Validating a Real-Time Word Learning Model Tested in USARSim and on a Real Robot

Richard Veale and Paul Schermerhorn and Matthias Scheutz
Human-Robot Interaction Laboratory
Cognitive Science Program
Indiana University
Bloomington, IN 47406, USA
{riveale,pscherme,mscheutz}@indiana.edu

***Abstract*— In this paper we present a novel vision-based word-learning model that was developed and first tested using the USARSim simulation environment before being validated in the real world. We describe the learning architecture and the steps we took to integrate USARSim into the system. Models that were trained in the simulator evolve similarly to those trained on input from cameras, and perform comparably to their real-world counterparts on a subsequent *real-world* color recognition task.**

I. INTRODUCTION

Robot control architectures are becoming increasingly complex; prototyping and testing these systems can be time-consuming and expensive. Simulated environments like USARSim [2] can facilitate the process by allowing faster and easier debugging. They can also be invaluable tools for developing and testing real-time models of embodied situated cognition. Specifically, sufficiently accurate simulations of real-robots will allow us to develop and test cognitive models effectively, without the need for the physical robot to be available throughout the process. Rather, the robot will only be needed for the final validation of the already fully developed and tested simulation model.

This paper describes work in our lab integrating USARSim into a robot development environment for the development of cognitive models, specifically to explore the effects of physical embodiment and situatedness. We will describe the details of the modeling setup using a specific neural network model of word learning that is currently under development in our lab. Using this model, we show the sequence of model development and testing in USARSim, followed by a final validation on the physical robot.

II. BACKGROUND

Cognitive modeling has a long tradition in cognitive science, going back to the late 70ies when the main architectures ACT and SOAR emerged ([1], [7]). Subsequently, various other symbolic architectures were introduced (including EPIC [8] and Prodigy [4]) as well as different kinds of neural network architectures. Typically, cognitive models developed for these architectures are only run in simulation, i.e., with simulated inputs and outputs, instead of being connected to real sensors and effectors. Yet, for studying embodied cognition, it is important to be true to the real-time real-world nature of sensory input and motor outputs. Hence, the model will either have to be run on a robot or in a physical simulation environment that can faithfully simulate the important physical aspects of sensors and actuators. Connecting a model to a real-time system (robot or simulation), however, is challenging because cognitive architectures have typically not been designed to support it.[1] What is needed is an infrastructure that can connect the inputs and outputs of the architecture with the outputs and inputs of the sensors and actuators. We will describe our ADE system that has been successfully connected to the USARSim simulation environment as well as several robots and embedded devices.

III. A BRIEF OVERVIEW OF THE ROBOT AND THE SETUP

We integrated the USARSim environment into our robot development infrastructure ADE, the *Agent Development Environment* [11]. ADE allows users to construct agent architectures using modular components (called *ADE servers* that provide services (e.g., access to sensors or effectors) to other ADE servers, and can be distributed across multiple hosts. An *ADE registry* serves as a "yellow-pages" service for ADE servers to connect them with the other servers they require. An ADE server submits a request for a service (represented by an RMI interface) to the ADE registry, and the registry checks the credentials of the requesting server and forwards the information about the new resource. From that point on, all communication between those two servers is direct, without having to go through the registry (see [10] for more details).

As noted, ADE servers interact via pre-defined interfaces. Servers have been defined for many sensors (e.g., speech recognition, laser rangefinder, and GPS localization) as well as many effectors (e.g., robot bases such as the Pioneers and Segways, and a humanoid RoboMotio torso). The ADE USARSim server fits nicely into the system by instantiating models of many of these sensors and effectors (some predefined by USARSim, others constructed in our lab [5]) and

[1] However, note there are examples of both SOAR and ACT-R controlling robots, and, of course, neural network architectures (e.g., [14]).

replicating their interfaces, allowing other ADE servers to utilize those resources as if they were physically instantiated. The USARSim server communicates with the simulation environment via a socket connection to the GameBots engine, which allows the server to instantiate models and monitor and manipulate their states. In addition to an interface for the provided Pioneer robot and SICK laser rangefinder models, we have constructed in our lab a full model of the RoboMotio Reddy humanoid torso, including the head with manipulable eyes, eyebrows, and lips, movable head, and arms with three degrees of freedom. The model was created using a combination of Blender, Inkscape, and the Unreal editor (for details, refer to [5]).

The ADE vision server has been configured to work with a variety of cameras (including IIDC Firewire and USB cameras) and can provide information about visually detected faces, color blobs, and environmental conditions (e.g., darkness) to other ADE servers as requested. Using the vision server in the USARSim environment required us to extend the server to analyze frames from the UT virtual camera instead of a real camera. Our solution is based on the "SDL Hook for USARSim on Linux," which modifies the SDL library to intercept the frames and redirect copies of them to a socket in addition to displaying them on-screen.[2] When the vision server is directed to use USARSim instead of a camera, the only difference is in the initialization (e.g., open a socket instead of initialize a camera) and the method used to grab frames. All subsequent operations (i.e., analysis performed on the frames) is performed in the same way it would be if the frames were coming from a camera; the analysis code does not need to be modified to operate differently than it normally would. Similarly, because other ADE servers only need to know the interface exported by the vision server, whether the vision server is operating in the real world or the UT environment is completely opaque outside the vision server; other servers neither know nor care from which environment the visual information comes any more than they care what kind of camera provides the frames in a real environment.

The tests described below (see Sec. V) were conducted using the ADE vision server configured to use two "cameras." The tests conducted in the real world use a Unibrain Fire-i IIDC Firewire camera, whereas the tests in the Unreal environment use the virtual camera defined by the SDL hook. Identical vision processing was performed in each case before the results (in this case information about the color, size, and relative location of each blob detected) was passed on to the word learning model component, also encapsulated as an ADE server.

IV. MODEL DEVELOPMENT: THE WORD LEARNING MODEL

The word learning model is an associative, incremental model implemented in several interconnected artificial neural networks. The desired behavior is to learn word-reference associations in an unsupervised fashion. It has been hypothesized that human infants learn such associations in a statistical manner, tabulating co-occurrences, usually in a batch-fashion (e.g. [13]). However, it has been shown that such simple models are not sufficient to fit the human data [12] [6] [15]. Humans are embodied agents whose experiences are inextricably situated in time and space [3]. In light of this, temporal aspects and environmental context must be taken into account in the learning model.

We introduce a learning model in which learning is *incremental*, i.e. the system is modified by every experience, and then it is the modified system which goes on to encounter further experiences. This type of model affords important qualities, perhaps most importantly the ability to learn based on co-occurrences in a non-linear fashion. In other words, the association strength learned from experiencing some A and B co-occuring two times will not be simply double the association strength of experiencing them co-occurring once.

The model can be deconstructed into its two constituent networks, one of which learns words (as phoneme-strings), and the other which learns colors. Learned types (words or colors) are represented by a set of "concept nodes". Lower-level perception is represented by a three-dimensional feature-map on the color side, and as a set of nodes which react to different phonemes on the word side. The primary learning rule for associations based on co-occurrences is a non-linear Hebbian-type rule, which can be generally stated as:

$$w_{xy}^{t} = w_{xy}^{t-1} + S \cdot \left(sig(\alpha_x \alpha_y) \cdot M e^{Be^{C w_{xy}^{t-1}}} - (w_{xy}^{t-1} \cdot D) \right)$$

where S is a scaler variable[3] which scales with the magnitudes of the activations of nodes x and y, w_{xy} represents the association weight between nodes x and y, and α_x is the output activation of node x. sig is a sigmoid soft-thresholding function[4]. The doubly-exponentiated term is an application of a Gompertz function to the previous weight, and produces a horizontally asymptotic exponential growth to the weight based on the previous magnitude of the weight, but on a logrithmic scale. This is the primary factor causing the non-linear incremental learning described earlier. The coefficients B (-4) and C (-0.66) modulate the growth rate and scale of the function, and M provides the upper bound. D is a decay constant (0.02).

A. THE COLOR NETWORK

The color network is the simpler of the two. It is composed of a three-dimensional feature map (with each dimension accounting for one dimension of a color in RGB format), with each node connected to concept nodes. Color concept nodes are created when appropriate. In the case of our

[2]The SDL hook is made available by Can Kavaklıoğlu at `http://cankavaklioglu.name.tr/shul.html`.

[3]$S = sig(\alpha_x + \alpha_y)$

[4]$sig(x) = \frac{1}{1+e^{-\lambda(x-C)}}$, C is the displacement determining where the soft threshold is set, λ (6.0, as tested)is a parameter determining how quickly (steeply) the function grows to its asymptote.

composite system, this is when a sufficiently new word-color pair is experienced. Input comes into the color network in the form of activation given to nodes in the color map. Nodes to be activated are determined by finding the node in the map which minimizes the distance between its feature weights (i.e. R, G, B values) and the RGB values of the incoming color. Nodes surrounding the winning node then receive smaller amounts of activations based on their distance from the winner.

For these experiments, we use a feature map of size $10 \times 10 \times 10$, with each node in the feature map initialized to evenly-spaced places between 0 and 255 in each dimension. We find this as an acceptable compromise between speed and space concerns, while maintaining a sufficiently fine granularity for discriminating different colors.

After a winner has been selected, activation of all other nodes is set:

$$\forall_{xyz \in CN}, \alpha_{xyz} = e^{\frac{-d^2_{xyz}}{2\sigma^2}}$$

α_{xyz} is the activation of node with coordinates x, y, z in the feature map. σ is a parameter, which we have set to 0.8. CN refers to the RGB feature map. d is a *distance function* which determines the distance between the winner and the node in question, and is defined as:

$$d_{xyz} = \frac{\sqrt{(r_{xyz} - r_{win})^2 + (g_{xyz} - g_{win})^2 + (b_{xyz} - b_{win})^2}}{d_{max}}$$

where r, g, b are the feature value of subscripted node to that feature (r for red, g for green, b for blue), and d_{max} is a constant, maximum distance used to normalize the values into the desired range of activation [0, 1], which is determined by the size of the color feature network, among other things:

$$d_{max} = \sqrt{\frac{255^2}{10} \cdot 3}$$

10 is the width of our network in a given dimension (i.e. there are ten nodes), and 3 is the number of dimensions.

Energy then flows from the winning node to any connected concept nodes, modulated by the weight of that connection, i.e.

$$\forall_{n \in WN}, \alpha^t_n = \alpha^{t-1}_n + \alpha_{win} \cdot w_{win,n}$$

where, again, α is the activation of the subscripted node at the superscripted time (where time is measured by the number of sounds heard since input began).

B. THE WORD NETWORK

The word network is a recurrent network, with a layer representing the reactive activations to experienced aural input, and a recurrent layer representing the activations of the previously experienced input. These two layers are connected to the (word) concept layer via an array of soft-thresholding interneuron nodes. This network effectively recognizes words based on the sequence of phonemes they contain. It does this by incrementally pooling activation into word concept nodes as phoneme-strings contained in that word-concept are experienced.

The network is assumed to know when a word ends and a word begins. Sounds enter the system as deconstructed probabilities, one for each phoneme. These are the probabilities that the uttered sound was an instance of that phoneme. Thus, if a perfect /a/ is uttered, the probability for the /a/ phoneme will be very close to one, while others will be close to zero. In the case of more ambiguous sounds, such as /b/ and /p/, it may be that each receives relatively high activation, especially if the environment is noisy.

The network is instantiated with the full score of these "phoneme nodes", one representing each salient phoneme present in the language and dialect. Input thus enters the system and induces an activation in each of those phoneme nodes. There is also an equally sized recurrent layer of phoneme-nodes, which hold the activation of the phoneme layer from the previous sound. The activation spreads along efferent links to arrays of "interneurons" (N for each word node, where N is the number of phonemes). Each phoneme node is connected to one interneuron node in each word cluster, and each recurrent phoneme is connected to *all* interneuron nodes. These interneuron nodes apply a soft threshold to their activation (using a sigmoid as in footnote 4) and pass that output on to a layer of "word nodes". The only weights which are currently trained are those of the connections between the recurrent phoneme layer (nodes notated p_o), and the interneuron layer. This weight is updated using a Hebbian rule based on three values. For a recurrent layer node p_o and an interneuron node i, the weight between p_o and i will update according to the following rule:

$$w^t_{p_o,i} = w^{t-1}_{p_o,i} + \left(S \cdot \eta \cdot sig(\alpha_i \alpha_{p_o} \alpha_{p_n}) - w^{t-1}_{p_o,i}\right)$$

η is a learning rate (0.1), and S is (again) a scaler to reduce undesired decay of weights when the connected nodes are not activated, calculated as:

$$S = 0.01 + sig(\alpha_i + \alpha_{p_n} + \alpha_{p_o})$$

where p_n is the normal phoneme node corresponding to the recurrent phoneme in question p_o.

Throughout a word experience, there is a rising threshold value which is applied to the activations of the word nodes. This threshold depends on the number of sounds experienced since the word began. The output activation of word nodes (as used in learning rules involving links between word nodes to other nodes) is also determined by the application of a sigmoidal soft-thresholding function based on this threshold. When a word ends, a winner is chosen from among the word nodes based on highest activation. If the activation of this winner surpasses the threshold, then the experienced sequence of sounds is considered recognized. If it does not, then the experienced sound sequence does not sufficiently match any remembered words, and so a new word node is added to the network (trained on the sound sequence).

C. THE INTEGRATED SYSTEM

We integrated the system by removing the color concept nodes and simply connecting the word nodes directly to the RGB feature map. This is justified since the color modality

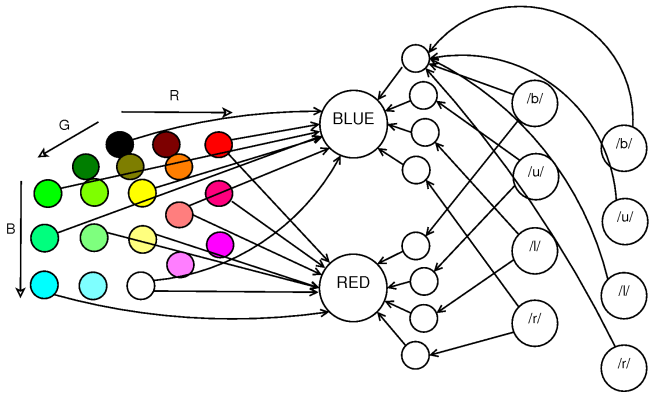

Fig. 1. The integrated word-learning system. On the left is the 3-d feature map representing red, green, blue dimensions of RGB color space. The layered network on the right is the recurrent network recognizing phoneme-sequences. Not all links are shown (only the top interneuron node of the top word node has all of its afferent connections displayed).

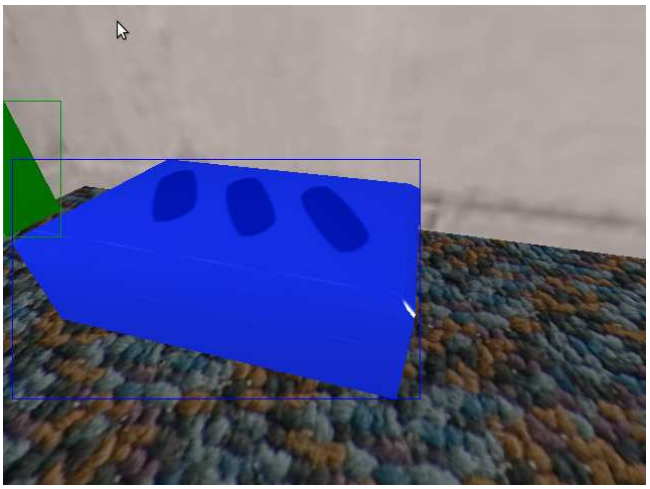

Fig. 2. Blob detector's view in the simulator of a typical bluebox situation.

is in a sense supervised by the word modality. It only will carve out an area of color space and associate it with a word when a word is presented.

The easiest way to understand how the composite system functions is to observe that the word-learning side will behave as it would alone, except that it will receive *additional* activation energy from the visual modality. This energy flows from feature map space, representing those colors which were previously experienced simultaneous to a word being experienced.

This additional information will have two interesting effects we would like to focus on. First, it will make word-recognition more robust, providing sufficient activation to a word-node if the associated color is present, even in situations where all of the contained phonemes were not sufficiently recognized to exceed the threshold for recognizing that word (a form of "perceptual co-modulation"). Second, since the learning rules contain terms representing the activations of the respective connected nodes, higher word node activations will result in higher connection weights. which translates to stronger memories. Stronger memories will be forgotten more slowly and will be easier to recall (recognize) later than memories of words which were less strongly associated with visual color experiences.

D. SETTING UP THE SYSTEM FOR EXPERIMENTS

For the testing and experiments presented in this paper, the system was set up to receive the necessary information from both its modalities in a relatively simultaneous fashion. The system is implemented as a C++ library, with functions which allow the system to be fed input (causing updates to the internal networks based on that input). To integrate this C++ library with the ADE framework, it was necessary to create an ADE server which calls the library. JNA was used to allow access to the C++ library's necessary function calls from Java.

The color map side was then updated every 150ms by querying the vision server for detected color blobs, their RGB values, and their areas. The area was normalized into the range [0,1]. This value was used to determine how much to activate a winning RGB map node. Phoneme information was fed from pre-configured files representing the output the phoneme-deconstruction program described above would produce on perfect or near-perfect input. These files were fed into the system on cue, with one "sound" fed every 150ms.

For the simulated training sessions, the agent was navigated to a block of the appropriate color and arranged so that the block took up a significant portion of the visual field ($> 55\%$). In the real world training sessions a block of the appropriate color was placed in front of the robot so that it took up a similar portion of the visual field. For testing, after the system had been trained, it was shown real-world objects. To do this with a simulation-trained agent, it was necessary to disconnect the ADE server running the network from the vision server in the simulator, and reconnect to a vision server connected to the real-world robot. This was accomplished using ADE's recovery abilities–by simply killing the simulation server and having the ADE system recover by hooking into a server connected to the real-world robot.

V. MODEL VALIDATION: COMPARISON OF SIMULATION AND ROBOT DATA

Our evaluation tests are intended to examine how closely matched the behavior of the model is between the simulated environment and the real world. The learning task involved showing the robot a colored object during 10 presentations of a 4-phoneme word. Each phoneme is presented for 150ms, and there is a random (1–5s) time interval between presentations. Although the camera was allowed to move between training runs, it was held stationary while training. To prevent possible speech recognition confounds, the phoneme input was presented in perfect form so as to elicit the maximal recognition probability for each phoneme. The target object was a blue box in both the real world and USARSim. Fig. 2 shows the robot's view of the simulated box, while Fig. 3 shows a frame from a real-world training session.

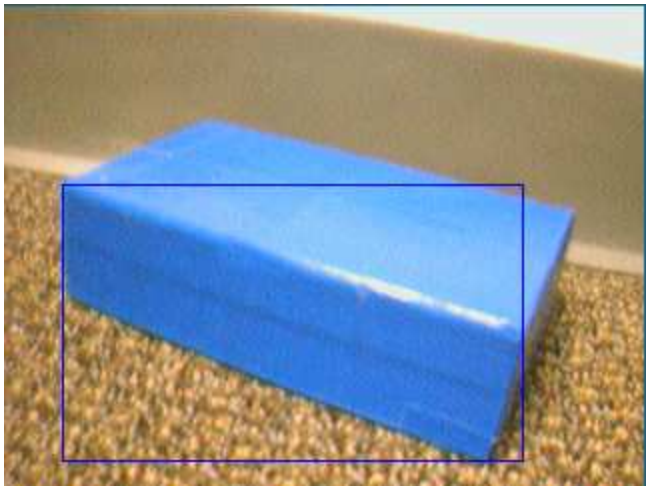

Fig. 3. Blob detector's view in the real world of a typical bluebox situation.

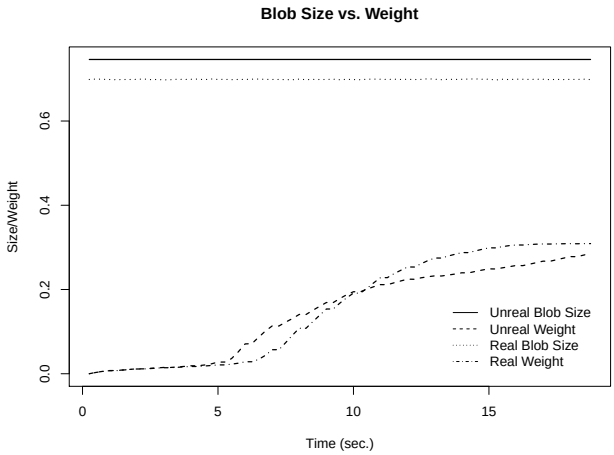

Fig. 4. Plot of blob size and color-word association weight over time

The training results are presented in Fig. 4. These results are averaged over multiple runs (16 in Unreal, 21 real-world). The results demonstrate that the learning behavior of the system is comparable in the simulator and in the real world. Because the camera is held stationary, there is little change in the average blob size as the training sessions progress. However, note the horizontally asymptotic exponential growth behavior of the weight between the color and the associated word node, caused by the learning rule proposed above. The regular "dips" in association strength are actually an effect of the implementation of word-recognition in the system. They occur because the initial phoneme of a word will not cause any major activation in a word node (since it is not a sequence). Thus, at these times, the word and color may actually be *de*-associated, but only by a small amount.

Regarding the real-virtual comparison, the growth of the weights is very strongly correlated in the two environments (Pearson's $r = .982, p < .001$). There are minor differences in the overall pattern of growth, caused by the difference in average blob size and a greater blob size variance in the simulated environment. However, the learning is clearly very similar in the two environments. Moreover, as an additional validation, we tested the "recognition" ability of the trained systems in a real-world test. That is, we performed the training as described above, then after a short interval (2s) to allow the system to settle (due to decay), we presented input from the *real* camera to both those trained in the real world and those trained in USARSim. When presented with the blue blob input, without any phoneme input, real-world trained systems' word activation rose from an average of .0001 to .88, while the USARSim trained systems' average activation rose from .0001 to .77. That is, the activation of the word associated with the color increased substantially in virtue of being presented with that color. The increases are quite similar, as confirmed by the lack of a significant difference found by the t-test ($t = -1.1346$, $p = 0.2677$).

It was difficult to get the blob sizes in the real world versus the simulated tests to match up exactly, and this explains some of the small differences in the growth rate data. In the controlled situations used in the experiments, and with only pertinent aspects of the network extracted and reflected in the results presented above, the effects of noise in the blob-detection algorithm are not obvious. However, this is justified. More so than visual noise or failures of the blob-detection algorithm (which would often result in additional, superfluous blobs being detected), it was differences in the size of the blob that adversely affected the results. Since the blob size has a doubly-exponential effect on the learning rate of the system, we see an odd comparison in our data. On the simulated side, blob size ranged over a larger variance of values then in the real world, where control over the positioning of the camera is not so coarse. This variance resulted in a more awkwardly-shaped graph (i.e. differently shaped than the curve that would be generated by the learning algorithm discussed above) for the simulated side, since functions with drastically different growths and shapes are being averaged together with no account for the blob size, an exponent. These small differences sometimes had the effect of the system falling on two sides of a bifurcation, resulting in relatively close blob sizes causing two drastically different learning curves–one which grew to maximum within the allotted time, and another which did not even come close to doing so.

It is also important to recognize that the differences observed in our results are *not* due to noise in the real world and the lack thereof in the simulated world. Even if the simulated image was somehow intentionally degredated to account for noise, the effect would be more failures of the blob detection algorithm in ambiguous or noisy visual situations, which we endeavored to avoid in the experimental setup presented in this paper and which are not reflected in the results. It is entirely possible, however, that given a different experiment (for example, one that examined the robustness of recognition, as is discussed below) the differences in noise in the real world versus the simulated one would be salient in explaining the differences.

VI. CONCLUSION AND FUTURE WORK

This paper presented our integration of a vision-based word learning architecture with the USARSim simulation environment. The existing vision component of the ADE robotic architecture infrastructure was extended to allow input from Unreal, allowing us to use the simulated environment as a drop-in replacement for real-world testing. We demonstrated the validity of the integration by comparing tests of the learning mechanism conducted using the simulator with tests conducted in the real world. The results showed that the training progressed comparably in the two environments. Moreover, on the real world color recognition task, the system performed similarly regardless of whether it was trained in the simulated environment or in the real world.

Future work will involve further development of the word learning system presented in this paper in multiple directions. For these experiments, parameter values and even function types were chosen using trial-and-error. In future work, a genetic algorithm may be used to optimize these parameters for the task at hand. The complexity of the word-learning side of the network leaves a lot to be desired, only recognizing contained phoneme-sequences. Modifications to the network will also be performed to allow for other effects on word-recognition, such as a phoneme being present in a word. Eventually, we hope to match the word-learning behavior of the system to data from real-world children's word-learning behaviors. Observations from that domain may give important clues towards optimizing the parameters and learning functions to match that behavior. Finally, an exciting next step will be to move from only recognition to also initating action. Hearing a word or seeing a color could actually cause the agent to initiate action, perhaps a refocusing of attention. We expect this to result in an interesting dynamical feedback loop, as attention focus on an object brings it larger in to the visual field, activations for that will increase, causing faster learning to take place. The fast learning would in turn cause higher activations the next time around, resulting in more probable, stronger attention shift towards that object. Research suggest that children may also learn words in this fashion [9].

VII. ACKNOWLEDGMENTS

This work was in part funded by ONR MURI grant #N00014-07-1-1049 to second author. Thanks to You-Wei Cheah for his work on the USARSim integration into the ADE vision server.

REFERENCES

[1] J.R. Anderson, D. Bothell, M.D. Byrne, and C. Lebiere. An integrated theory of the mind. *Psychological Review*, 11:1036–1060, 2004.

[2] Steven Balakirsky, Chris Scrapper, Stefano Carpin, and Michael Lewis. Usarsim: Providing a framework for multi-robot performance evaluation. In *Proceedings of PerMIS*, 2006.

[3] Randall Beer. Dynamical approaches to cognitive science. *Trends in Cognitive Science*, 4(3):91–99, 2000.

[4] Jaime Carbonell, Oren Etzioni, Yolanda Gil, Robert Joseph, Craig Knoblock, Steve Minton, and Manuela Veloso. Prodigy: an integrated architecture for planning and learning. *SIGART Bull.*, 2(4):51–55, 1991.

[5] Kyle Carter, Matthias Scheutz, and Paul Schermerhorn. A humanoid-robotic replica in usarsim for hri experiments. In *IROS Workshop on Robots, Games, and Research*, 2009 (under review).

[6] George Kachergis, Chen Yu, and Richard Shiffrin. Temporal contiguity in cross-situation statistical learning. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, 2009.

[7] John Laird, Allen Newell, and Paul Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64, 1987.

[8] D.E. Meyer and D.E. Kieras. A computational theory of executive cognitive processes and multiple-task performance. part 1. *Psychological Review*, 104(1):3–65, 1997.

[9] Larissa Samuelson and Linda B Smith. Memory and attention make smart word learning: An alternative account of akhtar, carpenter and tomasello. *Child Development*, 69:94–104, 1998.

[10] Paul Schermerhorn and Matthias Scheutz. Natural language interactions in distributed networks of smart devices. *International Journal of Semantic Computing*, 2(4):503–524, 2008.

[11] Matthias Scheutz. ADE - steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied Artificial Intelligence*, 20(4-5):275–304, 2006.

[12] Richard Shiffrin and Mark Steyvers. A model for recognition memory: REM–retrieving effectively from memory. *Psychonomic Bulletin and Review*, 4(2):145–166, 1997.

[13] Joshua B. Tenenbaum and Fei Xu. Word learning as bayesian inference. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, 2000.

[14] J. Trafton, N. Cassimatis, M. Bugajska, D. Brock, F. Mintz, and A. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man and Cybernetics*, 25(4):460–470, 2005.

[15] Chen Yu and Linda B. Smith. Rapid word learning under uncertainty via cross-situational statistics. *Psychological Science*, 18:414–420, 2007.