Towards Self-Reflecting Machines: Two-Minds in One Robot

Juan Cristobal Zagal and Hod Lipson

Computational Synthesis Laboratory, Mechanical and Aerospace Engineering,
Cornell University, Ithaca, NY 14853, USA
{jcz35, hod.lipson}@cornell.edu

Abstract. We introduce a technique that allows a robot to increase its resiliency and learning skills by exploiting a process akin to self-reflection. A robot contains two controllers: A pure reactive *innate* controller, and a *reflective* controller that can observe, model and control the *innate* controller. The reflective controller adapts the innate controller without access to the innate controller's internal state or architecture; Instead, it models it and then synthesizes filters that exploit its existing capabilities for new situations. In this paper we explore a number of scenarios where the innate controller is a recurrent neural network. We demonstrate significant adaptation ability with relatively few physical trials.

Keywords: Self-modeling, self-reflection, machine learning, evolutionary robotics.

1 Introduction

The process of self-reflection underlies much of humans' and primates' ability to adapt to vastly varying conditions with little or no physical experimentation. In this work, we are interested in exploring robotic systems that use similar processes to

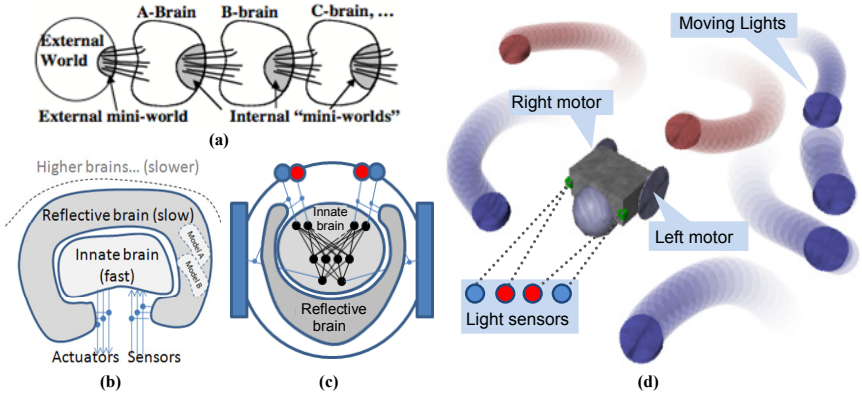


Fig. 1. (a) Minsky's brain chain, from [7]. (b) A proposal of nested brains architecture. (c) The robot contains an innate brain, and a reflective brain which can model and modulate the I/O of the innate brain. (d) Schematic of robot and environment consisting of moving sources of blue and red light.

model their own thinking, and then learn how to change their behavior with little recourse to physical trials or hand-crafted simulators. Understanding self-reflective processes may help make more robust adaptive systems that operate in rapidly changing physical environments, and shed light on these processes in nature [8].

In previous work we have explored how a real robot enhances its adaptation by exploiting self-models of its own morphology [4]. The robot was able to continuously improve its internal models while exploiting propioceptive data collected during its functioning. These models were used to explore new compensatory behaviors. The results suggested that a robot might be more resilient when self-modeling certain source of perturbation, gaining the capabilities of self-representation, prediction, and aided exploration of the source.

Here we wish to take this concept a step further, by having a robot model its own *controller* as well. Just as a robot benefits from modeling its own morphology and then using that model to determine how to best compensate for a new situation, can a robot benefit from modeling its own *controller*, then use it to compensate for a new situations?

The first question to answer is why a robot would need to model its own controller at all, instead of directly accessing and manipulating it. The reasons for this are many fold: First, there are many aspects of a control system that cannot be easily modeled, even if its architecture is perfectly known: Sensor and actuation lag time, noise, and computational errors and delays, for example. Second, the controller may change in unanticipated ways due to failure or change in the environment. Modeling an existing controller also takes time and effort, and direct manipulation of a controller could require an unwarranted increase in software and hardware complexity. Finally, in some cases the controller of a robot is simply inaccessible – either locked by design, or obfuscated by legacy code. The ability to modify performance of an existing controller without directly accessing it also serves as a safe adaption strategy, since the original controller is never modified and therefore its behavior can be restored at any time. This process may also shed light on the evolution of more opaque controllers such as biological nervous systems.

The approach we use here is based on the assumption that there are two controllers; one reflecting on the other, in the same way that metacognition is the ability to reflect upon one's own mental processes and to self-regulate them. Such metacognitive processes are recognized to be present in humans, non-human primates, and a few other mammals [3,10]. It has been recently demonstrated to exist in the rat as well [2], suggesting that it might be applicable to simpler systems such as robots.

The theoretical framework for human metacognition was initially laid out by Nelson and Narens [9]. They proposed that mental activity occurs at a higher *meta* (reflective) level and at a lower *object* (innate) level. The reflective level contains simulations of the object level and interacts by means of two information streams: monitoring and control. Monitoring is the process of observing the processing of the object level and control is the process of modifying the object level. A thought experiment in metacognition was proposed by Minsky [6]. Minsky suggested dividing an artificial brain in two parts. While the input-outputs of the first part (A-brain) are connected to the external world, the second part (B-brain) is only connected to the A-brain; thus A is the only world seen by B (Figure 1a). As proposed by Minsky, the B-brain might help to the A-brain even without having access to the real world, and by

just looking at the activity of the A-brain. Simple questions such as *Are you repeating*? *Are you feeling better?* And *How do you think?* might help to produce a better brain state in the world.

Other studies have examined metacognition in computation [1] with a focus on monitoring a set of variables that are known to be relevant to a specific problem solver or a learning process. An example of this is adjusting learning coefficients, such as weights, during a machine learning task.

We started our investigation in machine self-reflection [11] with the architecture presented in Figure 1b. A simulated wheeled robot (Figure 1c,d) was driven by a pure reactive innate brain. We showed how self-models generated by the reflective brain were useful to produce compensatory resilient behaviors when filtering the innate outputs. In this paper we investigate self-reflection upon time varying causal innate controllers implemented with recurrent neural networks (RNN). We also investigate the effect of filtering the inputs and outputs of the innate controller and the effect of adding time correlated noise to the outputs during monitoring.

The remainder of this paper is as follows: Section 2 describes the innate causal controller. Section 3 describes the process of self-modeling causal controllers using recurrent neural networks. Section 4 presents results on the synthesis of input and output modifiers as well as the effect of adding different amounts of noise to the innate controller outputs. Section 5 contains the conclusions of this investigation.

2 Innate Causal Brain

First, we synthesized an innate robot controller under a highly dynamic environment that contains sources of blue and red light moving in random circular patterns. We evolved the controller such that the robot seeks blue light while avoiding red light. A RNN implemented a controller that mapped the robot sensor inputs to motor outputs, as shown in Figure 2F, referred to as the *Innate-NN*. Four input neurons are fed by the light sensors z^k: $k = \{0...3\}$. The network contains two hidden nodes and two output nodes that generate the left u_0 and right u_1 motor signals. The output y_k of neuron k is computed as

$$y_k = \phi\left(\sum_j w_{kj} x_j - \theta_k \right) \quad \textbf{(1)}$$

where $\phi(\cdot)$ is the sigmoid activation function, x_j are the input signals, w_{kj} are the connection weights and θ_k is the threshold of neuron k. The controller is represented by a genome i of $N_i = 33$ scalar parameters (in the range $[-1, 1]$): 24 connection weights, 8 activation thresholds, and one motor scaling factor α. The reward perceived by a robot is defined in equation (2) by assigning a positive (negative) reward to the amount of blue (red) light intensity that is collected during the evaluation of controller i under environment e after a period of T time steps.

$$F_t^{e,i} = \int_0^t (z_t^{0,e,i} - z_t^{1,e,i} + z_t^{2,e,i} - z_t^{3,e,i})\, dt \quad \textbf{(2)} \qquad F^i = \prod_{e=1}^{N_E} F_T^{e,i} \quad \textbf{(3)}$$

To avoid exploiting the peculiarities of a unique environment, we used a set of N_e=3 randomly generated environments and we defined the fitness of a candidate controller as equation (3) indicates. Due to perceptual aliasing[1] an optimal motor action $U_t = \{u_t^0, u_t^1\}$ cannot be purely determined by the sensor state $Z_t = \{z_t^0,\ldots, z_t^3\}$ at a single time t. Thus, in order to provide predictive capabilities to the robot we have decided to use a time-dependent causal controller using a RNN. The algorithm runs with a probability of mutation $p_m = 1/N_i$ using Cauchy mutation and a probability of crossover $p_c = 0.9$. The population size is set to 30 individuals per generation.

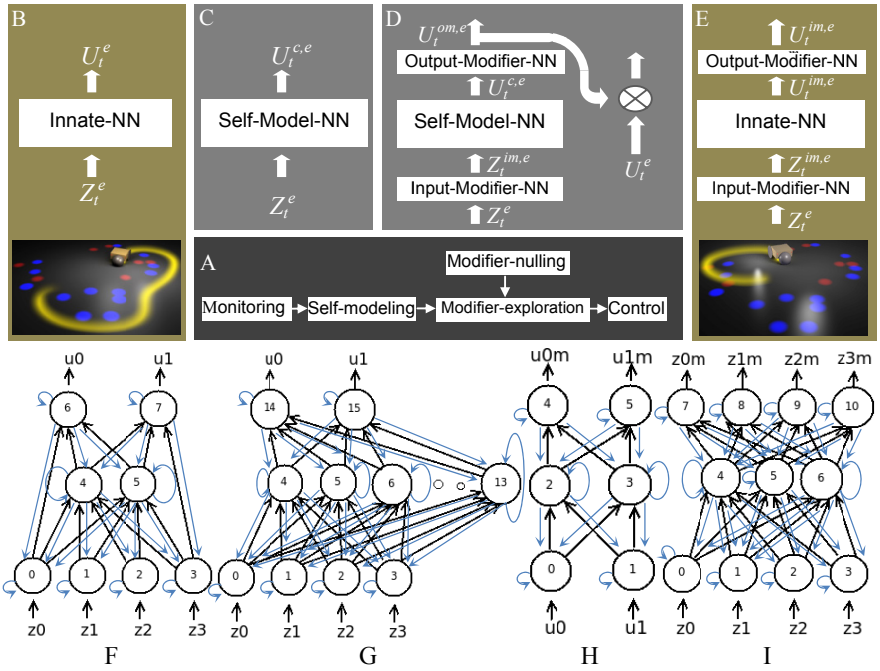


Fig. 2. Synthesis of the self-reflection process. A: The algorithm. B-E: Details of the main processing steps. F: The Innate-NN, four input nodes *{0, 1, 2, 3}* receive measurements z^k: $k = \{0 \ldots 3\}$, the network contains two hidden nodes *{4, 5}* and two output nodes *{6, 7}* generating the motor outputs u^k : $k = \{0, 1\}$. G: Self-model-NN, receives the same inputs and outputs as the innate neural architecture, however, there is a large number of hidden units allowing more complexity. The modifier NN's are in charge of modulating the inputs (I) and outputs (H) that are seen by the Innate-NN during the robot operation.

3 Self-Modeling Causal Innate Controllers

We approximate the innate controller with a candidate controller c implemented with a generic RNN controller (Self-model-NN) that has the same number of input and output nodes as the Innate-NN, though a much larger number of hidden units.

[1] Different locations trigger the same sensor state, albeit requiring different control actions.

Figure 2G show this network when considering 10 hidden units, 100 connection weights, 16 activation thresholds and one output scalar factor α.

In this self-modeling stage, let us also introduce an unanticipated environmental change. Suppose that the environment reward suddenly changes: Now the blue light is bad (like poison) and the red light is good. The innate controller is now unoptimal, and fitness reward is decreasing.

We allowed the robot to operate freely under an environment e executing its Innate-NN controller and we collected vector time series of sensor $Z^e = \{Z_t^e : t \in T\}$, motor $U^e = \{U_t^e : t \in T\}$ and reward $F = \{F_t : t \in T\}$. We fed the inputs of each candidate Self-model-NN controller c with the recorded time series of sensor data Z^e, resulting in a predicted motor actuation data $U^{c,e} = \{U_t^{c,e} : t \in T\}$ (Figure 2C). We then measured the quality of each candidate self-model controller c by its ability to reproduce the same input-output patterns as those observed during the operation of the Innate-NN controller on different environments. To find the best self-model, we minimized the distance $D(c)$ described by equation 4.

$$D(c) = \sum_{e} \int_0^T \left\| U_t^{c,e} - U_t^{e} \right\| dt \tag{4}$$

Figure 3 shows the convergence of self-models to innate behavior (in red). After 100000 generations self-model controllers are able to drive the robot very similarly as the innate controller does over the training scenario (a,c,d) and even predict the behavior of the innate controller under the test scenario (b,e,f). The similarity of motor commands (c,e) increases when minimizing the distance $D(c)$, resulting robot trajectory becomes closer to the innate trajectory (a,b). Examples of the convergence of resulting sensor channels are shown in (d,f).

4 Synthesis of Input-Output Modifiers

In this section we describe the synthesis of the input-output modifier RNN's that are illustrated in Figure 2H,I. During the controlling stage (Figure 2A) modifiers are in charge of filtering the innate sensor and motor signals. The challenge is how to train these modifier networks in such a way that (i) the reward of the robot goes up again and (ii) by only exploiting past experience, without any new trials.

The method is as follows: Given the time series of sensor Z^e and motor U^e data, we can estimate the quality of candidate modifiers by integrating the observed fitness variation ΔF_t, described in equation 5, *only* in while the modified controller motor action U_t is similar to the already recorded motor action U_t^e that was tested in reality at a given time t. We estimate this similarity by means of the binary function $\gamma_1(U_t)$ defined in equation 7 as function of a threshold $\beta = 0.06$ or by using the continuous function $\gamma_2(U_t)$ defined in equation 8.

Equation 6 shows the normalized fitness related to each candidate modifier. The confidence of the estimation is proportional to the integral across time of $\gamma(U_t)$ which represents either $\gamma_1(U_t)$ or $\gamma_2(U_t)$. Intuitively, the idea is to use the information of fitness variation that was already monitored during the robot operation.

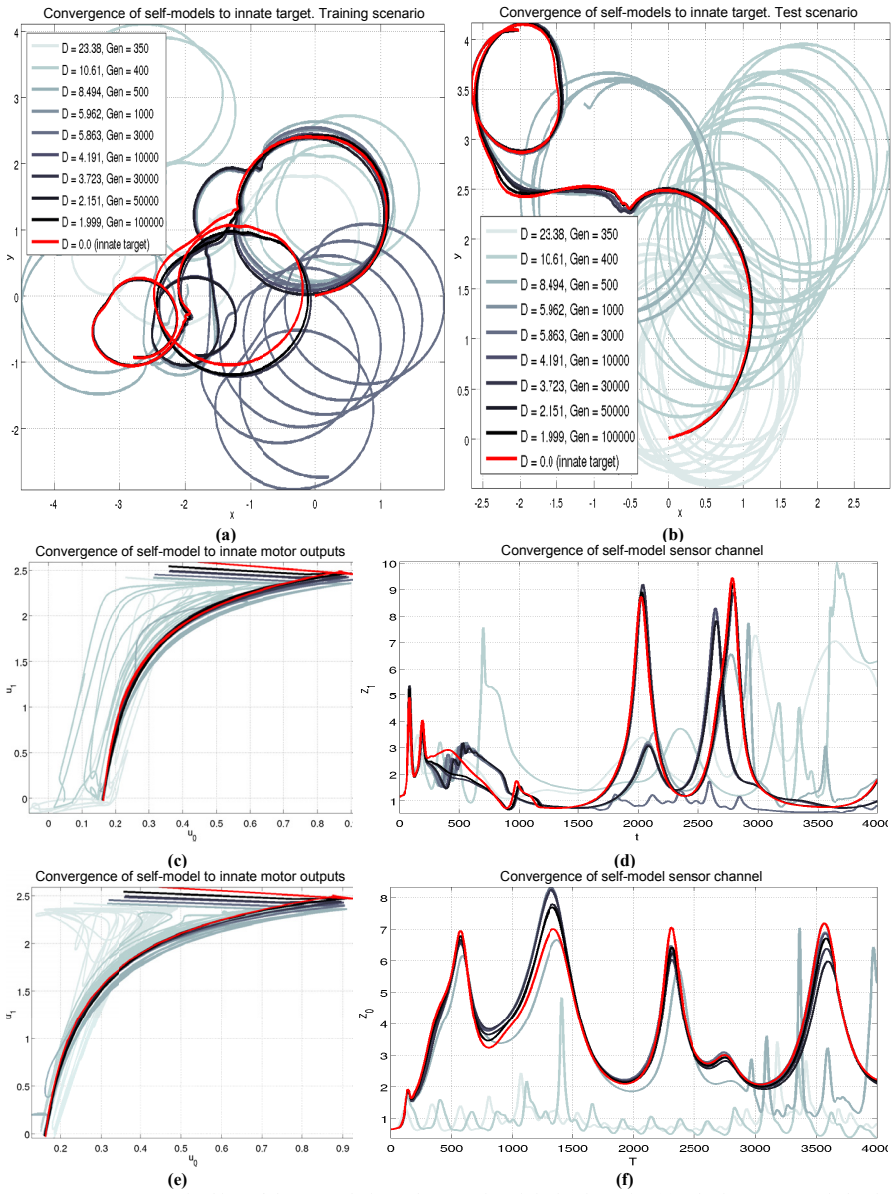


Fig. 3. Convergence of self-models as result from the genetic minimization of *D(c)*. The innate target behavior is shown in red. The self-model behavior is shown under training (a,c,d) and (b,e,f) test scenarios. The similarity of motor commands (c,e) increases when minimizing the distance *D(c)*, resulting robot trajectory becomes closer to the innate trajectory (a,b). Examples of the convergence of resulting sensor channels are shown in (d,f).

$$\Delta F_t = \frac{F_{t+\Delta t} - F_t}{\Delta t} \quad (5) \qquad \hat{f} = \frac{\int_0^T \gamma(U_t)\Delta F_t dt}{\int_0^T \gamma(U_t) dt} \quad (6)$$

$$\gamma_1(U_t) = \begin{cases} 1 & if \ \dfrac{\|U_t - U_t^e\|}{\|U_t\| + \|U_t^e\|} \le \beta \\ 0 & otherwise \end{cases} \quad (7) \qquad \gamma_2(U_t) = 1 - \frac{\|U_t - U_t^e\|}{\|U_t\| + \|U_t^e\|} \quad (8)$$

Figure 4 below shows activation patterns of $\gamma_1(U_t)$ (a,b) and $\gamma_2(U_t)$ (c) for candidate modifiers explored during self-reflection. The domain axes represent time during evaluation of the innate robot behavior in reality and time during self-reflection. The value of 1.0 (black in c) corresponds to maximum similarity while 0.0 (white in c) represent the highest discrepancy. Time segments showing increasing levels of similarity and high expected fitness are extracted during self-reflection (a). This process is enhanced when adding time correlated noise to the output of innate controllers during fitness monitoring (b). Results indicate that there is no *a priori* correlation between the activations of $\gamma(U_t)$ and the resulting fitness of a candidate.

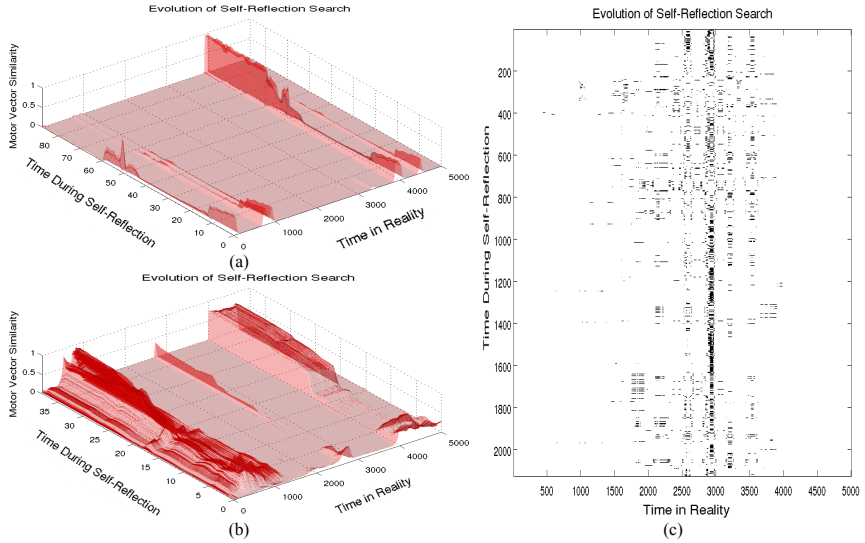


Fig. 4. Activations of $\gamma_1(U_t)$ (a,b) and $\gamma_2(U_t)$ (c). The value of 1.0 (black in c) corresponds to maximum similarity while 0.0 (white in c) represent the highest discrepancy. The effect of adding time correlated noise to the outputs of the innate controller during evaluation in reality is illustrated in (b). Binary activations illustrated in (c) are a good example of how different time structures are extracted during self-reflection.

We initialized the input/output modifiers such that their action is transparent at the beginning of self-reflection. We call this a *nulling* stage that consists on performing genetic search over the modifier search space until their output accurately mimic corresponding inputs. The solutions are then used as seeds for the actual modifier synthesis search process.

Table 1. Comparisons of self-reflection versus classical ER techniques. When the environment changes (blue is bad and red is good) the innate controller drives the robot to obtain very low fitness (-644). Recovery using self-reflection allows achieving a fitness of 56 after four hardware trials. A traditional innate recovery would have taken 40.000 trials to achieve similar results.

Design Option	Num. of physical trials	Resulting fitness
Output Modifier, self-ref.	*4 trials.*	56.2 ± 40
Output Modifier, classic ER.	*2000 trials.*	30.4 ± 55
Input Modifier, self-ref.	*4 trials.*	35.8 ± 66
Input + Output Modifier, self-ref.	*10 trials.*	60.3 ± 123
Innate formation.	*150.000 trials.*	-644.3 ± 52
Innate recovery, classical ER.	*40.000 trials.*	73.7 ± 80

5 Conclusions

We conclude that: (1) Accurate self-models of RNN innate controllers can be obtained using the proposed method. (2) Resulting self-models allow prediction of the robot innate behavior even under environments whose dynamics differs from the training scenario. (3) Self-reflection allow our robot to recover quickly; investing 4 or 10 hardware trials to achieve a level of fitness that otherwise would have taken thousands of trials with traditional ER techniques. (4) There is an advantage of inducing motor noise while monitoring the innate robot behavior. (5) We notice the need of pre-calibration of the modifiers prior self-reflection. The large amount of hardware trials is one of the main limitations of ER. Self-reflection appears as a promising alternative to expand the domain of applications of this field.

Acknowledgements

This work has been supported in part by the U.S. National Science Foundation (NSF) Creative-IT program, grant #0757478 and Chilean research FONDECYT project number #3080048.

References

1. Cox, M.T.: Metacognition in Computation: A selected research review. Artificial Intelligence 169(2):104-141 (2005)
2. Foote, A., Crystal, J.: Metacognition in the Rat. Current Biology, 17(6):551–555 (2007)
3. R. Hampton. Rhesus monkeys know when they remember. Proceedings of the National Academy of Sciences, 98(9):5359 (2001)
4. Bongard, J.C., Zykov, V., Lipson, H.: Resilient Machines through Continuous Self-Modeling. Science, 314(5802), 1118—1121 (2006)
5. Brooks, R.A: Intelligence Without Representation, Artificial Intelligence 47(1-3), 139—159 (1991)
6. Minsky, M.: The Society of Mind. Simon & Chuster Inc. New York, NY (1986)
7. Minsky, M.: Interior grounding, reflection, and self-consciousness. In: Proceedings of International Conference on Brain, Mind and Society. Tohoku University, Japan. (2005)
8. Minsky M.: The Emotion Machine, Simon & Schuster Inc. New York, NY (2007)
9. Nelson, T., Narens, L.: Metamemory: A theoretical framework and new findings. The psychology of learning and motivation, 26:125–141, (1990)
10. Smith, J., Shields, W., Washburn., D.: The comparative psychology of uncertainty monitoring and metacognition. Behavioral and Brain Sciences, 26(03):317–339 (2004)
11. Zagal, J.C., Lipson, H.: Self-Reflection in Evolutionary Robotics: Resilient Adaptation with a Minimum of Physical Exploration. In: Proceedings of the Genetic and Evolutionary Computation Conference, Late Breaking Papers, GECCO (2009)