Motion Synthesis Using Relative Joint Distances

Anzela Kravcevich

Master in Computer Science (MCS)

University of Dublin, Trinity College

Supervisor: Rachel McDonnell

Submitted to the University of Dublin, Trinity College May 2014

Declaration

Summary

In the field of animation, creating complex and realistic looking motion for human body models is a very difficult and expensive process. Several research papers that are looking into this field specifically focus on motion synthesis, a method to create complex motion using only a set of basic motions. One such method is a motion graph, a directed graph that identifies transitions between motions. By interweaving between these motions, a longer or even infinite synthesized motion can be created. Typically short sets of basic walk motions are used, however this can also be applied to long dance motions. Motion synthesis from motion graphs involves two steps, the first involving motion graph construction and the second is efficient path finding from a directed graph based on terrain constraints. This dissertation focuses on motion graph construction, however it does provide motion synthesis to demonstrate effectiveness of the proposed method.

The first step of motion graph construction is identifying transitions points, which are frames of high similarity between two motions. The reason high similarity frames are chosen for transition, is due to the small amount of interpolation required between them to provide a smooth flow. The method for motion similarity used was primarily focused on [Tang et al., 2008], which introduces a new method that uses joint relative distances (JRD). Previous works on motion similarity [Kovar et al., 2002, Lee et al., 2002] focus on computing the sum of square distances between points in a motion point cloud. This method computes the joint distances in a single pose and compares them to the values of a second pose. It also ignores global rotations and translations, which normally would have to be dealt with to ensure proper motion similarity computation. Results are demonstrated using error function bitmap images.

The next step involves transition point extraction from these error functions. This involves locating clusters of high similarity between two motions and is performed using three pruning steps which ensure only the highly accurate transition points are selected. Transition points are stored in a file format that can be loaded into any application once a motion graph is constructed.

For the demonstration of this concept, three interactive applications where developed to show the motion graph construction effectiveness. The first demonstrates a dance motion transition between a set of dances. The second focuses on bipedal motion such as walking and jogging. The last demonstration attempts to simulate a simplistic sample application that could be used in real world environment, a dance floor is created with randomized dances performed by several virtual characters.

The contributions of this paper include an analysis of some of the methods used for motion graph construction in the field of animation. Also a new motion graph construction method is proposed that requires little supervision and can simulate high motion synthesis results.

Acknowledgements

I would like to thank my supervisor Rachel McDonnell for their advice and support throughout this project. I would also like to thank Ludovic Hoyet for their additional supervision.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Realistic human animation has played a large part in areas of entertainment, medicine and education. However, animation can be a laborious process without the necessary intuitive tools that could perform trivial work. The reason realistic animation is difficult to achieve is due to the complexity of the human body containing on average 206 bones which all affect body motion.

Motion capture systems have the capabilities to record complex human motions with a relative high accuracy. Depending on the number of joints recorded, the number of motions captured could range from subtle animations such as breathing to fast pace action like running. Free basic motion capture databases are available online such as CMU Graphics Lab [CMU, 2014] and HDM05 [HDM, 2014] for beginner animators. Clips such as walking, running, kicking etc. are considered basic motions, a more complex clip performing specific motion would involve a set of basics interweaved. Hence, for more specific motions, access to expensive motion capture equipment is required as well as skill and time. Due to the extensive setup process of a single motion capture session, having the ability to reuse and generate complex motions from a basic set will reduce the cost and time consumption of animation. Several animation tools exist to offer this service for people starting the field of animation, such as Blender, 3ds Max and Maya. Even such tools require human interaction in relation to parameter tweaking and path setup.

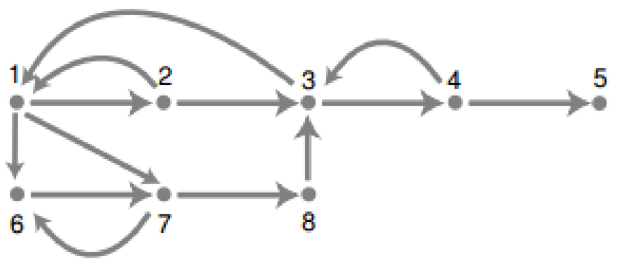


Fig. 1.1: A basic motion graph using two motions [Kovar et al., 2002]

One simple and automatic approach to solving this problem is using motion graphs. Directed graphs with frames of high similarity used as transition points between motion capture data. Two main areas of research in motion graphs are motion similarity for transition points and automatic generation of motion paths. This thesis will primarily focus on the former, evaluating different methods for motion similarity and extraction of transition frames between two motions. The Graphics, Vision, and Visualisation Group in Trinity College Dublin provided necessary motion capture clips for this project.

1.2 Aim

The focus of this research project is to develop a system that allows for synthesis of unique, infinite long motions that are of high quality and require minimum supervision. The main goal is to evaluate different methods for motion similarity and attempt to replicate an application that allows construction of motion graphs using a new method. Results will demonstrate the quality of transitions identified for the motion graph.

1.3 Outline

The report is structures as follows: the second chapter gives an overview of some of basic terminology of motion capture and animation as well as necessary background information used throughout the dissertation. The third chapter presents state of the art research work currently done on motion graphs. Chapter four describes implementation performed for motion graph construction starting from motion similarity, transition point location and utilisation. The fifth chapter gives an overview of experimentation performed to demonstrate motion synthesis based on the new motion graph construction method. The last chapter gives an overview to contributions and future work for the following dissertation.

Chapter 2

Background

2.1 Human Skeleton Model

In animation, a human body is represented as a rigid skeleton that controls a deformable skin. A skeleton hierarchy is composed of a series of joints with hierarchical relationships. Hence, each joint in a skeleton hierarchy is a child joint meaning it is attached to another joint and a parent joint meaning it has other joints attached to it, except for the root which is the highest joint in the hierarchy and can only be a parent joint. The root also represents the centre of gravity for the skeleton, meaning for any physical related computations, the total weight of the skeleton/character will be applied at this point. Translating this root node on the Cartesian plane, additionally moves all connected branches or joints hence moving the complete character in the environment. Joint rotations over time result in a character motion. Since the skeleton is represented in a tree like hierarchy, if a joint is rotated, it affects all child joint branches. A rotation or transition that is said to be local to a joint, means it only represents a transformation at that particular joint location as if the joint location was the center of its own Cartesian plane. A global transformation however represents a rotation or transition in respect to the complete Cartesian plane of the scene.

Depending on the desired level of accuracy and detail for a character motion, a skeleton could contain joints representing legs, torso, arms and head for simple bipedal motions or for further complexity bones such as fingers and facial markers. We use 29 joints as shown labelled in Figure 2.1. The motion capture files that where provided by the Graphics, Vision, and Visualisation group include 29 joints to record information of foot placement, arm, neck and back, enough to capture high quality body motion. Details such as face and fingers would provide unnecessary complexity.

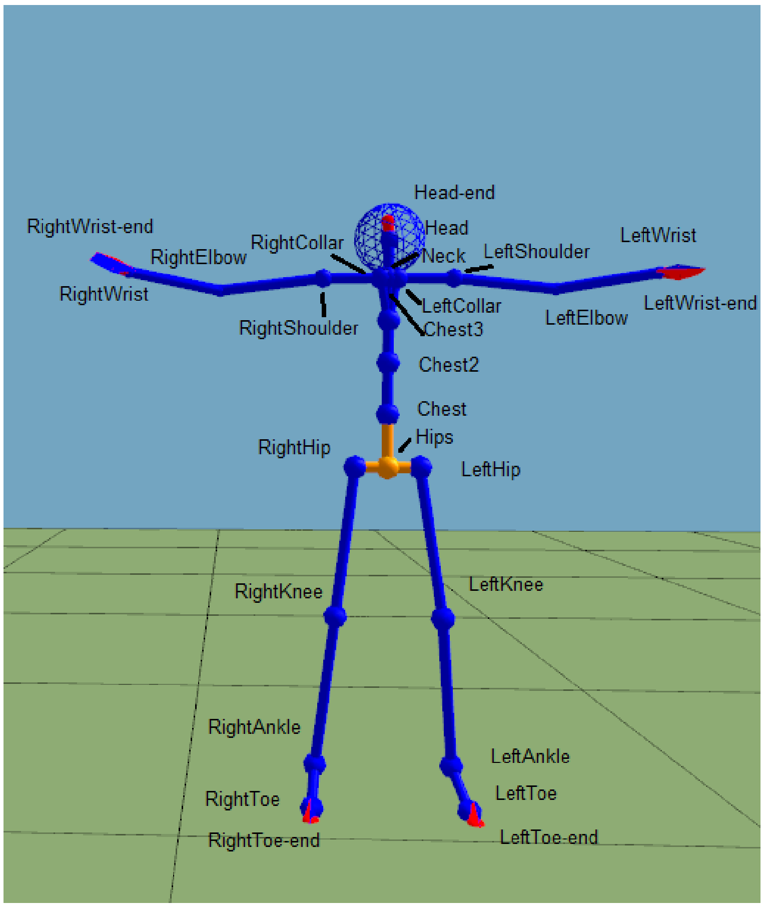


Fig. 2.1: Human skeleton model in the T pose, demonstrating 29 joints

2.2 Animation

Three main types of approaches exist to animate such a skeleton model: kinematics, dynamic and motion capture. Depending on the desired animation, the method chosen for animation could differ. Whether it is a physically correct human walk that looks robotic, or a fast yet expensive motion capture, a mix of approaches can be taken.

2.2.1 Kinematics

There are two main types of kinematics: one is done from observation called forward kinematics and the other one done from computation called inverse kinematics. Forward Kinematics relies on manual joint rotations specified by the animator. This method requires a lot of work and generally does not produce natural animations. Inverse Kinematics works oppositely, an animator is required to specify an end pose known as a keyframe (as well as intermediate poses if desired) and joint parameters are determined through computation. As an example, by identifying the final position on a Cartesian plain that is desired for the end-effector(finger) of an arm, individual joint rotations are computed for every joint starting from shoulder, elbow and wrist. A sample motion can be created by computing a spline from start to end position and interpolating between them over time to create animation. This method is used for inanimate objects or to complete a transition between one motion and another. However, it is possible to use keyframe interpolation to create very basic human motion, it is not viable for high quality complex motions.

2.2.2 Dynamics

Dynamic motions are computed from a set of torques around joints by applying physics and constraints such as degrees of freedom to produce natural looking motions. [Rose et al., 1996] An application to such a method is used to animate a character walking and regaining his balance based on his centre of symmetry. [Geijtenbeek et al., 2013]

The character is taught to react to environmental changes such as hills and drops, by providing physics based changes to the original walking motion. Even though this motion is viable to creating physically correct motion, it does not necessarily create the most realistic.

2.2.3 Motion Capture

Motion capture is an accurate way to record complex animations. Markers are placed on the human or animal model and multiple cameras are used to track the motion which is then used to animate digital character models in 2D or 3D computer animations. The main reason for using motion capture is to simulate realistic human motion that do not look robotic like dynamic or kinematic approaches. However, one of the drawback for this method is the difficult, time consuming and expensive setup. For motion capture: knowledge of the technology is required, calibration of cameras and lights is time consuming as well as paying for actors, animators and equipment is expensive. Figure 2.2 demonstrates a motion capture session in progress in the Graphics, Vision, and Visualisation Group in Trinity College Dublin.

Typical motion capture session involves:

- Actor suit up in a Velcro body suit and markers are placed on approximate locations of joints on the human.
- Camera calibration around the model, to ensure every marker is within range.
- Initial T pose, where a character stands with outstretched arms is performed to set up the joint locations.
- Markers are named based on their joint.
- Motion capture is performed, where an actor is asked to perform a certain action.

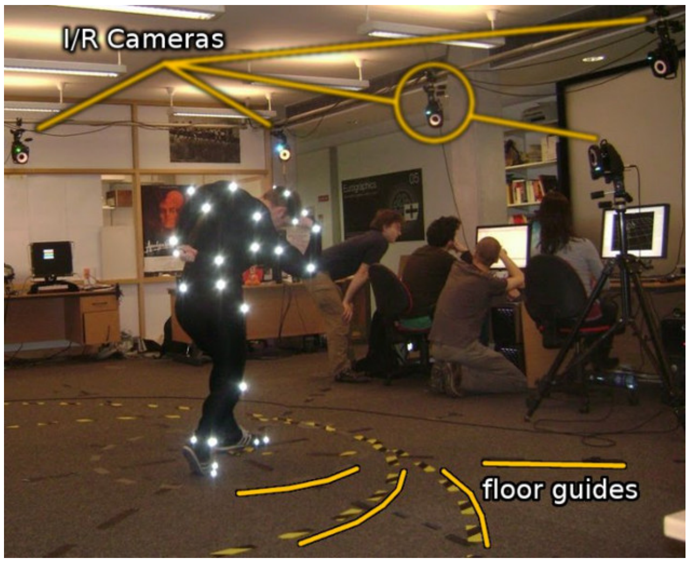


Fig. 2.2: Sample motion capture session in progress [Ger, 2014]

- Resulting files are exported to perform post processing. Occasionally during motion capture, the actor could obscure one of the markers and it will lose its information for a few frames. Animators look over the motions to ensure all information is properly applied.
- The final motion is applied to an avatar and rigging ensures that each bone controls a subset of the body.

Due to the extensive setup of motion capture sessions, only specific motions can be accurately recorded, hence it is sensible to be able to produce reusable animations.

2.3 BVH

2.3.1 Biovision Hierarchical Data

Biovision Hierarchical Data (BVH) is file format that stores skeleton setup and animation in a tree structure. It is separated into two parts. The first represents labelled joint hierarchy, their initial offsets and rotations. The second part stores the number of frames in the following motion, frame rate and per frame transitions and rotations for all the joints in the skeleton. A frame represents a single pose of a motion at any particular instance. Frame rate is the number of frames shown per second.

2.3.2 BVH Loader

The BVH parser I am using was written by a Ph.D. student from the Graphics, Vision, and Visualisation Group in Trinity College Dublin. The process is separated into 2 parts. The first part parses the skeleton joints, root offset and joint rotations. Identifying children and parent nodes, helps with the global rotation computation. The second part reads per frame root offset and rotation change for each node. All these values are preloaded into an animation class to be used during runtime.

2.4 Technology

For the purpose of implementation, C++ and OpenGL where used. OpenGL allows enough flexibility to be able to construct a 3D scene and control it real time, the BVH loader is also written using OpenGL and C++. BVH files where provided by the Graphics, Vision, and Visualisation Group in Trinity College Dublin. A set of stationary dance motions, walking, running and idle motions where provided. The dance motions where rotated by 90 and 180 degrees to evaluate performance of motion graph construction based on global offset.

Chapter 3

State of the Art

3.1 Previous Work

3.1.1 Initial Motion Synthesis

Before motion graphs [Kovar et al., 2002], several forms of motion synthesis allowed creating character animation only using constraints to generate physically valid motions based only on mathematical laws. [Witkin and Kass, 1988]. Later upon which was improved in [Rose et al., 1996], by applying the following idea along with inverse kinematics to introduce smooth transitions between two motion clips. Another method for transition between two clips at the time was to blend where they overlap, hence creating a warping animation that is different to its initial set [Witkin and Popovic, 1995].

3.1.2 Motion Graphs

The initial start to the field of motion graph was introduced using image blending to produce looping videos [Schödl et al., 2000], images are analysed based on a probability for transition and organised into a looping graph which provides information on most optimal looping videos that can be created. This inspired motion graphs in relation to creating looping animations, however methods to identify motion similarity varied. The method proposed by one of the original motion graph papers, involves comparing the distances of points between two motions over a window of frames. [Kovar et al., 2002] This method is called sum of square distances and is one of the main papers references throughout this dissertation. By identifying windows of frames that have high similarity between them, transition points are extracted to construct a motion graph. The sliding window ensures not pose by pose comparison, instead it focuses on overall motion change. A slightly different approach at the same time, with similar intention was to use pre computed velocity at each frame instead of a sliding window [Lee et al., 2002]. Both demonstrate methods for motion graph computation based on a small set of motion capture files. Similar methods for motion similarity can be seen used in [Arikan and Forsyth, 2002, Arikan et al., 2003, Safonova and Hodgins, 2007, Safonova et al., 2004, Lee and Lee, 2004, Heck and Gleicher, 2007, Lee and Lee, 2004]. A more simplistic approach to motion graph construction, focuses on splitting motion segments into groups based on similarity of left or right foot placement [Kwon and Shin, 2005], the main focus of this approach is to create a biped locomotion character. A recent study evaluates the following motion similarity methods based on user perception. [Krüger et al., 2011]. The study revealed that methods using joint positions rather than joint angles have greater results when computing error function.

The motion similarity method that was used for the following dissertation takes into account all of the above mentioned methods and proposes a new, more simplistic method for motion similarity. The idea is to find the distances between joint positions on a single pose and compare them to the same joint distances on a second pose. This method is called Joint Relative Distance method [Tang et al., 2008]. Instead of comparing the distance between the point clouds of two motions by creating an imaginary "overlap" on the Cartesian plane, this method focuses on identifying the joint differences within one pose before comparing it to the other. A method like this ensures global position and rotation of the overall pose can be ignored.

3.1.3 Motion Transitions

Transitions between motions are collected from a motion similarity function between two animation clips. In [Kovar et al., 2002] a local minima was identified from “sweet spots” between window of frames that contained high similarity between two motions. [Lee et al., 2002] computed a probability metric for transitions based on a set of joint weights that where later optimised [Wang and Bodenheimer, 2003]. A different approach for motion graph construction and transition calibration involves rhythmic-motion synthesis based on feet motion beats [Kim et al., 2003], or transitions based on foot placement to construct locomotion for a biped avatar [Choi et al., 2000, Park et al., 2002, Park et al., 2004]. A more manual approach to transitions between motions involves identifying transition frames based on a state machine like approach. [Mizuguchi and Calvert, 2001] This process involves large amount of animators contribution to ensuring feet constraints are labelled as well as perfect animation loops are prepared. A loop like approach to motion graphs can be seen in snap-together motions, where the most common pose is the centre of a motion graph and all paths from this pose loop back to it. [Gleicher et al., 2003] Such a method can be useful when designing virtual characters in games performing a repeated, simplistic task yet provides a sense of variety to it or give ability to control a virtual characters motion using interactive input [Shin and Oh, 2006]. This is then elaborated with several motion graphs intertwined.

3.1.4 Interpolation

Interpolation is motion generated to fill the gap between the end of one motion and the start of the next. It creates a smooth transition that should be barely noticeable for the user. Interpolation in motion graph transitions can be seen in [Ikemoto et al., 2007, Safonova and Hodgins, 2007, Sun and Metaxas, 2001]. Another method to transition between two motions is represented by blending. This creates an overlap between the start of the transition in one motion and the end of the continuing motion, plus several frames in between. Blending between motions can be seen in [Kovar et al., 2002, Mizuguchi and Calvert, 2001, Kwon and Shin, 2005]. A more advanced approach to blending between animations, computes a registration blend curve based on knowledge of the motion and their constraints [Kovar and Gleicher, 2003]. This is a more accurate form of blending instead of a linear frame blend. A study by [Wang and Bodenheimer, 2004] looked into the most efficient timing for blending during the transition, based on user evaluation. Quaternion spherical linear interpolation is another method to use with blending between poses, it allows for smooth transitions with a fixed velocity between a start and end of joint angles [Park et al., 2002, Kovar et al., 2002, Park et al., 2004].

3.1.5 Motion Synthesis

Motion synthesis represents a method of computing a path through a motion graph to construct a unique new motion. Once motion graph data has been pre computed, motion synthesis methods are proposed either real time based on user interaction controls such as keyboard, mouse or controller [Lee and Lee, 2004, Heck and Gleicher, 2007, Kwon and Shin, 2005, Park et al., 2002, Gleicher et al., 2003, Shin and Oh, 2006] or offline, using a predefined path [Kovar et al., 2002, Choi et al., 2000, Sun and Metaxas, 2001], joint position constraints [Arikan and Forsyth, 2002], annotations timeline [Arikan et al., 2003], start and end pose [Madhusudhanan, 2005].

3.1.6 Motion Constraints

Motion annotation helps to determine motion synthesis based on user desire, whether it is physical constraints [Fang and Pollard, 2003, Liu and Popović, 2002, Safonova et al., 2004] such as velocity, acceleration, force, momentum or a desired motion sequence [Arikan et al., 2003].

Chapter 4

Implementation

4.1 Introduction

Taking into consideration the state of the art, our aim is to create a different approach to motion graph construction using methods of joint relative difference computation as pose similarity. [Tang et al., 2008] However since JRD is a method to compute pose similarity, this does not include motion. A slight variation on the local minima computation will introduce the motion aspect to the pose comparison [Kovar et al., 2002]. To demonstrate the results of motion graph construction, a randomized simulation of the above solution is proposed.

4.2 Pose Similarity

One use of motion similarity method is to identify whether two motions are similar overall, however for the purpose of construction of motion graphs, the method involves identifying motion similarity function to find areas of low error to perform transitions between two motions. For motions with a large number of frames, such as a dance sequence that lasts over 30 seconds the resulting error functions will be quite large. For the purpose of this study, only a subset of motions will be used to demonstrate computation process.

Since a motion graph is constructed using a set of motions, a method is required to compare all available motions to find best transition points. An iterative per frame comparison methods, which compares two frames from two animations will result in an error value that represent motion similarity between them, these can then be represented visually in greyscale image. When searching such a function for transition points it is important that the motion similarity method takes into account following points to avoid any inconsistent transitions:

- Ignoring world coordinate position of the pose at specific frame ensures that identical poses where one is rotated by 90 degrees for example, are not missed.
- A single joint rotation parameter cannot fully express the effect it has on the final pose. An example such as a hip rotation which affects all children bones, will completely change the final pose, however if only a wrist is rotated, it will not have much of an effect on the final pose
- To identify smooth transition points between motions, velocity of joint rotations needs to be taken into account. If two motions, one that is stationary and the other in fast motion are identified as similar, this will result in a very unrealistic velocity transition.
- Limb positions such as left foot or right foot on ground are important to prevent unintentional sliding of motion.

4.2.1 Previous Work

In [Kovar et al., 2002], the proposed method for motion similarity computation is the sum of square distances between point clouds of two motions. To prevent problems described earlier, a sliding window of k size number of frames is computed using Gaussian weights over two motions to capture relevant information regarding all aspects described above instead of just the pose similarity. This takes into account motion velocity as well as the sum effect of individual joints on the final pose.

[Lee et al., 2002] present a different approach to compute motion similarity by incorporating both velocity and position into their sum of square distance computation:

$$D_{ij} = d(p_i, p_j) + dv(v_i, v_j) \tag{4.1}$$

Where the weighted differences of joint angles $d(pi, pj)$ and weighted difference of velocity $dv(vi, vj)$ are used to compute pose similarity D_{ij}. This distance function is used to compute a probability value of a transition between frames. The weights for positions and velocity are selected manually by determining which have more effect on the final pose. Weights where set for shoulders, elbows, hips, knees, pelvis and spine. The rest are set to zero as they have little impact on visible differences between poses.

In a study on human perception for motion similarity [Krüger et al., 2011], which made an initial assumption that computing joint angle differences between poses results in more accurate results rather than point cloud distances, later rejecting their claim based on gathered test data. The idea for using joint angles was that it would ignore the human weight, gender, height variables and only focus on motion. Since joints have different initial offsets from each other based on the unique human body structure, finding a method that can ignore these parameters and only focus on the motion is important for motion similarity. They gathered test results from human perception using both stick and point light figures as well as a distance measure computation to artificially compare motions. Methods used for the motion similarity study include: point cloud distances [Kovar et al., 2002], principal component analysis (PCA) [Safonova et al., 2004], quaternion and Euler angle based methods using 15,30,39 joints. These methods where applied to both motion capture animations as well as physically synthesized motions.

Results showed that higher dimensional feature sets showed greater correlation values between motions, than their lower dimensional counterparts. Motion similarity methods that use joint positions rather than rotations are advised to be used for best results.

4.2.2 Joint Relative Distance (JRD)

To incorporate all of the above considerations, the chosen method for motion similarity is described by [Tang et al., 2008]. Its main idea revolves around computing a relation between joint distances based on a precomputed weights analysis of joint contribution to final pose.

$$JRD_{p1,p2}(i,j) = |D(p_1^i, p_1^j) - D(p_2^i, p_2^j)| \tag{4.2}$$

Equation 4.2 represents the computation formula for finding the joint relative distance between 2 frames p_1 and p_2, where $D(x, y)$ represents Euclidean distance between two joints in a Cartesian plane. i and j represent joints within the given skeleton hierarchy, $(i, j = 0, 1, ..., n - 1)$ where n is 29 compared to the 25 used in the study [Tang et al., 2008]. The resulting value represents the error margin between the distances of two joints in two different poses.

Next step requires to measure these differences based on the weight of their relevance to the final pose. As described earlier in Section 4.2, different joints have different effect on the final pose. In this situations different joint pairs effect the final pose in different ways. Values between shoulder and hip for example do not have a large degree of change during motion. However, pairs like wrist and hip effect the final pose much more. Hence weights where computed based on joint relevance towards the final pose. Due to the format of weight computation, the weight values are computed for symmetrical joint pairs. Symmetric pair of joints are identified by an imaginary vertical line through the centre of the skeleton. An example of a symmetrical joint pair includes left wrist to hip and right wrist to hip. Top 30 values are demonstrated in Figure 4.3 [Tang et al., 2008].

$$f_{p_1,p_2}(i,j) = \begin{cases} \frac{1}{2}(JRD_{p_1,p_2}(i,j) + JRD_{p_1,p_2}(i',j')), & \text{if } (i,j) \text{ has a symmetric pair } (i',j') \\ JRD_{p_1,p_2}(i,j), & \text{(otherwise)} \end{cases} \tag{4.3}$$

A visual representation of symmetrical distances is demonstrated in 4.1. The yellow and orange lines represent the symmetrical pairs of joints between left wrist, left toe and right wrist, right toe.

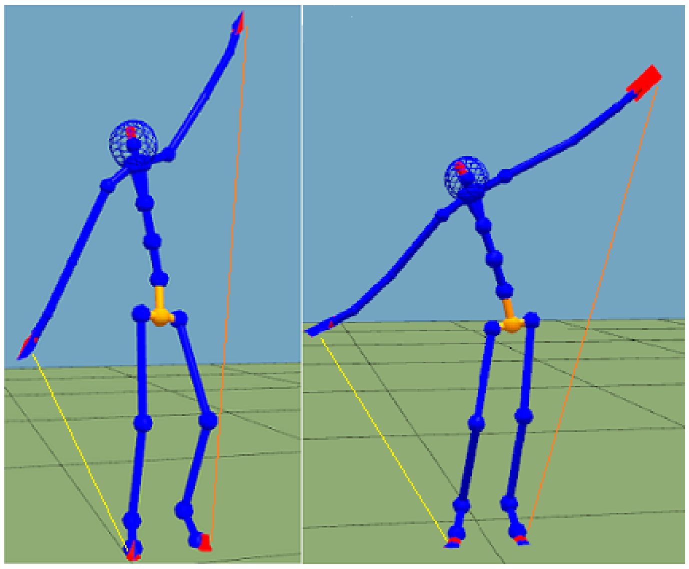


Fig. 4.1: Sample symmetrical joint pair comparison of two dance poses

Equation 4.3 demonstrates the method to incorporate symmetric pair joints into joint relative distance computation. We want the weights of symmetric pair of joints to be equal, hence the average between two symmetric pairs. The weights for joint pair i,j from Figure 4.3 is the applied to the result of $f_{p_1,p_2}(i,j)$ as demonstrated in Equation 4.4, where P represents skeleton hierarchy. The total sum value is determined by the number of chosen weights used in the error function computation.

$$y_{p_1,p_2} = \sum_{(i,j)\in P} w_{i,j} f_{p_1,p_2}(i,j) \tag{4.4}$$

In Equation 4.4, y_{p_1,p_2} represents the pose similarity error. Higher error meaning lower similarity between poses. 125 total weights where identified by JRD study [Tang et al., 2008]. Due to not all of them being available, an inquiry was made to the original authors asking for the complete table. However, with no reply, it was decided to use the first 30 until the a full table would be available. The table from the study is demonstrated in Figure 4.3 containing symmetrical pairs of joints along with their weights. Due to dissimilarities between skeletons used for BVH rigging as demonstrated in Figure 4.4, some of these values are ignored resulting in only 22 entries. This is enough to achieve relatively accurate results based on the weight distribution shown in Figure 4.2. To have more accuracy, a wider range of values should be used, however this could hinder computation.

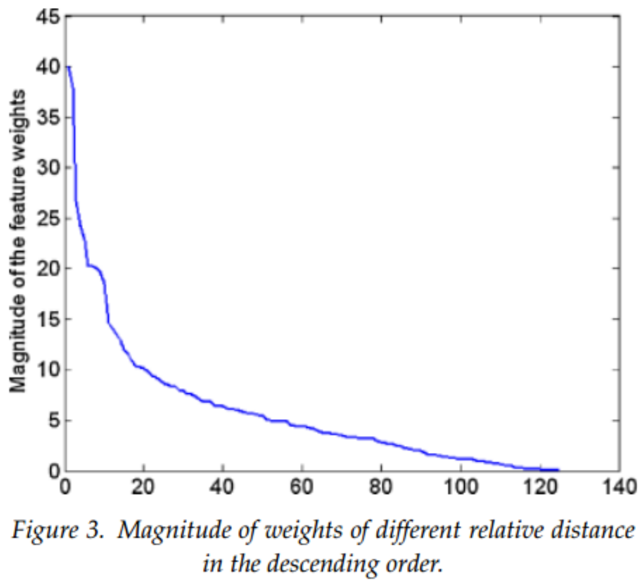
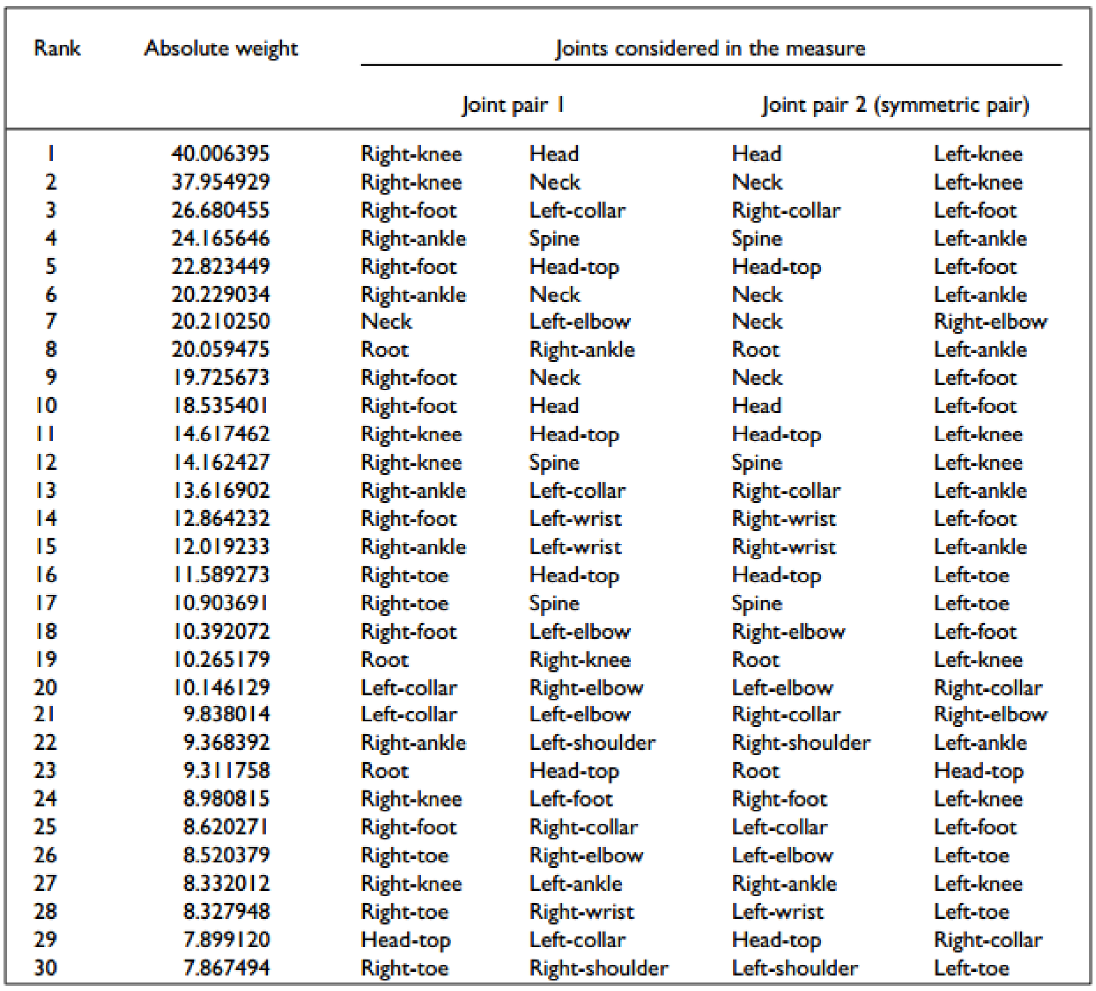


Fig. 4.2: Magnitude of weights of different relative distances between joints [Tang et al., 2008]



Rank	Absolute weight	Joints considered in the measure			
		Joint pair 1		Joint pair 2 (symmetric pair)	
1	40.006395	Right-knee	Head	Head	Left-knee
2	37.954929	Right-knee	Neck	Neck	Left-knee
3	26.680455	Right-foot	Left-collar	Right-collar	Left-foot
4	24.165646	Right-ankle	Spine	Spine	Left-ankle
5	22.823449	Right-foot	Head-top	Head-top	Left-foot
6	20.229034	Right-ankle	Neck	Neck	Left-ankle
7	20.210250	Neck	Left-elbow	Neck	Right-elbow
8	20.059475	Root	Right-ankle	Root	Left-ankle
9	19.725673	Right-foot	Neck	Neck	Left-foot
10	18.535401	Right-foot	Head	Head	Left-foot
11	14.617462	Right-knee	Head-top	Head-top	Left-knee
12	14.162427	Right-knee	Spine	Spine	Left-knee
13	13.616902	Right-ankle	Left-collar	Right-collar	Left-ankle
14	12.864232	Right-foot	Left-wrist	Right-wrist	Left-foot
15	12.019233	Right-ankle	Left-wrist	Right-wrist	Left-ankle
16	11.589273	Right-toe	Head-top	Head-top	Left-toe
17	10.903691	Right-toe	Spine	Spine	Left-toe
18	10.392072	Right-foot	Left-elbow	Right-elbow	Left-foot
19	10.265179	Root	Right-knee	Root	Left-knee
20	10.146129	Left-collar	Right-elbow	Left-elbow	Right-collar
21	9.838014	Left-collar	Left-elbow	Right-collar	Right-elbow
22	9.368392	Right-ankle	Left-shoulder	Right-shoulder	Left-ankle
23	9.311758	Root	Head-top	Root	Head-top
24	8.980815	Right-knee	Left-foot	Right-foot	Left-knee
25	8.620271	Right-foot	Right-collar	Left-collar	Left-foot
26	8.520379	Right-toe	Right-elbow	Left-elbow	Left-toe
27	8.332012	Right-knee	Left-ankle	Right-ankle	Left-knee
28	8.327948	Right-toe	Right-wrist	Left-wrist	Left-toe
29	7.899120	Head-top	Left-collar	Head-top	Right-collar
30	7.867494	Right-toe	Right-shoulder	Left-shoulder	Left-toe

Table 1. The features of top 30 weights

Fig. 4.3: Weights table [Tang et al., 2008]

4.3 Joint Weights

The following section gives an overview of the method performed in the Joint Relative Distance study [Tang et al., 2008] to train weights based on example. The weights of each combination of joints are computed using the tagged set of poses present from the user perception study of motion similarity. Two sets of poses where presented to the viewer and they had to tag them "similar" or "dissimilar", results in n pairs of tagged

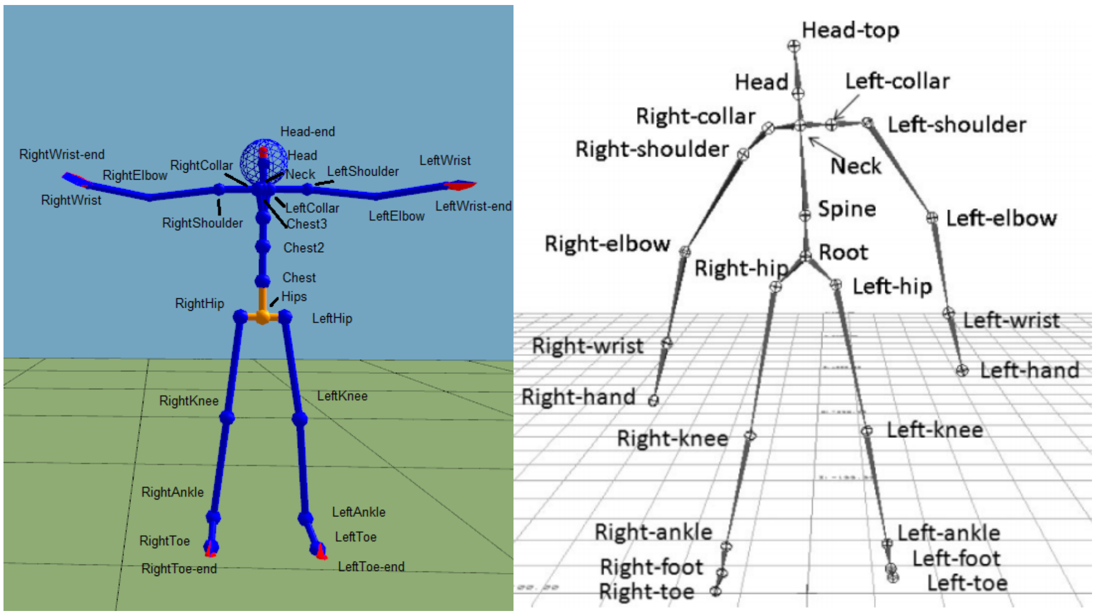


Fig. 4.4: Comparison of skeleton setup [Tang et al., 2008]

poses. The sum of joint relative distance from Equation 4.4 can also be written as $Y = WA$, where Y represents sum of error between two poses, W represents weights for all poses and A is the feature matrix computed using JRD method. The idea is to identify W value to be used for the computation of Y. Using the n number of pair poses described earlier, a constant can be set where $Y = 1$ for similar poses and $Y = -1$ for dissimilar. The previous equation can be hence rearanged into $W = YA^{+}$ where $A^{+} = A^{T}(AA^{T})^{-1}$ to compute the pseudo-inverse matrix of A to obtain the weight with the least square error. Using a number of these pairs of poses, a total average can be computed for each joint symmetrical pair. Results of this study are demonstrated using weight distribution in Figure 4.2 and the top 30 weights in Figure 4.3.

4.4 Error Function

An error function is a visual representation of motion similarity. It can help identify frames of high similarity between two motions. During the per frame motion similarity computation, the error values are collected and visually represented in a greyscale bitmap image. The error minimum and maximum values are recorded to rescale all error values to fit a 0 to 255 range to represent black and white pixels. Dark regions of the map will represent low error and white regions represent high error. Equation 4.5 shows rescaling computation, where the minimal and maximum value from a set are selected and applied to the old value. The reason rescaling the error function is required, is to create a clear distinction between low and high error values. Otherwise the error function bitmap looks grey and offers very little information regarding low and high error patches.

$$new = ((old - min) * 255)/(max - min) \tag{4.5}$$

Figure 4.5 demonstrates a sample error function for the first 500 by 500 frames of a dance motion. Motion one is represented as the width of the image starting from left to right and motion two is the height of the image from top to bottom. Every pixel in the bitmap image represents a pose by pose similarity comparison of two motions. Sample poses from 100 frame intervals including 0 are demonstrated along with the error function. Dark areas represent frames of high similarity. Upon evaluation it can be noted that clusters of low error frames represent stationary motions or motions which are moving with the same velocity and remain very similar for a duration. These types of clusters are required to be extracted as transitions.

Error function or distance grid has also been used in [Kovar et al., 2002, Kovar and Gleicher, 2004, Heck and Gleicher, 2007], for similar purpose.

4.5 Local Minima

Once a similarity graph between motions has been constructed, “sweet spots” or areas of high windows of similarity are extracted as motion graph transition points.

In [Kovar and Gleicher, 2004], instead of searching for local minima point, a 1D

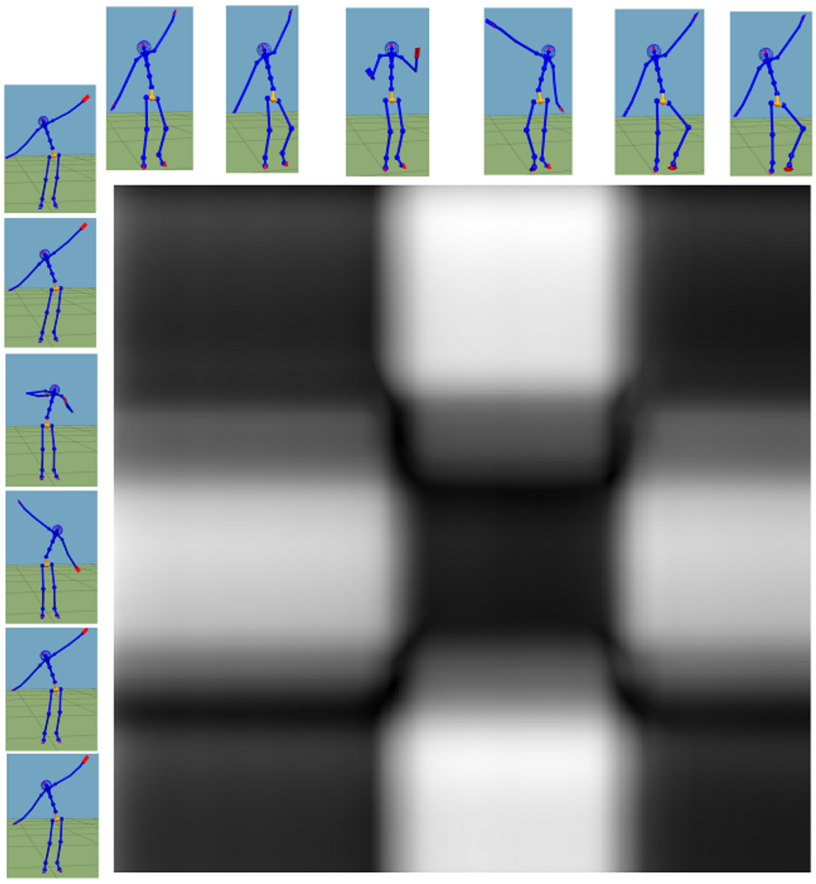


Fig. 4.5: 500 by 500 frame comparison of two dance motions

minima chain is computed instead. This chain represents a subset of two compared motions which have a high value of similarity, bridges are formed between sets of chains to generate new synthesized motion paths within the motion graph.

Parameter based transition points will store constraints about a given motion frame, in the example described by [Choi et al., 2000], motions are labelled by foot on ground position to allow motion synthesis based on a start and end location.

The method that has been implemented to perform a local minima computation for this research project involves a three step pruning process. During the per frame similarity computation, a 100 by 100 frame sectors local minima is recorded to be pruned later, a sample grid is seen in Figure 4.6. The smaller the sector size, the more local transitions can be recorded, however the size of a sector should not be smaller than the transition time value k, otherwise this will result in overlapping transition markers and excess computation time, which is unnecessary. Once all per sector local minimas have been computed, the three step pruning process is performed to optimise the resulting transitions:

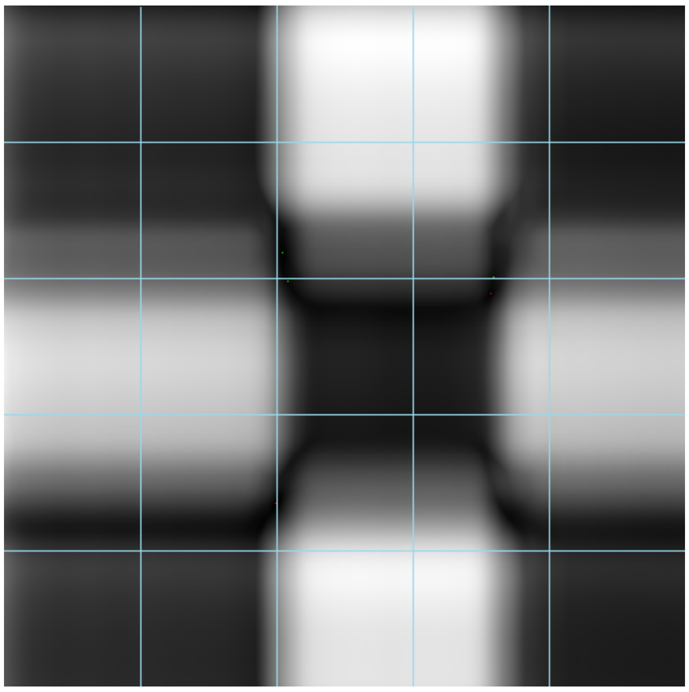


Fig. 4.6: Error function subdivided into sectors of 100 by 100

- The first pruning step involves checking this minima to be below a user specified threshold, this ensures that the level of motion similarity and transition points can be controlled by the user. The lower the threshold, the less points will be collected, but the quality of transition points will be higher. Figures 4.8 4.10 demonstrates values below threshold being distinguished from the rest in yellow.

- Second pruning process involves checking bordering local minimas that are too close and eliminating them. This check is necessary when local minimas are representing the same "sweet spot" within a close distance from each other, in such situations the bottom left most local minima is used and the rest are ignored. This is due to the transition method that is used, where transition frames used start from A_i to A_{i+k-1} from motion A and B_{j-k+1} to B_j from motion B, where i and j represent transition point. During this pruning step, duplicate "sweet spots" are removed and remaining go onto step three. To achieve this most optimally, local minimas are checked from top right to bottom left of the error function. This is explained more clearly in Section 4.6.

- In the last step, velocity of motion is checked. In [Kovar et al., 2002], the sliding window ensured velocity of a motion is compared before a local minima is computed. Since there is no velocity computation prior to the local minima, an average value around the local minima is computed to ensure the poses to be used for blending are of low error similarity, hence resulting in a smooth transition during stationary motions or motions of equal velocity.

4.5.1 Comparing Same Motions

To show effects of the local minima pruning process more clearly, Figure 4.7 demonstrates a sample 2000 by 2000 frame exact same dance motion similarity error function. The sections that where used to compute local minima are of size 100 by 100 frames. A clear black line can be seen going diagonally across the error function, this demonstrates a comparison of exact same frames hence why they will always return error values of 0. The image can be mirrored along the diagonal. Figure 4.8 then shows the same error function after the pruning. Areas in yellow represent local minima values that are below a user defined threshold, they have been collected and joined by a yellow coat. Red dots represent selected transition points that have met all the criteria, the number of points that was collected is 38. They have been enhanced visually to ensure clear comprehension. Most of the transition points that have been selected are located close to the diagonal. Contrary to belief, not all transition points are selected directly on top of the diagonal. This is due to the blending method used during transition as described in Section 4.6. Another thing to note, is during computation of an error function for two motions that are exactly the same except for one has been globally rotated by 90 degrees, results showed that the same error function has been created. Leading to a conclusion that JRD method ignored global position and rotations.

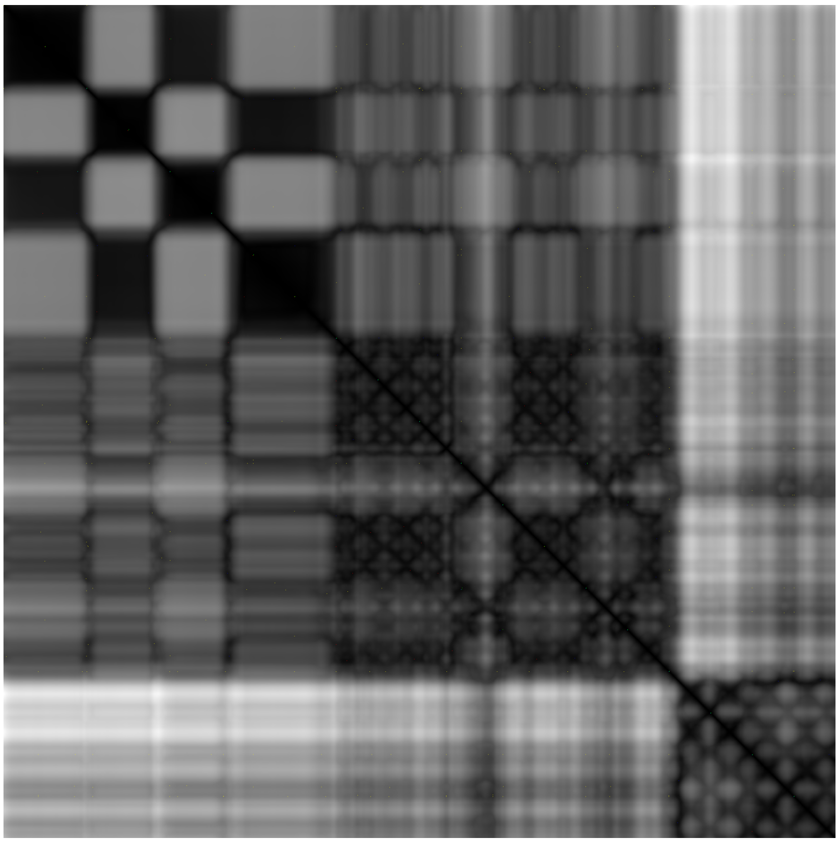


Fig. 4.7: Comparing 2 same motions

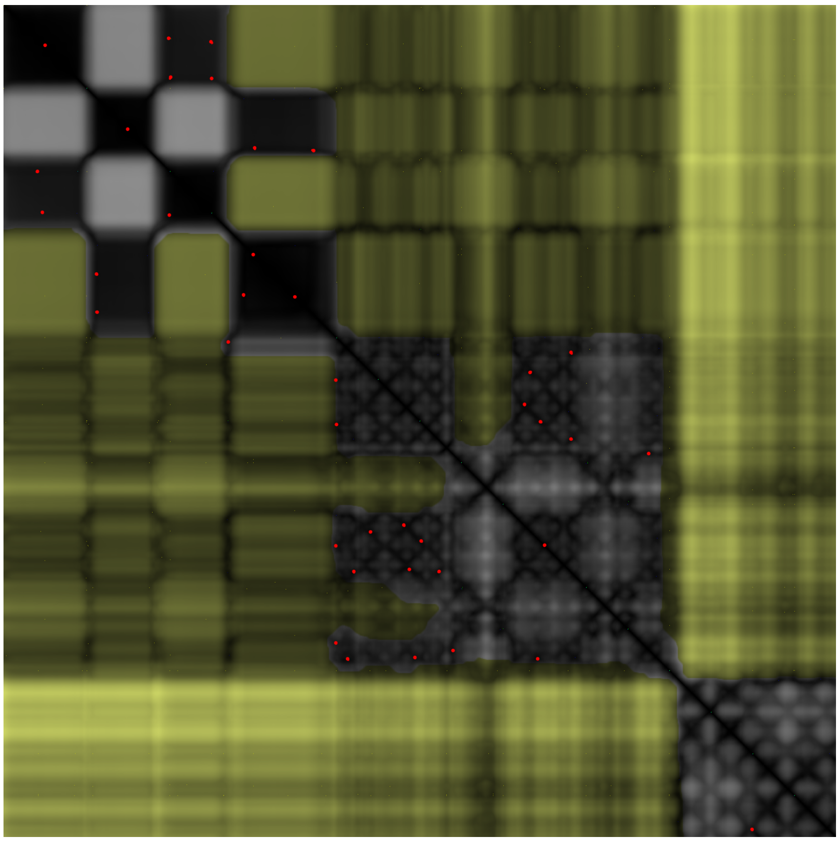


Fig. 4.8: Resulting 38 transition points after pruning

4.5.2 Comparing Different Motions

The following example shows the same error function computation, however instead of two same motions, two different ones where used. The actors where asked to follow an instructional dance video and attempt to replicate the dance. Due to human perception being a variant, the dances are slightly different, yet still following a similar pattern. Figure 4.9 shows the resulting error function, it no longer has a diagonal with a perfect error result of 0, instead a slight offset can be seen between the two motions specifically at the start when the dance is slightly stationary with a long period of holding a pose. A slight grey overlap can be seen between these, this represents two dancers changing the pose slightly off key. Figure 4.10 represents a similar pruning process, where yellow represents local minimas below a user specified threshold and red represents chosen transition points. In this case only 12 values have been returned.

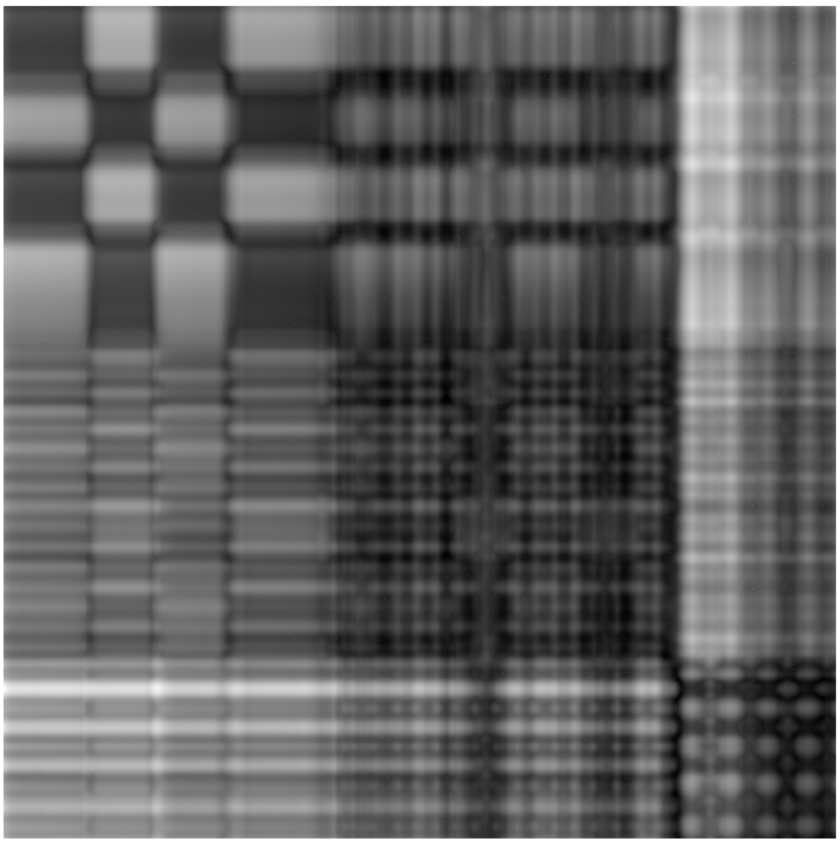


Fig. 4.9: Comparing 2 different motions

Once the pruning process is complete resulting values are recorded as most optimal transition points between two motions of any frame size. Similar approach was taken in [Schödl et al., 2000] et al in at attempt to generate video from a set of photographs based on their similarity metric.

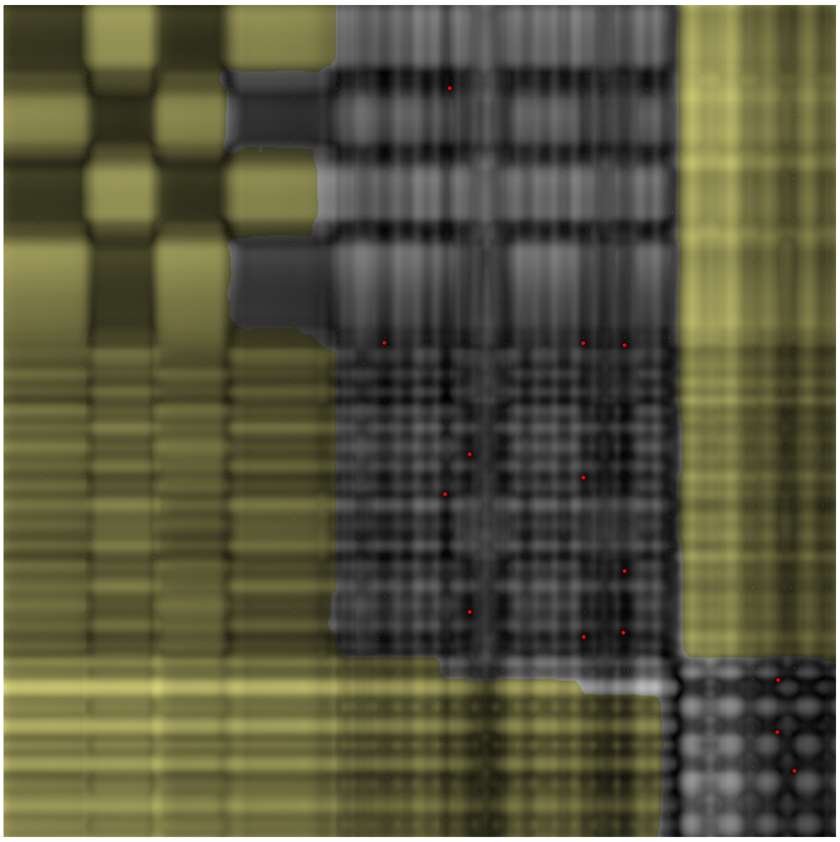


Fig. 4.10: Resulting 12 transition points after pruning

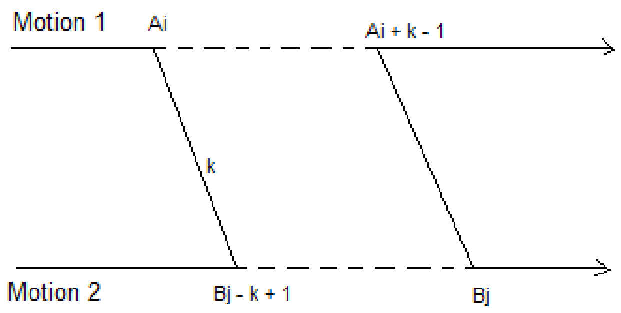


Fig. 4.11: Transition within error function

4.6 Creating Transition

Once frames are collected that have high probability of transition between 2 motions A_i and B_j, transitions are created by blending frames A_i to A_{i+k-1} and B_{j-k+1} to B_j inclusive, as demonstrated in Figure 4.11, where k is a user specified value for transition timer. It determines the number of frames available to perform full transition from A_i to B_j. $a(n)$ returns a transition value in the range of 0.0 to 1.0, where $a(p) < 0.0$ if $p < 0$ and $a(p) > 1.0$ if $p > k$.

$$a(p) = 2(\frac{p+1}{k})^3 - 3(\frac{p+1}{k})^2 \tag{4.6}$$

First step involves performing linear interpolation of the root node, based on the current $a(p)$ transition value, where $0 < p < k$. To ensure proper global root offset, $R_{A_{i+p}}$ and $R_{B_{j-k+1+p}}$, represent a frame difference between current and last displayed frames. Interpolation is computed between these offsets and is displayed in respect to total animation offset which is updated every frame.

$$R_p = a(p)R_{A_{i+p}} + [1 - a(p)]R_{B_{j-k+1+p}} \tag{4.7}$$

Next step focuses on computing spherical linear interpolation between joint angles of frame A_{i+p} and $B_{j-k+1+p}$ using quaternions. Resulting animation provides a smooth transition with a fixed velocity, hence why it is important that transitions happen between two motions with similar velocity, otherwise the transition will look unnatural.

$$q_p^i = slerp(q^i_{A_{i+p}}, q^i_{B_{j-k+1+p}}, a(p)) \tag{4.8}$$

Similar approach was taken in [Kovar et al., 2002] and Lee et al. Other transitions are available such as [Rose et al., 1996], where transitions are done using a mixture of spacetime constraints and inverse kinematics.

Constraints for foot positions ensures no sliding during motion, these constraints can be set when recording transition points and identifying foot placement, however due to the method of joint relative distance computation instead of the standard node difference computation, this is already ensured. When comparing two motions, one if the left foot on ground to a motion with right foot on ground, the resulting error will be high due to the joint differences, hence ensuring that joint positions are only slightly different. The joint difference weights have high values between joints such as feet and root as well as wrists and root.

4.7 Transition Logic

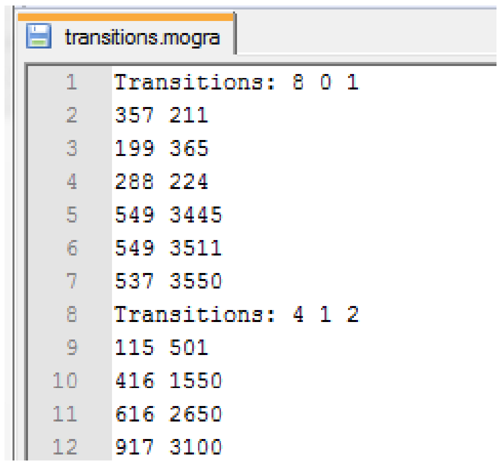
For demonstration purposes the transition logic involves 3 steps. To identify when a transition occurs, we decided to include user input command to trigger a random transition. When a call is made, a random transition is selected from the motion graph based on the closest available transition point to the current frame. When a transition is selected the 3 step transition begins.

- Before the transition frame Ai is reached, the displayed motion is treated as normal. Resulting value from $a(p)$ will always be below 0.0 meaning when spherical linear interpolation is performed, the full value of motion one is used.
- When the transition frame Ai is reached, a p increment will take place and a slerp will provide a transition. On the last transition frame, the animation fully switches to use motion B.
- Once transition is finished, motion is continued from Bj frame. Once the button is pressed again motion A becomes motion B and a new transition is looked up.

Synthesized motions are created randomly with user input. This can be further expanded, where transitions are picked based on a certain parameter, like a line by which a character has to walk, or a point which he has to reach. The development of this project started with the idea of creating infinite length dance motions from a set of dance clips, hence ideally creating a scene with randomized dancing characters.

4.8 Transition File

Figure 4.12 shows a sample transition file. To ensure saving computation time, a transition file can be precomputed to be loaded at any time. The format for the transition file can accommodate any number of motions. The file format is explained in a file sample 4.8.


transitions.mogra

```
Transitions: 8 0 1
357 211
199 365
288 224
549 3445
549 3511
537 3550
Transitions: 4 1 2
115 501
416 1550
616 2650
917 3100
```

Fig. 4.12: Transition file

```
MoGra.transitions:
 Transitions: N A B
X1 Y1
X2 Y2
N = Number of transitions
A = Motion 1
B = Motion 2
X1 - Xn = Ai
```

Y1 − Yn = Bj

The following file format is parsed into a 2D array where first element stores motion A. Every element in that array stores values B, A_i, B_j. When searching for closest transition to the current frame, using the current motion ID, that sub array is accessed and the closest transition frame A_i is located. An expansion on this could include a second check to ensure that the closest transition also has the smallest B_j, hence allowing for a longer motion B once the transition is completed, ensuring avoidance of dead ends. The loading of motions are interchangeable, when loading values into A sub array a mirror is also loaded into B where A_i and B_j are swapped positions.

4.9 Optimisation

Due to the large number of iterations required to be performed for the similarity computation, several forms of optimisations are proposed. Since the computation is done frame by frame comparison, palletisation is one form of optimisation. This is done using OpenMP, which is an API that supports multi-platform shared memory multiprocessing that is available for C++. Effect of parallel computing is demonstrated in 5.2.

The second form of optimisation is for the computation of motion similarity between two of the same motion. This is to ensure transitions between earlier or later points within the same motion. A sample error function for such a motion similarity can be seen in Figure 4.13. The diagonal creates a mirror image of the error function, hence only half as much computation is required for such situations. This is particularly useful for motions with large amount of frames.

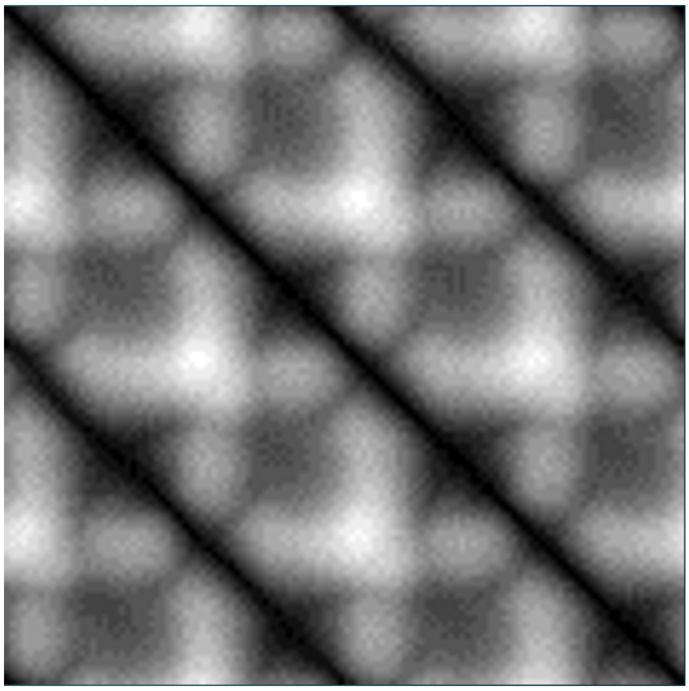


Fig. 4.13: Error function between the same walk motion

Chapter 5

Results

5.1 Experiment Setup

For the purpose of testing, three basic interactive applications where setup. The first two demonstrate spherical linear interpolation at a preprocessed transition point and the third demonstrates a sample application of the following research project.

5.1.1 Dancing Transitions

To demonstrate effectiveness of the transition location algorithm, an interactive character is placed in a scene starting from a dancing motion. With the help of user interaction by pressing a designated keyboard key, a transition is located based on the current frame and the closest best suited transition frame is returned. The transition logic is then assigned an A_i and B_j frames and until the current frame reaches A_i the application will display the motion as normal. Once reached A_i, the application will blend between the two motions based on a time sequence and apply spherical linear interpolation between joint angles of two motions. This is demonstrated by the transition of colours between green and red and vice versa. Once transition is finished, the motion will play out the new animation from frame B_j. The character can be set to transition between animations with a press of a button.

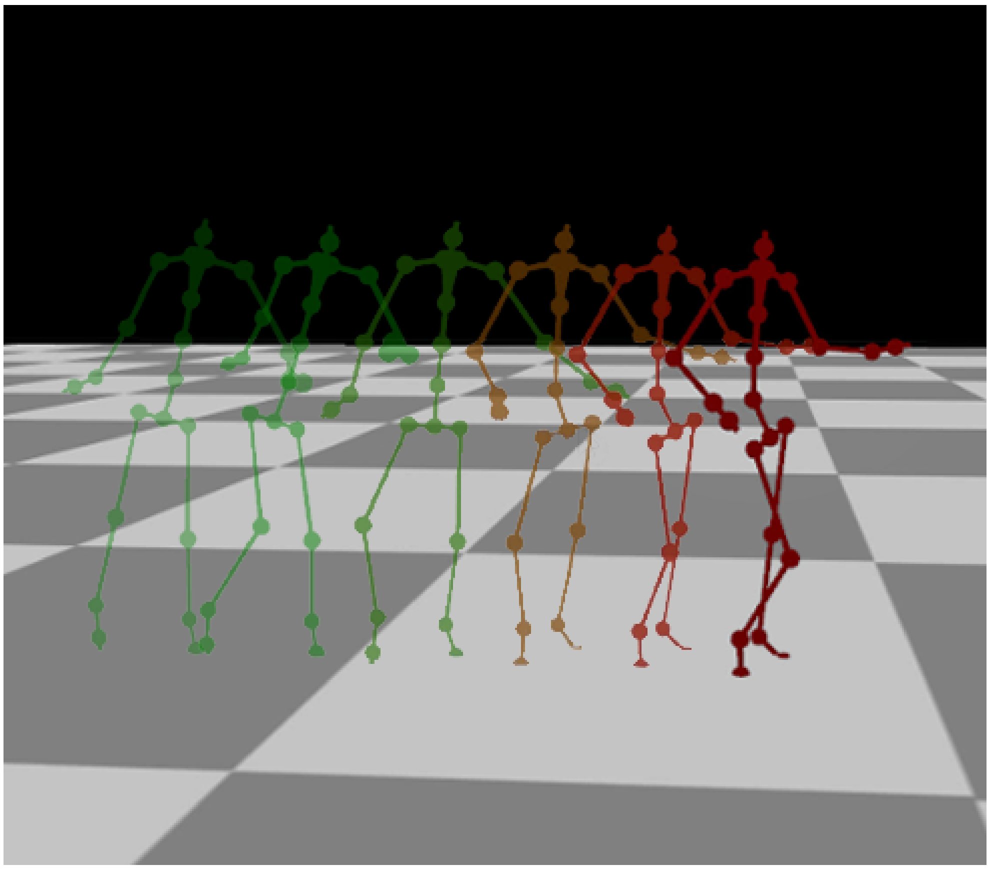


Fig. 5.1: Dance motion transition overtime

Figure 5.1 demonstrates a transition between two dance motions over a period of around 50 frames. The poses are offset slightly to demonstrates the difference in poses over time.

5.1.2 Bipedal Transitions

Demonstrates similar functionality as the first interactive application, however instead of purely dancing animations, the motion graph is constructed using walking and jogging motion clips. Similar interactive controls are applied with a moving camera that follows the virtual character. Sample application setup can be shown in Figure 5.2 The transitions do not work as smoothly as the dancing motion transitions due to very limited number of feet constraints beeing monitored. The character occasionally seen skipping a step with an unrealistic linear interpolation for the root position. Locomotion motion graphs that focus on feet transitions would work better in such a situation such as described in [Kwon and Shin, 2005].

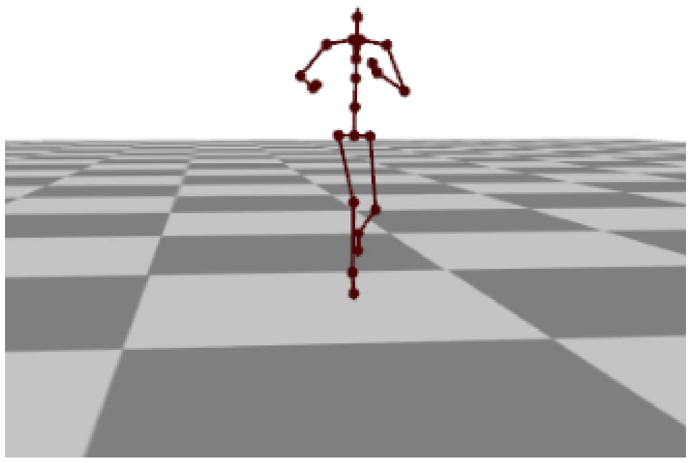


Fig. 5.2: Bipedal motion graph application

5.1.3 Dance Floor Simulation

A sample application that can be created using the following algorithm is a simple simulated dance floor with multiple randomly dispersed characters. Using only a set of dance motions the characters will perform unique, infinite long dance sequences that are randomly generated real time. This application requires no user interaction and will be randomly generated each time on load. This simple application can be further expanded to follow a more guided randomization of people placement and the actions performed based on the current situation. A preprocessing step to generate motion graph has to be performed during initial run, however after that the motion graph file can simply be loaded at any point for any desired application. Figure 5.1 demonstrates the application running.

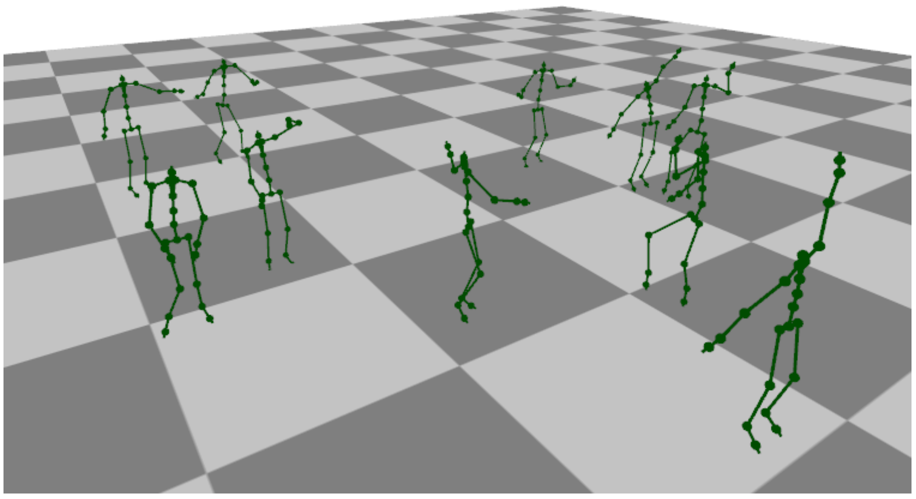


Fig. 5.3: Dance floor simulation with randomized dance motions

The randomization factor based on the C++ random library is used to set initial character positions as well as starting frame, starting animation and transitions calls. Since all the calls are random and not monitored for dead ends, the simulation is not perfect, however it does offer a proof of concept.

5.2 Computation Timings

Table introduces the computation timings recorded for motion graph construction. The system specifications where: 2.67GHz Intel Xeon Processor, 4GB of RAM , and a NVIDIA Quadro FX 580 graphics card, running on a 64-bit Windows 7 computer.

Table 5.1 on page 39 demonstrates computation timings for 2 scenarios, the first is a motion graph constructed using 3 dance motions of approximately 4,000 frames each, overall creating 24,000 iterations. The second scenario includes 3 walking and 3 running motions of approximately 100 frames each, hence resulting in 2,100 iterations. Each scenario was computed under 1, 10 and 20 number of parallel threads demonstrating computation efficiency. For both of these scenarios transition results are presented in Section 5.3.

Number of Threads	$\approx 24,000$ iterations (seconds)	$\approx 2,100$ iterations (seconds)
1	1906.41087	4.07130
10	512.43802	2.26303
20	498.30888	2.30014

Table 5.1: Computation timings for 2 scenarios using 1, 10, 20 threads

5.3 Transition Results

The following section demonstrate the number of transition points located for two scenarios described earlier. Along with the final number of transition points, values are also available after each pruning/condition step described in Section 4.5. For the purpose of the demo, the parameters that affected the results are as follows:

- The number of frames to make a transition (k) was set to 30, based on a study [Wang and Bodenheimer, 2003, Wang and Bodenheimer, 2004]
- The degree for error between two poses otherwise known as user threshold was set to 15.0f based on previous trial and error.
- The size of sectors for local minima computation was set to 50 by 50 frames.

Table 5.2 on page 41 shows results computed for motion graph construction of 6 walk/jog motions where values A and B represent the motion being compared. Not all motions have transitions available due to the very limited number of frames available for each. Most transitions are removed on the 3rd condition, which checks the average values around the local minima to be below the threshold. Due to the nature of the fast paced walk and jog motions, blending between them would naturally look akward, other methods would suit such a scenario better.

Table 5.3 on page 42 demonstrates similar results for 3 dance motions. Each dance motion contains on average 4,000 frames, hence the number of transitions is very large compared to the previous example. Motions which are compared to themselves have a higher number of transitions due to the same person performing the dance, rather than two different people. A possible improvement on this computation could include reducing the value for the error threshold. This will result in more accurate transition points.

A	B	Sectors	After 1st Condition	After 2nd Condition	After 3rd Condition (final)
1	1	9	9	9	0
1	2	9	9	8	0
1	3	6	4	4	0
1	4	9	0	0	0
1	5	6	6	6	0
1	6	9	6	5	0
2	2	9	9	9	0
2	3	6	0	0	0
2	4	9	9	8	3
2	5	6	5	5	0
2	6	9	9	9	2
3	3	4	4	4	0
3	4	6	6	6	0
3	5	4	4	4	0
3	6	6	6	5	2
4	4	9	9	9	0
4	5	6	6	5	2
4	6	9	9	8	2
5	5	4	4	4	0
5	6	6	6	6	0
6	6	9	9	9	0

Table 5.2: Number of transitions per motion similarity of 3 walk and 3 jog motions

A	B	Sectors	After 1st Condition	After 2nd Condition	After 3rd Condition (final)
1	1	6561	3620	2420	1512
1	2	6399	2182	1305	911
1	3	6399	863	476	362
2	2	6241	4881	3134	2592
2	3	6241	2537	1434	1304
3	3	6241	4413	2993	2445

Table 5.3: Number of transitions per motion similarity of 3 dance motions

Chapter 6

Conclusion

6.1 Contributions

The contributions made by the following dissertation includes an analysis of different methods of computation for motion graphs as well as a proposal for a new method. The method ignores global rotations and translations and has a modifiable accuracy measure by changing the number of weights that are taken into account during error function computation. The method was used for a motion graph construction of a set of dance motions and a set of walk and jog motions. Three applications where built around this and results where demonstrated based on computation timings as well as transition point numbers.

6.2 Conclusion

Based on the experimentation results, the following methods demonstrates a new way for motion graph construction with little supervision and relatively fast computation timings. However, best results where achieved for stationary simulations such as a dance motion. In simulation with a walking motion, several problems arose such as step skip upon transitions.

6.3 Future Work

To further improve the following method for motion graph construction, one of the most important efficiency related upgrades includes writing transitions to a database instead of a file. This is done in [Kovar et al., 2002] and many other research papers. Specifically for situations where motion files are 4000 frames long, a database can pre compute and store the following values, so loading would be much faster. From a database motion graph a path can be constructed real time or offline.

Since this dissertation was primarily focused on motion graph construction, the next step would involve computing an efficient path based either on terrain constraints [Arikan et al., 2003], user control [Lee and Lee, 2004, Heck and Gleicher, 2007, Kwon and Shin, 2005], predefined path [Kovar et al., 2002] or a start and end location [Madhusudhanan, 2005]. However, due to the nature of this method, bipedal motions require more work. For example marking feet position to ensure transitions avoid incorrect foot placement, hence skipping [Park et al., 2004]. The path construction will also ensure dead end avoidance in the motion graph.

Simple improvements to the local minima pruning method could include removing values on a diagonal between two of the same motion, during motion similarity. This will ensure unnecessary looping transitions are removed that effectively do nothing.

Bibliography

[CMU, 2014] (2014). Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/.

[Ger, 2014] (2014). Hardware skinning. http://www.antongerdelan.net/opengl/skinning_part_one.html.

[HDM, 2014] (2014). Motion capture database hdm05. http://www.mpi-inf.mpg.de/resources/HDM05/.

[Arikan and Forsyth, 2002] Arikan, O. and Forsyth, D. A. (2002). Interactive motion generation from examples. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 483–490, New York, NY, USA. ACM.

[Arikan et al., 2003] Arikan, O., Forsyth, D. A., and O'Brien, J. F. (2003). Motion synthesis from annotations. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 402–408, New York, NY, USA. ACM.

[Choi et al., 2000] Choi, M. G., Lee, J., and Shin, S. Y. (2000). A randomized approach to planning biped locomotion with prescribed motions.

[Fang and Pollard, 2003] Fang, A. C. and Pollard, N. S. (2003). Efficient synthesis of physically valid human motion. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 417–426, New York, NY, USA. ACM.

[Geijtenbeek et al., 2013] Geijtenbeek, T., van de Panne, M., and van der Stappen, A. F. (2013). Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6).

[Gleicher et al., 2003] Gleicher, M., Shin, H. J., Kovar, L., and Jepsen, A. (2003). Snap-together motion: assembling run-time animations. *ACM Trans. Graph.*, 22(3):702.

[Heck and Gleicher, 2007] Heck, R. and Gleicher, M. (2007). Parametric motion graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, pages 129–136, New York, NY, USA. ACM.

[Ikemoto et al., 2007] Ikemoto, L., Arikan, O., and Forsyth, D. (2007). Quick transitions with cached multi-way blends. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, pages 145–151, New York, NY, USA. ACM.

[Kim et al., 2003] Kim, T.-h., Park, S. I., and Shin, S. Y. (2003). Rhythmic-motion synthesis based on motion-beat analysis. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 392–401, New York, NY, USA. ACM.

[Kovar and Gleicher, 2003] Kovar, L. and Gleicher, M. (2003). Flexible automatic motion blending with registration curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 214–224, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Kovar and Gleicher, 2004] Kovar, L. and Gleicher, M. (2004). Automated extraction and parameterization of motions in large data sets. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 559–568, New York, NY, USA. ACM.

[Kovar et al., 2002] Kovar, L., Gleicher, M., and Pighin, F. (2002). Motion graphs. *ACM Trans. Graph.*, 21(3):473–482.

[Krüger et al., 2011] Krüger, B., Baumann, J., Abdallah, M., and 0004, A. W. (2011).

A study on perceptual similarity of human motions. In Bender, J., Erleben, K., and Galin, E., editors, *VRIPHYS*, pages 65–72. Eurographics Association.

[Kwon and Shin, 2005] Kwon, T. and Shin, S. Y. (2005). Motion modeling for on-line locomotion synthesis. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 29–38, New York, NY, USA. ACM.

[Lee et al., 2002] Lee, J., Chai, J., Reitsma, P. S. A., Hodgins, J. K., and Pollard, N. S. (2002). Interactive control of avatars animated with human motion data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 491–500, New York, NY, USA. ACM.

[Lee and Lee, 2004] Lee, J. and Lee, K. H. (2004). Precomputing avatar behavior from human motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 79–87, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Liu and Popović, 2002] Liu, C. K. and Popović, Z. (2002). Synthesis of complex dynamic character motion from simple animations. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 408–416, New York, NY, USA. ACM.

[Madhusudhanan, 2005] Madhusudhanan, S. (2005). Interactive human locomotion using motion graphs and mobility maps. Master thesis, Oregon State University.

[Mizuguchi and Calvert, 2001] Mizuguchi, B. and Calvert (2001). Data driven motion transitions for interactive games.

[Park et al., 2004] Park, S. I., Shin, H. J., Kim, T.-H., and Shin, S. Y. (2004). On-line motion blending for real-time locomotion generation. *Journal of Visualization and Computer Animation*, 15(3-4):125–138.

[Park et al., 2002] Park, S. I., Shin, H. J., and Shin, S. Y. (2002). On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '02, pages 105–111, New York, NY, USA. ACM.

[Rose et al., 1996] Rose, C., Guenter, B. K., Bodenheimer, B., and Cohen, M. F. (1996). Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH*, pages 147–154.

[Safonova and Hodgins, 2007] Safonova, A. and Hodgins, J. K. (2007). Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA. ACM.

[Safonova et al., 2004] Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 514–521, New York, NY, USA. ACM.

[Schödl et al., 2000] Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. (2000). Video textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 489–498, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Shin and Oh, 2006] Shin, H. J. and Oh, H. S. (2006). Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pages 291–298, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Sun and Metaxas, 2001] Sun, H. C. and Metaxas, D. N. (2001). Automating gait generation. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 261–270, New York, NY, USA. ACM.

[Tang et al., 2008] Tang, J. K. T., Leung, H., Komura, T., and Shum, H. P. H. (2008). Emulating human perception of motion similarity. *Journal of Visualization and Computer Animation*, 19(3-4):211–221.

[Wang and Bodenheimer, 2003] Wang, J. and Bodenheimer, B. (2003). An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 232–238, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Wang and Bodenheimer, 2004] Wang, J. and Bodenheimer, B. (2004). Computing the duration of motion transitions: An empirical approach. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 335–344, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Witkin and Kass, 1988] Witkin, A. and Kass, M. (1988). Spacetime constraints. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pages 159–168, New York, NY, USA. ACM.

[Witkin and Popovic, 1995] Witkin, A. and Popovic, Z. (1995). Motion warping. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 105–108, New York, NY, USA. ACM.