Motion Data and Machine Learning: Prototyping and Evaluation

Thierry Ravet
UMONS Numediart
31 boulevard Dolez
7000 Mons, Belgium
thierry.ravet@umons.ac.be

Nicolas d'Alessandro
Hovertone SPRL
4 rue des Soeurs Noires
B-7000, Mons (Belgium)
nicolas@hovertone.com

Jolle Tilmanne
UMONS Numediart
31 boulevard Dolez
7000 Mons, Belgium
joelle.tilmanne@umons.ac.be

Sohaib LARABA
UMONS Numediart
31 boulevard Dolez
7000 Mons, Belgium
sohaib.laraba@umons.ac.be

Abstract
In this work, we address the problem of graphical visualization to train and validate machine learning solution with motion capture data. We describe our experiment to build an efficient solution to explore and manipulate, spatially and temporally, motion data base. We present a prototyping tool for motion representation and interaction design based on the MotionMachine framework. This framework provides a coherent process chain to annotate the data, apply training algorithm and validate graphically the obtained results.

Author Keywords
Motion Capture, Gesture Recognition, Motion synthesis, fast prototyping

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous.

Introduction
Over the last ten years, an important amount of motion capture techniques have emerged. However most of these techniques – such as inertial suits or optical markers tracking – did remain expensive and cumbersome. More recently, the democratization of depth cameras – like the Microsoft Kinect – has considerably changed the scope of

markerless mocap research. Indeed the wide dissemination of these sensors gave many research groups the chance to enter the field and provide various resources (databases, software, results) to the scientific community [1].

Skeletal data acquisition generates a huge amount of high-dimensionality data. Such raw data is neither very readable, nor very reusable. Machine learning has appeared as a good way to extract the useful information in the studied data and to describe the properties of a collection of motion-captured data. A lot of authors have described how to use models like Randomized Decision Forests, Hidden Markov Models, Neural Networks or Restricted Boltzmann Machine in various tasks: interpretation of raw data [2], gesture recognition [3] motion synthesis [4] or animation retargeting [5].

If performance results for recognition tasks can be detailed in terms of accuracy and specificity, it is uneasy to quantify the quality of synthesized sequences regarding the dimensionality of the data. In such application, most studies present their method without giving information about the quality of their results, or just give a link to some sequences of synthesized motion. One solution is to use subjective tests. It requires good displaying conditions and tools to allow the evaluators to judge the naturalness of the resulting sequences.

Furthermore, data formatting and annotation to build the training data sets is not so obvious with skeletal data. Video annotation systems like Anvil [8] can be transposed to the mocap data domain but an adequate 3D space visualization system is an advantage by allowing the users to change the point of view. Indeed it is unlikely to correctly annotate a gesture without the capability to focus on the most informative skeletal joints.

In the next sections, we present MotionMachine, a motion data processing toolkit, and two different models that we trained with this framework. We highlight the advantages that this platform provides in the development of these machine learning models.

Related Works

Several tools exist to manipulate and visualize motion capture data with the goal of creating motion-enabled applications. The MOCAP toolbox [9] is among these tools, giving access to many high-level processing functions, but it is only available for Matlab and therefore not very suitable for real-time, iterative and interactive testing and performance. When it comes to performance-driven tools, we find software like the RAMToolkit [10] with the opposite issue: the tool is not generic and rather very specific to a single use case (Kinect-captured dancers or specific mocap markers placement). Other frameworks based on visual programming language allows motion data processing adapted for realtime streaming: we can mention EyesWeb [11] or Mubu [12].

MotionMachine Framework

MotionMachine is an open source C++ library that enables the rapid prototyping of motion features, their extraction on standardized motion capture data structures coming from typical motion capture file formats and live OSC streams and their selection so as to represent motion in the considered use case. It has first been introduced in [6].

General Structure

The overall data flow used in MotionMachine is presented in Figure 1. The library is built from two independent modules: one for data representation and feature

extraction (built on the top of the *Armadillo* C++ library [13]), the other to take care of 2D and 3D scenes visualization and general user interaction aspects (built on the top of the *openFrameworks* C++ library [15]).

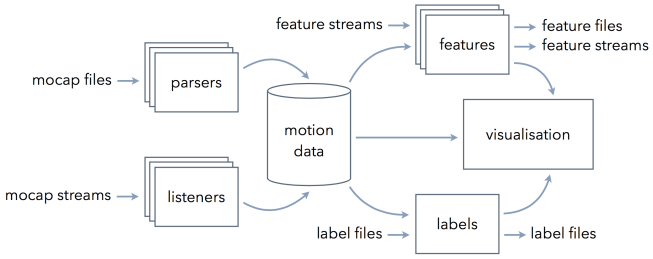

Figure 1: Overall data flow used in MotionMachine: modular structure to process mocap data files and/or streams into features files and streams, labels files and visualization[6].

Four important core features are available in the MotionMachine framework:

1. *Skeletal Model Independent Motion Data*: Most of motion capture devices provide skeletal data, i.e. changes in the position and/or orientation of 3D joints and/or segments. In MotionMachine, we have developed a model-independent formalism for storing and accessing such skeletal data through APIs[6].

2. *Collections of Motion Feature Extractors*: MotionMachine is built around the idea that developers can write custom code to be inserted in the motion capture data processing pipeline, while still preserving the intuitiveness and efficiency of the overall environment. The design principle underlying the available collection of feature extractors is essentially container-driven and based on the idea that offline batch and windowed real-time processing should both be available by default for any built-in or third-party feature.

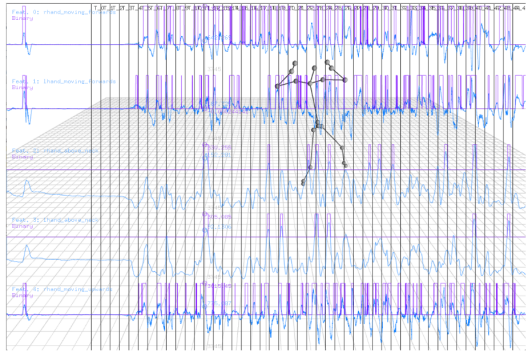

Figure 2: Visualisation of 3D scene and in 2D scenes(five features and annotations) for one contemporary dance sequence

3. *Interactive 2D/3D Scene View*: In MotionMachine, we wanted to improve the affordance of motion capture data processing by solving several visualization issues and bring the user faster to his/her valuable work. The library comes with an integrated 2D/3D scene viewer for displaying mocap data on screen and interacting with the contents [6]. 3D and 2D time lines are automatically synchronized helping to observe available motion capture data from different viewpoints.

4. *Annotation Layer*: In MotionMachine, we have integrated a lightweight annotation scheme. It allows the programmatic and UI-based insertion of *Labels* alongside the motion capture data. It means that the time tag of these *Labels* can be automatically derived from signal properties in the

feature extraction code or added manually by the user.

Motion Data and Machine Learning

In this section, we present two machine learning applications which have been implemented in MotionMachine to evaluate the platform efficiency in prototyping such processes.

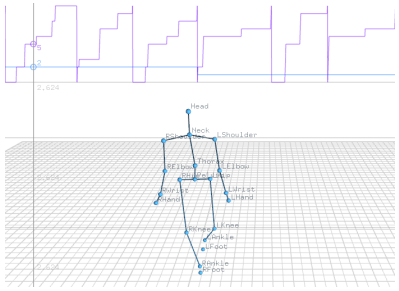

Figure 3: HMM-based gesture decoding of a dance step sample. Four HMMs (for four different dance steps) were trained with HTK. Each gesture was modeled with 10 states. On the top of the Figure, we can see the decoded gesture and the most likely current state. The cursor is pointing at a frame decoded as gesture 2 and state 5.

HMM Decoding with HTK

Our first application deals with gesture recognition. In previous works, we detailed how it is possible to implement a gesture decoding system based on Hidden Markov Models (HMMs) [17]. The HMMs represent the gesture as a succession of states. At each state, local statistics of the observations apply and both local statistics and state transition probabilities are determined by training. Hence it is possible to identify the most likely gesture corresponding to new observations and compute an approximation of the current state, which reflects the progression in the executed gesture.

The annotation of a traditional dance data collection was made in MotionMachine. The HMMs used for gesture recognition were trained using HTK (Hidden Markov Model Toolkit [16]). Each input data frame contains the body skeleton pose data. We can choose to describe this pose with the position of skeleton nodes or the orientation of the bones. The gesture recognition is performed by using a Viterbi algorithm. We used an implementation proposed by the *MLPack* library [14]. The Viterbi algorithm is applied on the whole temporal window of data stored in a MotionMachine Track data type. The decoding results (decoded gesture and most likely current state) are displayed in 2D scene view (see Fig. 3). A cursor is synchronized with the played skeletal data stream. This interface makes it easier for the user to verify the decoding timing accuracy and to visualize potential problems in the model.

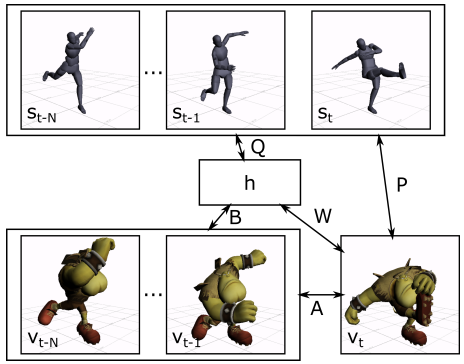

Figure 4: An Input-Output Temporal Restricted Boltzmann Machine: Input data are motion capture data performed by an actor. The output data are animation signal adapted for a specific virtual character.

Animation retargeting and difference visualization
This second application aims at automatically retargeting motion capture data to virtual characters for animation. As [4] and [5] have shown, Restricted Boltzmann Machine (RBM) can model temporal series such as motion data. We implemented the retargeting solution proposed by [5]. This algorithm provides a way to train a model that adapts automatically motion capture data for a given character.

We recorded a motion capture data collection of 10 gestures. Each gesture sequence was then manually adapted to the morphology of a cartoon-like virtual avatar by a 3D animator.

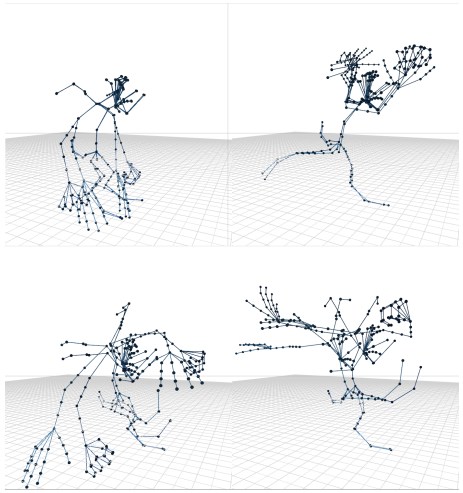

Figure 5: Visualization of the difference between motion captured poses and the synthesized data adapted to a virtual character.

With this double set of temporally aligned data, we trained a TRBM (temporal RBM) that respects the structure shown in figure 4. This TRBM model can then be used to adapt automatically new sets of motion capture data for the animation of the virtual character for which it was trained.

However, the comparison of original motion capture data and synthesized adapted data is often complicated for the human eye, as the differences can sometimes be very subtle. MotionMachine provides a user-friendly way to compare the data. To analyze the results, as we can see in figure 5, the difference between the original captured skeletal poses and the synthesized adapted data can be clearly put in evidence for each frame, and enable the user to visualize the influence of the model on the original data, and hence to verify accuracy of the machine learning process.

Conclusion

The high dimensionality of motion capture data makes it difficult for the user to manipulate it efficiently when designing machine learning based applications. As human beings, our understanding motion is mainly based on the visualization of the motion itself, displayed on a 3D character, rather than on abstract high-dimension coordinates time series. In this paper, we have described our proposal for a platform designed to help the user to prototype a machine learning processing chain for motion analysis, integrating what we consider important for such an interface. A 3D space visualization synchronized and superimposed with a 2D scene for rendering motion features time series allow the user to manipulate motion data collections, and to segment and annotate them efficiently. The framework that we propose provides parsing functions to extract the skeletal data from motion capture archives under different file formats. By implementing machine learning algorithms into the MotionMachine framework, we propose to follow the

visions of openFrameworks [15]: simplicity, intuitiveness extensibility. This results in a modular platform that can manage both offline motion data and real-time data streams.

Further works will include the addition of new motion analysis processes in the platform, as well as its use and validation in longer term projects and use cases. An evaluation of the input of the proposed vis-alizations for machine learning applications prototyping will be conducted. Furthermore, we will study the possibilities to export interactive 3D visualization for subjective evaluation with web technology like WebGl or mobile operating systems.

Acknowledgements

This work has been supported by the European Union (FP7-IC7- 2011-9) under grant agreement n 600676 (i-Treasures project) and by regional funds called Region Wallonne GREENTIC (convention number 1317957).

References

[1] Z. Zhang: Microsoft Kinect Sensor and Its Effects. IEEE Multimedia, vol.19, no.2, 4–10 (2012)

[2] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake:Real-time human pose recognition in parts from single depth images. In Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11), 1297-1304 (2011).

[3] Ying Yin, Davis, R.: Real-time continuous gesture recognition for natural human-computer interaction. IEEE Symposium onVisual Languages and Human-Centric Computing (VL/HCC), 113 - 120(2014)

[4] Graham W. Taylor, Geoffrey E. Hinton, Sam T. Roweis: Modeling Human Motion Using Binary Latent Variables. NIPS 2006: 1345-1352 (2006)

[5] Matthew D. Zeiler, Graham W. Taylor, Leonid Sigal, Iain Matthews, and Rob Fergus: Facial Expression Transfer with Input-Output Temporal Restricted Boltzmann Machines. Neural Information Processing Systems (2011)

[6] J. Tilmanne and N. d'Alessandro: MotionMachine: A New Framework For Motion Capture Signal Feature Prototyping. Proc. of EUSIPCO 2015.

[7] N. dAlessandro et al.: Towards the Sketching of Performative Control with Data. Proc.s of the eNTERFACE Summer Workshop on Multimodal Interfaces, 2013.

[8] Anvil, http://www.anvil-software.org/

[9] B. Burger and P. Toiviainen: MoCap Toolbox A Matlab Toolbox for Computational Analysis of Movement Data. Proc. of the 10th Sound and Music Computing Conference, (SMC) (2013)

[10] RAMToolkit, http://interlab.ycam.jp/en/projects/ram/ram_dance_toolkit

[11] The EyesWeb Project, http://www.infomus.org/eyesweb_ita.php

[12] Mubu, http://imtr.ircam.fr/imtr/MuBu

[13] armadillo, C++ linear algebra library http://arma.sourceforge.net/

[14] mlpack, scalable machine learning library http://www.mlpack.org/

[15] openFrameworks, C++ creative coding library http://www.openframeworks.cc/

[16] U. of Cambridge. The hidden markov model toolkit(htk), 2009. http://htk.eng.cam.ac.uk

[17] Thierry Ravet, Jolle Tilmanne, and Nicolas d'Alessandro. 2014. Hidden Markov Model Based Real-Time Motion Recognition and Following. In Proceedings of the 2014 International Workshop on Movement and Computing (MOCO '14).