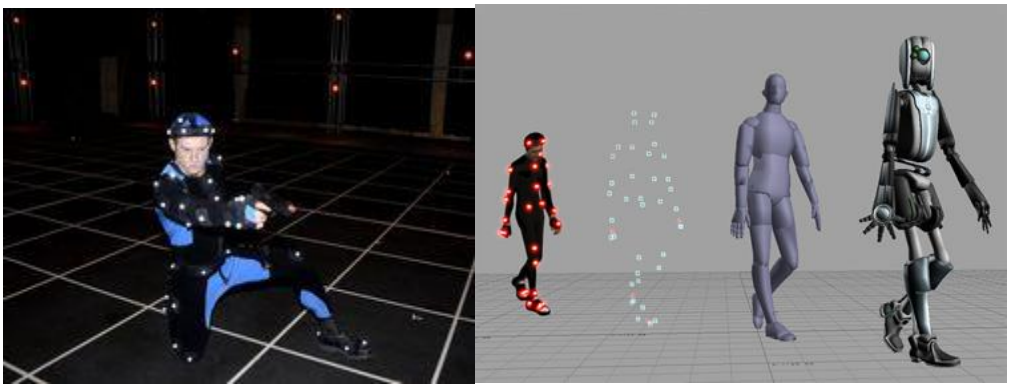
CS 560 – FALL 2017

Computer Puppetry

By:

Esteban Enrique Maldonado Cabán

Computer puppetry is a great technique that maps movements of a performer to a character that is animated in real-time. Motion capturing has been part of game development for decades now and it's a very efficient way of bringing realistic movement to their characters:



Studios provide suits with sensors that keep track of actor's movements who perform any type of motion. These could range from an attack, shooting a weapon, taking damage, a pitcher winding up the throw, etc.

It also has evolved so much that even movies are using this technique to capture more specific details like facial expressions. An example of this is Avatar:

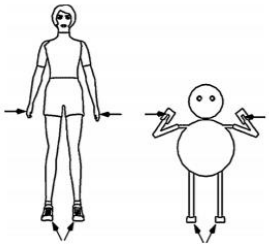
But regardless of which medium we use, computer puppetry has a lot of challenges. Motion capture comes with a lot of problems. The sensors attached to the motion the capture suits can provide a lot of extra data (noise). And sometimes the actors are given the task of animating a character that could be non-human, or simply different in some way.

An Importance-Based Approach

In a paper from the University of Wisconsin – Madison, a new approach for motion for online motion retargeting that transforms motions of a performer to a virtual character of a different size and proportion was presented. Introducing the notion of the importance of an end-effector, we have been able to generate realistic motion for a character in real-time while preserving the characteristics of captured motions as much as possible.

If the target character is quite different from the performer, there may not be a direct mapping. Indirect mappings are common in traditional puppetry, for example a marionette is controlled by strings that pull on its end-effectors.

When the virtual character and performer have different sizes and proportions, not all aspects of the motions can be preserved during mapping. At the lowest level, it is simply not possible to mimic both the locations of the end effectors and the joint angles. A system must make choices as to which aspects of the motion should be preserved and which should be allowed to change. We call an approach to motion retargeting that makes this choice explicitly an **importance-based approach**.



Artifacts of position-based approach.

In this article, they speak of an Inverse Kinematics solver that maps human actors to their somewhat odd-shaped character's anatomy:

"…These can fall in under 2 categories: analytic and numerical solvers. For computer puppetry, we make a number of demands on IK that require the development of a novel solver. First, we must achieve real-time performance on the entire body of the character. Second, we need the solver to provide predictably consistent solutions: small changes to the problems should provide similar answers. Finally, the solver must be able to account for the importance of the end-effectors that are determined dynamically in our system.

To solve an IK problem in real-time, we divide it into three sub-problems: root position estimation, body posture computation, and limb posture computation. The root position of a virtual

character is computed to provide a good initial guess for the body posture computation. If needed, we then adopt numerical optimization to refine the body posture, which consists of the root position, the root orientation, and the posture of the upper body. Finally, we use an analytic IK solver to compute the limb postures and blend them with the captured limb postures. …"

Motion Filtering

In general, the motion capture devices that I briefly discussed earlier are capable of providing real-time performance are particularly susceptible to noise. Magnetic motion capture systems which are widely used for real-time motion capture, suffer from the interference of low-frequency current generating devices such as a CRT-type display (Old TVs). And with that, there always exists some level of jitter or rapid random changes in reported positions and orientations that do not correspond to actual movements.

On the importance-based approach, they maintain a target orientation unit quaternion which is expressed as:

$$\mathbf{q}_e = (w\ (x\ \ y\ \ z)), \text{where } w^2 + x^2 + y^2 + z^2 = 1.$$

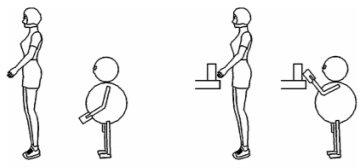
They use this to maintain an internal state vector:

$$\mathbf{x} = (\mathbf{p}^T\ \dot{\mathbf{p}}^T\ \mathbf{r}^T\ \dot{\mathbf{r}}^T)^T$$

Here the rotation vector **r** parameterizes the incremental orientation change of the actual sensor input **q(t)** at the current frame with respect to the target orientation $\mathbf{q_e(t-1t)}$ at its previous frame. **r (t)** can be measured as:

$$\mathbf{r}(t) = \ln\left(\mathbf{q}_e^{-1}(t - \Delta t)\mathbf{q}(t)\right)$$

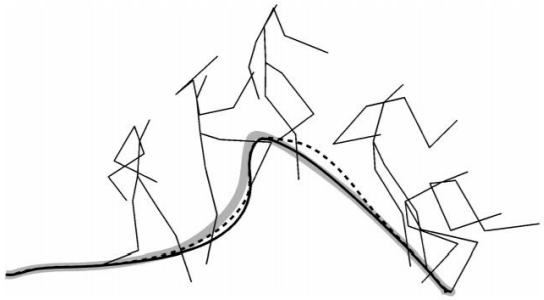
This vector r is converted into its incremental orientation to update the external target orientation and then resets to zero. This helps predict the current position and rotation by first-order approximations.



This helps in correctly interpreting different scenarios for the character where the actor-movement and orientation might be the same.

Importance Analysis

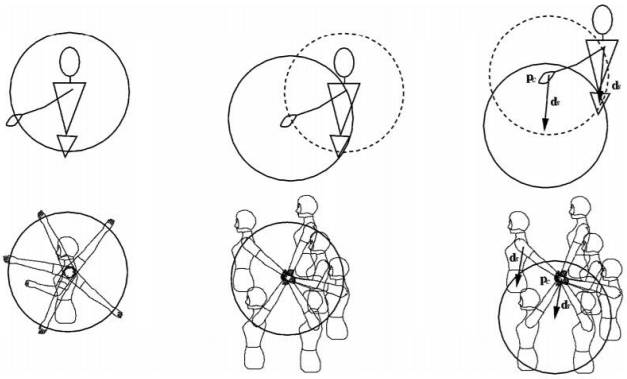
When the performer and the target character do not have the same size and proportion, not all aspects of the original motion can be preserved. A system must determine which aspects of the motion are important to preserve, so that other less important aspects may be changed to preserve them. There are three obvious choices of motion aspects to preserve: the position of the root of the character, the joint angles, and the positions of the end-effectors:



Trajectories of the left foot generated by varying importance measure.

Real-time IK Solver

For computer puppetry, we must position the character such that the important aspects of a captured motion are preserved while providing real-time performance. This means that for the importance-based approach we need an IK solver that can incorporate important measures and also have real-time performance. On this approach, we first attempt to make the character satisfy the constraints as much as possible by moving the root position. This lets us know the range of motion in each node.



Reachable ranges: range of hand, shoulder, and root position.

This lets us find out if a specific target cannot be reached by an end effector under specific motions, that way the character is shown without a limb just passing through its body in order to reach a close by item.

References:

Videos

- https://www.youtube.com/watch?v=Cnyj4b2bgBg

Articles/online references

- http://www.davidbordwell.net/blog/2010/02/23/motion-capturing-an-oscar/
- http://nookkin.com/articles/computer-graphics/computer-puppetry.ndoc
- http://www.grifu.com/vm/
- Computer Puppetry: An Importance-Based Approach
  (Online link: http://mrl.snu.ac.kr/Papers/tog2001.pdf)