Analyzing and Learning Movement Through Human-Computer Co-Creative Improvisation and Data Visualization

Table of Contents:

Abstract

Recent years have seen an incredible rise in the availability of household motion and video capture technologies, ranging from the humble webcam to the relatively sophisticated Kinect sensor. Naturally, this precipitated a rise in both the quantity and quality of motion capture data available on the internet. The wealth of data on the internet has caused a new interest in the field of motion data classification, the specific task of having a model classify and sort different clips of human motion. However, there is comparatively little work in the field of motion data clustering, which is an unsupervised field that may be more useful in the future as it allows for agents to recognize "categories" of motions without the need for user input or classified data. Systems that can cluster motion data focus more on "what type of motion data is this, and what is it similar to" rather than which motion is this. The LuminAI project, as described in this paper, is an example of a practical use for motion data clustering that allows the system to respond to user dance moves with a similar but different gesture. To analyze the efficacy and properties of this motion data clustering pipeline, we also propose a novel data visualization tool and the design considerations involved in its development.

1. Introduction

Humans collaboratively improvise movement in situations ranging from dance performances to pretend play to sports games. Computers with the ability to participate in these collaborative movement improvisations could have an impact on a variety of application domains, including improving naturalistic

procedural animation in game environments [3], fostering human creativity in gesture-based domains like dance or theater [8,9], and creating more engaging contexts for physical therapy and training [5].

One particular domain that has made advances in understanding embodied human-computer improvisation is the study of co-creative AI agents. A variety of recent research investigates how humans and computers may be able to create together in gesture-based domains including co-creative dance [9] and collaborative movement improvisation [8]. However, an obstacle that is pervasive throughout these projects is that humans and their AI collaborators bring significantly different sets of experiences to the co-creative interaction. Humans possess a vast amount of real-world knowledge, in contrast to AI agents, which draw their knowledge from comparatively small datasets. This contrast creates an imbalance during a co-creative interaction, since the humans are required to give more than they receive.

Unfortunately, many embodied creative domains like dance, pretend play, and theater are notable for their lack of large-scale, diverse, annotated datasets since motion-capture data can be time consuming and expensive to collect. Agents capable of *lifelong learning* [12] are particularly well-suited for embodied creative domains since they can learn interactively from human collaborators without supervision. However, the agent needs some way of reasoning about newly learned gestures in order to respond intelligently to its human partner. One intuitive way to reason about gestures is based on their perceptual or semantic similarity, an approach that is frequently used in improvisation in a variety of domains, such as theater and jazz [11]. Discerning gesture similarity in movement improvisation requires the ability to both cluster gestures based on different metrics on-the-fly and identification of which cluster a gesture belongs to in real-time.

Most existing research on gesture understanding focuses on gesture Classification (i.e. identifying and categorizing different clips of human motion [6]). However, this is not particularly useful for lifelong learning in creative domains, since human collaborators can perform a seemingly infinite number of novel gestures while classification systems try to label these gestures according to only a finite number of known categories. As a result, new gestures will not be incorporated into the agent's knowledge base, making it difficult for the agent to learn-through-interaction and thereby limiting its long-term ability to contribute to creative collaborations. In contrast, a system capable of unsupervised gesture *clustering* would be able to learn novel gesture types. Such a system could compare novel gestures to previously seen gestures and add new gestures to existing clusters, learning through interaction.

An unsupervised gesture clustering system could also dynamically respond to novel gestures by drawing on past experiences and finding a similar gesture it has seen before without needing a pre-programmed label (in effect, creating its own knowledge of gesture categories rather than relying on pre-programmed knowledge). Systems capable of gesture clustering also have the potential to be domain-independent, whereas existing classification algorithms can often only classify gestures based on a very domain-specific set of categories.

There is some existing literature on unsupervised gesture clustering, though it is focused primarily on only hand gestures [2, 14, 15]. Work is still needed to understand how to approach unsupervised gesture clustering with a full-body skeleton, which differs significantly from hand motion both anatomically and in terms of gesture duration (i.e. hand gestures tend to be shorter in length with less freedom of movement than full-body gestures). In this paper, we investigate the following central research question: *How can we implement a co-creative agent that can cluster arbitrary full-body motion data in real-time, thereby enabling the agent to draw on a breadth of learned experiences when responding to its human collaborator*?

A secondary research question we address in order to better answer the first is: how can we design a data visualization that best expresses the characteristic semantic properties of our pre-processing pipeline. The research process and design philosophy behind this visualization tool seeks to ask human-perception oriented questions such as “what makes us consider two gestures similar”, “how is similarity reflected in the lower-dimensional space”, “what do the dimensions correspond to anatomically” and “what does a low geometric distance between two points in this space really mean”. These questions are all exploratory and open-ended in nature; they are not suitable for a rigorous quantitative analysis due to the inherently humanist emphasis of trying to understand motion in an intuitive way.

Consequently, we believe that the ideal way to pursue these questions and gain insight into this pre-processing technique is by creating a program that empowers and encourages the users to explore and delve deeply into the pre-processing pipeline’s unique behaviors and the properties of the resultant low-dimensional space. This is a task that is well suited for an interactive data visualization. The two foundational principles this visualization tool is built on are as follows. The first, that it must be robust and powerful, being able to provide knowledge at a variety of abstraction levels from the specific movement of an arm to the shape of a cluster of 100 gestures. The second, that it must be facilitate exploratory and investigative user behavior in an intuitive and rewarding manner.

In the rest of the paper, we look at other related work in the areas of pipeline design and visualization, discuss a particular use case for which we designed our clustering pipeline, detail the implementation of the pipeline, discuss a preliminary evaluation of our pipeline using both traditional methods and our novel visualization tool, and end with conclusions and plans for future work.

2. Pre-Processing Pipeline Literature Review

The focus of our work is the development of a pre-processing pipeline, or novel representation, of human motion capture data that we will attempt to cluster, a form of unsupervised learning. Motion capture data represents the human body in a way completely unlike traditional 2d video. In a video, a still frame is composed of a 2d array with a color or intensity value for each element. In motion capture data, a still frame is composed of an array, with each index representing some part of the human body, and each element being some multi-dimensional geometric information about the related body part. Motion capture data is also non-trivial to capture, requiring the usage of motion capture sensors such as the Kinect or a motion capture suit. Furthermore, this motion data cannot be easily obtained from conventional video files due to the difficulties of identifying the precise positions of joints in 3d space given only image data.

As a result of this barrier to entry, most work on human motion has been done in video data of human action instead. The reasoning behind our focus on motion capture data is that our pre-processing pipeline, along with the clustering algorithm, will eventually be incorporated into a co-creative art installation referred to as the LuminAI project, in which users dance in front of a Kinect sensor, which prompts the on-screen agent to select and respond with a dance considered similar by the clustering produced. This necessitates a computationally efficient implementation of pre-processing for the whole human skeleton that also results in clustered items being visually similar.

One example of work done on video data is *Unsupervised Learning of Human Expressions, Gestures, and Actions* by O’Hara [10]. This paper presents two different approaches for processing video data into a more compact representation, which are referred to as Bag of Features [10] and Product Manifold [10]. The way that the authors evaluate the efficacy of these approaches is by processing a video data library, then clustering the output data. Once the clustering is obtained, the authors evaluate the pre-processing approaches based on the quality of this clustering [10]. The intention behind this work is very similar to ours in that both works emphasize the evaluation of a pre-processing pipeline for human gesture data.

However, their work does not naturally extend to ours as the Bag of Features and Product Manifold algorithms were designed to work with video data, rather than motion capture data, and do not translate across data representations in an intuitive way. In addition, this paper's main contribution is the evaluation of pre-existing algorithms, whereas ours is both the proposal and evaluation of a novel approach to pre-processing. As a result, their work does not directly address our research question.

Though there does exist work done for motion capture data, many of these works do not emphasize unsupervised learning. For instance, *Classification of K-Pop Dance Movements Based on Skeleton Information Obtained by a Kinect Sensor* by Kim, Kim and Kwak [6] focuses on the classification aspect of motion capture data rather than unsupervised learning. Generally speaking, any given representation can be used for classification or clustering depending on user intention. However, the representation proposed in Kim's paper uses supervised learning techniques in their pre-processing of the motion capture data. Their proposed pipeline utilizes Linear Discriminant Analysis, a supervised learning dimensionality reduction technique, as part of their FISHERDANCE algorithm [6]. Consequently, we are unable to borrow from this technique in their pipeline, as our work is with unsupervised learning and necessitates unlabeled data. Our main takeaway from this work was our adoption of their angle extraction step [6] in our pre-processing framework, which is a salient component of their framework that does not require implementation in a supervised learning context. We find that this work is unable to directly answer our research question due to its pipeline being restricted to the realm of supervised learning.

An interesting characteristic of this research field is the unexpected amount of work done on the analysis of human hands rather than the human body. *In Feasibility of Principal Component Analysis in hand gesture recognition system* [14], Srivastava evaluates the effectiveness of Principal Component Analysis in pre-processing motion capture data of human hands [14]. Though this paper focuses in classification as well, PCA is not a supervised learning technique and thus can be incorporated into our work. The issue encountered with this work is that despite the author's claim that he uses motion capture data, the paper itself suggests that video data was used instead [14]. In addition, the scope of their motion data library is comparatively narrow. Whereas our work emphasizes the evaluating the similarity of a dance gesture to up to hundreds of completely different gestures, their motion data library only contains 10 different categories of hand motion [14]. Furthermore, the duration of a hand gesture is much shorter than a full-body dance move. As a result, this work does not address our research question due to its focus on the human hand, rather than the whole body.

The work that is perhaps most like ours in purpose is *Clustering poses of motion capture data using limb centroids*. In this paper, Balci proposes a novel representation of a pose, a pose being the configuration of the body parts in a human skeleton at one atomic slice in time, called a limb centroid [1]. A limb centroid is effectively the center of mass of several given body parts [1]. The author specifies 6 different limb centroids each skeleton, resulting in from over 15 different body parts to track down to 6. This is a remarkable reduction in dimensionality, while seemingly minimizing the amount of discretization error introduced [1]. However, we are unable to reduce the dimensionality of the body any further from these 6 components. This issue becomes apparent in the context of their input data, which clusters individual poses within a gesture rather than the entire gesture itself [1]. The 6 features per frame is therefore still scaled linearly by the number of frames in the gesture, which causes the data to still be high dimensional when considering that the number of frames in a gesture can be up to 900. We find that this work, while being highly relevant to ours, does not immediately address the performance concerns we encounter with our emphasis on processing entire gestures. Limb centroids will likely be investigated further in future work.

Indeed, a survey of this field reveals that our work occupies an interesting intersection between the comparatively niche field of clustering, with the other comparatively niche field of whole-body motion capture data. As a result, there does not seem to be any work we encountered that directly answers our research question of how to best take advantage of motion capture data's salient qualities in an interactive installation, showing a clear research gap in existing literature. What work we can say is related to ours is usually only tangentially or indirectly so, which we drew our initial pipeline design inspiration from. As a result, we believe that our work fulfills an important niche presented by the lack of interdisciplinary work between performative and improvisational art, and machine learning.

2.1 Previous Work on Info. Visualization

The currently data visualization tool was developed as a response to several limitations we experienced with a prototype design that was hastily constructed for the express purposes of debugging and making sure the pipeline had no glaring issues. Though this design was, for the most part, functional, it was lacking in several quality of life features, like simultaneous viewing of different gestures on the same screen. This prototype tool was constructed using D3, 3JS and HTML.

Before we tried programming a unique visualization tool for this project, we briefly considered using commercially available tools instead. Tableau is one such example of a widely used visualization tool that requires minimal to no programming on behalf of the end user and functions mostly like excel. We found Tableau unsatisfactory for several reasons. The first and foremost issue is that Tableau's scripting features were too limited for us to achieve several of our implementation goals. The vision that we had in mind was a main overview screen displaying several gesture points plotted in some 2d or 3d space and a separate detail screen that would show the animation of whatever gesture point the user selected. Tableau did not support these features and has no support for 3d animation, in addition to not supporting the parsing of the data files which our pipeline generated.

The next approach we tried was D3, which is an open source Javascript library explicitly designed for creating web-based interactive data visualizations. Our decision to try D3 was motivated by several factors, the most convincing being that it was lightweight and could be run in a browser. Tools developed using D3 could be hosted on server and be accessed by anyone with an internet connection without the need for any downloading. Because the tool was designed as a companion program to the main pipeline, we believed it would be beneficial to have it be performance friendly as well. D3 has a low resource requirement and can easily be run on phones or other mobile computing devices. However, these benefits came at a cost that we ultimately found to be too steep.

The performance friendly and networkable features of D3 meant that it would not support any kind of 3D rendering technology such as WebGL. Being constrained to two dimensions, we had to configure the pipeline to produce only two resulting dimensions for each gesture, causing a substantial amount of information loss. We were, however, able to construct the main overview screen that rendered several points corresponding to gestures, with their positions being determined by the value of the gesture being processed into our novel representation. Clicking on these gestures would launch another window that contained a 3D animation corresponding to that gesture.

The rendering of these animations was done using the 3JS library. 3JS was designed to produce performance friendly web-enabled 3d apps such as games and simple 3d visualizations. Its advantages are that it integrated well with D3 due to them both being library extensions for Javascript and is fairly simple to use. The difficulty we experienced with D3 and 3JS was that their scripting features were not sophisticated enough for us to dynamically load data from D3 to 3JS due to issues with HTML security. Our final 3JS solution involved programmatically generating a new 3js animation for each gesture and

calling it from D3 due to our inability to resolve the aforementioned issue. The consequence of this design decision was that the tool had to be re-built every time we wanted to load a new animation set, and that the transition between the overview screen and the detail animation viewer was not seamless. Furthermore, the 3js' userbase was comparatively small, causing several features and core technologies to be too difficult to implement due to lack of documentation and poor learner's resources. The development of new features for this hybrid system was prohibitively difficult, motivating us to investigate alternatives.

We eventually decided that performance-friendliness and web-compatibility were not as important to us as feature-richness and moved to using the more powerful and resource intensive Unity3d platform, which our current implementation is built upon. Unity3d is considered the most popular game engine for new developers due to its scripting being abstracted away from low level considerations like collision detection. As a result, it has a variety of different learners' resources available for free on tutorial websites and in-depth documentation on the main website. Because Unity3d was developed to be a game engine, these scripting features are very sophisticated and allow us to implement several features that we had previously thought were unachievable. Unlike 3JS, Unity has a "What You See is What you Get" editor that allows for real-time rendering of the game, which greatly expedited development speed. In addition, Unity also provides several standard assets which are freely available that would have otherwise cost development time such as camera control and user interface systems, all of which had to be manually coded in 3JS. These advantages, in our evaluation, made it the ideal platform for developing a more sophisticated and VR enabled version of the visualization tool.

3. Pipeline Design

Fig. 1: Real-Time Operation

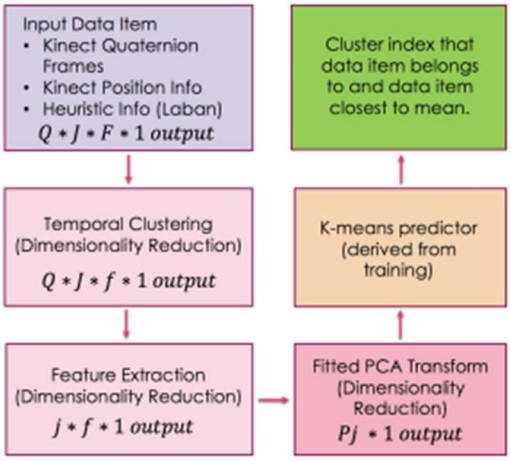
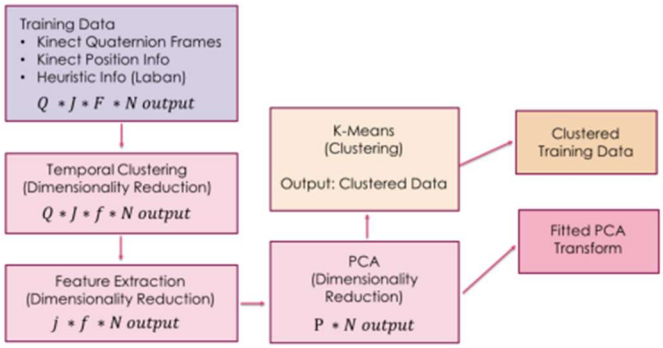


Fig. 2: Training



Our proposed pipeline consists of two similar implementations, one for real-time (Fig. 1) operation with users and one for training (Fig. 2) the model and updating it with newly witnessed gestures. The training component consists of three main parts: the pre-processing dimensionality reduction steps, the clustering and model fitting steps, and the export of a trained k-means model [16] with a fitted PCA transform model [13]. The pre-processing step reduces the dimensionality of motion data considerably using temporal clustering [17], then reduces dimensionality even further using a joint angle extraction technique [6]. Once this is applied to every item in the motion library, the data is then used in the model-fitting and clustering step, in which a PCA model is first fitted on the reduced motion library.

Once the PCA transform model is obtained, the dimensionality of the data is further reduced using the PCA model. Finally, the newly transformed data is clustered using a k-means model. The products of the model-fitting and clustering step are a PCA transform model, which can be used to apply PCA to novel data items, and a k-means model, which contains the clustering of the pre-processed motion library and can be used to place novel gestures in their appropriate clusters. In the final stage of the pipeline, these two models are exported for future use.

The pipeline running in real-time uses the pre-trained PCA transform model and k-means model and consists of three steps: motion recording, motion pre-processing, and motion clustering. In the first step, a participant is prompted to record themselves performing a gesture using the Microsoft Kinect 2.0 depth sensor. In the motion pre-processing step, the dimensionality of the novel gesture is reduced using temporal clustering, angle extraction, and the fitted PCA transform model. In the final step, the transformed gesture is placed into an appropriate cluster by the fitted k-means model. A gesture randomly selected from the target cluster of the novel gesture will be played back to the user (this step is specific to the *LuminAI* use case, in which we want the agent to respond with a gesture that is similar to the participant's gesture).

We decided to focus heavily on the principle of maximizing variance between different gestures when designing our pipeline. Temporal clustering, PCA, and k-means were chosen as a starting point from pre-existing papers [1, 6, 14, 17] specifically due to the way that all three incorporate elements of variance maximization in their design. In the remainder of this section, we will describe the implementation of each stage of the pipeline in more detail.

3.1 Input Data

The input data for the pipeline can consist of any feature vector where the geometric distance between any two feature vectors is a quantitative measure of the dissimilarity between them. This means that the pipeline can cluster gestures based on a feature vector consisting of joints-based skeletal data or a feature

vector of other movement qualities such as *Time*, *Weight*, *Space*, or *Flow* [7]. We focus on joints-based skeletal data in this paper, but plan to incorporate Laban feature vectors in future work as an alternative way of understanding meaningful similarity between gestures. The joints-based input data for the pipeline consists of gestures gathered using the Microsoft Kinect depth sensor--although this pipeline could be adapted to accommodate other motion capture devices such as a motion capture suit.

3.2 Temporal Clustering

The number of frames can easily grow into the hundreds with longer gestures, making a reduction in the number of frames necessary in order to facilitate real-time data processing. The objective of *temporal clustering* [17] is to find a user-specified number of ``keyframes" that best approximate the input motion. Temporal clustering achieves this by expressing the problem of finding representative ``keyframes" as optimizing the placements of consecutive contiguous partitions. Each partition is evaluated using a measure described in Yang et al. as the ``within-segment sum of squared error" which quantifies how ``different" the frames in each partition are from the partition's mean frame [17]. This creates partitions consisting of frames that are as similar to one another as possible, thus indirectly maximizing the difference or variance between one partition and all other partitions. In order to make this approach computationally tractable, Fisher's optimal partition algorithm [6], a dynamic programming approach, is used to identify these partitions, and an average of all the frames in one partition is used to produce a ``keyframe". In Yang et al.'s original paper on temporal clustering, there appears to be an error in the pseudocode in which the diameter calculation is calculated over all n rather than all j, j being the iterator for a for loop. Our implementation uses our modified pseudocode instead of the original.

3.3 Feature Extraction

Certain joints do not contribute as much to the overall representation of a gesture or dance as much as others do--for example, shaking your leg will have a larger effect on a gesture than shaking your foot. The significance of certain joints and their associated angles in different kinds of dance was noticed by Kim et al [6]. Kim et al. achieved remarkable accuracy in their classification model by extracting the scalar angles created from the positions of important joints and the positions of their neighboring joints, then discarding joints that were deemed insignificant [6].

Our implementation borrows from Kim et al.'s technique and extracts angles in the same way, but because our representation uses a reduced set of frames and therefore has lower dimensionality, we are able to keep more joint angles without reducing performance. The joints that are deemed significant are selected by the programmer before the system begins training (this also allows the joints under consideration to be modified according to a particular dance style or culture). In our current implementation, the joints that we have kept are the middle spine, left shoulder, left elbow, right shoulder, right elbow, left hip, left knee, right hip, right knee. Once the joints have been decided, our pipeline then extracts the angles of important joints, further reducing dimensionality. We use Kim et al.'s technique of computing the angle of rotation between the parent joint and the child joint of any ``important joint", thus producing the vectors from the ``important joint" to ``parent joint" and ``important joint" to ``child joint" [6]. This process is much like placing any point A in 3D space, placing two other points B and C in arbitrary locations, and measuring the angle BAC created, oriented in the plane created by the vectors AB and AC.

3.4 PCA

PCA is one of the most widely known approaches to dimensionality reduction available. It is considered a ``standard technique for finding the single best (in the sense of least-square error) subspace of a given dimension " [13]. The mathematical principle behind PCA is the creation of a set of principal components

that best express or explain the linear variance present in the data. A principal component is found by creating linear combinations of existing axes, with the first principal component exhibiting the most variance among data points, and the second principal component less so, and so on.

In addition to the motivation provided by the experimental success of PCA used in gesture-related domains, we were also inspired to use PCA in our pipeline because its mathematical principle is similar to that of temporal clustering, which also focuses on maximizing variance. The number of principal components is programmer-specified. Suppose that it is set to P, then the dimensionality of a single gesture from an arbitrary dimensionality to simply P. In addition to producing a transformed lower-dimension data set, the PCA model will also be fitted to the data set and will be able to transform novel data points into the same subspace as the data that was used to train it. This step exports the transformed data and the fitted model for future use in the clustering pipeline.

3.5 K-Means

K-means belongs to the family of partition based clustering algorithms, whose key principle is the definition and characterization of a cluster by its ``center point'', where the center point of a cluster is the ``average'' or the point that minimizes distance between it and all other points in the cluster [16]. K-means updates the centers of clusters iteratively until the clusters eventually converge and each data point is placed into its appropriate cluster [16]. K-means' biggest advantage is that it is relatively computationally efficient, but it suffers from several other issues such as requiring a pre-set number of clusters and being sensitive to outliers [16].

K-means' usage is nonetheless widespread, and the algorithm has been shown to work well in gesture-based domains [1, 10]. Balci et al.'s use of k-means alongside PCA also indicates that the two work well together [1]. Due to k-means' heavy reliance on a distance metric when comparing data-points, we find it intuitive to use for a pipeline that maximizes variance. Given N data items of dimensionality P, the dimensionality of the input is P x N. After k-means is fitted to this data set, it produces a clustering that assigns an index to each data point corresponding to the cluster it belongs to, and a trained k-means model that is able to predict what cluster novel data items belong to, provided that the novel data item goes through the appropriate pre-processing.

3.6 Pipeline Evaluation Overview

We conducted a preliminary evaluation of the three-stage pipeline for unsupervised gesture clustering of arbitrary full-body motion data that we developed in order to better understand its ability to cluster skeletally similar gestures and identify limitations. We initially set the number of principal components for the PCA model to two and the number of k-means clusters to three for our evaluation. We initially chose two as the number of dimensions because it allows for easy visualization and inspection of the data, but with a downside of a decrease in accuracy. We later changed the number of k-means clusters from three to four after observing four clear clusters in the visualization. This section details the findings from our evaluation which, while preliminary, offer insights that can inform future research.

3.7 Dataset

We gathered a dataset of 104 unique gestures in order to develop an initial understanding of how well our pipeline clustered gestures based on skeletal similarity. Four different members of our lab danced in front of a Microsoft Kinect sensor placed at waist level in order to record the gestures. The participants were prompted to cover a wide variety of motions that each differed greatly from one another. Participants alternated between isolated motions that engaged only one body part and whole body dances or motions

that engaged all four limbs. The participants were told not to perform certain gesture types due to the difficulties the Kinect sensor has with tracking them, such as motions that involve rotating the body along the upwards Y axis at rapid speeds (e.g. spinning) or gestures in which body parts were occluded (e.g. laying on the ground).

We attempted to label each gesture in the dataset according to a particular body part category (``hands", ``hips", or ``legs") in order to determine whether the clusters the pipeline created would match up with the labels we gave them as a way to measure the ``intuitiveness" of the clusters generated by our pipeline. Labels were assigned based on which body parts were primarily being used in the gesture--for instance, a one handed wave and two handed wave would both be put under the label of ``hands", whereas a gesture depicting a walking motion would be put under the label of ``legs". Unfortunately, it proved difficult to intuitively label some of the more complex movements involving multiple body parts, so we ended up only labeling 44 of the 104 gestures (we refer to this as the reduced dataset in the remainder of the paper, see for future plans to improve this evaluation metric).

3.8 Cluster Clarity

We visually evaluated our pipeline's ability to cluster items using the reduced dataset. Our hypothesis was that the clustering visualization would produce clearly identifiable clusters of the gestures that correspond to the ``hands", ``hips", and ``legs" labels applied to the reduced dataset. As the figure below shows, the red clustering on the left is the most obvious due to its density. Three more distinct clusters can be observed towards the top, bottom and right-hand side of the visualization. The clusters in the full dataset are less visually apparent, but this is to be expected as participants were instructed to perform varied gestures, meaning that not many gestures in the full dataset were similar to one another. As a point of comparison, all of the gestures except for one outlier appear to be clustered in a small and very dense clump in the center of the visualization generated from the results from running only PCA and k-means on the data without our pipeline pre-processing (not pictured due to space constraints), indicating that our pipeline did a better job of separating the gestures into distinct clusters.

Fig.3: Output Visualization of Clustered Gestures

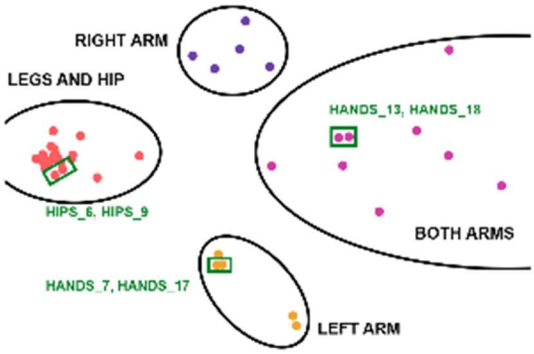


Fig.4: Gesture Homogeneity Table

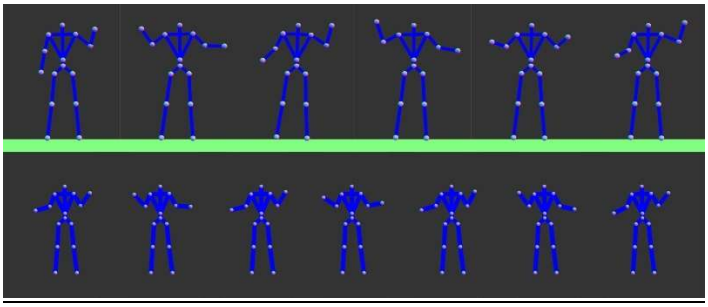
Cluster	# gestures	Legs %	Hips %	Hands %
Both Hands	9	0	11.11	88.88
Right Hand	5	0	0	100
Left Hand	5	0	0	100
Lower Body	25	68	32	0

We initially hypothesized that we would see three distinct clusters of gestures in the reduced labeled dataset after pre-processing - one for ``hips", one for ``legs", and one for ``hands". We actually found four distinct clusters. We evaluated the quality of these clusters based on how homogeneous each cluster was in terms of the labeled gestures it contained (i.e. a cluster consisting exclusively of ``hands" gestures was considered a better clustering than a cluster consisting of an equal mix of ``hands", ``hips" and ``legs"). We found that the clusters on the top (blue) and bottom (yellow) of the visualization consisted exclusively of ``hands" gestures (Fig. 3). The two clusters correspond to left and right arm motion, suggesting that our pipeline discretized the two body parts into their own individual clusters. The cluster on the far right (pink) also consists of 88% gestures labeled as ``hands". This cluster appears to be formed from ``hands" gestures that involve both left and right arm motion, explaining its positioning between the left and right arm clusters. The final cluster is shown on the far left (red) in the figure above and is the most mixed of all the clusters present, composed of all of the ``legs" and ``hips" motions together. The likely reason behind this is that at the time of recording, we did not realize that moving ones' hips almost certainly involves the reorientation of the legs. In addition, in all the ``hips" and ``legs" motions, the performers' arms were static by their sides, causing the upper bodies in these gestures to be identical to one another, likely explaining the density of this cluster. In spite of the unexpected results, we found that according to our evaluation metric, the clusters created for the reduced dataset were agreeable.

3.9 Visual Inspection & Exemplars

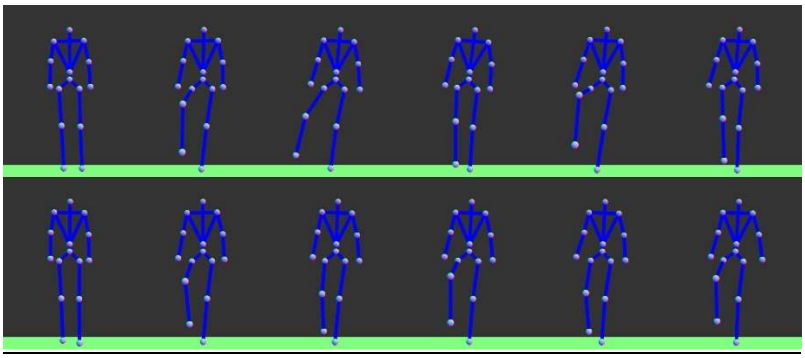
We conducted a qualitative visual inspection of both the reduced and the full dataset to supplement our quantitative evaluation of the reduced dataset clusters. We present several exemplar gesture comparisons from the reduced dataset here to highlight areas where the pipeline succeeded and failed. In Figures, gestures are depicted as a series of keyframes and should be read left to right. Gestures are presented in pairs for comparison, with one gesture on the top row and one gesture on the bottom row. Fig. shows the location of the exemplars we selected within the clustering plot for the reduced dataset (the exemplars are boxed and labeled in green).

Fig. 5: Gesture Pair: HANDS 13 HANDS 18



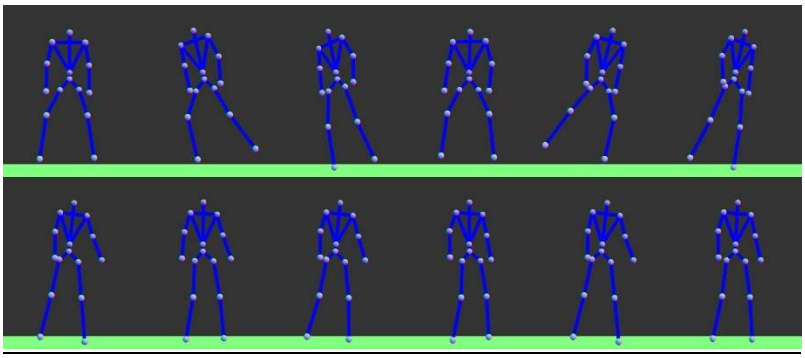
The two gestures pictured in Fig.5 were placed in the ``both hands" cluster. In the first gesture, the skeleton moves both of its hands in a circular fashion, engaging its elbows in the motion. This is visually similar to the second gesture, in which the skeleton performs a simple wave with both hands. The keyframes shown also suggest that the system has some understanding of ``rhythm" as the reduced keyframe set clearly depicts the ``left arm then right arm" rhythm of the gestures.

Fig. 6: Gesture Pair: HANDS 16 HANDS 8



The gestures pictured in Fig.6 both depict the raising and lowering of the knee--however, the gesture shown at the top adds more lateral motion to the knee joint. In spite of the difference between the two, we found their close clustering agreeable due to their intuitive visual similarity.

Fig. 7: Failure Case Example: HIPS 5 HIPS 9



The two sets of gesture keyframes pictured in Fig. 7 are an example of what we consider a failure case. The emphasis in the gesture shown at the top is clearly the lateral swaying and leaning of the upper body whereas the gesture on the bottom depicts only the lateral swinging of hips. This difference is lost because our angle extraction works only on specific body parts and does not take into account the rotation of the whole body. As a result, these two gestures are considered similar due to their close joint orientations. This effect was observed in several other gesture pairs.

3.10 Emergent Properties

We also noticed that the clustering visualization of the reduced dataset took on an unexpected emergent property--the placement of data points in this space allows one to immediately determine which body part was most active simply by looking at which quartile it lies in. This is an intriguing property because it suggests that the system, with no input from the user, has identified the body parts of the human skeleton that exhibit the most motion variance. It has learned on its own that limbs are an important part of motion and clustered data points using them.

3.11 Limitations

We extract angles from important joints using their Cartesian coordinates as part of the pre-processing step. This step introduces an invariance to the actual position of the user relative to the Kinect camera, as applying a transformation to all joints will have no effect on the angle extracted. A person performing a wave to the left of the camera will have the same joint angles as a person on the right. Angle extraction also makes the system blind to the direction a rotating joint is currently facing. These consequences of

angle extraction could interfere with dance styles or gestures that emphasize translational movement or the direction of angular movement.

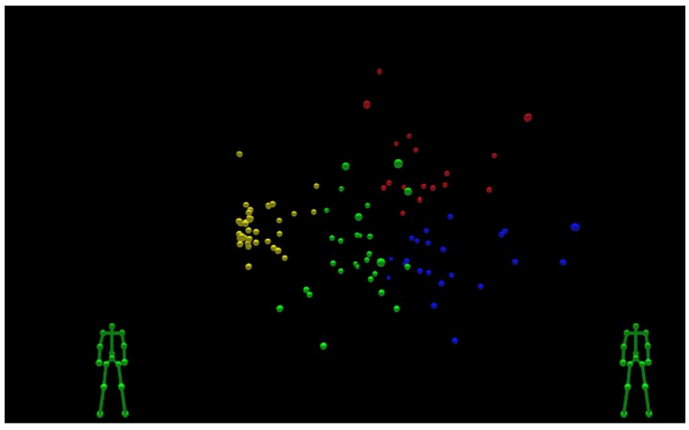
Each of the body parts is given a unique position in the feature vector used to describe a Kinect skeleton. This means that the system has difficulty equating similar motions mirrored across the Y axis of the human body. For human users, it is apparent that a hand waving motion is a ``wave'' regardless of which arm it is performed with. However, our system does not view these two to be similar as it has no preconception of the symmetrical human body, nor the relationship between the left and right arms.

Our pipeline gives equal weight to body parts that remain static and body parts that are moving from frame-to-frame, but we noticed during our evaluation that we intuitively placed a greater weight on moving body parts when comparing similar gestures. We hypothesize this to be the cause of some of the failure cases observed in the reduced dataset ``legs and hips'' cluster. Due to the similar positions of the upper bodies in the gestures from that cluster, gestures that are sometimes visually dissimilar to humans due to movement of an angle, like the hip, are placed together due to their upper body

4. Info. Visualization Interface and Operation Summary

This section will begin with giving a summary of the characteristics, behaviors and user interface of the visualization tool followed by an analysis of how well the visualization conforms to pre-existing data visualization design principles. The main screen that the user will spend most of their time interacting with is what we refer to as the “overview screen” (Fig. 8). It takes the shape of a black void which is populated with several multi-colored spheres. Each of these spheres represents a gesture, with its position determined by the resultant value of the gesture going through dimensionality reduction with an output dimensionality of three. Each sphere will also have one of several different colors. The color of a sphere, which will henceforth be referred to as a gesture point, encodes the cluster assignment of this gesture point. Consequently, it is expected that within this black space there will be several clusters of similarly colored spheres in a roughly spheroid shape, as per the nature of K-means [16].

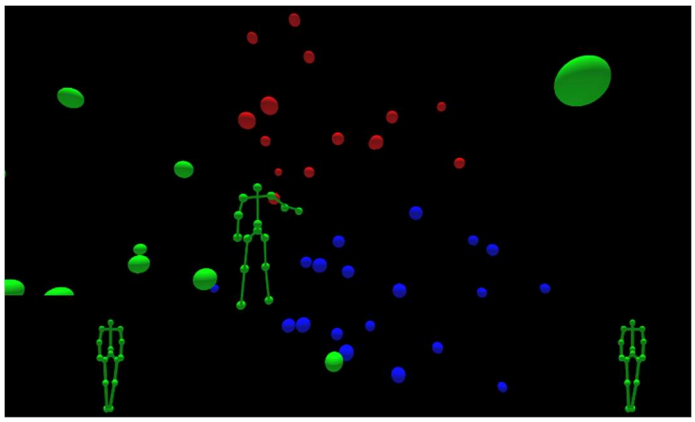
Fig. 8: Screenshot from the Tool Demonstrating the User Interface



The camera position in this tool is not fixed and controlled by the user using 6 degrees of motion input using keyboard keys and the mouse. By using the two together, the user can easily traverse this 3d space and decide whether they want to focus on one smaller region of the clustering or zoom outwards and get a broad overview. While navigating the pre-processing subspace, the user will come into situations in which they come close to gesture points. If the distance between the user and the gesture point is lower than some certain threshold, then the gesture point is disabled and a small animated avatar is displayed in

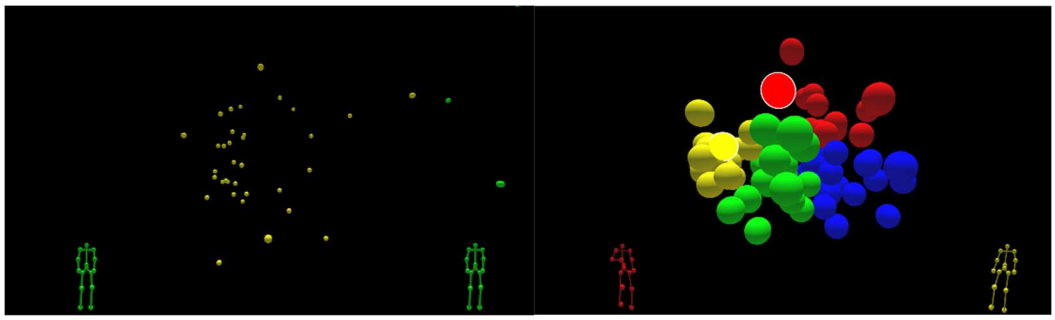
its place (Fig.9). Displaying this avatar instead also prevents the gesture point from obscuring points behind it, as the avatar has a much lower surface area. This animated avatar is of the same color as the gesture point and will play the corresponding motion capture animation on loop until the user leaves the threshold distance. This threshold behavior can be considered as having a spherical collider centered on the camera's position, where gesture points that are within the collider are activated to play the animation. The size of this spherical collider, or the threshold distance, can be controlled by the user.

Fig. 9: Screenshot from the Tool Showcasing Avatar Animation



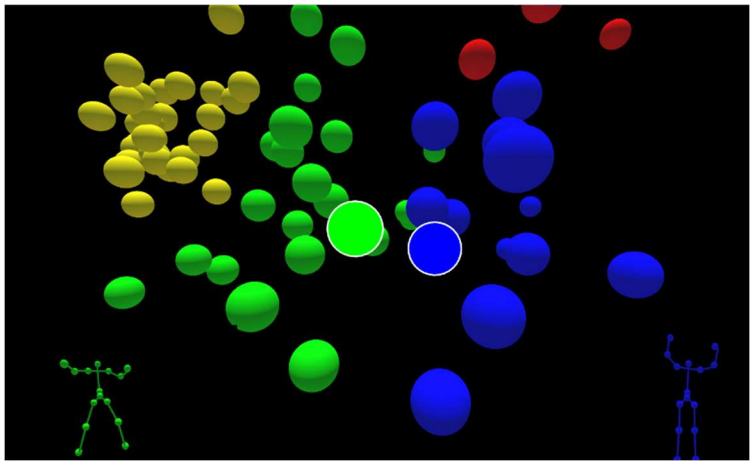
The size of the gesture points can also be scaled by the user. Making these gesture points very large allows them to better convey the general shape of a cluster, whereas making them smaller allows for easier in-cluster exploration and comparison (Fig. 10).

Fig. 10: Comparison Screenshots Showing Gesture Points of Varying size



The behaviors described above we consider part of the "overview" screen. The following behaviors described below constitute the "detail" overlay. By pressing the "alt" key, the user unlocks the mouse from controlling the camera and is free to move the cursor around. By clicking on different gesture points, the user is able to control the two small sub-views and their associated avatars located on the bottom left and right; clicking on a gesture point assigns the avatar on the left or right, determined by mouse click type, the motion capture data corresponding to the clicked gesture point. The "selected" gesture points are also identified with a white aura around them (Fig. 11).

Fig.11: Screenshot from the Tool showing Detail Overlay and Selection Highlighting



4.1 Design Guidelines: Schneiderman's Mantra

Schneiderman famous mantra for information visualization design "overview first, zoom and filter, then details on demand" [19]. The phrase o*verview first* [19] is a guideline which suggests that all info visualizations should begin with presenting the user an abstract and high-level representation of some more-than-substantial part of the dataset. This *overview* is necessary because it provides a frame of reference and context to the dataset. *Zoom and Filter* [19] recommends data visualizations provide the user with a way to discard or unselect data points that aren't relevant or of interest to the user's exploratory goal. One example would be to zoom in closely on the city the user wants to explore in Google Maps, as opposed to viewing the entire globe. Finally, *details on demand* [19] advocates for a feedback loop consisting of the user actively seeking out additional relevant information as they see fit, rather than having this information force-fed to them. The purpose of this step is to ensure that once the user can locate a subset of data cases they're interested in, they are not constricted by it and able to iteratively refine their exploration criteria and the specifications of their desired knowledge.

We believe that our data visualization satisfies the criteria put forth by Schneiderman. The *overview* screen provides a high-level overview of the data that summarizes the shape of clusters and the associated distribution of gesture points. By controlling the positioning of the camera, the user can interactively and iteratively display a subset of gesture points that they're interested in. The mouse clicking and detail overly both provide details on demand and assist in the zoom and filter step by making it easier to investigate the individual gestures of a cluster without changing the current subset of points on display. Furthermore, changing the activation threshold determines which of these subset gesture points shown should perform their associated animations. Making all the points display their avatars will allow the user to compare individual animations in this subset and gain insight to what kind of gestures characterize it. Showing only the spherical gesture points gives the user knowledge about the shape of this subset of gesture points. This feature is assisted by scaling the gesture points or avatars up or down as per user control.

4.2 Design Guidelines: Tufte's Other Principles

In addition to adhering to Tufte's mantra, the design of the visualization tool also obeys several other established principles such as Graphical Integrity, the Data-Ink Ratio and avoiding Chartjunk [18]. Tufte's conception of graphical integrity emphasizes "telling the truth". Visual representations of data should neither over or under-represent its effects and phenomena visually [18]. As such, the graphical

representation of numbers and objects must be directly proportional and commensurate with the data's quantitative elements. Our tool achieves this by having the position of the gesture points be linearly proportional to their respective output value after pre-processing. Tufte's data-ink ratio is a principle that suggests an information visualization that minimizes the amount of 'ink' used while maximizing the amount of meaningful data conveyed is a good one [18]. Our implementation accomplishes this by having only the gesture points and animated avatars be colored amidst black space. We chose to omit any axes or axes markings due to our gesture point positions being determined by PCA, which produces output values with no consistent meaning to them in most situations. Any relevant information regarding a point's position is to be inferred by the user by examining neighboring points. The emergent behavior described earlier regarding the semantic meaning of axes values is thus preserved by this approach while avoiding visual noise. This design choice also avoids "chartjunk", which is described by Tufte as illustrations and graphical effects that are unnecessary and serve as visual candy [18].

4.3 Results and Comparative Analysis

The original visualization tool's operation consisted of two different modes. The first we refer to as the *overview* screen (fig. 12). This screen's main function is to provide a frame of reference and summary of the gesture-point distribution's characteristics. Each gesture is represented as a colored circle, where the position of the circle is determined by the product of our pre-processing pipeline, with a dimensionality output set to two, for that respective gesture. The color of each gesture point encodes the cluster assignment of that gesture. In this mode, the user may not move the camera nor zoom in on any specific point. By clicking on each gesture, the user is presented with a new browser window displaying the second operation mode we refer to as the *visualizer* screen (Fig. 13). The visualizer, being coded in 3js, enjoys several benefits such as being WEBGL enabled, easily hostable on a server and having a low system overhead. Its layout is a wireframe avatar, with large spheres representing human joints in a fixed pose at any given time. The pose of the avatar is determined by the current frame from its respective motion capture data. The user is able to cycle the frame displayed by the avatar by pressing the A or D keys, which increment or decrement the current frame counter by one. This avatar's motion is not continuous and non-interpolating.

Fig. 12: Screenshot from Original Visualization Tool Overview Screen

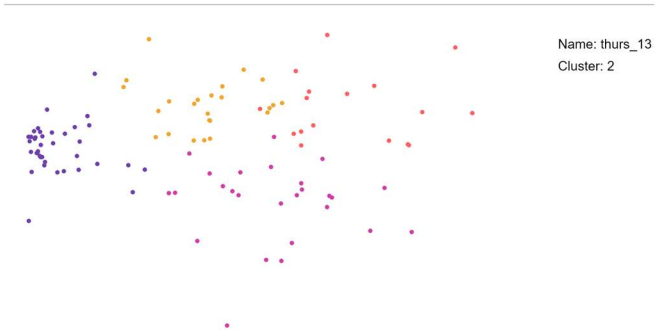
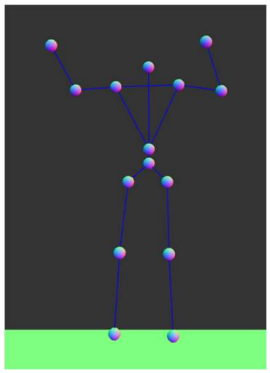


Fig. 13: Screenshot from Original Visualization Tool Animation VIewer

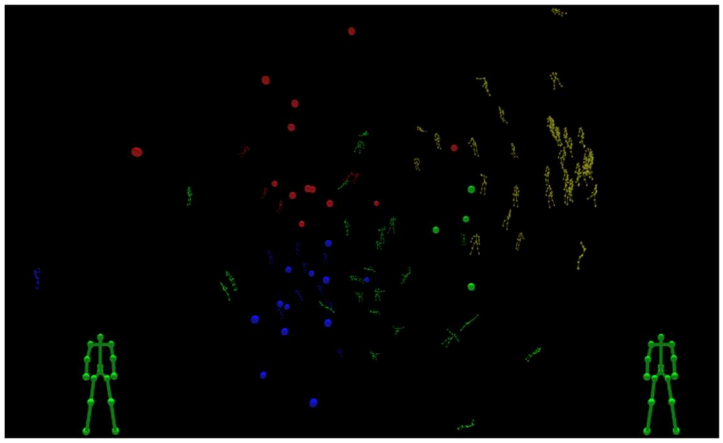


One of the biggest problems we experience with this current arrangement is that the user has no understanding of what the gesture could look like before clicking on it. Having the user divert their attention from the overview screen to another breaks the flow of user interaction and is intensely distracting if not frustrating. Furthermore, the user will likely click on several different gesture points hoping to understand the specifics of each cluster resulting in numerous open windows whose management and placement is needless tedium. Finally, the technical limitations of using D3 and 3JS together prevent 3js from communicating to D3 which visualizers/windows are currently open. Consequently, the overview screen is unable to give the user any indication that a specific gesture's visualizer is currently open or what gestures the user has clicked on.

Our second iteration of the visualization tool provides several enhancements over the original. The overview mode now displays individual gesture information by replacing gesture points with animated avatars in the same position should the camera get close enough. This completely alleviates any difficulties born from having two discrete screens and the associated window management problem. Because all of the code for the second iteration runs within the Unity engine, we can easily highlight user selected gesture points with a white border. Though the current implementation only has two separate detailed gesture viewers, the code foundation allows easy expansion to four or even six.

Unity also allows dynamic rescaling of the avatar to be bigger or smaller as the user sees fit, in addition to the possibility of a VR-enabled implementation. We believe that virtual reality can greatly enhance the user experience due to the visceral sensation of moving the camera close enough to a gesture point that the avatar appears roughly the size of a human being. All of this is accomplished while maintaining a simple and intuitive user interface. Though Unity is more performance intensive, it still is lightweight enough that the tool can be built to function on a WEBGL rendering backend should this option be selected. This extra performance overhead is effectively negligible relative to the computational load handled by the tool in rendering several dozen animated avatars performing different looping gestures at a high framerate.

Fig. 14: Screenshot of from Normal Tool Usage



Finally, having the gestures be output in three dimensions results in much less information loss regarding gesture similarity. On a qualitative level, several gesture pairings that appeared close together on the original two-dimensional visualization were shown to be quite distant and unviable in three dimensions. The natural consequence of this is that the second design is much more precise in presenting the similarities and differences between gestures, a core objective that spurned its creation. As such, we believe that this second iteration visualization tool is the superior way to experience LuminAI's pre-processing pipeline in a human-intuitive manner.

5. Future work

We plan investigate how to mitigate some of the limitations of the pipeline highlighted in the evaluation section. This will include technical pipeline efficiency and accuracy improvements as well as collecting a larger dataset and exploring more rigorous methods of assessing clustering quality. Further work is necessary to fully understand the ability of the pipeline to find meaningful clusters for larger datasets that contain varied gestures that involve the motion of many body parts at one time. This was challenging to assess using our preliminary approach to evaluation, both because we could not visualize clustering visualizations with more than two dimensions, and because it was difficult for us to come up with meaningful labels for full-body gestures. In the future, we might explore how the algorithm performs in relationship to labeled datasets generated by expert dancers/choreographers who are able to more accurately label complex movements and/or investigate whether or not users of the system can discern a difference in gesture responses generated using our pipeline vs. random responses. In addition, the gesture clustering pipeline we have built can support reasoning/clustering along non-skeletal metrics of similarity, such as Laban movement analysis. We plan to further explore how different ways of reasoning about movement can affect co-creative movement improvisation within the context of LuminAI. We also plan to further improve upon the design of the visualization tool by expanding on its current functionality. The highest priority would be adding support for VR controls in navigating the camera and selecting gestures for detailed view. Additional work would include adding user interface elements that would enable rapid switching between different datasets, motion capture dataset editing and the adjustment of other display elements like gesture sphere color palette or animation speed.

6. Conclusion

In this paper, we have combined multiple strategies used for gesture dimensionality reduction with a k-means clustering technique to develop a pipeline for unsupervised clustering of arbitrary full-body motion data. We conducted a preliminary evaluation of our pipeline and found that it is able to efficiently and

intuitively cluster gestures involving the movement of isolated body parts. The pipeline is resilient to noisy data and produces clear clusters in response to gestures that are intuitively similar in terms of skeletal positioning. Our main contribution is the novel combination of existing strategies for clustering and dimensionality reduction into a pipeline that can be used in a variety of movement improvisation domains. Our secondary contribution is the design and evaluation of an effective and novel information visualization tool that aids in understanding the behaviors of the aforementioned pipeline.

7. References

1. Koray Balci and Lale Akarun. 2008. Clustering poses of motion capture data using limb centroids. In 2008 23rd International Symposium on Computer and Information Sciences. IEEE, 1–6.
2. Adrian Ball, David Rye, Fabio Ramos, and Mari Velonaki. 20r1. A comparison of unsupervised learning algorithms for gesture clustering. In 2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI). IEEE, 111–112.
3. Amit Bleiweiss, Dagan Eshar, Gershom Kutliroff, Alon Lerner, Yinon Oshrat, and Yaron Yanai. 2010. Enhanced Interactive Gaming by Blending Full-body Tracking and Gesture Animation. In ACM SIGGRAPH ASIA 2010 Sketches (SA '10). ACM, New York, NY, USA, 34:1–34:2. https://doi.org/10.1145/1899950.1899984
4. Sarah Fdili Alaoui, Jules Françoise, Thecla Schiphorst, Karen Studd, and Frédéric Bevilacqua. 2017. Seeing, Sensing and Recognizing Laban Movement Qualities. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. ACM, 4009–4020.
5. Bernadette Hecox, Ellen Levine, and Diana Scott. 1976. Dance in physical rehabilitation. Physical therapy 56, 8 (1976), 919–924.
6. Dohyung Kim, Dong-Hyeon Kim, and Keun-Chang Kwak. 2017. Classification of K-Pop dance movements based on skeleton information obtained by a Kinect sensor. Sensors 17, 6 (2017), 1261
7. Rudolf Laban and Lisa Ullmann. 1971. Mastery of Movement (3rd ed.). Macdonald & Evans Ltd, London, United Kingdom.
8. Brian Magerko, Christopher DeLeon, and Peter Dohogne. 2011. Digital Improvisational Theatre: Party Quirks. AAAI Press, Reykjavík, Iceland
9. Mikhail Jacob, Gaëtan Coisne, Akshay Gupta, Ivan Sysoev, Gaurav Verma, and Brian Magerko. 2013. Viewpoints AI. http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/view/7398
10. Stephen O'Hara, Yui Man Lui, and Bruce A Draper. 2011. Unsupervised learning of human expressions, gestures, and actions. In Face and Gesture 2011. IEEE, 1–8.
11. R Keith Sawyer. 2006. Group creativity: Musical performance and collaboration. Psychology of Music 34, 2 (2006), 148–165.
12. Daniel L Silver, Qiang Yang, and Lianghao Li. 2013. Lifelong Machine Learning Systems: Beyond Learning Algorithms.. In AAAI Spring Symposium: Lifelong Machine Learning, Vol. 13. 05.
13. Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. 2014. A survey of dimensionality reduction techniques. arXiv preprint arXiv:1403.2877 (2014).
14. Tanu Srivastava, Raj Shree Singh, Sunil Kumar, and Pavan Chakraborty. 2017. Feasibility of Principal Component Analysis in hand gesture recognition system. arXiv preprint arXiv:1702.07371 (2017).
15. Tian-Shu Wang, Heung-Yeung Shum, Ying-Qing Xu, and Nan-Ning Zheng. 2001. Unsupervised analysis of human gestures. In Pacifc-Rim Conference on Multimedia. Springer, 174–181.
16. Dongkuan Xu and Yingjie Tian. 2015. A comprehensive survey of clustering algorithms. Annals of Data Science 2, 2 (2015), 165–193.
17. Yang Yang, Hubert PH Shum, Nauman Aslam, and Lanling Zeng. 2016. Temporal clustering of motion capture data with optimal partitioning. In Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry-Volume 1. ACM, 479–482.
18. Tufte, Edward R. The Visual Display of Quantitative Information (1983). Print.

19. Ben Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96). IEEE Computer Society, Washington, DC, USA, 336-.