User-Defined Gesture Recognition For Real-Time 3D Character Control

Abdesselam Guerroudj, Sheldon Andrews
École de Technologie Supérieure

Introduction

Gesture-based control of 3D characters is an intriguing problem wherein the overall goal is to provide expressive and interactive control of an animated character using only simple gestures and tracking devices. This type of technology has numerous benefits for virtual reality and video game applications. Machine learning methods are well suited to the task of gesture recognition. In particular, recurrent neural networks (RNNs) are capable of encoding both temporal and spatial aspects of the mapping between gesture and character motion.

We propose training a long short-term memory (LSTM) to recognize bespoke gestures that are defined by the end-user. We specifically target control of 3d characters for real-time video games using HTC-Vive trackers.

Training will be split into two stages :

• **Learning the null class of motions.** These are movements that do not map to any gesture, such as idling, hand gestures during conversation, itching or scratching, and other motions we want the system to reject as valid gestures.

• **Learning active gesture class of motions.** These are movements the user does intend to use to control the 3d character, and they are customized by the user during a calibration phase before online play.

Figure 1: Example User-specified motion control of virtual character using VR controllers.

Data Preparation

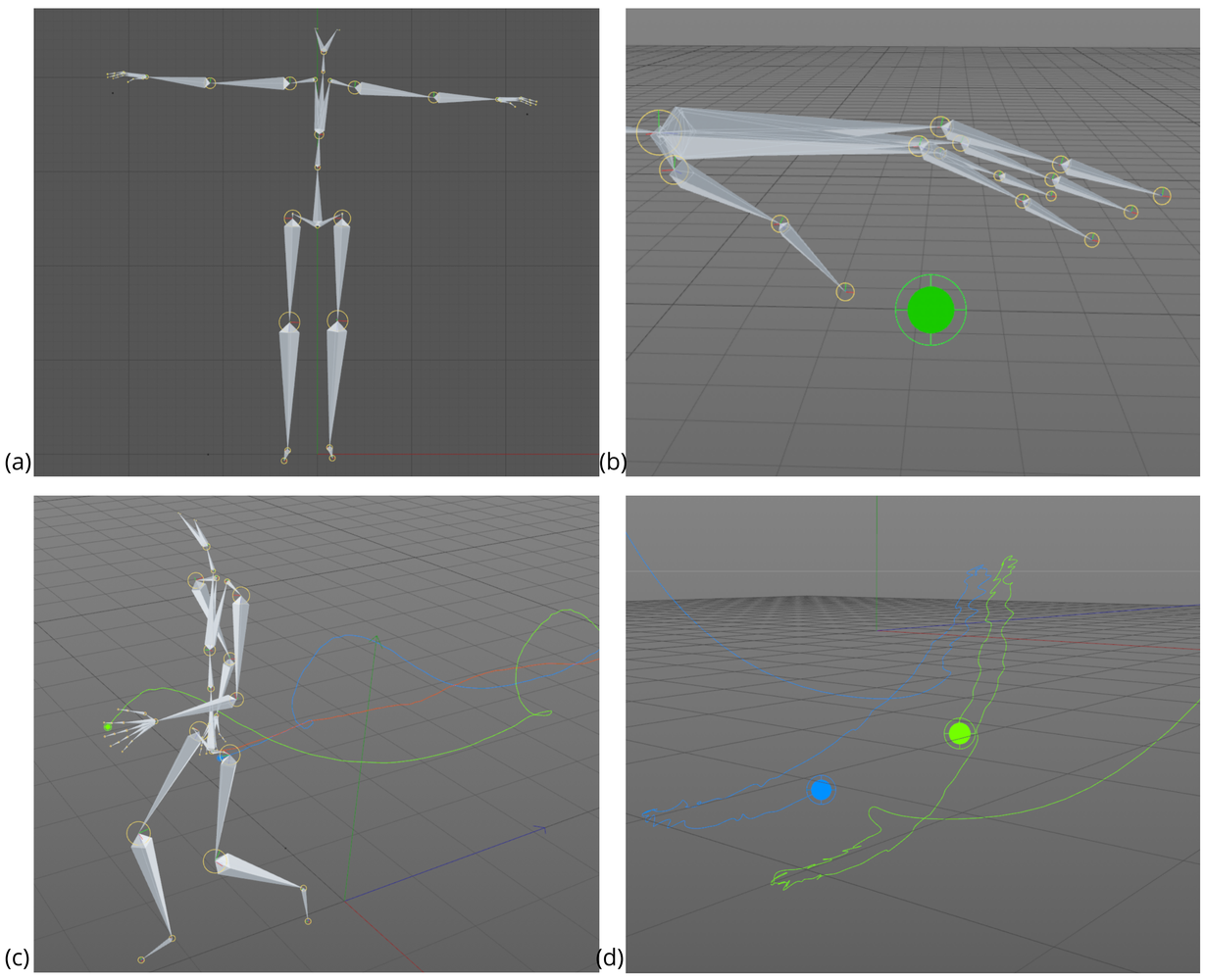

Figure 2: Data preparation steps. (a) Original skeleton from CMU database. (b) Added virtual markers on each hand. (c) Virtual markers positions tracking. (d) Localization of virtual markers positions relative to root bone and perturbation.

For starters, both classifiers will be trained by leveraging the large existing database of human motion, CMU [1] dataset, and placing virtual trackers on the hands of the animated skeletons. Then, the animations will be perturbed.

We split the CMU Mocap database into the two classes: movements, and gestures. We modified the motion skeletons by adding a single point at each hand mimicking the position of a VR controller.

The positions of the newly added points are tracked, then localized relative to the root bone. This effectively eliminates locomotion and adds more generalizability to our data. To add more robustness, we added per-frame white noise on the positions of the two points. For each motion file, we generate 20 different iterations with white noise added at each frame generating perturbed motions. Perturbations are to mimic real-time controller movement noise. This results in a much more robust model.

We used 56 motion files for each class with 20 different iterations for each resulting in a dataset is 6 columns by 6,373,900 rows. Columns contain positional coordinates (X, Y, Z) for both hands. Rows correspond to total frames of all motion files used.

[1] MoCap, C.M.U., 2003. The data used in this project was obtained from mocap. cs. cmu. edu. the database was created with funding from nsf eia-0196217. City.

Training

Due to the sequential nature of the gestures and movements involved in real-time control, Hidden Markov Models (HMMs) and Recurrent Neural Nets (RNNs) are the most suitable training algorithms to use. However, HMMs need strong assumptions to be made about the model such as the dependency of the current state on only the previous and next states. In addition, HMMs are not complex enough to discriminate between a user idling and performing a gesture. This makes RNNs much more suitable due to their discriminative nature, and their complexity. Since we have access to large datasets, RNNs was our choice. Furthermore, the importance of sequence in motion requires us to choose LSTMs to handle learning the long-term dependencies and avoid the potential vanishing gradient problem.

To train on the data, we chose to train an LSTM network using Keras library built on top of TensorFlow in Python.

One more necessary step to help with the training was changing the data from the original shape of (6373900, 6) to (84984, 360)

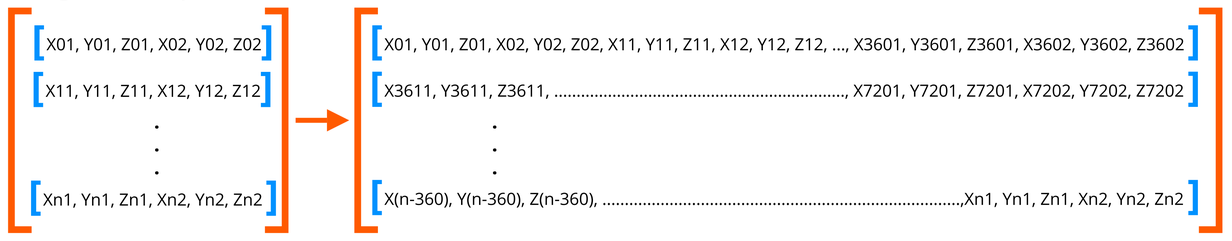

Figure 3: Reshaping training data to longer arrays to preserve the temporal information of the motion.

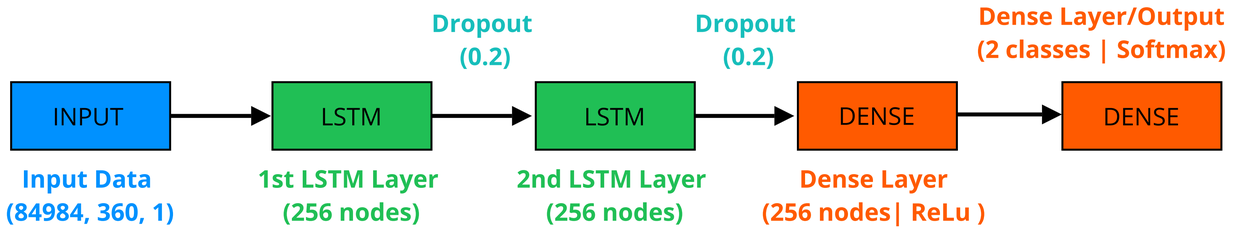

Figure 4: LSTM network model. Loss function: Categorical Cross-entropy. Optimizer: Adam. Learning rate: 1e-3 (decay 1e-6)

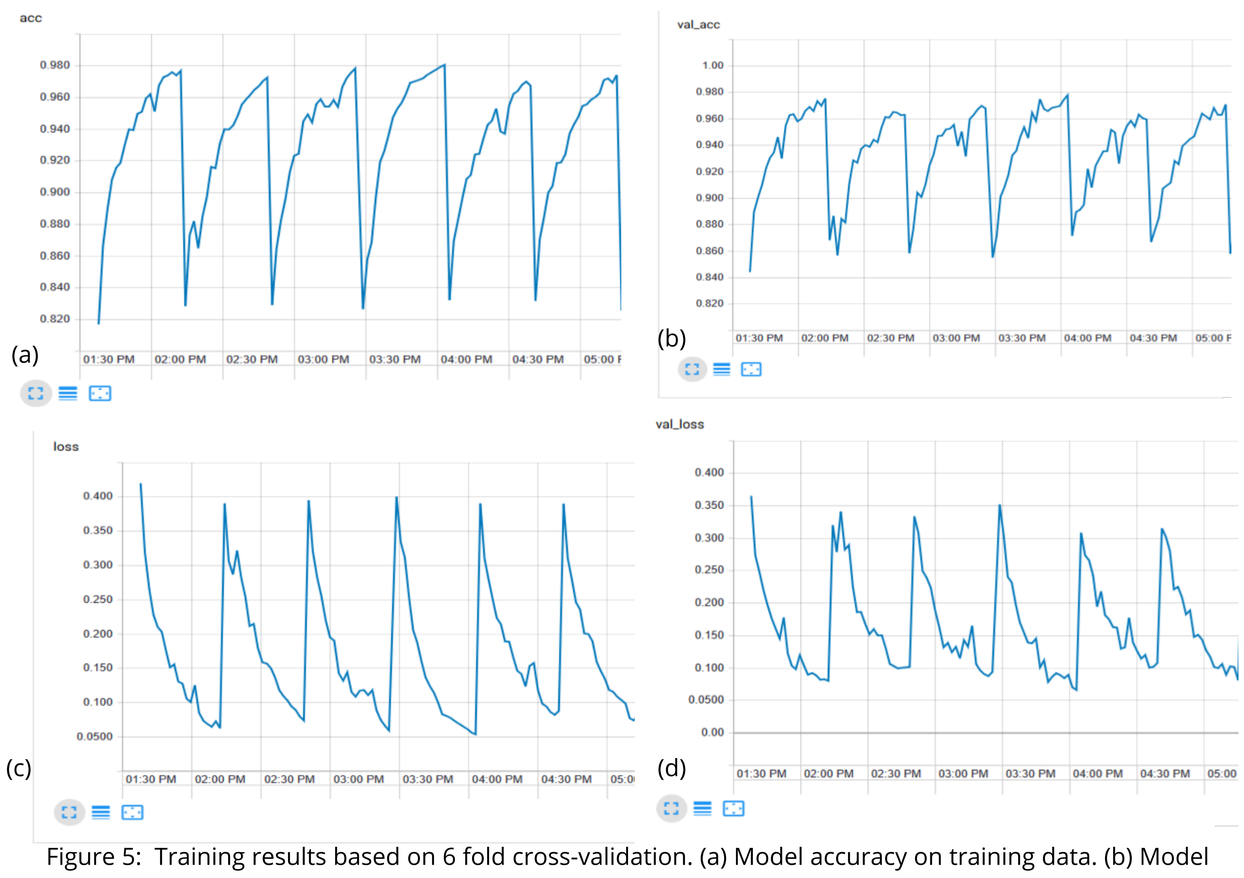

Figure 5: Training results based on 6 fold cross-validation. (a) Model accuracy on training data. (b) Model accuracy on validation data. (c) Model loss on training data. (d) Model loss on validation data.

Training Results: best accuracy = 98.06% (training set) 97.79% (validation set)

Conclusion and Future Work

The training results obtained are backing our expectations of this system being able to recognize user gestures in real-time. An immediately required addition would be to normalize the data to be independent of the player's height. Furthermore, we are planning to work on the second phase of building this system. In the upcoming phase, the system will take user-input and train on it so that it recognizes the gesture everytime the user performs it. Since it is highly unlikely that the user will perform the motion exactly the same each time they attempt to control the character. We, therefore, plan to generate a robust training set using only a small number of examples using a similar idea to the aforementioned technique. However, rather than generating variations of skeleton motion, we plan to fit parametric curves to trajectories from the HTC Vive trackers and perturb the motions in the parameter space of the curves. This includes not only spatial but also temporal variations of the motions. In this way, only a small number of example motions will be used to produce hundreds.