Laban Driven 3D Pose Generation

Joseph Hines
College of Computing and Informatics
Drexel University
Philadelphia, Pennsylvania
jth95

Devan Juraniec
College of Engineering
Drexel University
Philadelphia, Pennsylvania
drj48

Abishek S Kumaar
Digital Media
Drexel University
Philadelphia, Pennsylvania
ask85

I. Abstract

Laban movement analysis allows researchers across many fields to observe, describe, and categorize human movement in a structured and methodical fashion as in computer vision, computer animation, and game agents. Our work enables users to apply an emotional style to an existing pose of a rendering of a figure. By providing the pose and the desired emotional connotation derived from Laban Analysis, our model will generate a new pose that is similar to the source pose but expresses the desired emotion. A motion database of dances labelled by Laban Analysis created by A. Aristidou et al. [1] and the work done by Zhou et al. [2] to generative-ly apply sets of movement characteristics to create stylized animation have inspired our unique application of Laban Analysis.

II. Introduction

Laban Movement Analysis defines the qualitative aspects of human movement categorically through concepts of space harmony, effort, body, and shape [3]. Although this style of analysis has proven itself useful in several areas of study, like theatre and physical therapy, some of its most innovative applications are in the areas of animation and robotics laban. Laban Analysis has been used to improve the likeness of animated character's movements to that of a human with the goal of creating believable emotions. Our study takes Laban Analysis along with its animation and robotics applications and applies it to a rendering of a figure. The model is given an emotional state that it uses as a prompt to transform the figure from its initial pose to its next emotional state, which is calculated as described in the next section.

III. Approach

For the purpose of simplicity, we allow five emotional states: afraid, excited, relaxed, bored, and neutral. The first four are extremes in emotional qualities on a 2D spectrum containing 4 quadrants known as Russell's circumplex model of emotion; refer to fig. 1 of appendix. Each extreme has its own Laban movement qualities with assigned numerical ranges of movement. In order to train the system, the data must be broken up and labeled based on their Laban metrics. Each input motion sequence is broken up into poses or frames, and these poses are saved and labeled within our dataset. The poses themselves are comprised of the 27 features for Laban analysis, as shown in fig. 2 from the set.

A. Pre-processing data

Each of the source videos from the motion database comes labeled with the emotion the video is primarily expressing (e.g. "Happy"). We apply this label to all frames that come from that recording. For each frame within the video, we translate the points of the figure to its coordinates in the space and label the point based on its location on the body. One of the issues we ran into while pre-processing the data was the discrepancies among the placement of points on the figures across the videos in the database. For example, on some figures, there was a point located where the knee is, but on others, the point closest to that location was on the lower calf. In addition to variance in the in the placement of points on the figure, the order in which the data was represented was not consistent. In one video, the points may be listed in shoulder, elbow, wrist order, but on another video, it may be listed as wrist, shoulder, elbow. After some investigation, we noticed that the points used 1 of 2 orderings, which we were able to detect by checking the distance between some points.

The data was downloaded in .c3d format and parsed using the *c3d* python module. This provides an iterator over all frames of a capture, which was used to iterate over the frames, process, and save them. Each input file produces a numpy array file that contains all of the parsed data, and is saved to disc. These data files can then be loaded from disk, augmented, shuffled, and used as input for training and testing. At various points throughout this process, the data is checked for invalid poses and removes some duplicate poses. This helps ensure that the data quality passed to the model is high and that the provided frames are as distinct as possible, allowing the model to better differentiate between each emotion.

Each pose is represented by the 38 dots found on the figure, its x, y, and z values, and its 17 features, for a total of 131 representative elements. From the select categories within the database from which we drew our data, our initial dataset included around 1 million entries. After the data is processed, 30,000 distinct frames were found in the dataset. This data is then augmented by a number of random scales and translations to each pose. In addition to the data we gathered from the database, we added an additional 10% - 20% of the total input

in random noise to the generator's input in order to prevent the results of our training from becoming too deterministic.

B. Accessing Data and Setup

During the process of loading the data and setting the system up to train, emotion labels were embedded, which changed the descriptive labels of the names of emotions to integers that the generator and discriminator could work with. Several poses had values of 0 for multiple points on the figure and needed to be eliminated from the dataset to avoid skewing the results of training. From the single pose's coordinates, we had a cap of the acceptable number of 0's in the coordinates for the pose. Any pose with less than 3 0's in its data was set to true and kept in the data set. Any pose with 10 or more 0's was set to false and removed. Each value was cast back to 32 bit float and then to pytorch tensor. At this stage, we also split the data to use 80% for training and 20% for testing. When the data was loaded using an implementation of pytorch dataset, the data loader shuffled the data, separating the data itself from the emotion labels.

The generator and discriminator are modeled off of the work of Zhou et al. [2]. Both the generator and discriminator have encoders and decoders that consist of several convolutional layers and convolutional transpose layers. The residual convolutional layer takes input and output channels, kernel size, stride, padding, ratio as parameters to calculate the root ratio with a value between 0 and 1 as a float. The result of the residual convolutional layers is then passed to a forward function to calculate the affine transform and the PReLU function as described in Zhou et al. [2]. The process only needs our forward function and the image to be completed. The forward function converts the float to a tensor and adds the ratio multiplied by x to the result of the PReLU affine calculation. We wanted to preserve some data from the previous layer, which is x', our input tensor, while y continues on as our output. This process, in conjunction with the transpose residual convolutional layers, work together in both the generator and the discriminator to reduce the input size and later expand it back to its original dimensions.

At this point in the setup, checks are done to determine if cuda was available and if the generator, discriminator, and the adversarial loss can be moved to the GPU's memory. If both are a available, the aforementioned classes are moved. The data loader will go over the dataset and move it to the new memory location.

Along with this model, we created three other versions during our experiments. Zhou et al. [2] specified a scheme for improving convergence by passing the local joint translations as input to the generator and having the generator output the local joint rotations. Then, through a constrained forward kinematics process, the model's output is converted back into global joint positions. To this end, we created a version of our dataset and model that converts all poses to local translations by subtracting the root (pelvis) point from every point in the pose, effectively centering the pose at the origin. Aside from this, we also created a version that used a simplified version of the pose, reducing the 38 points to 16. This simplified view of the model can more easily be created through forward kinematics. The generator was modified to produce local rotations, which were used to return to global coordinates.

Finally, we created a version of the model that used the local translation data and setup (without the rotation output), but with a much different generator and discriminator. Since the architecture from the Generative Tweening paper was considered most with the temporal aspect of poses, we wanted to investigate how well it fit our modified version of the problem. To this end, we made a simplified version of the model that use fully connected layers with batch normalization and the same PRelu activation function.

C. Generator

In order for our model to be able to work with the emotion labels during training, the generator itself provides a label for each row of the data in the batch with a number 0-4 for the associated emotion. This process of embedding and converting categorical data can be done with one-hot encoding or a similar process. To encode and decode, we created similar convolutional layers to Zhou et al. [2] as mentioned earlier, but we ended up applying a more simplified method, reducing by larger fractions than N/64 as was done in our reference. Changing the kernel size is one of the major changes we made, and we maintained the same number of layers, for both the encoder (6) and decoder (8). When called, the method passes data through all layers, allowing us to perform backpropagation. During the encoding process, we upsample, passing a fraction parameter, which is a ratio that gets progressively smaller throughout the layers. Progressively smaller ratios are also passed as parameters during the decoding process, but unlike the encoder, the decoder performs downsampling.

We then pass the data through a forward function with pose, label, and noise attributes to append the pose, to embed a label (0-4), returning one hot encoding, and to add noise. With that output,we called the model to output a tensor to be passed on to the discriminator.

D. Discriminator

Our discriminator is almost identical to what was used in Zhou et al. [2] and extracts the meaning from the data and makes a classification. Like the generator, the discriminator uses several residual convolutional and transpose residual convolutional layers to break down the data and derive meaning from the input of the generator. By leveraging pytorch's LabelEmbedding, the discriminator is able to easily represent the multiple emotional classes that we have chosen.

E. Loss

Our loss functions are implemented using the Pytorch 'nn' module's cross entropy method and are derived for both the generator and discriminator as cross entropy losses. The generator loss can be described as how well the generator performs in fooling the discriminator as the discriminator determines what is a real pose and what is a fake pose. Mean

squared error loss was also considered initially for the first setup but was changed to cross entropy loss. It should be noted that the final loss for the discriminator is an averaging of the real and fake losses, which allows us to prevent skewing towards one loss in particular. Throughout the process, we experimented with various loss functions such as CrossEntropy and MSE.

F. Training

For input to our model, we take a single pose, defined by the 38 3D points, the 17 features as mentioned prior, a desired emotional label, and a representation of the target style. This representation draws on the Motion DNA concept from Zhou et. al [2]. This allows for the general feel of the emotion type to be encoded and to influence the input pose to be transformed to the desired emotion. The output is a pose defined by the same 38 points that better match the style of the desired emotion, while staying true to the input pose.

For training, the generator takes the input, and produces the defined output. The discriminator is fed "real" poses of the target pose type and the "fake" generated poses. The generator and discriminator alternate training, allowing each to be properly updated without conflating the losses. During training, we generate poses with random emotional values to prevent over-fitting. This process allows us to encode the notion of Laban metrics into a neural net, which can then properly modify an input pose to give it a specified emotional feel.

Since the model is striving to generate a novel frame/pose with a specified emotional label, it has two tasks. The output of the generator must be "real" enough for the discriminator to not immediately throw it away. Along with this, the generated data must match the specified label. When provided "sad", the generator must create output that is both believable, resembles the input pose, and is sad, as defined by Laban analysis and clustering in the 2D map.

The training process for the generator is performed by supplying it with input data, target labels, and noise. The data for each pose is treated as the starting pose. The accompanying labels are shuffled so that they no longer match the input pose. The new label is treated as the user's input, which is the emotion that the generator needs to transform the starting pose into.

Across the various versions of the model, there are three different input and output sizes. The version based on Zhou et al. [2] takes $114 + 17 + N_{noise}$ values per pose as input and outputs 114 $(38 * 3)$ values. The local translation version is largely the same, omitting the features for an input of $114 + N_{noise}$ values and 114 output values. The rotation version takes $48 + N_{noise}$ values for input and outputs 45 values. The inputs are 16 3D points and the outputs are 15 3D local rotations.

The training for the discriminator involves determining which poses are real and which poses are fake. Our discriminator trains on "real" poses from our dataset and the "fake" poses created by the generator. As stated earlier, the discriminator and generator are trained at separate times in order to strengthen each individually.

During the initial training phases, the results of the generated poses were not ideal, and upon further investigation, we found that the datasets for both bored and afraid contained a large amount of t-poses. We were able to make this discovery by coloring each pose based on its label, which was a useful addition to help visualize the data within each dataset and to evaluate the quality of what the model was being trained on. In order to rectify this issue, we added a function that takes a matrix with one pose per row and removes rows that are close based on a threshold of tolerance. After tuning that parameter we were able to reduce our dataset, only including poses that are on more extreme ends of each emotion to help the model distinguish among them. In addition, we also added data augmentation to improve the overall quality of our datasets by combining random scaling and translation. By doing so, we were able to take the reduced dataset, which contained 30,000 unique frames and expand it to any desired scale of complexity. The augmented dataset was smaller than our initial dataset, but contained higher quality data that yielded better results.

Due to the complexity of GAN's and the nature of the generator and discriminator essentially playing a zero-sum game, an unstable or unbalanced setup will not converge to anything meaningful. An area where we struggled was in finding balance. The generator needs to have access to enough novelty in order to continue to explore the space and to trick the discriminator. Without enough, the discriminator will easily identify the generator's fake images, but, with too much, the discriminator will not be able to identify the fake images. One way we approached resolving the issue of balance was by altering the amount of noise, which can be tuned by adjusting the size of the latent space, the space that the generator explored to find novel output. In addition, the batch size, learning rate, and model architecture will impact this as well.

Our initial experience with this issue resulted from a small latent space where the generator did not have enough randomness to come up with solutions that were different and discrete enough from its training data. The discriminator quickly tended to 0, and the generator could not get below a 1 mean squared error. To counter this, we raised the latent space to 128, but did not see improvement. This change gave the generator too much randomness and allowed it to constantly fool the discriminator. Based on the results, we lowered to using 32 noise values.

Ideally the generator and discriminator losses would oscillate with respect to each other. As the discriminator learns the generators current tricks causing the generator's gradients to point somewhere new. The generator should then create a novel solution, lowering its loss while raising the discriminator's, in other words, fooling the discriminator. The discriminator would learn this new trick and learn to account for it, causing its loss to lower. Overtime, in an ideal and stable setup, these oscillations would occur less frequently and with

lower magnitude. This means that the generator has found what the discriminator is looking for, and is able to create new poses that both match the target labels and are classified as real by the discriminator.

IV. Evaluation Approach

When passing the output from the generator through the discriminator, the "real" or "fake" classification it assigns to the generator's output is the basis for our evaluation of the generator's ability to produce output that fits the parameters of our defined Laban analysis. Therefore, in order for our results to be accurate, the discriminator must be well-trained and accurate in its classifications. Our evaluation of the generator's success is also measured by calculating loss from the discriminator classification. We then back-propagate through both the discriminator and generator to obtain gradients, and use the gradients to change only the generator weights. Since we did not have time to automatically analyze if the generator matched the target emotional label, we had to review select outputs manually.

Overall, we have created a GAN that learns how to generate poses from the provided input. We overcame many issues over the course of the process, and feel confident that we could have a feasible solution if we could overcome the mode collapse problem. Once this is done, we could further analyze our model and its output to see how well its output matches the emotional target.

A. Introducing local translations and rotations for generator, Range constrained forward kinematics

One way the output of the generator could be more coherent and allow for convergence is to implement a constraint on the allowable angles for output poses. This allows the gradient descent to converge on a value appropriately and not miss. The parameters allow the joint rotations to be constrained to range of values, which was implemented in Zhou et. al [2].

Many public motion datasets do not capture the joint rotations directly, but instead use markers mounted on human performers to capture the joint coordinates, which are then transformed to rotations. Due to insufficient number of markers, the imprecision of the capture system, and the data noise, it is not always guaranteed that the calculated rotations are continuous in time or within the range of joint flexibility. In contrast, joint positional representation is guaranteed to be continuous and is suitable for deep neural network training Zhou et. al [2]. Therefore, we use the joint translations as the input to our generator and discriminator. However, for the motion synthesized by the generator network, we prefer to use a rotation representation, since we want to guarantee the invariance of the bone length and to restrict the joint rotation ranges according to the flexibility of the human body. These kinds of hard constraints would be more difficult to enforce for a joint positional representation.

V. Related Work and Novelty

The following bodies of work are our inspiration. The first one deals with imparting style in motion poses while using a GAN in a generative sense of modelling stylistic motion creation. It also aids in reviewing emotion classification of motion capture sequences, other prior art for this project deal with Laban motion analysis as imparting a formalism in labelling the various humanoid movements that showcase a mood, emotion, or personality.

A. Generative Tweening:

Zhou et. al [2] used input poses at sparse key frames, a GAN, and an input database of motions to create the in between poses to create a continuous set of motion states. They use a concept called a motion DNA, which is a random sampling from the motion dataset (CMU motion capture data set). They convolved the selected motion set as an average pooling with the input layers of the generator. Then by using a loss function for the generator, they created an output motion style that was indicative of the sparse motion key frame and the selected motion sampling. The discriminator of the GAN decided if the style was real or fake. A good output is reached when the in between filler styles and their constituent poses is considered real, as shown in fig.10.

While this research is definitely a step forward in creative motion generation, the project does not have a metric for a specific input style and is a randomized selection of poses from the motion dataset. The result is unreal and sometimes produces inconsistent styles that have no meaning. We hope to inform the style generation method with a metric of what kind of poses constitute a certain kind of mood or personality. The user may select the desired mood/personality thereby creating a more plausible and user driven output motion style. We will be using a method similar to the motion DNA method implemented in this paper.

B. Emotion classifying based on laban parameters:

This paper is crucial in our implementation and conceptual outlook. Their aim was to classify input dance poses into various emotional states as depicted in a model of emotion classification known as the circumplex model. The circumplex model provides a visual representation on a 2D graph of various emotion representations. The group asked dancers to perform the same type of dance in twelve different styles of emotions. The paper then lists 27 laban features or categories that inform the classifier during training, such as what to extract from the current motion style in order to classify a certain emotional style. Various methods were used including Naive bayes classifer and PCA in classifying emotional styles based on laban features of the input poses. These classifications were mapped onto the circumplex model and validated. There are two advantages to citing this method. First, the motion database we use will be the same as the one used here. They have labelled emotions for laban features in the database poses that we can directly extract and apply to our GAN method. Second, we have metrics to calculate laban features for any input pose. If we want our GAN to enforce style and pick said style from the database based on any specific metric, rather

than just the emotion label themselves, then that is possible as well.

C. Performance:

In a creative environment, such as stylized motion generation, it makes sense that a certain style generated be validated by an expert, refer to fig.4. Durupinar et. al[4] used a perceptual approach to determine if poses changed by some manual factors can be perceived as an emotional or personality change. Laban features were tweaked manually by an animator using certain metrics to show an altered pose/state. A Laban expert was then asked to verify if said output style was indeed an intended one and validated their method. There is no automatic generation of any kind in regard to the Laban style generation and using a GAN to perform the same provides a degree of novelty to this method.

D. Novelty

Our work contributing to the application of Laban Movement Analysis in AI differs from the work we used as inspiration in both concept and application. Although we were heavily influenced by the work of Zhou et. al [2] and the researchers' approach to using Laban Analysis to generate animation based on certain styles of movement, like Indian dance and martial arts, our approach and goals for our research are to modify poses to better express an emotion as defined by Laban metrics. Our work is implemented with the goal of training the model to generate an accurate and believable representation of an emotion while maintaining the integrity of the initial pose. Future applications for this study include using it as a tool or a starting point for animators and game creators.

VI. Conclusions

The application of Laban Analysis in AI has been based on a compilation of several studies inspired by the potential they hold to simplify and assist in processes including animation and game creation. Future applications of our research could be applied to animation as a jumping off point for animators to transition a character from one emotion to another, or as a way for game developers to streamline the process of creating reactions of non-player characters.

In game environments, virtual agents are ever evolving to be more expressive and embodied in the environment they are present in. As using the methods listed in this paper, laban metrics can be enforced as a formalism to create new poses automatically and also imbue this ability to change its poses based on prior parameters in behavior modelling as well.

One more extension is the ability to parametrize actions using physics based techniques coupled with Laban Analysis for creating reactive and on the fly pose generation or for smooth transitioning, which is still an open problem in computer graphics. The physics system would inform the system of Laban parameters and generate new poses based on a required internal state. This provides a two fold interactive virtual agent with both physics interactions, as in a depiction of objects as real matter, and with emotional coupling, which is a more accurate model of life. However, a precursor for this kind of simulation would be a Laban labeling and classification of physics based objects not limited to inverse kinematics.

References

[1] P. Charalambous A. Aristidou and Y. Chrysanthou. "emotion analysis and classification: Understanding the performers' emotions using the lma entities". *Computer Graphics Forum*, 34, no. 6:262–276, 2015.

[2] C. Barnes J. Yang S. Xiang Y. Zhou, J. Lu and H. Li. *[Online]. Available: https://arxiv.org/pdf/2005.08891.pdf. [Accessed: 09-Oct-2020]*, May 2020.

[3] Movement analysis. *[Online]. Available: https://labaninternational.org/scope-of-practice/movement-analysis/.*

[4] Funda Durupinar, Mubbasir Kapadia, Susan Deutsch, Michael Neff, and Norman I Badler. Perform: Perceptual approach for adding ocean personality to human motion using laban movement analysis. *ACM Transactions on Graphics (TOG)*, 36(1):1–16, 2016.

VII. APPENDIX

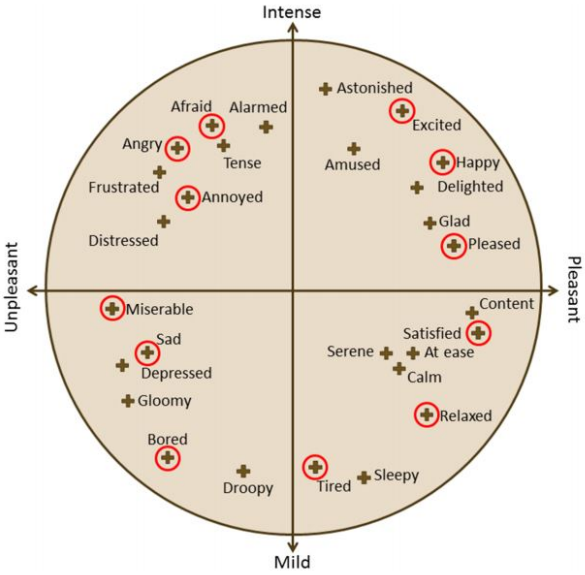

Fig. 1. The Russell's circumplex model of affect: arousal is represented by the vertical axis and valence by the horizontal axis [1], this will also be a template GUI for selecting an input emotion, we will have only 5 emotional states: afraid, excited, relaxed, miserable, ,neutral

	Measurement					
f_s	Description	max	min	mean	std	#
f_1	Feet–hip distance	ϕ_1	ϕ_2	ϕ_3	ϕ_4	
f_2	Hands–shoulder distance	ϕ_5	ϕ_6	ϕ_7	ϕ_8	
f_3	Hands distance	ϕ_9	ϕ_{10}	ϕ_{11}	ϕ_{12}	
f_4	Hands–head distance	ϕ_{13}	ϕ_{14}	ϕ_{15}	ϕ_{16}	
f_5	Pelvis height	ϕ_{17}	ϕ_{18}	ϕ_{19}	ϕ_{20}	
f_6	Hip-ground minus feet-hip	ϕ_{21}	ϕ_{21}	ϕ_{23}	ϕ_{24}	
f_7	Centroid height	ϕ_{25}	ϕ_{26}	ϕ_{27}	ϕ_{28}	
f_8	Centroid–pelvis distance	ϕ_{29}	ϕ_{30}	ϕ_{31}	ϕ_{32}	
f_9	Gait size	ϕ_{33}	ϕ_{34}	ϕ_{35}	ϕ_{36}	
f_{10}	Head orientation	ϕ_{37}	ϕ_{38}	ϕ_{39}		
f_{11}	Deceleration peaks					ϕ_{40}
f_{12}	Hip velocity	ϕ_{41}	ϕ_{42}		ϕ_{43}	
f_{13}	Hands velocity	ϕ_{44}	ϕ_{45}		ϕ_{46}	
f_{14}	Feet velocity	ϕ_{47}	ϕ_{48}		ϕ_{49}	
f_{15}	Hip acceleration	ϕ_{50}			ϕ_{51}	
f_{16}	Hands acceleration	ϕ_{52}			ϕ_{53}	
f_{17}	Feet acceleration	ϕ_{54}			ϕ_{55}	
f_{18}	Jerk	ϕ_{56}			ϕ_{57}	
f_{19}	Volume	ϕ_{58}	ϕ_{59}	ϕ_{60}	ϕ_{61}	
f_{20}	Volume (upper body)	ϕ_{62}	ϕ_{63}	ϕ_{64}	ϕ_{65}	
f_{21}	Volume (lower body)	ϕ_{66}	ϕ_{67}	ϕ_{68}	ϕ_{69}	
f_{22}	Volume (left side)	ϕ_{70}	ϕ_{71}	ϕ_{72}	ϕ_{73}	
f_{23}	Volume (right side)	ϕ_{74}	ϕ_{75}	ϕ_{76}	ϕ_{77}	
f_{24}	Torso height	ϕ_{78}	ϕ_{79}	ϕ_{80}	ϕ_{81}	
f_{25}	Hands level					ϕ_{82}–ϕ_{84}
f_{26}	Total distance					ϕ_{85}
f_{27}	Total area					ϕ_{86}

Fig. 2. Laban features for humanoid movement, these features can be calculated and quantified for motion capture poses and sequences

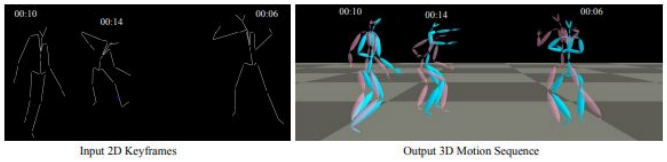

Fig. 3. Motion Generation given manually made 2D poses. Left: 2D keyframe inputs drawn on the x-y plane and their times[2]. Right: the synthesized 3D poses at keyframes (pink) and nearby frames (blue).

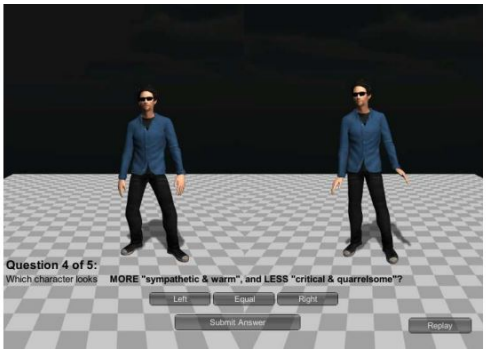

Fig. 4. A visual representation of the work done in the PERFROM paper, input pose is changed by some manual method and validated with a movement expert/laban expert

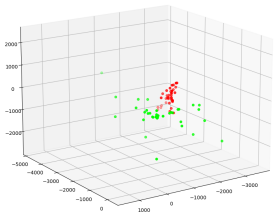

Fig. 5. Base version of the model struggling to converge

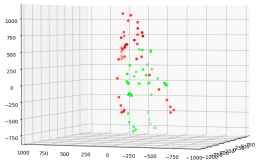

Fig. 6. Updated local translations version of the model with fully connected linear layers instead of convolutional layers

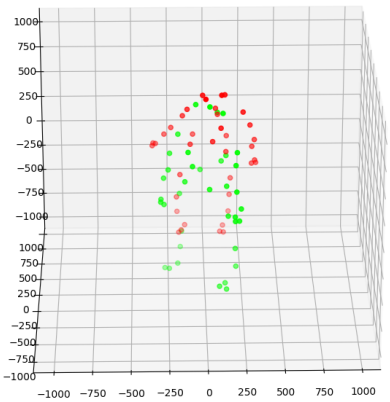

Fig. 7. Model outputting the same thing for each label signifying mode collapse

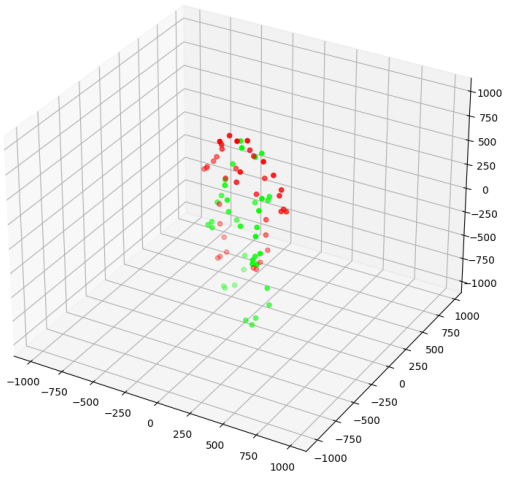

Fig. 9. Simplified version of the model (local translations with fully connected linear layers) converging, but not producing different output for different labels.

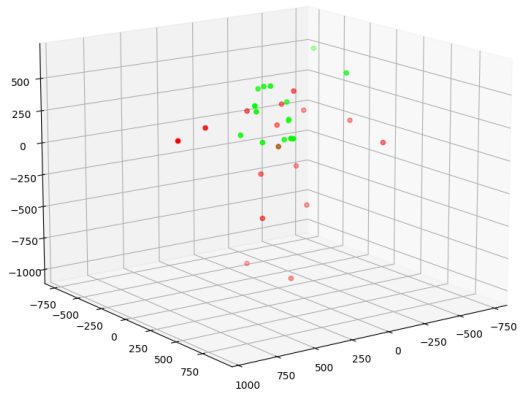

Fig. 8. Model with rotations (no constraints) struggling to converge

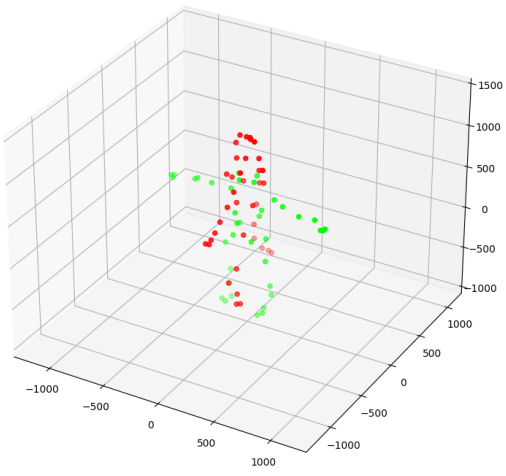

Fig. 10. Simple version converging to the optimal solution, which in this case was a t-pose as they had not all been pruned from the data