Hierarchical Attentive Multiple Models for Execution and Recognition of Actions through Training Inverse and Forward Models Explorationally

Alex Cannon & Imoleayo Abel

Abstract

We present an extension to Demiris' and Khadhouri's Hierarchical Attentive Multiple Models for Execution and Recognition of actions, which is more honest to the developmental paradigm of robotics. Specifically, we replace their human-coded prediction models with ones that the robot learns themselves through exploratory motor babbling guided by Categorical Intrinsic-Based Motivation.

1 Introduction

1.1 Social Learning

Intelligent biological organisms do not learn everything they know from scratch. Quite to the contrary, they learn most of what they know from other organisms. A human infant, for instance, does not learn to speak a language on its own, rather it learns that language through imitating its use by other humans. Similarly, robots need not learn everything independently. Just as the infant's language learning is bootstrapped by the language already learned by its caretakers, a robot with the ability to learn socially (i.e. with the ability to learn to perform tasks through interactions with other agents), could bootstrap its learning of tasks with the knowledge of those tasks already acquired by its counterparts. A robot with the ability to learn socially, that is, would not need to "reinvent the wheel", as the saying goes, to learn a new task. If one of its counterparts had already learned the task, the robot could, in theory, learn the task much more quickly from this counterpart.

1.2 Imitation Learning

One such approach to implementing social learning in robots has been through imitation. The idea here is that a robot can learn to perform a task simply by watching a demonstrator perform this task. After watching the demonstration, the robot will then be able to imitate the observed task. One of the main obstacles to imitation learning, however, is representing perceived movement in motor-based terms. [1] For example, how does a robot map its perception of a human demonstrator walking forward to its own motor commands for moving itself forward? If it the robot is to imitate the human demonstrator's forward movement, it must somehow construct this mapping. In other words, the robot must ask itself: "Given that I perceive the demonstrator perform some action, what is that action in my motor terms–the one I must perform to imitate the demonstrator?"). Biologically such a mapping occurs through mirror neurons, which are active both when an action is perceived (e.g. when the organism watches a demonstrator move forward) and when that same action is performed (e.g. when the organism moves forward itself). A large part of imitation learning, then, is concerned with implementing an artificial representation of these mirror neurons. This is the problem that Demiris and Khadhouri try to tackle with their HAMMER architecture as described in their paper "Hierarchical attentitive multiple models for execution and recognition of actions". The work presented in the following sections is a direct extension of the work in this article.

1.3 Related Work and Motivation

HAMMER

Demiris and Khadhouri's HAMMER architecture, from which the architecture presented in this paper (Hierarchical Attentive Multiple Models for Execution and Recognition of Actions through Training Inverse and Forward Models Explorationally, henceforth referred to as HAMMER-TIME) get its name, relies on, at its lowest level, pairs of inverse and forward models. Each inverse model is given a sensory state S_t and a target state S_{t+1}, and outputs the motor action that it thinks moves the agent form S_t to S_{t+1}. Similarly, each forward model is given a sensory state S_t and a motor state M_t, and outputs the target state S_{t+1} that it thinks is produced by S_t and M_t. In their paper, Demiris and Khadhouri describe the role of these inverse and forward model couplings within their HAMMER architecture:

> If HAMMER is to determine whether a visually perceived demonstrated action matches a particular inverse-forward model coupling, the demonstrators current state as perceived by the imitator is fed to the inverse model. The inverse model generates the motor commands that it would output if it was in that state and wanted to execute this particular action. The motor commands are inhibited from being sent to the motor system. The forward model outputs an estimated next state, which is a prediction of what the demonstrators next state will be. This predicted state is compared with the demonstrators actual state at the next time step. This comparison results in an error signal that can be used to increase or decrease the inverse models confidence value, which is an indicator of how closely the demonstrated action matches a particular imitators action.[2]

Although Demiris and Khadhouri go on to describe the attention control system they implemented on top of these inverse and forward model couplings, this is how the core of the HAMMER architecture works.

In their experiment, Demiris and Khadhouri test their HAMMER architecture using pre-coded inverse and forward models. Each inverse model corresponds to a higher level action, and each forward model is simply based on kinematics.

> A number of inverse models were implemented including move effector towards object (in the experiments below objects included a soda can and an orange), pick object, drop object, and move away from object using the ARIA library of primitives provided with the ActivMedia Peoplebot, similarly to our previous experiments with these robots...Forward models were hand- coded for each of the inverse models, using kinematic rules to output a qualitative prediction of the next state of the system for each of these inverse models. For example, given the current position and speed of the hand and a motor command to move the hand to a certain direction, the predicted next state would be closer or further away from an object.[2]

Given S_t, in other words, the inverse model supplies the S_{t+1} that corresponds to the higher level action it represents, from which it derives the primitive motor action M_t that it believes connects S_t to S_{t+1}. Each forward model makes qualitative predictions about the sensory state in the next time step (i.e. derives S_{t+1}) based on S_t, M_t, and the kinematics that Demiris and Khadhouri had "hard-coded" into it. This decision of Demiris' and Khadhouri's to hard-code the inverse and forward models that they use within their HAMMER architecture is what motivated the research presented in this paper.

In our experiment, in contrast to that of Demiris and Khadhouri, we set out to create a simplified HAMMER architecture based on trained inverse and forward models rather than precoded ones. Instead of pre-coding the inverse and forward models into the robot, we represent each forward and inverse model as an artificial neural network that we first have the robot train through an exploratory, motor-babbling phase.

CBIM

Fortunately, a robot controller that trains forward models through motor-babbling has already been developed. The CBIM controller (short for Category-based Intrinsic Motivation), developed by Rachel Lee, Ryan Walker, James Marshall, and Lisa Meeden, was designed as a way of instilling a robot with a sense of productive curiosity, in which the robot focuses on sensorimotor contexts in which it is making maximal learning progress:

> On each time step the robot consults this memory in order to determine which action to take. First the robot senses the world. Next, it generates a set of candidate actions, either by enumerating all possibilities or, if the space of actions is continuous, by generating a random sample of possible actions. Then it concatenates each candidate action with the current sensory information and probes the memory to find all matching regions. With some high probability it selects the candidate action associated with the region with the maximal learning progress. Otherwise it chooses a random region from the matched set. It then executes the selected action, observes the outcome, and uses this data to train the expert associated with the selected region.[3]

The "regions" Meeden et. al. are referring to above represent nodes in a Growing Neural Gas. The forward model within each of these regions, therefore, is associated with a node in the GNG. In our experiment, to have CBIM train inverse models in addition to forward models, we simply inserted an inverse model into every "region" as well. The controller still selects actions based on the learning progress of the appropriate forward model, but also trains at each time step the corresponding inverse model.

2 Experiments

2.1 Training Setup

Using the Pyrobot simulator developed by Douglas Blank, we trained our inverse and forward models on a simulated robot running this modified CBIM controller. We did this by placing the robot in a "playpen" with either a red or a green puck to "play" with (see Figure 1). The robot then spends a pre-specified number of time steps motor-babbling (via CBIM), during which it trains its inverse and forward models to identify predictable relationships between its motor actions and sensory input (sensory input consists of the robot's distance from the puck, the robots location and heading, and the puck's location and color). If the robot either stalls (i.e. runs into a wall) or loses sight of the puck in its camera, it is reset to a random position in the top half of the pen facing south such that it will be able to see the puck. In such a situation the puck is reset as well, to both a random color (either red or green) and a random location in the bottom half of the pen (such that it will be in the robots field of view). After resetting the puck and the robot, the modified CBIM controller takes over once again, dictating motor-babbling actions that continue to train its inverse and forward models.

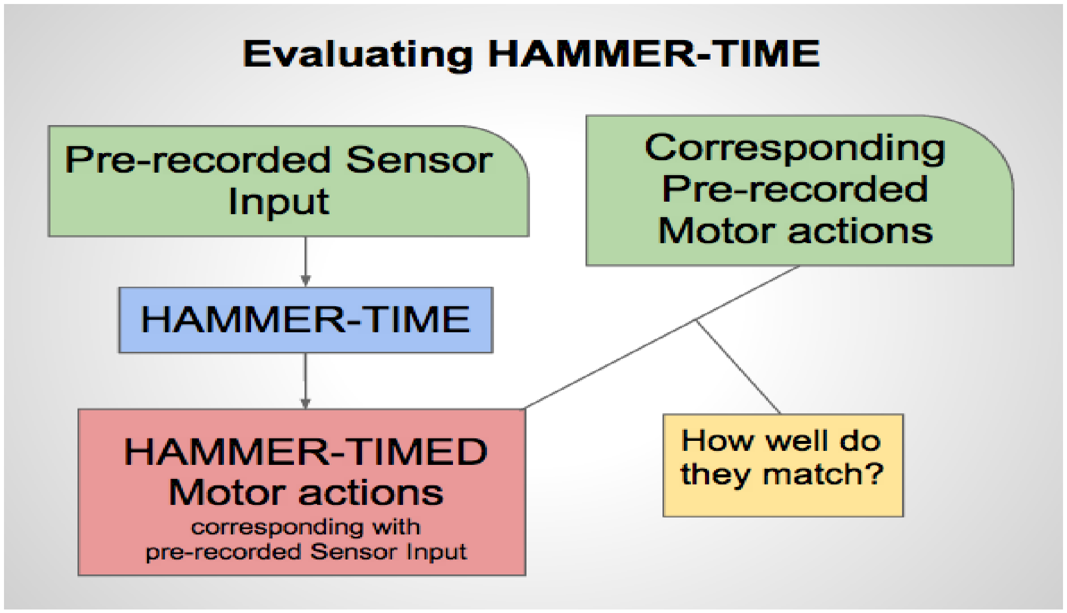
Two main criteria were taken into account when constructing this training environment. The first was simplicity, in the hopes that in a simple environment the robot would train its inverse and forward models after relatively short training trials. The second criteria was congruency with Demiris and Khadhouris experiment. In their experiment, to restate what has been described previously, the robot was given inverse and forward models that corresponded to a set of possible actions that could be performed on one of two objects (a soda can and an orange). This was our inspiration for the two differently colored pucks–one was our soda can and the other our orange (for a more elaborate discussion of the role of puck color in our experiment, see the Results and Discussion section). Similarly, just as Demiris and Khadhouri used the locations of the object and the hand (see Figure xx) as input, we passed as our input the locations of the puck and the robot. In essence, using resources we had available to us, we sought to create an environment that was as if Demiris and Khadhouris robot had been placed in a pen with first a soda can and then an orange, and been told to go play (i.e. to learn to interact with the objects through motor babbling).

CBIM Trials

To test the robustness of the CBIM training across small changes in parameters, we ran several different CBIM trials, each producing its own set of forward and inverse models, and consequently their own unique (but similar structured) HAMMER-TIME architectures. Of these trials, the three most relevant are:
(1) The first trial ran for 7,000 steps, during which the robots choice of motor-babbling [translation, rotation] actions was limited to only the set (-1,-1), (-1, 0), (0,-1), (1,0), (0,1), or (1,1) (7k_1001)
(2) The second trial ran for 25,000 steps, but its choice of motor-babbling actions was expanded to the set (i, j) for i, j in -1.0, -0.9, -0.8, -0.7, , 0.9, 1.0 (25k)

(3) The third and final trial also ran for 25,000 steps with a similar set of candidate actions, only the error threshold on the controllers GNG was raised from 0.75 to 0.95 to try and discourage the creation of so many GNG nodes (and the forward and inverse models that accompany them) (25k_errThresh95).

2.2 HAMMER-TIME Evaluation Framework


Evaluating HAMMER-TIME
Pre-recorded Sensor Input
Corresponding Pre-recorded Motor actions
HAMMER-TIME
HAMMER-TIMED Motor actions corresponding with pre-recorded Sensor Input
How well do they match?

In Demiris and Khadhouris experiment, they could evaluate the accuracy of their HAMMER architecture by looking at which inverse-forward model pair was active, because each of pre-coded these pairs corresponded with some human-defined action. In the most basic sense, they proved the efficacy of their architecture by showing that their robots move-actuator towards orange inverse-forward model pair was active when perceiving a demonstrator move their hand towards an orange.[2] In our experiment, because our inverse-model pairs do not correspond to any easily human-defined action, we do not have this luxury. Moreover, we do not have the resources to run our HAMMER-TIME architecture with a human demonstrator. To overcome these two obstacles, we developed a multi-layered framework with which to evaluate HAMMER-TIME.

The basic idea behind the framework is to record the sensor and motor stream of a robot performing many slightly different versions (i.e. trials) of the same task; in lieu of Demiris and Khadhouris pick orange evaluation task, our experiment uses approach and hit puck. For each one of these evaluation trials (we recorded fifty in our experiment), the sensor stream of that trial is fed through the HAMMER-TIME architecture, which creates motor stream estimate (aka a set of HAMMER-TIMED motors) that it thinks is associated with the sensory stream (i.e. every motor M_t is thought to carry S_t to S_{t+1}).

A more detailed description of this framework is found below:

2.2.1 HardBrain

We implemented a hardBrain framework that builds upon a hammer trial class. The hardBrain scheme generates various puck-approaching tasks as instances of the hammer trial class. Each task object is initiated with the robot and puck at random initial locations and includes motor actions performed to accomplish the puck approaching task, the sensor value at each time step, as well as the initial positions of robot and the puck (in the pen) as object variables. These are the tasks upon which our implemented hammer architecture was tested (referred to previously as "evaluation tasks"). Each hammer trial object also included a dictionary to hold names of CBIM trained sets of inverse and forward models as keys and their corresponding motor predictions as values.

2.2.2 hammer.py

Here, all trial objects containing demonstrations to be imitated are passed in as input and for each trial object, all sets of CBIM trained inverse and forward models are used in the hammer architecture to make predictions on motor values. For each inverse and forward model pair, the inverse model acts upon sensory states at consecutive time steps in the original demonstration and generates motor actions with which the corresponding forward model makes a prediction of the next state of the demonstration. The predicted state is then compared with that of all other forward-inverse model pairs in the same training set. The inverse model motor actions for the pair with the least euclidean predicted state distance from the actual demonstration is then saved as the hammer motor prediction for that time step for that set of CBIM trained models. After predictions for all time steps are made, the name of the CBIM trained set and the list of motor values are then saved as key-value pairs in the motorlog object variable of the trial object. This procedure is repeated for all sets of inverse-forward models with all trial objects. A short pseudocode is provided below.

```
for each trial_object:
        trial_object.hammerMotors = {}
        for each CBIM_set:
                for each forward-inverse model pair:
                        make predictions for next state
                predictedMotors.append(motor actions for closest prediction)
                trial_object.hammerMotors[CBIM_set] = predictedMotors
```

3 Results and Discussion

3.1 CBIM Training

Initially we feared that incorporating inverse models into the CBIM controller while still selecting motor babbling actions based on decrease in the error of the forward model would leave the inverse models relatively untrained by comparison. Empirically, however, our fears to do not seem to have been borne out.

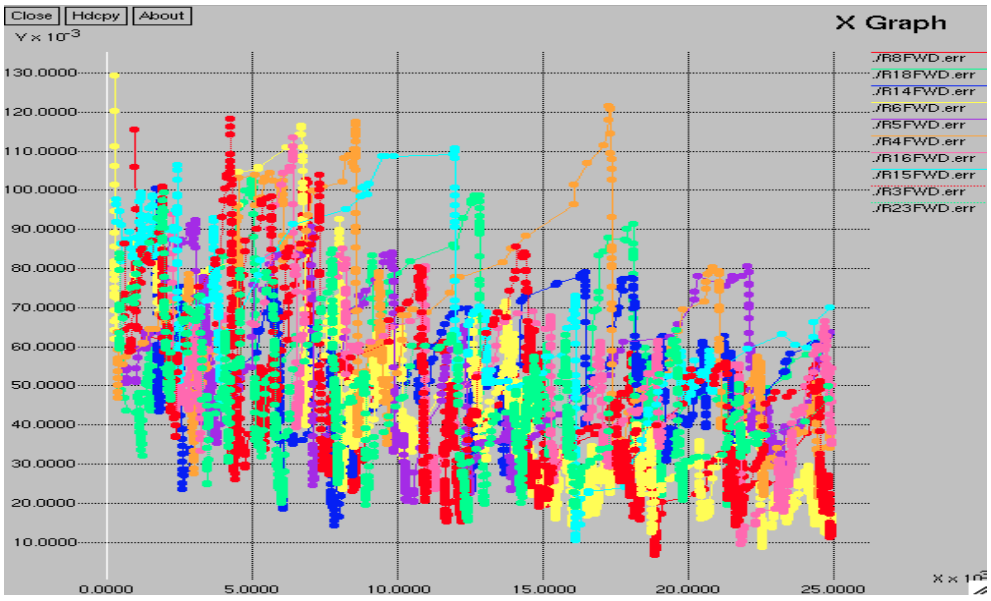


Fig 2: Forward Models CBIM Training Error.

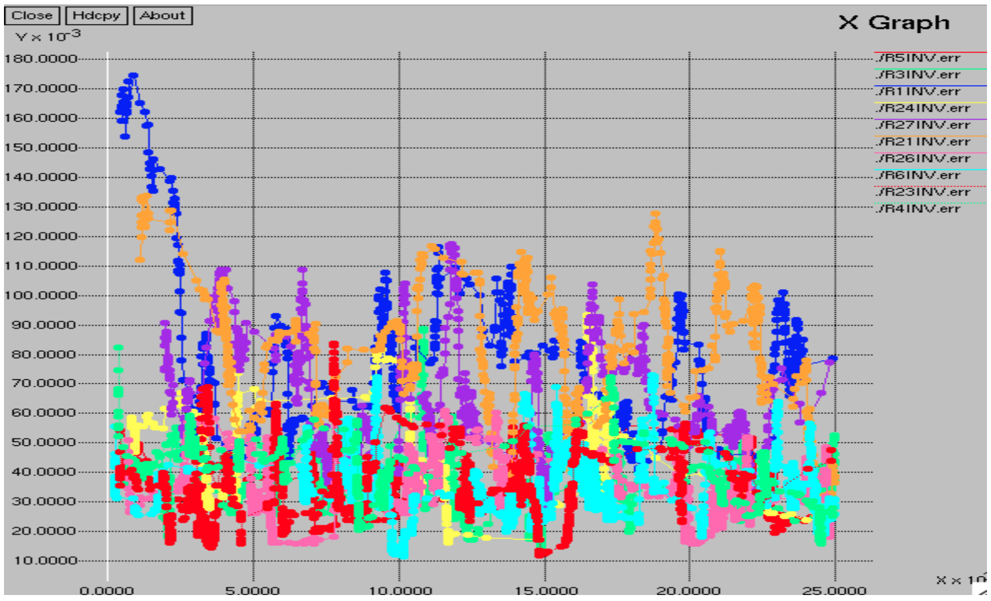


Fig. 3: Inverse Models CBIM Training Error

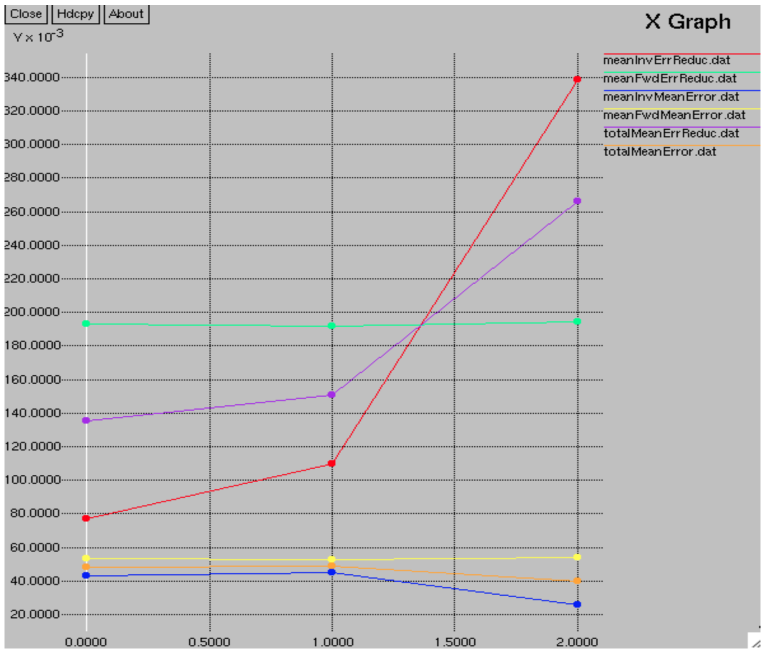


Fig. 4

Figures 2 and 3 show that error of both forward and inverse models are similar over the course of the training period.

Moreover, while error rates among the experts seem to vary sporadically throughout the training period, overall trends indicate a clear increase in the average error reduction for both forward (green line in Figure 4) and inverse models (red line in Figure 4) across all three CBIM training trials.

3.2 HAMMER-TIME

After producing the HAMMER-TIMED motor actions explained in the Experimental Setup, we compared each of them to the motor streams of the evaluation tasks from which they had been derived (see Figure 1). Figure 5, for example, shows the trail of a robot performing an evaluation task (a task that HAMMER-TIME is supposed to imitate). Figure 6 shows HAMMER-TIMEs attempt at an imitation of that task. The green line in Figure 7 shows the error in Euclidean distance between the two robots at each time step (the other lines correspond to other CBIM trials—other inverse and forward models).

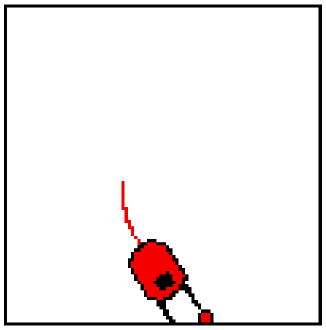


Fig 5.

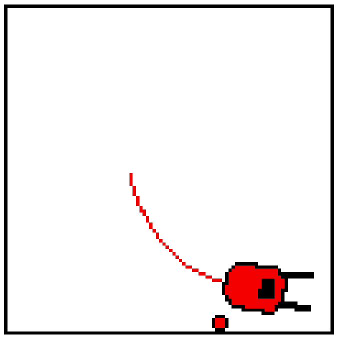


Fig. 6.

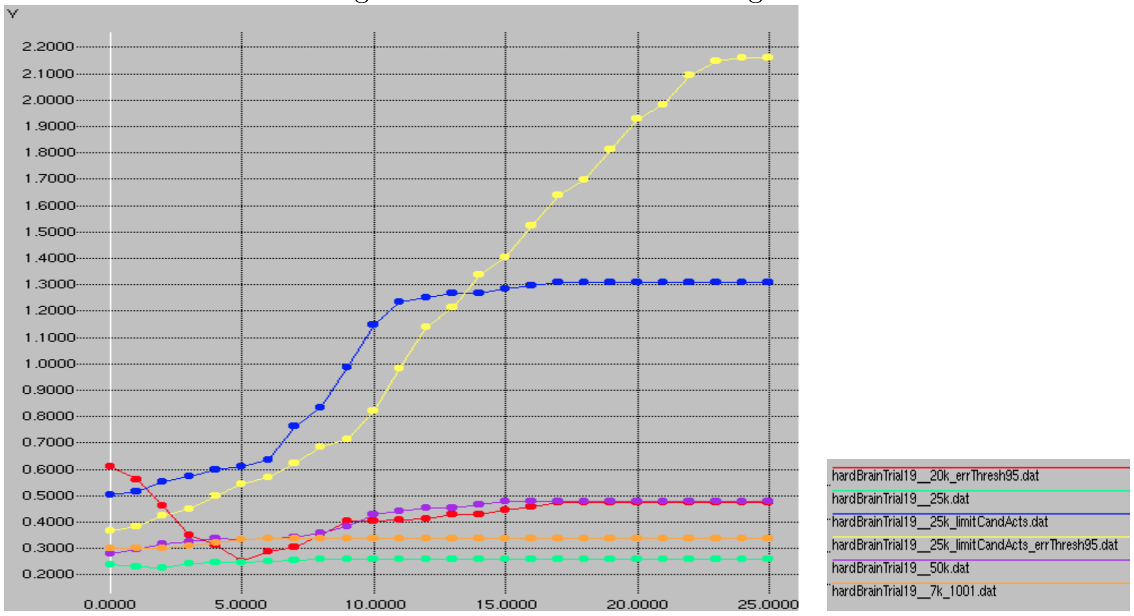


Fig. 7

Fig 5. shows the trail of the robot for an actual demonstration while Fig 6 shows the trail of the robot implementing HAMMER's predictions for the same task. The inverse and forward models used to make the prediction shown in Fig. 6. are from the CBIM_25k training set. Fig 7. shows the Euclidean error of all sets of CBIM models used overall in our experiments. The errors are computed by taking the Euclidean distance between the predicted and actual position of the robot at every time step in the demonstration

4 Summary and Conclusions

Such similarities such as those between Figure 5 (the evaluation task) and Figure 6 (HAMMER-TIME's imitation of that task) were found across the bulk of our evaluation tasks. These results support our hypothesis that trained inverse and forward models, as opposed to pre-coded ones like those found in [2] are compatible with the HAMMER architecture.

Acknowledgments

We would like to acknowledge the efforts of Professor Lisa Meeden in providing us with the knowledge base with which we performed this experiment. We also acknowledge her helpful suggestions and support through the implementation process. Our appreciation also goes to all students in the Adaptive Robotics class of Spring 2012 for their constructive comments during our check point and final demonstrations.

References

[1] Cynthia Breazeal and Brian Scassellati. Robots that imitate humans. *TRENDS in Cognitive Sciences*, 6(11):481–487, November 2002.

[2] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple modes for execution and recognition of actions. *Robotics and Autonomous Systems*, 54:361–369, 2006.

[3] Rachel Lee, Ryan Walker, Lisa Meeden, and James Marshall. Category-based intrinsic motivation, 2009.