Advanced Robotics Projects for Undergraduate Students

Douglas Blank
Computer Science Program
Bryn Mawr College
Bryn Mawr, PA 19010
dblank@cs.brynmawr.edu

Deepak Kumar
Computer Science Program
Bryn Mawr College
Bryn Mawr, PA 19010
dkumar@cs.brynmawr.edu

James Marshall
Computer Science Department
Sarah Lawrence College
Bronxville, NY 10708
jmarshall@slc.edu

Lisa Meeden
Computer Science Department
Swarthmore College
Swarthmore, PA 19081
meeden@cs.swarthmore.edu

Abstract

The benefits to using robots in the artificial intelligence and robotics classrooms are now fairly well established. However, most projects demonstrated so far are fairly simple. In this paper we explore advanced robotics projects that have been (or could be) successfully implemented by undergraduate students in a one-semester or two-semester course. We explore what makes a good undergraduate advanced project, why such advanced projects are now possible, give example projects, and discuss the benefits of such projects.

Introduction

In order to enable students to attempt ambitious robotics projects within an academic semester, it is essential that they are provided with a high-level framework for communicating with physical robots and simulations as well as for constructing robot control architectures. Pyro is one such tool that we have used extensively at our institutions to enable advanced artificial intelligence (AI) and robotics projects (Blank *et al.* 2006). There are other Robotics Control Frameworks available, including Player/Stage (Vaughan, Gerkey, & Howard 2003), ARIA (MobileRobots.com 2005), CARMEN (Montemerlo, Roy, & Thrun 2003), MARIE (Cote 2005), and most-recently the Microsoft Robotics Studio (Microsoft.com 2006). This paper will explore Pyro; however, many of its properties for enabling advanced robotics projects may also apply to other Robotics Control Frameworks.[1]

There are a number of key features that make Pyro especially useful for implementing student projects. First, the majority of Pyro is written in Python, which includes a small set of very useful high-level tools for handling dictionaries, lists, strings, and file manipulation that ease the programmer's burden. Students who already know at least one programming language can quickly learn Python while also learning about robotics. Second, Pyro includes a large collection of machine learning tools, such as a neural networks, reinforcement learning, and emergent models of computation, that students can readily incorporate into their own projects. Third, Pyro uses the object-oriented paradigm allowing students to easily extend existing classes to create specializations of their own design. Fourth, the same Pyro brain can be executed on a simulator or a real robot. This allows students to efficiently explore possibilities within simulation, and only move to the physical robot when a good possibility for success has been discovered.

Finally, there are a large number of additional Python libraries being developed and made available through open source, free software, and shared source licenses. Many of these libraries include AI-specific tools, such as natural language processing. Although there are many open source/free software libraries available, many suffer "bitrot" and can no longer be compiled or run. This often occurs because of some associated interface has changed and the code is no longer being actively maintained. Python-shared code is often not as prone to bitrot because Python changes slowly and is usually backwards-compatible with previous versions.

Python's ease-of-use combined with Pyro's abstractions and useful libraries creates a fertile environment with which to spark student's interests. However, the successful advanced project requires additional direction. The following section explores several successful student projects.

Advanced Projects

Our undergraduate students have attempted a variety of different types of advanced projects over the last decade. These projects fall into four main types: emulations, replications, extensions, and original research. Emulation projects capture the essence of a hard task, but cleverly sidestep currently intractable problems. Replication projects take an existing model from the primary literature, re-implement it, and then attempt to replicate its described behavior. Extension projects also take an existing model, but try to extend that model in interesting ways, perhaps by testing it in a new domain, or by adding additional features to overcome known issues. Finally, original research projects are the rarest, and involve creating something new.

What makes a successful student project? Of course, students may learn (and even enjoy) a project that is never com-

[1]For a comprehensive comparison of many Robotics Control Frameworks, see (Kramer & Scheutz forthcoming).

```
Person: Help.
Robot:  How can I help you?
Person: Room.
Robot:  What room would you like directions to?
Person: Two Three Two.
Robot:  You said room Two Three Two, is this correct?
Person: Yes.
Robot:  You are at the Front Door. Go to the T Junction and turn
        left, go past Women's Restroom, go past Men's Restroom,
        go past Room 247, go past Room 233. Finally you will
        reach Room 232.
```

Figure 1: Transcript of a spoken interaction with the Tour Guide Robot. From (Chiu 2004)

pleted, or one that works incorrectly. However, for the purposes of this paper we will focus on the projects which are successfully completed. We have found that such a project often begins with a clever idea that exhibits a core AI or machine learning concept. The key is to help the student focus on the essential aspects of the project. This section will examine a series of projects of these types and explore the aspects of Pyro that made them possible.

Greeted by the future: A Tour Guide Robot

A good example of an emulation project is the tour guide robot created by three undergraduate students at Bryn Mawr College. The students designed a robot control program that gave tours as well as providing directions to various locations in the science building at Bryn Mawr College (Butoi, Chiu, & Thompson 2004b; 2004a; Chiu 2004). The project was entirely conceived by the students after observing several campus tours given to prospective students. The students formulated the project and obtained funding from the Computing Research Association's (CRA) CREU program to carry out this task. They used the MobileRobots' PeopleBot robot to implement their project. As a background, the students studied the design and implementations of several similar earlier projects: Rhino (Buhmann *et al.* 1995; Burgard *et al.* 1998), Minerva (Thrun *et al.* 1999), Alfred (Maxwell *et al.* 1999), Grace (Simmons *et al.* 2003), and Virgil (Thrapp, Westbrook, & Subramanian 2001).

The robot control program they designed gave tours of three hallways of the science building and provided directions to any location on the second floor of the building. The tour lasted 10 minutes (about the length of a typical tour to prospective students) and was delivered using sections of pre-scripted speech and a pre-planned path. Figure 1 shows a transcript of a spoken interaction with the robot for giving directions:

The tour guide robot used a finite-state machine control architecture and its main modules included: navigation, localization, and speech understanding and generation. Navigation and localization were accomplished by using a combination of obstacle avoidance, dead-reckoning, and vision-based (color-match filtering) landmark detection synchronized with the pre-planned paths and tour-text.

The robot's performance was fairly robust, but the students did not approach the problem in the traditional manner. For example, rather than attempt a global localization, the students pointed the camera at the floor and looked for a series of colored tape. In this manner, the robot merely headed in a general direction avoiding obstacles while looking for the tape. Although the mechanism was simplistic, the effect was dramatic. In this manner, they only emulated more sophisticated tour guide-giving robots. However, by building on what they knew, they were able to create an original and entertaining demonstration.

The entire project was implemented in Pyro (Blank *et al.* 2006). They used the Festival Server for speech generation and the Sphinx system for speech recognition (Black & Taylor 2003; Lenzo 2005), both of which were wrapped in Python. An A* algorithm was used for computing and provising directions.

Developmental Robotics

An upper-level seminar-style course on developmental robotics is taught every other year at Swarthmore College (Meeden 2006), and has resulted in a number of good examples of replication projects and extension projects.

Developmental robotics is a recently formed interdisciplinary field that is inspired by the recognition that complex biological organisms are the result of an extended developmental process (Meeden & Blank 2006). It is argued that to create equally robust and interesting artificial entities may depend on modeling fundamental aspects of these developmental processes from biology. One example of developmental robotics model from the primary literature is Intelligent Adaptive Curiosity (IAC) (Yves Oudeyer & Kaplan 2005).

The goal of the IAC model is to create a control mechanism for a robot where the complexity of its activities increases and a developmental sequence arises without being manually constructed. The key idea is to use an intrinsic motivational system that strives to maximize learning progress. This is accomplished by forming a memory of all perception and action pairs the robot has experienced. This memory is divided into regions of similar contexts. Each region has an associated expert that tries to predict the next sensory state given the current sensory state and the proposed action. The learning progress of each expert is monitored, and the IAC model focuses the robot's attention towards regions where the most learning progress is occurring. In this way, the robot can first learn simple associations. Once these are understood, the learning progress will slow in these regions and the robot will begin to focus on more complex associations.

For their midterm project, which was completed in three weeks, the students were provided with a prototype of the IAC model written in Pyro. This prototype consisted primarily of three object-oriented classes representing the IAC brain, memory, and regions. They were asked to design an experiment for the IAC model, run the experiment, analyze the results, and produce a paper.

Some students tried to replicate the original model as closely as possible. Re-implementing the original work raised a number of detailed questions that could not be an-

swered from the published papers. The class contacted the authors to determine how to proceed, and were eventually successful in replicating the main results, although in simulation rather than on a physical robot. This process of replication allowed the students to come to a much deeper understanding of the model and its limitations.

Other students attempted to improve on the model by replacing the original memory, which split regions simply based on size, to a self-organizing memory that split regions based on the uniqueness of the exemplars. Because each component of the IAC implementation was a class, it was easy for students to swap in a new component. This extension proved to be quite successful and improved the model's performance and efficiency.

In the second half of the semester, students proposed a final project, which was completed in seven weeks. For this project students could propose their own model, replicate an existing model, or extend an existing model. In addition to producing a final paper (many of which are available on-line at the course web site (Meeden 2006), students were also required to give 30 minute presentation to the class on their project.

The final projects tend to be quite strong due to the fact that all of the students have gained experience doing the smaller and more focused midterm projects. The use of Pyro makes it possible to complete two substantial projects within a single semester.

Evolutionary Robotics

Designing complex robot behaviors is an arduous endeavor. However, evolving complex robot behaviors can be much more rewarding, if not as time consuming. One methodology that we have used for exploring Evolutionary Robotics is to combine a genetic algorithm with a feed-forward neural network and simulated robots. Thus, a gene is composed only of the weights of a neural network. The neural network receives sensor values as input and produces motor commands as output. A population of these genes can be evolved to perform a variety of tasks by simply defining a particular fitness function. Pyro allows the integration of evolutionary algorithms, neural networks, and robotics as each is represented by an object with well-defined interfaces.

Many projects can be fit into this paradigm, and students can easily create a program which can evolve to perform interesting, non-intuitive solutions to the task. It is nearly trivial to construct a fitness function to evolve simulated robots that, say, seek out light. However, a very interesting variation was presented by (Marocco & Nolfi 2006) at ALife X. In this variation, a group of robots are instantiated, all with the same gene. The robots' tasks are to "feed" on two light sources; however, only two robots can feed at each light (see Figure 2). Although the robots cannot see each other, they are equipped with ability to make sounds and to hear. Each robot has a set of directional microphones, and the ability to create a noise on each time step. As the robots evolve, they begin to use the ability to exchange information via their audio signals. In a word, they evolve a language, meaning, and associated behaviors.

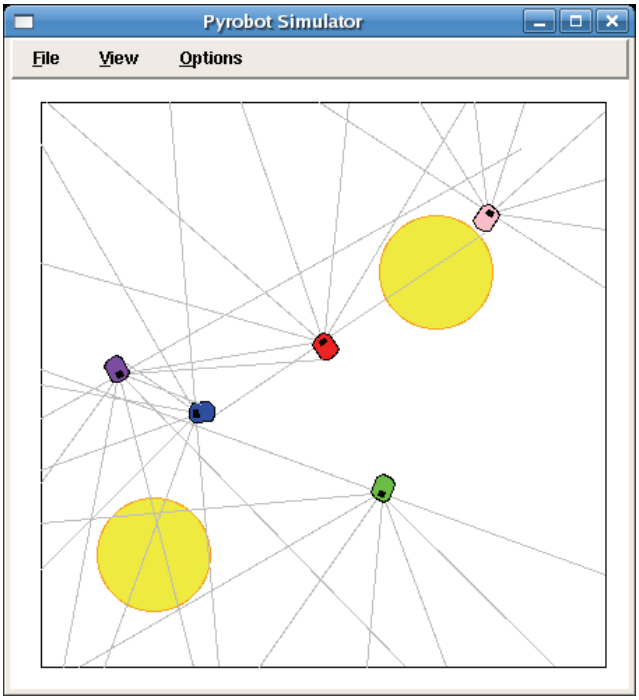

Figure 2: An experiment in evolutionary robotics. Pyro's simulator is shown evolving a set of five genetically-identical robots that develop a language. Based on (Marocco & Nolfi 2006).

Although we have not used this example with students yet, this will surely be an exciting replication project. In addition, clever students will no doubt have ideas for extensions.

Natural Language Interaction with Robots

Computer-based natural language interaction is an active area of research in Computational Linguistics and AI. While there have been several natural language systems built for specific computer applications, natural language interaction with robots remains largely unexplored. This senior thesis (for a double major in Linguistics and Computer Science) focuses on implementing a natural language interpreter for commands and queries given to a small mobile robot (Walker 2007). The goal is to implement a complete system for natural language understanding in this domain, and as such consists of three main parts: a system for parsing a subset of English our robot is to understand, a semantic analyzer used to extract meaning from the natural language, and a first-order logic engine, used by the robot for storage and deduction of facts. The semantic analyzer additionally implements the operational semantics of robot behaviors so the robot is also able to carry out the requests expressed in natural language. Using such a system we will be able to demonstrate that a mobile robot is capable of understanding

natural language commands and queries and responding to them appropriately.

The main tools being used in this project include the Scribbler robot (Parallax Corp. 2006), the Python-based natural language toolkit, NLTK (Loper & Bird 2002; Bird & Loper 2004; Bird 2005), and a Python-based robot programming environment, Myro (Balch *et al.* 2007) which is a descendant of Pyro (Blank *et al.* 2006). NLTK provides facilities for parsing context-free grammars and for creating successful parse trees. However, NLTK does not include a semantic analyzer. This project involves developing one and integrating it into NLTK. Additionally, for the first-order logic knowledge representation and reasoning module, there is no complete, general-purpose module available for doing simple *tell-ask* interactions. This will also be developed and integrated into NLTK. The Scribbler robot is a new platform and has never been used for such an exercise. Thus this project embodies a combination of replication, development, software integration, and original work. Effectively carrying out the goals of this project will be yet another demonstration of Python (and Myro's) capabilities for extensibility.

Bubbles: An Exploration into Aerial Robotics

Although we are computer scientists, we occasionally encounter students who are interested in building their own robot. As this can be quite a time investment and we can provide little guidance, we often try to redirect the student's interest to using the Handyboard and Legos. However, recently we at Bryn Mawr College did relent and ventured into the physical world of robot engineering. We did this for three reasons. First, we had a team of students and faculty (including one from Physics) willing to put time towards the project throughout a two-month period. Secondly, we were encouraged in multiple ways by Paul Oh and the Indoor Aerial Robot Competition (Oh 2006). Finally, and possibly most importantly, we were able to build on the infrastructure of Pyro. For example, although controlling a blimp in three dimensions can be seen as a very different control problem from that of the traditional mobile robot, the team decided to use Pyro's main 2D control system, and add an additional mechanism for dealing with height.

Three students from Bryn Mawr and Swarthmore Colleges took "Bubbles" to the 2006 AAAI annual conference to compete in the Scavenger Hunt competition. Although the robot did not perform the task (it was just too breezy in the conference center) the students did win a Technical Achievement Award for their original hardware design.

Neural Networks

In addition to reimplementing past projects, students have also performed original research. As a case in point, two of the authors (Marshall and Blank) collaborated on such a project with a Pomona College student during the summer of 2005. This work, which led to a conference publication coauthored by the student (Blank, Lewis, & Marshall 2005), addressed an important open question in the field of developmental robotics. During the following academic year, the student further extended this work for his senior thesis project.

This research grew out of an attempt to describe a general architecture for developmental robotics based on the key ideas of prediction, abstraction, and self-motivation (Blank *et al.* 2005). To make sense of its environment, a robot must be able to form abstractions in order to focus attention on the most relevant aspects of its sensory stream; it must learn to predict how the environment changes; and its behavior must be driven by internally generated motivations. Related work explored one approach for creating such a self-motivated robot (Marshall, Blank, & Meeden 2004), in which a neural network continually attempts to predict the behavior of the environment on successive time steps, using the robot's actual observations as the training feedback signal for learning.

One question that arises is what happens if some part of the environment is inherently unpredictable and hence unlearnable? For example, we would not want the robot to spend all of its time attempting to learn to predict the behavior of curtains blowing in the wind, or of random static on a TV screen. Like humans, robots need to be able to distinguish between tasks that are learnable and those that are inherently unlearnable.

We decided to study this question in the context of a simple neural network training task, designed to capture the essential features of the problem. The training data, which modeled the "environment" of a robot, consisted of a set of binary patterns, some of which had fixed (and hence predictable) associations, while others were unpredictable. The portion of unpredictable patterns in the data set represented "noise" in the environment, within which the actual task to be learned was embedded (represented by the predictable patterns). By changing the ratio of predictable to unpredictable patterns, we could vary the difficulty of the learning task. Furthermore, we developed a novel type of network that learned to predict not only the patterns of the data set, but also the network's own internal representations of those patterns.

Pyro's built-in support for neural networks made it easy to construct such self-predicting network architectures. Our student researcher implemented the code and ran the experiments over a period of a few weeks, even though he had no previous experience with Python or Pyro (he had, however, already taken undergraduate courses in programming, AI, and neural networks). His results were surprising and significant, and showed that self-prediction can lead to faster and more robust learning, especially in noisy environments. Figure 3 shows a graph summarizing the performance of three different network architectures on a data set consisting of 25% predictable and 75% unpredictable patterns. The self-predicting networks are consistently better than standard networks at learning the predictable portion of the data set. This work would not have been possible at the undergraduate level without Pyro's sophisticated abstractions and

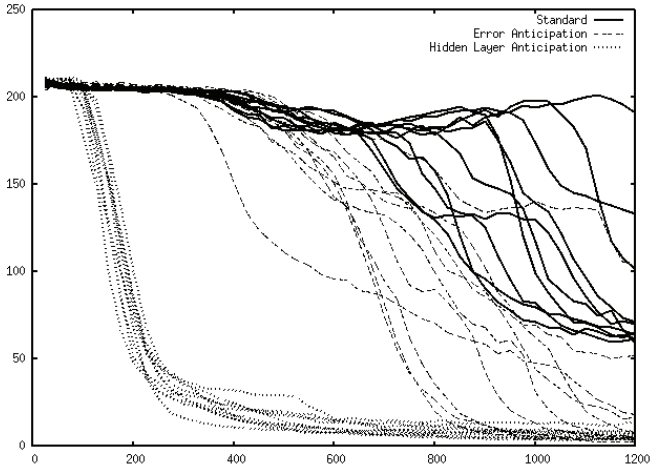

Figure 3: Performance of self-predicting networks compared to standard backprop on a data set of 25% predictable and 75% unpredictable patterns. Y-axis plots sum-squared error for the predictable patterns; x-axis plots training time. Ten separate runs are shown for each type of network.

capabilities.

Conclusion

By helping students select interesting, yet focused ideas, we have been able to assist undergraduates in successfully implementing a wide variety of robotics projects. No doubt, Python and Pyro have made this possible. We hope to continue the development and exploration of such tools, and look forward to future generations of students and their advanced robotics projects.

We would like to thank all of our students over the years for taking on projects such as these. Without such energetic and ambitious students, we would have nothing on which to report.

References

Balch, T.; Blank, D.; Guzdial, M.; and Kumar, D. 2007. IPRE: Institute for Personal Robotics in Education (www.roboteducation.org).

Bird, S., and Loper, E. 2004. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL demonstration session*, 214–217. Association for Computational Linguistics.

Bird, S. 2005. NLTK-Lite: Efficient scripting for natural language processing. In *Proceedings of the 4th International Conference on Natural Language Processing (ICON)*, 11–18. Allied Publishers.

Black, A., and Taylor, P. 2003. *Festival Speech Synthesis System: system documentation(1.1.1)*. http://www.cstr.ed.ac.uk/projects/festival.

Blank, D.; Kumar, D.; Meeden, L.; and Marshall, J. 2005. Bringing up robot: fundamental mechanisms for creating a self-motivated, self-organizing architecture. *Cybernetics and Systems* 36(2):125–150.

Blank, D.; Kumar, D.; Meeden, L.; and Yanco, H. 2006. The pyro toolkit for AI and robotics. *AI Magazine* 27(1):39–50.

Blank, D.; Lewis, J.; and Marshall, J. 2005. The multiple roles of anticipation in developmental robotics. In *AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems*. Menlo Park, CA: AAAI Press.

Buhmann, J.; Burgard, W.; Cremers, A.; Fox, D.; Hofmann, T.; Schneider, F.; Strikos, J.; and Thrun, S. 1995. The Mobile Robot RHINO. *AI Magazine* 16(2):31–38.

Burgard, W.; Cremers, A. B.; Fox, D.; Hahnel, D.; and Lakemeyer, G. 1998. The Interactive Museum Tour-Guide Robot. In *Proceedings of the AAAI 1998*, 11–18. AAAI Press.

Butoi, I.; Chiu, C.; and Thompson, D. 2004a. Greeted by the Future: A Tour Guide Robot, available at www.cra.org/Activities/craw/creu/crewReports/2004.

Butoi, I.; Chiu, C.; and Thompson, D. 2004b. Greeted by the Future: A Tour Guide Robot, available at www.cs.brynmawr.edu/TourGuide.

Chiu, C. 2004. The Bryn Mawr Tour Guide Robot. Senior Thesis available at www.cs.brynmawr.edu/TourGuide/thesis.pdf.

Cote, C. 2005. Mobile and autonomous robotics integration environment (MARIE).

Kramer, J., and Scheutz, M. forthcoming. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*.

Lenzo, K. 2005. *Sphinx Documentation*. http://fife.speech.cs.cmu.edu/sphinx/.

Loper, E., and Bird, S. 2002. NLTK: The Natural Language Toolkit. In *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 262–269. Association for Computational Linguistics.

Marocco, D., and Nolfi, S. 2006. Self-organization of communication in evolving robots. In Rocha, L. M.; Yaeger, L. S.; Bedau, M. A.; Floreano, D.; Goldstone, R. L.; and Vespignani, A., eds., *Artificial Life X: Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*. MIT Press.

Marshall, J.; Blank, D.; and Meeden, L. 2004. An emergent framework for self-motivation in developmental robotics. In *Proceedings of the Third International Conference on Development and Learning (ICDL 2004)*, 104–111. La Jolla, CA: Salk Institute for Biological Studies.

Maxwell, B. A.; Meeden, L. A.; Addo, N.; Brown, L.; Dickson, P.; Ng, J.; Olshfski, S.; Silk, E.; and Wales, J. 1999. Alfred: The Robot Waiter Who Remembers You. *AI Magazine*.

Meeden, L., and Blank, D. 2006. Introduction to developmental robotics. *Connection Science* 18(2):93–96.

Meeden, L. 2006. Cs81 developmental robotics syllabus. www.cs.swarthmore.edu/ meeden/cs81/s06/cs81.html.

Microsoft.com. 2006. Microsoft robotics studio.

MobileRobots.com. 2005. ActivMedia robotics MobileRobots developer support.

Montemerlo, M.; Roy, N.; and Thrun, S. 2003. CARMEN, Carnegie Mellon Robot Navigation Toolkit.

Oh, P. 2006. http://www.pages.drexel.edu/ vn43/main/ IARC2006/IARC2006.html.

Parallax Corp. 2006. The Scribbler Robot, http://www.scribblerrobot.com.

Simmons, R.; Coldberg, D.; Goode, A.; Montemerlo, M.; Roy, N.; Sellner, B.; Urmson, C.; Schultz, A.; Abramson, M.; Adams, W.; Atrash, A.; Bugajska, M.; Coblenz, M.; MacMahon, M.; Perzanowski, D.; Horswill, I.; Zubek, R.; Kortenkamp, D.; Wolfe, B.; Milam, T.; and Maxwell, B. 2003. GRACE: An Autonomous Robot for the AAAI Robot Challenge. *AI Magazine*.

Thrapp, R.; Westbrook, C.; and Subramanian, D. 2001. Robust Localization Algorithm for an Autnomous Campus Tour. In *Proceedings of ICRA 2001*.

Thrun, S.; Bennewitz, M.; Burgard, W.; Cremers, A. B.; Dellaert, F.; Fox, D.; Hahnel, D.; Rosenberg, C.; Roy, N.; Schulte, J.; and Schulz, D. 1999. MINERVA: A Second-Generation Museum Tour-Guide Robot. In *Proceedings of ICRA 1999*.

Vaughan, R.; Gerkey, B.; and Howard, A. 2003. On device abstractions for portable, reusable, robot code. In *Proceedings of IROS*, 2121–2427.

Walker, A. 2007. Natural Language Interaction with Robots, http://wiki.cs.brynmawr.edu/?page= Natural Language Interaction with Robots.

Yves Oudeyer, P., and Kaplan, F. 2005. Intelligent adaptive curiosity: a source of self-development. In Berthouze, L.; Kozima, H.; Prince, C. G.; Sandini, G.; Stojanov, G.; Metta, G.; and Balkenius, C., eds., *Proceedings of the Fourth International Workshop on Epigenetic Robotics*.