CSC 2529F Computer Animation
Graduate Project -Collaborative Motion Graphs

Alex & Philipp Hertel

April 15th, 2003

Introduction

There has recently been much interest in using motion graphs as a means of skeletal control in the context of computer animation. Motion graphs are favored for a number of reasons. For example, they:

- provide an intuitive means of planning and controlling a character's possible motions.
- are easily implemented.
- are robust and easy to work with.
- can be used as a second-order skeletal control, controlling a motion capture session.
- provide a mechanism for reusing motion capture data.

The reusability and control offered by motion graphs as a means of second order skeletal control can be taken one step further with the idea of collaborative motion graphs. Collaborative motion graphs are two or more individual motion graphs that interact, or collaborate, with one another. This interaction is governed by a (possibly empty) set of constraints.

Examples:

- Have two characters such as sword fighters, karate experts, dancers, etc., each with their own motion graph. Interaction between the two characters / motion graphs is regulated by constraints. This application of collaborative motion graphs Reduces an animators job to defining constraints.
- Have a single character whose skeleton has been segregated into separate motion graphs. The different parts of the skeleton interact based on constraints placed on the motion graphs.
- Have as scene containing multiple characters, each containing multiple motion graphs. This use of collaborative motion graphs has many applications, including automated animation of large crowds.

Project Overview

The purpose of this project is to explore the idea of collaborative motion graphs in the context of the second example above, namely the case where one character contains multiple motion graphs.

We were given a number of motion capture sessions featuring a dancer performing various dances, and chose to use a 687-frame tap dance session for this project. The high-level idea was to decompose the motion capture markers into two groups, upper body and lower body, each with its own motion graph. This allowed us to implement the idea of collaboration by defining some simple constraints which govern the interactions between the motion graphs, and consequently also the interactions between the upper and lower body of the tap dancer.

This project consists mainly of an application which demonstrates the use of collaborative motion graphs.

Using The Application

The application, CollaborativeMotionGraphProject.exe, is dialog-based. The construction of motion graphs follows a natural algorithmic process. We have implemented this process in six separate steps, corresponding with selection tabs in the main dialog box. Each time a step is finished and the user hits the 'Go To Next Step' button, the next tab is added to the main dialog box. A label at the top of the screen indicates what stage of collaborative motion graph construction the application is currently on.

Step 1, The 'Open Files' Tab (Figure 1)

This tab is entitled 'Open MoCap Session', and used to load a motion capture session into the program. The 'Open Files' tab is visible when the program starts.

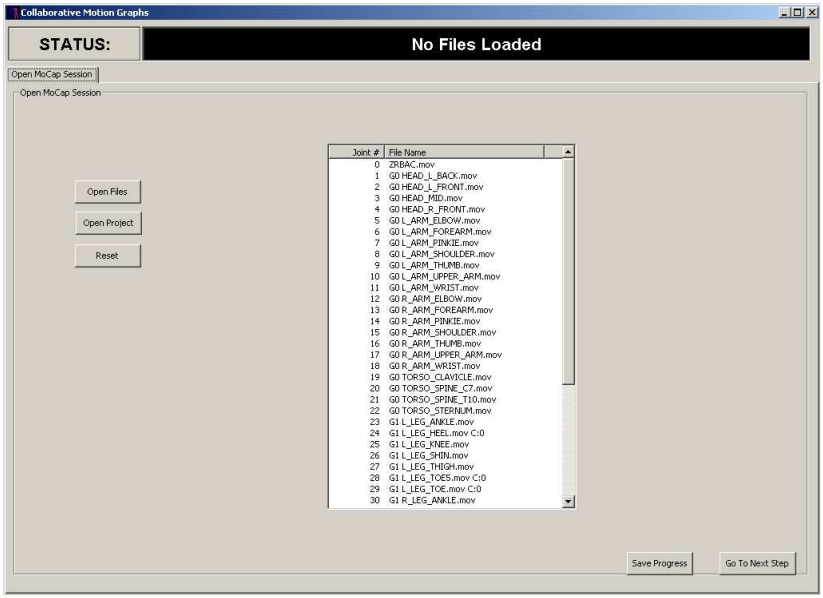

Figure 1: The 'Open Files' Tab

The following details describe how this tab is used to load a motion capture session:

- To load a binary (.cmg) file, simply hit the 'Open Project' button, and select the file to open.
- To load a motion capture session from individual (.mov) files, each representing a motion capture marker, hit the 'Open Files' button and select the files.
- Once a motion capture session has been loaded, press the 'Go To Next Step' button to create the next two tabs.

Step 2, The 'Simple Playback' Tab (Figures 2a and 2b)

This tab is used to view the motion capture session in its original form. If direction vectors, groups, and skeletal edges have been defined (see Step 3), they will be visible in this view. Figure 2a shows the tab with a raw motion capture session loaded, whereas Figure 2b shows the tab after Step 3 has been completed. Note that the green markers represent points of contact with the floor.

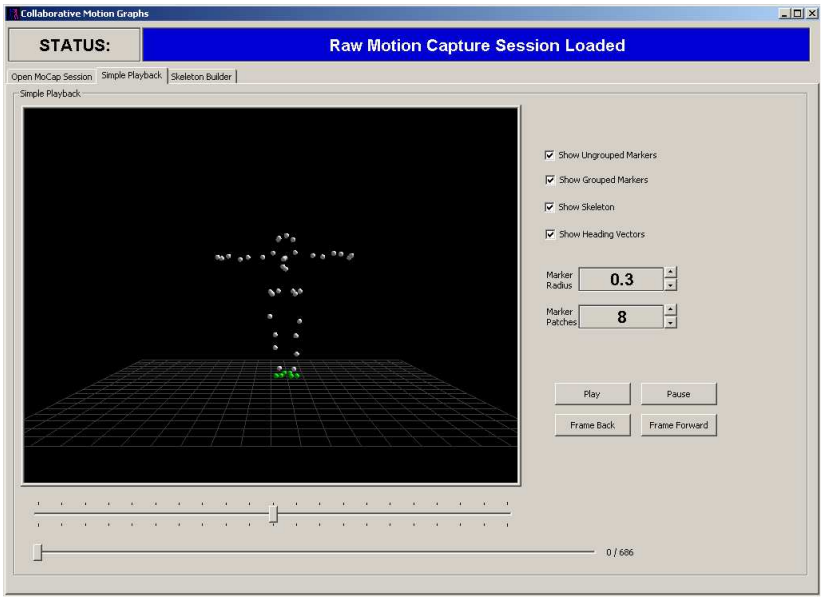

Figure 2a: The 'Simple Playback' Tab

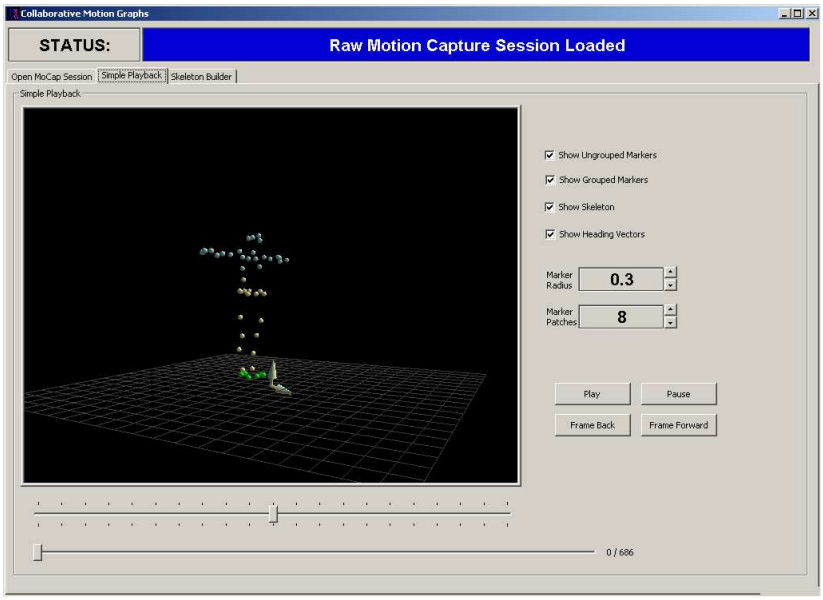

Figure 2b: The 'Simple Playback' Tab After Step 3 Has Been Completed

Most of this tab's user interface is self-explanatory. Below are some points of clarification:

- The motion capture markers are represented by spheres. The marker radius spin button changes their radius, and the marker patches spin button changes their detail level.
- The check boxes allow you to set the visibility of the different skeletal components. These check boxes are most useful when Step 3 has been completed.

Step 3, The 'Skeleton Builder' Tab (Figure 3)

This important tab is probably the most complicated one in the application. We have need of a more complex structure than simply a set of motion capture markers. This tab is used to augment the motion capture session by adding certain entities.

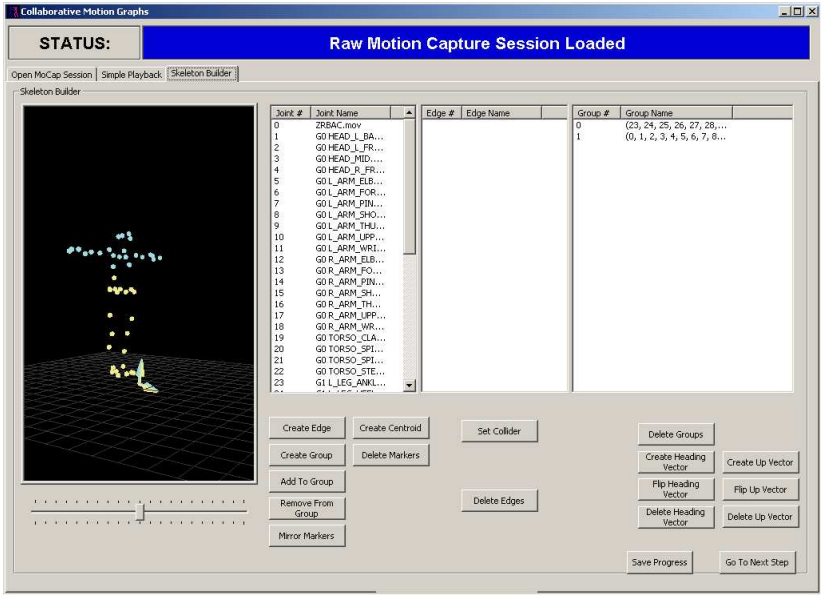

Figure 3: The 'Skeleton Builder' Tab

The entities that can be added by this tab are described below:

- Skeletal Edges - To create Skeletal Edges, ctrl-click two joint numbers, and hit the 'Create Edge' button. Edges may be deleted by selecting them and hitting the 'Delete Edges' button.

- Groups - Each group represents a separate motion graph. Defining disjoint groups is essential to the process of creating collaborative motion graphs, and at least two groups must be defined before continuing to the next step. To create a group, shift-click or ctrl-click the desired joint numbers, and hit the 'Create Group' button. To delete a group, select it and hit the 'Delete Groups' button. Individual markers may be added or removed from groups by selecting both the group and marker, and hitting the 'Add To Group' and 'Remove From Group' buttons, respectively.

- To create a new marker that is the centroid of a set of other markers, select the markers and hit the 'Create Centroid' button. Likewise, markers may be deleted by selecting them and hitting the 'Delete Markers' button.

- Collaborative motion graphs each need to know their heading. To create a heading vector, select a group and two markers, and hit the 'Create Heading Vector' button. Likewise, motion graphs each need to know which direction is up. Up vectors are likewise defined by selecting a group and the origin of a heading vector, as well as another marker, and hitting the 'Create Up Vector' button.

- Vectors may be flipped or deleted by selecting them and using the corresponding buttons.

- The 'Mirror Markers' button can be used to double the amount of motion capture data by taking the mirror image of the character. Clearly, this trick only works for characters with bilateral symmetry. For example, it wouldn't work for someone carrying an object in one hand. To mirror a marker on the plane of symmetry, select just that one marker, and hit the button. To mirror any

other markers (for example, both shoulders), instead first select both markers. This will extend them both. If mirroring is desired, it must be applied to all markers.

- Once at least two groups have been defined, and each has both a heading vector and an up vector, press the 'Go To Next Step' button to create the next tab. This will take a few moments, since the raw motion capture session is converted into frames of skeletons, which are necessary for the next step.

Step 4, The 'Animation Dissection' Tab (Figure 4)

This tab is used to dissect the motion captured animation into a motion graph. In this implementation, animation segments are contained in the nodes of the graphs. // The animation may be dissected manually or automatically, as described below.

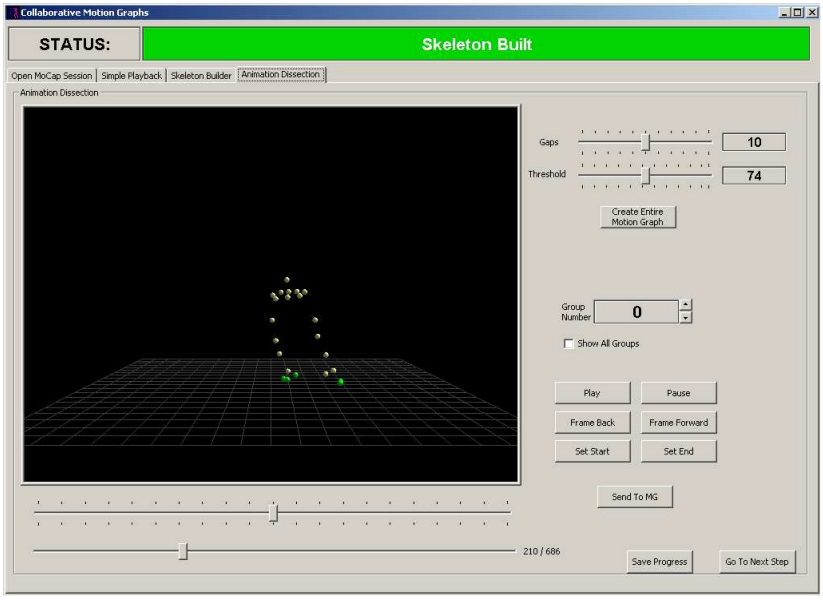

Figure 4: The 'Animation Dissection' Tab

- To manually dissect the motion capture session, select the desired group / motion graph by using the 'Group Number' spin button, move the animation slider to the start of the desired clip and hit the 'Set Start' button. Then move the slider to the end of the desired clip and hit the 'Set End' button. Next hit the 'Send To MG' button to send that set of frames to a new node in the motion graph. Repeat this process for every node you wish to create. This method is not recommended.

- To automatically dissect the motion capture session, select the desired group as above, set both the 'Gaps' and 'Threshold' parameters, and hit the 'Create Entire Motion Graph' button. The motion graph is then generated in the obvious way (see [1] or [2]) by computing skeletal difference metrics between all frames, and automatically creating edges between frames that have a difference metric below the user defined 'Threshold' parameter. In practice, a 'Threshold' value between 30 and 35 created interesting motion graphs, given the Tap Dance motion capture session.// The 'Gaps' parameter represents the minimum number of frames to skip when computing the difference metric. For example, without gaps, a frame i would be similar to frames i + 1, i + 2, i + 3, etc. We do not want to create edges from i to each of its near sequential frames. This would create a motion graph with a number of nodes that contain only one frame. It therefore makes sense to set a 'Gaps' value of at least 10 to make sure that the nearest frame is at least 10 frames away.

- Once each group has had a motion graph created, hit the 'Go To Next Step' button to activate the next tab.

Step 5, The 'Motion Graph Editor' Tab (Figure 5)

This tab is used to inspect the motion graphs and to manipulate them, if necessary. This tap is primarily used for re-arranging the positions of the motion graph nodes. Their positions are randomized during generation, which makes them hard to interpret. Planar graphs tend to be easier to follow. In addition, this tab may be used for viewing the animations within the nodes. Each group's motion graph will be displayed on the grid in that group's colour.

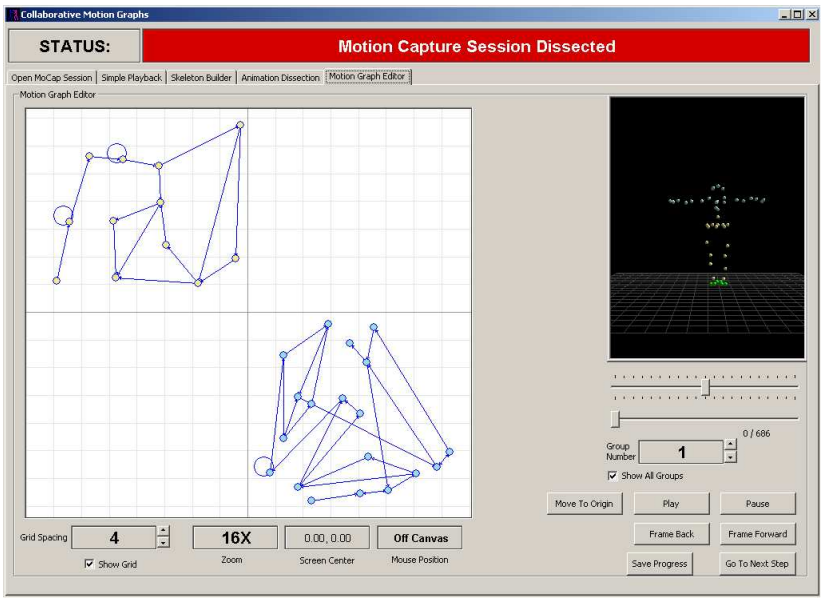

Figure 5: The 'Motion Graph Editor' Tab

The following details describe how this tab is used:

- To display a group's motion graph, select the group number using the 'Group Number' spin button. Note that only the motion graph which has its group number selected may be manipulated.
- To move a node, right click-drag it.
- To play the animation within a node, select it by right clicking on it, and hit the 'Play' button. By selecting an edge and hitting the 'Play' button, the animation at the tail of the edge will be played, followed by the animation at the head of the edge. This will allow you to view any discontinuities / artifacts. Note that no smoothing along the edges was implemented for this project.
- To center the screen on an arbitrary point, double-right-click anywhere on the canvas.
- If the character has been dancing for a while, she may drift off the screen. If this happens, simply hit the 'Move To Origin' button to re-center her.
- Hit the 'Go To Next Step' button to activate the final tab.

Step 6, The 'Collaborative Motion Graph Playback' Tab (Figure 6)

This tab is used to view the motion graphs as they interact / collaborate with each other. As the character dances, the respective nodes from each group that are playing are highlighted in the graph which is displayed on the left hand side of the screen. For this project, two different constraints were implemented for controlling the interactions between the graphs. They are described below.

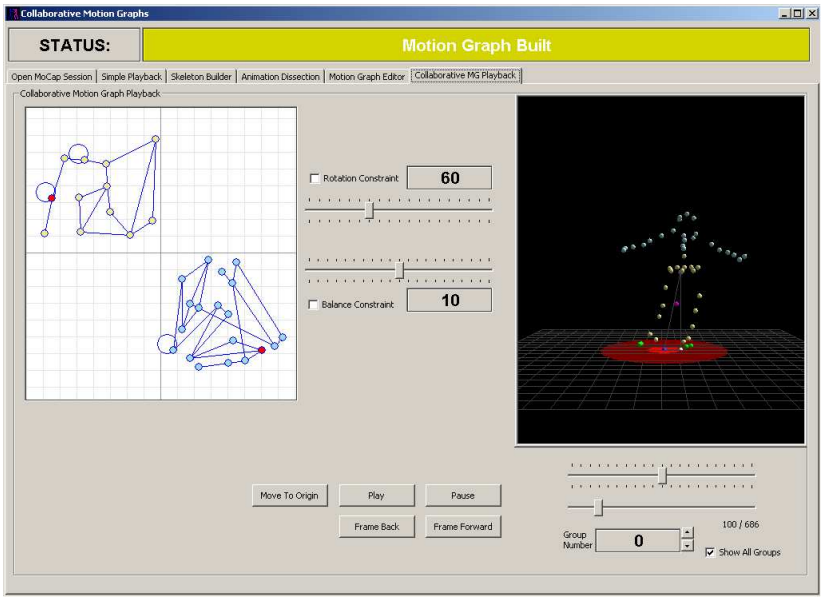

Figure 6: The 'Collaborative Motion Graph Playback' Tab

- If no constraints are selected, the motion graphs will randomly be traversed, and the animations corresponding to the nodes visited will be output to the screen.
- The upper slider and check box control the 'Rotation' constraint, which dictates how much the upper body may rotate with respect to the lower body. For example, if the constraint is set to 60 (degrees), then the angle between the heading vectors (after they have been projected into the X-Z plane) of the upper and lower bodies may not exceed 60 degrees.
- The second slider and check box control the 'Balance' constraint, which dictates how off-balance the character may be. The radius of the large burgundy circle below the character represents the constraint. The radius of the small bright red circle below the character represents how off-balance the character is. If their edges touch, the constraint is violated.

Design Decisions

The following were the most important design decisions made for this project:

- We chose to program our project in C++ rather than within the Maya framework because we are very familiar with the C++ language and believed that writing our own basic UI and our core code from scratch would probably be easier than learning a new scripting language and then trying to integrate our ideas into an existing system. We also wanted to be able to interface with the volumetric display, which is very straight-forward when using C++ and OpenGL.

Extensions

The following are some possible extensions to this project:

Superficial / Simple Extensions

- In our implementation we did not include the automated creation of transition frames in between two motion graph nodes. This has been researched extensively and is only made slightly more complicated in our case by the fact that the new frames must satisfy the collaborative constraints. This could be done quite trivially.

- A simple very superficial enhancement to our project would be a planar embedding algorithm for graph generation. It is somewhat important for the user to be able to follow the collaboration through the motion graphs. Very messy embeddings are much more difficult to follow than tidy planar or nearly planar embeddings. Again, this could very simply be accomplished.

- A fundamental change which could be quite easily made given proper facilities is the use of more complicated motion. Our system will work best when there is a relatively rich variety of motions in the graph. This will allow for a more connected graph, and therefore a more believable and sustainable animation.

Fundamental / Non-trivial Extensions

- Combining hierarchical motion graphs with collaborative motion graphs would be a very interesting extension to our work. A node could potentially contain a whole motion graph which dissects the motion and could potentially store different styles of the same movement. There could then be two levels of collaborative constraints. One set which selects the next motion, and then another more superficial set which chooses the style of the motion depending on what the collaborators are doing.

- The next extension is an interesting avenue of research, which leans towards behavioral control. This is something similar to a hierarchical motion graph, but has different criteria for placing graphs into the hierarchy. We will use an example to introduce the concept. A character could be subdivided into separate motion graphs as we have done in this project. Then at a higher level in the hierarchy he could be grouped with other characters and their grouping could be controlled by another collaborative motion graph, in which each group participant is represented by a motion graph and the collaboration is constrained by the activity which the group is performing. For example you could have a group of medieval soldiers battling, or two fencers duelling, or a pair of dancers, etc. Clearly, each type of activity requires different constraints to make it believable. Then at a higher level yet the entire scene can have a motion graph which controls all of the groups. For example, the entire battle could have a graph which dictates the strategy with which all the groupings of soldiers fight, or the dance floor can have a graph which ensures that no two couples collide by setting collision up as a collaborative constraint. This way, the idea of a second order

skeletal control is even stronger. Collaborative motion graphs seem to be a nice, natural way of organizing automated animation at many levels.

- Another potential extension is including a more complicated algorithm for choosing a path through the graphs.
- Also the generation of graphs from raw data could be made to incorporate the collaboration constraints so that the generated graphs are tailored to the collaboration.
- A very interesting extension would be to characterize a general collaboration constraint which could be the basis for a tool which the user could use to tailor collaboration constraints to his/her needs. Currently the constraints are hard coded into the program, and any change in the type of motion would require a recoding in order to integrate constraints for it.

Results

The project was quite successful. Besides the lack of transitions on the automatically-generated edges, which are easy to implement, our system created fairly realistic animation. With more data, we believe that the project will perform even more successfully.

It seems that motion capture data is so realistic in its movement that the dissection into upper and lower body creates a realistic-looking animation, even if played using random transitions. No doubt, a trained tap-dancer would be able to spot numerous unrealistic movements, but to the layman the movements, especially when restricted by constraints, do stand up to scrutiny.

Discussion and Conclusion

The most central topic regarding collaborative motion graphs is the characterization of the collaboration. It has two main aspects. The first is the type of collaborative constraints which are used to communicate between the collaborating graphs. The second is how one chooses a path through the graph, given these constraints.

The types of constraints are very dependent upon the motion which is being broken down into the collaborative motion graph. Clearly if a single body is being divided the types of constraints (volume preservation, limits on joint rotations, etc.) will be totally different than if a group of separate bodies is being coordinated by a collaborative motion graph.

In our implementation we created collaboration constraints specific to splitting the body into an upper half and a lower half. These include a rotational constraint on the spine and a balance constraint. It would be possible to create a generic skeleton which, when split into parts, could automatically apply rotational constraints, volume preservation constraints, and other constraints which are applicable to any motion applied to a subdivided skeleton. This would effectively create a general set of collaborative constraints for skeletal subdivision.

Far more complicated are the motion-specific collaborative constraints. A potential example for our project would be rhythm constraints which would clearly need to be specific to tap dancing. It is unclear whether a general set of such constraints could be devised which a user could modify by changing the values of certain variables. It seems as though each set will have to be hard coded. Though perhaps different classes of constraints could be generalized and applied as needed by the user.

The second important topic is the traversal of the motion graphs under the restrictions imposed by the collaborative constraints.

In our implementation, when a node's animation finishes we traverse to a random neighbor which violates no collaborative-constraints. This random choice could easily be weighted if certain combinations of motions are considered more probable within the context of an animation. Also greedy choices could be made to try to minimize the chances that collaborative-constraints will be violated in the future. This has its disadvantages though, because it could drastically decrease the chances of certain extremal

motions occurring. Often such motions add a great deal of character to an animation and their effective removal should be avoided.

When the collaborative motion graph is being traversed with collaborative-constraints enabled, it is possible that there may be no acceptable transition from a node which has finished its animation. If a transition to everyone of a node's neighbors leads to some collaborative-constraint being violated, this will occur. There are a number of contingencies available in this situation:

1. Turn down the collaborative-constraints and try again.
2. Ensure that the raw data is varied enough to make this improbable.
3. Incorporate the collaborative-constraint violation into the animation. Eg. Create a node for the character falling over and traverse to it when a balance constraint is violated.
4. Abort the animation and begin again.

The first option is interesting because it essentially softens the collaborative-constraints, allowing their violation if absolutely necessary. Depending on the nature of the constraints, though, this could lead to a unrealistic looking animation.

If a believable, continuous, smooth animation is desired, it would be best to begin with a large, sophisticated data sequence which would create a highly connected graph with many nodes. In this case problem nodes (nodes which are often involved when constraint violations occur) can be removed without seriously decreasing the variety of paths through the graph. Another option would be to add a new type of collaborative-constraint, namely a frame constraint. In other words, only allow a traversal to the problem node when the other graphs are in specific *frames* of specific nodes. The creation of such a constraint could easily be automated and would still allow the problem node to be expressed, albeit rarely.

In our implementation, we tried softening the constraints. Due to the details of our motion capture data this did not yield good results. We therefore just aborted the animation at "dead-ends" in our final implementation. We would like to try our implementation on more extensive sophisticated data to avoid such dead-ends.

In general, it seems desirable to have nodes which each comprise a recognizable human action, such as kicking or spinning, in contrast to nodes which contain a few frames of intermediary motion, or a fragment of an action. When the control of a single body is divided into a number of collaborative motion graphs the situation seems to be reversed. In real human motion, the beginnings and ends of actions seem to be coordinated. Separate but complete actions occurring together in one body can look very unnatural due to a lack of such temporal coordination. Stitching together a number of motion fragments in a highly constrained context seems to create a much more coordinated and therefore believable sequence.

References

[1] O. Arikan, D. Forsyth. Interactive Motion Generation From Examples. SIGGRAPH 2002.

[2] L. Kovar, M. Gleicher, F. Pighin. Motion Graphs. SIGGRAPH 2002.

[3] Y. Li, T. Wang, H. Shum. Motion Texture: A Two-Level Statistical Model For Character Motion Synthesis. SIGGRAPH 2002.

[4] K. Pullen, C. Bregler. Motion Capture Assisted Animation: Texturing and Synthesis. SIGGRAPH 2002.