WHOLODANCE

Whole-Body Interaction Learning for Dance Education

Call identifier: H2020-ICT-2015 - **Grant agreement no**: 688865

Topic: ICT-20-2015 - Technologies for better human learning and teaching

Deliverable 4.1

Data Integration, Algorithm and System Analysis, and Framework Description

Due date of delivery: June 30th, 2017

Actual submission date: July 10th, 2017

Start of the project: 1st January 2016
Ending Date: 31st December 2018

Partner responsible for this deliverable: Peachnote

Version: 4.0

Dissemination Level: Public

Document Classification

Title	Report on Data Integration, Algorithm and System Analysis, and Framework Description.
Deliverable	D4.1
Reporting Period	M1-M18
Authors	Vladimir Viro, Julian Schmidt
Work Package	WP4
Security	Public
Nature	Report
Keyword(s)	Similarity Search, algorithms, HLF, API, integration, web service

Document History

Name	**Remark**	**Version**	**Date**
Vladimir Viro, Julian Schmidt	Search an similarity algorithm and framework description	1.0	30th of June 2017
Stefano Piana	Analysis of suitability and quality of various LLF and HLF types for similarity search related tasks and particular application scenarios	1.1	6th of July 2017

List of Contributors

Name	**Affiliation**
Vladimir Viro	Peachnote
Julian Schmidt	Peachnote
Stefano Piana	UniGe
Katerina El Raheb	Athena RC

List of reviewers

Name	Affiliation
Katerina El Raheb	ATHENA RC
Akrivi Katifori	ATHENA RC

1 Contents

2 Introduction

The WhoLoDancE search and similarity engine exists in order to support data mining and real-time querying of the motion captured data collected within the project. In order to reduce the computational burden while retaining most of the semantically relevant content, the search and similarity algorithms are applied not to the raw video or the point cloud data collected during the recording sessions, but first of all to the algorithmically reconstructed skeletal configurations of the dancers, with the further addition of low-, mid- and high-level features (HLFs) derived from these data.

We refer to each skeletal coordinate (for example, a right elbow angle) as a dimension. Thus the recordings are for our purposes multi-dimensional time series of floating-point numbers and we are solving a problem of providing an adequate infrastructure for search and similarity measurements over high-dimensional time series. The dimensionality of recordings collected within WhoLoDancE is not constant and ranges between 72 and 168, depending on the amount of detail collected during the motion capture sessions. For example, the Flamenco recordings feature finger motions, while recordings of other genres, for which finger motions are less relevant and haven't been captured, do not.

We approach the problem of search in multidimensional time series with a variable number of dimensions by first performing the similarity computations on the one-dimensional time series constituting the data in our database and then aggregating these results across multiple dimensions in a suitable manner. This allows for the parallelization of most of the similarity computations across different dimensions, which is a useful feature given the large number of dimensions and the availability of parallel computing infrastructure (multi-core servers, GPUs, and compute clusters). The computation of similarity on a single dimension can also be parallelized naturally by partitioning the data, albeit at a slight loss in efficiency. These properties make our approach scalable and capable of supporting orders of magnitude more data than have been collected within the project.

3 Algorithm

3.1 Overview

We employ an index structure that organises the indexed time series within a tree structure, with time series that belong to nodes closer to each other in the tree being closer to each other according to the chosen distance metric (we use the standard Euclidean distance). The tree nodes correspond to words in a predefined alphabet. The words can be seen as approximations of the time series. The time series are split into a predefined number of equal length segments and their values are mapped to an alphabet in a certain way.

The index is filled gradually, with new time series being added to the corresponding nodes. When the number of time series in a node exceeds a certain threshold, the node is split in two.

In the following we will describe both the offline and real-time search and similarity application scenarios. The approaches share the multi-dimensional aggregation step, but differ in what happens with the one-dimensional data. We begin with the description of the one-dimensional search and similarity algorithms for both cases, and then proceed with the common description of the multi-dimensional aggregation step.

3.2 Search and similarity in one-dimensional time series

3.2.1 Index creation

Each index supports the search for queries of a predefined length. It contains all sub-sequences of the given length in all recordings. Therefore, it allows to find similar sequences starting at any position in the recording.

The first step of the index creation is a discretization step, whereby these sub-sequences are discretized into short sequences of literals (to be referred later as "words"). Such words are approximate representations of the time series. Their main purpose is allowing to determine minimal distances between sequences represented by these words in an efficient manner. The words are constructed as follows:

- To start off, the mean and the standard deviation values over the dimension in all recordings are calculated.
- Then each given subsequence is divided into segments of the same length and each segment is represented by the mean of its samples.
- These values are discretized into a small alphabet of a pre-defined cardinality. The letters of the alphabet correspond to intervals that are defined according to a Gaussian curve and the mean and standard error of the data set.

Another useful property of this representation is that the minimum distance between two sub-sequences represented by these words can also be calculated when the whole words or single letters have a different cardinality.

The next step is the creation of a search tree. Each node of the tree represents a word. Nodes can be differentiated into the root, internal nodes and terminal nodes. Terminal nodes - the leaves of the tree - contain pointers to the sub-sequences of time series that are characterized by the word associated to the node. Internal nodes are generated when the number of time series in a leaf exceeds a given threshold. Then the space of possible words covered is split into two not overlapping subspaces and new terminal nodes are created as children. The children represent words where the cardinality of one position is increased. Finally, the root represents the entire space.

To build the tree we start with an empty root node and successively add the words.

3.2.2 Ad-hoc real-time search

The aim of the ad-hoc search for a given dimension is finding a given number *k* of most similar time series in the recordings.

The search begins with an approximate search step. We traverse the tree for a terminal node that represents the word we get from the query time series. When doing this, we calculate the representation of the query for the resolution at any active node and follow the path according to the split policy. If we arrive at a terminal node we take the time series the node points to as approximate results. If no such node exists, the traversal fails at an internal node. We then just follow the path to the first "children" of the active node down until we reach a terminal node.

For the exact search step we calculate the actual distance of the query to the approximate results. We use the distance of the k-best time series among them as upper limit. Then the tree is traversed again. This time we do not only follow exact matches to the word we get from the query, but also other "children" whose minimal distance to the query representation is lower than the upper limit. The active nodes are managed in form of a priority queue. The upper limit also is constantly updated by maintaining an ordered list of the best *k* potential matches. Finally, when the traversal is finished, we return this list as result for the given dimension.

3.2.3 Offline data mining

For applications that do not require ad-hoc querying and analyse the dataset as a whole, we can address the similarity computation differently. We are interested in finding out all relevant correspondences in the dataset. On its face, the problem has at least quadratic complexity in the size of the dataset, as we would need to compare every pair of sub-sequences (assuming we are only considering sub-sequences of a fixed length). This becomes infeasible quickly with the size of the dataset. It is, however, possible to identify pairs of similar motions in linear time by using the already created index, as we have previously discovered when working with musical sequences.

The algorithm can be formulated in a map-reduce framework. We traverse all the leaf nodes of the index and for all pairs of (sub-)sequences contained in these nodes we count how often the pairs have co-occurred in the leaf nodes. This is done independently for all dimensions. The (sub-)sequence pairs with the highest number of co-occurrences are most similar.

The algorithm is highly parallelizable and our dataset can be processed in this way in under 10 minutes on a multi-core server. The output can be used for linking recordings together and finding interesting associations between them, as had been demonstrated during the internal review meeting held in Milan in December 2016, and again, more extensively, at the meeting held in London in June 2017 .

3.2.4 Combining similarity results over multiple dimensions

When combining results from different dimensions, user defined weight templates play an important role. These templates allow to define which features are used in the search and also differentiate the importance among those features.

We count how often a position in a recording appears in the k-best results for any dimension. Then we sum up over all dimensions and correct for the given weights in the template. This allows us to build a distribution of matches over positions in the recordings. We then select the positions with the highest values unless the time series defined by a position overlaps with a time series of a position with an even higher value. Overall, we extract two times *k* results in this step.

We then take these immediate results and calculate their exact distance to the query in all dimensions. These distances are then multiplied with the respective weights in the template and the products are summed up. The *k* positions with the lowest sums of distances are the final result.

3.3 Framework Architectural Goals

3.3.1 Workflow

The search and similarity platform should support automated submission of recordings and time series data by authorized users. It should support flexible querying scenarios, in which users can specify the features to be used for searching and their weights by themselves. The functionality is exposed via a REST API that consumes and produces data in JSON format. The API should be well documented and easily consumable in a variety of environments. A simple authentication scheme should be employed. Multiple deployment instances of the engine should be available to technical partners for experimenting with the API, the low-, mid- and high-level features that they can generate and submit to the search engine on their own, and the custom weighted templates without interfering with other partners or the production deployment. The deployments should be easily upgradeable in order to support short development and deployment cycles.

3.3.2 Performance

The search engine should support sub-second latency for complex searches spanning hundreds of time series dimensions over a multi-gigabyte dataset. For the real-time streaming search scenario, the search latency should be lower than 200ms. Addition of new data to the system should be fast at the target scale - the time series data should be available for real-time querying and offline similarity computations in less than a minute after its submission to the search engine that already contains gigabytes of time series data.

4 Service Implementation

4.1 Application

We have implemented the Search and Similarity Engine as a stand-alone Java application that can be interacted with over a REST API. The API implements a Swagger API definition, from which client implementation in multiple languages can be automatically generated.

4.1.1 Self-Hosted Documentation

The application hosts its own documentation that is available at the root endpoint of the API, e.g. http://search.wholodance.peachnote.com. The documentation describes all available REST endpoints and the data structures consumed and returned by the API.

4.1.2 Access Control

The access to the API is secured with HTTP Basic Authentication mechanism, which requires the user to provide a username and password before accessing the service. The authentication is transmitted using an HTTP header of any request sent to the service.

The API supports the Cross-Origin Resource Sharing (CORS) mechanism that makes it easy to consume the API from within any web page that wishes to make use of it.

4.1.3 Data Management

The search engine ingests submitted time series and meta data and persists them in a PostgreSQL database and on the file system. The configuration of the database access and the file system storage is provided using a configuration file.

4.1.4 Deployment

The application is automatically updated on every change in its source code repository resulting in a successful build.

Multiple deployments of the application are instantiated and configured with independent database and storage backends. This makes it possible for the technical partners who might prefer to work with their own deployment without interfering with the others.

The deployments are available under the following URLs:

Main deployment: http://search.wholodance.peachnote.com

Partner deployments:
http://polimi.search.wholodance.peachnote.com
http://polimi2.search.wholodance.peachnote.com
http://unige.search.wholodance.peachnote.com
http://unige2.search.wholodance.peachnote.com
http://athena.search.wholodance.peachnote.com
http://athena2.search.wholodance.peachnote.com
http://motek.search.wholodance.peachnote.com
http://motek2.search.wholodance.peachnote.com

4.2 API

4.2.1 Data structures

The search and similarity engine operates with three metadata entity types: the motion sequence recordings, the dimensions, and the weighted dimension templates, as well as the time series data describing the recordings in different dimensions.

The recordings are identified with IDs. They are associated with metadata, a map of key-value pairs (required or optional), such as duration, start time, title, genre, etc.

The recordings can be associated with multiple time series. The time series can come from various sources:

1. Motion capture (e.g. joint configurations)
2. Features extracted from the raw motion data (e.g. fluidity, fragility, etc.)

These time series may be recorded / computed for different frame rates.

The joint configurations and the computed features are identified by immutable IDs of string type. E.g. the angle of the left elbow can be called "left_elbow_angle", the change in this variable can be called "left_elbow_angle_derivative_1", a smoothed second derivative can be called "left_elbow_angle_30fps_derivative_2_smoothed_xxx", etc. Each such variable can be associated with metadata describing it (a map of key-value pairs, e.g. frame rate, a link to an outside description, the identification of the creator of this variable, an identifier of the group of variables this variable belongs to, etc.)

All time series in the system must refer to a recording to which it belongs. The reference consists of the recording ID and the time offset relative to the beginning of the recording, at which the time series begins.

Variables can be combined in named weighted sets (weighted dimension templates) used for search and similarity computations. Such weighted sets are represented by a map of key-value pairs, in which the keys are strings representing the variable names (like “left_elbow_angle”) and the values are the weights (floats from the range between zero and one) that should add up to one (otherwise they are normalized accordingly by the system on submission). For example, such a weighted set can be named “upper body” and weigh all variables describing the lower body movement and their derivatives as zero.

4.2.2 Data submission

Recordings, time series and weighted feature sets can be submitted to the search and similarity platform.

A recording can be submitted along with its metadata if no other recording with the same ID is already present in the index.

A time series can be submitted if it refers to an existing recording and no time series of its type has been associated with the recording before. The time series submission consists of the associated metadata and a pointer to the file containing the time series data. The file will be asynchronously fetched by the search engine and ingested. The submission process can either succeed or fail.

A weighted feature set can be successfully submitted if its description contains an ID that is not already in use, a name and a non-empty set of weighted variable names with weights between zero and one.

4.3 Search

The search and similarity platform responds to search requests.

The search requests can be of two types: “live” and “in-the-database”.

Live requests consist of the time series data of features that the users are searching by and optionally an ID of a weighted feature set to be used for weighting the variable importance during the search.

For “in-the-database” requests the time series data are replaced by the pointer to a place in a recording in the database. Additionally, search requests may specify filter conditions referring to the metadata associated with the recordings retrieved from the search index.

The search results consist of a list of pointers to places in the recordings (recording IDs and offsets within the recordings) and the estimated similarity of these places to the query represented by positive numbers between zero and one.

5 Consolidation and Assessment of LLF, MLF and HLF for Similarity Search

Following the needs analysis as defined through the work in WP1, all partners both technical and dance experts have selected a subset of the HLF which seems appropriate and satisfying in order to analyse the movement of the recordings to deliver meaningful Learning Scenarios. In particular, this selection was based on the work reported in the deliverable "D1.6 HLF Definitions for the Learning Scenarios", as well as the results of focus groups and questionnaires. Taking into account the lack of standard universal features that are used in movement analysis and are proved appropriate for dance learning in different dance genres, according to previous relevant work, the selection was based on a) research work in non-verbal communication and movement expressivity, b) relevant representation models for movement description and c) user-centered methodologies (focus-groups and questionnaires).

Following this approach, we have decided to focus on specific Movement Qualities and Movement Principle descriptors. These concepts are described in detail in D1.6 HLF questionnaires, while additional information about the process and detailed conceptual modelling can be found in D3.1 Report on semantic representation models, and D3.2 Report on emotional representation models. Within this framework, the WhoLoDancE Movement Library interface, which was designed and developed by Athena and PoliMi, provides an annotation interface which integrates these concepts as controlled vocabularies offered to the end-users. The annotation interface allows the dance experts to annotate specific sequences in order to collect ground-truth data for validating the HLF extraction algorithms. In addition, the selected HLF, as well as the related concepts and their relationships (as described in detail in D3.1) can be used as similarity metrics, to filter the dimensions of similarity in a more meaningful way.

In order to assess the suitability of movement features for the similarity search framework we generated a set of preliminary features based on algorithmic extractions developed by PoliMi and UniGe, the set of evaluated features is described in the following sections. The Similarity Search user interface, was developed as an extension of the WhoLoDancE Movement Library interface which was developed by Athena and PoliMi and provided the framework to assess the similarity search algorithm for LLF and MLF during the first period of the project. The search page is accessible at the following link: http://mir.deib.polimi.it/wholodance/similarity and is shown in Figure 5.1**Errore. L'origine riferimento non è stata trovata.** and Figure 5.2.

5.1 Low-level features

A preliminary evaluation of the functionalities of the search engine was conducted with low-level features the features included in the process are the following:

- Absolute positions of joints
- Velocities of Joints

To assess the functionalities of the search engine with low-level features we created a website that enables the users to choose a dance sequence from a selection of recordings from different genres, proposes different search criteria (search templates) based on low-level features, and searches for similar sequences.

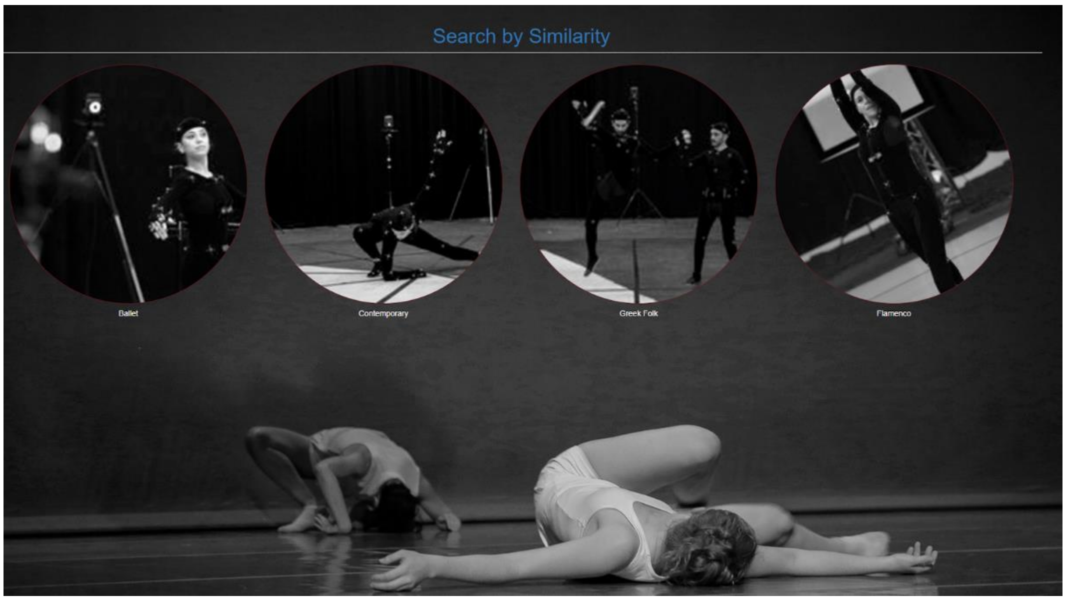
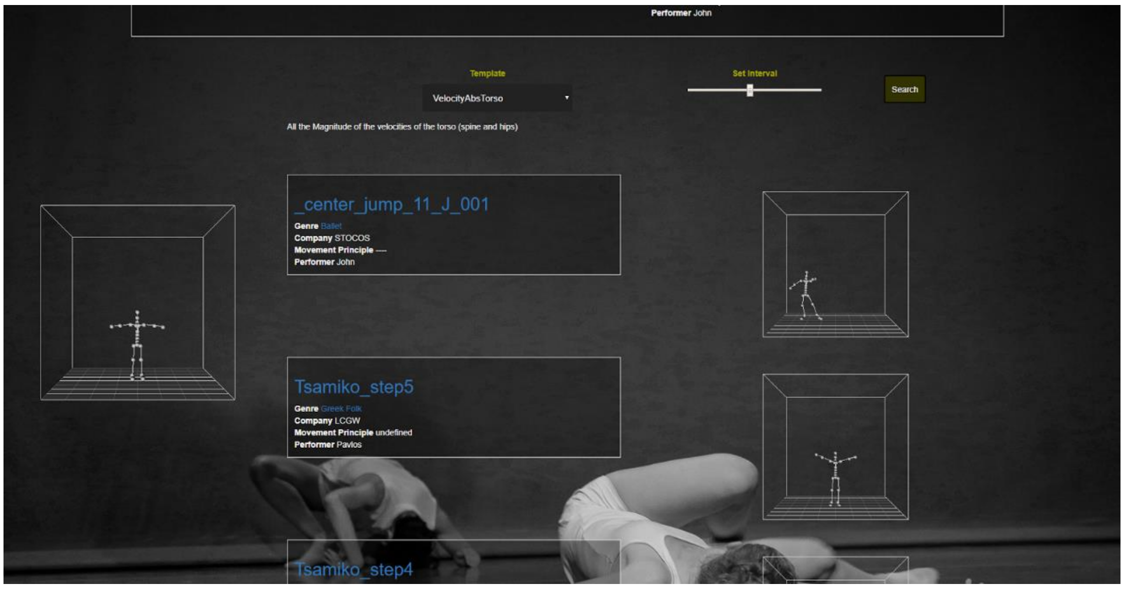


Figure5. 1 The main page of the search page, the user can select among four dance genres



Performer John

VelocityAbsTorso

All the Magnitude of the velocities of the torso (spine and hips)

_center_jump_11_J_001
Genre Ballet
Company STOCOS
Movement Principle —
Performer John

Tsamiko_step5
Genre Greek Folk
Company LCGW
Movement Principle undefined
Performer Pavlos

Tsamiko_step4

Figure 5.2: a dance sequence was selected, the 3D visualization of the performance is displayed on the left, the selection of the search templates if on the middle and the search results are shown on the right.

5.2 High-level features

In order to assess the descriptive power of High level features related to movement principles and qualities (described in D1.6) that are currently in refinement, we created a second search page that focuses on high level features, the search engine in this case includes preliminary versions of a subset of the features identified and described in D1.6 the list includes:

- Energy of movement
- Symmetry
- Lightness

- Coordination of different body parts
- Smoothness

The interface of the search tool is the same as for the one focussed on low-Level features: the users chooses a dance sequence from a selection of recordings from different genres, then select a search criteria (template) from a drop-down menu. The results of the search will show similar dance sequences according to the selected high-level feature (Figure 5.3).

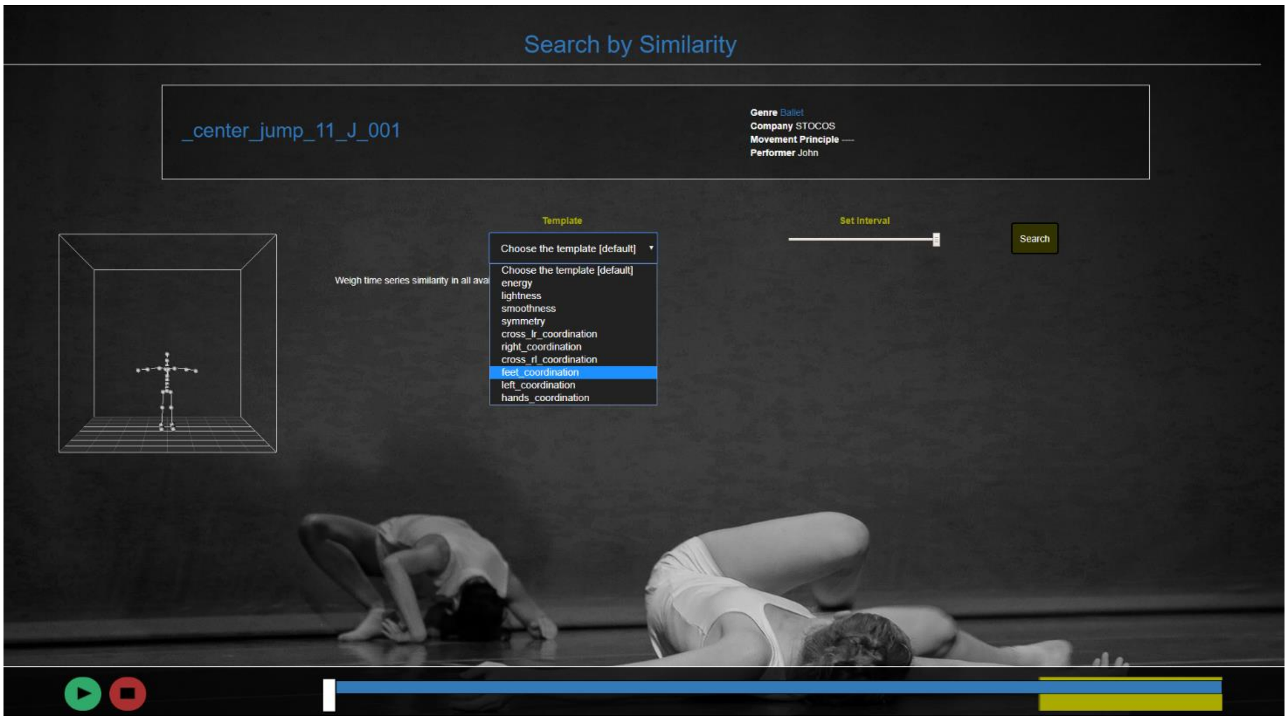


Figure 5. 3: Web page Interface of the HLF based search engine.

The search page is accessible at the following link: http://mir.deib.polimi.it/wholodance/similarity/hlf.php

5.3 Assessing the descriptive power of low and high level features

Both the low-level and high-level features developed so far proved to be suitable to be used in the similarity search engine, to assess the quality of the search results using both Low and High level features. We will proceed in the following months with a perceptive study involving dance experts and teachers that will qualitatively evaluate the results given by the search engine.