CSCI 520, Assignment 2

Name: Hui Li

ID: 3194819503

Email Address: hli997@usc.edu

VS Version: Microsoft Visual Studio 2010 Professional Version 10.0.40219.1 SP1Rel

===============================================================================

<Description of what you have accomplished>

**0. Objectives:**

0.0 In this project, we implement three interpolation schemes to interpolate human motion data obtained from an optical mocap system, including Bezier interpolation for Euler angles, SLERP and Bezier interpolation for quaternion.

**1. Folder Index:**

IDE-starter: the visual studio solution folder for this project

IDE-starter\VS2010\interpolate: in this folder, we have the calculated interpolation results .amc files

graph: the graph folder for this project, including all excel files, graph files, data files and one .py file to manipulate the result data

video: the video folder for this project, it has three files corresponding to Bezier Euler, Bezier Quaternion and SLERP Quaternion animation .jpeg files

**2. Some Instructions:**

2.0 If you want to use the python program I write for this project to manipulate the result data, please use the command: python manipulate\_data.py [input\_filename] [output\_filename] [r/l]. It will output a .txt storing either root or lfemur joint data.

**3. What I have done:**

3.1 There are two additional functions to help finish the computation of interpolation, they are:

// convert degree to radian

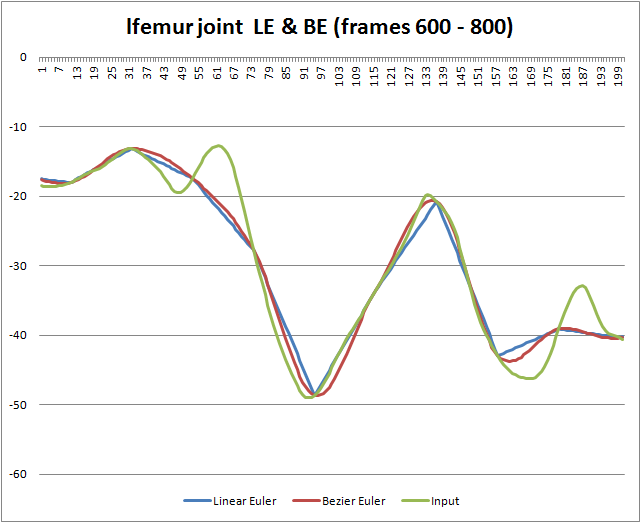
double degree2Radian(double degree);

// compute the matrix multiplication

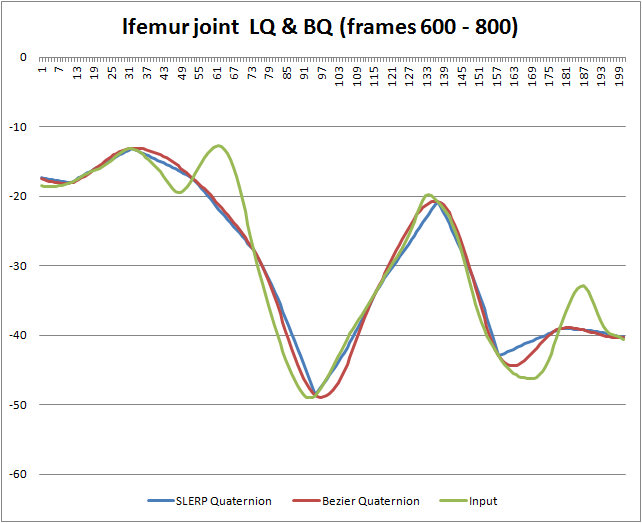
void matrixMul(const RotMatrix matrix\_top, const RotMatrix matrix\_new, RotMatrix matrix\_out);

**4. Result Graphs:**

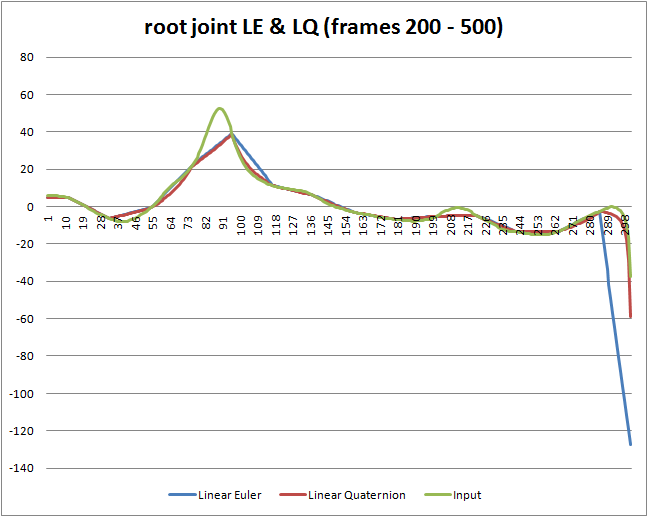
graph 1: compares linear Euler to Bezier Euler (and input)



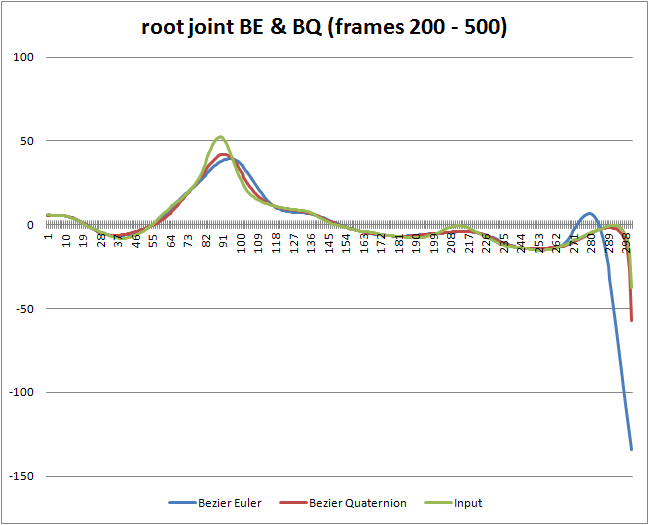
graph2: compares SLERP quaternion to Bezier quaternion (and input)



graph3: compares linear Euler to SLERP quaternion (and input)



graph4: compares Bezier Euler to Bezier SLERP quaternion (and input)



**5. Analysis:**

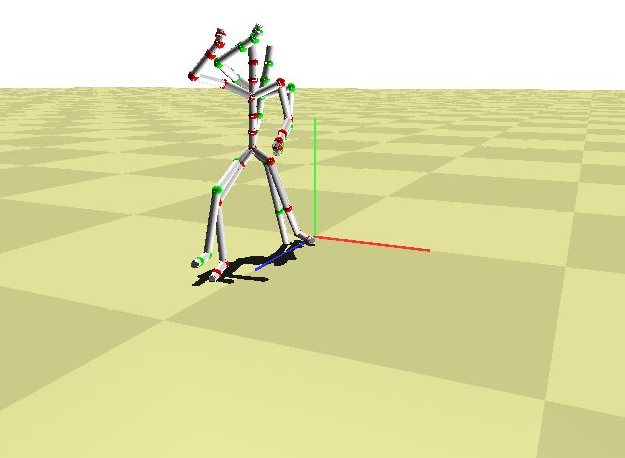
In this project, I implemented all three interpolation techniques, including Bezier Euler, Linear quaternion and Bezier quaternion, which are all work well.

* Linear Euler:

From the first and the third graph, we can see at each key frame the corner is very sharp, which means the motion changes dramatically during those frames. The computation time of this method for Martial Arts motion while N = 40 is 0.786096

* Bezier Euler:

From the first graph, we can see that this Bezier Euler method has a smoother curve. However, from the fourth graph, we can see the result of Bezier Euler has a big different from the input and Bezier quaternion, The root joint rotates early and dramatic declines after frame 480. When I watch the animation of martial arts, I notice sometimes the skeleton rotates to a strange angle, like:



I don’t know if this is the result of gimbal lock, but Bezier Euler method has a bad result based on the video. The computation time of this method for Martial Arts motion while N = 40 is 5.018578

* Linear quaternion:

Linear quaternion method works not bad in the interpolation. It has a similar problem with linear Euler, the corner of key frame is very sharp, but it has a more accurate changing trend of the curve. The computation time of this method for Martial Arts motion while N = 40 is 3.214151

* Bezier quaternion:

From graph 2 and graph 4, we can see Bezier quaternion has a great interpolation effect, and the motion is very fluent. However, the computation time of this method for Martial Arts motion while N = 40 is 10.745613. It takes the longest time in these four methods.

Based on the computation time, we can see that lerp takes the shortest time, it is also the simplest method. The Bezier quaternion spends the longest time, but it has a great effect.

**6. Observations:**

In order to implement these three techniques, I read the corresponding chapter on Rick Parent's book. In Bezier Euler and Bezier quaternion, I use the approach in the book to compute a1 and bN, the first and the last control point in the first and the last frame.

While I convert the Euler to the Rotation matrix, I use the rotation matrix multiplication. While the first time doing this calculation, I changed the order of multiplier and multiplicand. As a result, I always got the wrong answer. Finally, I found that in matrix multiplication, it has associative law but not commutative law.

The last issue is about the SLERP. First, we always pick the shortest path, so we need to consider cosTheta <0. Second, if the start point and the end point are the same point, we do not need to interpolate, we can just return this point. Third, if the angle is too small, we implement linear interpolation instead of SLERP.

<Also, explain any extra credit that you have implemented.>

**7. Extra Credit:**

For extra credit, I add a performance counter to interpolate.cpp file, which could calculate the computation time of different interpolation techniques.