D3.1 - Data acquisition and analytic tools for individual actions

Project No	GA824160
Project Acronym	EnTimeMent
Project full title	ENtrainment & synchronization at multiple TIME scales in the MENTal foundations of expressive gesture
Instrument	FET Proactive
Type of action	RIA
Start Date of project	1 January 2019
Duration	48 months

Distribution level	[PU][1]
Due date of deliverable	Month 18
Actual submission date	July 2020
Deliverable number	3.1
Deliverable title	Data acquisition analytic tools for individual actions
Type	ORDP (Open Research Data Pilot)
Status & version	
Number of pages	
WP contributing to the deliverable	3
WP / Task responsible	UNIGE
Other contributors	ALL
Author(s)	UCL, DU, UNIGE, KTH, Universiteit Maastricht
EC Project Officer	Teresa De Martino
Keywords	**Computational models, Data collection, Machine learning, Movement analysis, Software libraries, Sonification**

[1] **PU** = Public, **PP** = Restricted to other programme participants (including the Commission Services), **RE** = Restricted to a group specified by the consortium (including the Commission Services), **CO** = Confidential, only for members of the consortium (including the Commission Services).

Contents

Abbreviations

EU	European Union
EC	European Commission
IMU	Inertia Measurement Unit
WP	Work Package

1 Introduction

This deliverable describes the hardware and software tools used for acquiring and analysing data in the context of individual action. Some of the content of the deliverable has been reported as deliverable D3.2 (Phase 1) which is primarily a software deliverable. A more comprehensive and more detailed description is given in the current deliverable.

2 Data Acquisition Tools

This section describes the software, sensors, and other devices used for capturing data in four main settings of the EnTimeMent project: controlled lab settings, unconstrained lab settings, musical performance settings, and everyday (e.g. home) settings.

2.1 Neuroimaging

2.1.1 Neural recordings

Recording of neural signals in response to human vision of bodies and movements will be performed with ultra-high field (UHF) 7-Tesla Magnetic Resonance Imaging (MRI) using a Blood Oxygen Level Dependent (BOLD) acquisition protocol. BOLD signals will be recorded with a field of view (FOV) covering the whole brain at a repetition time (TR) of ~1 sec and at a voxel resolution of ~1.6mm^3. The recorded signals will be saved in DICOM format and be the basis for further signal analyses (see Sections 3.1.1 and 3.2.6).

2.1.2 Stimulus presentation

Experimental setup: Stimuli will be recorded with the above-mentioned equipment. Stimuli will consist of actors performing various actions recorded in parallel with both mocap and normal videos. The stimuli will be presented to human subject in the MRI scanner via a projection screen at the end of the MRI bort connected to a laptop or PC. Participants will see the screen through a head mounted mirror attached to the head coil.

2.2 Sensing Architecture for Multimodal MoCap Lab Studies

The current recording architecture is represented in Figure 1, and it is composed of:

1) Cameras - 16 OQUS cameras (mixed setup with OQUS 700+, OQUS 700, OQUS 300) and 2 professional (i.e. broadcast) cameras

2) Microphone - a respiration microphone

3) IMU Sensor - 2-4 accelerometers

4) Other Sensors -There is the possibility of adding other sensors, e.g. biometric sensors, or other devices.

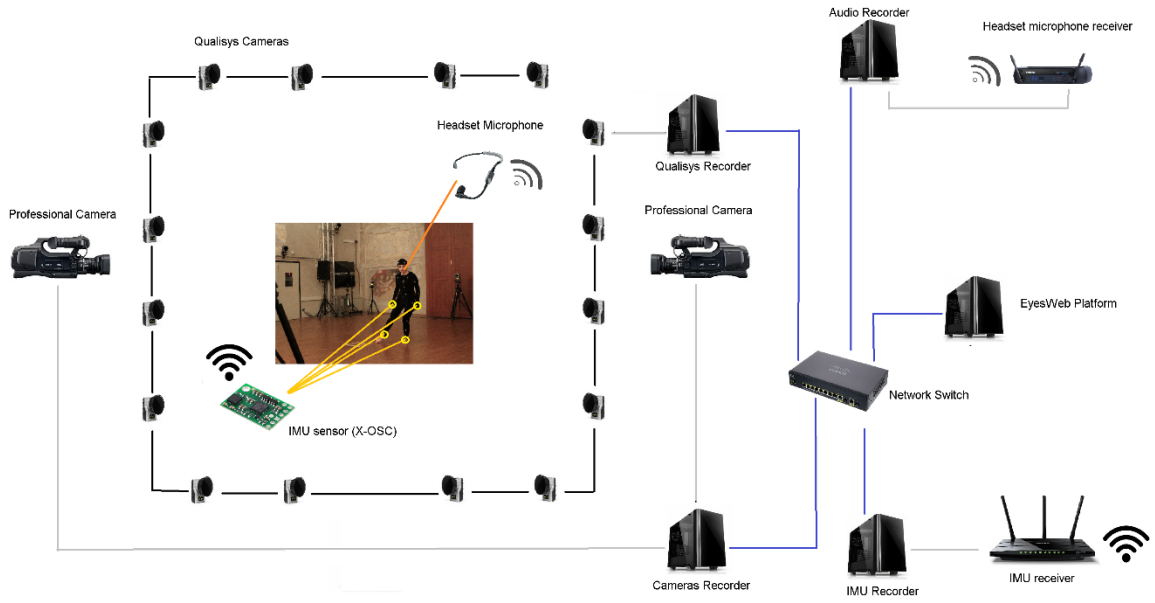

Figure 1 Sensing Architecture for multimodal MoCap Lab Studies

5) Synchronization system - Synchronization of the multiple sensors is guaranteed by the EyesWeb software platform. EyesWeb is used to generate the reference clock used by all other devices. The generated reference clock is sent to the recorders in a format compatible with each specific device. The synchronization signal is called SMPTE. SMPTE timecode is a set of cooperating standards to label individual frames of video or film with a time code defined by the Society of Motion Picture and Television Engineers.

6) The Qualisys Motion Capture system receives SMPTE encoded in an audio stream. Also, the two broadcast video-cameras and the *Audio Recorder* use SMPTE encoded as an audio signal. The *IMU Recorder* receives the reference clock via network, through the OSC protocol.

7) To guarantee synchronization, for every recorded frame or sample, EyesWeb keeps track of the SMPTE when the data is received. As a matter of fact, not all streams can be hardware-synchronised (e.g., with a genlock signal). Therefore, a software synchronization is performed by EyesWeb by keeping track of the time stamp at which the data is received in a separate file. This information is used to synchronize the read or play of multimodal data. IMU sensors data is an example of a device synchronised in this way with other channels.

8) Qualisys Recorder - dedicated to the processing and the recording of data streams from Qualisys cameras. It computes the motion tracking by Qualisys Track Manager (QTM). In the latest version of the software, QTM provides both markers and skeleton information in real time. The recorded data can be exported in C3D or TSV format.

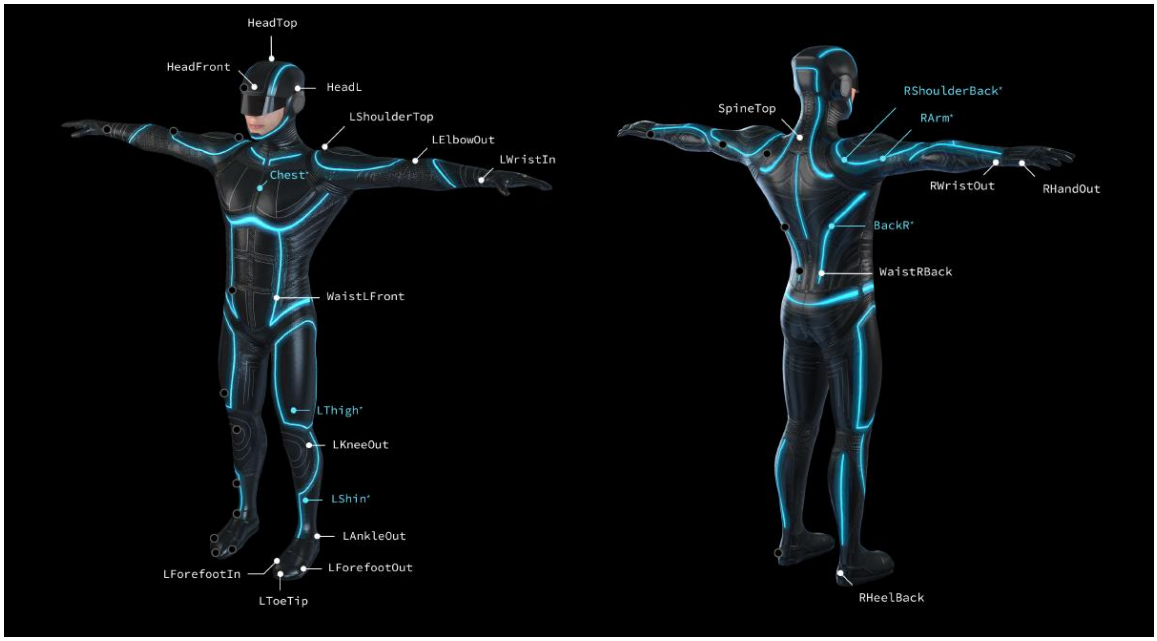

Figure 2 The Qualisys Animation Marker Set for 3D animation applications.

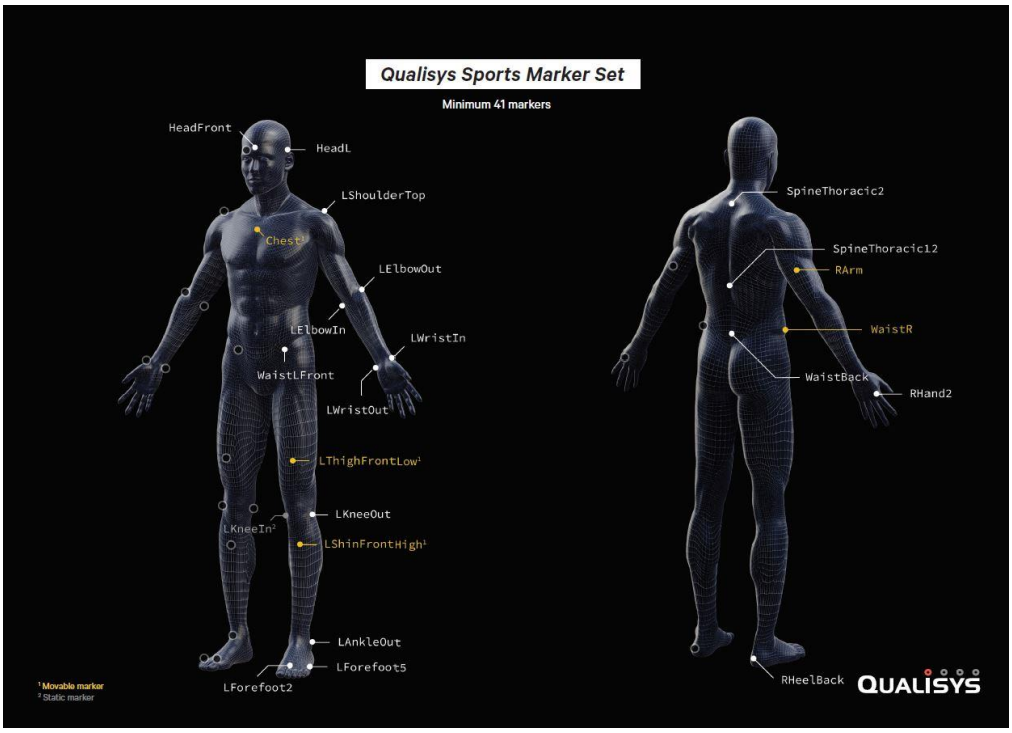

Figure 3 The Qualisys Sport Marker Set, adopted in recordings for several experiments described in D1.2

In EnTimeMent, we adopted the two specific marker sets recently introduced by Qualisys for 3D animation (Animation Marker Set, Figure 2) and sports (Sports Marker Set, Figure 3). Both marker sets greatly improve the skeleton solver performance. The experiments involving MoCap recorded at the UNIGE premise (see deliverable D1.2) are based on the Sports Marker Set (Figure 3). The recorder module streams synchronised data to the EyesWeb Platform, using the SMPTE signal sent by EyesWeb software to the Qualisys SyncBox, to obtain a fine-grain synchronization of all multimodal channels.

9) Video Recorder - 2 professional cameras are recorded, synchronised with the SMPTE signal. The ambient audio is recorded in the left audio channel of the video. The SMPTE signal is recorded in the right audio channel.

10) Audio Recorder - radio headset microphones send the audio signal (voice, or respiration) to this module. Each track contains the respiration or the voice signal of the user.

11) IMU Recorder – In several experiments the user wears IMUs. Typically, 2 sensors on the wrists and 2 on the ankles are the minimum setup. The recorder stores all the synchronised signals from each IMU: acceleration, gyroscope, magnetometer.

12) EyesWeb Platform - The EyesWeb platform has been updated to:

- synchronise the selected devices adopted in the project and store the obtained synchronized multimodal channels;
- manage the recording sessions;
- playback and visualization of the recorded synchronised signals;
- analyse both in real time and off-line the signals

13) The Skeleton Receiver is a further new software module developed in EyesWeb for this EnTimeMent platform. EyesWeb thus now supports the input of skeleton data both in real time and from *.qtm* exported files. The skeleton stream is composed of a hierarchical set of segment positions (in millimetre) and the segment rotation data expressed as quaternions.

2.3 Sensing Systems for Music Performance Settings

Music performances are captured using professional audio-visual recording equipment. Recordings made for the project use multiple (usually at least 3) professional high-definition video cameras; audio is recorded using separate microphones to a ProTools workstation, achieving the best possible separation between instruments. Different media streams are synchronised using time code/genlock.

2.4 Sensing System for Everyday Settings

A much more flexible architecture will be used in settings that demand high flexibility (e.g. participant homes, gym) or lab contexts designed to reflect such settings, i.e. where the recording is more ubiquitous and not relegated to a specific recording space. Due to the constraints of such settings, wearable sensors will be used wherever it is the practical option.

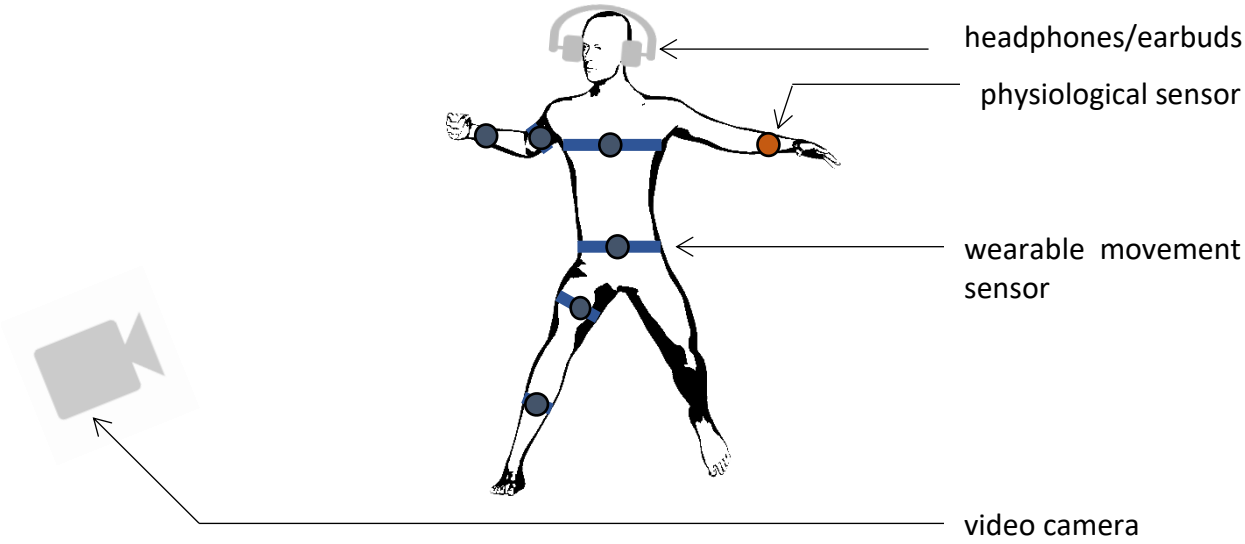

Figure 4 Sensor Architecture for Everyday Settings

As shown in Figure 4, the sensor system is composed of four possible components:

1) Body movement sensors - These sensors will mainly be of wearable form. They will record angle and/or position data for one or more anatomical joints.

2) Physiological sensors - These will capture and record physiological signals (e.g. electrodermal activity). They will be used when it is practical to do so. For example, it may not be practical to use it when the participant will perform activities where their hands will be in water (e.g. when washing a bathtub).

3) Video cameras - Video cameras will be used to additionally capture body movements of participants for the purpose of observer analysis. Again, these will only be used as much as is practical. Pilot studies showed that activities in the home, for example, may involve multiple spaces in the home making it impossible to avoid the participant going out of camera view.

4) Headphones/Earbuds - Where it is helpful to do so, we may use headphones or earbuds to record participant responses and may additionally provide automated instructions (e.g. self-report prompts) or sonification through these. Otherwise, audio will be recorded using video cameras or audio microphones, instructions will be provided verbally, and sonification will be through speakers (phone speakers or external speaker).

3 Data Analysis Tools: Software, Libraries, and Computational Models

In this section, we describe the tools and algorithms used for data modelling and movement sonification as well as protocols and methods used for preparing or pre-processing the data.

3.1 Data Preparation and Pre-processing Tools

Here, we discuss the techniques used for preparing the different forms of data.

3.1.1 fMRI Data Pre-processing

The recorded BOLD signals in DICOM format will be transported from the 7T MRI to dedicated workstations for analyses. First, the DICOM files will be converted to 4D data matrices representing the brain volumes (x,y,z) over time (t). A number of pre-processing steps will be performed on the data before final analyses commences. Publicly available software libraries in addition to custom MATLAB scripts will be used for the analysis of the functional MRI (fMRI) data.

3.1.2 Video Data Pre-processing

Music performance clips are selected for analysis and saved as synchronised media files (i.e. separate audio and video files, *.wav* and *.mp4*, with the same start and end points). Metadata is created listing key information such as performers names and item performed.

Video files are analysed using OpenPose pose estimation software (https://github.com/CMU-Perceptual-Computing-Lab/openpose). OpenPose output in the form of multiple JSON files (one for each video frame) is processed by selection of body parts (e.g. for Indian classical musicians, lower body data is discarded), and noise reduction using a movement smoothing algorithm. Data are combined and output as *.csv* tables of coordinates for each selected body part. Visualisation of processed data overlaid on original video views allows data quality check (e.g. that body parts are not confused with clothing items or between individuals). All processes are saved as Python scripts.

3.1.3 Features Engineering

In this section we describe how the features can been extracted and engineered from the raw data in the context of analysis based on a traditional machine learning model. The timeseries of the different measurements are sampled in fixed-width sliding windows. From each sampled window a vector of features is obtained by computing standard measures previously employed in literature to describe human actions such as the mean, the signal-pair correlation, and the signal magnitude area for both the time and frequency domains. The Fast Fourier Transform can be exploited to find the frequency components for each window. A new set of features can be also employed in order to improve the learning performance, including energy of different frequency bands, frequency skewness, and frequency kurtosis. Table 1 below, contains the list of all the measures applied to the time and frequency domain signals.

Table 1. List of possible measures for computing feature vectors.

Function	**Description**
Mean	Mean value
Var	Variance
Mad	Median absolute value
Max	Largest value in array
Min	Smallest value in array
SMA	Signal magnitude area
Energy	Average sum of squares
Iqr	Interquartile range
Entropy	Signal Entropy
Correlation	Correlation coefficient
Kurtosis	Signal Kurtosis
Skewness	Signal Skewness
MaxFreqInd	Largest Frequency component
argMaxFreqInd	Index largest frequency component
MeanFreq	Frequency signal weighted average
SkewnessFreq	Frequency signal Skewness
KurtosisFreq	Frequency signal Kurtosis
ampSprec	Amplitude Spectrum of the frequency signal
Angle	Phase angle of the frequency domain

3.1.4 Composite Features

In addition, EnTimeMent explores the use of composite measures, created by combining two or more individual measures or by enriching them. Composite measures are known to be able to capture richer information embedded in complex biological signals. For example, Figure 5 below, adapted from a recent development in EnTimeMent (Coste et al., 2020), shows how the (weighted) combinations of several kinematic variables (position; velocity (1st derivate of

position signal); acceleration (2nd derivate of position signal); and jerk (3rd derivate of position signal) according to an optimal level of temporal and spatial granularity (time interval between two successive silhouettes ΔT; pixels size) can offer a more complete picture than the raw data (here only position signals from a few motion capture markers) .

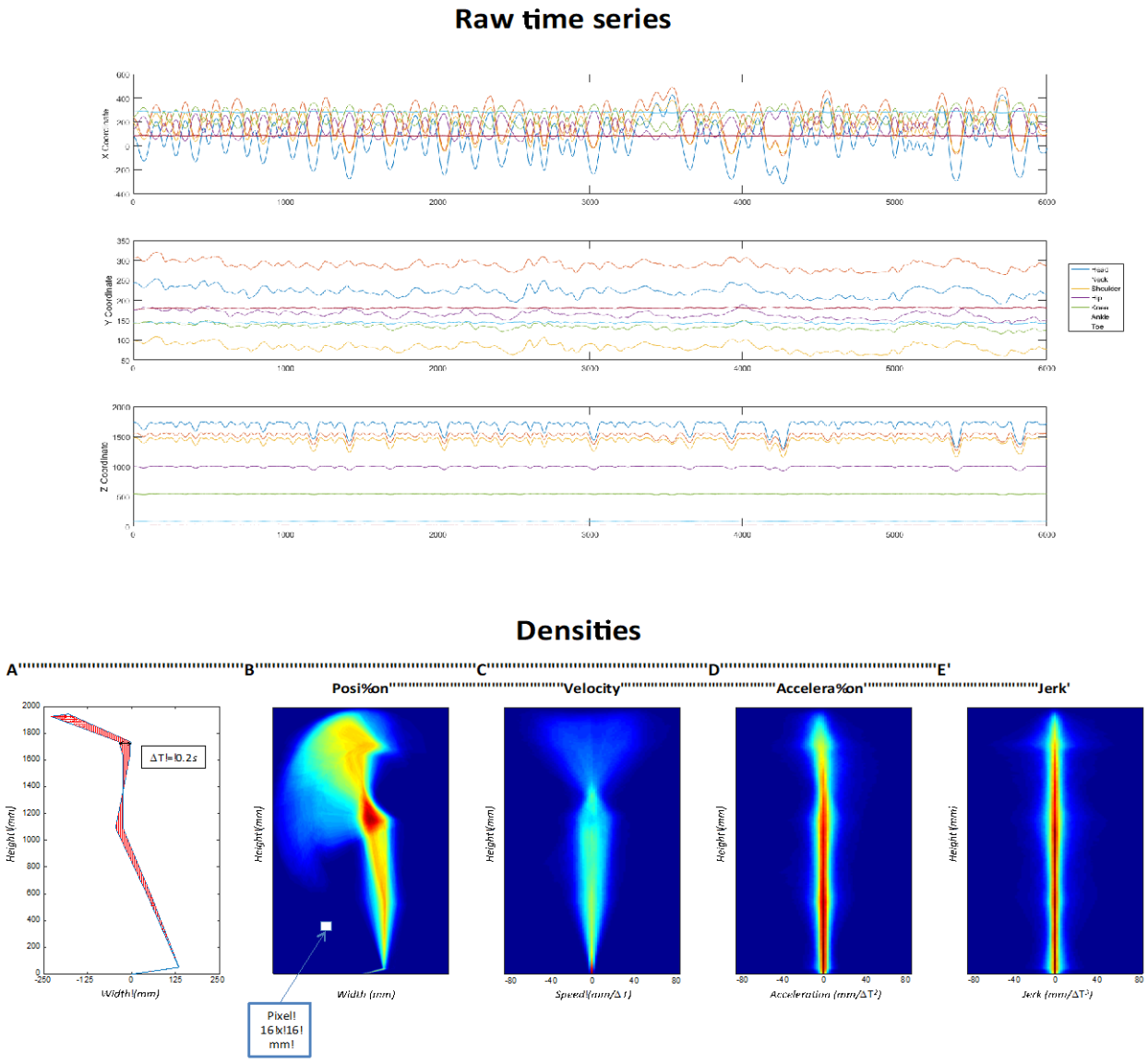

Figure 5 Top: Raw time series of seven 3D (x,y,z) motion capture markers placed on the head, neck, shoulder, hip, knee, ankle and toe. Bottom: A) Silhouette between two successive positions obtained by linking markers to form segments, with a temporal granularity of ΔT = 0.2 s and a spatial granularity of 16 mm2 pixel. From B) to E) Mean densities of position, velocity, acceleration and jerk, respectively. Colour gradient indicates the frequency of appearance of body segments within each pixel (occurrence), the red coloured pixels being the most visited places in space by the participant improvising postural motions, while those in blue were the least visited.

Coste, A., Bardy, B. G., Janaqi, S., Słowiński, P., Tsaneva-Atanasova, K., Goupil, J. L., & Marin, L. (2020). Decoding identity from motion: how motor similarities colour our perception of self and others. Psychological research, 1-11.

3.1.5 Model Selection (MS) and Error Estimation (EE)

MS and EE deal with the problem of tuning and assessing the performance of a learning algorithm. A possibility is the resampling techniques which rely on a simple idea: the original dataset D_n is resampled once or many (n_r) times, with or without replacement, to build three independent datasets called learning, validation and test sets, respectively L, V, T with r in $\{1, \dots, n_r\}$. Note that the intersection of all these three set is always empty.

Then, in order to select the best combination of the hyperparameters H in a set of possible ones $\mathfrak{y} = \{H1, H2, ..\}$ for the algorithm A_h or, in other words, to perform the MS phase, the procedure that minimize reconstruction error has to be applied. In this step the reconstruction error is computed considering only the Learning and Validation sets. Since the data in L is independent from the ones in the V, the idea is that H^* should be the set of the hyperparameters which allows to achieve a small error on a data set that is independent from the training set.

Further, to evaluate the performance of the optimal model, or in other words, to perform the EE phase, a new learning set is built in the following way: $L = L \cup V$.

Since the data in $L \cup V$ are independent from the ones in T, we have built an unbiased estimator of the true performance, measured with the metric M, of the final model.

3.1.6 Convolutional Autoencoder Backbone

A convolutional autoencoder backbone can be used in order to reduce the dimensionality of the features present in the dataset simply reconstructing the input data as best as possible. Moreover, as other model based on convolutional operations, we are able to extract features automatically from the data avoiding operations needed for traditional machine learning models such as those described in Section 3.1.3. Therefore, a convolutional autoencoder can be used at the pre-processing step in order to identify best timescales which properly fit data, treating the different time scales as hyperparameters for the machine learning model that will be most appropriate to use. The idea is that each feature has its own timescale which is different from the others. Therefore, if we apply different size of convolutional filters for each feature, we can identify the best timescale needed for the best feature representation. As the script below (Box 1) shows, we need to build different input layer for each input feature (in the script, *encoder1-decoder1* refer to the first input feature) and, each of this input feature can best be approximated through the convolutional filter size (in the script , *ks_feat1, ks_feat2, ks_feat3* refer to the convolutional filter size for each input feature).

This step is needed in order to better understand the insights of the problem analysed identifying features that allow a better representation of the data. Since this is not completely sufficient for the purposes of our studies, we need to concatenate the encoder backbone to a machine learning model able to handle classification or regression tasks. Therefore, at this point of the pipeline we can chose several machine learning models such as traditional machine learning models, neural networks, convolutional neural networks or models belonging to the recurrent neural network (RNN) family able to handle time information (RNN, LSTM, Clockwork RNN).

```
class m_autoencoder(nn.Module):
  def __init__(self, nf, ks_feat1, ks_feat2,ks_feat3):
      super(m_autoencoder, self).__init__()
      self.encoder1 = nn.Sequential(
          nn.Conv1d(1, nf, ks_feat1, stride=2, padding=1),
          nn.BatchNorm1d(nf), nn.LeakyReLU(),
          nn.Conv1d(nf, nf*2, ks_feat1, stride=2, padding=1),
          nn.BatchNorm1d(nf*2), nn.LeakyReLU(),
          nn.Conv1d(nf*2, nf*3, ks_feat1),
          nn.BatchNorm1d(nf*3), nn.LeakyReLU(),
      )

      self.decoder1 = nn.Sequential(
          nn.ConvTranspose1d(nf*3,nf*2,ks_feat1),
          nn.BatchNorm1d(nf*2), nn.LeakyReLU(),
          nn.ConvTranspose1d(nf*2, nf, ks_feat1, stride=2, padding=1, o
utput_padding=1),
          nn.BatchNorm1d(nf), nn.LeakyReLU(),
          nn.ConvTranspose1d(nf, 1, ks_feat1, stride=2, padding=1, outp
ut_padding=1),
          nn.LeakyReLU(),
      )
```

Box 1. A pytorch script where a convolutional autoencoder is shown.

3.2 Statistical and Machine Learning Methods and Architectures

This section summarises the various statistical and machine learning methods and architecture implemented for the projects including traditional ones as well as novel ones proposed within the EnTimeMent project.

3.2.1 Granger Causality

Granger causality, in its standard and linear formulation, is based on (linear) Autoregressive Models (AR). AR models belong to the family of the Linear Dynamical Systems, which has been

extensively used in modelling human motion (Bissacco, 2005; Del Vecchio, Murray, & Perona, 2003; Lu & Ferrier, 2004).

An AR model of a time series x is defined as:

$$(1) \quad x(t) = \sum_{j=1}^{l} a_j x(t-j) \; + \; \varepsilon_R(t)$$

where $x(t)$ is the value of the timeseries x at time t, l is the order of the model (i.e., the length of the history observed in the model), a_j ($j = 1, \dots, l$) are the weights for the history (the model parameters), and $\varepsilon_R(t)$ is the residual (prediction error).

There are two widely used criteria for selecting the optimal order of a linear predictor (i.e., the order that guarantees the best goodness of fit of the model): the Akaike's Information Criterion (AIC) (Akaike, 1974) and the Schwarz's Bayesian Information Criterion (BIC) (Schwartz, 1978). The parameters a_j can be computed by using Ordinary Least Squares.

Since Granger causality is based on AR models, the validity of the inferred causal relations depends on the validity of the AR models (more specifically of the unrestricted AR models, see next paragraph). To assess the validity of an AR model, different tests can be carried out ranging from tests of the non-correlation of the residuals to tests of the goodness-of-fit of the model (for example, the goodness-of-fit can be measured as the sum of squares of the residuals).

A time series X is said to "Granger cause" a time series Y, if the past values of X provide statistically significant information to predict the next value of Y (Granger, 1969). The prediction is computed using AR models. Two AR models are required: an unrestricted AR model where the history of all timeseries is assumed to contribute to the prediction of the current value of a time series; and a restricted AR model where the time series whose causality value (on the other timeseries) is computed is excluded from the history. Given two time series X and Y, the unrestricted model is defined as:

$$x(t) = \sum_{j=1}^{l} a_{U,j} x(t-j) + \sum_{j=1}^{l} b_{U,j} y(t-j) + \; \varepsilon_U(t)$$

(2)

$$y(t) = \sum_{j=1}^{l} c_{U,j} x(t-j) + \sum_{j=1}^{l} d_{U,j} y(t-j) + \; \eta_U(t)$$

While the restricted model is defined as:

$$x(t) = \sum_{j=1}^{l} a_{R,j} x(t-j) \; + \; \varepsilon_R(t)$$

(3)

$$y(t) = \sum_{j=1}^{l} d_{R,j} y(t-j) \; + \; \eta_R(t)$$

Then the magnitude of the causality from X to Y and from Y to X can be measured respectively as:

$$(4) \qquad \mathcal{F}_{x\to y} = \ln\frac{\mathrm{H}_R}{\mathrm{H}_U}, \qquad \mathcal{F}_{y\to x} = \ln\frac{\mathrm{E}_R}{\mathrm{E}_U}$$

where E and H are the model error variances:

$$(5) \qquad \mathrm{E}_R = var\big(\varepsilon_R(t)\big),\ \ \mathrm{E}_U = var\big(\varepsilon_U(t)\big),$$
$$\mathrm{H}_R = var\big(\eta_R(t)\big),\ \ \mathrm{H}_U = var\big(\eta_U(t)\big)$$

Once the Granger causality values have been computed, we need to test their statistical significance, i.e., we need to infer the significant causal relations. A significance test can be done by carrying out an F-test of the null hypothesis that the model parameters referring to the time series of which we compute the "causal strength" (on the other time series) are all zero (e.g., parameters $b_{U,j}$ in model (2) to test the significance of $\mathcal{F}_{y\to x}$). When more than two time series are analysed some corrections (here, the Bonferroni correction) are applied to the F-test.

When the interaction of more than two time series is addressed, repeated pair-wise Granger causality computations can lead to misleading results. To avoid that, a simple extension of Granger causality, sometimes referred to as Conditional Granger causality, has been proposed by Ding et al., 2006 (Ding, Chen, & Bressler, 2006). Suppose we have three time series X, Y and Z, then the Conditional Granger causality from Y to X given Z is defined as the log ratio of the error variance of the restricted model where only Y is excluded from the history (when modelling X) and the variance of the unrestricted model, where the history of all time series X, Y and Z is included.

The Granger causality analysis, including AR model validation and statistical tests of causal interactions, will be carried out by using the "Granger Causality Connectivity Analysis" MATLAB toolbox.

Akaike, H. (1974). A new look at the statistical model identification. In IEEE Trans. Autom. Control (pp. 716–723).

Ancona, N., Marinazzo, D., & Stramaglia, S. (2004). Radial basis function approach to nonlinear Granger causality of time series. Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics, 70(5), 7. http://doi.org/10.1103/PhysRevE.70.056221

Bissacco, A. (2005). Modeling and learning contact dynamics in human motion. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05 (pp. 421–428). http://doi.org/10.1109/CVPR.2005.225

Chen, Y., Rangarajan, G., Feng, J., & Ding, M. (2004). Analyzing multiple nonlinear time series with extended Granger causality. Physics Letters, Section A: General, Atomic and Solid State Physics, 324(1), 26–35. http://doi.org/10.1016/j.physleta.2004.02.032

Del Vecchio, D., Murray, R. M., & Perona, P. (2003). Segmentation of human motion into dynamics based primitives with application to drawing tasks. Automatica, 39, 1–36. http://doi.org/10.1109/ACC.2003.1239860

Ding, M., Chen, Y., & Bressler, S. L. (2006). Handbook of Time Series Analysis. (S. Schelter, N. Winterhalder, & J. Timmer, Eds.)Wiley (Wiley). Wienheim.

Freiwald, W. A., Valdes, P., Bosch, J., Biscay, R., Jimenez, J. C., Rodriguez, L. M., … Singer, W. (1999). Testing non-linearity and directedness of interactions between neural groups in the macaque inferotemporal cortex. Journal of Neuroscience Methods, 94(1), 105–119. http://doi.org/10.1016/S0165-0270(99)00129-6

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. Econometrica, 37, 424–438.

Lu, C., & Ferrier, N. J. (2004). Repetitive motion analysis: segmentation and event classification. In IEEE Trans. Pattern Anal. Machine Intell. (pp. 258–263).

Schwartz, G. (1978). Estimating the Dimension of a Model. Ann. Stat., 5, 461–464.

3.2.2 Traditional Machine Learning Methods

Traditional machine learning methods can be a useful tool for assessing the goal of interest. In particular, due to their easy implementation and their power strength and stability, they can be used to provide a benchmark in the target prediction. The nature of these methodologies does not directly assess time series information and therefore, multi-time scales. In order to use these methods, a pipeline needs to be designed. For example, time information can be assessed through fixed-width sliding windows. The size of these windows can be changed to study how it influences prediction and what the most suitable settings are. Once the size of the windows is fixed, once can proceed in two ways:

1. Make the prediction using fixed-sliding windows computed on raw data, or
2. Make the prediction using fixed-sliding windows computed on features engineered as shown in Section 3.1.3.

Features engineered used as training set provide a better prediction due to the fact that they can capture the underlying pattern. The weakness of this approach, however, lies in the fact that features may not generalise well to unseen instances and they also need to be carefully developed to truly define the pattern within the data. Due this weakness, deep learning models – in particular recurrent neural network (RNN) family of models – which learn a feature representation of the raw data automatically are being increasingly used. However, the advantages of these methodologies are numerous and we can summarize them in the following points:

1. reliable results;
2. a faster execution speed;
3. minor complexity;
4. an easy understanding of the underlying problem;
5. fewer hyperparameters to be tuned.

For these reasons, their use can provide a quick and reliable benchmark for deeper and more accurate future analyzes through Deep Learning models.

One algorithm worth mentioning is Random Forest (RF). RF is a state-of-the-art powerful learning algorithm, first developed in (Breiman 2001). RF is based on multiple Decision Trees (DTs) (Rokach and Maimon 2008). A binary DT is a flowchart-like structures in which each

internal node represents a test of a feature, each branch represents the outcome of the test, and each leaf node represents a class label. A path from the root to a leaf represents a classification rule. Each node of the DT, starting from the root node, is built by choosing the attribute and the cut that most effectively splits the set of samples into two subsets based on the information gain. Given the definition of DT, it is now possible to briefly describe the learning phase of each of the n_t trees composing the RF. From D_n, n_b samples are taken with replacement, and D' is built. A tree is constructed with D', but the best split is chosen among a subset of n_v predictors over the possible n_f predictors randomly chosen at each node. The tree is grown until its depth reaches the maximum value of n_d or until all the samples in D' are correctly classified. During the classification phase of a previously unseen X samples, each tree classifies X into a class Y; the final classification is the mode of all the answers of each tree in the RF. Note that n_b, n_v, n_d and n_t are the hyperparameters of the RF.

Breiman, L. Random forests. Machine Learning 45(1),5-32 (2001)

Rokach,L., Maimon, O.Z.: Data Mining with Decision Trees: Theory and Applications, vol. 69 Worlds Scientific (2008).

3.2.3 Using Standard Machine Learning Algorithms for Music Performance Data

Music performance data will be analysed using EnTimeMent libraries as discussed in Section 3.1.1, and using standard machine learning algorithms as appropriate to explore problems such as:

- Identification of individuals
- Identification of repertory items
- Identification of mutual influence and coordination between individuals
- Prediction of musical events (e.g. cadences or section boundaries)
- Prediction of audio features from movement and vice versa

3.2.4 Deep Learning

In Section 3.2.2 we discussed traditional machine learning methodologies and a possible pipeline that can be followed in order to obtain a stable benchmark. As discussed, those techniques have a limitation due to the fact that extracted features may not be sufficient in capturing patterns in the data. An evolution of traditional machine learning methods has led to Deep Learning algorithms which use many more parameters and can obtain better results based on features which are automatically extracted from raw data; these learnt features can better define the relationships in the machine learning problem.

These architectures can be used to:

1. Extract features automatically, and the features can then be used with any other machine learning method that does the classification or other machine learning task;

2. Both extract features automatically as well as for machine learning prediction based on these features.

In particular, if our target is managing timescale information, we can build an architecture where we use convolution filters with different dimensions. The size of this convolutional filter can be seen as a hyperparameter of our model chosen so that we minimize the input reconstruction error. In this step of the pipeline, a convolutional autoencoder (see Section 3.1.6) is often used. When an optimal value convolutional filter size is found from each feature, we can replace the decoder part of the convolutional autoencoder with a properly machine learning algorithm (either deep learning or a traditional algorithm).

Although deep learning methodologies are very powerful tools, they require a lot of parameters and, therefore, hyperparameters for a correct tuning of the model is important. Moreover, they need a lot of data for a better generalisation, suffering when the cardinality of the dataset is very low. A last consideration is that they can be very unstable, and they can provide poor results if the correct choice of parameters is not found.

3.2.5 RNN Family of Algorithms

In the deep learning methods described in Section 3.2.4, we mentioned RNN algorithms which were designed to address temporal information correlating past/present and future knowledge. RNN family can handle timeseries either by directly analysing raw data or based on features extracted automatically using a convolutional backbone.

Multi-time scales can be assessed though several points:

1. Since simple RNNs - such as RNN, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Unit (GRU) (Chung et al. 2014). – cannot on their own address multiple timescales, a practical approach can be a combination of these algorithms with a convolutional autoencoder backbone described in Section 3.2.4. Once features are extracted using different convolutional filters, we obtain different timescales for each feature analysed: each feature is better adapted to its own temporal filter size. A simple RNN that then further processes the temporal information for each feature can then be stacked at the end of the convolutional autoencoder backbone.
2. Since simple RNNs cannot directly handle multiple timescales, some of their extensions can be considered. A possible solution is the architecture called Clockwork RNN (CW-RNN) (Koutnik et al. 2014) (Figure 6). The hidden states are factorized across g modules with increasing clock rates $\{T_1, \dots, T_g\}$. The state of module i gets updated every t_i timesteps. At each update, module i receives the contribution from equal or slower modules j such that $T_i \leq T_j$.
3. A combination of the two methodologies above could be employed, i.e. using a RNN algorithm designed to handle multiple timescales with a convolutional autoencoder backbone that used filters of different sizes for each feature.

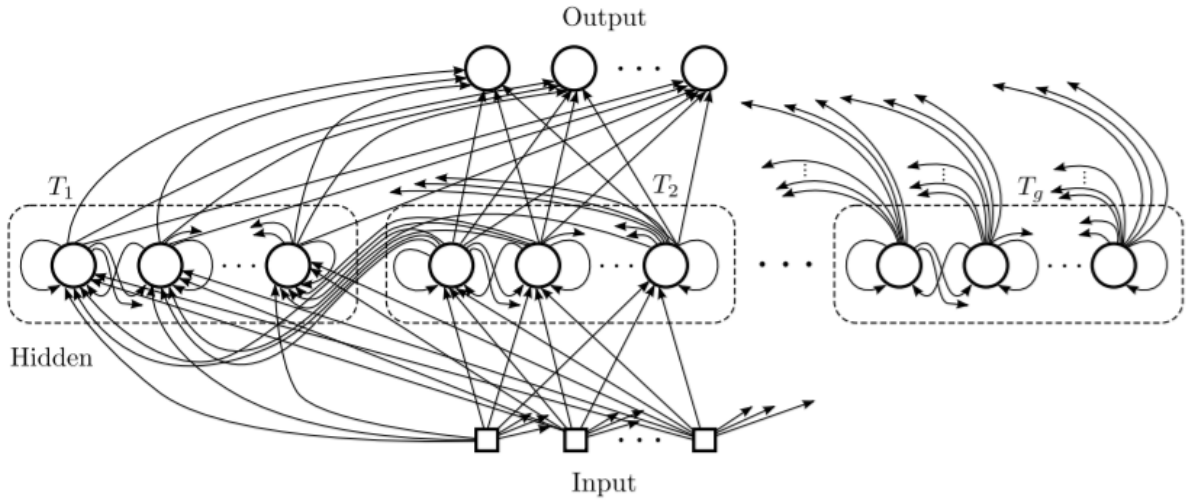

Figure 6 From (Koutnik et al. 2014). CW-RNN architecture is similar to a simple RNN with an input, output and hidden layer. The hidden layer is partitioned into g modules each with its own clock rate. Within each module the neurons are fully interconnected. Neurons in faster module i are connected to neurons in a slower module j only if a clock period $T_i < T_j$.

Chung, J. et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv preprint arXiv:1412.3555 (2014).

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. Neural Comput. 1997;9(8):1735–80.

Koutnik J. et al. "A Clockwork RNN". In: Proceedings of the 31st International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014. 2014, pp. 1863-1871. URL: http://jmlr.org/proceedings/papers/v32/koutnik14.html.

3.2.6 Relating Neural Activation Patterns to Computational Models of Human Movement

The deep neural networks described in the previous sections will be used to generate stimulus features by extracting unit activations from layers of interest. Next these features will serve as input for models that project the features onto the fMRI recordings. Various techniques will be used such as representational similarity analysis, encoding models and population receptive field modelling. We will develop in-house custom scripts to implement these models.

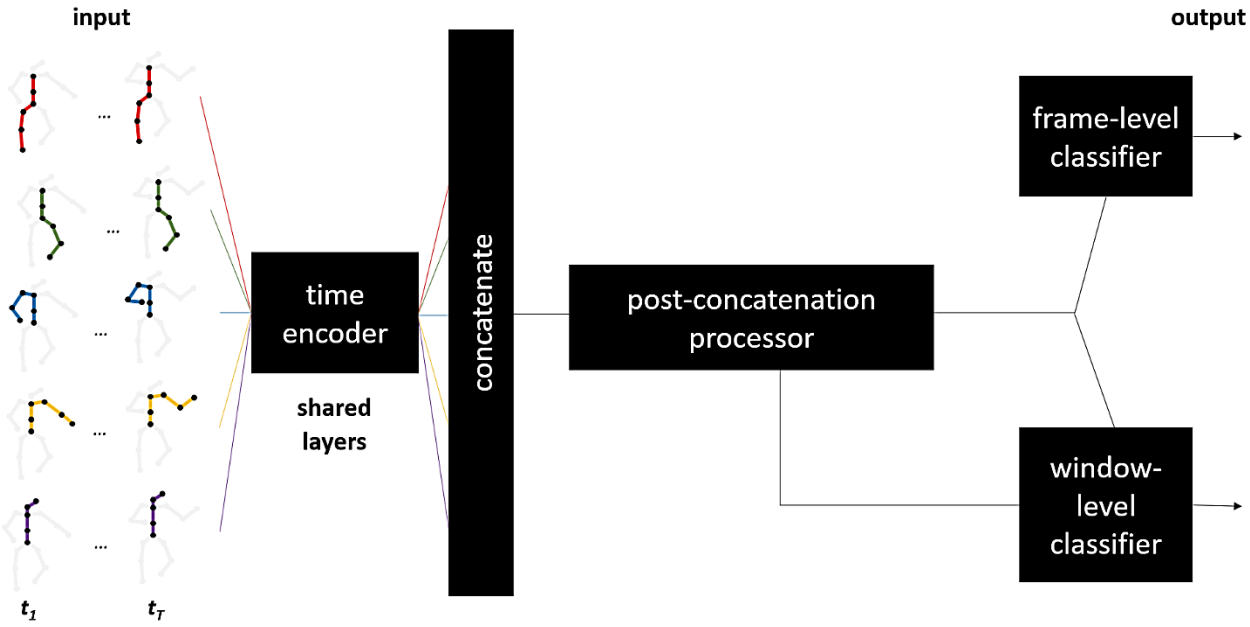

Figure 7 Multi-Time Neural Network Architecture

3.2.7 Multi-Time Neural Network (MultiTimeNN)

We propose a neural network architecture (named Multi-Time Neural Network, MultiTimeNN) for learning movement behaviour at multiple timescales of interpretation and based on anatomically distributed encoding of temporal information. As discussed in Section 3.2.4, neural network methods enable features to be automatically learnt from raw data eliminating the need for handcrafting appropriate features.

The distributed encoding idea draws from documented findings (e.g. in Kerr et al. 1994) that a single movement, for example standing up from a chair, could be specified in terms of multiple events that involve separate anatomical segments and occur at different time scales (i.e. different durations, and so different onsets and/or offsets). For instance, for the sit-to-stand example, Kerr et al. (1994) defined 4 events: a forward trunk flexion which kicks off the movement and an adjustment which starts just before the flexion ends completes the movement. In between these two events, there are an angular knee displacement of the knee and a vertical trunk displacement both of which together start slightly earlier than the adjustment event and although the knee displacement is concluded just before the end of the movement, the trunk displacement is completed earlier.

The approach of learning movement behaviour at multiple timescales is similarly based on the event segmentation literature (e.g. Zack and Swallow 2007) that suggests that observers parse an observed activity as multiple events and that the timescales of the events can vary within observers depending on the observer's attention and/or goal.

In other words, these bodies of findings highlight that there may be multiple timescales to both the expression of a movement and its interpretation. Our MultiTimeNN accounts for this by consisting of: 1) a time encoder that processes different anatomical groups of segments separately, similar to the discussion in Sections 3.1.6 and 3.2.5; to limit the number of trainable parameters, the time encoder is shared by these anatomical segment groups; 2) classifiers that learn the same behaviour or affective state at different timescales; to leverage the relationship across timescales, a multitask learning paradigm is used.

Figure 7 provides an overview of the MTNN architecture for classification at two timescales (frame level and window level) and with body movement input grouped into 5 segments (spine and right lower limb, spine and left lower limb, spine and right upper limb, spine and left upper limb, spine and head and neck).

Kerr, K. M., White, J., Barr, D., & Mollan, R. (1994). Standardization and definitions of the sit-stand-sit movement cycle. Gait & Posture 2(3), 182-190.

Zacks, J. M., & Swallow, K. M. (2007). Event segmentation. Current Directions in Psychological Science 16(2), 80–84.

3.2.8 Body Attention Net (BAnet) - now published in ACII 2019 (Wang et al. 2019)

We further propose a novel neural network architecture that weights different time points for the separate anatomical segment groups after applying the shared time encoder on these segment groups (see Section 3.2.7). Unlike the MTNN, a single timescale of behaviour of affective states is learnt. The BAnet further weights the different segment groups after the weighted time encoder outputs are concatenated. This additional weighting within a neural network is based on the neural attention paradigm well-established in machine learning.

As can be seen in Figure 8, the time encoder of the BAnet is implemented as a LSTM network. The temporal attention network is also shared by the different segments to further control the number of trainable parameters of the BAnet. For this same reason, the weight computation in the temporal attention network is based on a 1x1 convolution (conv) kernel which is shared by each time point for the current anatomical segment. Using a *softmax* function, the computed weights are normalised to a [0, 1] range such that the weights add up to one. The normalised weights are then applied to the original output of the time encoder. The sums of the weighted, time-encoded information for each anatomical segmented are concatenated before they are processed by the body attention network. The body attention network is implemented using a fully connected (fc) network (i.e. a multilayer perceptron) with *tanh* activation. Similar to the temporal attention, the computed weights are normalised

using a softmax function and they are applied to the concatenated sums of the output of the temporal attention network.

The BAnet concludes with another fully connected network that is used to classify the output of the body attention network. Unlike the MTNN, classification is done at a single timescale for the BAnet.

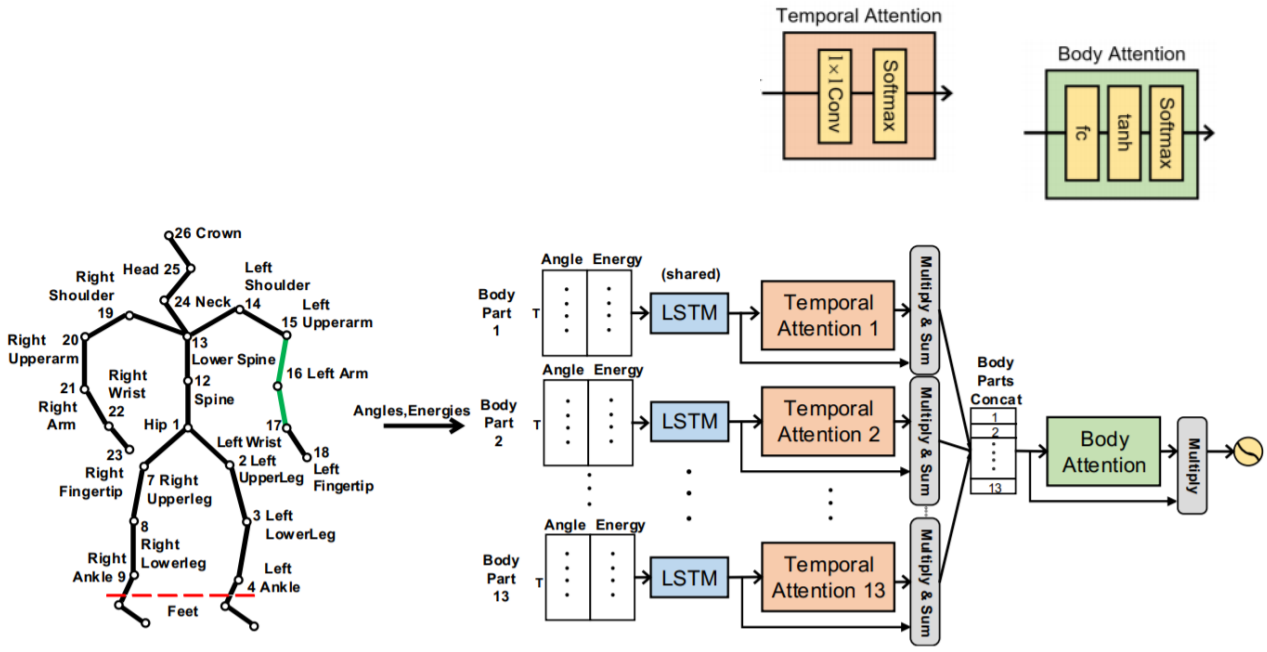

Figure 8 Body Attention Net Architecture

Gers, F., A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with LSTM. Neural Comput. 2000;12(10):2451–71.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. Neural Comput. 1997;9(8):1735–80.

Wang, C., Peng, M., Olugbade, T.A., Lane, N.D., Williams, A.C.D.C. and Bianchi-Berthouze, N., 2019, September. Learning temporal and bodily attention in protective movement behavior detection. In 2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW) (pp. 324-330). IEEE.

3.2.9 Multi-spatial-temporal Graph Convolutional Networks (MST-GCN)

Another proposed solution, that can also be extended to groups of individuals, is based on graph convolutional networks (GCN). Just like regular convolutional neural networks, GCNs use operations performed on neighbours of nodes, but instead of working on arrays they work on graphs, such as skeletons with nodes given by body markers. The proposed model, called a multi-spatial temporal graph convolutional network (MST-GCN), is trained in a number of

stages in which information is gradually spread spatially or temporally in an interleaved manner. As can be seen to the right in Figure 9, each stage consists of a number of residual layers and for each temporal layer, dilated convolutions are used, which means that the temporal receptive field is doubled for each new layer. Spatially, the processing is done using a GCN that has a neighbourhood system based on the connectivity of body markers with respect to the centre of the body. The outputs from each stage are then aggregated into a long feature vector that characterizes the movement of the individual over different time horizons. Fine-scale movement patterns are represented by features from the earlier stages and more coarse-grain patterns by those in later stages. The combined feature vector can later be used by a classifier to predict different qualities of the movement, without knowing the exact time frame over which significant patterns occur.

A similar network structure can be used to aggregate information from multiple individuals, that are interacting within a group, by adding another GCN on top of the MST-GCNs representing each individual. This group GCN also processes information using a neighbourhood system, but now the system is based on the relative positioning of the individuals of the group. A similar group GCN has been used in a study that looked at conversational groups (Yang et al., 2020), with the goal of predicting whether an arriving newcomer would be accommodated or ignored by a group of individuals already engaged in a heated discussion.

A conclusion from that study was that the proposed network structure benefits from the fact that the structure of the data is preserved over the processing with data represented in terms of features on each body marker. This differs from attentional networks, where information is collapsed either temporally or spatially with most weight put on the most discriminative patterns. With GCNs, data can instead be processed over a number of stages, gradually increasing the receptive field both spatially and temporally, which is beneficial if there are no particular discriminative patterns on a particular time scale, but the overall movement is important. Thus, the model of choice depends on the particular movement quality that the model is intended to capture.

Yang, F., Yin, W., Inamura, T., Björkman, M. and Peters, C. Group Behavior Recognition Using Attention- and Graph-Based Neural Networks, accepted to ECAI 2020.

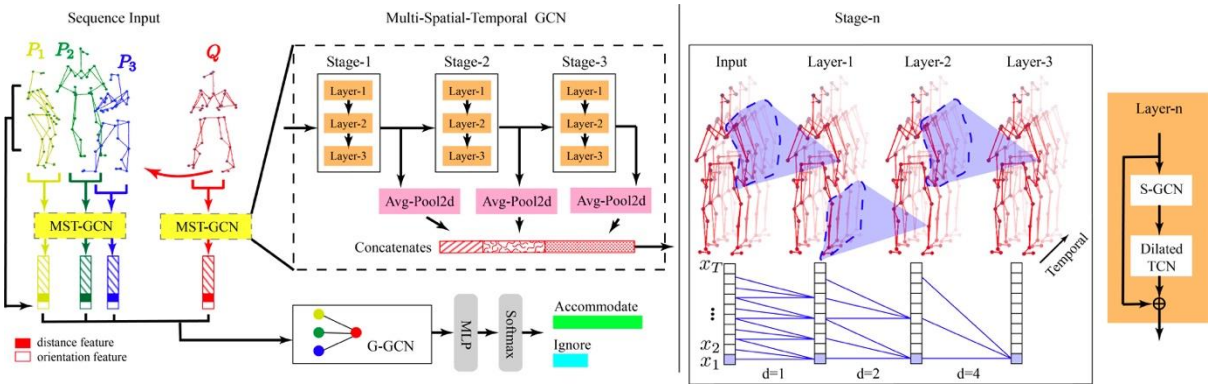

Figure 9 Multi-spatial-temporal Graph Convolutional Networks for representation of individuals with a separate Graph Convolutional Network for representation of a group of multiple individuals.

3.3 Sonification Libraries

3.3.1 Interactive Sonification Libraries for Movement Qualities

Expressive movement sonification is the process of translating a movement into a sound that "evokes" some of the movement's expressive characteristics. It can be applied in the design of multimodal interfaces enabling users to exploit non-verbal full-body movement expressivity in communication and social interaction. In EnTimeMent, sonification models are being developed, inspired by several sources: humanistic and artistic theories (including morphological vocabularies of sonic objects, the analysis of literature in cinema soundtracks), and research in cross-modality. During the first EnTimeMent public event (the "A Tempo! First Project Workshop", https://entimement.dibris.unige.it/events) a set of examples of interactive sonification techniques and paradigms followed in EnTimeMent were presented, with particular reference to the three EnTimeMent project Scenarios. Examples of interactive sonification approaches followed in the project include the following: (i) embodied sonic training: interactive sonification of movement qualities to augment the awareness and proprioception during a motor-cognitive rehabilitation exercise, to enhance and overcome motor problems, (ii) interactive sonification as a means to alert a person during everyday life in case of dangerous movements or to predict injuries. This involves the exploitation of the EnTimeMent automated analysis and prediction of full-body movement at multiple time scales, and a mapping of these data to the auditory domain. Current research and software libraries in this direction start from the scenarios presented at the A Tempo! First Project Workshop.

3.3.2 Software Application for The Sonification and Visualisation of Neural Network Attention Scores - now published in NIME 2020 (Gold et al. 2020)

The BAnet neural network (Wang et al. 2019) provides weights that indicate the joint groups to which it has paid most attention in reaching its conclusion of protective or non-protective behaviour. In order to expose this information in a multi-modal fashion for exploration by a

physiotherapist and patient, an application is under development to provide multi-temporal sonification of the movement attention over time. Multi-temporality is supported through the use of musically coherent works divided into multiple channels. The gain of each channel is controlled by the relative score for attention, thus permitting musical time to continue separately from movement time. Movement can be played forward or backward or freely manipulated, and scaled in time to permit exploration. The sonification is complemented by a visual representation of a figure decorated with Bezier curves whose line weight is proportional to the attention score of the respective joint group. A proof of concept has shown that changes in the attention score can be seen and heard appropriately and ongoing work is beginning to create the capability for sequential and parallel dyadic sonification and visualization, additionally incorporating 3D animation.

C. Wang, M. Peng, T. A. Olugbade, N. D. Lane, A. C. de C. Williams, and N. Bianchi-Berthouze. Learning Temporal and Bodily Attention in Protective Movement Behavior Detection. In 2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW), pp. 324-330. IEEE, 2019.

Gold, N.E., Wang, C., Olugbade, T., Berthouze, N. and Williams, A., 2020, July. P (l) aying Attention: Multi-Modal, Multi-Temporal Music Control. In Proceedings-International Conference on New Interfaces for Musical Expression. NIME.