MOTION MACHINE: A NEW FRAMEWORK FOR MOTION CAPTURE SIGNAL FEATURE PROTOTYPING

Joëlle Tilmanne, Nicolas d'Alessandro

TCTS lab, numediart Institute, University of Mons, Belgium

ABSTRACT

Motion capture (mocap) is rapidly evolving and embraced by a growing community in various research areas. However, a common problem is the high dimensionality of mocap data, and the difficulty to extract and understand meaningful features. In this paper, we propose a framework for the rapid prototyping of feature sets, *MotionMachine*, which helps to overcome the standard problem of mocap feature understanding by an interactive visualisation of both features and 3D scene. Our framework aims at being flexible to input data format, and to work both offline or in real-time. The design of the feature extraction modules in C++ is intended for modules to be used both for visualisation in the MotionMachine framework and for integration in end-user applications or communication with other existing softwares. We present two examples of use-cases in which the main features of this framework have successfully been tested.

Index Terms— mocap, feature prototyping, library

1. INTRODUCTION

Motion capture (mocap) is the process of recording a live motion event and translating it into mathematically-usable signals. These signals correspond to the tracking of a number of key points in space over time, combined as one 4D (three spatial dimensions plus time) representation of the performance [1]. A *skeletal* model is commonly adopted for full-body and hand motion tracking. These models represent motion as the evolution of the 3D cartesian coordinates of the joints or the 3D angles between body segments over time. Raw skeletal data can be used to play the captured performance back on 3D animated stick figures, but observing and manipulating the raw position or angle data directly is extremely difficult.

Nowadays mocap is substantially moving from a niche technology to the mainstream. On the one hand, expensive and complex professional mocap systems (such as optical marker-based or inertial systems) are becoming more affordable and easier to use. On the other hand, low-cost motion capture sensors (e.g. depth cameras) are making their way into consumer products. As a result, mocap is spreading into the workflow of a growing number of practitioners in fields such as biomechanics, animation, health, sports, humanities, arts, functional interactions, etc.

Because raw skeletal data is particularly unusable, a large majority of applications using mocap data for motion analysis, editing, synthesis or recognition require some kind of motion parametrisation. Such parameterisation can vary from straightforward editing of the raw data (e.g. scaling, referential change) to seeking for higher-level motion features such as "smoothness" or "activity" [2]. There are many disciplines that could benefit from advances in motion capture technologies and, in most cases, the performance of the designed application strongly depends on how the motion processing chain is adapted to the considered use case.

Figuring out such adequacy requires to prototype motion feature extraction schemes. By prototyping, we mean the ability to rapidly define motion features, extract them on pre-recorded or live-streamed mocap data and select the most appropriate set of features. However there is currently a lack of motion feature extraction libraries or even of recognized standard features, contrarily to areas such as audio or image signal processing. We foresee four aspects to take into account so as to address this situation:

- There is actually no existing formalism to skim through an existing set of mocap signal processing algorithms and rapidly create, aggregate and test new motion features. As a result, the existing corpus of motion features extraction algorithms is not easily accessible nor extendable.
- Motion processing chains are nowadays highly depending on a given sensor or platform. Building sensor-independent skeletal models, and therefore skeleton-to-skeleton adaptation mechanisms, could greatly help the transfer of existing motion features to new use cases.
- There is a lack of adequate visual feedback and interaction to evaluate the relevance of a given attempt. Often animating 3D stick figures and prototyping motion features are achieved in incompatible environments. Furthermore, no satisfactory annotation tool exist for mocap data.
- There is a need for the prototyped motion feature extraction algorithms to be directly applicable to real use cases.

In this paper, we introduce *MotionMachine*[1], a new tool for rapidly prototyping motion features, so as to adequately fit the considered use cases. It comes as a cross-platform C++

[1] www.numediart.org/motionmachine

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

API enabling the creation and extraction of motion features on mocap data and the real-time visualisation of such features in combined and synchronised 2D and 3D scenes. Section 2 gives an overview of related work in the field. In Section 3 we present the tool and its functionalities. Then we describe two scenarios in which *MotionMachine* is useful in Section 4. Finally we highlight conclusions and future work.

2. RELATED WORK

As presented in the introduction, mocap data feature prototyping is a research field beneficial to any application using 3D mocap data. However, in most works the discussion about the features themselves is overlooked. In many cases, no feature extraction is performed at all and the raw data (Cartesian coordinates or 3D angles) is used almost directly as an input. This will often lead to non optimal results because the high dimensionality of the data will hamper the robustness of the application. In the case of machine learning based applications, the mocap training datasets are often quite small compared to the dimensionality of the data. Some a priori knowledge about the motion should hence be considered, through an efficient feature selection, before training the models. Very low level features such as velocity, acceleration or angle between segments are some of the most common features encountered in various works, e.g. in [3, 4].

Principal component analysis (PCA) is a common way of reducing the dimensionality of mocap data, e.g. [5–7], based on the assumption that, despite the high dimensionality of the original motion description space, most human movements have an intrinsic representation in a lower dimensional space [8]. However such an approach is highly dependent on the set of training data and can hardly be generalized to new motions.

Some previous work have proposed interesting sets of features for motion analysis in different contexts. For instance, Müller proposes a set of 39 geometry-based relational binary features (e.g. hands above head, hand moving forwards, etc.) [9]. Such features are more suitable for generic applications than features based on raw numerical data.

In addition to low level features and relational features, many other features have been proposed in different use cases. However, they are hardly ever the focus of the research and are rarely adopted in other use cases nor compared to other features. We only name a few to illustrate the diversity of the approaches. Kahol et al. propose a feature which they call "activity" which represents velocity, acceleration and mass of body segments, and which is measured across a dynamical hierarchical model of the human body [2]. Megali et al. use a short-term Fourier transform (STFT) of acceleration combined with a K-means clustering to transform the STFT acceleration vector into mono-dimensional space [10]. Many motion features have also been proposed in the field of motion analysis based on RGB videos (e.g. [11]), which has been active for way longer than the 3D mocap data field, and could be adapted to mocap data in the future.

Burger and Toiviainen propose a Matlab toolbox for analysing and visualizing mocap data which is aimed at investigating music-related movement [12]. However all features and skeleton visualisations are static, and the Matlab based implementation is not designed for integration in final applications. Alemi et al. [13] have recently presented Mova, an interactive movement analytics framework. Their work concentrates on the feature extraction and visualisation for motion analysis. The interaction mainly consists in being able to chose the features to visualize.

Another important missing functionality for mocap data is the lack of an appropriate annotation tool. Annotations are often an essential aspect when designing new features, for instance for discriminating between specific gestures, or to isolate the useful parts of a continuous mocap data recording. Anvil, a software developed originally for video annotation has a plugin which supports only BVH files, one specific mocap data format [14]. Moreover, the plugin requires an RGB video to be loaded at the same time as the mocap data. Motion curves can also be visualized at the same time as the 3D scene. However the interface is not open source nor very flexible, and is not designed for the addition of other features.

Very few works propose a framework for mocap feature extraction and all of them target the motion analysis use case. No existing platform aims at rapid prototyping of feature extraction, visualisation, understanding and selection for any kind of application. In *MotionMachine* we aim at proposing such a framework focused on the mocap signal feature prototyping problem, by choosing an appropriate architecture.

3. ARCHITECTURE

MotionMachine is a C++ library that enables the rapid prototyping of motion features, their extraction on standardised mocap data structures coming from typical mocap file formats and live UDP streams and their selection so as to represent motion in the considered use case. The overall data flow used in *MotionMachine* is presented in Figure 1. The library is built from two main modules: one for feature extraction, the other to take care of 2D and 3D scenes visualisation. In the following subsections, we show how the tool addresses the feature extraction issues mentioned in Section 1.

3.1. Skeletal Model Independent Motion Data

Nowadays the number of available mocap sensors is significantly growing. Most of these devices provide skeletal data, i.e. changes in the position and/or direction of 3D joints and/or bones. For the moment, most of the achieved motion processing is model-specific and the corresponding know-how is not directly transferrable to another model, though the underlying morphology is similar.

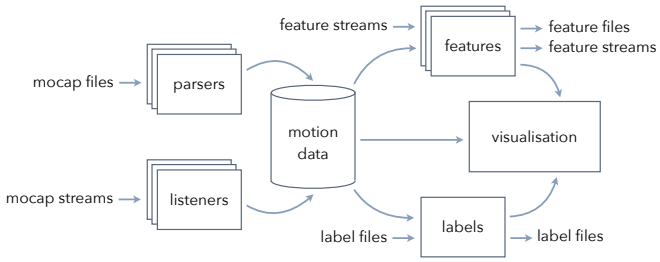

Fig. 1. Overall data flow used in *MotionMachine*: modular structure to process mocap data files and/or streams into features files and streams, labels files and visualisation.

In this work, we propose a model-independent hierarchical formalism for mocap data storage and manipulation. This formalism is based on imbricated data structures that are aimed at being generic, configurable and easily extensible, so as to fit most of the motion capture situations:

- *Node* stores the 3D position and/or orientation of a joint. A name field can optionally be associated with the *Node*.
- *Frame* is a time-tagged aggregation of *Nodes*. A bone structure connecting the *Nodes* can optionally be added.
- *Trace* is an aggregation of time-tagged *Nodes* corresponding to the same body joint. It represents the evolution of that *Node* over time.
- *Track* is an umbrella over the whole mocap data file. It can be sliced according to *Frames* or *Traces*.

All *MotionMachine* data types can be loaded/saved from different file formats. Currently, the supported formats are our open flat file format and two typical mocap file formats: BVH and C3D. The structure of data types is illustrated in Figure 2.

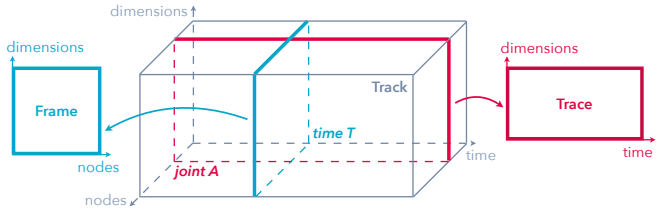

Fig. 2. Schematic view of the structure of mocap data in *MotionMachine*: *Track* is sliced into *Frame* or *Trace*, then into *Node*. *Node* contains position and/or orientation.

3.2. Feature Extraction Plugins

Rapid prototyping of motion processing chains requires to work with a high-level consistent syntax and to build up system complexity by growing a list of standardised modules. Therefore we have integrated a plugin architecture, so as to quickly extend the default function set with custom feature extraction techniques. We have already tested this approach with several motion feature extraction algorithms:

- A *Signal Processing* toolbox with a set of basic algorithms such as mean, min/max, thresholds, peak picking, etc.
- A *Geometry* toolbox with geometrical primitives: point-to-point and point-to-plane distances, point velocity in the 3D referential or projected in a given direction or plane, angle between bones, bounding box features, etc.
- The full implementation of Müller's full-body continuous and binary motion features, as described in [9].
- A set of experimental features that aim at describing the hand postures: open/closed, orientations, intra-finger extensions, left/right asymmetry, pointing finger, etc.

3.3. Interactive 2D/3D Scene View

Among the most critical aspects of prototyping a motion processing chain, we find the instantaneous user feedback. It often takes time for the user to understand complex mocap-related features and figure out how to display the right information so as to evaluate the relevance of his/her approach. Moreover integrating user interaction (such as manipulating the 3D scene or reading a feature value by hovertone the displayed graph with the mouse) often leads to extra work, irrelevant for the studied case.

In *MotionMachine*, we wanted to improve the affordance of mocap data processing by solving several visualisation issues and bring the user faster to his/her valuable work. Such improvements were achieved by balancing the apparent complexity of the environment. As a result, the library comes with an integrated 2D/3D scene viewer for displaying mocap data on screen and interacting with the contents. Here are several key aspects that this tool solves at the mocap signal level:

- An important aspect of mocap visualisation is to map a given set of motion signals onto the right representation. In our data structure, we handle these possible mistakes: *Node* jointly handles position and/or orientation signals and the 3D visualisation adapts accordingly.
- Motion capture data is prone to have holes in the signal streams, i.e. times for which a given *Node* position or orientation is undefined. Misinterpreting such holes in the data can lead to wrong feature extraction and visualisation. Our *Frame* visualisation scheme takes care of those possible degradations and adapts accordingly: we have descriptive formalisms for missing that are handled by both the querying and drawing mechanisms.

3.4. Annotation

As described in Section 2, the annotation functionality is very important and it is hard to find a tool that accurately annotates mocap sequences. In *MotionMachine*, we have integrated a lightweight annotation scheme. It allows the programmatic insertion of *Labels* in the motion capture *Track*. It means that the time tag of these *Labels* can be automatically derived from

signal properties in the feature extraction code. Moreover these *Labels* get properly rendered in the 2D view and can be rearranged manually. *Labels* can also be imported from and exported to label text files (`.lab` extension).

4. USE CASES

In this Section, we introduce two use cases in which *MotionMachine* brings improvements in the motion data representation and the interactive prototyping of feature extraction.

4.1. Contemporary Dance Analysis

In this first use case, our goal is to analyse contemporary dance. Our contemporary dance database has been recorded in the framework of the i-Treasures European Project [2], and consists in free improvisations of six contemporary dancers on five 45-second music extracts. The 30 mocap sequences were recorded at 177 fps thanks to a Qualisys optical motion capture system. After post-processing, the skeleton was represented by 22 nodes. The complete analysis of our database is beyond the scope of the present paper. However the choice and fine tuning of a representative set of features required for our analysis was conducted using the *MotionMachine* framework, which facilitated the whole process. This analysis required the segmentation of the mocap sequence corresponding to the 45 seconds of music into one-second segments.

A bar of four beats had been added at the beginning of each original sound file, and the dancers were asked to perform a "clap" motion. The synchronisation of the mocap files with the original soundtracks, was performed by calculating the distance between both hands and extracting the first peak in the mocap sequence. The annotation function was used to generate labels for 45 one-second segments, based on this synchronisation. The features extracted were the 39 relational features proposed by Müller [9]. Müller's features are frame-by-frame binary decisions obtained by thresholding a set of continuous features computed on the full-body skeleton. The visual feedback enabled us to tune those thresholds in the conversion of continuous features to binary ones as well as to verify the soundness of the automatic segmentation. Finally the binary features had to be converted from one value per frame to one value per segment, and exported to XML file format.

All these operations only required a few lines of code thanks to the high-level functions proposed in *MotionMachine*, and the extraction of the same feature files for new dance recordings can be generated by a simple drag-and-drop of the new motion file in the visualisation window, illustrated in Figure 3. Our finished prototyping scheme can hence be released as a standalone API which will allow the user to drag an drop new mocap sequences corresponding to the same 45 seconds dance sequence preceded by a clap motion. The 39 Müller features will automatically be extracted, the dance sequence will be isolated and segmented into one second segments, the corresponding labels will be saved as a text file and features will be averaged on these segments and saved in XML format using the *TinyXML* library.

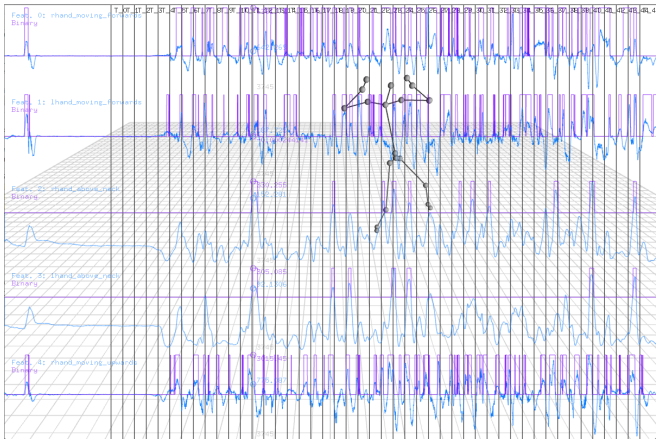

Fig. 3. Visualisation of 3D scene and five Müller features (continuous in blue and binary conversion in purple) and automatic annotation for one contemporary dance sequence.

4.2. Bi-Manual Musical Instrument

In this second use case, we have used *MotionMachine* to prototype the *mapping* layer of a new digital musical instrument (DMI). Wanderley *et al.* define the topology of a DMI with three main parts: the sensors, the sound synthesis engine and the mapping [15]. The role of the mapping is therefore to find the most appropriate interpretation of the performer's gestures and connect it to determined actions on the synthesiser.

New end-user depth cameras like the LeapMotion now give access to very sophisticated 3D skeletal models of the hands. In the current SDK, each hand is represented by 21 3D joints (position, orientation and velocity) plus arm and palm 3D position, orientation and velocity. Though very detailed, such raw data is barely usable as is and higher-level representations of hand motion are a necessary mapping strategy for the development of a usable musical instrument.

The ability of *MotionMachine* to run similarly on offline mocap data and online UDP streams makes it a very appropriate tool for designing mapping layers in DMIs. In this work, we have used our tool to prototype two higher-level motion features to be used in the control of a sound synthesiser:

- Pointing finger attributes: by clustering fingertip distances to a given reference plane, we are able to narrow down "pointing finger" positions and directions, i.e. fingers that bend more significantly towards the reference plane.
- Opening ratio of the hand: by computing the average angle between all the metacarpal bones and the palm, we are able to have a stable ratio for the hand closure, going from flat fully-opened shape to fist-like fully-closed shape.

[2] `www.i-treasures.eu`

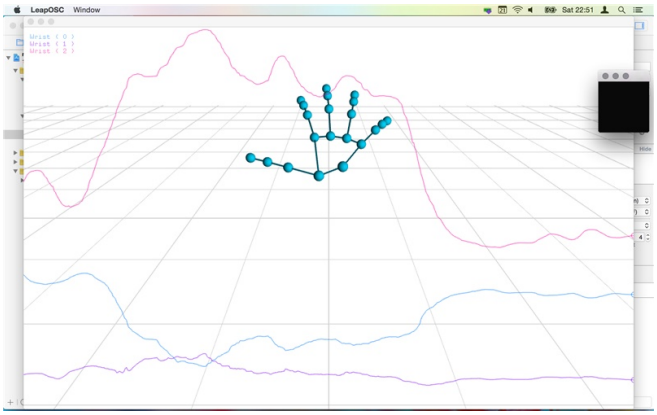

Fig. 4. Visualisation of the right hand skeletal model. Data stream comes from a UDP sender and the 2D view represents the wrist x, y and z positions.

In our first iterations, we have validated the consistency of these two motion features with the integrated visual feedback. We have also directly tested the usability of the musical instrument, thanks to the realtime UDP streaming of these motion features to the sound synthesiser. Such features have been rapidly developed. It encourages us to envision that many more could be tested in a reasonable amount of time.

5. CONCLUSION

In this paper we have presented a new framework for the prototyping of mocap signal features. We believe such a framework could be beneficial in most mocap-based research areas, such as motion analysis, synthesis, edition or gesture recognition. We described several issues that our framework is addressing: making motion features more understandable thanks to a superimposed 2D/3D scene visualisation, facilitating the use both offline (with recorded files) and online (UDP data streams), and ensuring maximum flexibility regarding the input data with generic 3D points and any possible skeleton configuration. Two use cases have been presented to illustrate some of the functionalities offered by *MotionMachine*. Further work will include the implementation of additional features, integration of automatic feature suggestion and/or selection techniques, and the test of new use cases.

REFERENCES

[1] A. Menache, *Understanding motion capture for computer animation and video games*, Morgan Kauffman Publishers Inc., 2000.

[2] K. Kahol, P. Tripathi, and S. Panchanathan, "Automated gesture segmentation from dance sequences," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004.* 2004, pp. 883–888, IEEE.

[3] F. Bevilacqua, L. Naugle, and I. Valverde, "Virtual dance and music environment using motion capture," *Proc. of the IEEE-Multimedia Technology And Applications Conference, Irvine CA*, 2001.

[4] F. Kistler, D. Sollfrank, N. Bee, and E. André, "Full body gestures enhancing a game book for interactive story telling," in *Interactive Storytelling*, vol. 7069 of *LNCS*, pp. 207–218. Springer, 2011.

[5] P. Glardon, R. Boulic, and D. Thalmann, "PCA-based walking engine using motion capture data," in *Computer Graphics Int.* 2004, pp. 292–298, IEEE.

[6] K. Forbes and E. Fiume, "An efficient search algorithm for motion data using weighted PCA," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 2005, pp. 67–76.

[7] M. Leman and L. Naveda, "Basic gestures as spatiotemporal reference frames for repetitive dance/music patterns in Samba and Charleston Charleston," *Music Perception*, pp. 71–91, 2010.

[8] C.-S. Elgammal, A.and Lee, "The role of manifold learning in human motion analysis," in *Human Motion*, vol. 36 of *Computational Imaging and Vision*, pp. 25–56. Springer, 2008.

[9] M. Müller, *Information retrieval for music and motion*, Springer, 2007.

[10] G. Megali, S. Sinigaglia, O. Tonet, and P. Dario, "Modelling and Evaluation of Surgical Performance Using Hidden Markov Models," *IEEE Trans. on Biomedical Engineering*, vol. 53, no. 10, pp. 1911–1919, 2006.

[11] A. Camurri, B. Mazzarino, and G. Volpe, "Analysis of expressive gesture: The eyesweb expressive gesture processing library," in *Gesture-Based Communication in Human-Computer Interaction*, vol. 2915 of *LNCS*, pp. 460–467. Springer, 2004.

[12] B. Burger and P. Toiviainen, "MOCAP TOOLBOX – A Matlab Toolbox For Computational Analysis Of Movement Data," in *Sound and Music Computing Conference*, Sept. 2013, pp. 1–7.

[13] O. Alemi, P. Pasquier, and C. Shaw, "Mova: Interactive movement analytics platform," in *Proceedings of the 2014 International Workshop on Movement and Computing (MOCO'14)*. 2014, pp. 37–42, ACM.

[14] M. Kipp, *Multimedia Annotation, Querying, and Analysis in Anvil*, pp. 351–367, John Wiley and Sons, Inc., 2012.

[15] D. Van Nort, M. Wanderley, and P. Depalle, "Mapping control structures for sound synthesis: Functional and topological perspectives," *Computer Music Journal*, vol. 38, no. 3, pp. 6–22, Sept 2014.