Comparing Body Positions

Programming Assignment #3
CS253 Spring 2014
Due 6:00pm Wednesday, February 17th, 2016

Motivation

In PA1, you read in sequences of body positions from a Microsoft Kinect II sensor; in PA2, you normalized the position and scale of those sequences. In PA3, you will measure distances between normalized poses, reporting on the magnitudes of motions between frames.

Pedagogically, the second assignment introduced persistent data, and encouraged you to define and create new object classes. In object oriented programming, classes are about abstraction and encapsulation. Every object class should be an abstraction of an easily explained concept so that you (or another programmer) can remember what it does. The class should then encapsulate all the data and functionality associated with that abstraction. The abstractions/objects you define are up to you, but candidates include points, poses, videos (or sequences of poses), input sources, and coordinate systems. You probably do not need all of these and you may have others, but at least some of these possibilities should have occurred to you while programming PA2.

In PA3, your objects will be put to the test. We are now introducing new functions, such as distance measures, and new control mechanisms, such as comparisons between frames.   If you have designed your objects well, this will mean introducing new methods and perhaps new data to existing objects, but your code will structure will remain similar. If you haven't designed your classes well, your code will turn into spaghetti. If this is the case, some redesign and re-coding of components of PA1 and PA2 may be in order.

Task

As in PA2, your program (now called PA3) takes two filenames as arguments. The first in the name of the input file, which is a sequence of poses in the same format as for PA1 and PA2. As in PA2, you should read in and normalize the sequence of poses. The second argument is still the name of an output file, but the output is no longer a sequence of poses. Instead, if the input video has N frames, the output file should contain N-1 floating point numbers, one number per line. The first number is the distance between the first frame and the second frame. The second number is the distance between the second frame and the third frame, and so on.

Distances between poses are defined in terms of distances between points. The distance between two points A and B is their Euclidean distance, defined as:

$$D(A,B) = \sqrt{(A.X() - B.X())^2 + (A.Y() - B.Y())^2 + (A.Z() - B.Z())^2}$$

where A.X() references the X coordinate of point A, and so on. The distance between two poses is then the sum of the distances between corresponding points. If P1 and P2 are successive poses and P[i] references the ith point in a pose, then the distance between two poses is:

$$D(P1, P2) = \sum_{i=1}^{25} D(P1[i], P2[i])$$

Submitting your program

You will submit your program through Canvas. You should submit a single tar file, and this file should contain all the source files of your program *and a makefile*, but no object or executable files. The makefile should create an executable program called PA3. To grade your programs, the GTAs will write a script that untars your file in a clean directory, runs 'make', and then tests PA3 on novel inputs. If your program does not compile, whether because of an error in your makefile or an error in your code, you will receive no credit on any of the test cases. Note that as always, we will test your code on the department Linux machines.

Hints

1. If there are errors in your PA2, fix them first. Inaccurate point positions will lead to inaccurate distance values.
2. You do not need to use the PoseDisplay class for this assignment. However, it may be handy for debugging.
3. If you have to scroll to see both the bottom and the top of any method, it is too big. Break it up into smaller methods.
4. As always, never trust a user or a file.