What if we are able to predict movement?- The Move2Data Preprocessing Framework

Rocío González Lantero

IE University

Advisor: Mikel Diez

Co-advisor: Borja González del Regueral

Abstract

Human movement prediction is a complex yet necessary task researchers are starting to work on. Although several studies regarding this exist, none have been entirely successful in generalizing results. Furthermore, there is no standardized proposed way to deal with human movement data. Therefore, this study will focus on developing a standardized preprocessing approach for human movement data, which will be tested using the most complex type of movement, namely dance. The main problems faced when preprocessing such data are the coordinate detection, missing values, value range, combination with extra features, and the model input preparation. The AIST Dance Database is used to test this. The proposed framework addresses all issues mentioned above.

Keywords: human movement, PyTorch, data preprocessing, mocap data, GRU

Acknowledgements

I want to express my gratitude to everyone who has supported me and helped me throughout the research process.

I would like to thank Tetyana Kretova and Manuel López de Blas for reading the draft and giving me constructive criticism to improve the report.

I would also like to thank David Kremer for helping me explore different methods that could be employed. Furthermore, I would like to express my gratitude to Paz Vega, CEO, and co-founder at Aitaca, for bringing me some insight into their work regarding human measurements from images and a new perspective on the struggles they face and for validating this solution by showing me potential applications the proposed solution could have.

Furthermore, I would like to thank my advisor, Mikel Diez, and co-advisor, Borja González del Regueral, for all the support, ideas, guidance, constructive criticism, and time invested in helping me with this project. They have not only guided me on this project but also showed me how professionals work and deal with the different issues one encounters when embarking on this kind of projects.

Introduction

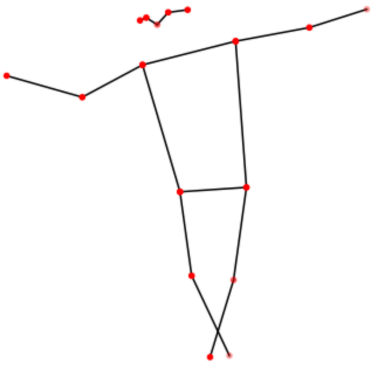


Figure 1: Transformed Human By Computer

Human movement prediction consists of capturing and synthesizing human positions or positional changes so models can understand them correctly. The first and critical step for human movement generation is to preprocess the data optimally. Many papers have attempted to predict movement, and each has preprocessed the data in its own way. However, none have been able to generalize or adapt their preprocessing to all kinds of data. Therefore, this paper aims to develop a standardized technique for human video data, which all models will be able to process, focusing on those developed on PyTorch.

An accurate and standardized preprocessing could open the doors to the rest of the community to develop more human movement models, which could disrupt the entire audiovisual industry, speeding up the process of creating choreographies for concerts, video clips, and other shows, consequently decreasing costs. It would also lead to shows and new combinations of movements, disrupting the dancing industry and challenging dancers, bringing them to their limits, and potentially spearheading human-computer collaborations. Furthermore, it could even affect the health industry, for instance, allowing to diagnose diseases through processing data derived from studying sizes and ways of walking.

The following will first analyze previous research and the techniques used, which will be tested, and their pros and cons will be analyzed and discussed. Finally, each technique's optimal combination and adaptation will be used for this research. Furthermore, some models will be tested to verify that they can process the data under the final proposed preprocessing framework.

Literature Review

There have already been some developments regarding choreography generation in past research studies. However, they have not incorporated the audio mapping. They have only continued generating random movements from a given movement sequence (Crnkovic-Friis & Crnkovic-Friis, 2016) or have been unable to generalize enough to be used in songs that have not been included in the training set (Alemi et al., 2017). Hence, a new development in this area could advance the state of these models and potentially improve the audiovisual industry. An overview of the previously used techniques and research projects relevant to this discussion will be provided.

The following areas have been researched as the fundamental building blocks of this research:

1. Animation: The use of mocap to define human key points, low dimensional Kalman Smoothing for missing values, COCO standard for mocap.
2. Audio Analysis: VGG-19 for feature extraction and Mahalanobis distance for clustering.
3. Human Movement Classification: farimotion for clustering, PCA and PPCA for dimensionality reduction.
4. Body Movement Generation from the Previous Movement: RNNs for predicting and the transformation from 2D coordinates to 3D.

5. Body Movement Generation from Audio: They have not been successful with generalizing.

Animation

One of the applications of animation that is of interest for this project is video games. Video game designers need movement and behavior to seem as natural as possible. Hence, they deal with movement analysis, generation, and interpolation.

A common practice in this area that could be useful for developing this study is mocap data. Mocap data is generated by many open-source algorithms striving to detect the key points of humans in videos (hips, chest, neck, head, collars, shoulders, elbows, wrists, knees, and ankles), also referred to as markers (Meredith & Maddock). The COCO key point dataset is the standard for training such models, which annotates 12 body and five facial key points per identified human, having 17 key points in total (Papandreou et al., 2018).

One of the most common algorithms implemented in python is Pose Estimation, which "employs a convolutional network which learns to detect individual key points and predict their relative displacements, allowing us to group key points into person pose instances" (Papandreou et al., 2018). However, one of the main challenges this model presents for further modeling is the absence of some markers while preprocessing, which gives missing values that need subsequent estimation. To estimate these values, low dimensional Kalman smoothing is one of the best approaches, as it reduces error variability and can estimate missing markers, even for many frames (Burke & Lasenby, 2016). This technique requires many frames with all marker positions to infer the missing key points through singular value decomposition (Burke & Lasenby, 2016).

Google also developed a model named BlazePose, which infers 33 key points, extending the standard list of 17 key points (COCO topology). The model goes through 2 steps, first detecting the area of interest, in this case, the human, and then detecting the key points in the first frame and inferring these positions in the subsequent frames. The rest was inspired by Leonardo's Vitruvian man, predicting all the key points by creating a circle surrounding the whole person and having as center the hip midpoint. This model was mainly developed for fitness trackers and the community to create new applications (Bazarevsky & Grishchenko, 2020). This model's added value is that it can estimate the 3D point coordinate position of all 33 key points from one video.

Audio Analysis

One of the extra features that can be included in human movement prediction is audio, which could be helpful for choreographers. Multiple researchers have been working on audio analysis and feature extraction.

One model that can be used for this matter is VGG-19, which was initially trained for image classification purposes (Simoyan & Zisserman, 2015). This model has 19 layers, composed of convolutional, max pooling, fully connected, and SoftMax layers. VGG-19 performs significantly better than other convolutional networks as it provides increased depth and uses a smaller receptive field (the area the model is looking at) and stride (number of pixels it shifts to look at the whole image) (Simoyan & Zisserman, 2015). Furthermore, VGG-like networks have been used for audiovisual recognition, using similar architectures for audio and video inputs (Ramaswamy, 2020). The audio features are extracted through spectrogram generation, representing these sequences with time, similar to a TFR (time-frequency representation), having time along the x-axis, frequency along the y-axis, and color representing the strength of the frequency (Wyse, 2017). Using this as an

input for a convolutional network has shown improvements compared to manual feature extraction through Python libraries, such as librosa, for tasks such as generation and classification (Wyse, 2017).

Furthermore, other applications that use audio features are created for classification and segmentation purposes, such as news segmentation or genre classification. Researchers have found that agglomerative clustering, a form of hierarchical clustering, using Mahalanobis distance or Kullback Leibler (KL2) distance gives the best results (Siegler et al.). This algorithm treats each observation as a separate cluster and then pairs the closest cluster pair with one another until it creates a huge cluster composed of all observations. The similarity is measured with the distance one establishes, in this case with the Mahalanobis or Kullback Leibler.

The Mahalanobis distance is calculated in a multivariate space, as opposed to the Euclidean distance. If the variables are entirely uncorrelated, both would have the same value, but if this is not the case, the Mahalanobis distance outperforms the Euclidean distance. Moreover, when many variables are involved, the Mahalanobis distance can deal with them easily, considering the covariance matrix when calculating the distance (De Maesschalck et al., 2000). Finally, the KL2 distance "is an information theoretic measure equal to the additional bit rate accrued by encoding random variable B with a code that was designed for optimal encoding of A" (Siegler et al.).

Human Movement Classification

Many applications aim to detect and classify human movement performing different tasks. One of the uses of human movement classification is used by robots when learning how humans move naturally. First, they detect the movement and learn which task they are performing to then be able to imitate it.

Aiming to develop state-of-the-art mocap data preprocessing, Facebook developed a library named fairmotion (Gopinath, 2020). One of their investigation areas was clustering such data and optimizing it. The proposed method uses a window of 5 frames and calculates the logarithmic kinetic energy per joint (Onuma et al., 2008). After this, they discovered that performing Principal Component Analysis (PCA) has the same results as more sophisticated dimensionality reduction techniques (Onuma et al., 2008). Furthermore, once this preprocessing is completed, they use the Euclidean distance and the standard clustering algorithms, such as agglomerative clustering (previously described), which give outstanding results for identifying whether a person is running or walking, or jumping.

Another proposed approach is to measure the dimensionality of each set of frames. This approach assumes that more complex motions have higher dimensionality than simple actions, detected by using PCA or Probabilistic Principal Component Analysis (PPCA), an extension of PCA. The main difference is that it defines a probability model for PCA, enabling it to model the noise. The noise in traditional PCA is that variance that moves in a direction outside the subspace, which is usually removed, but PPCA models it through probability (Barbicˇ et al., 2004), giving better results. Through the PPCA approach, all framesets are modeled through Gaussian distributions and compared to other framesets regarding their dimensionality (Barbicˇ et al., 2004). This approach can detect long-term behaviors, such as the difference between walking and running.

Body Movement Generation from the Previous Movement

Several techniques have been developed regarding movement generation without considering extra features. The most relevant to this paper is ChorRNN, which predicts new dance sequences for a solo dancer given a previous movement sequence. ChorRNN is a

"deep recurrent neural network trained on raw motion capture data" composed of LSTMs (Crnkovic-Friis & Crnkovic-Friis, 2016).

Moreover, there has been research to generate the next movement regarding different tasks, such as "walking, smoking, engaging in a discussion, taking pictures, and talking on the phone" (Martinez et al., 2017). One of the highlights of this paper is its preprocessing technique. They represent each pose "as an exponential map representation of each joint, with a special preprocessing of global translation and rotation" (Martinez et al., 2017). To predict the movements, they used a single gated recurrent unit (GRU) with 1024 units and a decoder to project the higher dimension output of the GRU layers to the 54 dimensions on their dataset, representing the human body markers. Additionally, they divide their prediction into two types: short- and long-term. For the short-term, they feed two seconds to predict the next 400 milliseconds, and for the long-term, they also feed two seconds to predict the next second. They used the "Euclidean distance, between the prediction and the ground truth" (Martinez et al., 2017) to evaluate the model. These techniques outperform previous work, especially in the case of the short-term model, and they discovered that being able to label the activities was very beneficial for model performance, allowing them to build one model per action.

An additional paper, aiming to predict choreographic movement (dance) from the previous pose, also proposes a very interesting and completely different preprocessing of the raw videos. By filming the same video from different angles, the authors of the research estimated the 3D coordinates. They first detected the 2D coordinates and then calculated the 3D coordinates. However, as it is an estimation there are some key point coordinates that could not be recovered resulting in missing values (Li et al., 2021). Additionally, the paper does not give the details of preprocessing times. However, it says that "running this

pipeline on a large-scale video dataset requires a non-trivial amount of compute and effort" (Li et al., 2021). To validate their technique, the researchers measured the difference between the initial 2D coordinates they detected and the 2D points they reprojected from their 3D estimations, overall getting low differences and, therefore, good results.

Body movement Generation from Audio

Several projects have added the complexity of including audio features to predict human movement, either for choreography prediction or even movement related to the playing of the violin. These first preprocess audio and video separately and then combine them. To do so the preestablished Python libraries allowing extraction of audio features, mainly librosa, were used.

Regarding modeling, researchers have used a U-net architecture in more controlled environments involving the generation of violin playing movement, combining CNNS and LSTMs (Kao & Su, 2020). The U-net architecture was developed to get as many features as possible, so convolutional models could be trained with a smaller sample when performing biomedical segmentation problems. It has a contracting path, which extracts the features from the images and downsamples, and the expansive path, which upsamples the training set (Ronneberger et al., 2015).

Furthermore, researchers have used conditional models involving choreography generation, most specifically, Factored Conditional Restricted Boltzmann Machines (FCRBM) combined with RNNs (Alemi et al., 2017). However, these have not been successful in the generalization phase, not being able to maintain performance when introducing audios not included in the training set.

Data & Methodology

The following explains the end-to-end process of the project. This paper will focus on movement generation while also considering audio, aiming to test different preprocessing techniques and validate them. Using audio-related movement will allow testing the most complex form of movement, which is dance. This movement is complex, as there is no standard as for walking, and there is much creativity involved. The creativity leads to movements that a standard will not be able to predict, inserting some extra randomness into the movement the model should learn to take into account. Furthermore, the preprocessing and postprocessing will be adapted to make it easier for dancers to learn the choreography.

Data

The AIST Dance Video Database, "a shared database containing original street dance videos with copyright-cleared dance music" (Tsuchida et al., 2018), will be used for the rest of the project. This dataset is stored online and allows users to download all videos, only the audios, and filter by genre, dancer, or choreography through an API. However, they also give one the option of downloading it manually through their website. It contains 13,939 dance videos comprising ten dance genres and 60 different music audios. Overall, 40 professional dancers appear in these videos, 25 male and 15 female, performing solo and in groups choreographies and filmed from nine different angles (Appendix 1). The videos come in mp4 format and are divided into four groups: basic dance, advanced dance, group dance, and moving camera (Appendix 2). Moreover, 50 videos belong to one of three situations: showcase, cypher, and battle (Appendix 3). Initially, this database was created to foster tasks like dance-motion genre classification, dancer identification, and dance-technique estimation (Tsuchida et al., 2018).

A newer version of this dataset is AIST++, which provides the 3D human key points to the dataset described above (Li et al., 2021). This dataset with the 3D coordinates was constructed at 60 frames per second and considered all nine camera positions used in filming the videos from the AISTDB. Therefore, the dataset is reduced to 1,408 sequences. However, it still covers the same genres, audios, and dancers. Additionally, the creators allegate it is the "largest and richest existing dataset with 3D human keypoint annotations" (Li et al., 2021). This dataset can also be downloaded through an API or directly through the website. One can download the mocap data, the 2D coordinates, the 3D coordinates, or the camera data only to calculate the 3D coordinates. Moreover, they offer to download their train-test-split and the clean data to input into their developed model.

In the following, both datasets (the old and the new) will be tested to determine which one is better for the problem at hand.

Methodology

Approach

As analyzed in the literature review, there is no standardized way to prepare the data for human movement generation. The models that deal with unstandardized movements, such as dance, are not entirely successful in generalizing with new movements or audio tracks. The proposed hypothesis for this issue is that there is a problem with the data input format. Different datasets are being employed, so the central hypothesis is that the problem is in preprocessing such data. Therefore, there will be several techniques tested in the following until reaching an end-to-end optimal preprocessing framework to predict movement with other features.

This research deals with two different data types, namely audio and video data. Both of these come combined into the video, so the first steps consist in separating them and

preprocessing them accordingly. The videos come in mp4 format, which have been processed by separating the audio and the frames. The audio has been transformed and saved as both wav and mp3 format, the video has remained being mp4 but without the audio. After this, they must be combined most optimally.

Audio Preprocessing

The dataset entails 60 different audio tracks used to dance in different styles. None of the audios have lyrics, and they have very well-defined beats. In previous studies, researchers used librosa to extract features. However, the results have not been generalizable to new audio tracks, and the researchers do not provide enough evidence to draw meaningful conclusions. The hypothesis for this study is that there are many relevant features that libraries cannot capture.

In order to see if more features can be extracted with other techniques, VGG-19 was used. This model extracts the features directly from the spectrogram of the audio. As mentioned in the literature review section, a spectrogram is similar to a TFR, graphically representing the relationship between time, frequency, and frequency strength (Wyse, 2017). The VGG-19 model (Appendix 4), a CNN, is then executed and extracts 25,088 features.

As an initial inspection, all audio tracks were processed to analyze the features extracted. Of the 25,088 initial features, 8,165 were kept after removing the constant ones. From these, PCA was performed to reduce dimensionality, and create clusters, to see the main differences between tracks. The number of components and clusters were treated as hyperparameters to create optimal clusters. Several combinations were tested, and three different metrics were measured: Silhouette, Calinski Harabasz, and Davies Bouldin.

The Silhouette score ranges between -1 and 1 and measures the inter- and intra-cluster distances. This means it measures the distance between observations in the same cluster (intra-distance) and the distance between clusters (inter-distance). The closer the value is to 1, the better, as the intra-distance is minimized and the inter-distance maximized. If it is negative, it means it has been assigned to the incorrect cluster, as there exists another cluster, which is more similar to the observation, and if it is 0, it means clusters overlap. The Calinski Harabasz score measures the ratio between dispersions in clusters; the higher it is, the better. Lastly, the Davies Bouldin score measures the cluster diameter vs. the distance between cluster centroids; the lower it is, the better.

After testing many combinations, it was found that the optimal combination was using five principal components and seven clusters, created through the Mahalanobis distance (De Maesschalck et al., 2000) and agglomerative clustering algorithm (Siegler et al.). This combination gave a Silhouette score of 0.41, a Calinski Harabasz score of 59.31, and a Davies Bouldin score of 0.48. The five principal components explained 39% of the variance. Therefore, it can be concluded that many of the features generated by VGG-19 are not relevant to differentiating between tracks.

To further inspect these clusters, some manual characteristics through librosa (Appendix 5) were computed to understand if VGG-19 was also differencing those key features used by other researchers and whether it was detecting additional ones.

Firstly, a multivariate analysis of variance (MANOVA) was computed to determine if the relationship between variables was related to the cluster to which they were assigned. It was concluded that the relationship was significant (Appendix 6). Therefore, it can also be concluded that VGG-19 detects all variables used in previous studies, so no previously

used features are lost. The next step consists of verifying if it adds value by measuring other valuable features.

The averages of each variable grouped by their corresponding cluster were compared to measure the capturing of additional valuable variables. It was established that all clusters except for three could be isolated only through these variables (Appendix 7). Hence, the hypothesis remains, being that VGG-19 is capturing something else, not captured by those variables, which could potentially improve human movement generation.

Video Preprocessing

For the preprocessing of videos, there are two initial steps:

1. Transform into tabular data.
2. Decide whether to deal with the project as a classification or regression problem.

Tabular Data Transformation

In order to deal with the video data optimally and without consuming too much computational power, instead of using image data (frames), it is proposed to use tabular data. For this, all frames must be first transformed into tabular data. For this matter, three different alternatives were tested:

1. Pose Estimation Algorithm: Although this option is easy to implement, it creates too many missing values when calculating the 2D coordinates.
2. Google Blaze Pose: This option estimates the 3D coordinates and creates fewer missing values, but it is computationally expensive.
3. AIST++: This dataset provides the 3D coordinates estimated by the camera positions and has a few missing values that can be easily inferred.

Pose Estimation

The first method proposed is using Pose Estimation (Appendix 8) to detect the location of 17 key points from the human body, located through cartesian coordinates (Papandreou et al., 2018). The model calculates the key points by estimating short-range and mid-range offsets and heatmaps. The main drawback of this methodology is the multiple missing values that it generates. As standard practices for filling missing values, such as using the average or filling them with zeros, can distort the skeleton's position and therefore learn unfeasible postures, the only option found in previous research is Kalman Smoothing (Burke & Lasenby, 2016). In this case, it also was not enough, as it requires more than one frame with all key points being detected. Over 11% of the sample would be lost when using this algorithm. The hypothesis behind this drawback is that this method only locates the key points in a 2D environment. Therefore, the algorithm cannot detect the key points behind one another (a person looking to the side). In consequence, this method was discarded.

Google Blaze Pose

Google Blaze Pose (Bazarevsky & Grishchenko, 2020) solves one of the issues mentioned above: only having 2D coordinates. This algorithm (Appendix 9) is computationally expensive, taking over 4 hours to preprocess one of the videos with the available computational power. 13,939 videos would take over six years to preprocess. Hence, this method was discarded. However, if there is no additional data regarding camera positions and the same video is not filmed from different angles, this would be the way to go. If this method is used, it is recommended to discard some of the key points, as the detail of 33 points is unnecessary (Appendix 10). One should only keep one key point per hand, foot, eye and get rid of the mouth.

AIST++

A previous study aiming to predict movement using the same dataset (AIST DB) developed a new way of preprocessing the raw video data. Researchers converted the videos into 3D data. The researchers first recovered the "camera calibration parameters and the 3D human motion in terms of SMPL parameters (Skinned Multi-Person Linear Model)", containing the 17 COCO notation key points and the 24 SMPL parameters (Li et al., 2021). SMPL can represent different body types on top of the skeleton (Loper et al., 2015). In the original AIST DB, the videos were recorded from different angles, which allowed the researchers to estimate the 3D coordinates, resulting in 1,408 different files (one per unique video) with their corresponding coordinates, estimated at 60 frames per second. This dataset does contain missing values, and researchers do not specify how these were filled. The only option found for filling out the missing values was using Kalman Smoothing (Burke & Lasenby, 2016). In contrast to using Pose Estimation, there were enough complete samples to infer the missing values. Due to the cleanliness achieved, this dataset will be used for the entirety of this paper.

Path Definition: Classification or Regression?

After having the complete data table, there are two possible paths, meaning the full table of human key point coordinates. The first path was to treat the problem as a classification problem, which will require clustering poses and labeling them accordingly. The second option is to treat the problem as a regression problem, specifically as an autoregressive time series. The model will therefore consider the previous pose to predict the following. If this were not treated so, the frames would have no cohesion, and they would be separate images that are unfeasible to be reproduced by a human. The extraneous variables of the audio features should also be added.

Labeling for Classification

There are two options proposed to label the data: movement clustering and rule-based labeling. For both of these options, movement is labeled, not poses. Therefore, the frames need to be differentiated. Researchers have previously used a 5-frame differencing to cluster large movement changes, such as walking or jumping (Onuma et al., 2008). In this case, details are essential, as slight movements, such as lifting the arm one centimeter, are relevant. Therefore, it was established that the differencing should be lower to achieve greater detail. As a result, a differencing of each two frames was used as a starting point.

Clustering

This section focuses on the movements and not the frequencies in which they move, so the 21 unique choreographies were selected, with 51 variables each (17 joints times three dimensions). This way, the computational power required was also reduced. Despite reducing the dimensionality, the number of remaining variables was too high for a simple clustering algorithm. Hence more dimensionality reduction techniques were necessary. As proven in previous studies, using PCA to reduce dimensionality had no apparent adverse effect on the clustering (Onuma et al., 2008). It is recommended to explain 90% of the variance to get optimal performance. Twenty-six principal components were selected to get 90% of explained variance. However, after creating the clusters and comparing the movement in the different clusters, the clusters did not show any patterns. The hypothesis behind this is that there is too much diversity of movements and not enough similarities between the videos for the clustering to be optimal, also not allowing the labeling of the clusters, introducing unexplainable bias. Therefore, this method was discarded.

Rule-based

The alternative method consisted in creating rules, which automatically classified the movement. No previous research was found, so it would consist of creating different rules and creating the labels manually. The movement dealt with in this paper, namely dancing, is very complex, so it would be challenging to cover all options. Additionally, the labels would be pre-made to the researcher's mind, introducing much bias and leaving no room for new innovative movements. Therefore, this method, as well as the classification method in general, was discarded.

Regression

To treat the project as a regression problem, the goal is to predict the 3D coordinates of each key point. For this, several issues need to be solved:

1. Length Standardization
2. Value range decrease

Length Standardization

For the model's architecture to be simpler and for this preprocessing framework to work for all models, the sequence length of all the videos in the sample should be standardized. This length standardization consists in cutting them into the number of frames of the shortest video. In this case, the shortest video has 426 frames. Therefore, longer videos are cut into groups of 426 frames. The series length can be established through a rolling window of length 426, shifting the windows each x frames (referred to as shift), defined by the programmer. If the shift is equal to the window length, the windows will not overlap, and the sample size in terms of the total number of frames will remain the same. However, if the shift is smaller than the window length, the windows will overlap and

upsample the dataset by duplicating some frames. In this case, a shift of one was established, meaning the dataset was upsampled as much as possible.

No Further Preprocessing - Autoencoders

After having all sequence lengths standardized, the first method uses the raw coordinates as an input. This consists in using the x,y, and z coordinates as they are. To make a quick test and see whether this is effective, autoencoders were run to see if they could decrease dimensionality and use that as input for an RNN. The best autoencoder had a dense layer with relu activation and a dropout layer of 0.3 for both the encoder and decoder (Appendix 11). Nevertheless, the errors were very high for both the training and the validation set (Appendix 12). As there is little overfitting, it is not about the sample size but about the model complexity (it being too low) or the preprocessing of the data. The model complexity was increased by increasing the number of layers. However, the errors only increased. Therefore, the problem must lay in the preprocessing of the data. The hypothesis for these huge errors is that the coordinate values cover a vast range several orders of magnitude larger compared to the cases in traditional machine learning, making it difficult for the model to predict, giving huge errors. To solve this issue, two solutions were tested:

1. Scikit-learn Normalization
2. Intra- and Inter-Movement Normalization

Scikit-learn Normalization

A common normalization technique used in many machine learning problems is scikit-learn normalization (Pedregosa et al., 2011). This method can also be back-transformed, meaning that the values returned by the model can then be transformed back into the coordinates with the exact dimensions as they had initially. This normalizer

converts all observations to the unit norm, ranging between -1 and 1. The autoencoders tested this method to see how well a neural network can deal with this data. The errors were greater when back-transforming; consequently, this method was discarded.

Intra- and Inter-Movement Normalization

As standard normalization techniques were unsuccessful, another more customized technique was tested. Previous research had proposed a standardization method consisting of creating quaternions of relative positions of the key points (Martinez et al., 2017). This technique consists of dissecting movement into two different types. The first one is the intra movement, similar to the relative movement used in physics, which describes the movement of one object A related to the position of object B. In this case, it describes the position of all joints related to one joint, referred to as the reference joint. The second is the inter-movement, similar to the absolute movement in physics, which describes the movement of a body from one space into another. In this case, it is the movement of the reference joint concerning the space in which it is. Usually, the hip is used as a reference joint. The data at hand contains both the left and right hip, so it was decided to use the left hip.

For the intra-movement normalization (Appendix 13), the sine, cosine, and r, the hypotenuse from the reference joint to the key joint, must be calculated. Although coordinate z is related to the inter-movement, it will be kept so it can also be predicted. However, one can also drop it and infer it with the inter-movement predictions. Moreover, all data concerning our reference joint, in this case, the left hip, will be ignored. This reduces the range covered by the original coordinates and is very easy to back transform.

Just the left hip coordinates will be included for the inter-movement normalization. Although the value range is not reduced here, the dimensionality problem of the data set is

removed, as now the model would only be dealing with three variables. If one does not want to create an extra model for this point, trajectories can be preestablished, making the human move in a predefined way, for example, in a circle.

Input Data Structure

There are two possible data structures for the input data, only considering the video, individual input, or array input.

The array input option has one column per key point and an array of length four in each observation (sin, cos, r, z). When using PyTorch, this can be inputted directly as a list or as a tensor. The problem is that both of these formats require very complex model architectures to read the input and define the target correctly. In the case of using PyTorch, as explained below (PyTorch TimeSeriesDataSet section), researchers could not adapt the function to capture the input and target correctly. Although this reduces the dimensionality in terms of the number of variables, using it as input overcomplicates the model's architecture.

Additionally, all saving formats do not support saving arrays into a data frame, and pandas recognizes the arrays differently depending on the saving format. Hence, this method is discarded. However, if someone is willing to continue this processing line, saving the data as pickle is recommended.

The second option divides each array into separate columns, having four columns per key joint. Using this technique, there are 68 numeric columns, which can be easily input into the model without overcomplicating the architecture. The only drawback is the dimensions, but one can build a neural network to cope with it. For this, using PyTorch is recommended, and it will be explained below how to structure the input accordingly. Therefore, this is the recommended method.

Audio - Video Integration

For the integration of the audio features, the audios were cut according to the frames per second. As mentioned above, AIST++ was preprocessed at a rate of 60FPS, meaning each frame represents approximately 16.67 milliseconds of the video (Appendix 14). Therefore, all audios were cut into 16.67 milliseconds long frames, and the corresponding features were extracted and matched with their corresponding frames.

PyTorch TimeSeriesDataSet

Once the dataset is clean and shows the intra-movement of the humans and the corresponding audio features, the next step consists in relating this data to time and structuring the input accordingly. This paper will focus on the intra-movement, as it is the most complex, dealing with a higher number of joints and variables. Furthermore, the intra-movement is more relevant in the choreographic world, as it defines the choreography itself, leaving room for choreographers to define the transitions and create the show through collaboration with machines.

For this matter, PyTorch will be used. Pytorch is one of the most used libraries, which "provides an imperative and Pythonic programming style that supports code as a model, makes debugging easy and is consistent with other popular scientific computing libraries, while remaining efficient and supporting hardware accelerators such as GPUs" (Paszke et al., 2019). Any training of the data at hand is computationally expensive, as video data is one of the heaviest types of data available. Therefore, having efficient model training is crucial. Another great advantage of PyTorch is that it is open-sourced. Using it is a great way to contribute to the community, showing different uses and making it available to the community for improvement.

On top of this library, there is an already built module for temporal data, named TimeSeriesDataSet, which eases the train-test-split and structures it so that any PyTorch model can directly use it (Beitner, 2020).

Time	Targets	Group Id
0	list of targets	Group 1
1		Group 2
2		Group 3

Figure 2: Example Columns for TimeSeriesDataSet

When building it (Appendix 15), one must define the training set with the corresponding ordered index (no gaps), enter the column's name identifying the time, the target variables, the group ids identifying the different time series one has in the sample, and the time-varying variables divided into categorical and numerical variables. Furthermore, one must indicate how many time steps one wants to use to predict and how many one wants to predict so that the splits can be done accordingly. Lastly, one must choose the PyTorch normalizer for the target.

In this case, dealing with multiple targets, it is compulsory to use the MultiNormalizer, inside which one can then choose which normalizer to use for each target separately. This normalizer is a wrapper in which all normalizers listed are executed to their corresponding target (related by position). There are mainly two normalizers to use. The EncoderNormalizer is fit in each encoder sequence separately, and the GroupNormalizer is fit by groups, established by the coder as an argument of the function. After this, the data is loaded into the data loader, ready to train. There is a bug in the original code one

downloads with the library. This error is given because the MultiNormalizer is not fitted yet. However, one cannot fit it before putting it into this function because one is also defining the target here. This error was debugged, and the error lay in the fit function, where one must set the "self_.fitted" attribute to true (2021).

Modeling

Once the data was clean and preprocessed as described above using the inter-movement technique, the QuaterNet architecture was used (Pavllo et al., 2018). This architecture was used to predict other movements, such as walking. Additionally, it was previously trained with the same type of preprocessing described above, switching from postures to the inter- and intra-movement method. Therefore, the first thing that is defined is a benchmark skeleton, which makes sure that the predictions do not go out of range. The proposed architecture is composed of a two-layer gated recurrent unit network (GRU), followed by one linear layer, which was demonstrated in this paper as well as in " On human motion prediction using recurrent neural networks" (Martinez et al., 2017) that performed better. Overall the network inputs the rotations and is trained to predict the future rotations of the skeleton across x timesteps, given n previous frames, and learns using the Adam Optimizer, as used in previous research (Martinez et al., 2017). In this case, the model was evaluated visually, as one of the goals is to create new movements, not precisely the ones given by the videos with which it trained. With this, it is meant that the model is supposed to generate choreographies, not the same ones as in the training set. Therefore the only necessary measure is that movements are feasible by humans, not the sequence itself.

GRU Layers

GRU is a layer for recurrent neural networks (RNN) used for time-dependent tasks, such as the one at hand. The main benefit of RNNs is that they have a memory, which captures past observations and uses them for future predictions. This is relevant for the task at hand, as dancing is time-dependent (related to the audio beat). The previous movement is relevant for predicting the next, as the sequence needs to flow and be feasible by humans.

One of the most common types of RNN layers is the Long Short-Term Memory layer. However, this one is much more complex than the GRU layers, as one needs to calculate more gates (input, forget, and output), which is computationally more expensive and overall more complex. On the contrary, GRUs only have two gates, update and reset, lowering the number of parameters to estimate and, therefore, computational power (Dey & Salem, 2017).

Results

This paper aims to establish the first approach for a standardized framework (Appendix 16) to have the optimal preprocessing of video data combined with the audio data. Nevertheless, this preprocessing framework can be used for any application of video data combined with other extraneous features. For instance, the health industry can use it to detect the rehabilitation of an injury by seeing how a person moves a particular body part in day to day activities, combined with the personal health information of a patient, such as weight, height, previous diseases, etc.

Speaking of choreographies, since they are usually created through the choreographer's creativity others can see them as a random movement. Therefore they can be considered as one of the most complex types of movement. The random movement has mainly three conditions it must fulfill:

1. It must go with the music, meaning it must go with the beat, melody, and overall music flow.
2. It must be feasible, meaning it needs to be able to join one movement with the next fluently.
3. The output must be structured to make it easy for dancers to learn the choreography directly from the output.

In the proposed framework, the main challenges of this kind of data are solved (Appendix 17): coordinate detection through the estimation of the 3D parameter with different video angles or using Google Blaze Pose, missing values which are estimated through low Kalman smoothing, value range, which is reduced through the inter-and intra-movement transformation, combination with extra features through the extraction of features at the same frame rate, and model input preparation done through python TimeSeriesDataSet.

There are currently two possible methods regarding coordinate detection, either calculating the 3D key points or using a model such as Google Blaze Pose. The first method calculates the key points by having the same video filmed from different angles, so all key points are detected. In this case, the AIST++ database was used, which was constructed using that method and did not affect the body's structure (Appendix 18). Combining this with visualizing brings the advantage that the key points are mirrored. While this could be confusing for a viewer, it is much simpler for a dancer to learn choreography, as he/she only needs to replicate movements and not mirror them again, just as in the dance studio. The alternative method is using an algorithm, such as BlazePose, which detects movements automatically. Right now this algorithm is at research level, meaning it is not yet usable by everyone as it depends on the resources one has available. Furthermore, it has not been

studied how it works at the deployment level, as the fine tuning and deployment phase is out of this project's scope.

After having the coordinates, the missing values must be filled. In this case, standard practices such as using averages, medians, or zeros are not possible, as it would distort the position of the human. Using Kalman Smoothing is proposed. This method requires all key points to be present in several frames, but it does not distort the distribution of the key points, adding close to no bias to our data.

Value range is one of the greatest challenges included in this project. This study has proven that standard normalization techniques used in other cases are not successful, as it worsens the performance of models. Therefore the recommendation is to use the inter- and intra- movement technique (Appendix 13).

In the intra-movement, one should select a reference joint and calculate the distance to the other joints, the cosine, and the sine. For the intra-movement, one should also keep the z position, so all three dimensions are still considered. For the inter-movement, one should only predict the position of the reference joint considering the space in which it moves with translation and rotation across the space. This transformation does not affect the body position, as when back-transforming and visualizing, one gets the same position (Appendix 18).

Once all transformed coordinates are collected, one should combine the extra features. In this case we added the extra features of the audio considering time. The audio must be cut at the same frame rate as the frames. For example, when working with a 60FPS dataset (as is the case above), the audio should be cut in 16.67 milliseconds by 16.67 milliseconds windows, and the features should be extracted from those cuts directly (Appendix 14). Once this is done, it should be joined with the transformed coordinate data.

The frame rate is a hyperparameter one should consider altering when modeling. In this case, it was decided to keep it at 60 frames per second because that is how the AIST++ database did it previously and, overall, the standard used in the community.

After having all the necessary features collected, one should standardize the series length. One should see how long the shortest time series is and cut all series to that same length. If the researcher wants to upsample the dataset, these cuts could be overlapping. In our case, the shortest series was 426 frames long, and it was decided to use overlapping windows.

The last step before modeling is structuring the data so the model will accept it as input. For this type of data, using PyTorch is recommended. The TimeSeriesDataSet function should be used. To do the train-test-split as wished, the TimeSeriesDataSet was used. As a result, the MultiNormalizer has been debugged and fixed. Furthermore, to ease the input definition, it is recommended to have each feature in a separate column instead of entering each key point as an array.

The Move2Data preprocessing framework has been executed without any issue and validated in a model already developed and previously trained by Facebook (Pablo et al., 2018), used to predict human movement. The model ran and trained without any issue after doing the necessary debugging to adapt it to our newly standardized data format. Lastly, the model has also been validated by Aitaca (Appendix 19), a startup that provides an AI-based 3D modeling engine that “transforms a full-body length video into an accurate 3D body model with comprehensive measurements” (Aitaca, 2022). Furthermore, the human body representation is mirrored adapting to the needs of the entertainment and media industry regarding dancing. It also considers the main parts of the human body used in this industry.

All of the above, fulfills the aim of this project, to provide the community with a standardized way of preprocessing data helping to build models that will generalize better once they are trained.

After modeling, it is just a matter of back-transforming the output and visualizing the key points, so the choreographers can see the produced choreography and use it for their next show. The original coordinates are estimated with the intra-movement transformation variables (Appendix 20). After, all points can be visualized via matplotlib, and then a video can be created through the OpenCV library.

Discussion

This paper aimed to establish a standardized framework for preprocessing human video data with audio. As shown in the literature review, there is no standardized processing technique. Each paper studies and uses one particular type of processing or uses the data as-is without delivering optimal results. Therefore, this paper took a step further into human movement prediction concerning extraneous variables, particularly music.

As analyzed above, the main issues when dealing with this kind of data are addressed in the end-to-end framework proposed in this paper (Appendix 16). This framework can also allow the open-source community to train and deploy new models, leading to better models for choreography generation as in other cases, such as natural language processing (NLP) models developed by the community and published on open source platforms.

This framework is one of a kind, as no other preprocessing framework for this subject has been found. It was developed out of a necessity when trying to model choreographic movement. Therefore, all previously used techniques have been tested and combined most optimally. This framework has been tried out on a random sample of 10%

of the dataset and it has always mirrored the human's position visually correct. This shows that no bias is being added and the values are simplified.

This advancement could also mean a disruption of the audiovisual and dance industry, as it would be able to generate choreographies at a much faster rate and even create human-computer collaborations, bringing new styles of dance and movement. Additionally, it could also be helpful with other extraneous features, such as body measures and health conditions, for the health industry, diagnosing diseases by seeing how people walk or perform day-to-day tasks after having an injury. This way their improvement and healing could be measured online.

Another interesting application could be in mass control for airports, public transport, or any space in general. This way, a supermarket could decide which kind of music to put as ambiance, so their customers shop quicker or slower. Further applications could also include the digital fitting room of clothing shops, showing the product's fit statically and how the fabric will move as one moves. Lastly, a huge application could be for the metaverse, a" post-reality universe, a perpetual and persistent multiuser environment merging physical reality with digital virtuality" (Azar & Barretta, 2022). In this virtual world, for it to be as authentic as possible, the human avatars will need to move as realistically as possible, and that is where this processing and the potential models built with them will potentially come into play. Apart from these, there are many other future applications we cannot even imagine today.

This preprocessing framework is a good starting point for further development. This first version must be tested by more models and applications, potentially customizing and optimizing some of the steps. However, it has the potential to be among the optimal ones. One must also consider that dance has a lot of creativity involved, that the model needs to

learn how to capture and predict. Moreover, by using this preprocessing, a model will be able to learn possible poses, and with a big enough dataset and suited architecture, it will be able to predict accordingly.

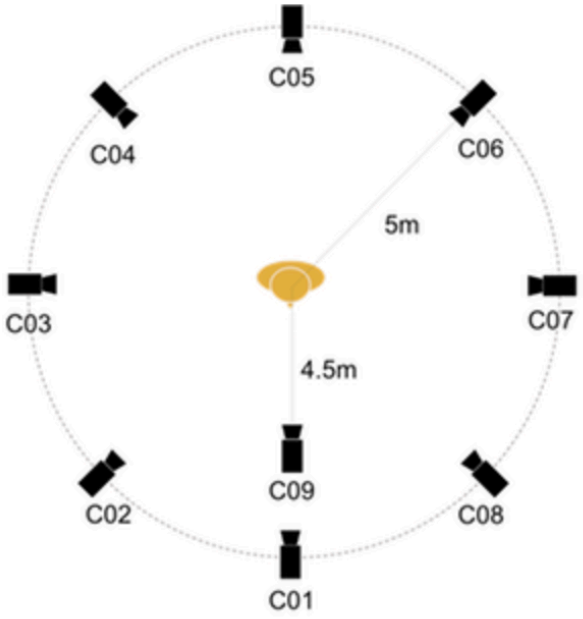
This paper discusses the preprocessing of such data in-depth, leaving future researchers and the whole community the opportunity to use it, create the models, and make the necessary changes for optimal prediction. It is recommended to create an unsupervised model, as the aim is not to predict the same choreography, just feasible and proposed poses limited by human conditions: fitness, health, disease, therapy, etc. Another potential research question could be how to create an error function suited for this problem, creating a body function that measures the possibility of human capabilities performing that same position and movement. Finally, it would be interesting to conduct further research in the 3D coordinate detection, training a much more efficient model. Overall, the three relevant further research questions could be:

1. How can we predict choreographic movement considering audio?
2. How can we measure human movement generation success?
3. How can we detect 3D coordinates from videos more efficiently?

Appendix

Appendix 1: Camera Positions for Videos

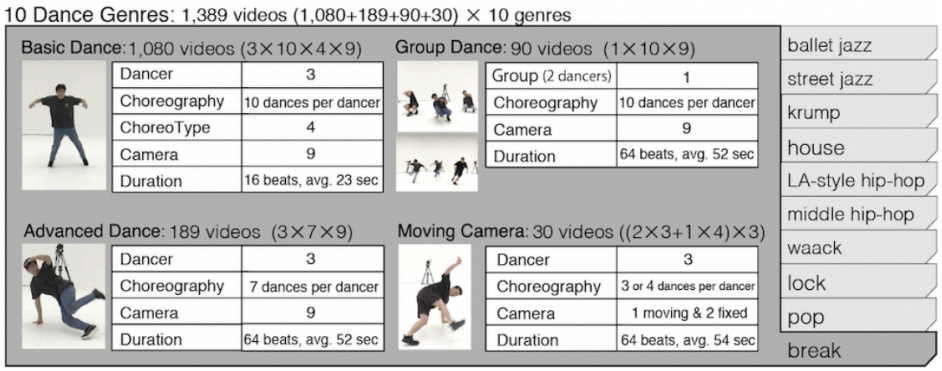
Displayed, one can find the position of the nine cameras filming simultaneously. Except for camera nine, they are five meters away from the dancer (Tsuchida et al., 2018).



Appendix 2: AISTDB Video Groups

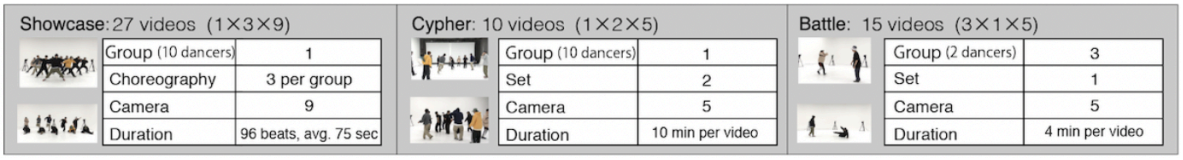
Below one can find the four different groups in which the videos are divided. All videos showing simple moves, with many repetitions and practically no floor work, are considered basic dance. Those with more complex and changing moves are considered advanced dances. If more than one person is dancing simultaneously, it belongs to group dance. Lastly, if the camera is not stable and changes its position, it belongs to the moving camera group. Below, there is only an example for the break genre; however, the distribution is similar for the rest of the genres listed at the right of the image (Tsuchida et al., 2018).

What if we are able to predict movement?



Appendix 3: AISTDB Situations

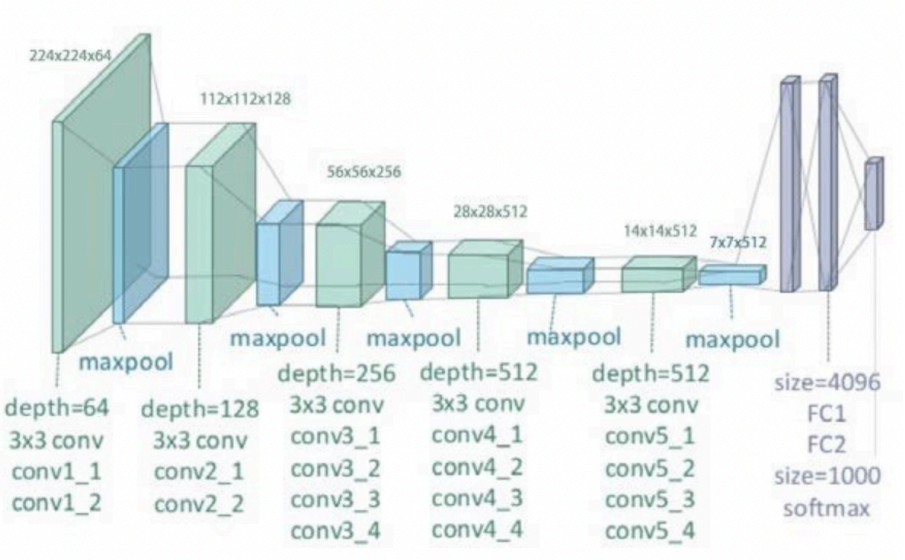
There are three extra situations included in the AISTDB. Only 50 videos belong to one of these situations. The first one is showcase; these videos are group performances that simulate real dancing shows. The second one is cypher. In the dance industry, a cypher is when all dancers stand in a circle and battle when going in the center by freestyling. Lastly, the battle is when two people are dancing against each other, also freestyling (Tsuchida et al., 2018).



Appendix 4: VGG-19 Architecture

Here one can see the VGG-10 architecture, composed of five-layer pairs of CNNs and max pool layers, followed by three fully-connected softmax layers (Zheng et al., 2018).

What if we are able to predict movement?



Appendix 5: Audio Features Definition

- Spectral centroid: weighted mean of the frequencies present in the sound.
- Spectral roll-off: frequency below which a specified percentage of the total spectral energy.
- Zero crossing rate: the rate at which the signal changes from positive to negative or back; higher values for highly percussive sounds like those in metal and rock.
- Spectral bandwidth: variance from spectral centroid.
- Spectral contrast: relative spectral characteristics.
- Spectral flatness: width, uniformity, and noisiness of the power spectrum.
- RMS: average loudness of an audio track.
- MFCC: small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope.

Appendix 6: MANOVA

This MANOVA was run to determine if the relationship between audio features was related to the cluster to which they were assigned. By looking at the p-values, it is

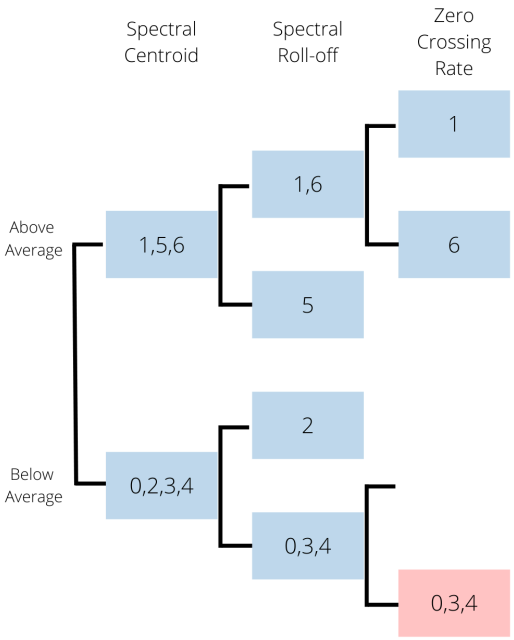
concluded that the relationship is significant. Therefore, it can also be concluded that VGG-19 detects the features used by previous studies.

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value   Num DF  Den DF  F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda   0.0033 8.0000 51.0000 1924.8726 0.0000
         Pillai's trace   0.9967 8.0000 51.0000 1924.8726 0.0000
 Hotelling-Lawley trace 301.9408 8.0000 51.0000 1924.8726 0.0000
    Roy's greatest root 301.9408 8.0000 51.0000 1924.8726 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
             model6         Value  Num DF  Den DF F Value Pr > F
----------------------------------------------------------------
              Wilks' lambda 0.5904 8.0000 51.0000  4.4233 0.0004
             Pillai's trace 0.4096 8.0000 51.0000  4.4233 0.0004
     Hotelling-Lawley trace 0.6939 8.0000 51.0000  4.4233 0.0004
        Roy's greatest root 0.6939 8.0000 51.0000  4.4233 0.0004
================================================================
```

Appendix 7: Audio Clusters Average Difference

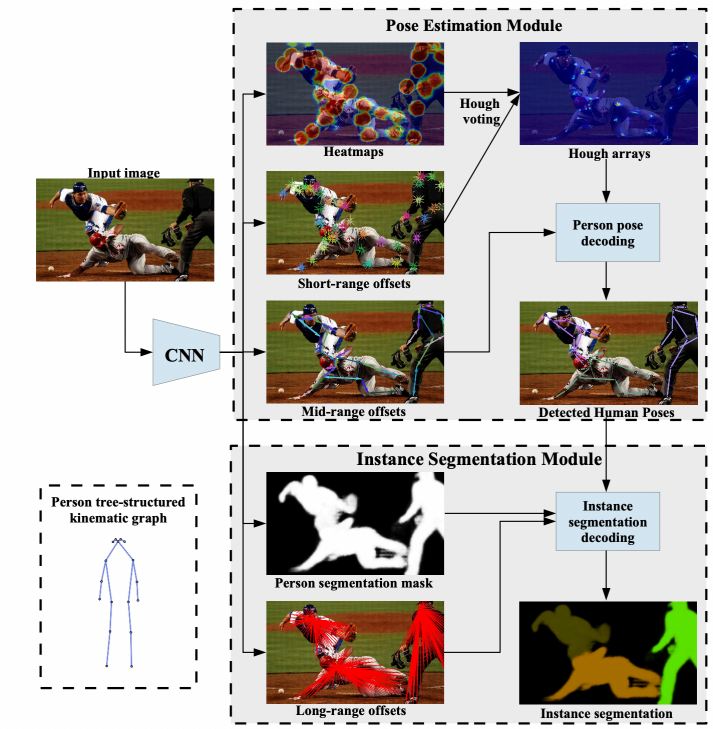
Below one can see how the seven different clusters behave regarding the different variables used in a previous paper. The variables that most divided the audios used in this project are the three shown below: spectral centroid, spectral roll-off, and zero-crossing rate. The numbers inside the boxes indicate the cluster number, and each time they belong to the box above, it means that they are above average and vice versa. As three clusters cannot be divided in terms of these variables, it is confirmed that VGG-19 is extracting more features than those used in previous papers.

What if we are able to predict movement?



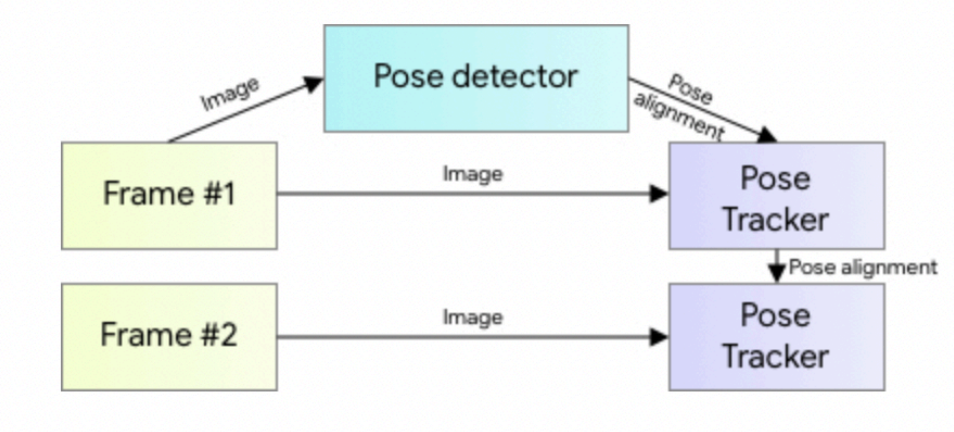
Appendix 8: Pose Estimation Algorithm

This paper focuses on the pose estimation module from the image below, explaining how the algorithm performs the pose estimation and outputs the key points (Papandreou et al., 2018).



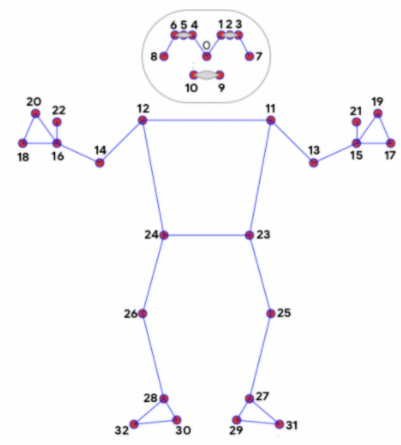
Appendix 9: Google Blaze Pose Pipeline

The pipeline of Blaze Pose works as follows. For the first frame the algorithm detects the object of interest and predicts all 33 key points. The remaining frames estimate the key points from the heatmaps and their corresponding offsets (Bazarevsky & Grishchenko, 2020).



Appendix 10: Google Blaze Pose Key Points

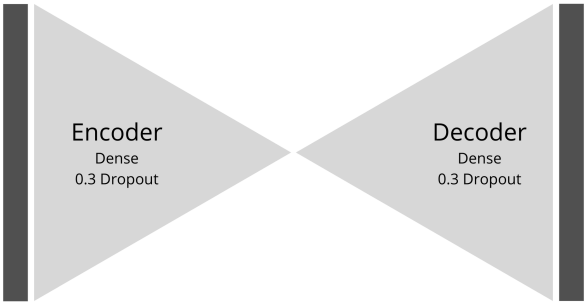
Google Blaze Pose expands the COCO standard (17 key points) to 32 key points. Depending on the task one performs, one may be able to discard some, such as the inner and outer eye corners, leaving only the eyes themselves.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

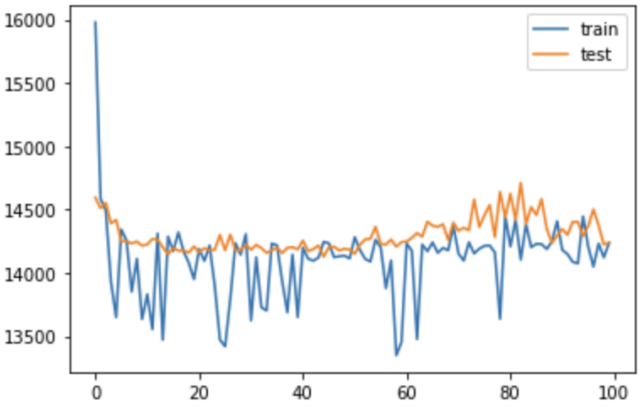
Appendix 11: Autoencoder Architecture

The autoencoder was used to see if it would be possible to reduce the dimensionality of the dataset by encoding and decoding and using the raw dataset with the original 3D coordinates.



Appendix 12: Raw Autoencoders Performance

Below one can see the performance of the autoencoders, measured as mean squared error. Looking at the y-axis, it is concluded that the errors are extremely high. Furthermore, the overfitting decreases. However, the model is still performing poorly.



What if we are able to predict movement?

Appendix 13: Intra Movement Normalization

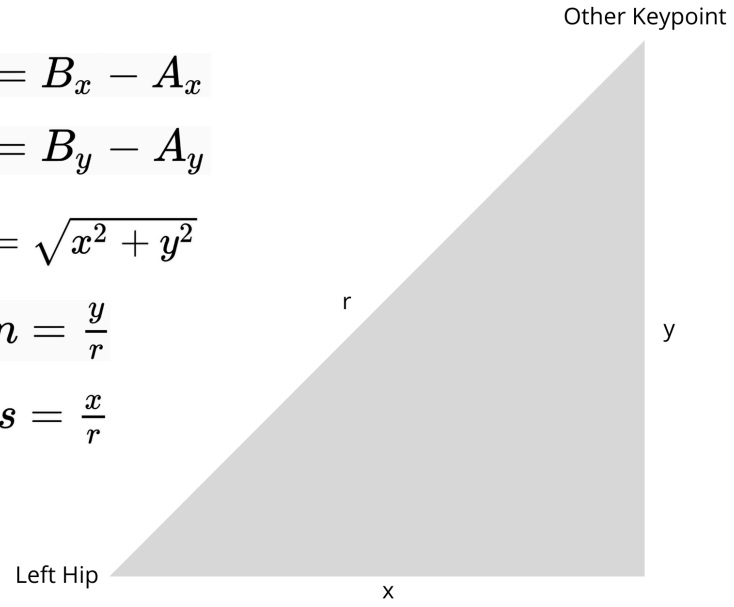
$$x = B_x - A_x$$

$$y = B_y - A_y$$

$$r = \sqrt{x^2 + y^2}$$

$$sin = \frac{y}{r}$$

$$cos = \frac{x}{r}$$



Appendix 14: Audio Window Calculation

Our video data is processed at 60 frames per second, meaning the dataset has one frame each $\frac{1}{60}$ seconds. One second is equivalent to 1,000 milliseconds, meaning $\frac{1}{60}$ seconds is equivalent to 16.67 milliseconds. Therefore the audio window is of 16.67 milliseconds.

Appendix 15: TimeSeriesDataSet Parameters

Below one can find the parameters used for the TimeSeriesDataSet. For the target_normalizer one must input as many normalizers as number of targets one is working with.

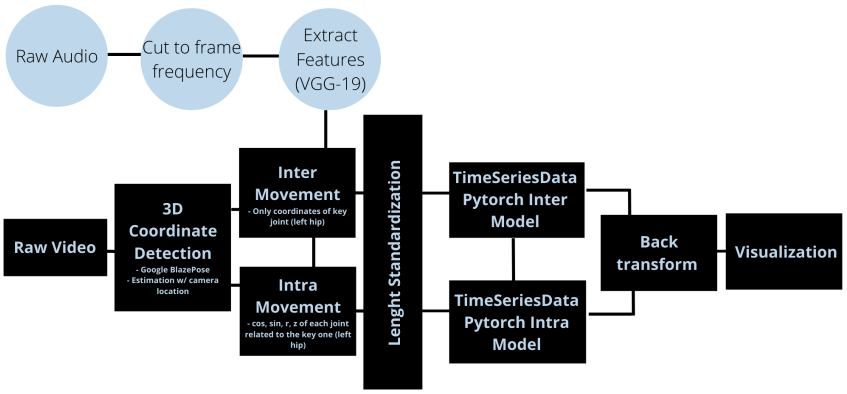
Parameter	**Used Value**
data	d[lambda x: x.index <= d["time"].max() - max_prediction_length]
time_idx	“time”
target	list of all targets
group_ids	“File_group”

What if we are able to predict movement?

time_varying_unknown_reals	list of all lagged values
min_encoder_length	0
max_encoder_length	120
min_prediction_length	1
max_prediction_length	60
target_normalizer	MultiNormalizer([TorchNormalizer(), …])

Appendix 16: Move2Data Preprocessing Framework

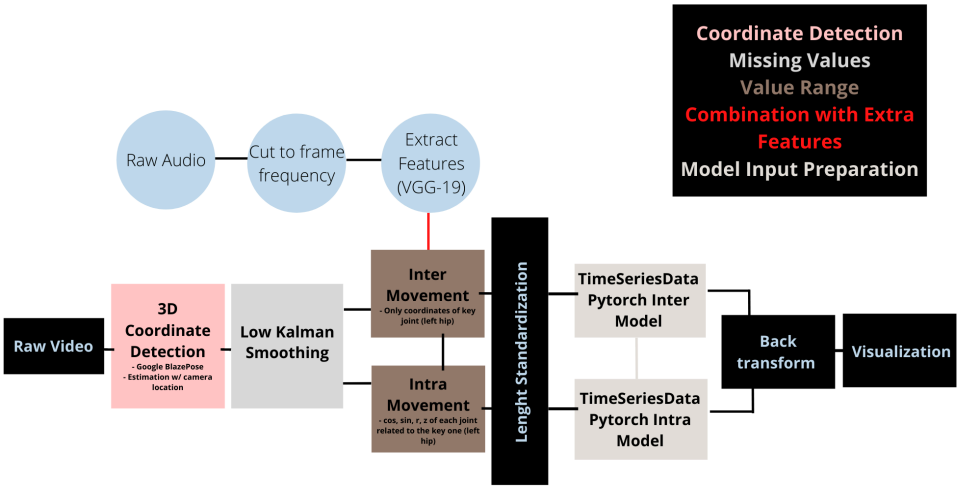
This is the end-to-end preprocessing framework proposed by this study.



What if we are able to predict movement?

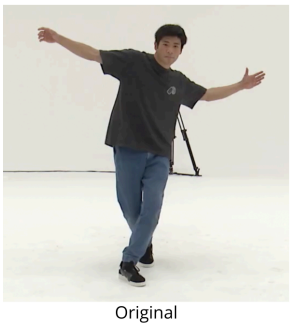
Appendix 17: Move2Data Preprocessing Framework and Challenges Solved

Complementing the preprocessing framework, the challenges solved in each step are color-coded.

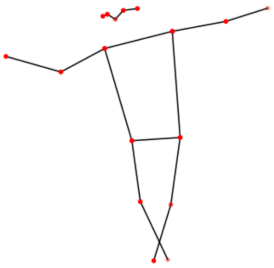


Appendix 18: Impact of Preprocessing Steps on Body

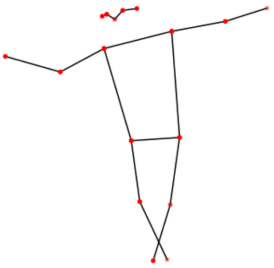
As you can see, the points are mirrored; this will allow dancers to learn the choreographies easier, replicating the same situation as in the dance studio. Apart from that, there are no changes in the dancer's pose from one step to the other.



Original



AIST++



Backtransformed from Inter- and Intra-Movement

Appendix 19: Aitaca Validation

Aitaca is a start-up that provides an AI-based 3D modelling engine that transforms a full-body length video into an accurate 3D body model with comprehensive measurements. Originally the solution was developed for the health sector, mainly to detect malnutrition. We are broadening our scope to more health and fitness applications as well as in the retail industry to reduce the number of returns.

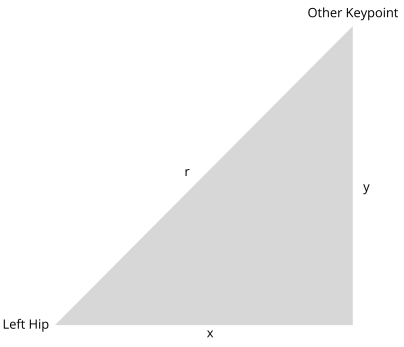
I, Paz Vega, CEO and Co-founder of Aitaca Remote Tech S.L., believe that Move2Data Preprocessing Framework project lays the first fundamental steps towards a much greater project, which will will unveil currently unimaginable possibilities. I'm confident the pre-processing framework provides a robust solution for developing a good model to predict movement. I see an interesting possibility in combining the Framework with our retail application allowing us to visualise how the clothing will move while the person, in the form of an avatar, is doing any day-to-day task. Therefore, I validate the findings of this paper and dare to define them as the right path to the advancement of human movement and pose modelling.

Date: Signature:

VEGA DIEZ-ROLLAN PAZ - 53623672T

Digitally signed by VEGA DIEZ-ROLLAN PAZ - 53623672T
Date: 2022.05.03 13:22:01 +02'00'

Appendix 20: Intra-movement Back Transformation



$$x = cos * r$$

$$y = sin * r$$

$$A_x = B_x + x$$

$$A_y = B_y + y$$

Appendix 21: Move2Data Github Repository

On this repository you can find the defined functions to go through the entire framework. Additionally, there is a tutorial on how to use the functions.

https://github.com/roglantero/Move2Data

Resources

Aitaca Remote Tech SL. (2022, January 24). Aitaca. Retrieved April 30, 2022, from https://aitaca.io/

Alemi, O., Françoise, J., & Pasquier, P. (2017). GrooveNet: Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks∗. https://doi.org/https://doi.org/10.475/123_4

Azar, A. T., & Barretta, R. (2022). Metaverse. Encyclopedia. https://doi.org/https://doi.org/10.3390/encyclopedia2010031

Barbicˇ, J., Safonova, A., Pan, J.-Y., Faloutsos, C., Hodgins, J. K., & Pollard, N. S. (2004). Segmenting Motion Capture Data into Distinct Behaviors. Proceedings of Graphics Interface 2004.

Bazarevsky, V., & Grishchenko, I. (2020, August 13). On-device, Real-time Body Pose Tracking with MediaPipe BlazePose [web log]. Retrieved February 23, 2022, from https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html.

Beitner, J. (2020). PyTorch Forecasting Documentation. Retrieved from https://pytorch-forecasting.readthedocs.io/en/stable/index.html

Burke, M., & Lasenby, J. (2016). Estimating missing marker positions using low dimensional Kalman smoothing. Journal of Biomechanics, 49(9), 1854–1858. https://doi.org/10.1016/j.jbiomech.2016.04.016

Crnkovic-Friis, L., & Crnkovic-Friis, L. (2016). Generative Choreography using Deep Learning.

De Maesschalck, R., Jouan-Rimbaud, D., & D.L. Massart. (2000). The Mahalanobis distance. Chemometrics and Intelligent Laboratory Systems, 50.

Dey, R., & Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS). https://doi.org/10.1109/mwscas.2017.8053243

Gopinath, D., & Won, J. (2020). fairmotion - Tools to load, process and visualize motion capture data. Opgehaal van https://github.com/facebookresearch/fairmotion

Kao, H.-K., & Su, L. (2020). Temporally Guided Music-to-Body-Movement Generation.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes.

Li, R., Yang, S., Ross, D. A., & Kanazawa, A. (2021). Ai Choreographer: Music conditioned 3D dance generation with aist++. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv48922.2021.01315

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., & Black, M. J. (2015). SMPL. ACM Transactions on Graphics, 34(6), 1–16. https://doi.org/10.1145/2816795.2818013

Martinez, J., Black, M. J., & Romero, J. (2017). On human motion prediction using recurrent neural networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://doi.org/10.1109/cvpr.2017.497

Meredith, M., & Maddock, S. (n.d.). Motion Capture File Formats Explained.

Multinormalizer always raise notfittederror, even when it was actually fitted - python pytorch-forecasting. GitAnswer. (2021). Retrieved April 19, 2022, from https://gitanswer.com/multinormalizer-always-raise-notfittederror-even-when-it-was-actually-fitted-python-pytorch-forecasting-999993218

Onuma, K., Faloutsos, C., & K. Hodgins, J. (2008). FMDistance: A fast and effective distance function for motion capture data. The Eurographics Association 2008.

Papandreou, G., Zhu, T., Chen, L.-C., Gidaris, S., Tompson, J., & Murphy, K. (2018). PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. 33rd Conference on Neural Information Processing Systems.

Pavllo, D., Grangier, D., & Auli, M. (2018). QuaterNet: A Quaternion-based Recurrent Model for Human Motion.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion , B., Grisel, O., Blondel, M., Prettenhofer , P., Weiss, R., Dubourg, V., Vanderplas , J., Passos , A., Cournapeau , D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011) 2825-2830.

Ramaswamy, J. (2020). What makes the sound?: A dual-modality interacting network for audiovisual event localization. ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). https://doi.org/10.1109/icassp40776.2020.9053895

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany.

Siegler, M. A., Jain, U., Raj, B., & Stern, R. M. (n.d.). Automatic Segmentation, Classification, and Clustering of Broadcast News Audio. ECE Department - Speech Group Carnegie Mellon University Pittsburgh, PA 15213.

Simoyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR 2015.

Tsuchida, S., Fukayama, S., Hamasaki, M., & Goto, M. (2018, November 4). AIST Dance Video Database: Multi-genre, Multi-dancer, and Multi-camera Database for Dance Information Processing. Retrieved from https://aistdancedb.ongaaccel.jp/.

Wyse, L. (2017). Audio spectrogram representations for processing with Convolutional Neural Networks. National University of Singapore.

Zheng, Y., Yang, C., & Merkulov, A. (2018). Breast cancer screening using convolutional neural network and follow-up Digital Mammography. Computational Imaging III. https://doi.org/10.1117/12.2304564