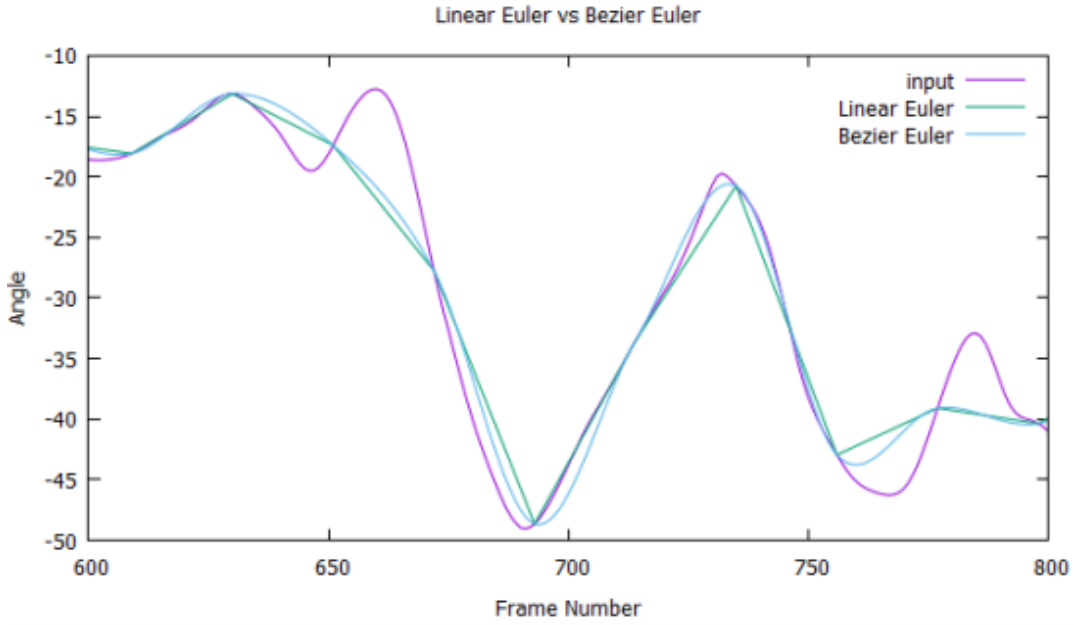
CSCI 520 Assignment 2: Motion Capture Interpolator

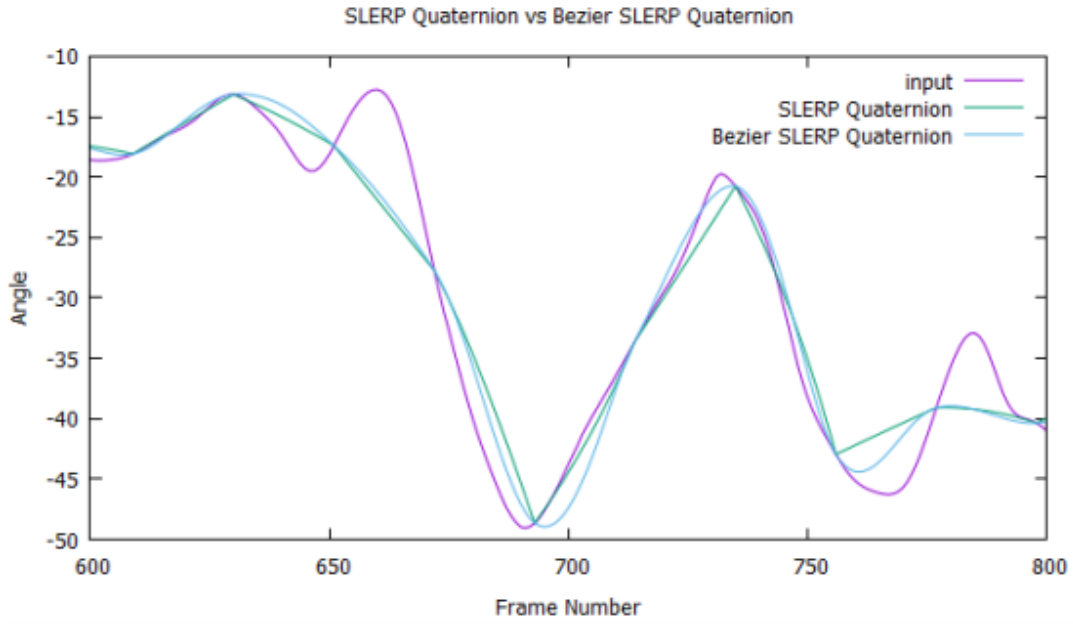
Richard Chien

VS Version: Microsoft Visual Studio 2017 10.0.17763.0

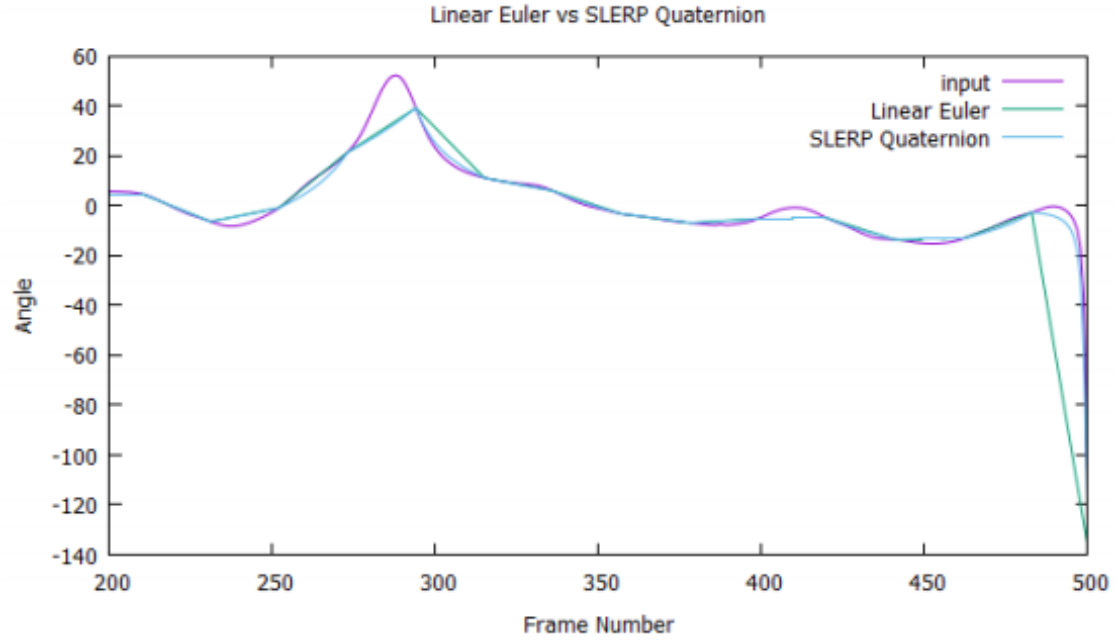
Graph 1



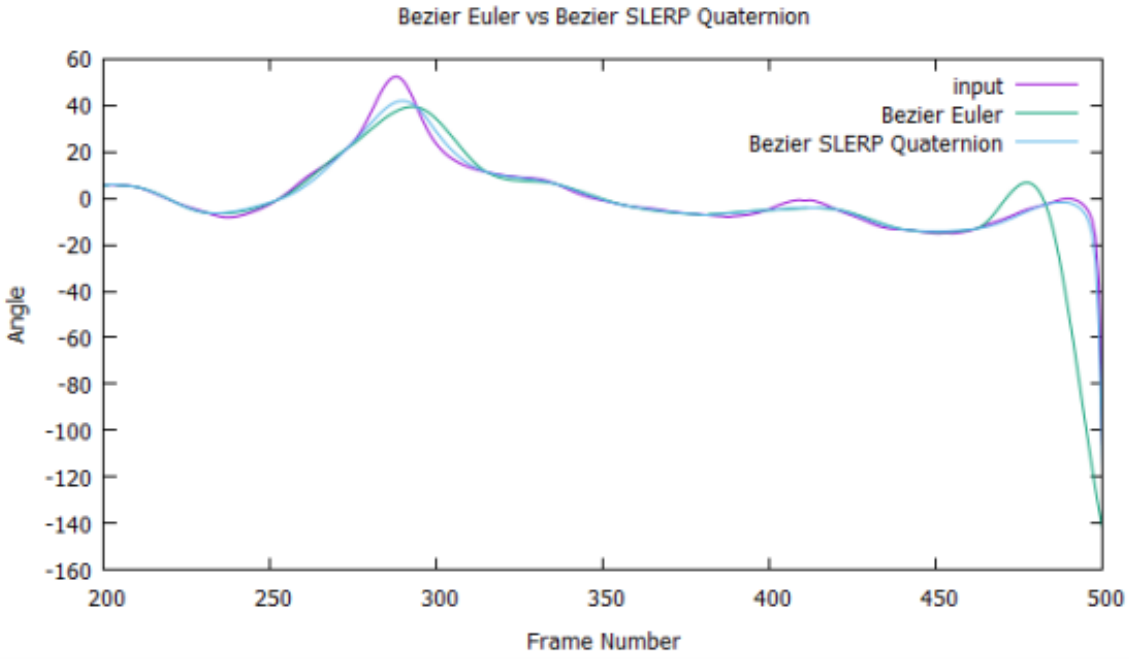
Graph 2



Graph 3



Graph 4



graphs 1, 2 lfemur joint, rotation around X axis, frames 600-800, for N=20, for 131_04-dance.amc

graphs 3, 4 root joint, rotation around Z axis, frames 200-500, for N=20, for 131_04-dance.amc

Analysis and Observation:

Firstly, the graph and curves differ. In graph 1, LE is simply a straight line from one key point to another, if the input motions are curve, it will lose many details in between. The BE seems better, it has some curves between key point, but it may interpolate in a wrong way (for example, if k_1 and k_2 are two key point, f''(k)*be''(k) < 0 where k is interpolating point between k_1 and k_2 and f(x) is interpolating curve). Most of the time, the BE is better than LE, but sometimes, it comes to the opposite way.

In graph 2, same as graph 1, LQ has a little change along curves, but still looks like a straight line. The BQ seems more smooth and curvy, but it may interpolate in a wrong way.

In graph 3, we can see that Quaternion deal better than Euler in big changes between key points, the curve of LQ stick to the input more likely than the LE.

In graph 4, both Euler and Quaternion are Interpolated with Bezier. The Euler now seems to have more artifact. The Euler can't deal with a big change of key point, especially shown as the curve near the end of frames (a peek and a dramatic downslope), I will talk about it more in the following words. In this case, BQ performs better.

Furthermore, lacking the information of the next key frame also makes the movements in linear interpolation methods look like having delays. The movements generated by Bezier methods look much smoother and more nature, but if the movement in input stops, you will see result movement sways a little bit before eventually stops. These can be seen from graph 1 and graph 2. Before the last 50 frames, the difference between Euler and SLERP is quite small, while you can see some tiny delays in graph 4 around frame 300, they are also identical in graph 3. In the last 50 frame, there's a rapid decline in the input, and the results of Euler and SLERP are very different. The results of SLERP follow the input curve quite well, especially with Bezier, while Euler's are not. Since the linear SLERP does follow the curve, the dramatic drop of Euler's cannot be caused by low sampling rate. What makes it even worse is that in Bezier Euler, not only it drops, but also it comes with a weird peak. This makes the result movement of Bezier Euler's result very unnatural. I assume the reason of this is because of the gimbal lock problem of Euler angle rotation, which makes the rotation discontinuous.

Extra Credit:

I plug in the performance counter to interpolate.cpp file, which calculate the computation time during Interpolation. For same input file and args, Linear Euler takes around 0.8 seconds, while Bezier Euler takes around 5.0 seconds. Linear Quaternion takes around 3.2 seconds and Bezier Quaternion takes around 10 seconds. There are even more than 10 times complexity of different interpolation techniques.