DANCE

D2.1 – First version of the DANCE software libraries

Version	Edited by	Changes
1.0	UNIGE	first version

TABLE OF CONTENTS

1. Introduction

The *DANCE software libraries for the analysis of expressivity, emotion, and social signals* are a collection of software modules, components and applications that are integrated in the EyesWeb XMI research platform.

The EyesWeb XMI platform is a modular system that allows both expert (e.g., researchers in computer engineering) and non-expert users (e.g., artists) to create multimodal installations in a visual way [13]. The platform provides software modules, called **blocks**, that can be assembled intuitively (i.e., by operating only with mouse) to create programs, called **patches**, that exploit system's resources such as multimodal files, webcams, sound cards, multiple displays and so on. The platform, **blocks** (i.e., software modules) and **patches** (i.e., programs) can be freely downloaded from the website:

http://dance.dibris.unige.it/index.php/dance-platform

This document presents an overview of the **blocks** included in the DANCE software libraries, developed in the framework of the project. Then we provide a list of **patches** for the analysis of *movement features* (energy, slowness, smoothness, weight, fluidity, suddenness, impulsivity), and *analysis primitives* (synchronization).

2. Blocks

In this section we provide some details on some of the main blocks for analysis of features and analysis primitives that are included in this first version of the libraries.

2.1 Movement features

In the DANCE project we aim to innovate the state of art on the automated analysis of the expressive movement. We consider movement as a communication channel allowing humans to express and perceive implicit high-level messages, such as emotional states, social bonds, and so on.

That is, we are not interested in physical space occupation or movement direction per se, or in "functional" physical movements: our interest is on the implications at the expressive level. For example: a hand movement direction to the left or to the right may be irrelevant, instead the level of fluidity or impulsiveness of such movement might be relevant. Example: let us consider the movement "Knocking at a door" [12]. We do not want to analyze the functional action of "knocking at a door", but the intention that lies behind it (e.g., the lover that knocks at the door of his beloved). To study it, we focus on the sets of non-verbal expressive features that are described in detail in Deliverable 5.1. In the following sections we illustrate a first set of models and algorithms for expressive movement analysis that are embedded in these DANCE software platform.

2.2.1 Feature: Fluidity

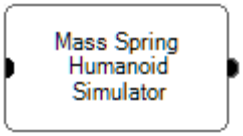
Definition: A Fluid movement can be performed by a part of the body or by the whole body and is characterized by the following properties:

Property 1 (P1): the movement of each involved joint of the (part of) the body is smooth, following the standard definitions in the literature of biomechanics (e.g. [19], [20], [21]).

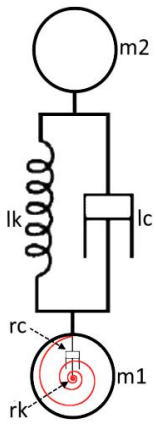
Property 2 (P2): the energy of movement (energy of muscles) is free to propagate along the kinematic chains of (parts of) then body (e.g., from head to trunk, from shoulders to arms) according to a coordinated wave-like propagation. That is, there is an efficient propagation of movement along the kinematic chains, with a minimization of dissipation of energy.

To implement P2, we propose a Mass-Spring Model.

2.2.1.1 Mass Spring Humanoid Simulator



This block implements a humanoid simulator in terms of a Mass-Spring-Damper physical model. Mass-spring models have already been exploited to simulate human gait, run, and jump [1, 2, 3, 4]. The model we propose represents the human body as a set of interconnected masses, where each mass represents a body joint. The model can be used to simulate, for example, the evolution of the movement of a dancer, and to compare it with the movement of a real dancer. We tuned parameters of the model in order to generate movements that are perceived as Fluid by human observers. In this way, we can use this model as a reference for Fluidity: i.e., Fluidity is



A simple model, two masses (m1 and m2) are linked by a spring (lk), and the resulting body segment is influenced by a rotational spring (rk) that controls its rotation and movement.

computed in terms of distance of the real movement of a dancer with the simulated movement of the model.
In particular, we compute the mean jerk values of the shoulders, elbows and hands from both the real movement data and the simulated movement. By measuring the distance of the overall jerk of the recorded data and the simulated one we can identify a quantity (*JI*) that roughly estimates the Fluidity of movement of a given segment.
More details about the model are available in [8].

2.2.2 Feature: Impulsivity

Definition: an impulsive movement can be performed by a part of the body or by the whole body and is characterized by the following properties:

Property 1 (P1): it is sudden, that is, it presents a high variation of speed (either from low to high or from high to low).

Property 1 (P2): it is executed with no preparation.

To compute P1 we apply a Savitzky-Golay low-pass filter [18] to compute the body (or part of the body) speed. Then, we perform an alpha-stable fit on the resulting speed to determine if it exhibits high variations.

2.2.2.1 Filtering and derivating (Savitzky-Golay filter)

The Savitzky-Golay filter is a low-pass filter able to compute not only the filtered signal but also the signal's filtered derivatives. For example, providing the sampled position data of the user's hand to the filter we can compute the hand's filtered position, speed and acceleration. The filtering and derivation process is performed by computing a weighted sum, that is, it can be performed in a very efficient way.

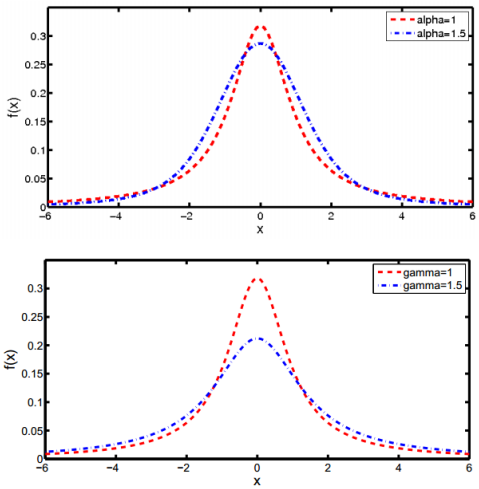
The Savitzky-Golay filtering block of the DANCE platform matches the operation of the Matlab function *sgolay(k,f)*. The filter order *k* must be less than the frame size *f*, that must be odd. If *k = f-1*, the filter does not produce any smoothing.

2.2.2.2 Alpha-stable fit

Alpha-stable distributions have been introduced by [5] and applied in several works on signal processing. They have been exploited as an alternative and more efficient approach than Gaussian distributions to model impulsive noise in [6] or to define a model of graphical textures exhibiting impulsive features that cannot be defined via Gaussian distributions [7].

Briefly speaking, an alpha-stable distribution can be modelled by a probability density function (*pdf*) characterized by four parameters (α, β, γ, δ):

- α ∈ (0; 2] is the characteristic exponent that defines whether the distribution includes impulses;
- β ∈ [1; 1] is the determines the skewness of the *pdf*;
- γ > 0 is the dispersion parameter and corresponds to variance in Gaussian distributions;
- δ ∈ (1; 1) is the shift from the origin of center of the *pdf* and corresponds to the mean value in Gaussian distributions.



In the top graph, two functions are displayed, the first obtained by setting α =1 and the second one by setting α =1.5, and leaving the other parameters unchanged. A lower value of α corresponds to thicker pdf tails. The two functions showed in the bottom graph exhibit γ values of 1 and 1.5.

This block applies the *stblfit* function, that is, a C++ porting of the stable fit Matlab algorithm available here: http://www.mathworks.com/matlabcentral/fileexchange/37514-stbl--alpha-stable-distributions-for-matlab

The stable fit block is exploited in the DANCE library to detect sudden movements. Taking as input the 3D right hand position of the user, we consider a time window containing 3D position of the right hand and we compute the absolute velocity of the hand by differentiating the hand position and computing the module of the vectors resulting form the differentiation process (using a Savitzky-Golay filter). Then, we apply *stblfit* function. The α parameter, that varies in (0,2], is scaled and multiplied by γ. This process implies 2 consequences:

(i) when α tends to zero, the scaled value of α tends to one and vice-versa;
(ii) movements exhibiting *low* (resp., *high*) velocity will correspond to *low* (resp., *high*) values of γ;

That is, the result of the product of α by γ will be high for sudden movements α) with large velocity variability (γ high).

2.2 Analysis primitives

Analysis primitives are unary, binary, or *n*-ary operators that summarize with one or more values the temporal development of low-level features in an analysis time unit (a movement unit or a time window). The simplest unary analysis primitives are statistical moments (e.g., average, standard deviation, skewness, and kurtosis). Further examples of unary operators, that are more complex, include shape (e.g., slope, peaks, valleys [14]), entropy [15], recurrence [16], and various time-frequency transforms). Models for predictions (e.g., HMM) can also be applied as in [17]. Binary and n-ary operators can be applied e.g., for measuring relationship between low-level features computed on the movement of different body parts. For example, synchronization can be used to assess coordination between hands. Causality can provide information on whether the movement of a joint leads or follows the movement of another joint

2.2.1 Primitive: Synchronization

Synchronization is an important concept in human-human communication that has been widely addressed by the HCI research community and in movement studies [24]. Research addressing synchronization in human-human or human-machine interaction includes [16], an interactive system for active fruition of music based on inter-personal synchronization, and [23], a rehabilitation system that demonstrates effectiveness in stabilizing the walking of patients affected by Parkinson's disease and hemiplegia.

Synchronization deals with user's movement qualities both at the *intra-* and at the *inter-personal* level: that is, this analysis primitive can be applied to different movement features. In the case of intra-personal synchronization, it is used to determine whether the user's joints movement is coordinated [22]. In the case of inter-personal synchronization, it is used to measure the level of, e.g., entrainment in a group of users [15].

In the DANCE platform we exploit *Event Synchronization*, a particular type of synchronization presented in [10].

Definition: the level of synchronization of two timeseries t_{vw} and t_{ve}, we compute the level of synchronization Q^τ as:

$$Q^\tau = \frac{c^\tau(t_{v_w}|t_{v_e}) + c^\tau(t_{v_e}|t_{v_w})}{\sqrt{m_{v_w} m_{v_e}}}$$

the quantity of synchronization Q^τ depends on the normalized sum of coefficients c^τ, expressing how much the events contained in the timeseries t_{vw} and t_{ve} are synchronized (that is, their distance in time is less than the threshold τ):

$$c^\tau(ts_{v_w}|ts_{v_e}) = \sum_{j=1}^{m_{v_w}} \sum_{i=1}^{m_{v_e}} K_{ji}^\tau$$

$$c^\tau(ts_{v_e}|ts_{v_w}) = \sum_{i=1}^{m_{v_e}} \sum_{j=1}^{m_{v_w}} K_{ij}^\tau$$

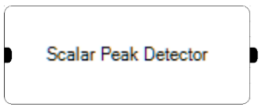
the contributes of each event to the coefficients c^τ are defined as:

$$K_{ij}^{\tau} = \begin{cases} 1 & \text{if } 0 < {t_i}^{v_w} - {t_j}^{v_e} < \tau \\ 1/2 & \text{if } {t_i}^{v_w} = {t_j}^{v_e} \\ 0 & \text{otherwise} \end{cases}$$

$$K_{ji}^{\tau} = \begin{cases} 1 & \text{if } 0 < {t_j}^{v_e} - {t_i}^{v_w} < \tau \\ 1/2 & \text{if } {t_j}^{v_e} = {t_i}^{v_w} \\ 0 & \text{otherwise} \end{cases}$$

In this first version of the libraries we developed three blocks to measure intra-personal synchronization by applying Event Synchronization [10]. The first one, given an input signals containing, for example, the speed of movement of the user's hand, finds the position of peaks (local maxima) of the signal. Then, a timeseries is generated, containing zeros everywhere except on the samples corresponding to the signal's peaks. The second one computes the Event Synchronization defined by the above equations and the third one implements the SPIKE synchronization [11].

2.2.1.1 Peak Detector



Each time the block receives as input a numerical value it provides an output value that can be:

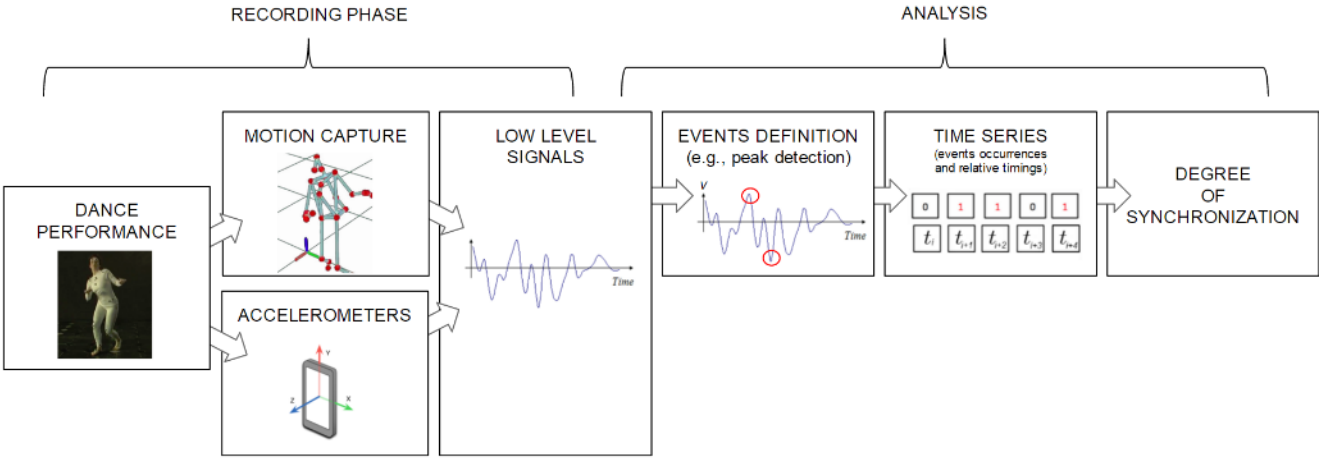
- 0: if no peak has been detected in the input signal;
- 1: if a peak has been detected;

The block has an internal buffer, implemented as a FIFO shifting queue, on which the minimum and maximum values are computed. If the input signal is decreasing and the difference between these two values is greater than a threshold then a peak is detected and a value of 1 is provided as output.

2.2.1.2 Event Synchronization

The first synchronization technique we implemented is the Event Synchronization (ES) algorithm [9]. It is based on time delay patterns between a pair of time-series containing event occurrences.

Example: this block can receive two timeseries as inputs, corresponding to the peaks (i.e., local maxima) of the velocity of user's hands, captured by accelerometers or extracted from motion captured data. The following figure illustrates the process of Simple Event Synchronization extraction:



The dancer's movement is captured by either a set of markers of a motion capture system or a set of accelerometers. Signals representing the user's limbs velocities are computed. By applying the peak detection algorithm described in Section 2.2.1.1 we obtain one timeseries per limb, containing zeros when no peak is detected and ones when a peak is detected. We compute the degree of synchronization between pairs of timeseries.

2.2.1.3 SPIKE synchronization

The SPIKE synchronization algorithm quantifies the similarity between a set of input timeseries [11]. As illustrated in Section 2.2.1.1, we construct two timeseries containing events occurrence timings depending on the peaks detected on the input signals provided to the Peak Detector block (e.g., the peaks of the user's hands velocity).

Unlike the Event Synchronization block, this block accepts as input two timeseries of different types: they can be integer vectors (containing only -1 and 1, where the former indicates the absence of an event and the latter indicates the presence of an event) or real numbers vectors in which each element represents the occurence time of an event.

A coincidence profile, that is, a vector containing similarity information for each event of the input timeseries, is provided as output by the block together with an overall synchronization value.

3. Patches

The DANCE software libraries are composed by three collections of patches: the first is based on IMU sensors data, the second is based on motion captured data, thethird integrates IMUs and motion capture. In future versions we will integrate other sensors (e.g., for respiration).

The blocks described in Section 2 have been exploited in combination of the existing blocks provided by the EyesWeb XMI platform to define patches (i.e., programs) for the analysis of features and analysis primitives starting from both IMU sensors and motion captured data.

Inertial Movement Units (IMUs) are devices endowed with sensors that capture, sample and transmit data in real-time. Sensors, in the patches reported in Table 1, can capture data such as acceleration, gravity and magnetic field.

Motion captured (MoCap) data can be generated using dedicated hardware (e.g., the Qualisys motion capture hardware, in the case of the DANCE project). The patches listed in Table 2 compute movement features and primitives starting from MoCap data.

3.1 Patches computing features and analysis primitives from IMUs

Table 1 summarizes summarize the patches for computing features and analysis primitives from IMUs. To use and test the patches:

1. dowload the playback patch
   http://dance.dibris.unige.it/user_files/DANCE_Platform/release_march_2016/DANCE_platform_reader.zip
2. download the zip file extract the IMU patches
   http://dance.dibris.unige.it/user_files/DANCE_Platform/release_march_2016/DANCE_features_imu.zip
3. download and extract the sample IMU data and videos of sample dance fragments used by the patches to demonstrate examples of computation of features; **the data has to be extracted in the same folder of the patches**
   http://dance.dibris.unige.it/index.php/2-pagedance/16-dance-platform#download-sample-data
4. run EyesWeb and load the patches indicated in the table for each feature, and **run the patches in the indicated order**

Table 1: Patches computing features and analysis primitives from IMUs[1] (you need the sample IMU data to run these patches)

FEATURE/ ANALYSIS PRIMITIVE	DESCRIPTION	INPUT	OUTPUT
Energy	Computes Kinetic Energy starting from 3D accelerations provided by the sensors. Acceleration is integrated to compute velocity, since Energy = 1/2*mass*velocity^2 The resulting energy is normalized by the maximum energy that depends on the accelerometers range.	° Linear Acceleration x,y,z from IMU	Energy index [0,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_energy_imu.eywx			

[1] inputs and outputs are sent by the IMUs to the patches via the OSC communication protocol, see https://en.wikipedia.org/wiki/Open_Sound_Control

Slowness	This feature indicates whether the movement is performed slowly or not. It is computed by measuring the mathematical smoothness on the gyroscopes trajectories.	° Gravity x,y,z from IMU	Slowness index [0,1]
patch file : DANCE_library_slowness_imu.eywx			
Smoothness	This feature is based on Energy and Slowness. If movement exhibits high (respectively, low) slowness and no (respectively, many) energy peaks are detected then smoothness is high (respectively, low).	° Acceleration x,y,z from IMU ° Energy ° Slowness	Smoothness index [0,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_smoothness_imu.eywx			
Weight	This feature is related to the Laban's Weight quality (for details, see: Rudolf Laban and Frederick C. Lawrence. 1947 : *Effort*. Macdonald & Evans.) It is computed by extracting the Energy vertical component normalized to the overall amount of Energy in the movement.	° Linear Acceleration x,y,z from IMU ° Gravity x,y,z from IMU	Weight index [0,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_weight_imu.eywx			
Suddenness Impulsivity	Suddenness is computed using alfa-stable distributions. As described in Section 2.2, an alpha-stable fit is performed on peaks of accelerations. A movement is sudden when the product between alpha and gamma is high (see Section 2.2). The algorithm takes as input the 3D joint accelerations on a time window on which the suddenness has to be computed, and then it fits it into the alfa-stable distribution. The output of the app gets close to 1 (i.e., very sudden movements) when there are abrupt increases of the joint's velocity in the input signal, and vice-versa. Impulsivity is computed as a product of the Suddenness algorithm (see above) and the Direction Change (DC) algorithm. The Direction Change algorithm detects significant changes in the acceleration direction (i.e., considering two accelerations components, for example ax, and ay, DC is detected when it appears that ax +c > ay changes to ay +c > ax (c is a constant that is empirically set).	° Acceleration x,y,z from smartphone (50Hz)	Impulsivity index [-0,1] Suddenness index [--1,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_impulsivity-suddenness_imu.eywx			
Event Sync	As described in Section 2.3, event synchronization is computed by comparing events occurrences timings (i.e., peaks in joint accelerations). The more such events are close upon time the higher the synchronization level, and vice-versa.	° Events time series (extracted from IMU or MoCap etc.)	Synchronization index [0,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_event_synchronization.eywx			
SPIKE Event Sync	Another method to compute event synchronization is the SPIKE-synchronization algorithm. SPIKE uses event timings (see detail in Section 2.4) to compute three	° Events time series (extracted from IMU or	Synchronization value [0,1] Synchronization

	quantities: - the degree of synchronization - the mean distance between events occurrences (i.e., a measure of the variance of events in the considered time) the coincidence profile, a vector that binds a single synchronization value to any event that has been found in the time series.	MoCap etc.)	distance [0,1] Coincidence profile
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_spike_synchronization.eywx			

3.2 Patches computing features and analysis primitives from motion capture data

Table 2 summarizes the main patches for computing features and analysis primitives from motion capture data. To use and test the patches:

1. dowload the playback patch http://dance.dibris.unige.it/user_files/DANCE_Platform/release_march_2016/DANCE_platform_reader.zip
2. download the zip file and extract the motion capture patches and extract the motion capture patches http://dance.dibris.unige.it/user_files/DANCE_Platform/release_march_2016/DANCE_features_mocap.zip
3. download and extract the sample motion capture data and videos of sample dance fragments used by the patches to demonstrate examples of computation of features; **the data has to be extracted in the same folder of the patches** http://dance.dibris.unige.it/index.php/2-pagedance/16-dance-platform#download-sample-data
4. run EyesWeb and load the patches indicated in the table in the feature, and **run the patches in the indicated order**

Table 2: Patches computing features and analysis primitives from motion capture data (you need the sample motion capture data to run these patches)

FEATURE/ ANALYSIS PRIMITIVE	DESCRIPTION	INPUT	OUTPUT
Impulsivity Suddenness	Impulsivity is computed as a combination of the suddenness algorithm (see above) and the direction change. The direction change estimates whether the trajectory of the movement exhibits on the time window significant direction change (i.e. about 90 degrees). As described in Section 2.2 and in the previous table, suddenness is computed by performing an alpha stable fit on the input data. For example, the input data can be the 3D position of the user's hand. Then, an alpha-stable fit is computed on a vector (i.e., a time series) of position values over a short time window (e.g., 1 second). The product of alpha and gamma (see Section 2.2 for details) represents the level of suddenness. A sudden movement is also impulsive if it is "non-planned": this lack of intentionality and premeditation is modelled by computing the direction change (i.e., the variation of movement's direction). If a sudden movement is also non-prepared (that is, the hand movement does not evolve on "a line") then it is also an impulsive movement. For details, see: R. Niewiadomski, M. Mancini, G. Volpe, and A. Camurri. 2015. Automated Detection of Impulsive Movements in HCI. In *Proceedings of the 11th Biannual Conference on Italian SIGCHI Chapter* (CHItaly 2015). ACM, New York, NY, USA, 166-169. DOI=http://dx.doi.org/10.1145/2808435.2808466	° Position x,y,z from MoCap data	Impulsivity index [0,1] Suddenness index [-1,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_impulsivity-suddenness_mocap.eywx			

Fluidity	This feature is on the humanoid model described in Section 2.1.1 based on masses and sprngs. Given a rest position and an initial buffer of MoCap data, Fluidity is computed as the distance between model evolution (that represents the most fluid movement) and the real movement.	° Position x,y,z from MoCap data	Fluidity degree value
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_fluidity_mocap.eywx			
Event Sync	As described in Section 2.3, event synchronization is computed by comparing events occurrences timings (i.e., peaks in joint accelerations). The more such events are close upon time the higher the synchronization level, and vice-versa.	° Events time series (extracted from IMU or MoCap etc.)	Synchronization index [0,1]
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_event_synchronization.eywx			
SPIKE Event Sync	Another method to compute event synchronization is the SPIKE-synchronization algorithm. SPIKE uses event timings (see detail in Section 2.4) to compute three quantities: - the degree of synchronization - the mean distance between events occurrences (i.e., a measure of the variance of events in the considered time) the coincidence profile, a vector that binds a single synchronization value to any event that has been found in the time series.	° Events time series (extracted from IMU or MoCap etc.)	Synchronization value [0,1] Synchronization distance [0,1] Coincidence profile
patch files: 1. DANCE_platform_reader.eywx 2. DANCE_library_spike_synchronization.eywx			

4. IMU and motion capture sample data

Download:
http://dance.dibris.unige.it/user_files/DANCE_Platform/release_march_2016/DANCE_sample_data.zip

As reported in the above paragraphs, you have to download and extract some sample data in order to run the DANCE example patches. Without some sample data the example patches will not start, or will start but will not provide any output. The sample data is contained in a zip file and it is a collection of 2 trials consisting in data recorded by a motion capture system, a videocamera, and 4 IMU sensors placed on the dancer's limbs (wrists and ankles). The zip archive contains the following folders and files:

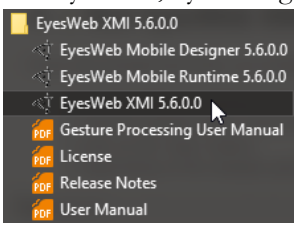
- folder: imu
  - file: 2015-12-15_t010_imu01_acceleration.txt (CSV format, 3D accelerometer data captured by IMU number 1 during trial number 010 on 2015-12-15)
  - file: 2015-12-15_t010_imu01_gyro.txt (CSV format, 3D gyroscope data captured by IMU number 1 during trial number 010 on 2015-12-15)
  - file: 2015-12-15_t010_imu01_magnetic.txt (CSV format, 3D compass data captured by IMU number 1 during trial number 010 on 2015-12-15)
  - .... (the same files are present for IMUs number 2, 3, and 4 and for trial number 011)
- folder: mocap
  - file: 2015-12-15_t010_mocap.qam (3D model of the dancer captured during trial number 010 on 2015-12-15)
  - file: 2015-12-15_t010_mocap.tsv (3D motion captured data of the dancer captured during trial number 010 on 2015-12-15)
  - file: 2015-12-15_t011_mocap.qam (3D model of the dancer captured during trial number 011 on 2015-12-15)
  - file: 2015-12-15_t011_mocap.tsv (3D motion captured data of the dancer captured during trial number 011 on 2015-12-15)
- folder: video
  - file: 2015-12-15_t010_video01.avi (video and audio file of the dancer captured during trial number 010 on 2015-12-15)
  - file: 2015-12-15_t010_video01_timings.txt (timings of the video frames contained in the corresponding video file 2015-12-15_t010_video01.avi)

- file: 2015-12-15_t011_video01.avi (video and audio file of the dancer captured during trial number 011 on 2015-12-15)
- file: 2015-12-15_t011_video01_timings.txt (timings of the video frames contained in the corresponding video file 2015-12-15_t011_video01.avi)

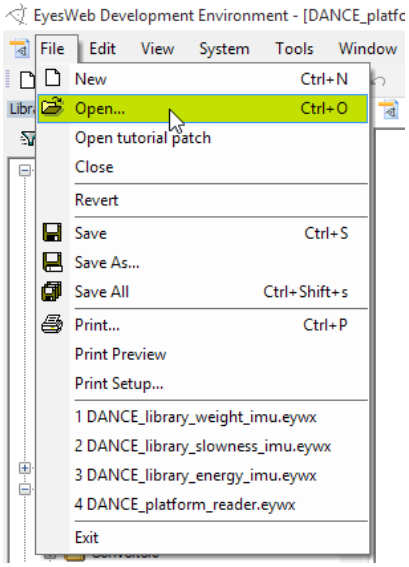
5. Run EyesWeb XMI, load one or more patches and execute them

When you downloaded EyesWeb, you installed it and you downloaded some patches plus the needed sample data you are ready to run the patches:

1. run EyesWeb, by clicking on the corresponding shortcut in the start menu:



2. load an example patch: select the file->open menu item; browse for the patch file; select "open":



3. click on the "play" button:

4. depending on the example patch your executing, different display windows will appear

Appendix

The algorithms of the DANCE software libraries presented in this document are described in detail in the attached papers:

- S. Piana, P. Alborno, R. Niewiadomski, M. Mancini, G. Volpe, A. Camurri, Movement Fluidity Analysis Based on Performance and Perception, in press, Proceedings of ACM CHI 2016: Conference: Human Factors in Computing System.
- P.Alborno, S. Piana, M. Mancini, R. Niewiadomski, G. Volpe, A. Camurri,, Analysis of intra personal synchronization in full-body movements displaying different expressive qualities. Submitted to the International Working Conference on Advanced Visual Interfaces - AVI 2016.
- S. Piana, A. Staglianò, F. Odone, A. Camurri, "Adaptive Body Gesture Representation for Automatic Emotion Recognition", ACM Transactions on Interactive Intelligent Systems, to appear

References

[1] B. R. Whittington and D. G. Thelen. 2009. A simple mass-spring model with roller feet can induce the ground reactions observed in human walking. Journal of biomechanical engineering.

[2] G. Dalleau, A. Belli, M. Bourdin, and J. Lacour. 1998. The spring-mass model and the energy cost of treadmill running. European journal of applied physiology and occupational physiology 77, 3 (1998), 257–263.

[3] O. Girard, J. Micallef, and G. P. Millet. 2011. Changes in spring-mass model characteristics during repeated running sprints. European journal of applied physiology 111, 1 (2011), 125–134.

[4] L. P. Nedel and D. Thalmann. 1998. Real time muscle deformations using mass-spring systems. In Computer Graphics International, 1998. Proceedings. IEEE, 156–165.

[5] P. Lévy. 1925. Calcul des probabilités, volume 9. Gauthier-Villars Paris.

[6] G. Tsihrintzis and C. Nikias. 1996. Fast estimation of the parameters of alpha-stable impulsive interference. Signal Processing, IEEE Transactions on, 44(6):1492-1503.

[7] E. E. Kuruoglu and J. Zerubia. 2003. Skewed-stable distributions for modelling textures. Pattern Recognition Letters, 24(1-3):339-348.

[8] S. Piana, P. Alborno, R. Niewiadomski, M. Mancini, G. Volpe, A. Camurri. 2016. Movement Fluidity Analysis Based on Performance and Perception, in press, Proceedings of ACM CHI 2016: Conference: Human Factors in Computing System.

[9] P.Alborno, S. Piana, M. Mancini, R. Niewiadomski, G. Volpe, A. Camurri. 2016. Analysis of intra personal synchronization in full-body movements displaying different expressive qualities. Submitted to the International Working Conference on Advanced Visual Interfaces

[10] R. Q. Quiroga, T. Kreuz, and P. Grassberger. 2002. Event synchronization: a simple and fast method to measure synchronicity and time delay patterns. Physical review E, 66(4):041904

[11] T. Kreuz et al. 2015. SPIKY: A graphical user interface for monitoring spike train synchrony. Journal of Neurophysiology

[12] F. E. Pollick et al. 2001. Perceiving affect from arm movement. Cognition 82.2.

[13] A. Camurri, P. Coletta, G. Varni, and S. Ghisio, "Developing multimodal interactive systems with eyesweb xmi," in Proceedings of the 2007 Conference on New Interfaces for Musical Expression (NIME07), 2007, p. 302305.

[14] G. Castellano, M. Mortillaro, A. Camurri, G. Volpe, and K. Scherer. 2008. Automated Analysis of Body Movement in Emotionally Expressive Piano Performances. Music Perception, 26(2):103–119.

[15] D. Glowinski, M Mancini, R. Cowie, A. Camurri, C. Chiorri, C. Doherty. 2013. The movements made by performers in a skilled quartet: a distinctive pattern, and the function that it serves. Front. Psychol. 4.

[16] G. Varni, G. Volpe, A. Camurri. 2010. "A System for Real-Time Multimodal Analysis of Nonverbal Affective Social Interaction in User-Centric Media". IEEE Transactions on Multimedia, Vol.12, No.6, pp.576-590.

[17] F. Bevilacqua, et al. 2009. "Continuous realtime gesture following and recognition." Gesture in embodied communication and human-computer interaction. Springer Berlin Heidelberg. 73-84.

[18] A. Savitzky and M. J. E. Golay. 1964. Smoothing and differentiation of data by simplified least squares procedures. Analytical chemistry, 36(8):1627–1639.

[19] P. Viviani and T. Flash. 1995. Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. Journal of Experimental Psychology: Human Perception and Performance 21, 1 (1995), 32.

[20] P. Morasso. 1981. Spatial control of arm movements. Experimental brain research 42, 2 (1981), 223–227.

[21] S. Piana, A. Stagliano', A. Camurri, and F. Odone. 2015. Adaptive Body Gesture Representation for Automatic Emotion Recognition. In Transactions on Interactive Intelligent System. ACM press, in printing.

[22] B. Mazzarino and M. Mancini. 2009. The need for impulsivity & smoothness-improving hci by qualitatively measuring new high-level human motionfeatures. In SIGMAP, 62-67

[23] Y. Miyake. 2009. Interpersonal synchronization of body motion and the walk-mate walking support robot. Robotics, IEEE Transactions on, 25(3):638-644

[24] N. Hogan and D. Sternad. 2007. On rhythmic and discrete movements: reflections, definitions and implications for motor control. Experimental Brain Research, 181(1), 13-30