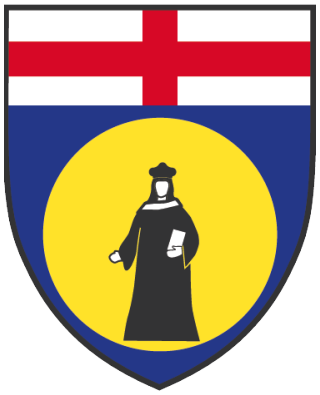
University of Genoa



Computer Engineering

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

Serafino Gabriele
Lo Luca

Advisors:
Volpe Gualtiero
Camurri Antonio

March 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, and 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.

The concept of salience is complex and has been explored in cognitive neurosciences, computer vision, human-computer interaction, and performing arts. In this work we focus on a simplified concept of salience, focusing on changes in the full-body movement quality of the dancer, from the perspective of an external observer.

We focus on repetitiveness and predictability of movement, e.g., a chaotic Vs repetitive movement, and on the emergence of sudden changes Vs smooth and continuous movements, e.g., an arm sudden stop followed by a new action by the same or another limb. We start with Motion Capture (MOCAP) data of full-body movement. An in-depth analysis of a movement dataset recorded at Casa Paganini led us to individuate and evaluate a set of relevant movement feature candidates to explain salience.

Then, supervised machine learning approaches were applied for the automated analysis of salience. For handling input features to the ML approach, we considered different temporal scales and automated analysis using overlapped sliding windows, such we can embed information from multiple frames in a unique sample. 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 the human full-body context in real-time.

Contents

List of Figures

List of Tables

Acronyms

3D Three Dimensional. 11, 16, 27, 28, 39

CPD Change Point Detection. 10, 12, 13

CUSUM Cumulative Sum. 12

DFT Discrete Fourier Transform. 48, 52

DNN Deep Neural Network. 54

EST Event Segmentation Theory. 17

EWMA Exponentially Weighted Moving Average. 12

GLR Generalized Likelyhood Ratio. 12, 13

LODO Leave One Dancer Out. 58, 60, 67

LOOCV Leave One Out Cross Validation. 60, 61

LOSO Leave One Saliency Out. 58, 60, 65, 67

LOTO Leave One Take Out. 36, 58, 65, 67

LTM Long Term Memory. 9

ML Machine Learning. 8, 11, 20, 21, 33–35, 54

MOCAP Motion Capture. 2, 15, 16, 20, 36

NaN Not a Number. 39, 40

PELT Pruned Exact Linear Time. 13

QTM Qualisys Track Manager. 11, 16

RuLSIF Relative unconstrained Least-Squares Importance Fitting. 13

STM Short Term Memory. 9

TSV Tab-Separated Values. 16, 18, 28, 36

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 motion capture techniques. Raw data collected from the dancers will be evaluated with a set of low-level features, and hence, by embedding information about multiple frames thanks to a sliding windows technique, mid-level features have been evaluated, and prediction of saliency is performed using a Machine Learning (ML) model.

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 [22][18]. So salience is something that catches our attention and is based on its definition and it exists in various contests. In this thesis we want to focus on the salience in human movements, in this introduction, we hope to give the reader a first idea of how this concept is intended by us.

A lot of work took place over time to define in the domain of the human movement, or more in general in the context of human attention, a specific meaning of salience.
A first thing that should be taken into consideration is the observer of salience.
If we think about the "attention" of a subject, the salience can be viewed as something interpreted in the environment by the subject which captures a specific event, for instance, if the subject is pointing something, a movement of the limbs toward a specific point of the environment, or a saccade (which are rapid eye movements that shift gaze from one point to another)[26], for example, Wang W., Sheng J., and other researchers made work for predicting a subject viewing fixation during dynamic scene [24]. Then, in that case, contextual information is useful for helping an external automatic agent in enhancing the user experience [20], because not all the points within the en-

vironment (context) are salient for the subject.
To collect those kinds of information a tool called "saliency map" is used, this object is useful for giving the agent the following information: topological information about the scenario, a local "distinctiveness" for each position rather than information about the visual features that make a given position distinct, and adaptability (since is motor-unspecific). This last concept is important in a multi-modal scenario [20][26].
However, in our analysis, the concept of salience is more related to a gesture, or a movement, performed by the subject which captures the attention of an external viewer.
This point of view can be considered the opposite of the one discussed before. A first approach for addressing the problem is by defining a concept of the rarity of several gestures that the actor does [14]. Of course, if a gesture is rare, it can be salient for a viewer. This approach can be employed in different context characteristics, for instance can be observed within a group of actors, and hence looking for those who capture the attention of the others [14], or more in general, those who capture the attention of an external viewer. Hence a salient movement can be pointed out over time (by looking at several frames of a specific actor), or over space, by looking if an actor, for instance, is faster [13] or performs different gestures concerning others. Thus, three different levels [12] of context information can be analyzed for understanding a salient behavior of a subject: instantaneous level (by looking the velocity direction of all the subjects within a frame), short-term level (by looking at 2-3 seconds before, for simulating a Short Term Memory (STM) [4] of subject's movement), or by using contextual information (an attention amplitude map [12]) alongside a Long Term Memory (LTM) [4].
Another important discussion can be done from the point of view of the salience's hierarchy [10], indeed different levels of salience can be analyzed. For instance, a higher level can be conceived by looking at the quantity of movement of the subject, because a salient action can appear with a greater movement. A middle level can be evaluated by taking the salient point of a specific action and finally at a lower level, an evaluation of how much each point has contributed to the action, for defining the most significant body's joint involved for the specific action which represents the salience.
Our context is characterized by a single subject (actor/dancer) who performs some dance movements, and the salience is defined from a more holistic level (by looking at the general kinetic energy of the body, information about his general dancing behavior, etc.) to a more granular salience level (by looking

the kinematics of limbs).
A more detailed definition of salience (for us) can be read in Chapter 3.

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 detect automatically anything critical without the necessity of human supervision.

For us understanding how to set up a system able to recognize, hopefully in real time, a salience in the movement, is an interesting challenge.
We think that this work can be useful for different environments: art, rehabilitation, sport, or the detection of psychological causes of movement.
Further, this project combines different disciplines, and hence it is very engaging.

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 Change Point Detection (CPD) methodologies and the starting point of our Thesis, that exploits the results of a PhD Thesis [8] and methodologies used by us for improving or extend the existing 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 which is the annotation used for labeling the samples.

Chapter 4: describes the low level features extracted from the raw movements data of the dancers. For low level features we describes the transfor-

mation 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 preprocessing of the data, continuing with the definition of mid level features, definition of samples, model selection, for arriving to the results over different level of generalization for testing 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

For dealing with the problem of salience detection, different approach can be checked, from change point detection algorithm to machine learning models [24], but since we talk about salience over time, a common approach is to running algorithms over data which embed in someway temporal information, for instance, Sliding Windows over time are considered (Chapter 5.1.2) before using models.
In the next discussion some of CPD algorithm used for dealing with the problem will mentioned, most of them uses sliding windows.

2.1 CPD algorithms

Among all the algorithms used in literature CPD are worth to be mentioned, below a list of the most commonly used.

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 [19]. 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 [19], Cora Gasparotti (Figure 1), Marianne Gubri (Figure 2) and Muriel Romero (Figure 3), 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 have 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 1:
Cora Gasparotti

Figure 2:
Marianne Gubri

Figure 3:
Muriel Romero

Raw data has been collected by using a Qualisys [21] 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 with 16 cameras was used (fs = 100Hz). (Figure 4)

Figure 4: 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 5)

Figure 5: Markers Qualisys

The MOCAP data was collected in a Tab-Separated Values (TSV) file by using Qualisys Track Manager (QTM) [21] 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 Ph.D. Thesis [8], in that discussion, the problem of salience detection, has been defined with some psychological approach related to the body behavior, 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 [25]. 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 [8] 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 start to point to something different.
- C4. Character Location: if the character moves on the stage, or, in our cases, if the dancer moves from one point to another.
- C6. Causes: These are the causes and appraisals 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 who are trained on EST [25] principles by using a software for segmentation: ELAN [3] (Figure 6).
This software allows to make the segmentation of videos (or an audio track) by defining some classes of event segments 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 at the Density of Chest Trajectory.

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

All these features have been extracted by using an application (patch) developed for EyesWeb XMI (Figure 7), 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 the Head is a measurement of how the dancer's head moves in a curvilinear trajectory, the higher the value, the more the movement follows 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 to a smaller region.

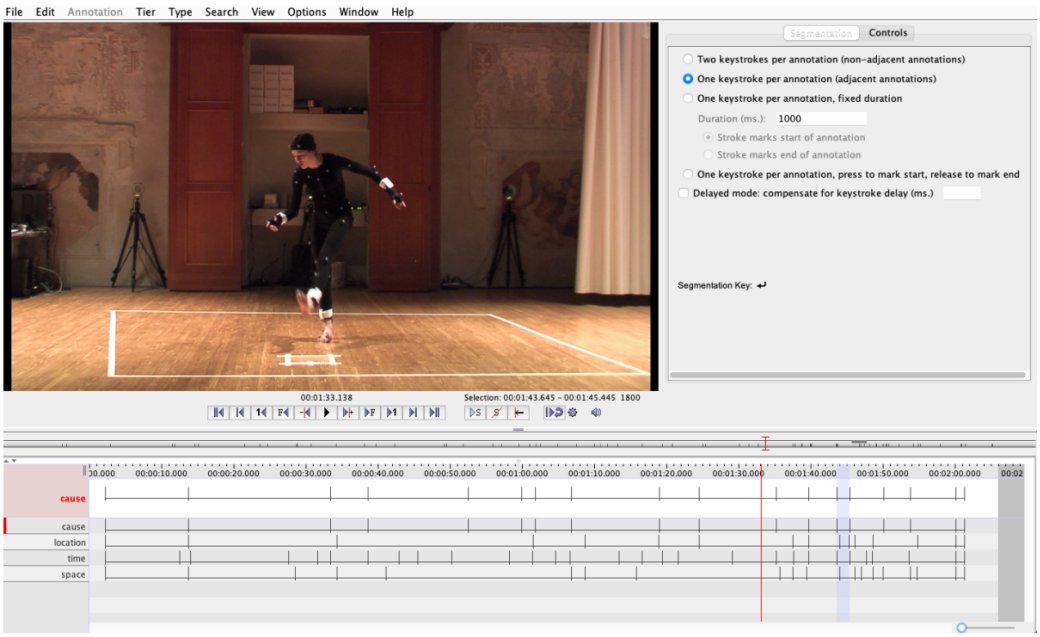


Figure 6: Annotation of the different salience with ELAN.

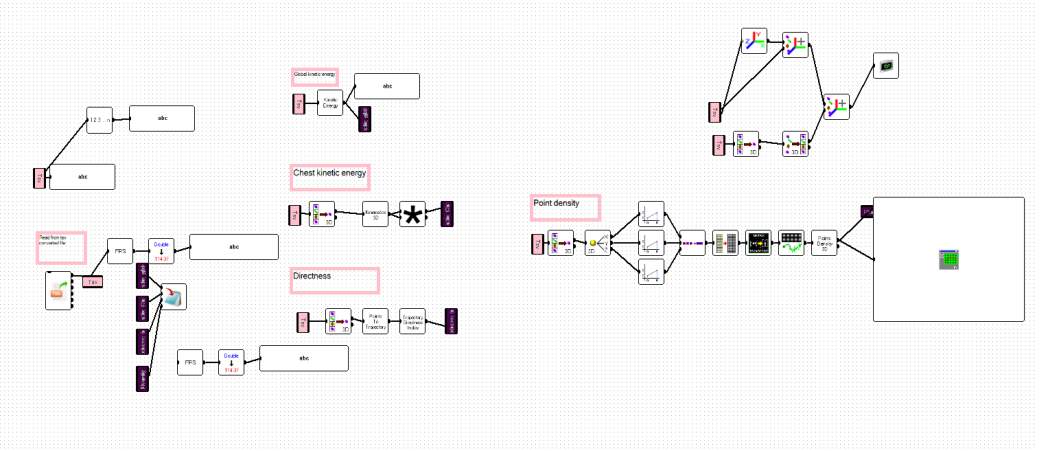


Figure 7: One of the patches implemented on EyesWeb for extracting the features.

2.3 Structure of workflow

In this chapter, it will be explained which is the workflow that we did during our study (figure 8).
From the raw data analysis 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 [19] (chapter 2.2).

2. Salience and Ground Truth: After reviewing the material provided in the previous step, we evaluated which is the better definition of salience in our context, because we had to take into 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 point 2 we had to choose wisely the *takes* which better represent the context because not all the videos were adapted 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 explored a machine learning approach.

7. Sliding Windows Template: For the specific model we built a template of sliding windows able to convey the right piece of information based on the used model (chapters 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: Based on the approach used parameters have been chosen, for ML cross-validation has been done (chapter 5.3).

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

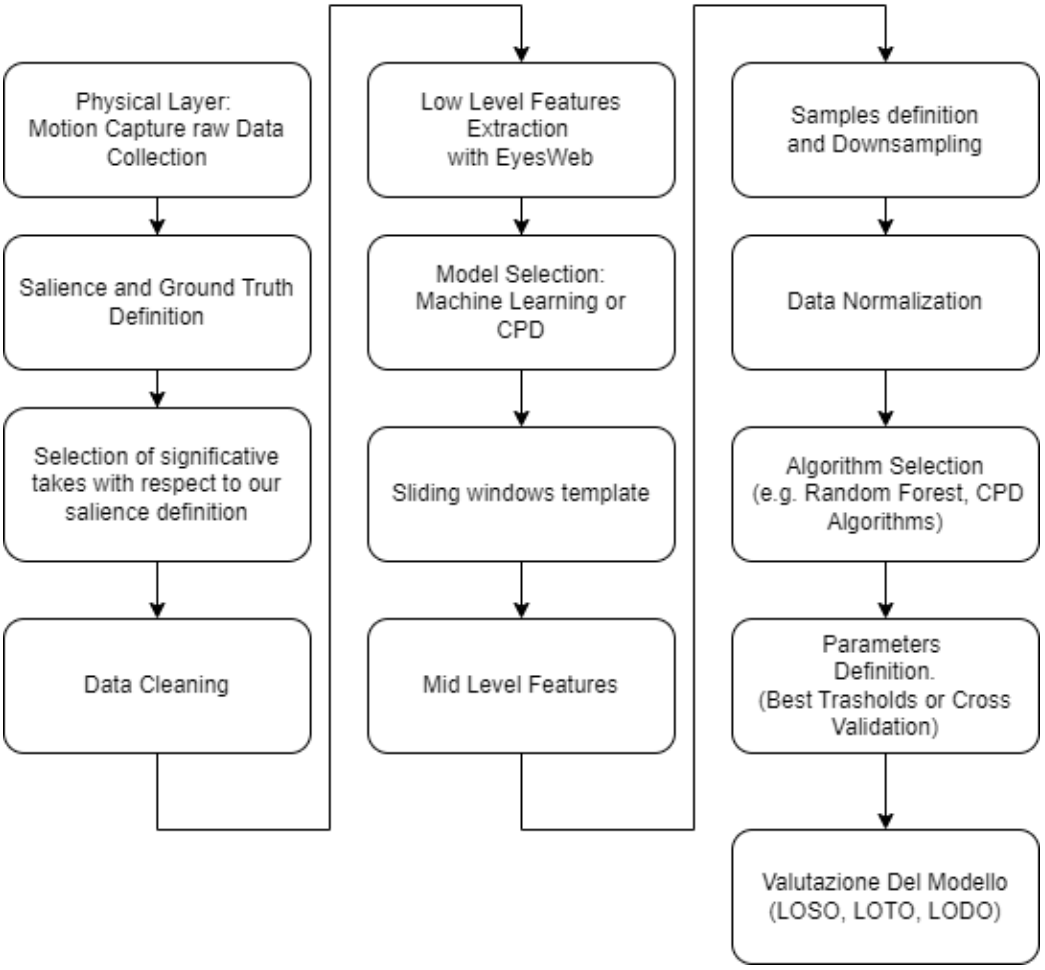


Figure 8: Workflow chart

3 Salience definition

When we started to approach this problem, one of the first challenges that we had to deal with, was to define what is a specific event that changes in some way the behavior of the dancer.
We address this event as a ***Salience***.
The salience in the context of movement can be viewed in different ways, from a 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 points of view, for instance, you can make a comparison between a movement that is more *impulsive* or more *sudden* [16], or you can look at the movement in terms of *Lightness* or *Fragility* [17].
The first issue that we have noticed is that, if we thought of the salience as something that is too much related to the causes, and the psychological aspect of human behavior, 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 at the movement of the dancer from the point of view of Kinesiology, so if there is a movement behavior that is similar to the observation of the whole set of frames that 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 terms 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 at the set of salience resulting from the step before, and since the quantity of frames labeled "salience" is significantly lower with respect to the frame which is labeled "non-salience", we became aware that for an algorithm of machine learning this scenario could be a problem because a unbalanced dataset is often hard to address [23]. To try to reduce the oddness between the two classes, we defined the concept of event as something that is more frequent, by using a more granular segmentation, hence visible changes in the movement such as an arm that 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 over the considerations made before (Figure 9). 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 detects 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 occurs 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 changes 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≠j).

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

to as a local movement.
During the phase of taking the Ground Truth, we did not make meaningful segments, we just considered the events occurrence, then to avoid the perceptual fusion problem (two different frames are perceived as one if the time between them is lower than 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 represents the timestamp in seconds (with centesimal precision) of a frame occurrence 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 (the floor function is for rounding down).

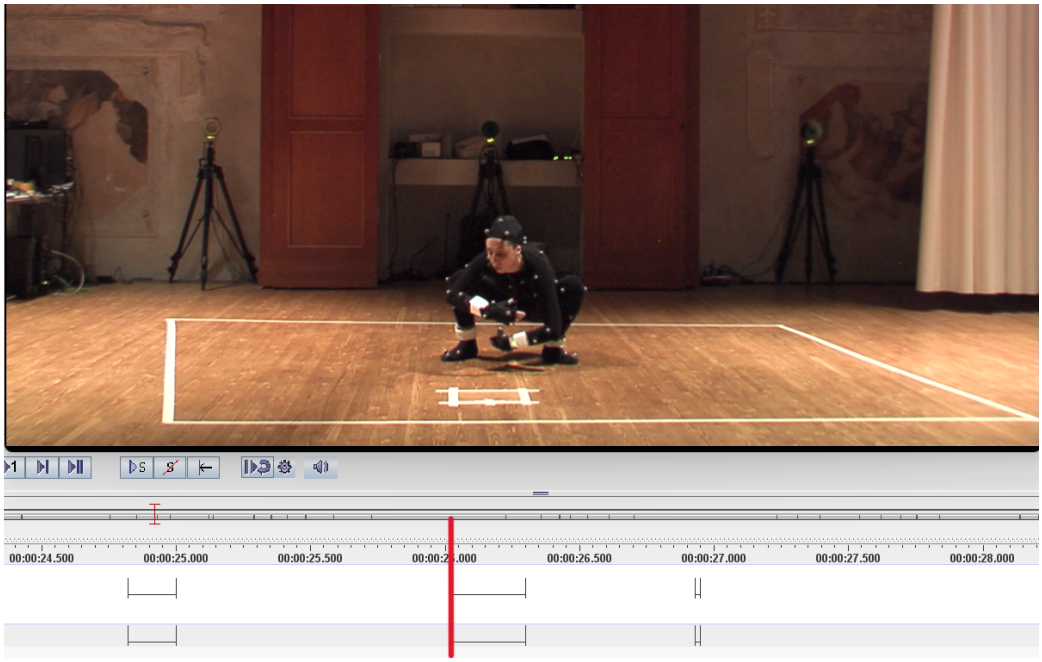


Figure 9:
The red line is over
the frame were
the salience is annotated (S_4).

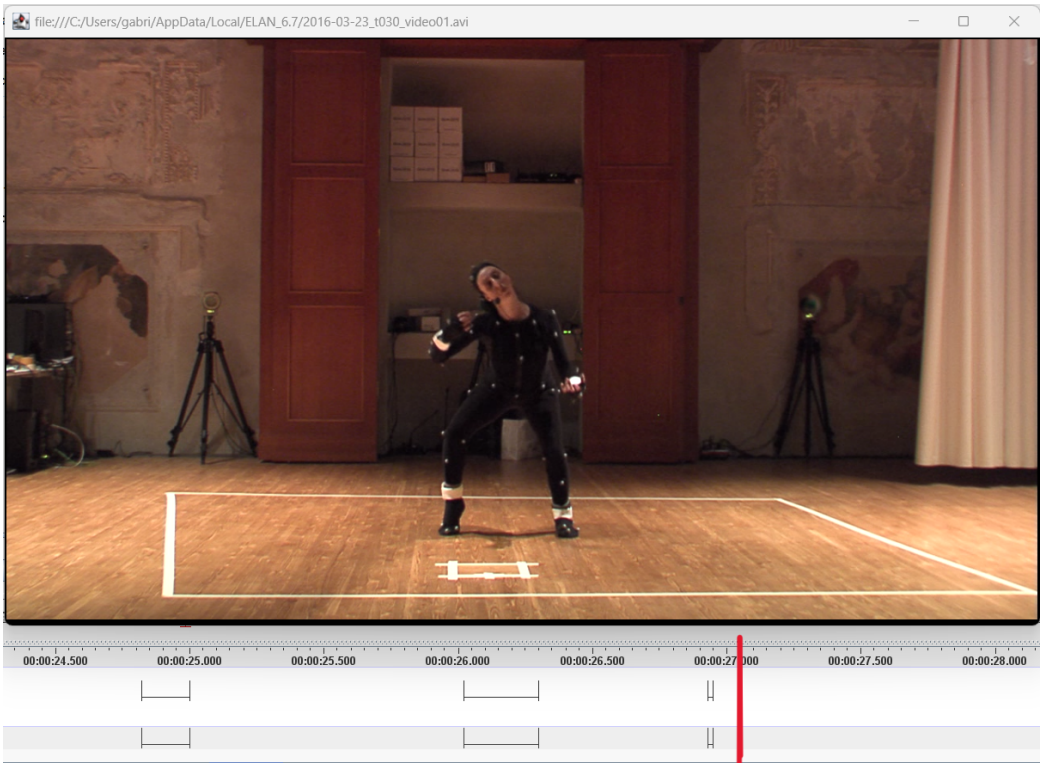


Figure 10:
After ≈ 50 frames (1 second)
with respect to the
salience in Figure 9
the dancer is extended.

Salience	Description
S_1	Changes in the position over the stage.
S_2, S_3	Moving from a periodic action to a chaotic one and vice versa.
S_4, S_5	Moving from a couched position to an extended one, and vice versa.
S_6, S_7	Changes on head's dynamic.
S_8	Changes in the movement of distal parts.

Table 1: saliences

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 with 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 at 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 of 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.
Postural tension is a measure of how much the dancer's body is twisted; for evaluating that, unit vectors of head, shoulder, trunk, and hips are compared in terms of how much they are parallel, the less they are parallel, the higher is tension.
In the beginning, we thought that combining the postural tension with other features 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, and left ankle. To do 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 11 you can see the EyesWeb patch used for extracting the kinetic energy, this patch is used for all the markers listed before, by changing the block employed for taking the 3D values of the markers, the marker's labels as input.

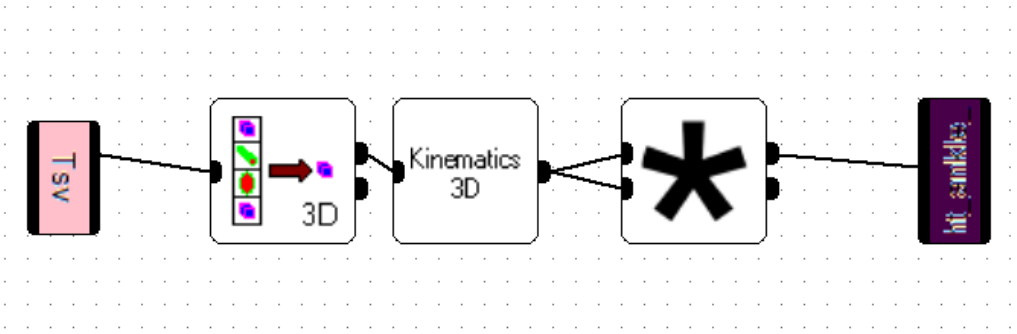


Figure 11: EyesWeb patch for extracting the kinetic energy from a marker

From the TSV all the marker values have been taken (pink block on the left), and then the specific marker was chosen by using the 3D block, the Kinematics block extracted 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 stores 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 12).

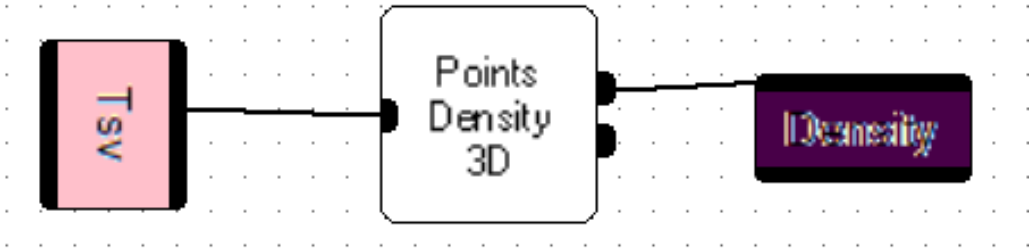


Figure 12: EyesWeb patch for extracting the Point Density

4.3 Repetitiveness - S_2, S_3

In order to have a measurement of how much the dancer makes repetitive actions we took into 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 at first to use a measure of general kinetic energy (Figure 13), 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 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.

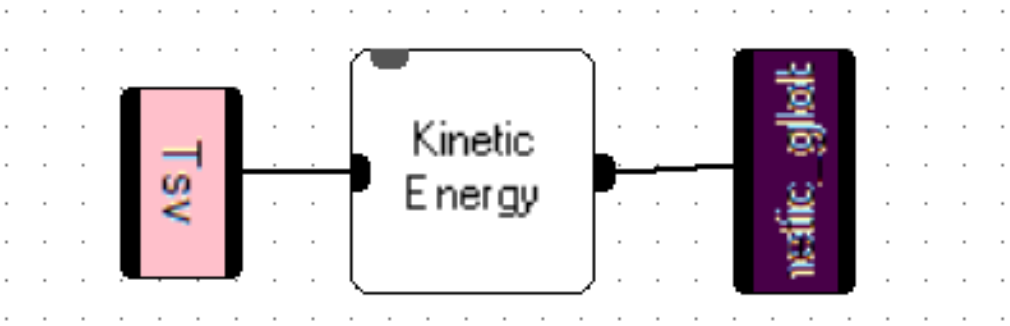


Figure 13: EyesWeb patch for extracting the Global kinetic energy.

4.5 Feature Vector

All the features stored in the EyesWeb XMI variables have been collected in a text file (Figure 14), hence, for each frame of the *take*, a row vector of low-level features is built 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.

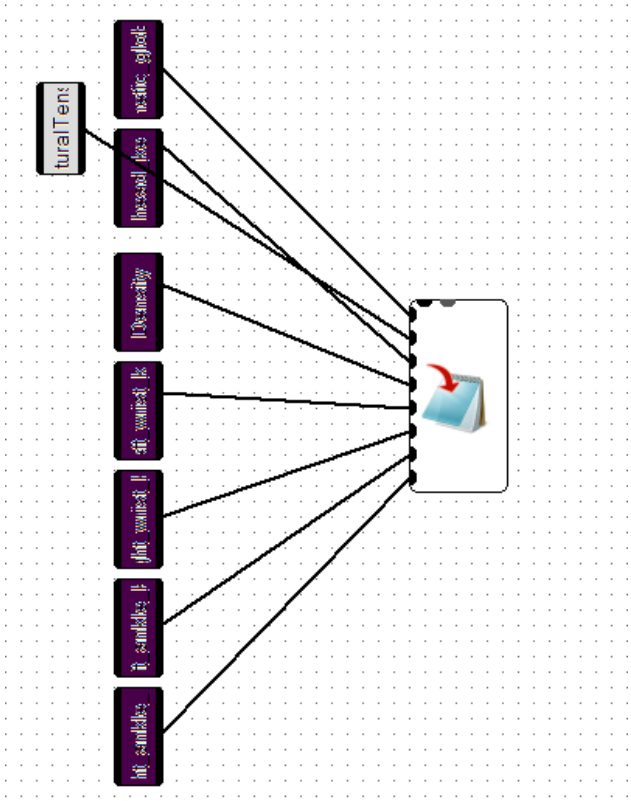


Figure 14: Saving the variables
which store the features values
over the frames on a text file (EyesWeb).

Features	Description
f_1	Global kinetic energy.
f_2	Head kinetic energy.
f_3	Point Density.
f_4	Left wrist kinetic energy.
f_5	Right wrist kinetic energy.
f_6	Left ankle kinetic energy.
f_7	Right wrist kinetic energy.

Table 2: Low Level features.

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 possibility to make a Ground Truth, and hence to use a ML algorithm which exploits its information (*Supervised*), we go through 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 mathematical and statistical demonstrations behind the concept exposed will be omitted.

In general, the goal of a machine learning algorithm is to minimize a function that expresses the distance from the real output $y \in \boldsymbol{y}$, $\boldsymbol{y}$ set of output vectors, 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:

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

The algorithm also learns a model,

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

with L a function which expresses 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 to attributes called

features.
For making an example, by imagining using a ML model for predicting which is the age ($\boldsymbol{y}$) of a specific person, we can define a sample as a set of information about a person, for instance: height, weight, gender, etc.
Thus all this information are the features that 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** features, and **n** samples, $X^{n,d}$ is our *Dataset*.
A feature can be in general categorical, or ordinal. In the first case, values do not have a concept of distance among them, and hence they cannot be ordered (e.g. the gender in the example, or by remaining on a human dataset, the color of eyes). In the second case, instead, values can be ordered (e.g. in the example, the weight or the height).
For what concerns 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 exist a lot of loss functions (L) that can be used; among them, one of the most employed is the Categorical Cross-Entropy Loss:

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

Where N is the number of samples and $p_{i,j}$ is the probability that the observation i belongs to class j.
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.
One of the following Loss functions can be employed:

- 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 into groups (classes) and create 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 the algorithm tries to learn the patterns). During training, the algorithm sets the weights $\boldsymbol{\omega}$ associated with the features (it is a measure of 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 to the dataset. Intuitively this phenomenon 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 that are different.
For trying to find a properly good level of generalization (and so, if it's necessary to reduce the complexity of function f), a phase called cross-validation is adopted. The dataset is divided into 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 $\boldsymbol{\omega}^*$ be the best weights for the specific dataset, the best function is

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

Below an example of function f:

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

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 the least square as a 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 terms stands for the norm L_2. This is a model for regression problems, since f is linear it is used for problems that are not too complex.
The best model is given by searching the $\boldsymbol{\omega}$ vector which minimizes 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 in the second part of the video she was smoother 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 to 85 seconds (4250 frames).

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

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

- t005 - We thought that this was one of the most valuable *takes*, all the movements were easy to segment, majorly for annotating the repetitiveness (S_2, S_3). Indeed this video was the one that performed worst in Leave One Take Out (LOTO) with an F1_score of 0.66 (Chapter 5.4.2) with respect to other videos when it was used in the test set and not for training.
  Annotated for 85 seconds (4250 frames).

- t015 - Also this video, was easy to segment and it contained good material for all the kinds of salience we defined. So we used it.
  Annotated for 45 seconds (2250 frames).

For what concerns Marianne Gubri, the following 12 takes were chosen:

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

- t008 - This *take* was easy to segment because movements were divided in time properly, useful for all kinds of 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 was clearly segmented.
  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 segments of equilibrium (absence of salience), with brief segments in which there were few saliences very highlighted, in those frames, the dancer had lost her equilibrium.
  Annotated for 69 seconds (3450 frames).

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

- t041 - Also for this *take* there were a lot of saliences, 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 it was more related to an interruption of extended slow movement with an angular one, rather than to a loss of equilibrium.
  Annotated for 73 seconds (3650 frames).

- t043 - This *take* contained all kinds of saliences, not a huge quantity and not defined perfectly as in other videos, but good enough.
  Annotated for 114 seconds (5700 frames).

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

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

For finishing, Muriel Romero's *takes*:

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

- t026 - All kinds of salience were detected within this video, even repetitiveness. The video 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 kinds, difficult to make segmentation (as for t026 or t018). Annotated for 98 seconds (4900 frames).

- t030 - Very good for pointing out the salience for a crouched and extended position (S_4, S_5).
  Annotated till 81 seconds (4000 frames).

Hence we have approximately 66000 frames, and according to the Ground Truth taken, only 641 were annotated as salient, thus we can say the dataset is extremely unbalanced.
As we previously said (In Chapter 3), the choice of intending salience even from a more granular point of view was useful for having a higher quantity of salient frames. The next chapters will explain how this problem was 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 them. This phenomenon is related to the impossibility of computing the 3D positions of some of the markers on the dancer's body (Chapter 2.2), often because markers are not visible from Qualisys Cameras [21].
After having discarded all the *takes* with too many 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 methods for replacing the missing values, among them, the following [11][5]:

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 commonly used, but not suitable for each scenario.

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 (e.g. linear interpolation or polynomial 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.

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 the loss of valuable information.

7. Domain-specific Methods: In some cases, domain-specific knowledge can help in imputing missing values. For example, in a specific domain.

For our scenario, since our data are continuous in time, and since NaN_s often is multiple occurrences (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 interpolation between the values before the missing values and after them.
Then we did a forward fill for the markers which start from 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 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 value after the last NaN, "fka", then we counted the number of unknown values from the "fkb" to the "fka" indexes, we call it n_nan, then the computed expression for replacing the specific NaN value is:

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

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 that takes into account a group of frames after and before, instead of looking just at one frame (as linear interpolation does).

Another approach consists of considering a spatial neighbor. Instead of looking at the frames before or after the missing value, it can look at the values taken by markers that are spatially closest to the missing one (KNN) and, after having evaluated 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 problems related to streaming data over time.
In general, sliding windows (Figure 15) is an object that takes into 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 a sequence of frames belonging to the video. Thus the sliding windows concept is related to the shifting over time of that *window*.
At 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 there exists a *window* over each consecutive set of **n** frames (with **n** fixed length of the *window*), instead of 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 than **n**, the phenomena of *overlapping* occurs, indeed the consecutive window covers the previous one for a number of frames equals to **n**-$\boldsymbol{\tau}$.

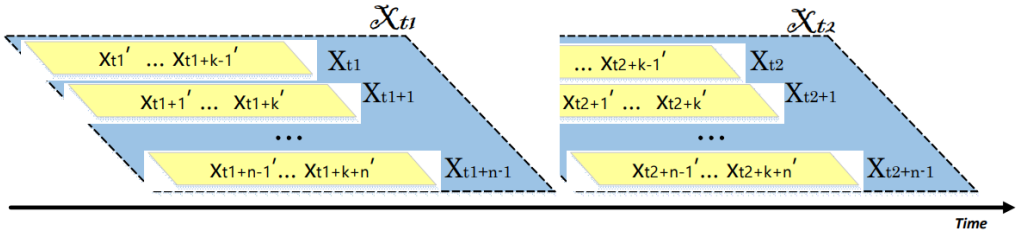


Figure 15: Generic sliding windows [2].

Dealing with the problem of salience detection by using a machine learning method is sometimes difficult because classic machine learning algorithms do not 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 consider 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 some way to a specific frame, and not to a set of them, the window, despite being 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 that respect some behavior, often comparing those values with a threshold (e.g. RuLsif [8]).
However, this approach gave us bad results, thus we had the idea of rebuilding the window to have a better representation of the salience's neighbor.
Conceptually our window is designed to have information about what has happened before, and what will happen later.
Practically the window i referred to frame i was built as follows: we set the window length to $n+1$ frames and we define $k=n/2$ 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 have 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 the 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 an 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 ($k \in [0,20]$). If we compute the mean value of that vector we obtain:

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

Hence if we took a window that was lower than 120 we risked not storing enough frames for representing the dynamic of the system. Thus for us, it was a lower bound.
But the more frames we took, and a better representation of the system we could obtain, then we set n=150. We could have chosen 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 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 with 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.
- 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.

Hence a sub window w_k can be viewed as a matrix which has in its column several low-level features $f \in \mathbf{F}$, and in its rows, the different frames i=1,..,$\mathbf{N}$ (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 that 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 [6][7]:

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:

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

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

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

5. Minimum value:

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

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

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

6. Signal Magnitude Area (SMA):

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

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

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

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

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

The function "simps" computes 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) according to the following formula:

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

And the function bincount is useful for counting how many instances of a specific occurrence exist in the array.

10. Correlation Coefficient between the two arrays:

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

11. Kurtosis:

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

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

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

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

12. Skewness:

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

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

skewness=abs(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 that are evaluated by computing the Kurtosis and the Skewness of the signal's spectrum.
    Discrete Fourier Transform has been computed on both the signals ($x_{j,1}$,$x_{j,2}$), then only positive frequencies have been considered and over the two positive spectra the kurtosis index and the skewness index have been computed. Finally, the distances between the two vectors in terms of the two indexes have been evaluated by computing the absolute difference (as for points 11 and 12).
    Thus we had 2 other features (freq_kurtosis and freq_skewness).

These mid-level features (14 in total) have been evaluated over all the columns (low-level features), then we had in total 98 of them (14x7).
Finally, there are 5 other mid-level features evaluated on the low-level features from f_4 to f_8.
For having a measurement of **repetitiveness**, and hence if the dancer moves from a redundant and smooth movement to a more angular and disordered movement and vice versa, which is a pretty important salient event (S_2/S_3), we developed a specific index: *repetitiveness*.
This index consists of two different components, 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 than one large amplitude on the positive spectrum of the signal, we have to evaluate *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 *peakness* is a quantitative measurement, *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 **repetitiveness** index.
Computation was performed as follows:

Four_x=DFT(x),

pos_Four_x=(Four_x where frequencies f>0),

max_amplitude=max(pos_Four,T),

Where T is a 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 an amplitude different from the previous one.

repetitiveness=peakness*regularity,

return repetitiveness.

For the function regularity(), the computation was performed as follows:

distances=[], void list.

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

We append 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-Maximum),

if span$\neq$0: *return 0.1*,

else *return 1*,

It returns 0.1 (in case of a signal that can be interpreted as noise) for attenuating 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 with 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 compute *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.
To 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.

Mid-level feature	Description
Mean	Change in the mean of two signals
Variance	Change in the variance of two signals
MAD	Change in the median absolute deviations of two signals
Maximum	Change between the maximum values of two signals
Minimum	Change between the minimum values of two signals
SMA	Change between the evaluation of magnitude area under two signals.
Energy	Difference in the energy between two signals
IQR	Change in the interquartile value between two signals
Entropy	Difference in the Entropy of two signals
Correlation Coefficient	Difference between the Corr. coeff. of two signals.
Kurtosis	Difference in the comparisons with a Gaussian of the two signals.
Skewness	Difference in the symmetry within the signals with respect to their mean.
Kurtosis	Kurtosis evaluation of frequency.
Skewness	Skewness evaluation on frequency
Repetitiveness	Ratio between a measure of periodicity in the movement of two signals.

Table 3: Mid level features

5.1.4 Normalization

For what concerns normalization of the data, we computed two different kinds 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 this 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 *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 vectors that define a salience are close to each other in the future space.
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 ones for *k-1* frames. Thus features should be similar to those computed in the previous window.
Of course, if we put too many windows in the neighbor (after and before) equals *1*, we risk incorporating information (frames) that represents noise because those frames are more significant for defining zeros.
Then an upper limit of the number of frames to label with 1 is k (*n/2*, dimension of a sub window w_l where l$\in$[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 label *1*, we have w_1 of that window 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 since it determines whether is a salience or not (conceptually if the distance between w_1 and w_2 is zero in terms 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 the beginning to the end) 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 saliences (s_{i+1}, s_{i+2}); otherwise we take the frame in the middle of the pair of saliences and we set that sample's label 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 s_i with *1*, otherwise we label *k/2* windows with *1*, after the salience. 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 saliences is too short, the "absence of salience" in the middle doesn't have enough strong characteristics 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 had considered *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 classes 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 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.
As we can see, classes are still very unbalanced, but this time we have a lot of significant samples for the positive class.

5.3 Model Selection

About the Machine Learning (ML) model, since we have labels associated with 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.
We prefer to use classical machine learning since we already did the feature engineering step, and thus we don't need an algorithm able to find out features directly from the raw data like a Deep Neural Network (DNN) often does [9].
Among all the classical machine learning algorithms we prefer to use Random Forest.

5.3.1 Decision tree and Random Forest

A decision tree [15] is a model composed of several nodes where decisions take place. It can be viewed as a tree because it is a graph in which each node represents a decision and each leaf represents a final state, a class. Specifically, a certain number of features are evaluated over the dataset for each node. Therefore, the leaves represent groups of samples belonging to specific classes (which are the outputs).
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). 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 (Figure 16).

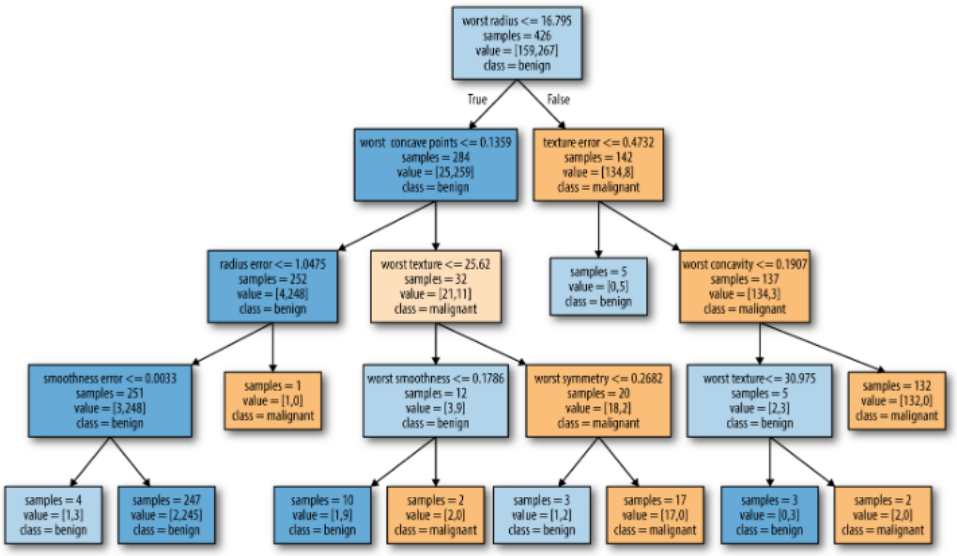


Figure 16: An example of decision tree for cancer breath classifying [15].

If the decision tree grows too much (too deep) there is the risk of overfitting the data, however, they are interpretable.
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 reducing 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 tree 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 Random Forest uses the same *hyperparameters* of the decision trees since it is a collection of them. The parameter N_Estimators is not considered as *hyperparameter*, because the more trees (estimators) there are in the model, the better will be the prediction.
Of course the computational cost increases with the number of trees.
The best Advantages of using a Random forest are the following [1] :

- Naturally handles both regression and (multi-class) 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.

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

5.3.2 Cross Validation

Cross Validation is a crucial step for any machine learning algorithm, because choosing the best *hyperparameters*, means choosing a good level of complexity, despite Random Forest has a good performance without setting any of them, however doing cross-validation even for random forest is a good practice.
During the 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 instantiation 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 corresponds 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 instantiations 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 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 downsampling before computing models, by randomly removing a number of ones such that the number of zeros was equals to them.
This downsampling was needed to have good results even in the negative class (*0*).

5.4 Results

For evaluating the results there exist several metrics, since we deal with a binary classification problem we choose to use the following:

- Accuracy, the accuracy expresses a general measure of how the system guesses correctly the output $\hat{y}$, indeed it takes into account how many samples have been correctly classified with respect to the entire test set.
  The related formula is the following:

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} T_i}{\sum_{i=1}^{n} (T_i + F_i)}$$

  Where $i \in \mathbf{C}$ ($\mathbf{C}$ set of classes), T_i stands for correctly classified samples of class i and F_i for wrongly classified samples of class i.
  Since we have two classes (positive and negative), the equation becomes:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$

Where TP stands for true positive, TN for true negative, FP for false positive, and FN for false negative.

- Precision, the precision indicates how many samples belonging to a specific class (positive) have been correctly classified with respect to how many samples which belong to the other classes (negatives) have been mistakenly classified into the positive class:

$$\text{Precision} = \frac{TP}{(TP+\sum_{i=1}^{n}(FP_i))}$$

Where $i \in \mathbf{C}$ ($\mathbf{C}$ set of classes), TP indicates how many samples of positive class are correctly classified, and FP_i indicates how many samples of class i are classified in class positive (i≠j, where j positive class).
Since we have two classes, the equation becomes:

$$\text{Precision} = \frac{TP}{TP+FP}$$

- Recall, the Recall indicates how many samples belonging to a specific class (positive) have been correctly classified with respect to how many samples which belong to the same class have been wrongly classified into negative classes:

$$\text{Recall} = \frac{TP}{(TP+\sum_{i=1}^{n}(FN_i))}$$

Where $i \in \mathbf{C}$ ($\mathbf{C}$ set of classes), TP indicates how many samples of positive class are correctly classified, and FN_i indicates how many samples of positive class are classified in a negative class i, (i≠j where j positive class).
For two classes, the equation becomes:

$$\text{Recall} = \frac{TP}{TP+FN}$$

- F1_score, for having a significant index which combines both *precision* and *recall*, we consider F1_score.
  This metric is computed as the Harmonic mean of precision and recall, hence it penalizes the low values, thus it has a high value only if both metrics are high:

$$\text{F1\_score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

  All the metrics presented before have been employed.
  For what concerns, precision, recall, and f1_score, they were evaluated both for the saliences (class with label *1*) and the absence of salience (label *0*), with this approach we can achieve a measurement of the overall quality of the model.
  In the next sections, an analysis of the performance has been conducted, by considering a low level of generalization (LOSO), for arriving at the highest level of generalization possible for our context (LODO).
  These different levels of evaluation are important because they help to understand which are the most useful samples, *takes* or dancers for the system, and how it generalizes properly.

5.4.1 LOSO

Leave One Saliency Out (LOSO) is a set of experiments.
Each experiment consists of removing a random salience from the dataset, then training the model over the remaining saliences in the dataset, and testing the model only over that specific salience removed.
By doing that we are going to check if the model can represent properly that specific salience without ever seeing it.
Often this approach is called Leave One Out Cross Validation (LOOCV), the difference with respect to the literature is that in our case, we don't take out just a sample, instead, we test the model over a whole salience, which is composed of multiple samples, that's because all the samples belonging to a salience (or a not salience) are very close in the space of features between each other, so we have surely over-fitting.
Hence this is the least general test we can do.
Another difference with the classic approach is that in LOSO, for computational limits, we were able to perform only 30 experiments (over 619) for the

salience, and 30 experiments for the absence of salience (over 87); LOOCV instead ideally perform tests over all the dataset, hence it makes **N** experiments, where **N** is the number of samples.
In the next discussion, we will refer to metrics as the average evaluation over the 30 experiments.

The results were very good for detecting the saliences, the results obtained by evaluating metrics over them (class *1*) gave us an Accuracy of 97%, an F1_score of 97%, indeed precision was 100%, and recall of 97%.
However, when the experiments were evaluated over the absence of salience samples, the results weren't so good.
So the F1_score was only 51%, with 73% of precision, 46% of recall and an accuracy of 46%.
The variance of metrics over the ones is way small (order 10^{-2}) because, over 30 experiments, only one of them gave us bad results, instead the variance of zeros was approximately 0.2
This behavior can be due to the lack of absence of salience samples in the training set, further, addressing a granular salience as we did for distal parts implies a too-dynamic behavior of the system. Finally, if we consider the average value of the metrics obtained with the salience evaluation and with the absence of salience evaluation, then we obtain:

- Accuracy=72%
- F1_score=74%
- Precision=87%
- Recall=72%

Which is our overall LOSO evaluation. In the next pages you can check the tables about the full experiment results.

	Accuracy	F1 score	Precision	Recall
mean	0.97111111	0.9745098	1.0	0.97111111
variance	0.02420247	0.01884275	0.0	0.02420247

Table 4: Mean and Variance of metrics for salience (label *1*) LOSO

	Accuracy	F1 score	Precision	Recall
mean	0.46264347	0.50909219	0.73333333	0.46264347
variance	0.18130719	0.183746	0.19555556	0.18130719

Table 5: Mean and Variance of metrics for non salience (label *0*) LOSO

Accuracy	F1 score	Precision	Recall
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
0.13333333	0.23529412	1.0	0.13333333
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0

Table 6: Metrics Results for positive salience (label *1*) LOSO

Accuracy	F1 score	Precision	Recall
0.45945946	0.62962963	1.0	0.45945946
0.36486486	0.53465347	1.0	0.36486486
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
1.0	1.0	1.0	1.0
0.45945946	0.62962963	1.0	0.45945946
1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
0.0	0.0	0.0	0.0
1.0	1.0	1.0	1.0
0.89189189	0.94285714	1.0	0.89189189
0.0	0.0	0.0	0.0
0.68918919	0.816	1.0	0.68918919
0.2027027	0.33707865	1.0	0.2027027
0.85135135	0.91970803	1.0	0.85135135
0.2027027	0.33707865	1.0	0.2027027
0.01369863	0.02702703	1.0	0.01369863
0.72972973	0.84375	1.0	0.72972973
0.0	0.0	0.0	0.0
1.0	1.0	1.0	1.0
0.0	0.0	0.0	0.0
1.0	1.0	1.0	1.0
0.05479452	0.1038961	1.0	0.05479452
1.0	1.0	1.0	1.0
0.0	0.0	0.0	0.0
0.02702703	0.05263158	1.0	0.02702703
0.0	0.0	0.0	0.0
1.0	1.0	1.0	1.0
0.82432432	0.9037037	1.0	0.82432432
0.10810811	0.19512195	1.0	0.10810811

Table 7: Metrics Results for not salience (label *0*) LOSO

5.4.2 LOTO

Leave One Take Out (LOTO) evaluation is more general than LOSO, if LOSO tests the performances by training the model over the entire dataset without looking at just one salience, LOTO evaluates the performance of the model by training over all the *takes* except for one, hence the level of generalization of this kind of experimentation is stronger than LOSO.
Since we have 21 *takes* (5 for the first dancer, 12 for the second dancer, and 4 for the third dancer), we have in total 21 experiments to perform. This case was computationally feasible, hence we did all the experiments.
This time we evaluated the metrics associated with the saliences (positive) and the metrics associated with the absence of salience samples (negative) together and then computed directly the average, since in the test we had a whole *take* which in general is composed by both, salience and not salience.
Results had similar behavior with LOSO, indeed we had an F1_score related to the salience very high (89%), but very low for the not salience (46%).
Accuracy is still good, but it's biased by the fact that in tests there were more ones than zeros because downsampling is performed over the training set.
As it can be appreciated, the results (Average F1_score) have a slight decrease in the performance with respect to LOSO, indeed the average metrics are:

- Accuracy=85%
- F1_score=73%
- Precision=77%
- Recall=74%

This is due to the generalization, as we previously said LOTO is more "difficult" than LOSO.
On the next page, we report the results for all the 21 experiments.

	Average				positive			negative		
	Accuracy	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
mean	0.847	0.729	0.762	0.739	0.899	0.876	0.939	0.463	0.553	0.443
variance	0.015	0.040	0.042	0.032	0.008	0.013	0.011	0.122	0.133	0.130

Table 8: Mean and Variance of metrics LOTO

Average				positive			negative		
Accuracy	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
0.761	0.492	0.631	0.524	0.862	0.771	0.977	0.122	0.492	0.070
0.775	0.552	0.681	0.558	0.868	0.788	0.966	0.236	0.575	0.149
0.819	0.450	0.410	0.500	0.901	0.819	1.000	0.000	0.000	0.000
0.547	0.427	0.518	0.506	0.690	0.554	0.913	0.165	0.482	0.100
0.841	0.457	0.424	0.495	0.914	0.849	0.989	0.000	0.000	0.000
0.747	0.644	0.711	0.633	0.835	0.763	0.922	0.453	0.659	0.345
0.997	0.989	0.998	0.980	0.998	0.997	1.000	0.979	1.000	0.959
1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
0.966	0.908	0.969	0.864	0.981	0.966	0.997	0.834	0.973	0.730
0.664	0.642	0.640	0.650	0.731	0.771	0.694	0.554	0.510	0.606
1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
0.736	0.424	0.368	0.500	0.848	0.736	1.000	0.000	0.000	0.000
0.698	0.694	0.745	0.787	0.730	1.000	0.575	0.658	0.490	1.000
0.850	0.818	0.843	0.803	0.895	0.858	0.935	0.741	0.829	0.671
1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.000
0.904	0.805	0.947	0.750	0.944	0.894	1.000	0.667	1.000	0.500
0.909	0.801	0.869	0.761	0.948	0.920	0.977	0.654	0.818	0.545
0.802	0.668	0.659	0.678	0.879	0.893	0.865	0.456	0.425	0.492
0.911	0.744	0.728	0.762	0.951	0.959	0.943	0.536	0.498	0.581
0.871	0.803	0.869	0.771	0.919	0.872	0.970	0.688	0.865	0.571

Table 9: Metrics results for LOTO

5.4.3 LODO

This is the most generic experiment, indeed Leave One Dancer Out (LODO) is an approach in which all the *takes* related to a dancer has been used for testing, and the other two dancers' *takes* were used for the training phase, as for LOTO, we can expect a further decrease in the performances, first because all the patterns learned from the model belong to other dancers, second because with respect to others approaches (LOSO, LOTO), the quantity of samples in the training set is way lower, hence less information.
Thus, although the general behavior is similar to the previous evaluation, the general F1_score was decreased a lot.
We did all the 3 experiments of course, and the average values of the metrics are the following:

- Accuracy=81%
- F1_score=67%
- Precision=72%
- Recall=67%

With respect to LOTO, the F1_score is decreased by 5 percentage points, other metrics as well. By looking at the experiments (Table 11) we can see how the samples related to the dancer Muriel are more important for describing the system, indeed the experiment by taking out all her *takes* has an average F1_score which is more then 20% lower with respect to the other two experiments.

	Average				positive			negative		
	Accuracy	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
mean	0.815	0.671	0.724	0.671	0.889	0.848	0.940	0.452	0.601	0.402
variance	0.003	0.012	0.004	0.009	0.001	0.005	0.000	0.035	0.005	0.041

Table 10: Mean and Variance of metrics LODO

In the next page we report the table associated to the three experiments.

	Average			positive			negative		
Accuracy	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall
0.855	0.732	0.736	0.728	0.914	0.911	0.918	0.550	0.561	0.539
0.856	0.764	0.794	0.744	0.912	0.887	0.938	0.617	0.702	0.550
0.735	0.516	0.643	0.540	0.842	0.747	0.964	0.191	0.539	0.116

Table 11: Metrics results for LODO

6 Conclusions

In this thesis we explored a machine learning approach for dealing with the problem of salience detection in the context of movement, we started by understanding how the raw data can be obtained from a physical layer, and then how to exploit them for achieving meaningful low-level features.
We defined a concept of salience in our domain, and hence extracted proper features for representing them.
We built our samples by using a prototype of sliding windows over the frames and a specific set of mid-level features.
By selecting the best windows to represent as best as possible the two classes, we have collected the results.
Experiments pointed out a good representation of salience with an average F1_score always close to 90%, but not satisfying results for the negative class (F1_score under 50%).
This lack of information for the negative class is probably due to the dynamic of the system because by including local saliences in the ground truth, the probability of having 2 frames labeled *1* very close to each other is increased and all the windows in the middle can "feel" the influence of the two saliences (before and after), hence in the space of features those samples can easily collide with samples related to salience, that's why we discarded (Chapter 5.2) windows in the middle of two saliences which are too close, however by doing this, the remaining windows labeled with *0* were few, and hence the class is not represented very well. We can see the number of zeros to prune as a *hyperparameter* which depends on the dynamic of the system.
For our context, if we prune more, or fewer zeros, we will have a decrease in the performances.
Summarizing, approaches for solving this issue can be enlarging the number of frames embedded for each window (but having a larger delay on the online system), annotating more *takes* for having more data, considering a concept of salience less dynamic and more holistic, lastly, try to use a more complex algorithm in order to define better the boundaries between zeros and ones, such that you can prune fewer zeros.

6.1 Future Researches

In a future approach, it can be possible to try this algorithm over a different definition of salience, maybe from a more holistic point of view and by en-

larging the number of frames within the sliding windows, or trying it over a larger quantity of videos for improving the quality of the information learned by the model.
Our first goal was to put the basis for an online system that exploits these results for detecting salience in real time, hence it could be for sure a good direction to pursue for developing new technologies useful in different contexts, such as art, rehabilitation, sports, etc.

References

[1] D. Adele Cutler, Richard 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] R. C. Atkinson and R. M. Shiffrin. *HUMAN MEMORY: A PROPOSED SYSTEM AND ITS CONTROL PROCESSES*. 2. The psychology of learning, motivation: Advances in research, and theory, 1968, pp. 89–195.

[5] Stef van Buuren. "Flexible Imputation of Missing Data". In: *Chapman Hall/CRC Interdisciplinary Statistics Series* 1.1 (2012), pp. 3–15.

[6] 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).

[7] 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).

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

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

[10] T. Hu et al. "Human interaction recognition using spatial-temporal salient feature". In: *Multimed Tools Appl* (2019).

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

[12] M. Mancas et al. "Gesture in Embodied Communication and Human-Computer Interaction". In: *8th International Gesture Workshop* (2009).

[13] M. Mancas et al. “Gestures Saliency: a Context-based Analysis”. In: *Proceedings of the Gesture Workshop* (2009).

[14] M. Mancas et al. “Real-time motion attention and expressive gesture interfaces”. In: *J Multimodal User Interfaces* (2008).

[15] Andreas C. Müller and Sarah Guido. “Introduction to Machine Learning with Python: A Guide for Data Scientists”. In: *O’Reilly Media* 1.2 (2017), pp. 70–92.

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

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

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

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

[20] C. Peters et al. “Fundamentals of Agent Perception and Attention Modelling”. In: *Emotion-Oriented Systems: The HUMAINE Handbook.* 1.1 (2011), pp. 293–319.

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

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

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

[24] W. Wang et al. “Revisiting Video Saliency Prediction in the Deep Learning Era”. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 43.1 (2021).

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

[26] M. Zehetleitner, M. Hegenloh, and H. J. Müller. “Visually guided pointing movements are driven by the salience map”. In: *Journal of Vision* (2011).