Figure 3), making it simpler to correlate movements between adjoined joints (Figure 2 on the right).

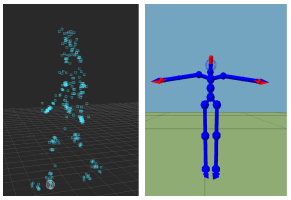

Figure 2: C3D markers vs BVH Skeleton

Motion capture data is recorded as a series of *motion channels*, each representing one spatial location and/or orientation of a joint. Since the amount of channels is dependent on the number of joints and the number of degrees of freedom (DOF) of each joint, the size of the frame can vary from file to file.

3.2 Long Short-Term Memory Neural Networks

The Long Short-Term Memory (LSTM) network is a type of Recurrent Neural Network (RNN), which is a special type of neural network designed for sequence problems like, for instance, texts, speech, and animations. Traditional RNNs contain cycles that feed the

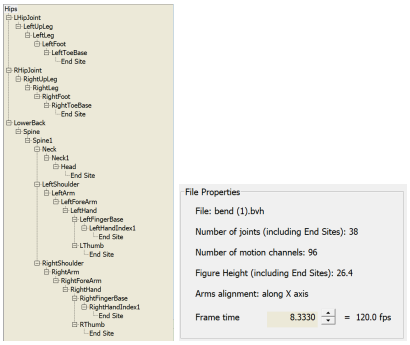

Figure 3: Example of a skeleton and motion channels definition in a BVH file

network activation from a previous time step as inputs to influence predictions at the current one (Brownlee, 2018; Hochreiter and Schmidhuber, 1997).

Despite the fact that RNNs can learn temporal relations, their main limitation is regarding training in a problem known as the "*vanishing gradient*". This problem happens when, during training of a recurrent process, the weights change become so small that they have no effect in learning the data (or so large in the other way around: "*exploding gradient*").

LSTMs solve these problems by design. All information being propagated through the network should pass first by three *gates*. These gates are activation functions especially designed to work on the data so to only allow relevant information to continue being propagated during training. The three gates are:

Forget Gate decides what information to discard from each layer;

Input Gate decides which values from the input to update the memory;

Output Gate decides what to output based on the input and the memory.

In the literature, several variants of this architecture can be found where the number of gates can vary to suit specific contexts and needs .

3.3 Related Works

Two distinct classes of works involving automatic recognition of the human skeleton, can be found in the literature: *human pose estimation* and *human action classification*. We argue that, despite the clear similarities between the two in terms of their goals (i.e. the recognition of human motion), they significantly differ in most of the technical aspects involved in how to tackle with the problem, namely, data representation and processing:

Human Pose Estimation aims at detect and estimate human poses in video format (resulting from

actual recordings of people or graphical renditions of 3D motion capture data), then extracting skeletal information that can include depth estimation (3D) or not (2D). The most common approach used in this context involves analyzing the pixels of each frame of a video using classical computer vision techniques.

Just to mention a few examples of works that adopt this approach: (Du et al., 2016; Tekin et al., 2016; Toshev and Szegedy, 2014).

Human Action Classification aims at interpreting 3D motion from skeletal data analysis of actual spatial recordings (as described in Section 3.1).

In this particular sense, not many works have been found in the literature. Some examples that have, somehow, influenced our work are:

- In (Bütepage et al., 2017) the authors trained a LSTM to predict future 3D poses from the most recent past. The system is described as an encoder-decoder network for generative 3D models of human motion based on skeletal animation;
- In (Gupta et al., 2014) the authors propose an approach to directly interpret mocap data via *v-trajectories* that are sequences of joints connected over a time frame to allow finding similar mocap sequences based on pose and viewpoint;
- Another example of human motion prediction using deep learning LSTMs is presented in (Martinez et al., 2017);
- A slightly different application but still related to human motion prediction is the multi-people tracking system presented by (Fabbri et al., 2018), where the authors developed a system capable of extracting information of people body parts and temporal relations regarding the subjects' motions. In this work, the system automatically detects each human figure and respective skeletal information from frames of video recordings, even if partially occluded, by matching those with a database of body poses.

The common point between all the related works presented here is that, despite the fact they studied automatic approaches to identify human motion from motion capture data, none of them focused on labelling the data in order to facilitate future queries and content retrieval, which seems to remain an unsolved problem.

4 EXPERIMENTATION

In this section its described the experiments made with LSTM networks implemented using Python and Keras (`https://keras.io/`) with Tensorflow (`https://www.tensorflow.org/`) using the aforementioned data set built upon the CMU Motion Capture Library.

4.1 Ontology of Human Actions

Since the scope of this work focuses only on tagging motion capture actions, instead of designing a complete ontology for every possible creative media an animation studio would be interested in cataloging, we opted to simplify the representation to consider only a selected set of human actions.

We are using the freely available CMU Motion Capture library (CMU Graphics Lab, 2018) for our experiments. Thus, the list of actions our system is able to classify reflects the actions available in this database.

In our experiments we considered the following actions: `bending down`, `climbing`, `dancing`, `fighting`, `jumping`, `running`, `sitting down`, `standing up`, and `walking`.

The definition of this ontology is important because training the neural network to recognize its classes means that it needs a carefully designed data set of mocap files for each class. Our training data set is composed of 1136 files divided into those 9 categories, representing more than 820,000 frame samples.

4.2 Data Representation

There are two main concerns in terms of representing data for a neural network: how to represent the input (or training) data and how to represent the output (in our case the labels of each class in the ontology).

For the input, we followed the data representation presented at (Bütepage et al., 2017), where each frame of a mocap recording is represented in the Cartesian space of each joint's rotational data plus the positional data for the 'Hips', thus resulting in an 1D-array of size $3 \times N_{joints} + 3$ where N_{joints} is the number of joints. Still agreeing with the authors, we normalized all joint's rotational angles, centering the skeleton at the origin. So, each joint data is represented according to the following format: `<Joint>_ZRotation`, `<Joint>_YRotation`, `<Joint>_XRotation`, e.g. `LeftUpLeg_Zrotation`, `LeftUpLeg_Yrotation`, `LeftUpLeg_Xrotation`.

Except for the Hips that also include the XYZ positional values.

Several tests have been performed with the size of the sample, i.e. the number of frames (N_{frames}), and a comparative of the results is presented in Section 4.5 below.

Regarding the output, each possible outcome is represented as a classical 'one-hot' binary string, where the number of bits relates to the size of the ontology (number of classes) and each label having a different bit highlighted. For instance, since we have 9 different labels in the ontology, the first label "`bending down`" is represented by the sequence `100000000`.

These representations also influence the size of the first and last layers of the network as described in Section 4.4.

4.3 Class Prediction

Since we opted to split the training samples into subsets of N_{frames} frames, we had to do the same with our working data set for consistency. Thus, each motion capture file being analyzed is also split into samples of the same size, and each sample is then submit to the network for prediction. As a sample can be matched with different classes, in the end the resulting class is obtained by taking the **mode** of the class prediction array, i.e., considering the most frequent label as the answer .

This approach has the advantage of classifying each file in terms of how likely that file belongs to a given class of the ontology, thus allowing for multiple interpretations of its content, much like how it would happen in a real scenario. Consider a recording where the actor starts running in preparation for a jump. Imagine now an animator searching for either 'run cycles' or 'types of jump', the system should be able to understand that the particular file would be a suitable response in either case.

4.4 Implementing a LSTM

As mentioned before, we opted to implement the prototype for our tool in Python using Keras with Tensorflow. The main reason for this choice was due to the simplicity that these tools offer, making it more adequate for quick prototyping.

The solution that was used for the experiments discussed in this paper rely on three layers : the first one (the input layer) is a LSTM layer of 15 neurons. This value was chosen arbitrarily based on several tests and can be modified to fit different needs like for instance, different skeletal structures or computational performance, the second layer is a similar LSTM layer (stacked LSTM layer) with the sole purpose of increasing depth of the network (our experiments showed that deeper networks can perform better while predicting lengthier animations), and finally, the third layer (the output) is the one responsible for encoding the predicted outcome to one of the classes in the ontology as described in previous sections.

It is important to notice that although the size of the input layer does not necessarily need to match the size of the input data, the output layer does need to match the size of the output, i.e., the number of possible labels that can be outputted.

Listing 1: LSTM implementation in Keras

```
1  nNeurons    = 15
2  numLabels   = 9
3  sampleSize  = 5
4  dsSize      = len( trainingDataSet )
5  nJoints     = 38
6  dsShape     = ( sampleSize, nJoints * 3 )
7
8  model       = Sequential()
9  model.add( LSTM( nNeurons, return_sequences = True,
10                  input_shape = dsShape ) )
11 model.add( LSTM( nNeurons ) )
12 model.add( Dense( numLabels, activation = 'softmax' ) )
13 model.compile( loss = 'mse', optimizer = Adam( lr = 0.001 ),
14                metrics = ['accuracy'] )
15 model.summary()
```

4.5 Preliminary Results

Several experimental tests have been performed with our tool covering different network architectures and several subsets of the ontology. In our experiments, as a way to better assess the accuracy of the model. We prepared a series of motion capture files carefully editing their content to portray only a single action per file.

In this section, we describe three of such experiments:

1. We performed a series of 5 predictions with the model considering all 9 categories. Our prediction data set in this case was composed by 54 files (6 for each category);
2. Later, a subset containing only the four larger classes data sets have been considered for a second round of predictions, and the results are presented next in Section 4.5.2;
3. Finally, the previous experiments' results showed that the despite the differences in size of the training data sets, four specific classes appeared to have been better modeled by the network, so we decided to performed a third round of predictions using only these four ones. The results for these are presented in Section 4.5.3.

4.5.1 Experiment # 1

Table 1 summarizes the results obtained from these predictions for the first trial. It's important to notice that the larger the size of the sample, the lower the amount of samples, although temporal relations are better represented.

# files	# frames	Sample size	Time slice (ms)	# samples	Accuracy
1136	828263	1	8.3	828263	29 of 54 (53.7%)
		5	41.7	162650	25 of 54 (46.3%)
		10	83.3	82823	29 of 54 (53.7%)
		15	125.0	55214	25 of 54 (46.3%)
		20	166.7	41409	27 of 54 (50%)

Table 1: First experiment accuracy results considering different page size and all 9 labels of the ontology

Figure 4 shows the confusion matrix we obtained by taking, as an example, the best result in our experiment (the one where sample size = 10). In this experiment we can clearly see as the darker regions of the matrix, that some of classes like '`jumping`' or '`walking`' has been better learned by the network than other classes.

Figure 4: Confusion matrices for the predictions considering sample size = 10 (on the left) and the overall result after combining all 5 predictions (on the right)

These results were indicating that the network was underperforming while predicting some of the classes, most likely due to the problem of *underfitting* (Brownlee, 2018) since the size of the training data set for each category varies significantly (the smaller data set corresponds to 31% the size of the largest one).

4.5.2 Experiment # 2

In order to try to minimize these effects, a second round of predictions have been performed considering only the four larger data sets available: `bending down`, `dancing`, `standing up`, `walking` where each of them have more than 100,000 frames, changing the ratio between the smaller and larger data sets to 76%.

The results obtained on this second round of predictions are presented in Table 2 below. They are evidence that the low accuracy detected in the first experiment was due to underfitting the model, which means that with a larger training data set the network should perform better even when considering larger ontologies.

# files	# frames	Sample size	Time slice (ms)	# samples	Accuracy (%)
541	503050	1	8.3	503050	16 of 24 (66.6%)
		5	41.7	100610	13 of 24 (54.2%)
		10	83.3	50305	14 of 24 (58.3%)
		15	125.0	33536	14 of 24 (58.3%)
		20	166.7	25152	16 of 24 (66.6%)

Table 2: Accuracy results after second experiment that considered only the four larger training data sets

Once more, taking the best result as an example, calculating the confusion matrix for the experiment resulted in Figure 5. Here the results were significantly better relatively to the previous experiment. This can be observed considering that the values in the main diagonal of the matrix are higher than the rest of the matrix (ideally a confusion matrix would appear as an *identity matrix*).

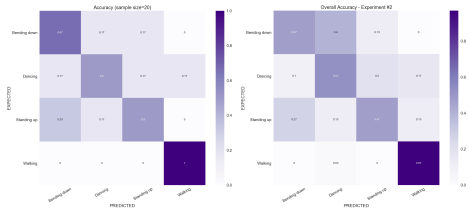

Figure 5: Confusion matrices for the predictions considering sample size = 20 and the four larger training data sets (on the left) and the overall result after combining all 5 predictions (on the right)

Although the results after this experiment represented an improvement in regard to the last experiment, they still were not as good as one could expect. After a careful analysis of the confusion matrices obtained after the combined results for each experiment, it was noticeable that a specific set of classes were performing better despite the fact those were not the larger data set at disposal. Figure 4 on the right show this combined matrix where it is possible to infer this alternative four classes of preference: "`bending down`", "`jumping`", "`running`" and "`walking`" as the four ones showing the most promising results.

Next section present the results for the third experiment considering exactly these four classes. Worth noticing that these data sets have significantly different training data set sizes and still the network were able to train satisfactorily in them. We hypothesize that this is due the nature of those specific actions that significantly vary from each, making it simpler for the network to differentiate them from each other.

4.5.3 Experiment # 3

In this experiment all training data set was composed of 903 files, separated into four categories: "*bending*

down", "*jumping*", "*running*" and "*walking*" and the prediction data set containing 24 files.

# files	# frames	Sample size	Time slice (ms)	# samples	Accuracy (%)
903	360668	1	8.3	360668	21 of 24 (87.5%)
		5	41.7	72133	21 of 24 (87.5%)
		10	83.3	36066	22 of 24 (91.7%)
		15	125.0	24044	23 of 24 (95.8%)
		20	166.7	18033	21 of 24 (87.5%)

Table 3: Accuracy results after third experiment

This was the most successful experiment of them all and the results clearly demonstrate the feasibility of the model and allow us to conclude that the other experiments would also perform better if more training would be available.

Figure 6 below depicts the results for the best experiment performed under the described conditions (size = 15) and also the resulting obtained by combining all five experiments with this ontology.

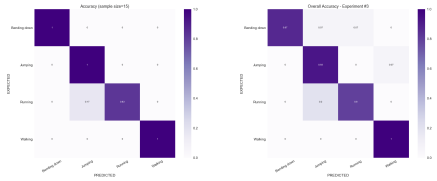

Figure 6: Confusion matrices considering sample size = 15 and four selected labels (on the left) and the overall result of the experiment on the right

4.5.4 Summary

Figure 7 compares the accuracy (vertical axis) obtained with the experiments using different sample sizes (horizontal axis). In orange it is shown the predictions considering all 9 categories, in blue the predictions using a subset containing only the four larger data sets and in gray the experiment selecting the four most successfully recognized in the first experiment.

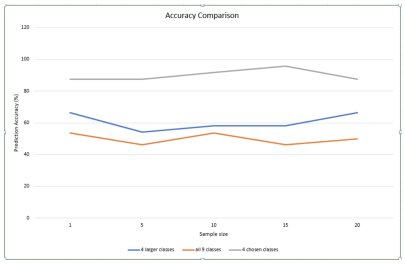

Figure 7: Comparison between the results of the three experiments

In summary, the results obtained with these experiments can be considered promising, indicating the viability of the model, showing that a stacked LSTM neural network can successfully learn how to classify human actions from motion capture data (proof-of-concept).

4.6 Limitations

- Right now our system has the limitation of only using CMU library skeleton structure (Figure 3), which means that all training files and prediction data have to have the same length (N_{joints}). A more generic approach that shall consider retargeting different skeleton structures into a baseline model, thus allowing using multiple representations together is under development;
- Also, another limitation regarding data is the fact that we are constrained to the size of the training data set available for experimentation. So, the results that have been obtained reflect that. Although, the significant improvement the second experiment showed relatively to the first, make us confident that this, at least, indicates the feasibility of the model and that, with larger databases, the system should perform much better in terms of accuracy classifying the data thus solving the underfitting problem;
- Allowing recognition of other features from the motion captured data, such as affective body postures (Kleinsmith et al., 2011), and gestures, e.g. a 'happy walk' or a 'sad handshake'. This feature would be of the most importance when trying to retrieve creative content that involves digital actors (pantomime) and crowd simulations;
- Extending the ontology in a way that would allow developing automatic recognizers for any other type of media related to the creative process in a studio.

5 CONCLUSIONS

Finding new (more efficient) ways of authoring creative content is a feature that interest the most to companies in this sector. This work aims at studying ways of improving productivity by reducing time and effort authoring new animations by reusing older medias into new projects. Since the volume of material produced by such companies can be extremely large, cataloging old 'assets' to facilitate tag-based searches for future reference is a key aspect when dealing with problems of this nature.

In this project, we are interested in developing a tool for automatic classification of motion capture content in terms of the actions the actor is performing, like walking, running, jumping, etc.

We designed a system that relies on deep learning, more specifically on long short-term memory (LSTM) neural networks, to analyze the content of motion capture files in BVH format and classify it according to labels defined by a simplified ontology composed of 9 action tags.

Our approach considered a separate data set of carefully edited mocap files for training the network on how to recognize each action. These data sets were adapted from the freely available CMU Motion Capture Library. After training, the network was tested using a different set of files that did not have been used during training. For the sake of assessment, each of these files were manually annotated with the expected label.

Comparing the results obtained by the classification software against the expected manually annotated tags, the system showed, in several of the tests, an accuracy in some cases better than 95%, what can indicate that the original hypothesis have been satisfied.

For the future, it's expected to improve training by adding other actions to the ontology like, for instance, considering affective body postures and/or other kinds of medias that might be of interest in a production pipeline like textures, sounds, etc.

The ultimate goal would be to design an extendable modular content annotator capable of annotating with different types of medias, based on a general-purpose ontology.

Another possible application that might gain from this automatic motion capture action recognition technology is authoring character animations for the purpose of retargeting crowd behaviors to different scenarios. In theory, such an AI system could help understanding each character's movements in a given situation and then help adapting the animations to new target scenarios, and facilitate authoring crowd simulation.

Acknowledgment

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under the Grant Number 15/RP/2776 and in part by the European Unions Horizon 2020 Research and Innovation Programme under Grant Agreement No 780470.

REFERENCES

Brownlee, J. (2018). *Long Short-Term Memory Networks with Python - Develop Sequence Prediction Models With Deep Learning*. Machine Learning Mastery. [eBook].

Bütepage, J., Black, M. J., Kragic, D., and Kjellström, H. (2017). Deep representation learning for human motion prediction and classification. *CoRR*, abs/1702.07486. http://arxiv.org/abs/1702.07486.

CMU Graphics Lab (2018). CMU graphics lab motion capture database. http://mocap.cs.cmu.edu/.

Delbridge, M. (2015). *Motion Capture in Performance - An Introduction*. Palgrave Macmillan UK, first edition edition.

Du, Y., Wong, Y., Liu, Y., Han, F., Gui, Y., Wang, Z., Kankanhalli, M., and Geng, W. (2016). Marker-less 3D human motion capture with monocular image sequence and height-maps. In *European Conference on Computer Vision*, pages 20–36. Springer.

Fabbri, M., Lanzi, F., Calderara, S., Palazzi, A., Vezzani, R., and Cucchiara, R. (2018). Learning to detect and track visible and occluded body joints in a virtual world. *CoRR*, abs/1803.08319. http://arxiv.org/abs/1803.08319.

Gupta, A., Martinez, J., Little, J. J., and Woodham, R. J. (2014). 3d pose from motion for cross-view action recognition via non-linear circulant temporal encoding. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2601–2608.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. D.O.I.: 10.1162/neco.1997.9.8.1735, https://doi.org/10.1162/neco.1997.9.8.1735.

Kleinsmith, A., Bianchi-Berthouze, N., and Steed, A. (2011). Automatic recognition of non-acted affective postures. *Trans. Sys. Man Cyber. Part B*, 41(4):1027–1038. D.O.I.: 10.1109/TSMCB.2010.2103557.

Martinez, J., Black, M. J., and Romero, J. (2017). On human motion prediction using recurrent neural networks. *CoRR*, abs/1705.02445. http://arxiv.org/abs/1705.02445.

Menache, A. (2011). *Understanding Motion Capture for Computer Animation*. Morgan Kaufmann, second edition edition.

SAUCE Project (2018). Smart asset re-use in creative environments - SAUCE. http://www.sauceproject.eu.

Tekin, B., Rozantsev, A., Lepetit, V., and Fua, P. (2016). Direct prediction of 3d body poses from motion compensated sequences. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 991–1000.

Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.