An Artificial Mirror Neuron System for Executing and Recognizing Transitive Actions

Evan Lloyd

***Abstract*—The theory of mirror neuron systems offers a plausible explanation for the ability of primates to infer the intentions of others by observing and recognizing their actions. Such systems are interesting from an artificial life perspective, as they represent the unification of behavior, perception, and cognition with applications towards learning. While some work has gone into the construction of artificial mirror neuron systems[1][2], these approaches suffer major limitations. The mirror neuron systems so far created are either uni-directional (i.e. capable of recognizing actions but not executing them, as in [1]) or operate in an overly simplified environment. A more realistic system would be both bi-directional and embedded in a physical simulation. In this report, I present my implementation of a system that, although simple, meets these criteria.**

I. Introduction

A. Mirror Neurons

Mirror neurons are neurons found in the premotor cortex that fire both when an action is observed and when it is performed. They were discovered in macaque monkeys by single neuron recordings. Mirror neurons range in the specificity of actions they respond to; some respond to a broad class of actions, such as any sort of grasping, while others respond only to a very specific action, such as using a precision grip to bring an object to the mouth. Mirror neurons are not restricted to relating proprioceptive and visual information to actions; for example, macaques show mirror neuron responses to distinctive *sounds* associated with an action, such as paper tearing or a peanut shell cracking[6]. While there is no direct evidence for mirror neurons *per se* in the human cortex, there is indirect evidence for analogous mirror neuron *systems* coming from functional imaging studies. For example, a recent study[3] showed highly localized regions active during the execution or observation of specific hand, finger, mouth, or foot actions. The link between observation and action that mirror neurons provide may also be important to imitation learning, in that an individual could learn a new skill by mapping perceptual sequences of an observed action to the motor sequences that would allow him to carry it out.

In monkeys, mirror neuron responses seem to be restricted to goal-directed actions such as reaching for food or bringing it to the lips[5]; by relating an observed action to its target, a monkey may be able to understand the intentions of the actor, which could have social value. In humans, abstract actions having no tangible goal are also represented. It is speculated that auditory mirror neuron sytems are essential to the understanding of speech[10]; indeed, Broca's area, a brain region unique to humans and associated with language, is believed to be functionally analogous to the monkey premotor area F5 in which mirror neurons were first discovered. Functional imaging studies in which mirror-like responses were observed both in producing and listening to specific speech sounds[4] seem to show that such mirror systems do exist, although it is unclear what their exact roles are.

B. Artificial Mirror Neurons - Motivation

Why attempt to imitate biological mirror neuron systems? There are several competing models in the literature[11], [12], [8] of cognitive architectures integrating mirror systems, but relatively few attempts at implementing these ideas in realistic scenarios. Just as the imitation of neural function has been host to a wealth of applications in AI research, explicitly modeling mirror neuron systems may open up new possibilities and potentially help to solve outstanding problems in the field. On the other hand, it may be that all the interesting aspects of mirror neuron systems have already been captured through, for example, visual-motor feedback systems as in [7]. This is very much an open question, and likely can only be addressed by implementing such systems and observing their performance. Regardless of the utility of explicitly implementing mirror systems, however, the model itself offers an extremely useful metaphor for understanding and modeling a diverse range of behaviors and cognitive functions. It explains feedback involved in the execution of complex motor tasks (e.g., via the hand-eye system), imitation, and deducing the intentions of others in a concise, unified framework.

C. Related Work

In [1], a simple mirror neuron system was developed to recognize and distinguish among several different types of grasping behaviors through observation of a simulated hand. It used feedforward neural networks to learn to recognize actions that were performed by a virtual hand and arm controlled by inverse kinematics to execute different types of grasps on objects of varying sizes, shapes, and locations. This model deviated from biological mirror system architecture in an important respect: it was disconnected from the motor *control* mechanism, and thus was capable only of action recognition. Additionally it failed to capture the goal-orientedness property of monkey mirror neurons, in the sense that the target object's position was an *input to*, rather than an *output of*, the action recognition system.

In [2], a sensory-motor map was learned in an unsupervised setting in a simple blocks world. While the system was successful both in predicting the result of an action and in determining what action produced an observed effect, the range of actions and possible world states was very small.

In [12], mirror neurons are treated as the mediators of a multi-modal associative memory, allowing the sytem to, for example, retrieve the visual appearance of an action given its somato-sensory or motor representation. On top of this a self-organizing cognitive control mechanism, capable of learning by imitation, is developed. While the functional aspects of the system are discussed at great length, no actual implementation is presented.

In [9], human actions (different types of grasps, as in [1]) are recorded with motion capture technology, providing both kinesthetic and visual information, which is used to recognize the action type. In a separate experiment, an artificial mirror neuron system embodied in a robot learns to recognize some simpler actions (poking and sweeping) and attempts to imitate them. In contrast to the other models, the system learns not through a neural network, but by a adjusting a Bayesian model, parameterized as a mixture of Gaussians, with an EM procedure. As the robot had simple flippers instead of hands, the arm system had only 6 degrees of freedom.

D. Goals for This Project

Initially I had hoped to leverage several open source libraries to get a physical simulation controlled by inverse kinematics up and running quickly, and from this train and demonstrate mirror neuron systems through the interactions of several agents. Unfortunately those first steps turned out to be orders of magnitude more difficult than I anticipated. Hence I include only a single agent, although I demonstrate the ability of its mirror neuron system to anticipate the results of its own actions. Specifically, I show that the same neural architecture can accomodate both action performance and recognition. By learning the motor-sensory map in both directions simultaneously(from the same training examples), the system approximates true Hebbian learning.

To demonstrate the mirror neuron system, a transitive action is learned. That is, given a target object, the actor is capable of performing a series of movements that bring about a desired state relative to that object. The same system is also capable of anticipating the target of an action through observation of the actor's movements. The mirror neuron network learns from stochastically generated action instances; from a set of random initial parameters, a separate controller based on inverse kinematics generates the motion required to complete the action. The performance of the network can be evaluated by noting both how well it performs the action, and how well it predicts the target of the action. For simplicity, I include only one type of object (a tennis ball), and fix the actors legs and feet to the ground, so that only the upper body is controlled.

In contrast to prior work, I will not be attempting to *classify* different types of actions. This is a very difficult task well represented in the literature; the methods presented here are not intended to advance the state of the art. Instead, I will be investigating the problem of identifying the *goals* of an actor performing an action. While the interpretation of an actor's intentions is certainly dependent on *which* action is being performed, the ease with which humans perform this task indicates that the anticipation of an action's target is performed in parallel with recognition of the type of action itself.

II. Implementation

A. Open-Source Libraries

The development of this project was greatly assisted by the use of open-source libraries. Each library was chosen for its maturity and feature base. While the libraries were quite useful in their own right, integrating them turned out to be decidedly non-trivial. The author intends to release the integration code to the open-source community to encourage further collaboration among the users of the various projects.

1) Ogre3D: Ogre3D[15] is an open-source mid-level graphics library supporting hierarchical scene structures, loading of 3D meshes, and hardware-accelerated rendering of articulated meshes. Version 1.6.2 was used for this project.

2) Bullet Physics: Bullet Physics[14] is an efficient open-source physics engine supporting rigid and soft body simulation with constraints. For this project, it has been integrated with the Ogre3D rendering engine by registering callbacks defined in the library to functions responsible for updating the locations of the rendered models, as well as the states of their articulations. Version 2.74 of the library was used; the specific settings and algorithm choices used are listed in Appendix A.

3) OpenTissue: OpenTissue[16] is an open-source meta-library supporting a variety of features, including physics modeling, character animation, and math and GPU programming. For this project, only the inverse kinematics features were used.

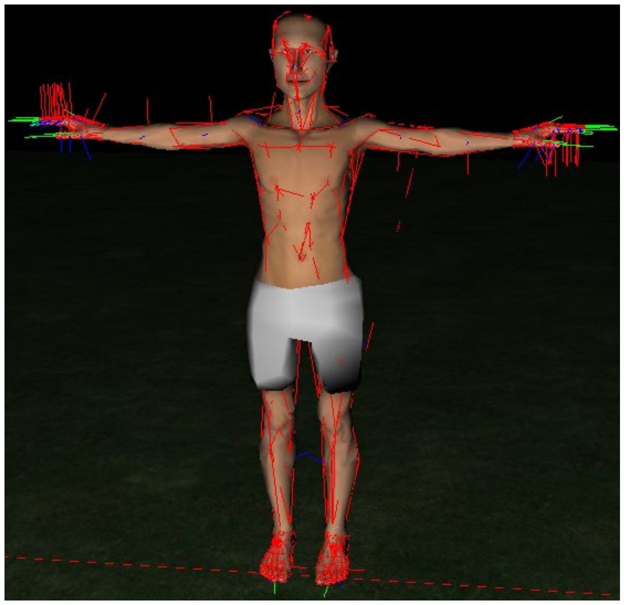
4) FANN: FANN (Fast Artificial Neural Network)[17] is a simple, efficient neural net library supporting a variety of network structures that learn via the backpropagation algorithm. Version 2.1 beta of the library was used.

B. Graphical Model

The human character model for this project (Figure 1) was generated by MakeHuman[18]. It is an articulated model with one bone representing the head, neck, shoulders, feet, hands, torso, and hips. The arms and legs are composed of two bones each. Each finger and toe is articulated with three bones.

C. Physical Model

To create the physical model of the character, the graphical model, with the exception of the hands, was decimated to obtain a much simpler version. For each bone in the model's articulation, a rigid body is created as the convex hull of a set of vertices roughly corresponding to the part of the mesh most strongly influenced by that bone. These divisions of the mesh were done by hand to ensure no overlap. Masses corresponding to those of the "medium" aviator from [13] were assigned to these bodies. At each joint, a ball-and-socket style constraint is applied. In some cases, some dimensions were locked to simulate hinge joints (such as the knees and distal joints of the fingers). In general angular limits were specified that allowed a range of motion that appeared to be appropriate for a moderately flexible individual.



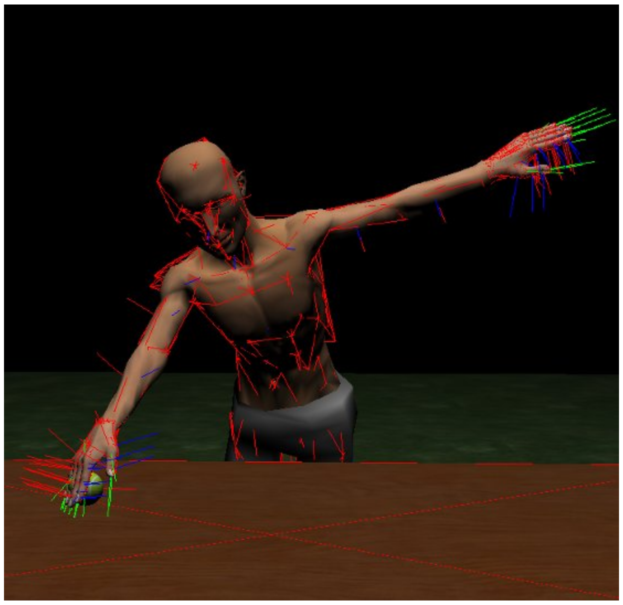


Figure 1. The graphical model in the reference pose with physical model overlayed in wireframe.

Figure 2. Reaching for a ball.

D. Motor Control

The basic method of motor control is through the definition of bone orientation targets. This is accomplished on several levels. At the highest level, sequences of poses are defined that will carry out the desired action. A mid-level controller is responsible for switching between target poses, essentially iterating through this sequence. At the lowest level, a PID controller is used to bring the body into the desired configuration. This is analogous to the concept of key-framing in animation, where the animation being sequenced is defined procedurally rather than by an animator; also, the interpolation process (i.e., the low-level motor control) is done in the context of a physical simulation, and as such can be realistically perturbed by external factors.

1) Low-level Motor Control: Low-level motor control is implemented by a PID controller operating on angular momenta. At each physics tick, for each body in the model the disparity d^i between the target pose Θ^i_T and the current pose Θ^i (represented as quaternions) is calculated according to

$$d^i = \Theta^i_T \cdot (\Theta^i)^{-1}.$$

From this a target angular velocity, ω^i_T for that part is defined, proportional to the magnitude of the disparity, scaling linearly up to a maximum velocity:

$$\omega^i_T = \min(\frac{\omega_{max} d^i}{\|d^i\|}, sd^i).$$

The vector difference e^i between the i^{th} body's current angular momentum vector ω^i and its target is computed as $e^i = \omega^i_T - \omega^i$ and used as the error input to the PID controller formula, which gives a correctional angular impulse

$$\Delta\omega^i = K_P \cdot e^i + K_D \frac{de^i}{dt} + K_I \int_0^t e^i dt.$$

The proportional gain K_P, the derivative gain K_D, and the integral gain K_I are global constants. The derivative $\frac{de^i}{dt}$ is approximated as $\frac{e^i(t)-e^i(t-\Delta t)}{\Delta t}$ where Δt is the amount of time elapsed since the previous physics tick. The integral $I^i = \int_0^t e^i dt$ is approximated as a running sum which is decayed each tick according to $I^i(t+\Delta t) = (I^i(t) + e^i \Delta t) \cdot c$, for a constant c, $0 < c < 1$. Additionally a maximum response is defined for each part and used to clamp the value of $(\Delta\omega^i)$. This prevents the body from exerting infinite torques, and approximates variability in muscle strengths. For example, the legs can generate much higher torques than the fingers. Finally, the computed angular impulse is applied to the body.

2) Mid-level Motor Control: At a slightly higher level, a motor action is defined as a sequence of poses the model should go through. When the sum of the orientation error magnitudes for each body is below a threshold, the next pose in the sequence is set as the low-level controller's target. This level of control also specifies whether each body part should be controlled, since most actions do not require specific positioning for every part of the body. For example, as reaching is done only with the right arm, the left arm is ignored. When the sequence is complete, the high level controller is signaled.

3) High-level Motor Control: The high-level motor controller is a finite state machine that sequentially sets new targets for the mid-level controller as the target poses are attained. The sequences that get passed down to the lower level are generated by the mirror neuron system, as discussed in section II-E. An action can either be cyclic, in which case attainment of the final goal resets the system to the starting state, or not, in which case after the final state the system continues to hold the most recently attained pose.

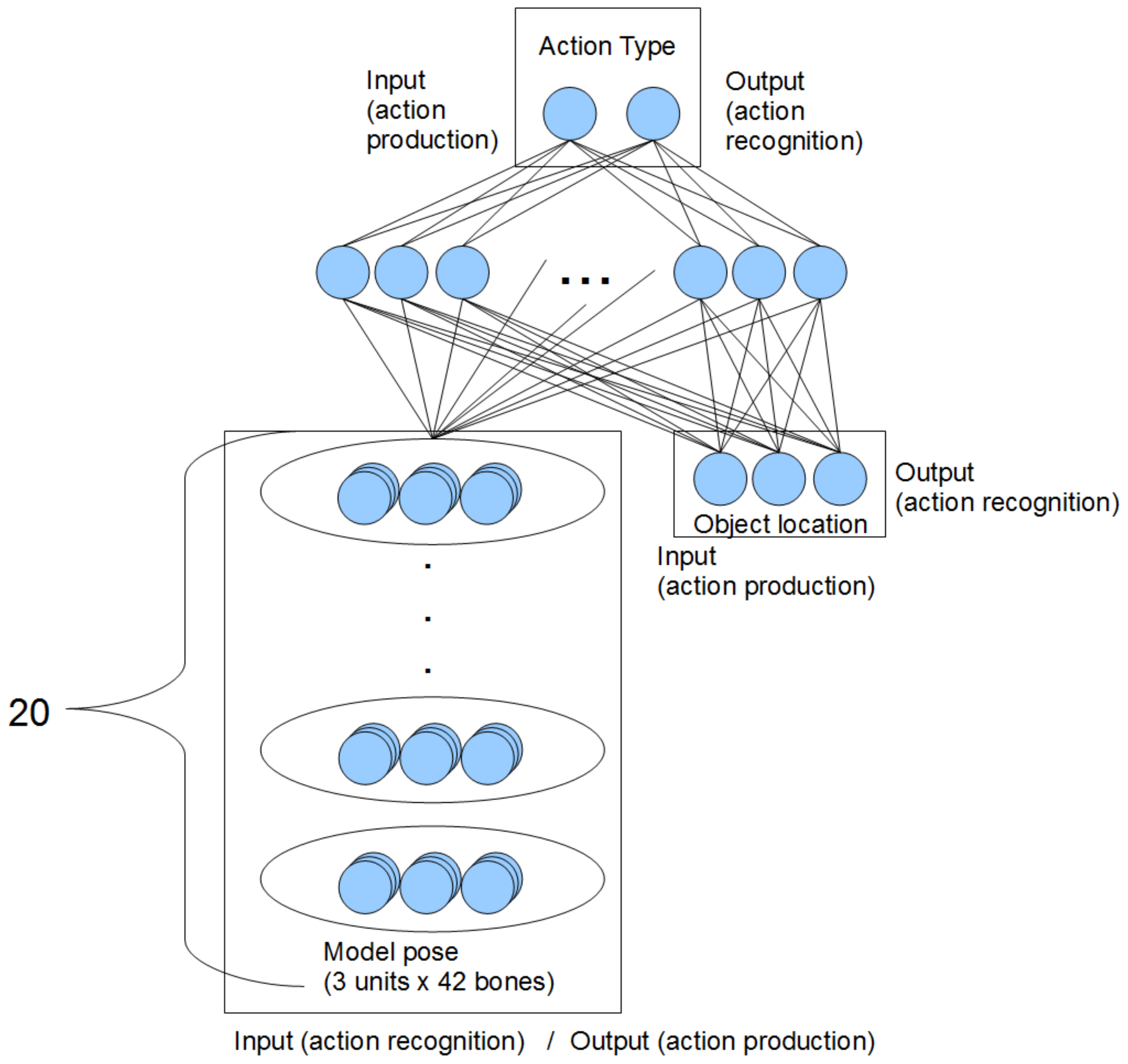


Figure 3. Mirror net architecture. This architecture is implemented as two different fully connected feed-forward neural nets, one for action recognition and one for action production. Despite the fact that they are implemented separately, this archicture does approximate a more plausible Hebbian learning mechanism, in the sense that each network is set up in the same way, with the only difference being which units are considered input or output; corresponding units in each network could be thought of as the same neuron in the mirror neural net. Each network is trained on the same instances.

E. Neural Architecture

I model the action production system as a fully connected feed-forward neural network with one hidden layer containing 50 units(Figure 3). Its inputs are the model's current pose and a position in space. The "reach" action uses this position to direct the action towards the object. The output of the controller is the encoding of a sequence of poses. As in [1] this is implicitly a sampling of a configuration evolving in time, although the entire sequence is generated simultaneously. For this project a sequence length of 20 poses was chosen; while this number is somewhat arbitrary, it cannot be too large since the size of the neural net depends directly on the sequence length.

Action recognition is done by sampling the actor's pose at regular intervals, passing as input the entire observed sequence up to a particular point in time, sampled at 20 evenly spaced points up to the present. This requires an explicit starting time for the action observation sequence, which is a drawback of this method of sequence encoding. The output of the action recognition network is an action type (although currently only one type of action is represented) and an object location, which can be interpreted as the target of a transitive action.

Figure 3 actually shows two separate feedforward networks, however each has nearly identical architecture, with the exception of which units are considered input or output.

1) Encoding Scheme: Model poses are encoded with 3 units per bone, representing the Euler angles defining that bone's orientation relative to its parent. This encoding scheme has the advantage that small errors in the encoding map to small errors in the resulting bone pose, which would not be the case with, for instance, a quaternion representation. Because there are 42 bones being controlled (upper body only), and 20 poses to represent the action sequence, there are a total of $42 \cdot 20 \cdot 3 = 2520$ model pose inputs.

Initially I tried encoding object location as simply *x*, *y*, *z* coordinates, i.e., one coordinate value per node. While this worked very well for action execution, it failed miserably for action recognition; error rates remained large during training and the system failed to predict any value for the target location. In the final version, I changed the encoding to a more distributed representation with three banks of units (one bank per coordinate axis) where each unit responded to a particular number according to a Gaussian function; the means of the functions were evenly spaced in the range of possible coordinate values. This works much better, although there is some ambiguity in how to extract a single target location from this representation. In this project I simply take the average coordinate values of units whose responses are above a threshold.

Because the action production system requires the model's current pose, the first pose in the action sequence becomes an input. Hence, only the last 19 poses are output by the production network; the first pose is assumed to be the model's pose when the action was generated. As in action recognition, the action production system must be explicitly invoked, which is handled by a user command during the simulation.

2) Generating Training Data: Inverse kinematics is used to generate action sequences for the learning module. The inverse kinematics model is given the same joint constraints as for the physical model. The steepest descent algorithm from OpenTissue is used to optimize the joint angles, bringing the model's bones as close as possible to the desired configuration. The target pose is specified on a per-bone basis as a target position, rotation, or both. Additionally a bone can be flagged as "don't care" and will not be considered in the objective function.

For the reaching action, the character is made to reach towards the target object by moving the hand to a position slightly above it. To accomplish this, the inverse kinematic target for the hand articulator is simply set to this position above the target. The generated animation is sampled to generate the target sequence for the learning module. 10,000 training instances and 5,000 test instances were generated by randomly assigning a position to the target (on a table in front of the character) and following this procedure. For each training instance, the starting pose was set to the reference pose.

F. Running the Simulation

The physical simulation is allowed to perform updates before the rendering of each frame. To guarantee frame-rate independence, which is necessary since the simulation is discrete, the update rate of the simulation is locked at 120 Hz. Articulated models are updated by setting the orientation of each of their bones to match that of the corresponding rigid body. The model is translated and rotated to the position and orientation of the root bone (for human characters, the hips) to be consistent with the graphics engine's coordinate system. At each frame, characters with motor controllers receive updated joint configurations and apply angular impulses according to the rules outlined in section II-D.

G. Challenges

The major challenges in implementing this project came from the different design philosophies and representations used in the integrated open source modules. Specifically, differences in the coordinate system definitions made it very difficult to integrate Bullet with OpenTissue; lack of clear documentation for these projects made the process largely one of trial-and-error. Tuning the low-level pose PID controller proved to be difficult as well.

III. Results

A. Video Demonstration

A video demonstration is available at the author's web site for the project, http://www.cs.ucla.edu/~elloyd/LabMan.

The video starts by demonstrating the physical dynamics of the human actor.

Next, it shows the behavior of the low-level motion controller in maintaining a specific pose, demonstrating its resilience to perturbations created by gravity and by user-applied forces.

In the following segment, the character is shown reaching for the target object under the influence of the full motion controller. The target is repeatedly moved around to random locations, which demonstates what the training instances would look like. The same experiment is then repeated under the additional influence of gravity.

Finally, a mirror neuron system trained on the reach action is demonstrated. The target is randomly positioned, and the production neural net queried for an action sequence which is then loaded into the motion controller. While the character is moving, the recognition neural net is continously mapping the observed sequence, attempting to predict the location of the object to be grasped from the action sequence alone. The currently predicted location is represented in the video by a transparent red ball.

B. Neural Performance

The neural nets were trained using FANN's implementation of the iRPROP algorithm, which was allowed to run for between 100 and 500 epochs on the 10,000 training set instances. Several different runs were made to determine the effect of different numbers of hidden units. As a result of these tests, I decided to use 50 hidden units. The networks successfully managed to learn the data, while still generalizing to the test set of 5,000 instances. Mean Squared Error was 0.001665 and 0.000099 for action recognition and action production respectively.

C. Simulation Performance

The simulation is quite efficient, enough to be run in real-time on most modern systems with recent graphics cards; on the test system (Core 2 Duo E6850, GeForce 8800GT), frame rates were typically higher than 700 frames per second.

IV. Conclusions and Future Work

A. Conclusion

I have successfully implemented an efficient physical simulation of a human character, taking into account realistic mass proportions and joint constraints. The character is controlled at a high level by a simple neural network, which approximates the action recognition and production capabilities of mirror neuron systems. At a lower level, a PID controller acting on angular momenta keeps the actor at the desired pose despite perturbations resulting from gravity or user intervention. This also allows the character to transition between target poses smoothly.

While only one type of action is learned, this is still interesting in the sense that the system learns to anticipate the *goal* of the action from its observed trajectory. This ability would be a useful feature of a more sophisticated action recognition system capable of distinguishing among many different types of actions. Indeed, this information may actually assist the task of action identification; for example if a monkey seems to be directing itself towards a banana, it becomes more likely that he is performing a grasping action than, say, a smashing action.

B. Improvements and Future Work

To more accurately reflect the structure of real mirror neuron systems, it would be desirable to have feedback between the high- and low-level motor controllers to increase the accuracy of the motor actions and respond to unexpected changes. For example, as it stands currently if some external force nudged a ball the character was reaching for, he would futilely attempt to grasp it in its former position. At the low level motor control level, a simple refinement would be to add a velocity target for each articulation, which would model the effect of muscle tension by making the body respond more stiffly to perturbation.

The systems presented here could benefit greatly from more realistic models of perception. While there is plausibility in a character being able to sense the configuration of his own body kinesthetically, it is of course absurd that he could do so for another agent. Visual perception of hand configuration was demonstrated in[1], but this method would require much improvement since it relies on the color-coding of articulation points on a specific geometric model. Using characters of different sizes and shapes would be an interesting experiment as well, as a realistic action recognition system ought to be able to handle variability in actor appearance.

Finally, the neural architecture needs improvement, as it is not scalable due to the method of sequence encoding. A more realistic network design would use a recurrent model. Additionally, it would be preferable to use a true bi-directional network (such as a self-organizing map) instead of two separate networks for control and recognition. The system would benefit greatly from improved encodings to eliminate the dependency on the starting pose for action recognition; as is, the system's output is unpredictable if starting from a pose far from the starting poses in the training data.

References

[1] E. Oztop and M. Arbib, “Schema Design and Implementation of the Grasp-related Mirror Neuron System,” Biological Cybernetics 87, 2002: 116-140.

[2] M.V. Butz and S. Ray, "Bidirectional ARTMAP: an Artificial Mirror Neuron System," Neural Networks, 2003. Proceedings of the International Joint Conference on Neural Networks, vol.2, 20-24 July 2003: 1417-1422.

[3] V. Gazzola and C. Keysers, "The Observation and Execution of Actions Share Motor and Somatosensory Voxels in all Tested Subjects: Single-subject Analyses of Unsmoothed fMRI Data," Cereb. Cortex, vol. 19, no. 6, June 2009: 1239-1255.

[4] S.M. Wilson, A.P. Saygin, M.I. Sereno, M. Iacoboni, “Listening to speech activates motor areas involved in speech production,” Nature Neuroscience, vol. 7, June 2004: 701 - 702

[5] V. Gallese, L. Fadiga, L. Fogassi, G. Rizzolatti, “Action recognition in the premotor cortex,” Brain, vol 119, April 1996: 593-609.

[6] E. Kohler, C. Keysers, M. A. Umilta, L. Fogassi, V. Gallese, G. Rizzolatti, “Hearing Sounds, Understanding Actions: Action Representation in Mirror Neurons,” Science, vol. 297, 2 August 2002: 846-848

[7] R. Garrido, A. Soria, M. Trujano, “Visual PID Control of a redundant Parallel Robot”, Electrical Engineering, Computing Science and Automatic Control, 5th International Conference on, 2008: 91-96

[8] V. Gallese, M. N. Eagle and P. Migone, “Intentional Attunement: Mirror Neurons and the Neural Underpinnings of Interpersonal Relations,” J Am Psychoanal Assoc, vol. 55, no. 1, 2007: 131-175

[9] G. Metta, G. Sandini, L. Natale, L. Craighero, L. Fadiga, “Understanding mirror neurons: a bio-robotic approach,” Interaction Studies, vol. 7(2), 2006: 197-232.

[10] G. Rizzolatti and M. A. Arbib, “Language within our grasp,” Trends in Neurosciences, 21(5), 1998: 188-194.

[11] G. Csibra, “Mirror neurons and action observation: Is simulation involved?” Interdisciplines 2004: http://www.interdisciplines.org/mirror/papers/4

[12] J. Wiedermann, “HUGO: A Cognitive Architecture with an Incorporated World Model,” Complex Systems, European Conference on, 2006: http://sbs-xnet.sbs.ox.ac.uk/complexity/complexity_PDFs/ECCS06/Conference_Proceedings/PDF/p108.pdf

[13] J. Haley, *Anthropometry and Mass Distribution for Human Analogues*. Volume 1: Military Male Aviators, March 1988.

[14] Bullet Physics: http://www.bulletphysics.com

[15] Ogre3D: http://www.ogre3d.org

[16] OpenTissue: http://www.opentissue.org

[17] FANN: http://leenissen.dk/fann

[18] MakeHuman: http://www.makehuman.org

Appendix

Since the libraries used in the project were open-source, readers interested in reproducing the results of this project may be interested in the details of the physical simulation, as well as a more detailed description of the steps involved in integrating the various libraries. Source code will be available on the author's website for the project, http://www.cs.ucla.edu/~elloyd/LabMan/

A. Bullet Physics

btDiscreteDynamicsWorld with btDefaultCollisionConfiguration was used as the simulation controller. The update rate was set at 120 Hz, with a maximum number of substeps set at 20. Joint constraints were implemented with the Bullet Generic6DofConstraint class, with the angular limit constraint motor enabled to prevent sudden snapping of the limbs near the joint limits.

B. Ogre3D

Bone orientations are set to absolute (non-inherited) mode. A custom class derived from btDefaultMotionState is responsible for updating the bone orientations at every physics engine tick. The root scene node of the model is updated with the position and rotation of the root bone in the skeleton, since that is where the graphics engine puts the bounding box to determine whether the model should be displayed.

C. OpenTissue

OpenTissue inverse kinematics required some modifications to work with Bullet. Although both systems parameterize joint configurations using Euler angles, Bullet uses the XYZ convention while OpenTissue uses ZYZ. Additionally, to get the actual angle values to line up with Bullet, additional transformations had to be carried out each time the bones' orientations were referenced, since Bullet defines the Euler angles to be relative to the bones' *initial* configuration, while OpenTissue defines them relative to the model's *current* pose.