Techniques for automated analysis of saliency in human full-body movement

Serafino Gabriele
Lo Luca

Advisors:
Volpe Gualtiero
Camurri Antonio

2024

Abstract

In contemporary research, the topic of salience has become a focal point, with a growing number of studies dedicated to its exploration. Various fields, including Medical, Financial, Military, Psychology, Kinesiology, have approached the problem in diverse ways.
Our thesis aims to develop effective heuristics and algorithmic approaches for detecting salience in the context of movement. Specifically, we focused on the domain of dance movements, defining a concept of salience within this context.
In our purpose the salience is meant as a change in the behaviour of the dancer: static movement with respect to dynamic movement, repetitive movement with respect to a chaotic one, using an arm and then stopping it for using the other one, ans so on...
The study involves a thorough analysis of Motion Capture (MOCAP) data, where we navigate dataset constraints and boundaries to select significant features that aptly represent our chosen domain.
To address the salience detection problem, we employ several approaches: among them, a Supervised Machine Learning (ML) Algorithm.
For the ML approach we defined a specific prototype of Sliding Windows (SW) able to convey information through several frames of the videos, with the goal of giving to the specific model the most significant piece of information in order to have the best results.
Then we tested the algorithm at different levels of generalization.
The goal of this work is to develop a system able to recognize the salience in human full-body context in real time.

Contents

List of Figures

List of Tables

Acronyms

3D Three dimensional. 10, 16, 27, 28

CPD Change Point Detection. 10, 12, 13, 20

CUSUM Cumulative Sum. 12

DFT Discrete Fourier Transform. 48, 50

DNN Deep Neural Network. 52

EST Event Segmentation Theory. 17

EWMA Exponentially WEighted Moving Average. 12

GLR Generalized Likelyhood Ratio. 12, 13

IMU Inertial Measurement Units. 9

LODO Leave One dancer Out. 55

LOSO Leave One sample Out. 55

LOTO Leave One take Out. 35, 55

ML Machine Learning. 2, 10, 21, 32–34, 52

MOCAP Motion Capture. 2, 15, 16, 20, 35

NaN Not a Number. 38, 39

PELT Pruned Exact Linear Time. 13

QTM Qualisys Track Manager. 10, 16

RuLSIF Relative unconstrained Least-Squares Importance Fitting. 13

SW Sliding Windows. 2

TSV Tab-Separated Values. 16, 18, 28, 35

1 Introduction

The thesis focuses on defining movement saliency and developing a model for its automatic detection and analysis. The work uses the datasets from the archives of Casa Paganini - Infomus, an international Research Center, and consists of videos of dancers performing on-site captured using a motion capture techniques.

1.1 The problem of automatic salience

Salience comes from the Latin salire, meaning "to leap". Something with salience leaps out at you because it is unique or special in some way.[14][11] So salience is something that catches our attention and based on its definition and it exists in various contests. In this thesis we want to focus on the salience in human movements. Even if we narrow the definition of salience in a context of human movements, what is a salience is still not really defined and based on our own interpretation. A dad watching his toddler, it's more interesting to detect any dangerous salient movement of the baby. Maybe he is going to touch something sharp or eat something bad. A dance teacher could set his focus on the dancer's movements and detect if the student's movement is performed correctly. Or a school teacher, during a test, could focus her attention on her students to see if someone is trying to cheat. Our definition of salience is defined in further defined later on (Chapter 3).

One of the major challenge in automate the analysis of salience in movement is the variety and complexity of human behaviours. First of all the human movements is the result of the combination of different part of the body from the evident wave of a human hand to the slightest grin. Humans are capable of different and diversified movements with a big range of possibilities and this makes salience in movement a subjective interpretation. But we need also to contextualize the situation where that salience movement is performed. if the same salience is taken and placed in another context, it could lose its saliency and become something that don't catch our attention anymore. Bringing up the example of the dad and toddler, if we place the child in a safe environment without any dangerous or sharp item out of his reach, the same salience lose its importance here. So salience in movement is subjective and so really hard to generalize and this makes really hard to develop specific algorithm able to generalize every salience of movement.

Other than the subjective and diverse context of salience in movement, we need to consider the nature of multidimensional data of movements and how to manipulate it and what feature extract from it. The data could also be subjects of noise. There are several ways to capture movement from a human body. For example an Inertial Measurement Units (IMU) that measure body's specific force, angular rate and sometimes the orientation of the body using a wearable sensing device or a smartphone. An optical motion capture system where it is required to wear markers and many cameras to cover wide areas. Or simpler, a normal video camera. Every techniques and devices differs for its price and efficiency. In the case of the camera, we can extract a blob, a figure that represent the subject's silhouette in the video.

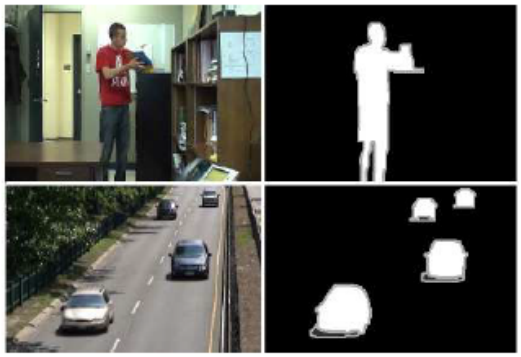

Figure 1: Blob extraction

But here we have a problem to extract the shape of the person from the background. And what if we have multiple moving person in the background? Depends on the method to capture human movements, we have to deal with different kind of noise, leading to a hard time to distinguish important pattern.

Developing an automatic analysis of salience in movement, it's a challenging task where it's required an interdisciplinary approach and integrate different techniques, because it's not just about the movements, but also the context and how we perceive them. We have to deal with a lot of data from different kind of sensors or devices with different methods which can be noisy, expensive and hard to interpret. Figuring out algorithms more suitable for our situation and that can work across different situations is a big challenge.

1.2 Motivation and goals

We have seen how salience in movement differs from context to context. Being able to develop an automatic analysis of salience in movements can help humans to detect automatically anything critical without the necessity of a human supervision.

Our objective is to develop automatic systems to analyze human movements to detect salience movements so a machine can automatically detect and/or predict it giving the possibilities to detect salience movement in humans.

1.3 Thesis structure

Here we describes the structure of the Thesis, giving a quick overview of the content described in each chapter.

Chapter 2 starts with a state of the art of the Change Point Detection (CPD) methodologies and the starting point of our Thesis that continues the work of a previous PhD Thesis and how we will continue the work

Chapter 3 defines what is a salience for us compared to the salience introduced in chapter 1, and show how we extracted the ground truth and it refers to accurately annotated or labeled training data used as the basis for evaluating model performance. It represents the objective truth against which model predictions are compared for accuracy assessment.

Chapter 4 describes the low level features extracted from the raw movements data of the dancers. For low level features we describes the transformation of the data collected by recording the dancers with the QTM using a motion capture techniques to retrieve a 3D position of each marker placed throughout the body and transform it into features like the kinetic energy, acceleration, trajectory and so on.

Chapter 5 is about the supervised ML approach, describing the full work starting from the low level feature, preprocessing, the model selection to the results of different settings to test the model.

Chapter 6 is the conclusion chapter where we make an overview of the

work done and its result. Also we consider the possible future work that can be done, starting from this Thesis.

2 Methodologies

2.1 CPD algorithms

This chapter is a list of state of the art Change Point Detection (CPD) algorithms.

Cumulative Sum (CUSUM) uses cumulative sums of deviations from a baseline (such as the mean) to detect shifts in the data. CUSUM is sensitive to gradual changes and can be configured to detect shifts of various magnitudes. It computes the mean and standard deviation of the timeseries and by computing the z-score of the timeseries and cumulative summing all the z-score values, detect if the value goes over a defined threshold value.

Bayesian CPD Utilizes probabilistic models and Bayesian probability theory to estimate the likelihood of change at different points in the data. Can handle a wide range of data distributions and can be customized for specific change scenarios. Can handle uncertainty in both data and model parameters using posterior probability distributions.

Generalized Likelyhood Ratio (GLR) Maximizes the likelihood ratio between statistical models of data before and after a change point. Effective for abrupt or gradual changes in data distributions. Identifies significant shifts in data distributions.

Exponentially WEighted Moving Average (EWMA) Uses weighted averages of historical data to detect changes in real-time data streams. (Data in the past have less weight with respect more recent data) Responsive to recent changes while considering historical trends.

Binary Segmentation It is a sequential approach: first, one change point is detected in the complete input signal, then series is split around this change point, then the operation is repeated on the two resulting sub-signals. The benefits of binary segmentation includes low complexity (of the order of $O(Cn \log n)$, where is the number of samples and the complexity of calling

the considered cost function on one sub-signal), the fact that it can extend any single change point detection method to detect multiple changes points and that it can work whether the number of regimes is known beforehand or not.

Pruned Exact Linear Time (PELT) An efficient algorithm for finding change points, balancing accuracy and computational efficiency. (Optimization function which tries to find the best configuration of change point, it set a cost for each data in the data-series, the cost is higher if the probability of having a change point is lower). Suitable for real-time analysis of large datasets.

Kernel Change Point Detection Utilizes kernel functions to identify change points, representing data in higher-dimensional spaces. (for non linear data relationship and changing point). Captures non-linear changes in data patterns, allowing for detection of complex historical trends.

Relative unconstrained Least-Squares Importance Fitting (RuLSIF) is a method that focuses on learning the differences in distributions between two datasets. It aims to assess whether a change has occurred in the underlying data generating process by comparing the distribution of a reference dataset with a test dataset. RuLSIF is sensitive to subtle changes in the distribution of the test dataset concerning the reference dataset. It's particularly useful when you have a labeled reference dataset to compare against, enabling the detection of distributional shifts. This approach is better then GLR because it uses a dataset labeled so it can find small changing in the test dataset.

2.2 Related Work

The material we started from was provided by researchers at
Casa Paganini - InfoMus [12]. By exploiting the existing results, our goal was to find algorithms and heuristics for dealing with the problem of salience.
The given scenario is the following: three dancers from Casa Paganini - InfoMus [12], Cora Gasparotti (Figure 2), Marianne Gubri (Figure 3) and Muriel Romero (Figure 4), performed dancing movements by emphasizing some specific tasks, in order to highlight a combination of graceful and fluid movements, as well as angular and hurried gestures.

Those performances has been recorded with two professional cameras (frontal and lateral view, 1280 $\times$ 720, 50fps) at Casa Paganini, and the result of this procedure was a set of *takes* (each *take* has a duration between 30 seconds to 180 seconds).
So, for each dancer, we have between 5 to 15 *takes*.

Figure 2:
Cora Gasparotti

Figure 3:
Marianne Gubri

Figure 4:
Muriel Romero

Raw data has been collected by using a Qualisys [13] Motion Capture (MOCAP) system:

- Qualisys Cameras: these are specialized for capturing and tracking movements in three-dimensional space.
  These cameras operate based on the principle of detecting and analyzing markers placed on objects or individuals within their field of view. The cameras emit light, which is then reflected by small, passive markers, such as reflective spheres or retro-reflective devices. This process allows the cameras to discern and meticulously track the position and movement of each marker within their field of view.
  In our context, a Qualisys system endowed of 16-cameras was used (fs = 100Hz). (Figure 5)

Figure 5: Camera Qualisys

- Markers: are integral components of motion capture systems, they are identifiable points placed on objects or the human body. These markers can take various forms, such as reflective spheres, strips, or discs, each designed for specific tracking purposes.
  In our cases we used 64 spherical markers positioned on the whole body. (Figure 6)

Figure 6: Markers Qualisys

The MOCAP data was collected in a Tab-Separated Values (TSV) file by using Qualisys Track Manager (QTM) [13] which is a software used for files handling MOCAP data, hence for each frame the position of each marker in the 3D space (x,y,z) is available.

2.2.1 State of the art

One of the most important work related to our context was the one done by E. Ceccaldi in her PhD Thesis [6], in that discussion, the problem of salience detection, has been defined with some psychological approach related to the body behaviour, for the definition of the salience, *takes* have been segmented (in the time).
In this work, segmentation is based on boundaries between events, as described by a cognitive theory on how this process unfolds in the human mind, namely the Event Segmentation theory [16]. According to this theory, a boundary is perceived whenever a meaningful (i.e. salient) change is perceived in the ongoing situation. Following the theory, in [6] changes were operationalized as follows:
Thus, salience is defined to a higher level as follows:

- C1. Time: which is the timing and the rhythm of the movement, for instance if the dancer accelerates her actions.
- C2. Space: if the attention and the direction of the movement starts to pointing something different.
- C4. Character Location: if the character moves on the stage, or, in our cases, if the dancer moves from a point to another.
- C6. Causes: Which are the causes and appraisal, if a new state of affairs leads to a subsequent event.

Hence the ground truth was taken by considering these 4 different aspects by psychologists which are trained on EST [16] principles by using a software for segmentation: ELAN [3] (Figure 7).
This software allows to make the segmentation of videos (or an audio track) by defining some classes of event segment and a hierarchy among them.
The *features* which have been considered in that case are the following:

- For the *Time* it was used the General Quantity of Movement and the Chest Quantity of Movement.
- For the Space the author employed the Directness of Head Movement.
- The Character Location has been detected by looking the Density of Chest Trajectory.

- The causes, instead, has been used only has an additional ground truth.

All these features has been extracted by using an application (patch) developed for EyesWeb XMI (Figure 8), a software platform able to build over the TSV, which contains our raw data, some useful characteristics such as: kinetic Energy, Point density, Directness, and so on...
For instance, kinetic energy (of all the markers) was taken as a measure of the overall amount of movement.
The Chest Quantity of Movement was evaluated by considering the kinetic energy of the marker related to the chest. The Directness of Head is a measurement of how the dancer's head moves in curvilinear trajectory, the higher is the value, the more the movement following a straight line.
For finishing, the Density of chest trajectory is an indicator of whether movement is localized in a small region in the space rather than spanning the whole space, i.e., higher density indicates that the actor has moved in a smaller region.

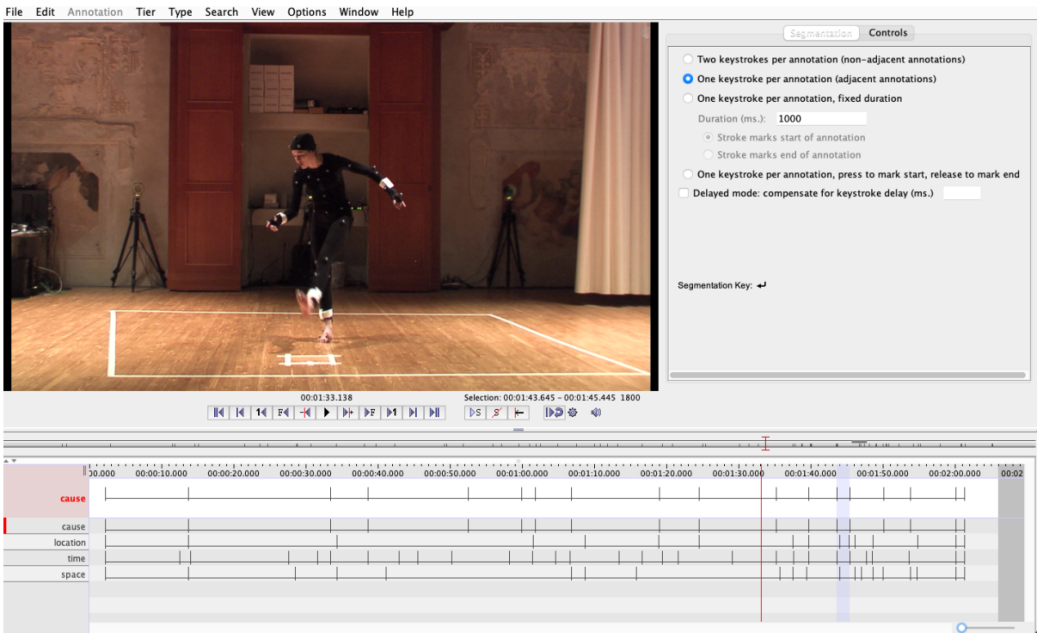

Figure 7: Annotation of the different salience with ELAN.

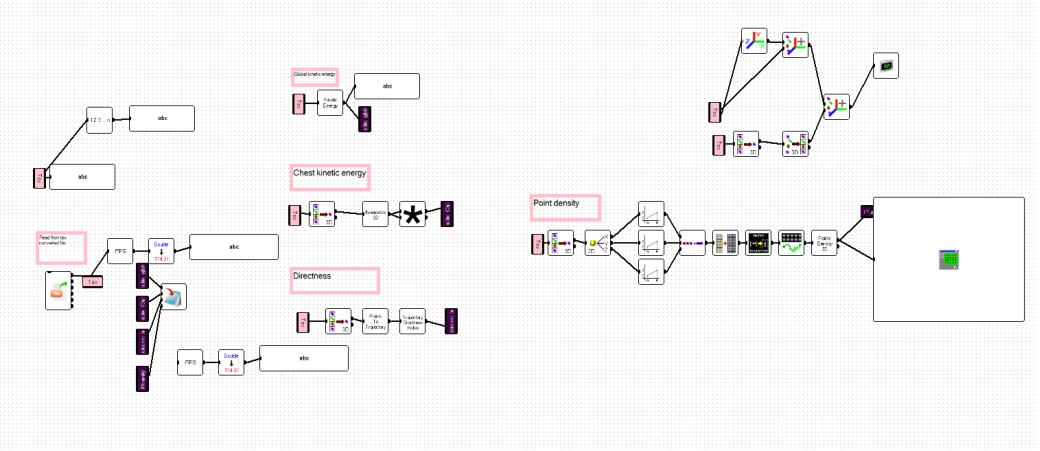

Figure 8: One of the patches implemented on EyesWeb for extracting the features.

2.3 Structure of workflow

In this chapter it will be explained which is the workflow that we did during our study (figure 9).
From the raw data analyzing to the salience classification and experiment testing:

1. Physical Layer: this layer was already achieved since the Motion Capture was given by the researcher at Casa Paganini - InfoMus [12] (chapter 2.2).

2. Salience and Ground Truth: After reviewing the material provided at the previous step, we evaluate which is the better definition of salience in our context, because we had to take in account the different possible movements of the dancers.
   Then, we wrote down the Ground Truth based on that definition of salience (chapter 3).

3. *Takes* selection: With respect to the point 2 we had to choose wisely the *takes* which better represent the context, because not all the videos were adapt for representing our domain (chapter 5.1).

4. Data Cleaning: We cleaned up the data with a Linear interpolation (chapter 5.1.1).

5. Low Level Features: Then, we have defined the *features* at a lower level, global kinetic energy, Repetitiveness etc.
   Hence these features have been collected thanks to EyesWeb (chapter 4).

6. Model Selection: We had explored two different approach for addressing the problem, a machine learning approach, and a Change Point Detection (CPD) technique (chapter 5 or 6).

7. Sliding Windows Template: For the specific model we built a template of sliding window able to convey the right piece of information basing on the used model (chapter 5.1.2 and 6.1).

8. Mid Level Features: For the Machine Learning (ML) approach, mid-level features have been selected, such as mean, variance, entropy etc. (Chapter 5.1.3)

9. Samples Definition and Downsampling: then for the ML model a specific definition of samples has been set (chapter 5.2).

10. Data Normalization: Data has been normalized (chapter 5.1.4)

11. Algorithm Selection: For the specific approach, a specific algorithm was used, for Machine Learning (ML), a Random Forest was employed for doing the classification (chapter 5.3).

12. Parameters Definition: Basing on the approach used parameters have been chosen, for ML a cross-validation has been done (chapter 5.3).

13. Model Evaluation: For finishing, different level of generalization has been explored (chapter 5.4).

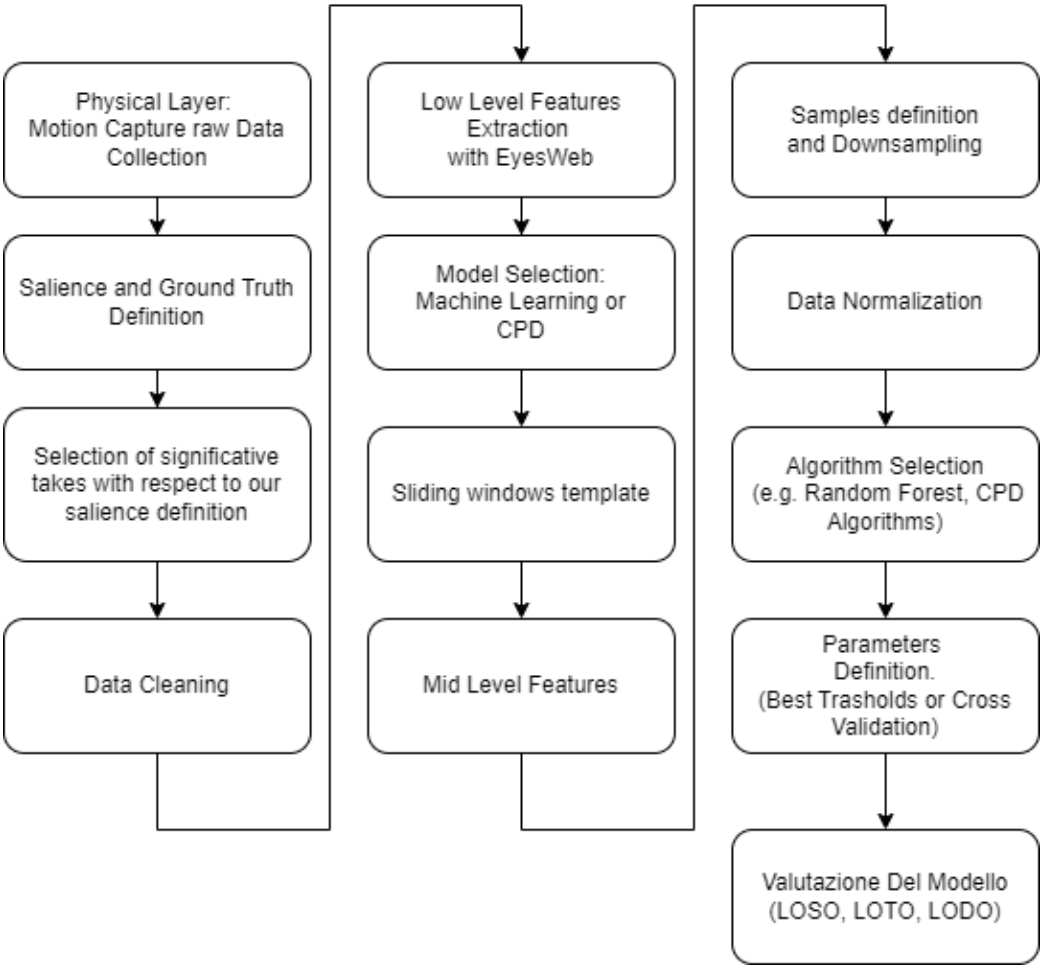

Figure 9: Workflow chart

3 Salience definition

When we started to approach this problem, one of the first challenge that we had to deal with, was to define what is a specific event that change in some way the behaviour of the dancer.
We address this event as a ***Salience***.
The salience in the context of movement can be viewed in different ways, from an holistic one, to a granular one, by considering a movement within a large temporal window, or by viewing it with a short temporal span.
The problem can be addressed from different point of view, for instance, you can make a comparison between a movement which is more *impulsive* or more *sudden* [9], or you can look at the movement in terms of *Lightness* or *Fragility* [10].
The first issue that we have noticed is that, if we thought the salience as something which is too much related to the causes, and the psychological aspect of the human behaviour, any algorithm over that kind of approach gave us bad results, because addressing an intention or an emotion is a tricky challenge.
So, we have focused on looking more the movement of the dancer from the point of view of the Kinesiology, so if there is a movement behaviour which is similar with the observation of the whole set of frames which precedes that specific frame, then for us is not a salience, if instead the frame subjected to evaluation is the first frame before a whole set of frame which has different characteristics in therms of movement with respect to the set of frames before the specific evaluated frame, then for us it is a salience.
After that, by looking the set of salience resulting on the step before, and since the quantity of frames labeled "salience" is significantly lower with respect to the frame which are labeled "non-salience", we became aware that for an algorithm of machine learning this scenario could be a problem, because unbalanced dataset are often hard to address [15]. For trying to reduce the oddness between the two classes we had defined the concept of event as something which is more frequent, by using a more granular segmentation, hence visible changes in the movement such as an arm which starts to move with respect to the other one, or a repetitive movement and then a chaotic one, and so on...

3.1 Ground Truth

Then we thought to rebuild the Ground Truth, so with ELAN [3] we took annotations of all the events building upon the considerations made before (Figure 10). Since each event is detected between two sets of consecutive frames, events can be addressed as follows:

- S_1 - This event occurs when the dancer moves from a specific position in the stage to another position in the stage.
- S_2, S_3 - This event detect a change in the movement from the point of view of the repetitiveness, hence, if the dancer moves from a repetitive and orderly action, to a chaotic one (S_2), or vice versa (S_3).
- S_4, S_5 - It occurs when the dancer moves from a crouched position to an extended one(S_4), or vice versa(S_5).
- S_6, S_7 - Event occurred when the dancer moves from a static position of the head to a dynamic one (S_6), or vice versa (S_7).
- S_8 - The event which occurs when the dancer change the dynamic of her distal points: for instance if she moves her left wrist in the first set of frames, and then moves her right ankle, or if she moves her left ankle, and then moves her right wrist, or if she moves both the wrists and then stops to move one of the other distal parts.
  Since there are 4 different distal parts, left wrist, right wrist, left ankle, right ankle, and all of them can be moving or not, we can think them as a set of boolean variables (1 if moving, 0 if not), we can think the vector v_0=[0,0,0,0] as the vector which represents the situation in which the dancers does not move anything, and the vector v_n=[1,1,1,1] in which the dancer moves all the distal parts (n=15).
  Thus we have 2^4 different states (st_s) and hence the events (S_8) represent the transition from st_i to st_j ($i, j \in$[0,16), i$\neq$j).

So, in total we have 23 different kind of salience.
The salience from S_1 to S_5 consider a more holistic change between the two set of frames, instead the set of salience which goes from S_6 to S_8 is referred

to a local movement.
During the phase of taking the Ground Truth, we did not make meaningful segments, we just consider the events occurrence, then for avoiding perceptual fusion problem (two different frame are perceived as one if the time between them is lower then 10 ms), we took the start of the segment (which is our salience) by looking frame per frame in the area of the perceived salience. For finishing, all the starting frames of the events have been exported in a text file where each value in that file represent the timestamp in seconds (with centesimal precision) of an occurrence of a frame representing a salience. For having the number of the frame instead of the timestamp we have computed the following formula: n_f=floor(n_s*50), since we have 50 frames per second (floor is the rounding down).

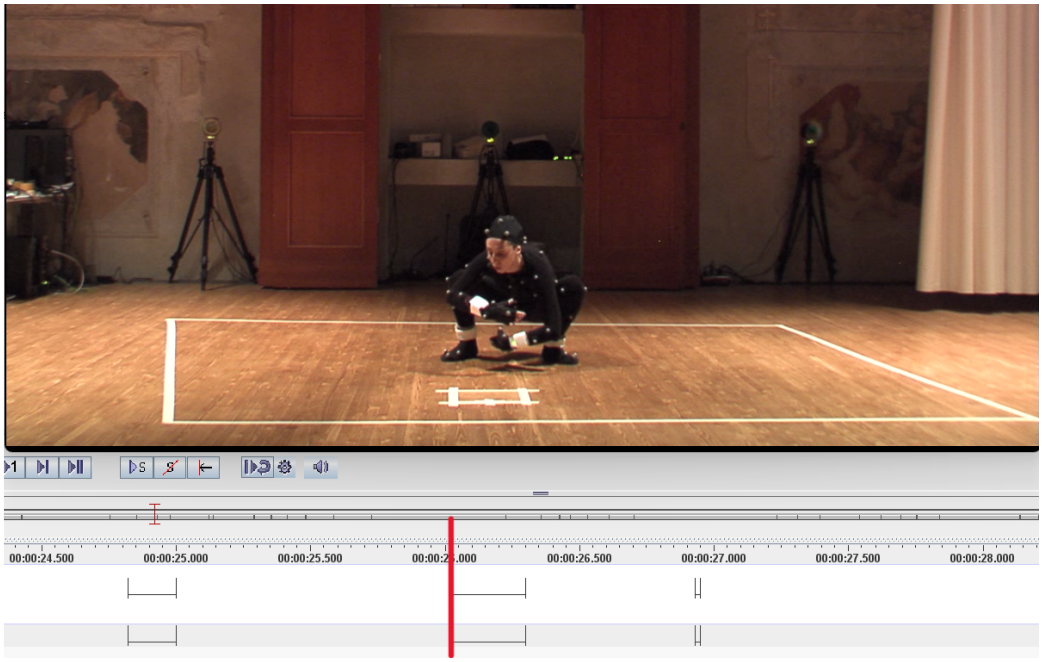

Figure 10:
The red line is over
the frame were
the salience is annotated (S_4).

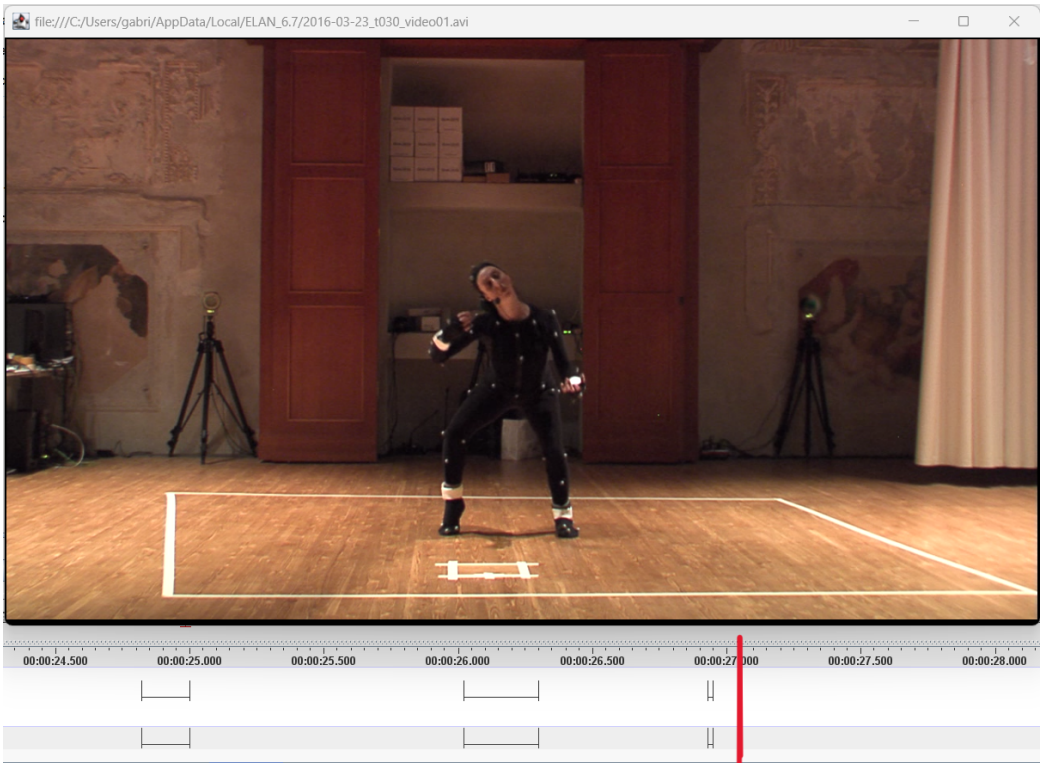

Figure 11:
After ≈ 50 frames (1 second)
with respect to the
salience in Figure 9
the dancer is extended.

4 Low level features

In order to describe correctly the domain and to detect properly the different kinds of salience, we had to find out which were the most representative features of our system.
Raw data was conceptually a three dimensional matrix in which on the first dimension we had the frame number, in the second dimension we had the ID_s of markers (e.g. ARIEL which refers to head marker, RPLM for the right wrist, and so on...), and on the third dimension we had the spatial axes (x,y,z), because each marker in each frame should have 3 values referred to its 3D location.
From these data we had to build significant low level features, because the spatial information of markers alone was not meaningful for salience detection.
In order to select features, we started from the definition of salience as reported in chapter 3.
For this discussion we start to talk about local features (S_6, S_7, S_8) for arriving to the holistic ones $(S_1 - S_5)$, because for addressing local salience, only one feature is enough, instead multiple features have been involved for a salience which is evaluated on the whole body like S_1.
Thus we prefer to define the feature of local salience as first.
All the features have been extracted by using an application (patch) developed for EyesWeb XMI, for each feature there will be an illustration on how the Objects within the application have been used.
Before talking about the low level features used in the process we want to mention the *Postural Tension* feature.
The Postural tension is a measure on how much the dancer's body is twisted; for evaluating that, unit vectors of head, shoulder, trunk and hips are compared in terms on how much they are parallel, the less they are parallel, the higher is tension.
At the beginning we thought that combining the postural tension to other features we could have good results, but by doing some tests we concluded that it was useless for our approach.

4.1 Distal Parts and Head movement - S_8; S_6, S_7

For representing the movement of the distal parts and the head, we considered the kinetic energy of the marker over the head and those related to the four distal parts: right wrist, left wrist, right ankle, left ankle. For doing that, we took from the TSV file the specific marker labels:

- Left wrist = LPLM
- Right wrist = RPLM
- Left ankle = LHEL
- Right ankle = RHEL
- Head = ARIEL

In Figure 12 you can see the EyesWeb patch used for extracting the kinetic energy, this patch is used for all the markers listed before, by changing in the block employed for taking the 3D values of the markers, the marker's labels as input.

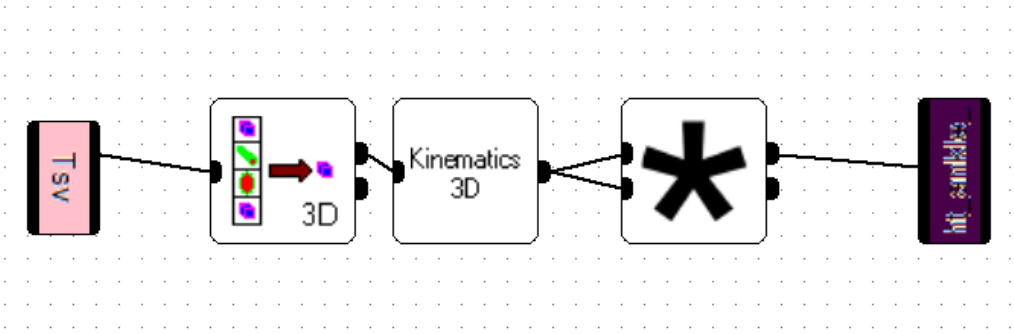

Figure 12: EyesWeb patch for extracting the kinetic energy from a marker

From the TSV all the markers values have been taken (pink block on the left), then the specific marker was chosen by using the 3D block, the Kinematics block extract the velocity of the markers along the time series, and then

thanks to the operator block (the fourth of the series), the square values have been calculated for having a good approximation of the kinetic energy by following the formula KE $\approx v^2$ (v stands for velocity). The last block store the results in a variable (purple block).

4.2 Point Density for crouched position - S_4, S_5

For measuring how much the dancer was crouched or extended we evaluated the *Point Density*. The concept behind this definition is the following: the more the distance among the different markers is higher on average (the dancer is extended), the more the value is higher, and vice versa (for the crouched position).
This value is also extracted by using EyesWeb XMI (Figure 13).

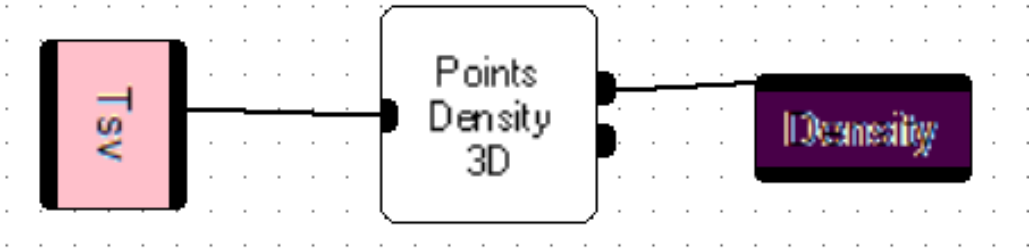

Figure 13: EyesWeb patch for extracting the Point Density

4.3 Repetitiveness - S_2, S_3

In order to have a measurement on how much the dancer makes repetitive actions we took in consideration the *Repetitiveness* index of the Point Density, and kinetic energy of the distal parts.
Since we considered this feature as a mid level feature (we built it over low level features), it will be explained in the details in chapter 5.1.3.

4.4 Stage dancer position - S_1

For the S_1 salience we thought as first to use a measure of general kinetic energy (Figure 14), because if a dancer moves toward the stage, of course the global kinetic energy, which is the sum of all the kinetic energies of all the markers, will increase, further a measurement of this value is enhanced by the kinetic energy of the ankles, because when the dancer is moving the kinetic energy of the ankles increase.

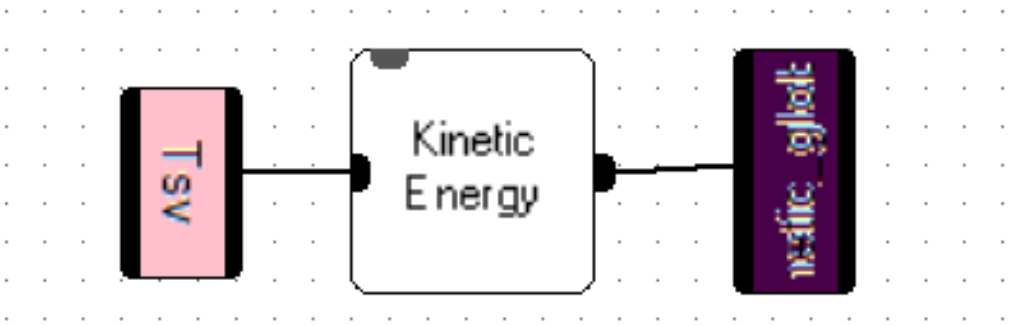

Figure 14: EyesWeb patch for extracting the Global kinetic energy.

4.5 Feature Vector

All the features stored in the EyesWeb XMI variables have been collected in a text file (Figure 15), hence, for each frame of the *take*, a row vector of low level features is build as follows: F=[f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8].
According to the discussion made before, the several features are the following:

- f1 = Global kinetic energy.
- f2 = Head kinetic energy.
- f3 = Point Density.

- f4 = left wrist kinetic energy.
- f5 = right wrist kinetic energy.
- f6 = left ankle kinetic energy.
- f7 = right ankle kinetic energy.

The text file has a quantity of rows equal to the quantity of frame of the specific *take* and 8 columns, one for each feature.

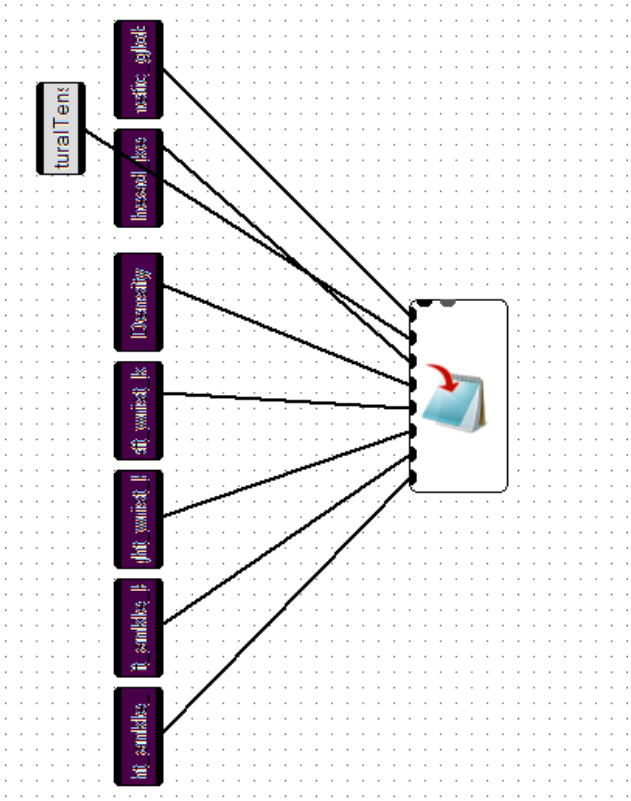

Figure 15: Saving the variables
which store the features values
over the frames on a text file (EyesWeb).

5 Supervised ML approach

Among all the existing methodologies for detecting a salience, we thought to address the problem by using a Machine Learning (ML) tool, of course one may think to use a CPD methodology or more in general, a statistic oriented algorithm, but since we had the possibilities to make a Ground Truth, and hence to use a ML algorithm which exploits its information (*Supervised*), we go throw that path.
Machine Learning is a branch of artificial intelligence focused on developing algorithms and models that allow computers to learn from data and make predictions or decisions without being explicitly programmed. It involves the utilization of statistical techniques to enable machines to improve their performance on a task through experience. In essence, Machine Learning enables computers to recognize patterns, infer insights, and adapt their behavior based on the data they are exposed to, thereby enabling them to solve complex problems and make smart decisions autonomously.
The next discussion is a brief introduction to ML technologies, hence analytical considerations behind the concept exposed will be omitted.

In general, the goal of a machine algorithm is to minimize a function which express the distance from the real output $y \in \boldsymbol{y}$, $\boldsymbol{y}$ vector of outputs, with the estimated one $\hat{y}$.
For doing that, it learns patterns within a set of samples, which are called *Training Set*. During the *training phase* a machine learning algorithm learn which are the best *hyperparameters* (θ) and the best vector of weights $\boldsymbol{\omega}$ that minimize the generic function:

$$\mathrm{L}(y,\hat{y}).$$

And the model as well,

$$\mathrm{M}(\mathrm{L}(y,\hat{y}),\theta).$$

with L a function which express the error (distance) between y and $\hat{y}$, $\hat{y} = \mathrm{f}(\mathbf{X},\boldsymbol{\omega})$, $\mathbf{X}$ set of samples and $\boldsymbol{\omega}$ vector of weights.
Each sample can be viewed as a set of values belonging attributes called *features*.

For making an example, by imagining to use a ML model for predicting which is the age ($\boldsymbol{y}$) of a specific person, there can be defined a sample as a set of information about a person, for instance: height, weight, gender, etc.
Thus all these information are the features which can represent a person, and "80kg", "1.7 meters", "Female", etc. Is an instance of the set of features, and hence a *sample*.
So, having **d** number of features, and **n** number of samples, $X^{n,d}$ is our *Dataset*.
A feature can be in general categorical, or ordinal, in the first case values that have not a concept of distance among them, and hence that cannot be ordered (e.g. the gender in the example, or by remaining on a human dataset, the color of eyes), the second case, instead, has values which can be ordered (e.g. in the example, the weight or the height).
For how concern the output $\boldsymbol{y}$, it can be ordinal ($\boldsymbol{y} \in \mathbb{R}$), or it can be categorical as well (hence if $\mathbf{C}$ is a set of categories, then $\boldsymbol{y} \in \mathbf{C}$).
If $\boldsymbol{y}$ is categorical, then we talk about a *Classification* problem, if $|\mathbf{C}|$=2, we can call it *Binary Classification Problem*.
In this case there exists a lot of loss function (L) which can be used, among them, one of the most employed is the Categorical Cross-Entropy Loss:

$$\text{CCEL} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} \log(\hat{y}_{i,j}).$$

Where N is the number of samples.
If $\boldsymbol{y}$ is ordinal, we address the problem as a *Regression Problem*, since here there is a concept of distance between the predicted output and the actual one.
There can be employed for instance one of the following Loss functions:

- Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2.$$

- Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|.$$

- Mean Squared Logarithmic Error (MAE):

$$\text{MSLE} = \frac{1}{N} \sum_{i=1}^{N} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2.$$

If $\boldsymbol{y}$ is known, the algorithm used will be in the context of the so called *Supervised* algorithms, if $\boldsymbol{y}$ instead is unknown, the algorithm will be *Unsupervised*, in that last situation, the model will try to divide the samples in group (classes) and creating boundaries by observing the data itself.
The function f($\mathbf{X}$,$\boldsymbol{\omega}$) is built during the training phase, given the set of samples $\mathbf{X}$ (from which it tries to learn the patterns), it sets the weights $\boldsymbol{\omega}$ associated to the several features (it is a measure on which are the most important features for the sake of prediction for the specific dataset).
The *hyperparameters* θ are useful for avoiding the so called *Over-fitting*, this issue comes out when the function f fits too well the samples in the training set and then it doesn't generalize properly the dataset, intuitively this phenomena is recognized when the model predicts well the samples in the neighbor (with respect to the space of features) to those used in the training set, but it misses the classification with samples which are different.
For trying to find a properly good level of generalization (and so, if it's necessary reduce the complexity of function f), a phase called cross-validation is computed, the dataset is divided in two different sets, one for learning the model and one for testing the prediction quality for each "instantiation" of *hyperparameters* θ, once θ are found, the best vector $\boldsymbol{\omega}$ is defined as well and the function is built as follows:
Let $\boldsymbol{\omega}^*$ the best weights for the specific dataset, the best function is

$$f^* = f(\mathbf{X}, \boldsymbol{\omega}^*).$$

Below an example of function f:

$$f = \boldsymbol{\omega}^T X + \mathrm{b}$$

b is a bias (it can be 0). Hence a Machine Learning model can be:

$$M = \|\boldsymbol{\omega}^T \mathbf{X} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\omega}\|_2^2$$

This is the Regularized least square (RLS), because uses a least square as loss function, and it uses a regularizer (for handling the level of complexity) which is the term $\lambda\|\boldsymbol{\omega}\|_2^2$, λ is the hyperparameter (θ) and the subscript "2" in both therms stands for the norm L_2. This is a model for regression problem, since f is linear it is used for problem which are not too complex.
The best model is given by searching $\boldsymbol{\omega}$ vector which minimize M by considering the right level of generalization (λ).

5.1 Data

In Our Context Data is given by Motion Capture (MOCAP) technologies, and stored in Tab-Separated Values (TSV) files (Chapter 2.2).
However, not all the *takes* were useful for our goal, so, before proceeding in cleaning the data we had to select them.
There were three different *dancers*: Cora Gasparotti, Muriel Romero and Marianne Gubri.
Each dancer has a specific number of *takes*, a *take* is a short video (50 fps, max 3 minutes) in which the dancer performed precise tasks for highlighting some specific movement, then by thinking at our salience definition, we have to do a selection.

For Cora Gasparotti 5 *takes* were chosen:

- t001: this *take* has captured our interest because she performed at the beginning a strike of fast and angular gesture, and on the second part of the video she was more smooth in the movement, then we had a lot of salience in the first part, and good frames which represent properly the non salience, in the second half.
  This *take* has been annotated till 85 seconds (4250 frames).

- t004 (2019/08/08) - Even for this *take*, the gestures were very frenetic and angular on a first part, then the gesture has became less frenetic, till to became completely smooth in the end of the video.
  Annotated 43 seconds (2150 frames).

- t004 (2019/05/22) - This *take* was easy to segment, hence all the salience were easy to see.
  Annotated for 85 seconds (4250 frames).

- t005 - We thought that this was one of the most valuable *take*, all the movement were easy to segment, majorly for annotating the repetitiveness (S_2, S_3). Indeed this video was the one which perform worst in Leave One take Out (LOTO) with a f1_score of 0.66 (Chapter 5.4.2) with respect other videos, because it was used into test set and not for training.
  Annotated for 85 seconds (4250 frames).

- t015 - Also for this video, it was easy to segment it was a good material for all the kind salience defined, then we used it.
  Annotated for 45 seconds (2250 frames).

For how concern Marianne Gubri, the following 12 takes were chosen:

- t007 - This video was good because salience like when the dancer was crouched or extended (S_4, S_5) were identified clearly, good also for S_8 (distal points movement).
  Annotated for 62 seconds (3100 frames).

- t008 - This *take* was easy to segment because movement were divided in time properly, useful for all the salience.
  Annotated for 39 seconds (1950 frames).

- t010 - Very smooth movement and easy to segment, good extension of the body for pointing out S_4, S_5 salience.
  This *take* has been annotated till 36 seconds (1800 frames).

- t018 - As for the previous one, even this *take* was useful for annotating clearly the salience, since the movement were good defined.
  Annotated for 50 seconds (2500 frames).

- t019 - This *take* was full of salience, and easy to segment.
  Annotated till 50 seconds (2500 frames).

- t024 - A video which alternated long static segment of equilibrium (non salience), with brief segments in which there were few salience very highlighted, in those frames the dancer had lost her equilibrium.
  Annotated for 69 seconds (3450 frames).

- t026 - It express a huge quantity of salience well defined.
  Annotated for 23 seconds (1150 frames).

- t041 - Also for this *take* there were a lot of salience, less then t024 or t026.
  Annotated for 31 seconds (1500 frames).

- t042 - It is similar to t024, but in this case, when a salience occurred was more related to an interruption of extended slow movement with an angular one, instead of a loss of equilibrium.
  Annotated for 73 seconds (3650 frames).

- t043 - This *take* had all the salience in its frames, not a huge quantity and not defined perfectly as other videos, but good enough.
  Annotated for 114 seconds (5700 frames).

- t047 - Full of salience well defined, the dancer did dynamic movement through all the *take* duration.
  Annotated for 20 seconds (1000 frame).

- t048 - Similar to t047, but with less salience.
  Annotated for 29 seconds (1450 frames).

For finishing, Muriel Romero's *takes* we had selected the following:

- t018 - This *take* had a large quantity of salience of almost all kind defined in Chapter 3.
  Annotated for 58 seconds (2900 frames).

- t026 - All kind of salience has been detected within this video, even the repetitiveness, it was a bit difficult to segment because of the large quantity of salience
  Annotated for 145 seconds (7250 frames).

- t027 - Full of salience of all kind, difficult to making segmentation (as for t026 or t018). Annotated for 98 seconds (4900 frames).

- t030 - Very good for pointing out the salience for crouched and extended position (S_4, $S - 5$).
  Annotated till 81 seconds (4000 frames).

Hence we have approximately 66000 frames annotated, and according to the Ground Truth taken, only 641 were annotated as salience, thus we can say the dataset is extremely unbalanced.
As we previously said (In chapter 3), the choice of intending the salience even in a more granular point of view was useful for having an higher quantity of salience. In next chapters it will be explained also how this problem has been solved.

5.1.1 Data Cleaning

Before extracting from EyesWeb XMI the low level features by using the patch described in chapter 4, we had to deal with the problem of missing data (NaN), indeed, a lot of videos (*takes*) has been discarded because of the massive quantity of missing data, this phenomena is related to a wrong reception of the signal coming from the markers in the dancer's body (Chapter 2.2), often because markers are not visible from Qualisys Cameras [13].
After having discarded all the *takes* with too much Not a Number (NaN), we had to choose the policy for replacing them on the selected videos (which had a reasonable number of NaN).
There exists a lot of method for replacing the missing values, among them, the following [8]:

1. Mean/Median/Mode Imputation: Replace missing values with the mean, median, or mode of the non-missing values in the respective column. This is a simple and commonly used method but may not be suitable for all types of data.

2. Forward Fill or Backward Fill: For time-series data, fill missing values with the most recent non-missing value (forward fill) or the next non-missing value (backward fill).

3. Interpolation: Interpolate missing values based on the values of neighboring data points. Common interpolation methods include linear interpolation, polynomial interpolation, and spline interpolation.

4. Using Predictive Models: Train a machine learning model to predict missing values based on other features in the dataset. This approach can be effective but requires more computational resources and may introduce bias if the predictive model is not well-trained.

5. K-Nearest Neighbors (KNN) Imputation: Replace missing values with the average of the nearest neighbors' values. This method considers the similarity between data points and is suitable for datasets with continuous features.

6. Delete Rows or Columns: If missing values are too numerous or occur in specific rows or columns, you may choose to delete those rows

or columns altogether. However, this approach should be used with caution as it may lead to loss of valuable information.

7. Domain-specific Methods: In some cases, domain-specific knowledge can help in imputing missing values. For example, in financial data, missing values in certain columns could be replaced with zeros if it's reasonable to assume that missing values represent the absence of transactions.

For our scenario, since our data are continuous in time, and since NaN_s often is a multiple occurrence (when markers are covered, multiple consecutive frames have been affected by NaN), we can use, for having a good approximation of the true value of the marker's position, a forward fill or a backward fill, or we can use an interpolation between the values before the missing values and after them.
Then we did a forward fill for the markers which starts from the frame 0 with NaN or a backward fill for missing data on the last frame of the specific *take*.
For missing values which are in the middle of known values, we rather opted for a linear interpolation, by proceeding as follows: given a column (marker's axis position) we took the value known before the first NaN, let's say "fkb", and we took the first known after the last NaN, "fka", then we counted the quantity of unknown from the "fkb" to the "fka" indexes, we call it n_nan, then the computed expression for replacing the specific NaN value is:

$$\begin{gathered} val\_diff = fka - fkb \\ incr = val\_diff/n\_nan \\ data[z, j] = (round(fka + incr * (z - i + 1)) \end{gathered}$$

where i is the index of first NaN, z goes from i to the index of the last NaN and j is the column related to the specific axis (x,y,z) of the specific marker (round is a function for making the rounding, for instance to the third decimal place).
The cleaning from missing values can be improved for future utilization by using an interpolation which takes in account a group of frames after and before, instead of looking just a frame (as linear interpolation does).

Another algorithm can consider the a spatial neighbor, instead of looking the frames before or after, it can look at the values taken by markers which are spatially closest to the missing one (KNN) and, after having evaluating

which are the closest markers and the values belonging to them, an estimation of the missing value can be computed.

5.1.2 Sliding Windows

The concept of sliding windows introduced in Chapter 2 is useful for dealing with problem related to stream data over time.
In general, a sliding windows (Figure 16) is an object which takes in consideration a set of *windows* separated by a specific *step*, a *window* has a specific dimension equal to the quantity of data within it.
In the case of video frames, a *window* is as a set of frames belonging the whole video. Thus the sliding windows concept is related to the shifting in the time of that *window*.
From each iteration the *window* is shifted by a specific *step* $\boldsymbol{\tau}$, this *step* can be viewed as a measure of definition, because if it is unitary (1 frame), the maximum definition is achieved since exists a *window* over each consecutive set of frames **n** (with **n** fixed length of the *window*), instead if the *step* is higher, not all the set of consecutive frames can be covered by a *window*, considering the whole video (but the computation is faster).
If $\boldsymbol{\tau}$ is lower then **n**, the phenomena of *overlapping* occurs, indeed the consecutive window cover the previous one for a number of frames equals to **n**-$\boldsymbol{\tau}$.

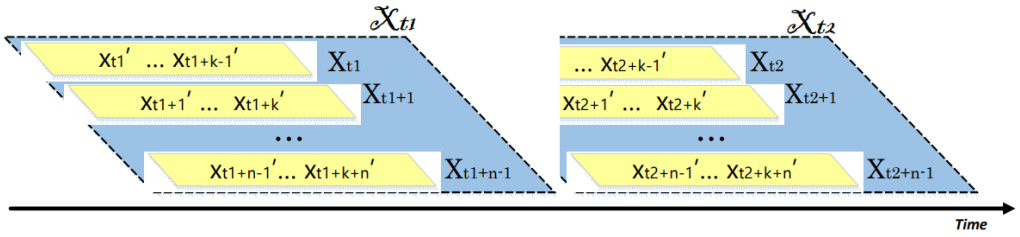

Figure 16: Generic sliding windows [2].

Dealing with the problem of salience detection by using a machine learning method is sometimes difficult because classic machine learning algorithms doesn't learn the information about the *time* relationship among the samples, because each sample is treated separately from the others.
Several approaches (CPD methods) discard machine learning models because of that issue.
Thus, considering the sample as the information conveyed over a multiple set of data is the main glint when using machine learning models.
Hence we needed to define a Sliding window prototype over our context and considering in some way the specific window as our sample, so each window can be labeled as a salience, or not.
Since the concept of salience is related in someway to a specific frame, and not to a set of them, the window, despite is referred to a group of frames, should be referred to just one of them.
A first approach can be the one often used by looking in the future, thus the window starts from frame i and goes to frame $i+j$, and conceptually that window is referred to frame i, but conveys information over j frames. Then a salience is detected if the window i has some statistical values which respect some behaviour, often comparing those values with a threshold (e.g. RuLsif [6]).
However, this approach gave us bad results, thus we had the idea of rebuilding the window for having a better representation of the salience's neighbor.
Conceptually our window is designed for having information about what is happened before, and what it will happen later.
Practically the window i referred to frame i was built as following: we define window's length as $n+1$, $k=n/2$ frames.
Hence we consider k frames before the i-th frame, and k frames after the i-th frame.
So the key added value is that now we had information about what comes before and what comes next a specific frame i and the features will be evaluated by considering a concept of distance between the k values before, and the k values after.
This is our first representation of sample, the whole dataset is hence divided in windows with step $\boldsymbol{\tau}=1$ (Sliding windows).
Another Positive side effect of this representation, is that since we look in the future only k values and not n values anymore, the hypothetical delay in a online context will be halved (without considering computational delays).
Finally we had to choose how to set n, and hence k.

If we consider a vector of values:

$$\mathrm{D} = [d_0, d_1 ... d_k ... d_{19}, d_{20}]$$

Where each value d_k is the average of the distances between the salience S_j and S_{j-1} of the specific take k. If we compute the mean value of that vector we obtain:

$$\mathrm{mean(D)} \approx 120 \text{ frames}$$

Hence if we took window which was lower then 120 we risked to not store enough frame for representing the dynamic of the system. Thus for us it was a lower bound.
But more frame we took, and more a better representation of the system we could obtain, then we set n=150, we could have choose more, but since the results were already satisfying, and the final goal of our thesis is to put the basis of an algorithm useful online, we didn't enlarge more the window, with the goal of avoiding further delay, which is, in our definition, k frames (75, hence 1,5 seconds).

5.1.3 Mid level features

Since ideally our window was composed of two sub-windows of dimension k, let's say, w_1 and w_2, and the frame to classify lies between them, the mid level features associated to the specific frame i-th was a concept of distance from w_1 to w_2.
For each frame i we had 7 distinct low level features ($f \in \mathbf{F}$) as previously discussed in chapter 4.
Below the list:

- f1 = Global kinetic energy.
- f3 = Head kinetic energy.
- f4 = Point Density.
- f5 = left wrist kinetic energy.
- f6 = right wrist kinetic energy.
- f7 = left ankle kinetic energy.

- f8 = right ankle kinetic energy.

Hence a sub window w_k can be viewed as a matrix which has in its column the several low level features $f \in \mathbf{F}$, and in its rows, the different frames $i \in \mathbf{N}$ (with $\mathbf{N}$ number of frames).
Since each column can be viewed as a signal over time $x(t)$, where t is a discrete index which goes from 0 to k, we can define $x_{j,1}$ and $x_{j,2}$ as the vectors containing data of column j of sub-windows w_1 and w_2 respectively, hence over them the following mid level features have been evaluated [4][5]:

1. Mean:

$$\text{mean1}=\text{mean}(x_{j,1}),$$

$$\text{mean2}=\text{mean}(x_{j,2}),$$

$$\text{mean}=\text{abs}(\text{mean1-mean2}).$$

   where abs is the absolute value.

2. Variance:

$$\text{var1}=\text{var}(x_{j,1}),$$

$$\text{var2}=\text{var}(x_{j,2}),$$

$$\text{variance}=\text{abs}(\text{var1-var2}).$$

3. Median Absolute Deviation (MAD):

$$\text{med1}=\text{median}(x_{j,1}),$$

$$\text{MAD1}=\text{median}(\text{abs}(x_{j,1}\text{-med1})),$$

$$\text{med2}=\text{median}(x_{j,2}),$$

$$\text{MAD2}=\text{median}(\text{abs}(x_{j,2}\text{-med2})),$$

MAD=abs(MAD1-MAD2).

4. Maximum value:

$$\text{max1}=\max(x_{j,1}),$$

$$\text{max2}=\max(x_{j,2}),$$

$$\text{Maximum}=\text{abs}(\text{max1}-\text{max2}).$$

5. Minimum value:

$$\text{min1}=\min(x_{j,1}),$$

$$\text{min2}=\min(x_{j,2}),$$

$$\text{Minimum}=\text{abs}(\text{min1}-\text{min2}).$$

6. Signal Magnitude Area (SMA):

$$\text{magnitude1}=\text{sqrt}(\text{square}(x_{j,1})),$$

$$\text{SMA1}=\text{simps}(\text{magnitude1},\text{dx}=1),$$

$$\text{magnitude2}=\text{sqrt}(\text{square}(x_{j,2})),$$

$$\text{SMA2}=\text{simps}(\text{magnitude2},\text{dx}=1),$$

$$\text{SMA}=\text{abs}(\text{SMA1}-\text{SMA2}).$$

Where "sqrt" is the square root function, the function "square" compute the square value of each element of the array and the function "simps" compute a sum of the rectangles with in the first argument the heights and in the second argument the base (integral approximation).

7. Energy (Average sum of squares):

square1=square($x_{j,1}$),

sum1=sum(squares1),

mean1=sum1/square1.length,

square2=square($x_{j,2}$),

sum2=sum(squares2),

mean2=sum2/square2.length,

Energy=abs(mean1-mean2).

where square1(2).length is the length of the specific array.

8. Interquartile Range (IQR):

First_Quartile1=percentile($x_{j,1}$,25),

Third_Quartile1=percentile($x_{j,1}$,75),

IQR1=Third_Quartile1-First_Quartile1,

First_Quartile2=percentile($x_{j,2}$,25),

Third_Quartile2=percentile($x_{j,2}$,75),

IQR2=Third_Quartile2-First_Quartile2,

IQR=abs(mean1-mean2).

The "percentile" function calculates the value corresponding to a specified percentile (or percentage) in a given array of data. It takes an array of data ("a") and a percentile value or a sequence of percentiles ("q") as arguments.

9. Signal Entropy:

$$\text{entropy1=entropy(bincount}(x_{j,1})),$$

$$\text{entropy2=entropy(bincount}(x_{j,2})),$$

$$\text{Entropy=abs(entropy1-entropy2)}.$$

Where the function "Entropy" is a measurement of disorder (uncertainty) adapt the following formula:

$$\text{H(X)} = \text{-} \sum_{i=1}^{n} p_i \log_2(p_i).$$

And the function bincount is useful for counting how much instances of a specific occurrence there are in the array.

10. Correlation Coefficient between the two arrays:

$$\text{Corr\_coeff=corrcoeff}(x_{j,1},x_{j,2}).$$

11. Kurtosis signal:

$$\text{kurt1=kurtosis}(x_{j,1}),$$

$$\text{kurt2=kurtosis}(x_{j,2}),$$

$$\text{Kurtosis=abs(kurt1-kurt2)},$$

Where the kurtosis is a measurement of the *tailedness* or *peakedness* of a probability distribution.

12. Skewness signal:

$$\text{skew1}=\text{skew}(x_{j,1}),$$

$$\text{skew2}=\text{skew}(x_{j,2}),$$

$$\text{skewness}=\text{abs}(\text{skew1-skew2}),$$

Skewness is a statistical measure that quantifies the asymmetry of a probability distribution. It indicates whether the data points in a distribution are concentrated more on one side of the distribution's mean than on the other.

13. Frequency measurement: Then there are 2 other values which are evaluated by computing the Kurtosis and the Skewness of the signal's spectrum.
    Discrete Fourier Transform has been computed of both the signals ($x_{j,1}$,$x_{j,2}$), then only positive frequencies has been considered and over the two positive spectre the kurtosis index and the skewness index has been computed, for finishing the distances between the two vectors in terms of the two indexes has been evaluated by computing the absolute difference (as for point 11 and 12).
    Thus we had 2 other features (freq_kurtosis and freq_skewness).

These mid level features (14 in number) has been evaluated over all the columns (low level features), then we had in total 98 of them (14x7).
For finishing there are 5 other mid level features evaluated on the low level features from f_4 to f_8.
For having a measurement of the **repetitiveness**, and hence if the dancer moves to redundant and smooth movement to a more angular and disordered movement and vice versa, which is a pretty important salience event (S_2/S_3), we had thought to compute an index, *repetitiveness*.
This index consider two different indexes, the first one, the *peakness*, evaluates how much the max amplitude of the spectrum of the signal $x_{j,l}$ (with l∈[1,2]) is in someway higher, with respect to the other amplitudes associated to the other frequencies.
If there were more then one max amplitude on the positive spectrum of the signal, we have to evaluate the *regularity*, this index (the second one), check if those peaks belong to the fundamental frequencies of the first harmonic (k*f_0, where k∈[1,inf)), or instead is just noise because those highlighted

peaks belong to different frequencies. Thus, if the *peakness* is a quantitative measurement, the *regularity* is a qualitative boolean value. Hence, over a specific signal the Discrete Fourier Transform (DFT) has been computed, the positive spectrum has been considered, then the *peakness* index is evaluated, this index is multiplied for the regularity index, the result is the **repetitiveness** index.
Computation was the following:

Four_x=DFT(x),

pos_Four_x=(Four_x where frequencies f>0),

max_amplitude=max(pos_Four,T),

Where T is the tolerance of 5% of the max amplitude,

regularity=1,

if max_amplitude.length>1: regularity=regularity(max_amplitude.indexes),

Where max_amplitude.indexes is the set of frequencies related to the amplitude within max_amplitude array,

peakness=$\sum_{\substack{i=1 \\ i \neq j}}^{n}(pos\_Four\_x_j - pos\_Four\_x_i)$,

Where $pos\_Four\_x_j$ is the max amplitude and $pos\_Four\_x_i$ is the generic amplitude.

repetitiveness=peakness*regularity,

return repetitiveness.

For the function regularity(), the computation was the following:

distances=[], void list.

distances.append(max_amplitude.indexes[0]),

We had appended to the list f_0

for i in length(max_amplitude.indexes):
distances.append($max\_amplitude.indexes_{i+1} - max\_amplitude.indexes_i$),

Thus the array distances collect all the distances between all the frequencies related to the max amplitudes,

Maximum=max(distances),

Minimum=min(distances),

span=abs(Minimum-Maximun),

if span≠0: *return 0.1*,

else *return 1*,

It returns 0.1 (for us its noise) for attenuate the peakness in the multiplication.

If the repetitiveness is high it means that the movement of the dancer related to the specific low-level feature associated to the specific sub window of length k is more repetitive, periodic and smooth, otherwise it is more "noisy", angular and chaotic.
By supposing that this value (*repetitiveness* index) is computed over $x_{j,1}$ (*repetitiveness1*), it is compared with the same index over $x_{j,2}$ (*repetitiveness2*), if it is lower, then we did *repetitiveness1/repetitiveness2*, otherwise *repetitiveness2/repetitiveness1*.
The more this value is close to 0, the more the sub window w_1 is different from w_2 in terms of repetitiveness.
This ratio is our mid level feature, and we have 5 of them, because this is evaluated for the Point Density and for the features related to the kinetic energy of distal points.
For summarize, for each frame we had 98+5, so 103, mid level features.
The output of this step is a set of samples, one for each frame.

5.1.4 Normalization

For how concerns normalization of the data, we had computed two different kind of normalization, the first one before computing the mid level features, by removing the mean value of the specific sub window, for the specific low level feature before computing the Discrete Fourier Transform (DFT) ($x_{j,l,i}$-mean($x_{j,l,i}$), with $j \in \mathbf{F}$, l∈[1,2] and $i \in \mathbf{N}$ the specific window of dimension n related to the specific frame), such that we removed the constant component from the signal spectrum (at frequency 0), which is like applying a low pass filter over the signal;
the second normalization was done at posterior, indeed is a normalization over range:

$$x'_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j},$$

Where j is the index of the specific mid level feature, and i is related to the frame (sample).
So, all the values lie between 0 and 1.

5.2 Samples Creation

Since not all the windows in the dataset are useful for our goal, and since we still have a lot of zeros in the dataset with respect to ones, removing samples and adding neighbor information should be a strategy.
First, for defining meaningful samples, over a specific salience (defined in the frame labeled with a 1) we have labeled some samples before, and some samples after, with 1.
Intuitively within a certain amount of frames close to the salience, we can assume that the feature vector which defines a salience, should be close to them in the space of features.
That's because shifting a window of a single step means that the frame evaluated is the successive one ($\boldsymbol{\tau}$=1), and the two sub windows (w_1, w_2) are equal to the previous one for *k-1* frames. Thus features should be similar to the previous window.
Of course, if we put too much windows in the neighbor (after and before) equals to 1 we risks to not incorporate enough information (frames) of the previous one.

Then an upper limit of the quantity of frames to label with 1 is k (*n/2*, dimension of a sub window w_l where l∈[1,2] and n dimension of the window). Indeed in the limit case in which we label the window which has distance k before the salience with 1, we have w_1 which hasn't any frame of the w_1 belonging to the windows associated to the salience, and w_2 identical to the w_1 related to the salience. So between w_1 and w_2 there isn't a significant distance, and it determines whether is a salience or not, conceptually if the distance between w_1 and w_2 is zero in therms of mid level features, the frame in the middle it's not a salience anymore (and vice versa for the sub windows k frames after).
So We scroll all the salience in the *takes* from left to right (from before to after) and for the zeros, if the distance d between a salience s_i to its successor (s_{i+1}) is less then n, then we skip to the next pair of salience (s_{i+1}, s_{i+2}); otherwise we take the frame in the middle of the pair of salience and we set that sample to *-1*, and the *k/2* in the neighbor to *-1* as well (*k/2* before and *k/2* after).
For the ones, if the distance d between s_i and its successor s_{i+1} is less then n, then we label *d/4* windows before s_i with *1*, otherwise we label *k/2* windows with *1* (identically for the windows before, but considering the distance between s_i and s_{i-1}).
We did that because if the distance between two salience is too short, the "non salience" in the middle doesn't have enough strong characteristic for representing a zero (too dynamic system and possible collision in the space of features), then we prefer to discard it for the sake of the classification prediction, and considering *d/4* or *k/2* for the windows to label with *1* for respecting the dynamic of the system.
Finally the situation is that we have 3 class in our dataset, label *1* which is related to the salience, label *-1* which is related to the not salience, and label *0*, which are the samples to discard.
After discarding the samples labeled with *0*, we have replaced the all the labels *-1* with *0*.
The number of samples **N** now is:

$$\mathbf{N} \approx 30500.$$

With approximately 24000 samples labeled with ones and 6500 samples labeled with zeros.

5.3 Model Selection

About the Machine Learning (ML) model, since we have labels associated to the samples in the training, we can use a *supervised* model, hence, we can choose if using a classical machine learning approach, or using a deep learning one.
Our dataset doesn't have so much samples, then we prefer to use machine learning, even because we already did the feature engineering step, and thus we don't need an algorithm able to find out features directly to the raw data like a Deep Neural Network (DNN) often does [7].
Among all the classical machine learning algorithm we prefer to use Random Forest.

5.3.1 Decision three and Random Forest

A decision tree is a tree-like model used in machine learning for classification and regression tasks. It recursively partitions the data based on feature values, creating a tree structure where each internal node represents a decision based on a feature, and each leaf node represents a class label or regression output.
At each node of the tree, the algorithm selects the feature that best splits the data into subsets that are as pure as possible in terms of the target variable (e.g., maximizing information gain or minimizing impurity). This process continues until a stopping criterion is met, such as reaching a maximum depth or when further splitting does not improve the purity significantly.
Decision trees are interpretable and can capture complex relationships between features and the target variable. However, they are prone to over-fitting, especially with noisy data or when the tree grows too deep.
Random Forest is an ensemble learning method that combines multiple decision trees to improve predictive performance and reduce over-fitting. Instead of relying on a single decision tree, Random Forest builds a "forest" of trees by training each tree on a random subset of the data and a random subset of features. During prediction, each tree "votes" for the final output, and the most popular class (in classification) or the average (in regression) is chosen as the prediction.
Random Forest inherits the interpretability of decision trees while mitigating

their tendency to over-fit. By aggregating the predictions of multiple trees, Random Forest tends to produce more robust and accurate results compared to individual decision trees. Additionally, it can handle large datasets with high dimensionality effectively.
The most important *hyperparameters* (θ) of the decision three are the following:

- **Criterion**: Determines the metric used to evaluate the quality of splits in the decision tree.
- **Max Depth**: Sets the maximum depth of the decision tree, limiting the number of levels it can have.
- **Min Samples Split**: Specifies the minimum number of samples required for a node to be split further during tree construction.
- **Min Samples Leaf**: Defines the minimum number of samples required to be at a leaf node.
- **Max Features**: Determines the maximum number of features to consider when searching for the best split.

For the Random forest the following *parameter* is added:

- **N_Estimators**: Number of trees in the forest.

Then it uses the same *hyperparameters* of the decision threes since its a collection of them. The parameter N_Estimators is not considered as *hyperparameter*, because more threes (estimators) there are in the model, better will be the prediction.
Of course the computational cost increase with the number threes.
The best Advantages of using a Random forest are the following [1] :

- Naturally handle both regression and (multiclass) classification.
- Are relatively fast to train and to predict.
- Grid search can be done only on one or two tuning parameters.
- Have a built-in estimate of generalization error.
- Can be used directly for high-dimensional problems.

- Can easily be implemented in parallel.
- Measures of variable importance.
- Visualization.
- Outlier detection.
- Unsupervised learning.

Hence since our dataset has 103 features, and we have the risk to over-fit the data since samples can be close each-other in the space of features, we thought that a model like random forest was a good choice.

5.3.2 Cross Validation

Cross Validation is a crucial step for any machine learning algorithm, because the choice of the best *hyperparameters*, mean choosing a good level of complexity, despite Random Forest has a good performance without setting any *hyperparamethers*, doing Cross Validation even for that is a good practice.
During Cross Validation phase, part of the dataset is used for training the model over certain parameters, and the rest is used for testing the model over that specific instantiation of θ, this procedure is repeated for each possible istantiation of the *hyperparameters*, the set of θ which performs better will be chosen as θ^* (best parameters).
Often, a K-fold split has been computed.
K corrisponds to the number of performed splits over the dataset (in training set and validation set), and the percentage of samples employed to the test as well (1/K).
Hence for searching the best parameters the following number of models will be compared: K*θ_m, where m is the total number of possible istantiation of θ. For our model we tried the following parameters:

- N_Estimators=1000.
- min_samples_leaf=[1,2].
- max_features=[0.1,0.2].

5.3.3 Downsampling

Since the number of samples labeled *1* were approximately 24000 and the samples with label *0* were approximately 6000, all the experiments, LOSO, LOTO, LODO (Chapter 5.4), were unbalanced as well, then we did a downsampling before computing models, by randomly removing a quantity of ones such that the number of zeros were equals to them.
This downsampling was needed for having good results even in the negative class (*0*).

5.4 Results

5.4.1 LOSO

5.4.2 LOTO

5.4.3 LODO

6 Conclusions

6.1 Future Researches

References

[1] D. Richard Cutler Adele Cutler and John R. Stevens. "Random Forests". In: *Ensemble Machine Learning: Methods and Applications* 1.5 (2011), pp. 157–176.

[2] Samaneh Aminikhanghahi and Diane J. Cook. "A Survey of Methods for Time Series Change Point Detection". In: *Springer-Verlag London* (2016).

[3] The Language Archive. URL: `https://archive.mpi.nl/tla/elan`.

[4] V. D'Amato et al. "The Importance of Multiple Temporal Scales in Motion Recognition: from Shallow to Deep Multi Scale Models". In: *IEEE International Joint Conference on Neural Networks (IJCNN)* (2022).

[5] V. D'Amato et al. "The Importance of Multiple Temporal Scales in Motion Recognition: when Shallow Model can Support Deep Multi Scale Models". In: *IEEE International Joint Conference on Neural Networks (IJCNN)* (2022).

[6] Ceccaldi E. "CEST: a Cognitive Event based Semi-automatic Technique for behavior segmentation". PhD thesis. Università di Genova, 2022.

[7] Courville A. Goodfellow I. Bengio Y. "Neural Networks and Deep Learning: A Textbook". In: *BM T. J. Watson Research Center International Business Machines* 1.1 (2018), pp. 1–12.

[8] Roderick J Little and Donald B. Rubin. *Statistical analysis with missing data.* John Wiley & Sons, 2019.

[9] R. Niewiadomski et al. "Automated Detection of Impulsive Movements in HCI". In: *Conference CHItaly* (2015).

[10] R. Niewiadomski et al. "Does embodied training improve the recognition of mid-level expressive movement qualities sonification?" In: *Multimodal User Interfaces* (2018).

[11] *Oxford Latin Dictionary.* Oxford: Clarendon Press, 1982.

[12] Casa Paganini. URL: `http://www.casapaganini.org/`.

[13] Qualisys. URL: `https://www.qualisys.com/`.

[14] *The Oxford English Dictionary.* Oxford: Clarendon Press, 1989.

[15] L. Wang et al. “Review of Classification Methods on Unbalanced Data Sets”. In: *IEEE Access* (2021).

[16] J.M. Zacks and K.M. Swallow. “Event segmentation”. In: *Current directions in psychological science* 16.2 (2007), pp. 80–84.