Assignment 5/6 Report
Liu Haixiang and Tomislav Pejsa

We have implemented an online motion retargeting system incorporating a hybrid inverse kinematics solver and continuous, proximity-based end-effector constraint inference. The system is based on techniques from the Computer Puppetry paper by Shin et al. [1] We applied our system to the problem of adapting existing motion capture motions to character figures with different proportions, demonstrating its effectiveness on a set of motions from the CMU motion capture database.

The system is implemented as an extension of the open-source animation system ZombieHorse, developed by Tomislav Pejsa. Motion rendering is accomplished using the open-source OGRE renderer, while the testbed application is build using wxWidgets framework.

1. Assignment 5 features
We implemented the following assignment 5 features:

1. Parsing and playing back motions in BVH format. Currently we support 3ds Max Biped-friendly BVH files. This is convenient because the entire CMU database has been converted to this format and made available for download: https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/3dsmax-friendly-release-of-cmu-motion-database. So far we have tested our system on 12 randomly chosen motions from the database, from 10 different capture sessions, and it was able to correctly load and display all of them.

2. Displaying motions. Our GUI application renders motions using OGRE. The skeleton is represented as a set of bone segments visualized as oriented, non-uniformly scaled boxes.

3. Scrubbing and playing at different speeds. Our program provides a slideable timeline which can be used for scrubbing, and the user can also modify playback speed. We support frame-locked playback and manual stepping through frames.

4. Camera control. Our program supports camera rotation, panning and zooming using the mouse. So far we have had no need to implement automatic camera repositioning and zooming based on motion extents, because all motions in the 3ds Max-friendly release of the CMU database have been edited to have the same scale, and to begin at the origin.

5. Loading multiple motions. Our program is currently set up to automatically load motions from a specific directory on startup. The loaded motions are then displayed in a list control, and can be selected for playback by double-clicking. Our current approach makes working with very large motion datasets impractical, since all motions are loaded at once; the solution is to load motions on selection (our resource management system already has support for this), and possibly have a background thread for loading resources.

6. Joint marker and traces. Our animation system can provide world positions for any joint. In our GUI program, we use this feature to render purple markers on joints. Similarly, we support joint "traces" – piecewise linear curves representing joint trajectories in the world space. The user can enable or disable markers and traces for any joint using appropriate checklists.

7. Splicing motions into sequences. Our program can play multiple motions as a sequence. When transitioning from one motion to the next, the target motion is rigidly transformed so that root position and orientation are continuous. Our animation system can also perform blend-based transitions, but we do not expose this functionality in the GUI program.

8. Quaternions. All rotations in our system are represented as quaternions.

9. Creating new motion clips and writing them out. Our program allows the user to select a segment of an existing motion clip and create a new motion clip out of it. This can make those long motions easier to handle. The program does not yet expose functionality for writing out motion files, but our animation system does have loaders and serializers for a custom XML file format, so adding this feature would be fairly trivial.

The interface of our testbed application is shown in Figure 1.
2. Project goals
Our goal was to build a system that can retarget motion to humanoid characters with different proportions (but same joint connectivity) from the original. The system is intended for online retargeting – it must be capable of operating at interactive frame rates, and inferring end-effector constraints on the fly. Moreover, rather than operating on each limb independently (like footskate cleanup methods), our system must be capable on adapting the entire character pose to simultaneously meet multiple conflicting constraints, necessitating use of a full-body IK solver.

In summary, the requirements for our retargeting system are as follows:
- Different figure proportions
- Dynamic constraint inference
- Full-body IK
- Interactive frame rate

3. System description
The system we implemented achieves our goals with differing degrees of success.

Our system supports humanoid figures with different proportions. Our testbed allows the user to edit figure features such as leg, arm, trunk and body scale. Examples of such differently proportioned figures can be seen in our demo videos.

Our constraint inference is based on the importance analysis method from the Computer Puppetry paper. In a preprocessing step, we manually annotate the environment with positions of “objects” that the character interacts with over the course of the motion. Constraints on end-effectors are assigned continuous weights computed from proximity of end-effectors (in the original motion) to environment objects. These weights specify how important a constraint is – higher weight means the motion will be adapted more to achieve the constraint. The intuition here is that objects close to end-effectors are the ones that the character is interacting with, so it is important that spatial relationships are preserved.
End-effector goals and weights are passed to the IK solver, which then solves for the adapted pose in such a way that constraints are met (to the degree specified by the weight values), while minimally changing joint orientations. We implemented a hybrid, full-body IK solver that solves for the final pose in three steps:

1. In the first step, an analytical approach is used to reposition the root so that all goals are reachable, or as reachable as possible, in proportion to their weight.
2. In the second step, a numerical solver is applied to the upper body (spine and collar bones) to adapt the body posture and move end-effectors closer to goals. The solver uses a gradient descent method and attempts to minimize goal distance and joint orientation change.
3. Finally, an analytical limb solver is applied to each limb; it poses the limb so that the end-effector constraint is met (assuming the goal is reachable).

The IK solver is based largely on the work of Shin et al. [1], Tolani et al. [2] and Lee et al. [3]

4. Results

We demonstrate our system on a set of motions from the CMU database, retargeted to differently proportioned figures. Results can be seen in our demo video.

Most of the system features work as advertised, but there are some outstanding issues. The posture solver in the Computer Puppetry paper is rather vaguely described, so we implemented a customized version using a simpler first-order continuous energy function, and employing gradient descent for optimization, which converges more slowly than conjugate gradient solvers. This is exacerbated by the fact that we numerically estimate gradient values, requiring us to take small steps to ensure convergence. As a result, the solver is currently rather slow, though able to sustain interactive frame rates at high-end machines. Moreover, there are some outstanding issues with motion smoothness – though our current formulation attempts to enforce smoothness implicitly (by using continuous goal weights and by minimizing difference from original motion), discontinuities still occasionally occur. Finally, posture modification yielded under the current formulation is usually very subtle, necessitating greater changes in the limb solve phase, which may be an issue if the animator wants to preserve the original limb poses more.

5. References

