Resilient Behavior through Controller Self-Diagnosis, Adaptation and Recovery

Juan Cristobal Zagal
Computational Synthesis Laboratory
Mechanical & Aerospace Engineering
Cornell University
Ithaca, NY 14853, USA
jcz35@cornell.edu

Hod Lipson
Computational Synthesis Laboratory
Mechanical & Aerospace Engineering
Cornell University
Ithaca, NY 14853, USA
hod.lipson@cornell.edu

ABSTRACT
We explore robot behavior recovery through a process akin to self-reflection. A robot contains two controllers: A primary "innate" reactive controller, and a secondary "reflective" controller that can observe, model and control the primary controller. The reflective controller adapts the innate controller without access to the innate controller's internal state or architecture. Instead, the reflective controller models the innate controller and then synthesizes input/output filters that adapt the innate controller's existing capabilities to new situations. The innate controller is subjected to a variety of sensory, motor, and internal control damage scenarios. The reflective controller diagnoses the level of failure using a self-model and the observed sensorimotor time-series data and is able to recover performance.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**]: Learning; I.2.9 **[Artificial Intelligence]:** Robotics---*Autonomous vehicles*

General Terms
Algorithms

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

1. INTRODUCTION
Living systems are able to maintain their performance while compensating for changes in their environment and in their own structure [16]. Here we explore processes that allow machines to model their own behavior and use those models to detect controller failure and identify paths to recovery. Such ability is known to exist in the fish and reptiles [6][21] and it has been recently discovered in some structures of the adult mammalian brain [1][9][10][7] at the scale of changing and reorganizing neurons and their function by experience.

PerMIS'09, September 21-23, 2009, Gaithersburg, MD, USA.

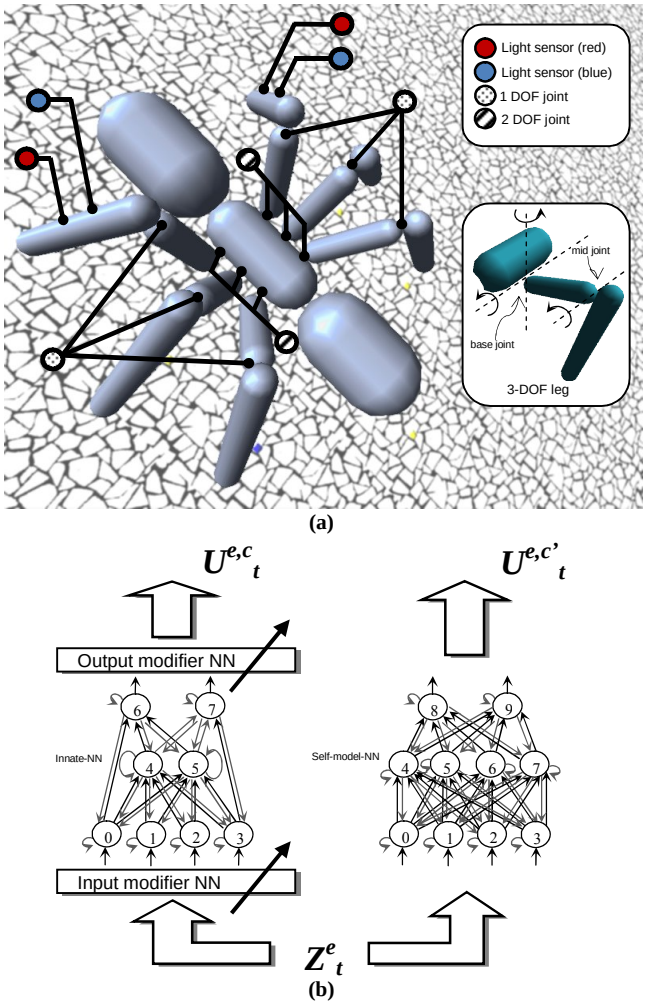


Figure 1. (a) Six-legged robot used for the experiments. The robot perceives light intensity by means of color specific sensors (red and blue) located at either side of its front legs. Each leg is independently controlled by three motors, two for the 2-DOF base joint and one for the 1-DOF mid joint. The robot is driven by 18 motors. A total of 15 rigid bodies complete the robot architecture. (b) The function recovery is achieved by the synthesis of modifier networks. A self-model (right) of the innate controller (left) is used to synthesize these modifiers.

In recent studies, we showed that a robot's resiliency increases when self-modeling its own morphology [3][4][18]. We explored how self-models of a robot body can be used for damage recovery [2]. 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 explicitly described, 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. Manual modeling of 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.

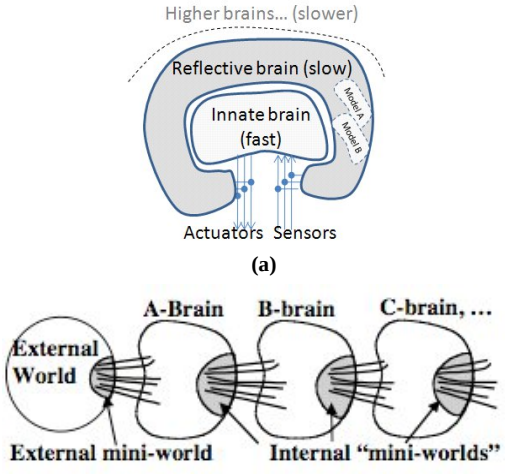


Figure 2. (a) A proposal of nested brains architecture. (b) Minsky's brain chain from [12].

The approach we use here is based on the assumption that there are two co-resident controllers; one reflecting on the other (Figure 2a). This architecture is a form of *metacognition*: 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 [8][15]. It has been recently demonstrated to exist in the rat as well [5], suggesting that metacognition concepts might be applicable to simpler systems such as robots.

Minsky [11] points out that a brain can be better understood as a "society of minds" interacting with each other. He proposed a thought experiment consisting on 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 2b). 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.

As pointed out by Minsky in recent work [13] there must be some brains that critique the performance of other brains or sets of brains. They might also be able to identify certain ways of thinking and to reconfigure thinking states by activating or reconnecting certain brain areas. If we are about to explore how to implement such a system using current robots the first question that arises is *how a critic system might identify and manipulate a certain way of thinking?* We hypothesize that minds should be able to perform some sort of self-modeling of other minds as a way to compare and reason about patterns of activation.

The remainder of this paper is organized as follows (Figure 8 presents a description of the entire process): In section 2 we describe the simulated robot and experimental environment used for our study. We also describe the generation of an innate robot behavior. In section 3, we present experiments showing the first stage of self-reflection by reverse engineering of the robot innate controller. In section 4, we describe different types of controller damage used for the experiments. In section 5 we describe how the self-model is used for damage diagnosis. In section 6 we describe the process of damage recovery through the synthesis of input/output modifiers. Finally in section 7 we present the conclusions of this work.

2. ROBOT AND ENVIRONMENT

We used a simulated six-legged robot for our experiments. The robot is free to walk on its environment, comprising a plane surface covered with moving emitters of red and blue light (see Figure 4); these light sources are initially randomly distributed along the surface and follow different patterns of motion. The robot architecture is illustrated in Figure 1a. The central portion of the robot is composed by three solid cylinders. The central cylinder is connected to six legs by means of 2-DOF motors. The forward walking behavior is the result of the rhythmic oscillation of the limbs. Each motor $i=\{1,\ldots,18\}$ follows a reference signal $r^i = \theta_i + a_i sin(\omega t + \phi_i)$, where θ_i is a pre-defined central angle of oscillation for motor i. PID dynamic compensators are in charge of ensuring successful reference following for each motor. These pattern generators were obtained by evolving the amplitude a_i and relative phase ϕ_i of each oscillator such that the robot maximizes frontal displacement. A similar strategy was reported in [17]. Figure 3 shows comparisons of the reference signals given by the oscillators versus the actual angle which is achieved by each motor during a normal walking of the robot along an observation period of duration $T=1500$ time steps.

Two pairs of color-specific light sensors are located at the front legs of the robot, generating the measurement signals z^0 (blue) and z^1 (red) from sensors located at the left leg, and signals z^2 (red) and z^3 (blue) from sensors located at the right leg. At each simulation step, the read-out of a light sensor z^k is computed as the instantaneous light intensity at the sensor due to contributions of all the environment light sources l_i of corresponding color using and inverse-square law.

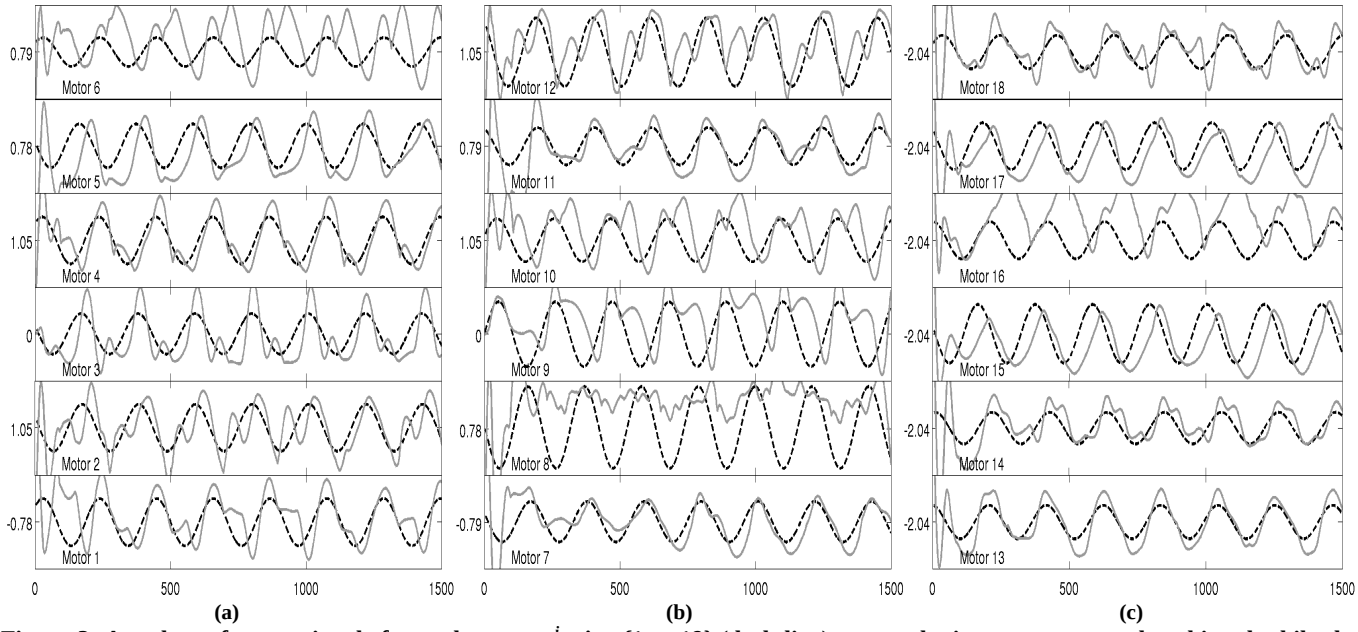


Figure 3. Angular reference signals for each motor r^i : $i = \{1,\ldots,18\}$ (dash line) versus the instantaneous angle achieved while the robot walks (gray solid line). The central angle of oscillation is presented in radians at the left of each plot. In (a,b) the reference following for motors attached to the robot torso are illustrated. In (c) the reference following for the femur-tibia joints are shown.

Once the robot is provided with the forward-moving behavior, it learns to follows the moving sources of blue light while avoiding the sources of red light. This behavior was obtained by evolving the weights of an "innate" recurrent neural network (RNN) controller (Innate-NN) shown on Figure 1b (left). Four input neurons {0, 1, 2, 3} are fed by the four light-measurement signals. The network contains two hidden nodes {4, 5} and two output nodes {6, 7} that generate the left u_0 and right u_1 motor modulation signals. The signal u_0 modulates the amplitude of the left legs oscillators {1,2,3,7,8,9,13,14,15} and u_1 the amplitude the remaining right leg oscillators {4,5,6,10,11,12,16,17,18}. The output y_k of neuron k is computed as

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

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 c of $N_c = 34$ scalar parameters (in the range [−1, 1]): 26 connection weights, and 8 activation thresholds. 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 c under environment e after a period of T time steps.

$$F_t^{e,c} = \int_0^t (z_t^{0,e,c} - z_t^{1,e,c} + z_t^{2,e,c} - z_t^{3,e,c})\, dt \qquad (2)$$

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 c as follows:

$$F^c = \prod_{e=1}^{N_E} f_T^{e,c} \qquad (3)$$

Due to perceptual aliasing (where different locations trigger the same sensor state, albeit requiring different control actions) 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; however, we hypothesize that a causal controller (such as a RNN) might have good overall performance over the evaluation period T.

The algorithm runs with a probability of mutation $p_m = 1/N_c$ using Cauchy mutation and a probability of crossover $p_c = 0.9$. The population size was set to 30 individuals per generation. Figure 4 shows the innate behavior that result after about 1000 evaluations.

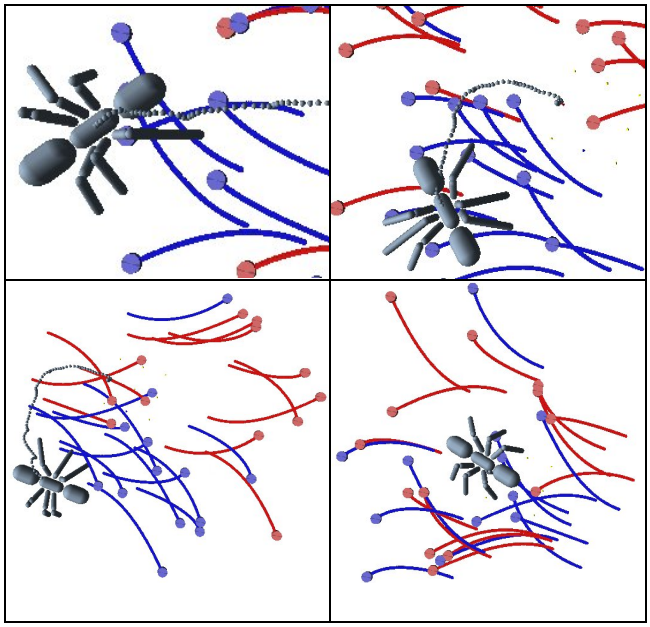


Figure 4. Illustration of the innate behavior. The robot moves toward areas of higher intensity of blue light while avoiding red lights (the discs are emitters of colored light). The figure shows four different scenarios. Motion traces are represented with the same color of the source.

3. SELF-MODELING CONTROLLER

In this section we describe the process by which the reflective controller acquires a self-model of the innate controller. Going back to Minsky's formulation, this is equivalent to the B-brain asking the A-brain about its way of thinking. The self-model is represented as a recurrent neural network controller (Self-model-NN) that has the same number of input and output nodes as the Innate-NN, though not necessarily the same number of hidden units. Figure 1 (right) presents the architecture of the self-model that we use for our experiments (contains 4 hidden nodes). It is represented with a genome c' of $N_{c'} = 62$ scalar parameters (in the range [−1, 1]): 52 connection weights and 10 activation thresholds.

We also considered the special case of a self-model (Self-model-NN_T) that has the same architecture of the Innate-NN. This is for the sake of testing some hypotheses described in the next section. We note corresponding test genome as c_T'.

We allowed the robot to operate freely under an environment e while executing its Innate-NN controller and we recorded vector time series of sensor data $Z^e = \{Z_t^e : t \in T\}$ and motor data $U^e = \{U_t^e : t \in T\}$. We fed the inputs of each candidate Self-model-NN controller c' with the recorded time series of sensor data Z^e, resulting in predicted motor actuation data $U^{c',e} = \{U_t^{c',e} : t \in T\}$. 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 in different environments. To find the best self-model, we minimized the signal distance $D(c)$ described by equation (4).

$$D(c) = \sum_e \int_0^T \left\| U_t^{c',e} - U_t^e \right\| dt \qquad (4)$$

Figure 5 shows results from optimizing the self-model, using a genetic algorithm, with the settings described in previous section. The search is steered toward minimizing the distance $D(c)$ over the space of candidate self-model controllers. The figure corresponds to an average of eight runs of the minimization procedure.

The standard error is depicted with vertical error bars. In order to avoid overfitting, a 30% of the data was used for validation. The minimization stops when the error in the validation data starts to increase (early stopping). The convergence of candidate self-models is illustrated on Figure 7. The figure shows how the similarity of 3D trajectories increases when minimizing the distance $D(c)$ over the training environment (a) and over different test environments (b,c). As it can be seen from the figure, resulting self-models can predict the robot performance under unseen environments to a large extend.

We note as c'^* the optimal self-model solution obtained during this stage.

4. CONTROLLER DAMAGE

In this section we describe the different scenarios of damage introduced to the innate controller for testing the here explored recovery method. Figure 6 shows the Innate-NN under six different types of controller damage.

The scenarios are summarized on Table 1. They consists on disconnecting different inputs to the network, disconnecting synaptic links inside the network, swapping synaptic links and also introducing constant perturbations on specific neurons.

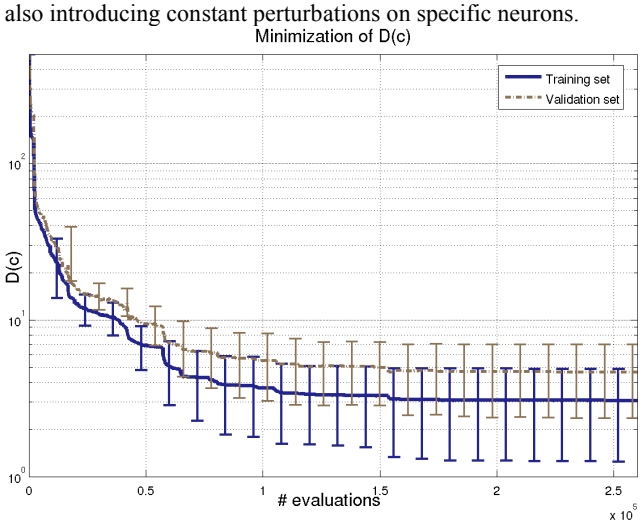


Figure 5. Results of minimizing the distance *D(c)* using genetic search over the space of self-models of the robot controller. Blue continuous line corresponds to the minimum distance achieved at corresponding evaluation. Burlywood line shows results obtained over validation data set. Standard error is depicted with error bars.

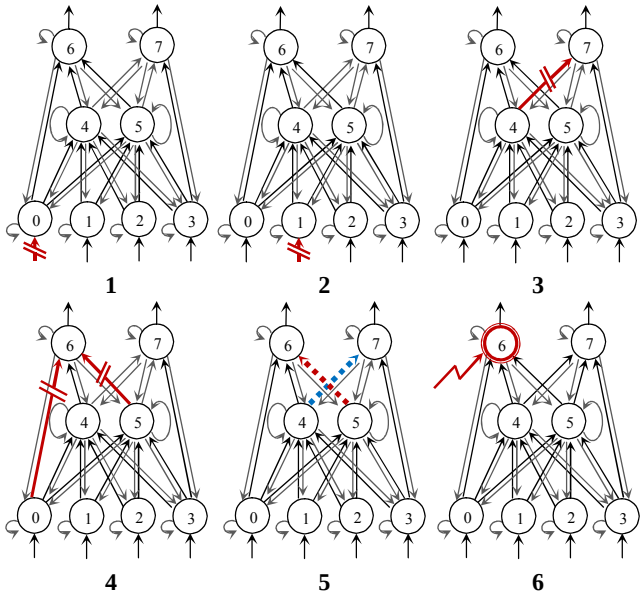
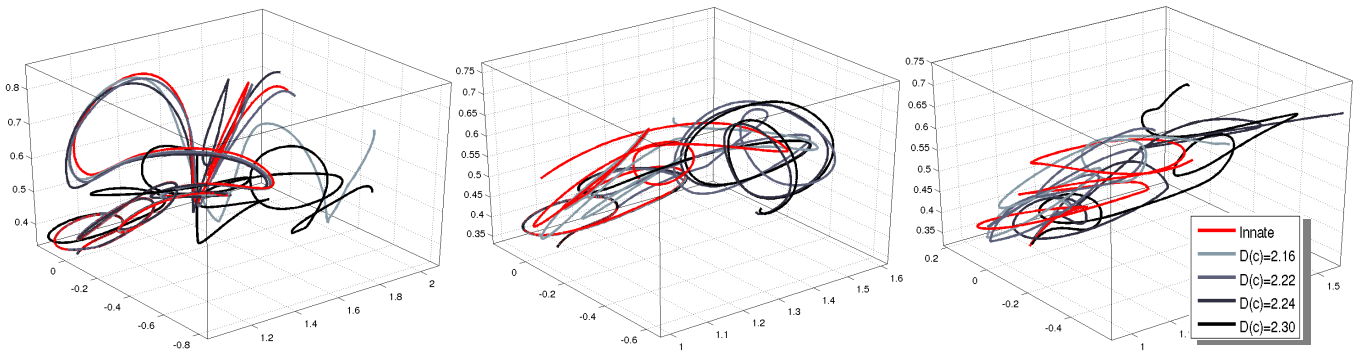


Figure 6. Different damage scenarios introduced to the Innate-NN controller for testing. Red or blue colors are used to indicate the location of the change introduced to the network. Link disconnections are presented in {1,2,3,4}, an exchange of neuron connections is introduced in {5} and a constant perturbation is introduced in {6}.



(a) (b) (c)

Figure 7. Convergence of robot trajectories induced by different candidate self-models (varying shades of blue). As it can be seen, a minimization of *D(c)* increases the similarity of resulting robot trajectories versus the innate target (red). Trajectory curves were obtained from environments of varying initial configuration of lights. Curves shown in (a) correspond to the environment used as training, (b) and (c) correspond to environments of increasing variation from the innate environment. As figures (b) and (c) show, resulting self-models are able to predict the robot behavior under unseen situations to a great extend. The axis scaling was adjusted for the data to fit in a cube.

Table 1. Damage Scenarios Tested

Scenario	Description
1	The left blue sensor input is damaged (input to node 0 is disconnected).
2	The left red sensor input is damaged (input to node 1 is disconnected).
3	The link from neuron 4 to neuron 7 is disconnected (weight changed to take the value of 0).
4	Two links entering node 6 are disconnected (weights changed to take the value of 0).
5	The weights connecting neuron 5 with neuron 6 and neuron 4 with neuron 7 are swapped.
6	A constant perturbation is introduced to left motor output (neuron 6).
7	No change

5. FAILURE DIAGNOSYS

In this section we describe how the self-model constructed by the reflective controller can be used to diagnose the damage introduced to the innate controller. As previously described, the self-model was derived from data collected during normal operation of the robot.

First we introduce one of the failures described on Table 1 to the innate controller and we then let the robot to operate on its environment while executing its now damaged innate controller. During this period we collected time series of sensor $Z^e = \{Z_t^e : t \in T\}$ and motor $U^e = \{U_t^e : t \in T\}$ data.

It is natural to expect at this point a difference between the sensorimotor relationships observed under failure versus those already explained by the self-model. We also expect a reduction in reward received from the environment.

Since the self-model was proven to provide good behavioral explanations of the innate sensorimotor relationships it is expected that small deviations from that solution might have some explanatory power of the failure.

Moreover we hypothesize that the topological distribution of these variations might have some degree of correlation with the actual source of the perturbation itself; we will further analyze these hypotheses in the remainder of this section.

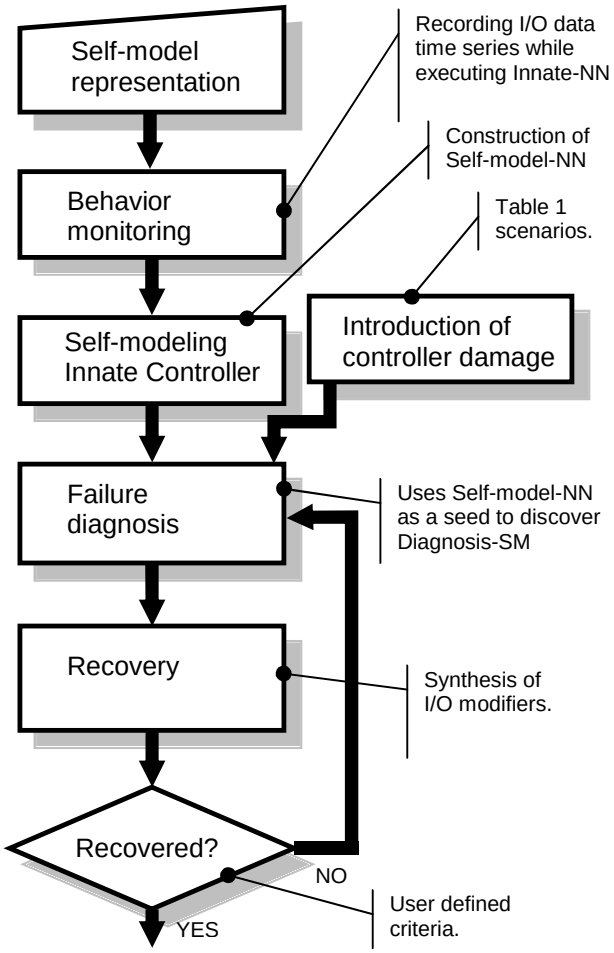


Figure 8. Flow diagram of the method to obtain a robot resilient behavior through controller self-diagnosis adaptation and recovery.

We used the following strategy for producing new diagnosis self-models (Diagnosis-SM): Given the sensorimotor signals obtained during the perturbed functioning of the robot, we started a new genetic search over the same self-model search space that was described in section 3. A first population of diagnosis self-models was strongly seeded with the optimal self-model (c'^*). In this case the search was steered toward minimizing the difference between the Diagnosis-SM outputs versus the outputs observed during the operation of the damaged innate controller, this was achieved by feeding each candidate Diagnosis-SM with the observed sensor time series Z^e, and by using the distance described in equation (4). If there were no damage at all, the solution would be the seed itself (validated by our experiments). In presence of damage, however, genetic search will steer the Diagnosis-SM network to produce similar motor patterns as those provided by the damaged Innate-NN. We trained the Diagnosis-SM using the same stopping criterion as described in section 3. We note the resulting optimal Diagnosis-SM as c''^*.

Before analyzing our diagnosis results we should note some hypothesis:

H1: *The solutions for Diagnosis-SM, c''^* and for Self-Model, c'^* have a similar explanatory power over the innate system c.*

Thus, any explanatory property assigned by an observer to the Self-model must also hold for the Diagnosis-SM.

H2: *The self-model synaptic difference vector $\Delta c = |c''^* - c'^*|$ represent topological changes of the self-model network, but does not necessarily have any counterpart in the real system.*

This is since the topology of the self-model is different from the topology of the innate controller (with exemption of the examples presented here to test H3).

H3: *The topological changes represented by Δc might better approximate changes of a target system whose architecture is similar to the topology of the self-models.*

A first estimation of the level of failure is given by quantitative measurements of parameter variation under each failure scenario. Figure 9 shows the standard deviation of the synaptic difference vector Δc that results when comparing the Self-Model with the Diagnosis-SM on each test scenario. The figure shows two cases. The first is when Δc is defined in a space of self-models of generic architecture ($\Delta c = |c''^* - c'^*|$). The second is when Δc is defined in a space of self-models whose architecture intentionally match the innate architecture ($\Delta c = |c_T''^* - c_T'^*|$).

It is interesting to observe (see Figure 6 as reference) that when disconnecting a blue sensor at the left side (failure 1) the level of compensation is greater than disconnecting a red sensor (failure 2). It appears that swapping the neuron connections also induces a small level of compensation (failure 5). The remaining failures {3, 4, 6} appear to induce a similar level of compensation as those required by failure 1.

Another type of diagnosis deals with the topological localization of the synaptic compensations that are hypothesized from the resulting difference vector Δc. We remark however, that this entails the careful consideration of the abovementioned hypotheses.

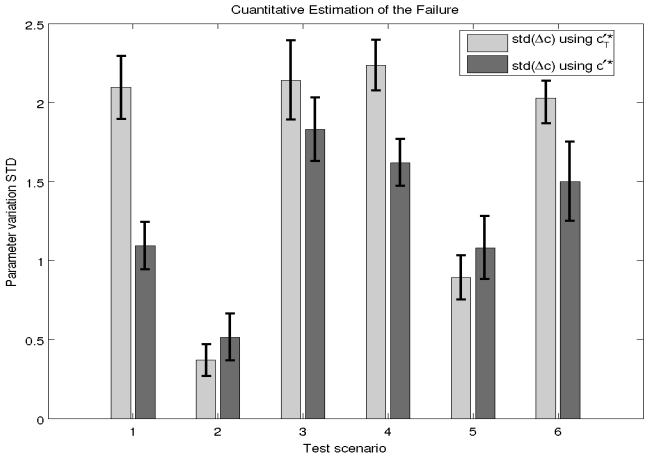


Figure 9. Quantitative estimation of the level of damage introduced under each test scenario to the innate neural network. The estimation is a result of comparing the original synaptic weights of the Self-model with the new synaptic weights resulting from evolving a Diagnosis-SM. Each bar represents the standard deviation of the weight difference vector Δc. Results are presented for the generic self-model architecture as well as for a test architecture.

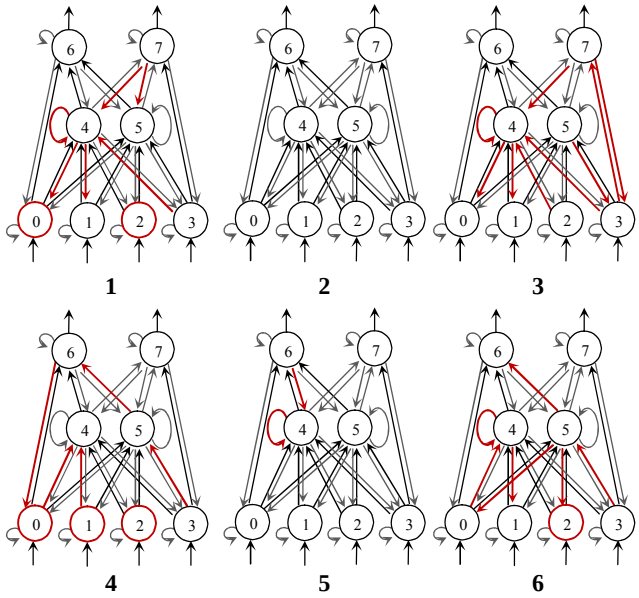


Figure 10. Most relevant compensatory modifications resulting from comparing Diagnosis-SM with the Self-Model for each one of the damage test scenarios. Arrows or nodes in red represent parameters on the Diagnosis-SM deviating more than the typical standard deviation (2.0) from the Self-model. In this case the Self-Model-NN_T architecture was used.

Figure 10 shows the localization of the synaptic weights that are experiencing a larger degree of change on each scenario. In the case of failure 1, it is interesting to observe that disconnecting a blue sensor induces a change on the bias of the same sensor as well as on the bias of the counter sensor of the opposite side. There is also a change on the synaptic weight of the counter motor modulator.

Disconnecting a forward connection between neuron 4 and neuron 7 produces a larger degree of compensation over most part of the network (failure 3). Disconnecting two forward connections produces a strong compensation on the network as well, see failure 4.

Figure 11 shows bar plots comparing the original Self-model synaptic weights (grey) versus the weights of resulting Diagnosis-SM (black) on each test scenario.

6. ADAPTATION AND RECOVERY

In this section we study how to recover function by filtering the inputs and/or outputs of the innate controller which is being subject of different types of damage. The filters are implemented using RNN's as described in [20]. Figure 1b shows a diagram of this procedure when applying the filters to the Innate-NN during robot functioning. The idea is to recover the innate function by synthesizing input/output modifier networks using the Diagnosis-SM as a model of the damaged innate controller to be recovered, and the Self-Model-NN that was generated in 3, as a model of the target system to be recovered.

These self-models are exploited by re-injecting sensor signals that were recorded during the robot operation on a real environment. The procedure allows evaluating the quality of modifier filters without further accessing the damaged innate controller. More details of this process are presented in [19][20].

Figure 12 shows reward levels resulting from the operation of the robot on two different environments. The different damage scenarios are presented. In each case there is a bar representing the reward level obtained before (black) damage, after damage (grey) and after recovery (white). As it can be seen, the procedure allows recovering performance to a great extend in most of the failure scenarios under analysis.

7. CONCLUSIONS

We have illustrated how the resiliency of a simulated robot increases using a self-reflection process that involves controller self-diagnosis, adaptation and recovery.

The proposed approach might also be useful for reusing existing hardware for new tasks, by applying the presented monitoring and controlling stages. The algorithm could be implemented, for example, inside a pre-existing robot, and modify its behavior by modulating its original controller's input and output signals.

In a broader sense, we have presented a case where system identification techniques can be used to infer the parameters of a controller, instead of the parameters of the dynamical system under control. While adaptive control aims to dynamically compensate for a plant whose parameters are uncertain, we address here the problem (and envision possible advantages) of adding uncertainty to the controller itself.

The proposed technique for reverse engineering a controller (section 3) can be of interest beyond robotics. For example, a common problem faced by growing production plants is related to the task of inferring the actual inner workings of their legacy control components.

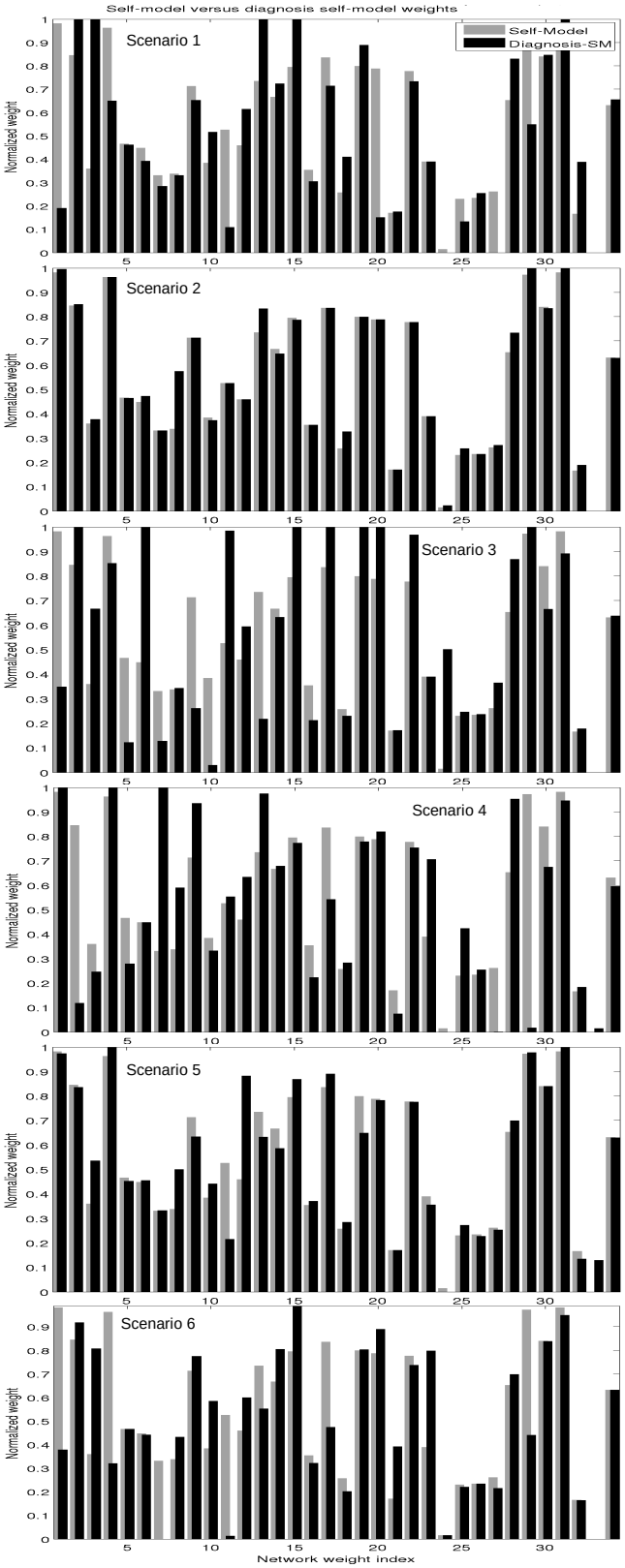


Figure 11. Comparing synaptic weights of Self-model (grey) versus resulting Diagnosis-SM (black) on each test scenario.

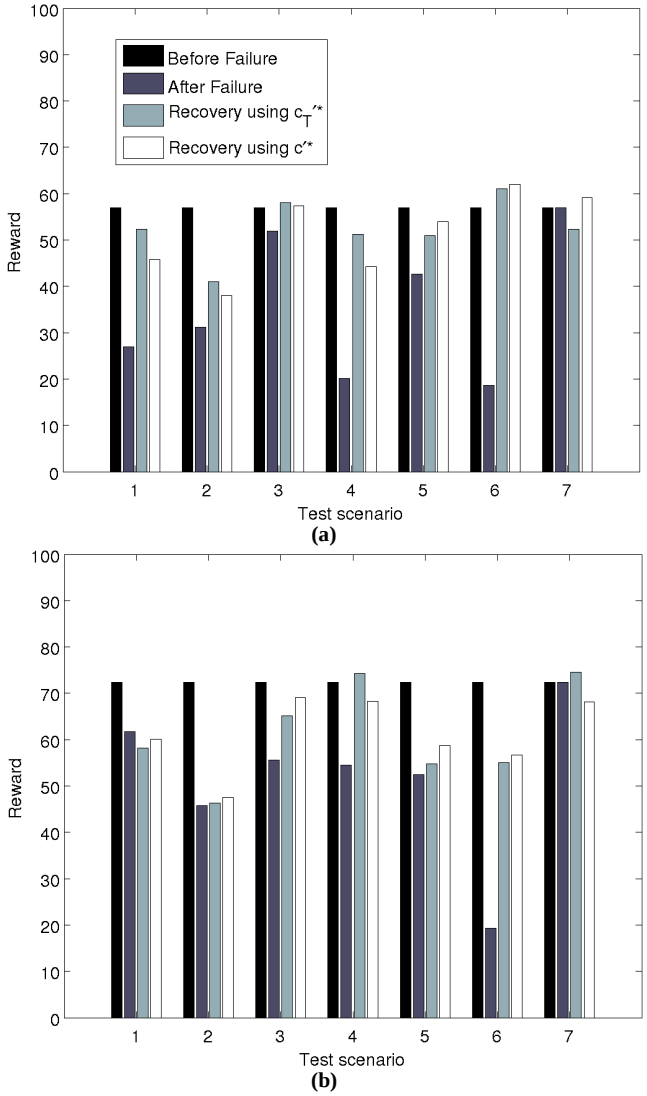


Figure 12. Reward levels obtained by the robot before failure, after failure and after recovery. Recovery results were obtained for the case of using self-models of ideal test architecture (solution $c_T'^*$) and the generic architecture (solution c'^*). Resulting reward levels are shown for two different environments (a,b) and considering the different failure scenarios. The simulation was set to be deterministic, having the same initial conditions under each scenario.

Although such knowledge is usually available for third party contractors it may be lost or too expensive. In theory, a meta cognitive system can be superimposed on an existing system, adding the possibility to monitor, control and further expand the capabilities of an existing system.

8. AKNOWLEDGEMENTS

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.

9. REFERENCES

[1] Bjorklund, A., Lindvall, O. 2000. Self-repair in the brain. Nature, 405, 892.

[2] Bongard, J., Lipson, H. 2004. Automated damage diagnosis and recovery for remote robotics. In IEEE International Conference on Robotics and Automation. Proceedings. ICRA '04. 3545-3550.

[3] Bongard, J., Lipson, H. 2004. Once more unto the breach: Co-evolving a robot and its simulator. In Artificial Life IX: Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, 57–62.

[4] Bongard, J., Zykov, V., Lipson, H. 2006. Resilient Machines Through Continuous Self-Modeling. Science, 314(5802):1118–1121.

[5] Foote, A., Crystal, J. 2007. Metacognition in the Rat. Current Biology, 17(6):551–555.

[6] Font, E., Desfilis, E., Pérez-Cañellas, M. M., García-Verdugo, J. M. 2001. Neurogenesis and neuronal regeneration in the adult reptilian brain. Brain Behav Evol, 58, 276-295.

[7] Draganski, B., Gaser, C., Busch, V., Schuierer, G., Bogdahn, U., May, A., et al. 2004. Neuroplasticity: changes in grey matter induced by training. Nature, 427(6972), 311-2.

[8] Hampton, R. 2001. Rhesus monkeys know when they remember. Proceedings of the National Academy of Sciences, 98(9):5359.

[9] Horner, P. J., Gage, F. H. 2000. Regenerating the damaged central nervous system. Nature, 407(6807), 963-70.

[10] Kempermann, G., van Praag, H., Gage, F. H. 2000. Activity-dependent regulation of neuronal plasticity and self repair. Progress in brain research, 127, 35-48.

[11] Minsky, M. 1986. The society of mind. Simon & Schuster, Inc. New York, NY, USA.

[12] Minsky, M. 2005. Interior grounding, reflection, and self-consciousness. In: Proceedings of International Conference on Brain, Mind and Society. Tohoku University, Japan.

[13] Minsky, M. 2007. The Emotion Machine, Simon & Schuster Inc. New York, NY.

[14] Nelson, T., Narens, L. 1990. Metamemory: A theoretical framework and new findings. The psychology of learning and motivation, 26:125–141.

[15] Smith, J., Shields, W., Washburn, D. 2004. The comparative psychology of uncertainty monitoring and metacognition. Behavioral and Brain Sciences, 26(03):317–339.

[16] Varela, F. G., Maturana, H. R., Uribe, R. 1974. Autopoiesis: the organization of living systems, its characterization and a model. Currents in Modern Biology, 5, 187.

[17] Zagal, J.C., Ruiz-del-Solar, J., Palacios, A.G. 2008. Fitness based identification of a robot structure. In Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems. 733-740.

[18] Zagal, J.C., Delpiano, J., Ruiz-del-Solar, J. 2009. Self-Modeling in humanoid soccer robots. Robotics and Autonomous Systems, 57(8), 819-827.

[19] Zagal, J.C., Lipson, H. 2009. Self-Reflection in Evolutionary Robotics: Resilient Adaptation with a Minimum of Physical Exploration. In Proceedings of the 11th Genetic and Evolutionary Computation Conference, GECCO'09, Montreal, Canada. 2179-2188.

[20] Zagal, J.C., Lipson, H. 2009. Towards Self-Reflecting Machines: Two-Minds in One Robot. In Proceedings of the 10th European Conference on Artificial Life. ECAL'09, Budapest.

[21] Zupanc, G. K. 2006. Neurogenesis and neuronal regeneration in the adult fish brain. Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology, 192, 649-670. Spring