James Vanderhyde
CS 7496 Computer Animation
Dr. Irfan Essa
29 April 2003

Retargeting Motion Capture Data Using Spacetime Constraints

Introduction

Motion capture data can be really handy for creating beautiful and believable animations. However, sometimes the motion that we want is not exactly what was recorded. A new recording session for the new motion is likely to be expensive or even impossible to obtain, and often the motion we want is quite similar to the motion data that we already have. Therefore, we do something called *retargeting*—changing the data we have to meet the constraints we want, but keeping the overall beauty of the original motion. For example, we have some motion captured of a person jumping. However, for some reason we want the character to jump a little higher. Or maybe we have a motion of a person squatting down, but we want the hand to be holding something fixed throughout the motion. Is there some way we can edit the data that we have to take care of these things?

The other goal of this project was to create a good implementation of rotations using quaternions. Motion capture data is usually specified in terms of a skeleton hierarchy where the angles of the joints vary over time. The data is stored as Euler angles, that is, the amount each joint is rotated around the z-, x-, and y-axes. However, as we know, Euler angles are not good for manipulation because of the interaction between the rotation axes and the dreaded problem of gimbal lock, the loss of a degree of freedom due to the middle Euler parameter equaling 90º. Quaternions, however, have no such problems. A quaternion q is a four-dimensional vector written $w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ or (w, x, y, z). It can be used as a rotation of θ around the axis (r, s, t) as $q = (\cos\theta, r\sin\theta, s\sin\theta, t\sin\theta)$. The rotation is applied to a vector v by using quaternion multiplication: $v' = qvq^{-1}$.

Related Work

Shoemake [1] gives a good overview of the use of quaternions in computer animation. Parent [2] also gives an overview, though he has some mistakes that causes a loss in confidence of his algorithms (the derivation of Δq on page 60 and page 451 is bogus). However, Shoemake says that to apply a quaternion rotation to a vector, we use $v' = q^{-1}vq$, but this does not end up working correctly. Perhaps it is correct under a slightly different interpretation of the quaternion rotation, but it is hard to say what that interpretation is. What we have given above is what Parent gives, and it works as expected.

Witkin and Kass [3] explain how they used spacetime constraints to generate believable animations. We did not use their work very much, but is the foundation on which other spacetime constraint methods for animation are based. For example, Gleicher [4] uses spacetime constraints for motion editing (what we call retargeting above), the topic of this project. In section 4, Gleicher gives a description of how motion editing can be viewed as a constrained optimization problem (minimize $g(\mathbf{x})$ subject to $\mathbf{f}(\mathbf{x}) = \mathbf{c}$). He explains his representation of the problem, the constraints, and the objective function. We explain this in more detail in what follows.

The motion data can be viewed as a function $\mathbf{m}(t, \mathbf{x}) = \mathbf{m}_0(t) + \mathbf{d}(t, \mathbf{x})$. You can think of $\mathbf{x}$ as a set of specified joint angles at several key frames, t as time, and the function $\mathbf{m}$ as the interpolation between key frames. $\mathbf{m}_0(t)$ is the original motion data, and $\mathbf{d}(t, \mathbf{x})$ is the motion displacement. To create the new motion, we just add $\mathbf{d}$ to the original motion.

Gleicher further describes the two types of constraints in this problem. The first type is kinematic. Every joint in the human body has some kind of constraint associated with it. Some, such as elbow and knee, are restricted to rotation about a single axis over a certain angle range.

Others, such as the shoulder, have more degrees of freedom but are still constrained. The other type of constraint is user-specified. For example, the user can specify handholds or foot-plants, or the desired height of a jump.

Finally, Gleicher describes the objective function. He states that for an edited motion to be good, it must resemble the desirable elements of the original motion. To that end, the objective he uses is the minimization of $\mathbf{d}$, the displacement function. The closer it is to 0, the closer the edited motion is to the original motion. This is a quadratic function because it measures Euclidean distance.

In the rest of the paper, Gleicher goes on to describe how to solve this quadratic system in interactive time. Rather than follow what he did, we used a general quadratic programming solver called CFSQP [5] because we were not concerned with user interaction in this project.

Approach

We take Gleicher's suggestion and use a displacement vector for the optimization. The elements of the vector are values representing the joint angles at specific frames. We specify a few simple kinematic constraints and user constraints. The minimization function is the size of the displacement vector. We pass all of these to CFSQP, and it iterates until a solution satisfying the constraints and minimizing the objective is found. This result is applied back to the animation. Since we only specify key frames for the displacement vector, the values that are applied to the entire motion are interpolated between these keys. We use spherical linear interpolation to do this. We did not implement cubic interpolation for quaternions, but this would have been a simple next step. Note that in Gleicher's formula the displacement vector is added to the original motion, but we are actually multiplying because we use quaternions.

The example motion we spent the most time on is a squatting motion. The idea is to constrain the left hand to a specific location so it is as if the character is holding on to a fixed object such as a railing during the process of the squat. As free parameters we use the shoulder, elbow, and wrist rotations at a varying number of frames. The elbow and wrist are each represented by a single angle, and a constraint is placed on each of these angles based on the natural motion of the human body. The shoulder is represented as a quaternion because it can move with several degrees of freedom. Thus the number of parameters is six times the number of key frames. We constrain the end effector of the hand to be within a certain distance of the specified point. To calculate the global location of the hand, we must traverse the skeleton, applying the rotations at each joint; therefore, this is a nonlinear constraint. Thus we have two linear constraints for the elbow (minimum and maximum possible angle), two linear constraints for the wrist, and one nonlinear constraint for the handhold. We do not constrain the shoulder under the assumption that the objective function will keep the shoulder movement close to the original.

Gleicher also discusses weighting the various components of the objective function, and he mentions a previous result that explains what to do. Rather than compute these weights, we use the same hand-coded weights we used in Project 2 part 2 to find matches between motions. Specifically, we multiplied the shoulder values by 0.2, the elbow by 0.1, and the wrist by 0.05 (the assumption being that the smaller motions are less important to the overall motion). In our experience in Project 2 the weighting of these joints did not make a great deal of difference.

Results

We applied a handhold constraint to a squatting motion. Please see the video for the original and edited motions. We show examples using four, five, and six key frames. One can see that the constraints are held more tightly with more keys, but the overall motion looks less believable. This reflects Gleicher's statement: "If the keys are spaced far apart, the displacement curve will be smoother and will add only low frequency (typically subtler) changes to the motion. Having distant keys also means have fewer keys, which can reduce the time for solving the optimization problems. On the other hand, with distant keys, it may be impossible to meet all the constraints." We also found that increasing the number of keys (for example, on the order of the number of frames in the motion) drastically increased the running time of the constraint solver we used, to the point that it might take days to complete, rather than under a minute.

In this data set, there is a range where the constraint solver was not able to find a set of values for the parameters that met the handhold condition. This range was frames 58–76. We are not certain why this area was a problem. Note that before frame 58 the character is standing up and reaches down to the handhold, and after frame 76 the character is squatting down and reaches up to the handhold. Somewhere the transition should be made to reach back and out, but that position is never found by the solver. We attempted to help the solver by constraining the shoulder rotation to its natural kinematic degrees of freedom, but we could not figure out a way to do this through quaternions. It is not sufficient to constrain each Euler angle because of the interaction among the axes, and it is also not sufficient to restrict the shoulder joint to one degree of freedom, due to the range of motion required to reach the handhold. The other difficulty is that clearly the character should lean somewhat in order to do some of this reaching, but if we allow hip rotation for leaning, this introduces another constraint of the feet remaining on the floor, added degrees of freedom for the solver in the knee angles, and so on. Rather than deal with this added complexity, we only allowed the arm to move.

The three output motions that we show have a varying number of keys. Each has one key at the beginning (frame 1) and one key at the end (frame 139). Besides this, the first (output4) has keys just before and just after the troublesome range mentioned above. The second (output5) has three keys spread out in the first half of the motion, before the trouble range. The third (output6) has two keys before and two keys after the trouble region. One can see that the extra frames do indeed keep the handhold constraint more closely, but the believability of the motion clearly suffers. Adding frames after the trouble region seems to affect the solver in such a way as to make the motion before the trouble region behave rather badly. Notice how in the earliest parts of the motion the wrist moves around arbitrarily. We did not completely isolate the reason this occurs. We believe that the objective of being close to the original is not helping in this instance, or perhaps the weighting is just not right. On the other hand, output6 does show the quaternion slerping quite nicely as the shoulder bends around during the trouble region. We believe output5 looks the most believable and natural because the handhold is consistent during the first part of the motion and the character reaches up nicely during the second part of the motion. Unfortunately, adding keys to the second part to make the handhold more consistent causes the arbitrary behavior we mentioned in the elbow and wrist.

Conclusions

This project is very interesting because it theoretically renders Project 2 part 1 obsolete. Instead of looking carefully at all the frames of a motion to determine which joint angles need to be changed, the user of the system can write just a few constraint functions, and the system will do the retargeting on its own. Before this is really practical for a larger number of free parameters, we would probably have to implement the solver that Gleicher uses, rather than the general nonlinear constraint solver that we used. Gleicher reduces the nonlinear optimization to solving a linear system, so this would be much faster, though of course it comes with the expense of imperfect results. Gleicher states that this is OK because the user can specify more constraints to clean up the imperfections.

A further interesting application that should be possible with this system is the following: The solver should be able to consider two skeletons together interacting somehow. For example, you could have two characters walking and holding hands. The only constraint (other than kinematic constraints) would be that the location of the right hand of one character is equal to (or within a small range of) the location of the left hand of the other character. This should create a very nice motion that is fundamentally different from the original motion. These two-character handholds are examined by Gleicher in another paper [6] (recall the swing dancers).

The final thing we learned was the importance of applying the Euler angles in the right order. This is obvious, but what was not obvious was how the results would look if the order is incorrect. We were stumped for a long time, getting strange results, until the error was finally pointed out. Applying the y-axis rotation first instead of last was a very easy change to make that fixed a lot of problems. What should have made it obvious that this was the problem was the way the z-axis rotation was messed up when applying an independent y-axis rotation.

References

[1] Ken Shoemake, "Animating Rotation with Quaternion Curves," *ACM SIGGRAPH* 19(3) (1985): 245–254.

[2] Rick Parent, *Computer Animation: Algorithms and Techniques* (San Francisco, CA: Morgan Kaufmann Publishers, 2002), 58–60, 97–102, 450–452.

[3] Andrew Witkin, Michael Kass, "Spacetime Constraints," *ACM SIGGRAPH* 22(4) (1988): 159–168.

[4] Michael Gleicher, "Motion editing with spacetime constraints," *Interactive 3D graphics* (1997): 139–148.

[5] `http://64.238.116.66/aemdesign/FSQPframe.htm`

[6] Michael Gleicher, "Retargeting motion to new characters," *ACM SIGGRAPH* (1998): 33–42.