A new framework for the evaluation of locomotive motion datasets through motion matching techniques

Vicenzo Abichequer Sangalli, Ludovic Hoyet, Marc Christie, Julien Pettré

▶ **To cite this version:**

Vicenzo Abichequer Sangalli, Ludovic Hoyet, Marc Christie, Julien Pettré. A new framework for the evaluation of locomotive motion datasets through motion matching techniques. MIG 2022 - ACM SIGGRAPH Conference on Motion, Interaction and Games, Nov 2022, Guanajuato Mexico, Mexico. pp.1-10, 10.1145/3561975.3562951 . hal-03865727

HAL Id: hal-03865727
https://inria.hal.science/hal-03865727

Submitted on 22 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A new framework for the evaluation of locomotive motion datasets through motion matching techniques

Vicenzo Abichequer Sangalli
vicenzo.abichequer-sangalli@inria.fr
Inria, Univ Rennes, CNRS, IRISA
France

Ludovic Hoyet
ludovic.hoyet@inria.fr
Inria, Univ Rennes, CNRS, IRISA
France

Marc Christie
marc.christie@irisa.fr
Univ Rennes, Inria, CNRS, IRISA
France

Julien Pettré
julien.pettre@inria.fr
Inria, Univ Rennes, CNRS, IRISA
France

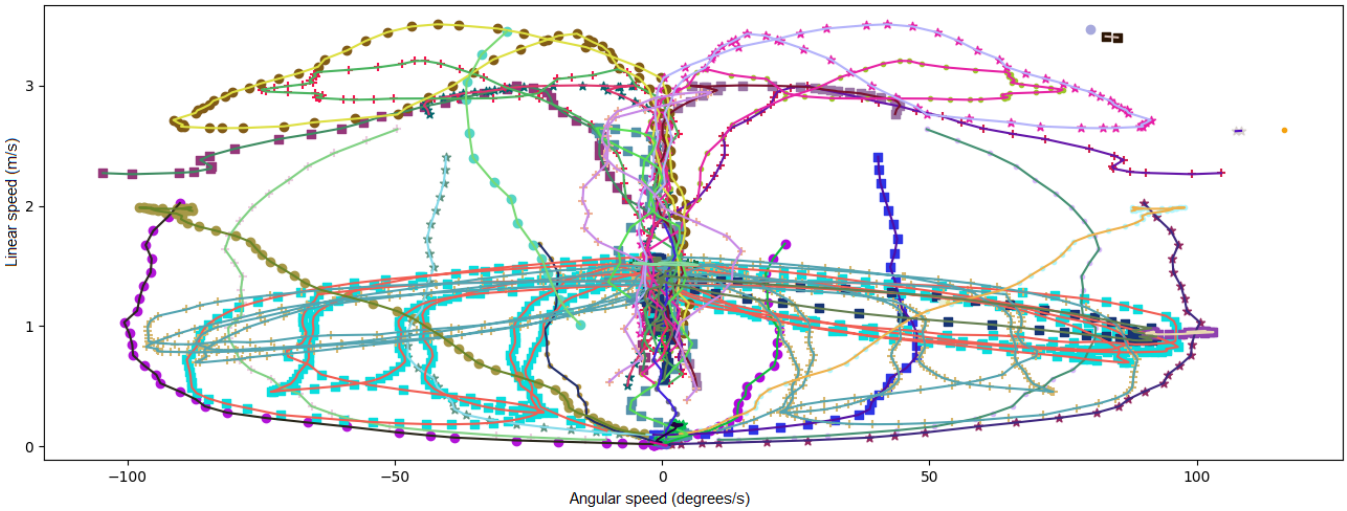

Figure 1: The many different motions that compose one of the datasets. Each motion clip is represented in a different color and symbol. Such representations are exploited to give the user an idea of how motions are distributed in the dataset.

ABSTRACT

Analyzing motion data is a critical step when building meaningful locomotive motion datasets. This can be done by labeling motion capture data and inspecting it, through a planned motion capture session or by carefully selecting locomotion clips from a public dataset. These analyses, however, have no clear definition of coverage, making it harder to diagnose when something goes wrong, such as a virtual character not being able to perform an action or not moving at a given speed. This issue is compounded by the large amount of information present in motion capture data, which poses a challenge when trying to interpret it. This work provides a visualization and an optimization method to streamline the process of crafting locomotive motion datasets. It provides a more grounded approach towards locomotive motion analysis by calculating different quality metrics, such as: demarcating coverage in terms of both linear and angular speeds, frame use frequency in each animation clip, deviation from the planned path, number of transitions, number of used *vs.* unused animations and transition cost.

By using these metrics as a comparison mean for different motion datasets, our approach is able to provide a less subjective alternative to the modification and analysis of motion datasets, while improving interpretability.

CCS CONCEPTS

• **Computing methodologies** → **Motion processing**; • **Human-centered computing** → **Visualization design and evaluation methods**.

KEYWORDS

animation, data visualization, data analysis, motion capture data, feature extraction

ACM Reference Format:
Vicenzo Abichequer Sangalli, Ludovic Hoyet, Marc Christie, and Julien Pettré. 2022. A new framework for the evaluation of locomotive motion datasets through motion matching techniques. In *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '22), November 3–5, 2022,*

MIG '22, November 3–5, 2022, Guanajuato, Mexico
© 2022 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '22), November 3–5, 2022, Guanajuato, Mexico*, https://doi.org/10.1145/3561975.3562951.

Guanajuato, Mexico. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3561975.3562951

1 INTRODUCTION

In processes that rely on motion datasets to perform animations, a relationship can be frequently found between the quality of a dataset and the output of the process. This relationship is important in many contexts, such as motion capture-based animation techniques, which is the one that interests us in this paper. However, quantifying the quality of a motion dataset is not a straightforward process. The typical process when trying to evaluate whether one has the correct motion dataset for a given scenario is to simulate the scenario, judge its perceptual quality and search through the whole dataset, trying to figure out if the gathered motions are adequate candidates. This is in general manually performed by animators, resulting in a subjective analysis.

Another major concern is, after analysing the dataset, what happens if the user adds or removes data. There is no objective way to quantify that the modifications made to the dataset had an impact, be it positive or negative. For example, a motion dataset might contain motions that are not used or might be lacking other motions: if the character needs to perform sharp turns and the dataset is tailored to slower turning velocities, then the character might not be able to turn as effectively. Similarly, if a character tries to run faster than what the currently available motions are capable of delivering, the dataset would need to be improved with faster running motions. Without having a bigger perspective related to the motion dataset, any modification becomes a cumbersome trial-and-error process.

This work therefore investigates how to optimize locomotive motion datasets used for animating virtual characters. To tackle this problem, an approach that considers motion matching, while also being independent of a specific animation technique, is proposed. The approach is also based on a number of novel metrics that quantify accuracy, coverage and relevance of a whole locomotive motion dataset. It also aids the user in performing more accurate optimizations, saving on memory footprint and making sure that every planned scenario is covered. To achieve this, this approach relies on motion matching to simulate different animation conditions while gathering evaluation data. This is later processed and compiled into an evaluation of the resulting animations. The main contributions of this paper are:

- a more robust and quicker way to make informed choices when analyzing locomotive motion datasets;
- a formal analysis cycle to evaluate the impact of the removal or addition of locomotive motions;
- reproducible tests to allow the evaluation and reversibility of changes.

2 RELATED WORKS

Motion datasets used to animate virtual characters are generally built using motions captured on real actors or manually created by animators. To efficiently control the locomotion of virtual characters, it is recommended to plan motion capture session, e.g. using dance cards as proposed by Zadziuk [2016]. If one does not have the ability to record motions, public datasets are also available, such as Adobe's Mixamo[1] and Carnegie Melon University's motion database[2]. No matter the way a dataset is established, modifying it, judging when a specific motion should be used or how to sequentially combine multiple motions can all be cumbersome jobs, as it requires manual visual analysis of the motions.

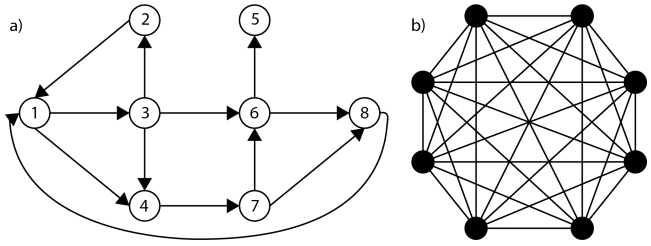

Figure 2: a) An example of a motion graph. b) An example of a motion matching graph.

To solve this issue, Kovar and Gleicher [2002] and Lee et al. [2002] proposed a structure called motion graphs, with the main objective of giving a wide view of how similar some motions are to each other and how they can be interlinked (see Figure 2.a). The structure is laid out as a strongly connected graph, where each node is a frame and each edge a potential transition to another frame of the dataset. By contrast, on Lee et al's work [2010], this assumption is taken as the opposite, where the edges are motions and the nodes are transitions. This means that the graph can be set either way. By the layout of the graph, some assumptions can be made, such as needing more motion clips when you have dead ends or to connect other weakly connected components, to avoid a logical motion discontinuity. One of the main advantages of approaching motion data using motion graphs is that the animator is able to create new motion from motion fragments, as in [van Basten and Egges 2009], by connecting similar poses to other parts of the dataset. Various extensions of the motion graphs approach were proposed by the community, e.g., fat graphs [Shin and Oh 2006], parametric motion graphs [Heck and Gleicher 2007], motion motif graphs [Beaudoin et al. 2008], motion graph++ [Min and Chai 2012].

Building on motion graphs, motion matching was recently proposed [Clavet 2016; Clavet and Buttner 2015]. As opposed to motion graphs, the approach does not have a concept of dead ends, and instead introduces a structure in the form of a fully connected graph (see Figure 2.b). As for motion fields [Lee et al. 2010], it introduces a pre-processing step to extract poses, but the responsiveness is maintained and the performance is enhanced, bringing a solution to its scaling problem, while also going back to a graph structure, simplifying its analysis. In a nutshell, motion matching enables potential transitions to occur between all poses of the graph at any evaluation period, and not only between preselected transitions. The core aspect is to search in the database of poses the one best fitting a set of features specified by the user, and to transit to this pose (while blending to accommodate for slight differences) if this pose has a lower cost than the next pose in the current clip.

[1] https://www.mixamo.com/
[2] http://mocap.cs.cmu.edu/

Several authors have then aimed to improve the motion matching approach, to address some of the core limitations in terms of memory usage and computation [Buttner 2019; Holden 2018; Holden et al. 2020, 2017, 2016]. In particular, Holden et al [2018; 2017] have shown that "motion matching is a special case of machine learning for animation where a Nearest Neighbor Regression is used to map from user controls to animation data".

2.1 Quality measurement

While the main works analyzed in the previous section [Buttner 2019; Clavet and Buttner 2015; Holden 2018; Holden et al. 2020, 2017, 2016; Lee et al. 2010; Reitsma and Pollard 2007] all conclude that their methods are limited by the quality of the input data, there is still not a clear way to evaluate the quality of such input motion dataset, as it is often a subjective process and many motion datasets can perform a given task with varying degrees of success and realism.

As it is difficult to predict the quality of animations resulting from motion graphs, Reitsma and Pollard [2007] proposed a number of task-based metrics evaluating a motion graph capacity to produce required animations. To achieve this, one of the steps is comprised of embedding all possible directions a character could take into a grid, which overlaid the navigable surface and took obstacles into account. Lee et al.'s work [2010] goes one step further by embedding the possible poses of a character and mapping them to motions. It then finds the one that fits best and follows it for a single frame, repeating the process on the next frame. This is accomplished by a reinforcement learning (RL) algorithm which provides immediate responsiveness, at the cost of being both resource-hungry (even when benefiting from data compression) and computationally heavy (due to the frequent use of kNN searches).

Apart from the work of Reitsma and Pollard [2007], most of the work on quality measurements focus on comparing animations directly. For instance, Hachaj and Ogiela [2020] use Dynamic Time Warping (DTW) to align and compare motion capture segments and trajectories. Similarly, Switonski et al. [2019] use DTW with a minimum distance classifier and different joint selection strategies to classify different motions, based on angular, velocity and acceleration features. Interestingly, they report that Euler angles are more efficient for analyzing raw data, while quaternion representations are more efficient when considering velocities and accelerations. Motion analysis and classification has also been explored using deep learning and long short-term memory (LSTM) neural network [da Silva et al. 2019], providing the ability to automatically label motion capture data with up to 95% accuracy. However, the principal objective of this work was to provide solutions for gathering motion capture data from other projects, in order to cut costs and increase productivity, and not to evaluate the overall quality of the resulting datasets.

Dirafzoon et al. [2016] propose a set of low dimensional features to classify motion data through the use of topological data analysis. Their approach uses the extraction of point clouds, and their reasoning is that by reconstructing the phase space of a system through the analysis of their time series, they can focus on persistence diagrams and use it as features for motion classification. By processing the point cloud, they are able to extract its maximum persistence, which is representative of the range of a joint motion. With that, they feed many instances of the maximum persistence metric to a k-NN classifier and are able to classify each window of motion into a set of individual activities (cycling, golfing, walking, waving and sitting) with over 97% accuracy.

While tracking systems provide information about the specifics of their equipments, most vendors do not provide much about the recognition, post-processing or the final motion data quality. Manns et al. [2016] try to shed some light over this matter by proposing a set of metrics that show the amount of information present in a motion dataset. Through the use of the Principal Component Analysis (PCA) technique, in conjunction with a Shannon entropy-based measurement, the system is able to calculate the uncertainty contained in the motion data, which translates into the amount of diversity in the input datasets. Even though they conclude that more work needs to be done in this area to prove the robustness of these metrics, they are able to classify different sets of motion capturing equipment and tell how well they adapt to different data-driven motion synthesis approaches.

van Basten and Egges [2009] evaluate three distance metrics for concatenating and blending animations: the difference between joint-angle values, point-clouds and PCA. Each of those generated a different motion graph and the authors performed tests to quantify the occurrence of artifacts, such as foot skating and path deviation, with these metrics. For foot skating, the point-cloud based approach was the best one, followed by the PCA approach, while for path deviation the joint-angle approach is generally the best one. The time to run a branch-and-bound algorithm through each graph was also considered, which showed that the point-cloud approach is significantly slower by generating highly-connected subgraphs, with the joint-angle approach resulting in the fastest case. Each metric has its advantages, but in the user study performed by the authors, comparing the motions generated by each metric to the ones from the ground truth, no metric was able to approach the grades given to the ones in the ground truth.

2.2 Optimizing motion datasets

When working with datasets, it is common practice to optimize it in order not only to reduce resource costs and processing time, but also to give a result that is not tainted by bad data. Such examples can be widely seen in the machine learning community with methods such as data pruning: in an evaluation performed by Saseendran et al. [2019], they list the impact on a machine learning algorithm's performance by removing certain samples from a dataset. There is also a technique called data wrangling, explored in Furche et al. [2016], which tries to better understand a dataset to deliver the cleanest and most useful part of a bigger dataset to an application that requests it.

As can be demonstrated by the needs of the machine learning community, optimizing datasets is one of the core principles of working with data. Specifically, when working with motion data, selecting the correct set of motions has also been a long investigated subject [Holden 2018; Holden et al. 2020, 2017, 2016; Kovar et al. 2002; Lee et al. 2010; Reitsma and Pollard 2007] that has shifted considerably, for a over two decades, in respect to interpretability, goals and performance. The main issue, however, is that all these

methods assume the quality of the dataset and there is no clear method to evaluate, visualize and optimize a whole motion dataset. This means that no matter how good a technique is, if the culprit is the data, there is no easy way of knowing this without shoveling through a huge amount of information.

2.3 Structure of a motion dataset

As one can see in the wild [Adobe 2022; Krebs et al. 2021; Lab 2000], motion datasets can be quite different in their gathering methods and purposes. Establishing a unique structure, consequently, is a difficult process, as there is a multitude of capturing equipment from different manufacturers, such as Vicon, Qualisys, Xsens, Nokov and OptiTrack. Not only will their capturing methods be different (some use cameras and others use inertial measurement units), the characteristics of the data generated by each of those systems also will. This means that there is no unique structure, but more so of a general guideline and the adaptations that go with it for the different purposes that data can have. Thus, the usual capturing strategy, as noted on the above-cited datasets, is to use the joints in the human body as capturing points: since bones will not bend or move, the only moving parts in the human body are the joints. This becomes an effective way of recording motion, as each bone can be represented as a static line connected by a joint, with the full set of joints representing the entirety of the human body.

2.4 Situation

Although a number of approaches addressed some of the limitations inherent to using motion datasets, such as responsiveness [Lee et al. 2010], performance [Clavet and Buttner 2015] or memory footprint [Holden et al. 2020], being able to interpret results obtainable using such motion datasets is still an open question. In particular, early work on motion fields [Lee et al. 2010] was difficult to interpret, as it was not a graph structure, which is well understood and extensively researched. They specifically noted that new tools should be built to aid analyzing, editing and authoring capabilities for new motion fields animation controllers. Subsequently, motion matching reconciled motion fields and graph structures, which are easier to interpret, unlike learned approaches [Fan et al. 2021; Holden et al. 2020].

The objective of this work is therefore to facilitate interpretation of such datasets by proposing a novel framework to analyze and identify the problem of lack of quality in motion data. It proposes new metrics to qualify a dataset and quantify whether motions are missing and, if so, how much and which type of it. By giving the user a better understanding of the limitations, possibilities and redundancies of a motion dataset, this work tries to endow the user with a tool to better evaluate a motion dataset on both macroscopic and microscopic perspectives.

3 A NOVEL ANALYSIS FRAMEWORK

3.1 Overview

This work proposes a framework to formalize the qualitative analysis of a locomotive motion dataset in three ways: accuracy, coverage and relevance (detailed in Section 3.3). As for the first one, the main objective is to find how much positional and velocity errors are introduced by the dataset. As for coverage, the aim is telling how sparse/dense a dataset is, thus showing where more motion is needed or can be removed. As for the latter, the relevance metric shows the usage frequency of each motion frame and a graph structure showing the relationships between all motion clips and frames.

To operate, the framework relies on a reproducible procedure (detailed in Section 3.4), currently tailored for locomotive motions. This procedure is based on a parameterization across a spectrum of linear and angular velocities, leading to a number of circular trajectories that the virtual character has to follow and which enable the evaluation of different datasets under the same conditions. Virtual characters are animated using the Motion Matching approach (introduced in Section 2), which works under the principle that every pose in the dataset is a potential transition at every frame. In the implementation of Motion Matching that we use, the criteria for transition is called *transition cost* and the lowest value is always selected. This *transition cost* is computed as the cumulative distance over a selection of joints between the current pose and the candidate pose. While our approach could be used with any animation method, Motion Matching was chosen for its responsiveness properties, and for the fact that it is now established as an industrial state-of-the-art standard in character animation.

3.2 Data collection and generation

To analyse a dataset, our approach takes a set of animation files and a few parameters to evaluate as input, such as the reference linear and angular speed intervals, represented by v_{ref} (m/s) and ω_{ref} $(degrees/s)$, respectively. The v_{ref} and ω_{ref} values are taken from two intervals, which are defined according to the sets $\{v_{ref} \in \Re | v_1 \leq v_{ref} \leq v_n\}$ and $\{\omega_{ref} \in \Re | \omega_1 \leq \omega_{ref} \leq \omega_n\}$. At each trial, one $(v^k_{ref}, \omega^k_{ref})$ pair is taken from the intervals as starting variables for a simulation. Then, some variables are recorded or computed as follows:

3.2.1 Positional data. Any trajectory in the system will be composed of 2-dimensional positions (on the X-Z horizontal plane). Considering that, when starting a simulation, the system will have a reference (v_{ref}, ω_{ref}) pair, which will be used to generate a reference trajectory, with constant (or piecewise constant) speed profile. This planned trajectory T_{ref} acts as a guide and the final trajectory T may drift from it due to the motions used in the simulation.

3.2.2 Rotational data. As is the case with positional data, a set of instantaneous orientations $\{\theta_y\}$ of the character is stored at the same time, representing the overall orientation of the character around the vertical Y axis.

3.2.3 Speed data. The positional and rotational data, along with the time between each frame (t), are used to compute the recorded linear speed (v_r) and recorded angular speed (ω_r) at each time step.

3.2.4 Statistical data. The data from the motion matching algorithm is also gathered from the simulation at every frame, such as which animation pose and clip from the dataset is being used, its status (if it is being replaced, replacing an older clip or being kept in use), as well as the cost for transitioning from every pose into another.

3.3 Quality measurement

These measurements are then processed to qualify a dataset in three ways: accuracy, relevance and coverage.

3.3.1 Accuracy. The accuracy of a dataset can be evaluated using a couple of metrics, which account for the behavior of linear and angular speeds. The first set of metrics is the two static errors SE_v and SE_ω, which correspond to the difference of the mean v_r and ω_r to the reference v_{ref} and ω_{ref} over the whole duration of the simulation, defined as

$$SE_v = \frac{1}{m}\sum_{i=0}^{m} v_r - v_{ref} \tag{1}$$

$$SE_\omega = \frac{1}{m}\sum_{i=0}^{m} \omega_i - \omega_{ref} \tag{2}$$

The second set of metrics is the dynamic speed errors DSE_v and DSE_ω, which represent how spread away from the mean the speeds are:

$$DSE_v = \sqrt{\frac{\sum(v_r - SE_v + v_{ref})^2}{card(\{v^k_{ref}] \in \Re | v_1 \leq v_{ref} \leq v_n\})}} \tag{3}$$

$$DSE_\omega = \sqrt{\frac{\sum(\omega_r - SE_\omega + \omega_{ref})^2}{card(\{\omega \in \Re | \omega_1 \leq \omega_{ref} \leq \omega_n\})}} \tag{4}$$

The third metric is the offset error, measured in meters, which is calculated as the absolute euclidean distance from the final trajectory (T) to the reference trajectory (T_{ref}). This metric is used to calculate the drift mentioned in Section 3.2.1, which is the absolute difference between what would be the optimal trajectory vs. what the final trajectory T is. It is defined as,

$$\begin{aligned} \text{given } T &= p_1, p_2, ..., p_n \\ \text{and } p_i &= (x_i, z_i) \\ \text{then } offset &= \sum_{i=1}^{m} ||p_i - p_{ref,i}|| \end{aligned} \tag{5}$$

All of these metrics will be shown as graphs, where one can judge how frequent and large the errors are.

3.3.2 Relevance. Considering that a dataset can also be optimized by removing frames or entire clips that are not used or redundant, calculating metrics to measure the relevance of said data is important. This work does that by calculating three metrics: transition cost, frame popularity and animation trajectory.

First, a transition cost metric, based on joint distance, is introduced. It stores the cost involved in transitioning to and using the next frame in an animation. Considering that in the Motion Matching approach there is a pose transition at every frame and that the pose most similar to the current one, in all the dataset, is always chosen for it, then the higher the distance in the joints from the chosen pose to the current one, the higher the transition cost.

Concurrently, we also propose to visualize the relevance of animation frames using a directed graph structure where both frame popularity and animation trajectory are presented. We believe this is particularly important given the difficulty of visualizing a large amount of data, especially when trying to understand how many frames of an animation were utilized and how frequently they were used. In this representation, each animation clip is plotted as a circumference, with its frames as smaller circles at the borders. The larger the frame circle, the more frequently a frame was accessed. Also shown in this graph is the sequence of frames, between the different clips, that were used when the character was being animated. This is displayed as arrows between frames, forming a walk in the directed graph. Figure 3 shows an example of the directed graph structure.

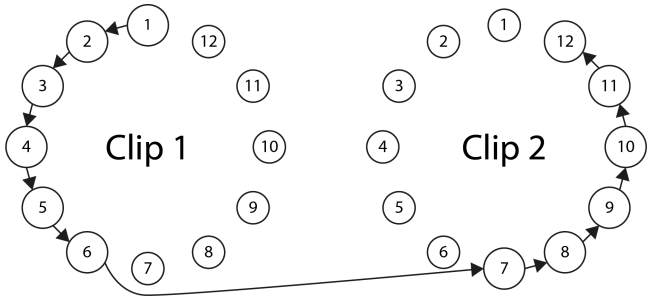

Figure 3: Both circles, surrounded by smaller circles, represent individual clips, with the smaller circles being individual frames. The arrows represent the walk taken between the frames in the two clips. The circles representing the frames used in this walk will also be larger due to their popularity.

3.3.3 Coverage. Similarly, as it is usually difficult for users to have an idea of how well a given motion dataset covers some important parameters that are required for a specific application, we provide users with the ability to generate coverage maps. In the case of locomotive motions explored in this paper, the coverage map is constructed in the v and ω domains, and shows how sparse a given dataset is. This is done so that the evaluator can have a general insight over the linear and angular speed coverage of a dataset. In order to filter out noise, this map is constructed by computing the mean v_r and ω_r over a 1-second sliding window through the dataset.

3.4 Procedure

To compute the different metrics representative of a motion dataset, our framework relies on a reproducible procedure, currently tailored for locomotive motions. This procedure is comprised of three tasks: setting up a scenario, running simulations and data post-processing.

To evaluate how a locomotive dataset behaves across a spectrum of (v_{ref}, ω_{ref}) pairs, we propose that each pair parameterizes a circular trajectory that the virtual character has to follow. Such a circular trajectory enables the evaluation of different datasets under the same conditions, standardizing the evaluation of the limitations of each dataset. To ensure that the virtual character reaches the circle at full speed, the first step is to make the virtual character move towards the tangent of the test circle and start turning with constant linear and angular velocities v and ω once it reaches it. Moreover, we also propose a second type of scenario to characterize locomotion datasets in terms of transitional capabilities, where the linear and angular velocities are changed over time during

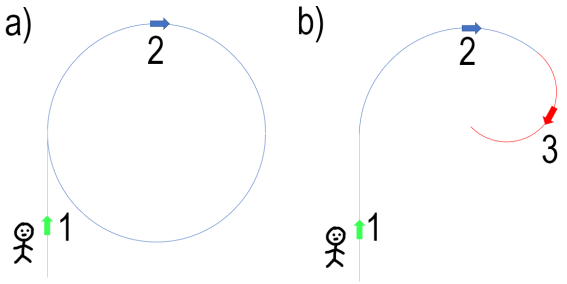

Figure 4: Example a) shows a simulation where a single speed is kept throughout the whole time. Example b) shows a simulation where there is a change of speed from section 2 to 3. The first section is always the same.

the simulation. In this second case, the scenarios are defined by a starting speed (v_{ref}), a switching speed (v_{sw}) and a time interval (t_{sw}). With these parameters, the simulation will switch the speed of the character between v_{ref} and v_{sw} every t_{sw} seconds. By changing v over time, the user can therefore test the responsiveness of the virtual character and the time it takes for it to react when there is a change in speed, aiding the user in the evaluation of transitions. Figure 4 shows an example of such simulations.

Since v and ω will both be fixed during some part of each simulation, it is important to note that the radius of the circle will vary to allow for such speeds.

Since the linear speed can also be seen as the time is takes to progress through the length of an arc,

$$v = \frac{radius \times \theta}{t} \tag{6}$$

one can define the relationship between v and ω as,

$$v = radius \times \omega \tag{7}$$

Then, during the simulation the data necessary for computing the different metrics is saved for each frame, as mentioned in Section 3.2.

4 EVALUATION

In this section, we build on the metrics presented in Section 3 to test the capabilities of different datasets and explore their limitations. Therefore, comparisons between different motion datasets, as well as potential directions for optimizing them are presented.

4.1 Technical details

All simulations were driven by Unity 2019.4.35f, using a modified version of the Motion Matching for Unity (MxM) plugin as the implementation for motion matching. All experiments were run in a computer equipped with an Intel Core i7-9750H CPU, 32GB of RAM and an Nvidia 2070 Max-Q. Post-processing analyses of the simulation data were performed using Python 3.10.

Three motion datasets are used: the first one is comprised of 37 different motion files from Mixamo (total of 6108 frames at 30Hz), another one is comprised of 129 different motion files from Unity's Raw Mocap Data for Mecanim dataset (total of 78897 frames at 30Hz) and the last one is a whole 5 minutes file of motion data recorded at the authors' lab with an Xsens full-body suit (total of 37031 frames at 120Hz).

4.2 Results

First, the framework's ability to interpret the accuracy of three different datasets, i.e. the ability to obtain motions at specific v and ω pairs, is evaluated. For the evaluations presented in this paper, several simulations were performed by sampling (v^k_{ref}, ω^k_{ref} values from the following intervals: v^k_{ref} ranging from 1 to 3 m/s by 0.25 m/s step, and ω^k_{ref} ranging from -100 to 100 degrees/s by 25 degrees/s step. A single v_{sw} of 4 m/s and a t_{sw} of 5 seconds were also selected for simulations involving changes in speeds. Figure 5 summarizes the mean trajectory error obtained across the (v^k_{ref}, ω^k_{ref}) pairs for each dataset and shows that the self-recorded dataset is the one generating a higher offset, followed by the Unity and Mixamo datasets, respectively. Figure 5 also shows that there are similarities among the lower ω trials across all datasets: the ω trials on -25 and 25 degrees/s are the ones with the highest offset, while the ones at 0 degrees/s score the lowest offsets across the board.

A further analysis of their offset errors also shows that each dataset has its own peculiarities, as seen in Figure 5. For instance, examples A and B have error margins ranging from 0 to 3.5 meters and 0 to 2 meters, respectively. However, Example 3 had error margins ranging from 0 to 14 meters. This is expected, as example C represents a dataset with poor coverage, especially when considering the tested (v, ω) pairs. One can also see that by the horizontal variations in the graph, ω has a greater influence over offset errors, meaning that matching v seems to be an easier task.

Concurrently to the accuracy of a given dataset, evaluating its given coverage is also of interest, i.e., how sparse or dense a given dataset is, for different (v,ω) combinations. Figure 1 gives an idea of how coverage in one of the datasets tested in this work looks like. Each motion is depicted as a different color and with a different type of marker. The motions with a low ω, thus around 0 degrees/s, are motions that mainly go straight. Instead, motions are constantly turning clockwise and counterclockwise when their ω is negative or positive, respectively. Having that, v will be just how fast they do it. Figure 1 also shows that the space around a v of 2m/s is sparse when looking away from the ω of 0 degrees/s. This means that any motion that falls into that area may have a hard time transitioning, incurring a high transitioning cost and potential v and ω errors, characterizing poor coverage in those areas.

Figure 6 shows how different v and ω ranges are covered in the three datasets evaluated in this section. The brighter parts represent the most dense part of every dataset, showing where the areas with the most data are available. In order to assess the quality of these maps, the symmetry of the points and their density has to be taken into account. From the figures, one can see that the Unity dataset is the most comprehensive one, followed by Mixamo dataset and the self-recorded dataset.

Apart from the more general metrics, some metrics deal with individual simulations, such as the transition cost, the v and ω errors over time, the use frequency from each motion frame and a graph structure showing the relationships between all motion clips

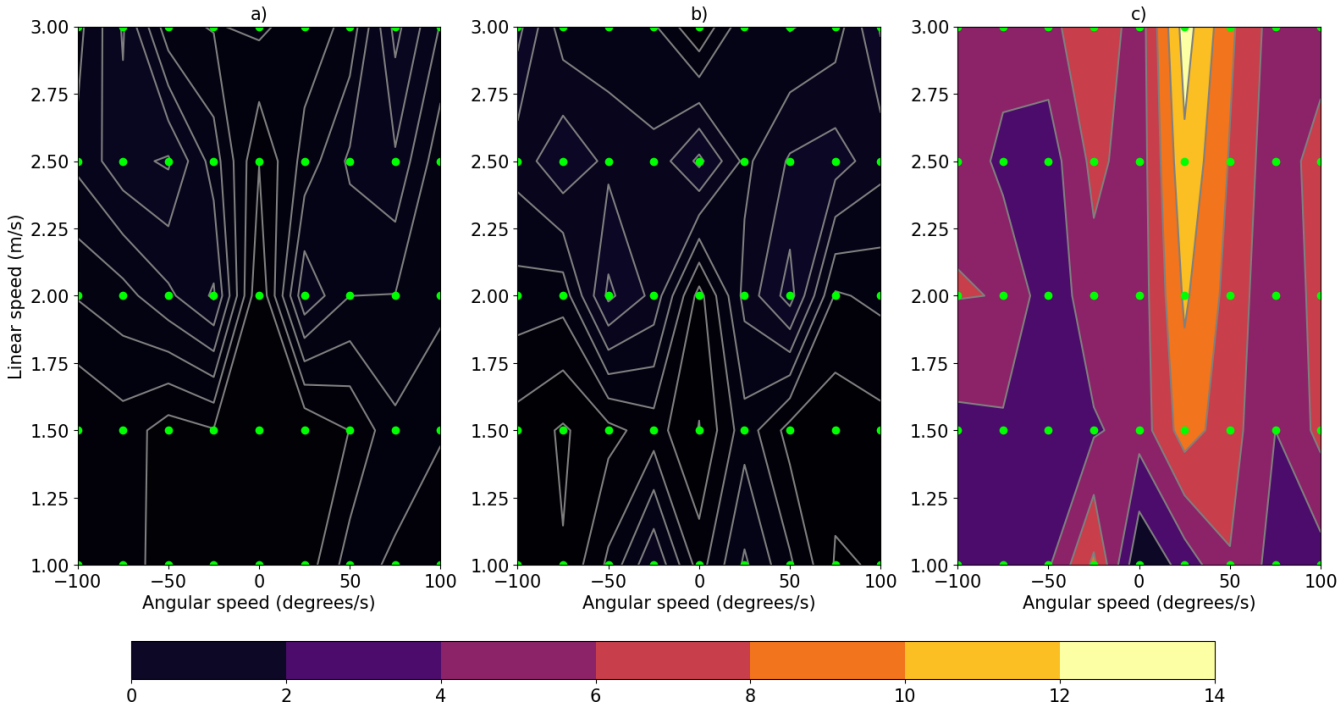

Figure 5: Offset error, in meters, on the Unity (a), Mixamo (b) and self-recorded (c) datasets, respectively. The dots throughout the graph show the different trials with various (v,ω) pairs, while the colored areas represent the error associated to each trial.

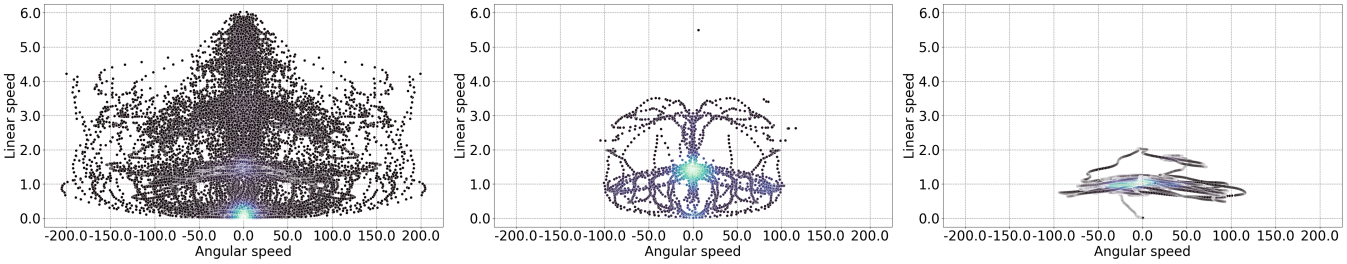

Figure 6: Linear (v) and angular (ω) speed coverage for three datasets used in our experiments: Unity (left), Mixamo (centre) and self-recorded (right) datasets.

and frames. These metrics can help tracing an error when using an motion dataset and ease the recapturing process by giving a more specific idea of what needs to be corrected.

As for the transition cost, one can see in Figure 7 the cost value changes according to the changes in v. In the example, the Mixamo dataset is used to switch between 1.5m/s (v) and 4m/s (v_{sw}) with a t_{sw} of 5 seconds and an ω of -50 degrees/s. Considering the motion density for the dataset in Figure 6, when looking at an ω of -50 degrees/s, there are no motion examples when v is 4m/s, while there are multiple ones when v is 1.5m/s. This means that during v_{sw} periods, the algorithm will have a harder time searching for a matching pose.

Considering an example using the Mixamo dataset with v set to 1m/s, v_{sw} to 4m/s and ω set to -100 degrees/s, the user can evaluate the v and ω error through the error distributions shown in Figure 8. As two speeds are taken into account, the simulation accounts for errors around v and v_{sw}, which results in a bimodal distribution. In this case, the v error will mostly roam around 0, although it also has a peak around 1.5m/s to account for when it switches to v_{sw}. The ω error is the same, roaming around -10 and 10 degrees/s.

As a way to verify the general usage of a motion dataset, the use frequency of each animation is displayed in Figure 9, showing how many times a frame from each animation was used. It also shows how many times a transition between different animations occurred through the "Transitions" bar in the graph. This helps in knowing which are the most relevant clips in the dataset.

As a complement to the use frequency metric, one can also see how the clips relate to each other by visualizing the origins and destinations of the transitions. Figure 10 shows a dataset with two

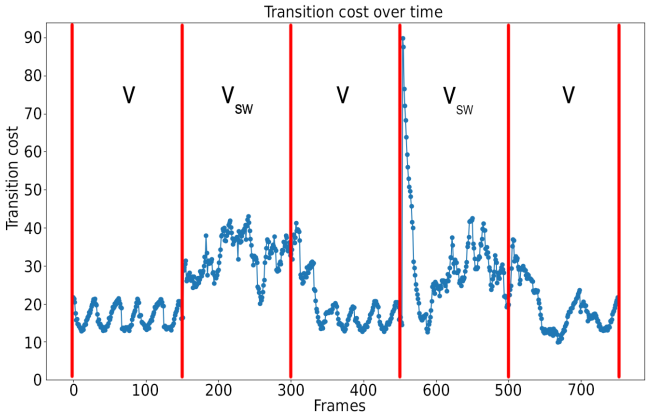

Figure 7: Variation in the cost of transition along a given simulation. In this example, the Mixamo dataset is used while v and v_{sw} are respectively set to 1.5 m/s and 4 m/s.

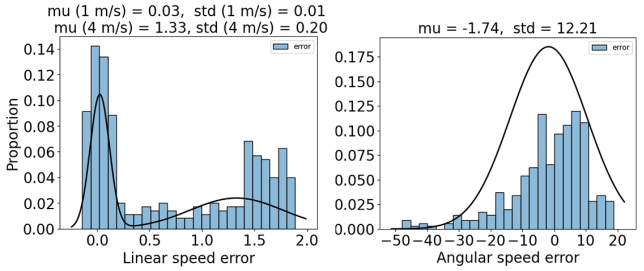

Figure 8: In the v and ω error distributions, the static and dynamic error can be visualized as the mean and standard deviation when fitted to a bimodal and a normal distributions, respectively.

animations transitioning between each other, which supports and gives more in-depth information to what is presented in Figure 9.

5 APPLICATION

As presented in Section 4.2, both the general and individual metrics quantify how a specific dataset fits into a purpose. This section will deal with some use cases in greater detail.

5.1 Completion

Having a dataset that can withstand the different situations planned for a simulation is crucial in lowering transition costs and increasing accuracy. Considering the amount of information gathered from the simulations, especially when looking at the relevance metrics, an evaluator can judge if adding data to a dataset is needed. The first way an evaluator can make this decision is by looking at the coverage metrics presented in Section 3.3.3. These metrics will enable the evaluator to gauge how sparse some areas are and judge if more motion is needed. In case the evaluator decides to add data, then the metrics in Section 3.3.1 can show the impact of these modifications.

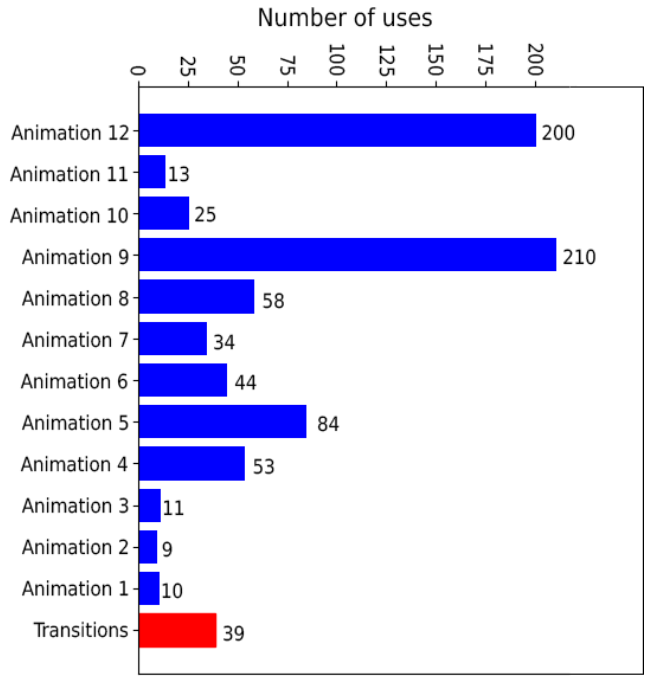

Figure 9: Distribution of clip usage and the number of transitions for a given simulation.

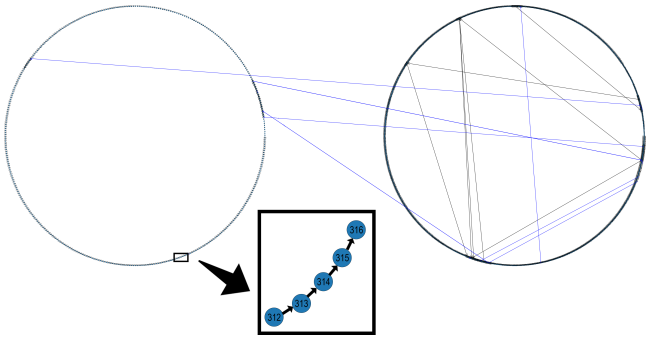

Figure 10: Each clip is represented as a circle, with its frames as individual dots around it. The lines represent transitions from a frame to the other.

At the time, these additions are done by either recording animation data empirically or gathering it from other datasets. The system aims, however, to have a virtual reality guiding system implemented, which orients the subject on how to behave in order to record specific motions.

5.2 Optimization

By having quality metrics to evaluate a dataset, this work implies a loop: getting data, running simulations, analyzing the post-processing metrics, removing or adding data and simulating again until a quality requirement is met. Through the algorithmic nature of motion matching, these are reproducible results, which means

that the evaluator can go back and forth between simulations in order to fine-tune the dataset.

6 DISCUSSION

Even though this work provides means to a more common grounded comparison of datasets, it is important to note that it is still a manual, subjective and iterative process. The main advantage, however, is that at least a decision can be made on interpretable metrics and reproducible simulations.

As noted in Section 4.2, datasets are divided in mainly three parts: motions that go straight and motions that turn clockwise or counterclockwise. Given that, if Figure 6a is divided in the middle of its x axis, a symmetry can be seen between the left and right sides. Therefore, since the Unity and Mixamo datasets are mostly symmetrical, the results will also follow their symmetry. Figure 5a, for example, shows errors that are similar, but not entirely symmetrical. However, the self-recorded dataset has a lower symmetry than both other datasets and Figure 5 shows the asymmetry in offset error caused by it. This can be explained by foot step desynchronization, which occurs when motion mirroring is not enforced: without mirroring, there is no guarantee that the same transitions will be available on both side of the ω spectrum.

If one looks at the offset error from the Unity and Mixamo datasets, it could be expected that the Unity dataset, by being the more complete one, would accrue less offset error, but this is not the case in the simulations. A dataset can seem complete when most of the ω and v pairs are covered and every area is dense. However, speed is not the only factor to take into account when blending from one frame to the other. In fact, the most important feature when trying to blend motions is the similarity between one pose and the other. Therefore, in order to fully grasp the blending potential of a dataset, one can employ pose similarity as a metric that can tell how similar the poses are in an area. When combining pose similarity with the speed coverage metric, one could have a clear understanding of what is missing to have a complete dataset.

6.1 Future work

As mentioned above, one limitation of our current framework is related to the fact that coverage currently only accounts for linear and angular velocities. A first addition would therefore be to include additional metrics, in particular related to pose similarity, that would broaden the coverage analysis of the dataset. Coverage estimation is a useful metric to showcase the capabilities and limitations of a dataset, although it mostly focuses on highlighting sparse areas. However, little can be gathered from dense areas, apart from the fact that there may be enough motion samples to decrease transition costs in those cases. The main problem is that there is no way, at this moment, to know for sure that a dense area indeed has enough motions to support a low-cost transition. This stems from the basis that although there are different samples, these motions could have a wildly different pose: a character could run backwards at the same speed that it runs forwards, for example. Hence, the system currently lacks a way to show how similar the poses are in each area which, when combined to the coverage map, could enable users to identifies areas of density and similarity. Such a feature would be valuable to add another dimension to the analysis and showing that even though there may be many motion examples in a dataset, those may be too diverse to enable low-cost transitions.

Completing a dataset can sometimes be a challenging task, as users need to sort through a multitude of motions in databases to find one that works or have a whole rig to record it themselves. This means that if a new motion needs to be added, significant efforts and time will be employed. For these reasons, providing relevant search-based or similarity-based interfaces for intuitively identifying motions to include from external databases would also be a relevant direction of research in the future. In the sense of optimizing a dataset, the system could also benefit from a more suggestive approach towards the modifications that can be done. As of now, the system shows the current popularity of every frame in the animations, but will not suggest which ones could be eliminated, for example.

Another future addition would be an online evaluation mode for motion datasets. This means that a performer would be provided with some information on which areas to record more motions for, instead of having to post-process all this information and have to go back to recording. An enhancement of this addition would be a 3D guide for motion capturing. Using a head mounted display to guide a performer when recording motion could be a possibility to explore, as it could enable the performer to record motions more precisely. The guide would be composed of a trajectory and (v,ω) measurements, which would give instant feedback to the user, enabling the correction of errors of the performance on the go.

6.2 Limitations

Although the system is able to successfully compare and provide grounds for an evaluation of motion datasets, it does so only for the displacive aspect of any motion. In this sense, the motion of a virtual character stopped in front of a table picking up a cup is essentially the same as an avatar poking another one, since even though the poses will be different, there is no spatial displacement. Thus, the upper-body portion of any motion is disregarded for coverage terms. Considering the many applications that upper-body motions can have, especially when environment or inter-character interactions are required, including such elements for automatically evaluating motion datasets will therefore be an important direction to explore in the future.

6.3 Conclusion

As noted in this paper, the analysis of motion datasets is subjective to the personal experience of the animator. Therefore, two animators may not rate a motion dataset the same as another, which not only introduces biases, but may also be subject to faults due to the sheer amount of data in these types of datasets. In order to provide a partial solution to this problem, we proposed in this paper a novel automatic and reproducible framework to evaluate motion datasets. Some of the metrics used in the paper are already known by the community, but they lack structured procedures to aid animators in analyzing their datasets.

As a step towards the complete analysis of motion datasets, this work focuses on locomotion, relying on mechanical metrics, such as linear and angular velocities, to calculate accuracy and coverage, as well as statistical metrics to calculate motion relevance. This

work also relies on these metrics, as well as motion matching, to provide a compromise between performance, optimization and result interpretability, finding an alternative to full machine learning approaches.

ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska Curie grant agreement No. 860768.

REFERENCES

Adobe. 2022. Mixamo. https://www.mixamo.com/#/

Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. 2008. Motion-Motif Graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '08)*. ACM Press, 117–126.

Michael Buttner. 2019. Machine Learning for Motion Synthesis and Character Control. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. ACM Press, Quebec, Canada.

Simon Clavet. 2016. Motion Matching and The Road to Next-Gen Animation. In *Game Developers Conference (GDC)*. San Francisco, California. https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road

Simon Clavet and Michael Buttner. 2015. Motion Matching and The Road to Next-Gen Animation. In *Nucl.ai*. https://www.youtube.com/watch?v=z_wpgHFSWss

Rogério E da Silva, Jan Ondrej, and Aljosa Smolic. 2019. Using LSTM for Automatic Classification of Human Motion Capture Data. In *VISIGRAPP (1: GRAPP)*. SCITEPRESS, Malta, 236–243.

Alireza Dirafzoon, Namita Lokare, and Edgar Lobaton. 2016. Action classification from motion capture data using topological data analysis. In *2016 IEEE global conference on signal and information processing (globalSIP)*. IEEE, Washington, DC, USA, 1260–1264.

Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. 2021. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences* 5, 6 (2021), 741–760.

Tim Furche, George Gottlob, Leonid Libkin, Giorgio Orsi, and Norman Paton. 2016. Data wrangling for big data: Challenges and opportunities. In *Advances in Database Technology—EDBT 2016: Proceedings of the 19th International Conference on Extending Database Technology*. OpenProceedings.org, Bordeaux, France, 473–478.

Tomasz Hachaj and Marek R Ogiela. 2020. RMoCap: an R language package for processing and kinematic analyzing motion capture data. *Multimedia Systems* 26, 2 (2020), 157–172.

Rachel Heck and Michael Gleicher. 2007. Parametric Motion Graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D '07)*. 129–136. https://doi.org/10.1145/1230100.1230123

Daniel Holden. 2018. Character Control with Neural Networks and Machine Learning. In *Game Developers Conference (GDC)*. San Francisco, California. https://www.gdcvault.com/play/1025389/Character-Control-with-Neural-Networks

Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned motion matching. *ACM Trans. Graph.* 39, 4 (2020), 53:1–53:12. https://doi.org/10.1145/3386569.3392440 All Papers/H/Holden et al. 2020 - Learned motion matching.pdf.

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Trans. Graph.* 36, 4 (2017), Article 42. https://doi.org/10.1145/3072959.3073663

Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Trans. Graph.* 35, 4 (2016), 1–11. https://doi.org/10.1145/2897824.2925975 All Papers/H/Holden et al. 2016 - A deep learning framework for character motion synthesis and editing.pdf.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion graphs. *ACM Trans. Graph.* 21, 3 (2002), 473–482. https://doi.org/10.1145/566654.566605 All Papers/K/Kovar et al. 2002 - Motion graphs.pdf.

Franziska Krebs, Andre Meixner, Isabel Patzer, and Tamim Asfour. 2021. The KIT Bimanual Manipulation Dataset. In *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, Munich, Germany, 499–506.

CMU Graphics Lab. 2000. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/

Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. New York, USA, 491–500.

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010. Motion fields for interactive character locomotion. *ACM Trans. Graph.* 29, 6 (2010), Article 138. https://doi.org/10.1145/1882261.1866160

Martin Manns, Michael Otto, and Markus Mauer. 2016. Measuring motion capture data quality for data driven human motion synthesis. *Procedia CIRP* 41 (2016), 945–950.

Jianyuan Min and Jinxiang Chai. 2012. Motion Graphs++: A Compact Generative Model for Semantic Motion Analysis and Synthesis. *ACM Trans. Graph.* 31, 6, Article 153 (2012). https://doi.org/10.1145/2366145.2366172

Paul S. A. Reitsma and Nancy S. Pollard. 2007. Evaluating motion graphs for character animation. *ACM Trans. Graph.* 26, 4 (2007), 18. https://doi.org/10.1145/1289603.1289609 All Papers/R/Reitsma and Pollard 2007 - Evaluating motion graphs for character animation.pdf.

Arun Thundyill Saseendran, Lovish Setia, Viren Chhabria, Debrup Chakraborty, and Aneek Barman Roy. 2019. Impact of data pruning on machine learning algorithm performance. *arXiv preprint arXiv:1901.10539* (2019).

Hyun Joon Shin and Hyun Seok Oh. 2006. Fat Graphs: Constructing an Interactive Character with Continuous Controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. 291–298.

Adam Switonski, Henryk Josinski, and Konrad Wojciechowski. 2019. Dynamic time warping in classification and selection of motion capture data. *Multidimensional Systems and Signal Processing* 30, 3 (2019), 1437–1468.

Ben JH van Basten and Arjan Egges. 2009. Evaluating distance metrics for animation blending. In *Proceedings of the 4th International Conference on Foundations of Digital Games*. 199–206.

Kristjan Zadziuk. 2016. Animation Bootcamp: Motion Matching: The Future of Games Animation...Today. In *Game Developers Conference (GDC)*. San Francisco, California. https://www.gdcvault.com/play/1023115/Animation-Bootcamp-Motion-Matching-The