Automatic Synthesis of Multiple Internal Models Through Active Exploration

Josh Bongard and Hod Lipson
Computational Synthesis Laboratory
Cornell University, Ithaca, NY 14853
Email: [josh.bongard | hod.lipson]@cornell.edu

Abstract

An important question in cognitive science is whether internal models are encoded in the brain of higher animals at birth, and are only subsequently refined through experience, or whether models are synthesized over the lifetime of an animal – and if so, how are they formed. A further question is whether animals maintain a single model of a particular body part or tool, or whether multiple competing models are maintained simultaneously. In this paper we describe a co-evolutionary algorithm that automatically synthesizes and maintains multiple candidate models of a behaving robot. These predictive models can then be used to generate new controllers to either elicit some desired behavior under uncertainty (where competing models agree on the resulting behavior); or determine actions that uncover hidden components of the target robot (where models disagree, indicating further model synthesis is required). We demonstrate automated model synthesis from sensor data; model synthesis 'from scratch' (little initial knowledge about the robot's morphology is assumed); and integrated, continued model synthesis and controller design. This new modeling methodology may shed light on how models are acquired and maintained in higher organisms for the purpose of prediction and anticipation.

Introduction

It has been argued that the brain evolved to control complex movement (Llinas, 2001). However, the inherent delays in biological sensors requires the use of models to correctly predict and anticipate the result of body motion. Evidence regarding the nature and number of models is growing (Wolpert, Miall, & Kawato, 1998), but this particular line of neuroscience inquiry is still in its infancy. For example, Imamizu *et al.* (2003) use brain imaging techniques to argue that when humans learn to use a new tool, a new model of that tool is created in the cerebellum. Yet despite the existence and modular nature of somatotopic maps in the same brain region (Penfield & Boldrey, 1937; Snider & Eldred, 1951), it remains unclear whether humans create different models of different body parts, or whether organisms maintain multiple competing models describing the same body part. In this paper we present an algorithm that allows a robot to automatically create multiple candidate models describing its own morphology, based only on recorded sensory data.

Internal models have been used since the beginning of classical AI research (see for example Nilsson (1980)), however in most cases the models are hard-coded, or are quantitatively updated using learning. Brooks (1987) proposed a model-less behavior based on purely reactive control. More recently, the field of evolutionary robotics (Nolfi & Floreano, 2000) has further reduced explicit modeling by using simulated evolution to generate reactive controllers automatically. In this method, a population of competing reactive controllers for a virtual robot are evolved using natural selection, after which the best controller is downloaded to the physical robot. However, often the transfer is unsuccessful due to inconsistencies between the physical and virtual environments. There are several approaches to this challenge, including adding noise to the simulated robot's sensors (Jakobi, 1997); adding generic safety margins to the simulated objects comprising the physical system (Funes & Pollack, 1999); evolving directly on the physical system (Thompson, 1997; Floreano & Mondada, 1998; Mahdavi & Bentley, 2003); evolving first in simulation followed by further adaptation on the physical robot (Pollack *et al.*, 2000; Mahdavi & Bentley, 2003); or implementing some neural plasticity that allows the physical robot to adapt during its lifetime to novel environments (Floreano & Urzelai, 2001; DiPaolo, 2000; Tokura *et al.*, 2001).

Here we reintroduce predictive internal models, but use evolutionary processes to simultaneously generate both the models and the reactive controllers based on them. In a continuous cycle of adaptation, models generate actions and actions are used to improve models. This co-evolutionary process, which we term the *Estimation-Exploration Algorithm* (EEA), can be used together with, or as an alternative to other controller-adaptation methods (Bongard & Lipson, 2004b). The advance of EEA over these other methods is two-fold: First, EEA uses automatically generated predictive models ("simulators") to minimize the number of trials actually performed on the target robot, thereby reducing the time, cost, and risk involved in physical learning. The second advantage of EEA is that not only are controllers evolved, but the robot's internal models themselves are continuously adapted, and are therefore not static. The models

change over time to better reflect the target robot itself and its local environment, and consequently controllers based on these models adapt to generate more effective behavior.

By using evolutionary algorithms to automatically generate accurate simulators, it is possible to move beyond simple parametric identification, in which some unknown physical parameters of the robot are inferred using feedback from the target robot, to topological identification. Evolutionary algorithms can search the space of possible robot morphologies in order to discover increasingly accurate simulators when the morphology of the robot may not be known. Indeed many evolutionary algorithms have been introduced that generate not only a robot controller suitable for a particular task, but also a suitable robot morphology (Sims, 1994; Adamatzky, Komosinski, & Ulatowski, 2000; Lipson & Pollack, 2000; Bongard & Pfeifer, 2001; Hornby & Pollack, 2002; Macinnes, 2003). The difference to the work here is that those methods evolve a morphology for a particular task; here, we demonstrate the use of the EEA to evolve a simulated robot morphology that functionally matches the morphology of the target robot and serve as a predictive internal model.

The next section describes the EEA in more detail. Section describes the inference of unknown physical parameters of a target robot. Section describes the inference of the target robot's morphology. The final section discusses various issues raised by the inference of robot morphologies, and provides some concluding remarks.

The Estimation-Exploration Algorithm

The estimation-exploration algorithm (EEA) is a co-evolutionary algorithm for performing system identification. System identification involves automated construction of a model of a target system, using only input supplied to the system and observed output (Ljung, 1999). The EEA divides the system identification into two components: the generation of accurate models (the estimation phase) and the generation of intelligent input data, or tests (the exploration phase). In previous papers (Bongard & Lipson, 2005b,a) we have demonstrated the application of the EEA to a number of system identification tasks, and have demonstrated that by intelligently selecting tests to perform on the target system, the internal structure of the target system can be inferred using less tests than if tests are selected at random.

An accurate model is defined as one that produces similar output data as the target system, when both the model and target system are supplied with the same input. An intelligent test can be much more broadly defined. In the case of system identification, an intelligent test is one that indirectly unveils hidden internal components of the target system, thereby allowing the generation of more accurate models. This can be achieved by evolving a test that causes maximal disagreement among the current set of candidate models; the resulting target system output from such a test will provide increased support of some models, and will prove the other models are inaccurate. In other situations, an intelligent test is one that elicits some desirable behavior from the most accurate model currently on hand; if the model is accurate, then that same test should produce the same desirable behavior on the target system. This different interpretation of an intelligent test is shown in Figure 1. In the work presented here the latter interpretation of an intelligent test is used: a test is a sensor-based neural network, and a good test is a network that, when downloaded on to the target system (which in this case is a mobile robot) causes the robot to maximize forward velocity.

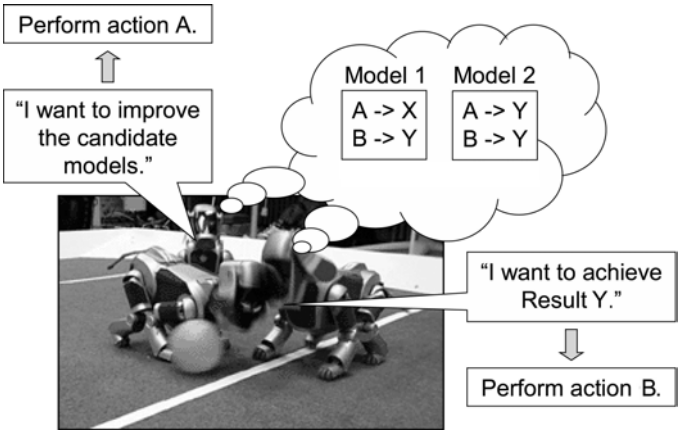

Figure 1: **Two complementary definitions of an intelligent test.** In this example a RoboCup team has access to a database containing two candidate models of the robot and its environment. The right-hand robot wishes to achieve a particular result, Y, so chooses to perform action B because both models $agree$ that Y will result, though the models are different. The left-hand robot, who wishes to know which model is correct, chooses to perform action A, because the two models $disagree$ as to the result of this action. Depending on which result occurs, one of the models will be invalidated.

Figure 2 illustrates the flow of the EEA pictorially. The algorithmic flow of the algorithm is given as follows:

1. **Initialization**
   (a) If an approximate model is available, goto 4).
   (b) If no model is available, generate a random test.
2. **Perform Target Trial**
   (a) Send evolved (or random) test to the target.
   (b) Record the resulting output.
3. **Evolve Candidate Models** (Estimation Phase)
   (a) If this is the first pass through the estimation phase, generate a random set of candidate models.
   (b) If it is the second or subsequent pass, seed the population with the best models evolved so far.
   (c) Provide the evolving models with all previous tests and outputs, plus the new test/output pair.
   (d) Output the best candidate models to the exploration phase.
4. **Evolve Informative Tests** (Exploration Phase)

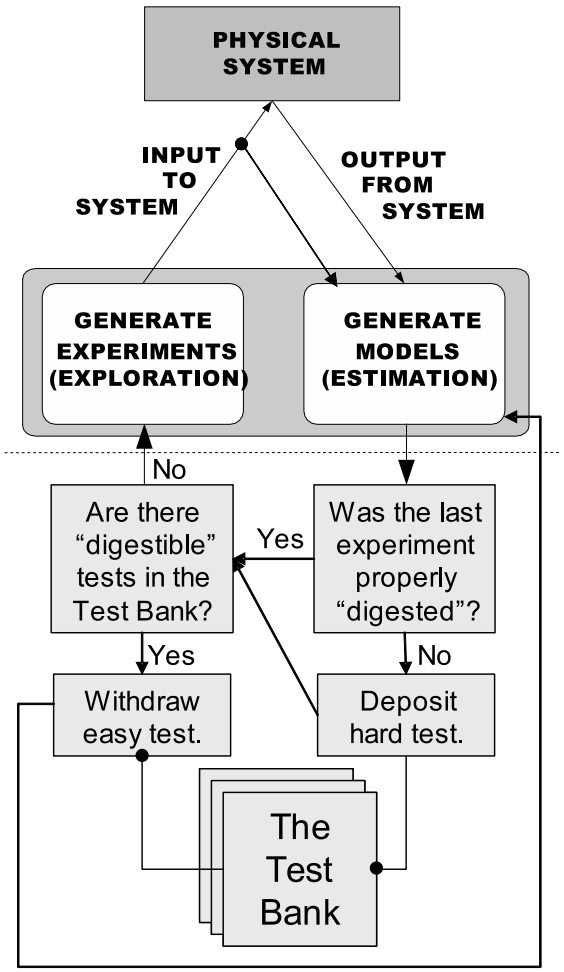

Figure 2: The enhanced estimation-exploration algorithm. The original algorithm is shown above the dotted line; the mechanism for 'managing challenge' is shown below the line.

(a) Always begin the exploration phase with a random population of tests.

(b) Evolve a test that causes the most disagreement between the candidate model(s) provided by the estimation phase, elicits some desirable behavior from the candidate model(s), or some combination of these two fitness criteria.

(c) Goto 2).

The EEA is cyclical in the sense that the evolution of models alternates with the evolution of tests, such that test data accumulates over the lifetime of the run: this stands in contrast to many system identification and machine learning methods in which a large amount of training data is gathered before inference begins. In the EEA, during the nth pass through the estimation phase there are n input/output data pairs available for model generation.

The algorithm is evolutionary in the sense that the estimation phase generates a population of candidate models using an evolutionary algorithm, and the exploration phase generates a population of tests also using an evolutionary algorithm. It should be noted however that an alternative search method, a heuristic algorithm or a combination of the two could be used in either phase in lieu of evolutionary search. Co-evolution implies that the evolutionary progress of one population is dependent on the evolutionary progress of the other. In the EEA, tests are evolved using the current most accurate models output by the estimation phase; and models are evolved based on the results of tests evolved in the exploration phase.

In the next two sections we describe the application of the EEA to the problem of automated model construction for mobile robotics. In section , we assume that the topology of the robot's body plan is known, and we infer some of the unknown parameters of the system, which we take to be mass distribution and sensor time lags. In section we assume less is known about the target robot: in this case we attempt to evolve the actual body plan of the robot using only observed sensor data.

Parametric Inference

In many robot applications the robot's state is known, but various physical aspects of itself and its environment are unknown (such as friction properties of foot contact or the robot's inertial tensor) so that behaviors generated using a model do not transfer well to the physical robot. This problem is particularly pronounced in evolutionary robotics (Nolfi & Floreano, 2000), in which behaviors are evolved in simulation and then transferred to the physical robot. In a previous paper (Bongard & Lipson, 2004b) we demonstrated the use of EEA to automatically improve various unknown physical parameters of the model robot such that behaviors could be transferred from simulation to the target robot. We will only outline this application here; the reader is referred to (Bongard & Lipson, 2004b) for more details.

Six steps must be followed to apply the algorithm to a given problem: characterization of the target system; initialization; estimation phase; exploration phase; termination; and validation.

1) Characterization of the target system. The target robot is a virtual robot operating within a three-dimensional physical simulator[1]. (As of yet we have not applied the EEA to a physical robot, but we are currently in the process of doing so.) Each of the nine objects comprising the robot has a different, randomly assigned mass between 1 and 7kg. Each of the eight sensors has an individual time lag that is selected randomly from the range $[0, 20]$ time steps; this value determines how many time steps elapse between the sensor reading and its input into the neural network controller. More details regarding the simulation of the robot can be found in (Bongard & Lipson, 2004b).

2) Initialization. To start, a random neural network is generated and supplied to the target robot. The target robot is evaluated in the simulator for 1000 time steps, and the resulting sensor time series are passed to the estimation phase.

[1]The robot simulator used throughout this paper is built on top of Open Dynamics Engine, `www.ode.org`.

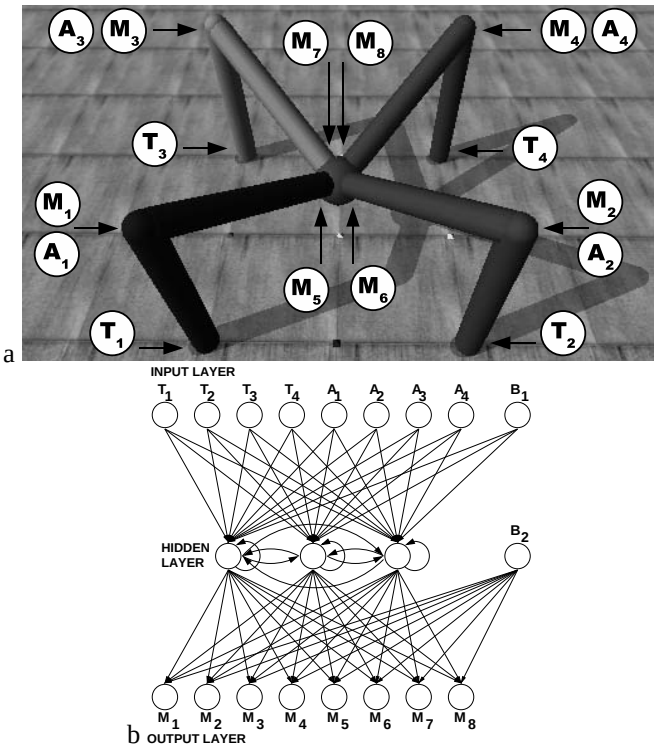

Figure 3: **a:** The morphology of the robot, including the distribution of its four touch sensors (T1-T4), four angle sensors (A1-A4), and eight motorized joints (M1-M8). **b:** The neural network controller of the robot, which connects the eight sensors to the eight motors via a single hidden layer, and an additional two bias neurons (B1-B2).

3) Estimation Phase. The estimation phase uses a genetic algorithm to evolve a population of 100 candidate models. A candidate model is given as a simulated robot that is identical to the target robot in all respects, except for the nine body part masses and the eight sensor time lags: the genome specifying a candidate model supplies these missing 17 values. The fitness of a candidate model is calculated as follows. The 17 values from the genome are used to increase the masses of the body parts and the time lags of the sensors: before labeling, the masses of all the body parts are assumed to be 1kg, and all sensor time lags are assumed to be zero. The model robot is then evaluated using all n neural network controllers have have been applied to the target robot so far, and the sensor time series are recorded. The fitness is then set to the inverse of the mean absolute difference between the n target robot and model robot sensor time series (see (Bongard & Lipson, 2004b) for details regarding this fitness calculation). The 50 most fit genomes are copied, mutated and crossed to produce 50 new genomes, which replace the 50 less fit genomes. This process continues for 30 generations, at which point the most fit model is output to the exploration phase. At the beginning of each pass through the estimation phase, the population is seeded with random genomes.

4) Exploration Phase. The exploration phase also evolves a population of 100 genomes for 30 generations, using the same mechanisms of mutation and crossover. However in this case each genome encodes 68 synaptic weights to specify a neural network controller. Each controller is applied to the model supplied by the estimation phase, and the fitness is set to the forward distance achieved by the model robot in 1000 time steps using that controller. At the end of this phase, the neural network with the highest fitness is output to the target robot. At the beginning of each pass through the exploration phase, the population is seeded with random genomes.

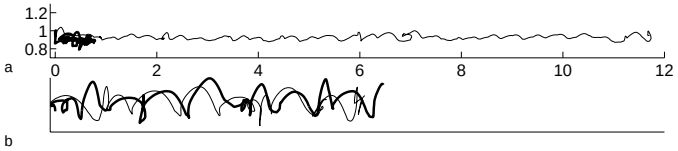

Figure 4: **Behavior recovery after controller transferal.** After the first pass through the exploration phase, the best evolved controller was supplied to the first model robot. The trajectory of its center of mass is given by the thin line in **a**. The same controller was then supplied to the target robot, and the resulting trajectory of its motion is given by the thick line in **a**. The movement of the 20th model robot using the 20th controller is given by the thin line in **b**. The motion of the target robot using the same controller is given by the thick line in **b**. The horizontal axis indicates forward distance, and the vertical axis indicates height (both are in meters).

5) Termination. The algorithm terminates after 20 neural networks have been evaluated on the target robot (*i.e.* 20 cycles through the EEA have been performed).

6) Validation. In this application, no validation was performed: the forward distance traveled by the target robot is considered an accurate metric of the algorithm's performance.

Results of Parametric Construction

Fifty independent runs of the EEA were performed: the model output by each pass through the estimation phase was recorded, as was the set of synaptic weights output by the exploration phase. Figure 4 shows results from one of the runs: as can be seen in Figure 4a, the neural network produced by the first pass through the exploration phase elicits very different behavior from the first model and the target robot, due to the inaccuracy of this first model. However after 20 cycles, the 20th neural network elicits very similar behavior from the 20th model and the target robot (Figure 4b), indicating that the 20th model was much more accurate.

Figure 5 reports the mean ability of the algorithm to infer the 17 unknown physical parameters. Clearly, the sensor time lags are more easy to infer than the body parts: this is not surprising, as it is these same sensors that provide the signals for inference. The morphological parameters however, are more difficult to infer because morphological must be indirectly inferred using sensor data. The algorithm infers, on average, the masses of body parts B2, B4 and B8 better than the other body parts. This may be due to the fact that B2, B4, B6 and B8 contain touch sensors (which pro-

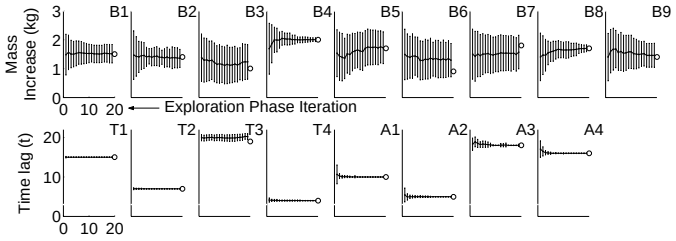

Figure 5: **Mean ability of the algorithm to identify hidden physical parameters.** The upper row shows the mean ability of the algorithm to infer the masses of the nine body parts B1 to B9 (B1 is the torso; B2, B4, B6 and B8 are the lower legs; B3, B5, B7 and B9 are the upper legs). The lower row shows the ability to infer the time lags of the touch sensors (T1-T4) and joint angle sensors (A1-A4). The open circles indicate the differences between the default robot simulator and the target robot (eg. the target torso is 1.5kg heavier than the default torso). Horizontal trajectories indicate the mean guess as to the correct parameter value, averaged over the best models output at the end of that pass through the estimation phase. Vertical lines indicate standard deviation.

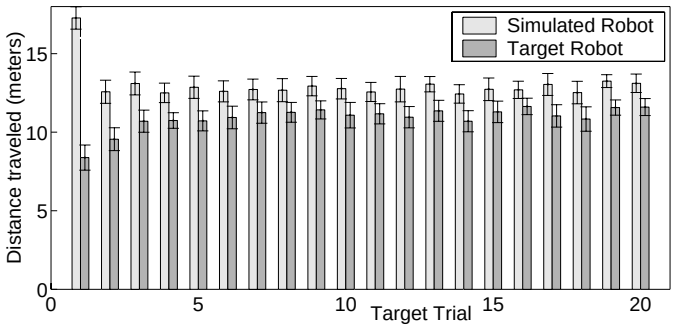

Figure 6: **Average transferal success after each target trial.** The light gray bars indicate the average distance traveled by the model robot using the best evolved controller output by that pass through the exploration phase, over all 50 runs. The dark gray bars indicate the average distance traveled by the target robot using the same controller, during each target trial. Error bars indicate standard error with a 95% confidence interval.

duce signals used for inference), while the other body parts do not. It may be that geometrical proximity to a sensor may make parameters about that body part easier to infer, however more experimentation is required to support this claim.

Figure 6 reports the mean performance of the EEA in terms of behavior transfer. As can be seen, the first evolved controller incurs a significant drop in performance when applied to the target robot; however after only a few more passes through the algorithm there is much less of a loss in performance. This indicates that, on average, the target robot travels only a little less far than the model robot using those later controllers, providing evidence that an accurate model has been evolved.

Topological Inference

In the previous example it was assumed that much was known about the target robot, and only some parameters of the system needed to be inferred: this is an example of parametric identification for nonlinear systems. We now demonstrate EEA's ability to reconstruct topological information about the target robot, such as which parts of the robot's body attach to which other parts: this is an example of structural identification of nonlinear systems, which is known to be a much more difficult endeavor Ljung (1999). As such, few attempts to perform structural identification of such systems have been made (Andrew, 1996; Gray *et al.*, 1998; Koza *et al.*, 1999; Rodriguez-Vazquez & Fleming, 1999; Chen *et al.*, 2005). To the best of our knowledge, this is the first attempt to perform structural system identification for nonlinear robot systems.

1) Characterization of the target system. The target system is assumed to be a robot composed of five cylindrical body parts: the robot itself is shown in Figure 7h. The robot contains two binary touch sensors, one in the anterior and posterior cylinders comprising the robot's spine; and four angle sensors, one in each of the robot's four actuated joints. The joints are actuated by one degree-of-freedom rotational joints: the two joints in the spine rotate through the sagittal plane; the two joints connecting the two front arms to the spine rotate through the frontal plane. All joints can rotate through $[-\pi/6, \pi/6]$ of their default angle, which are such that the robot lies flat on the ground plane.

Each cylindrical body part has a mass of 1kg. The three spinal parts have a length of 1m, and the arms have lengths of 80cm[2]; all body parts have a radius of 10cm. The arms are anchored at the center of the anterior spinal body part.

The neural network controller for this robot is similar to that shown in Figure 3b, except that the input layer contains six neurons (corresponding to the six sensors) and the output layer contains only four neurons (corresponding to the four motors at the joints).

2) Initialization. As in the previous case, a random set of synaptic weights are generated and downloaded to the target robot, which is allowed to move for 20 time steps. Unlike the previous case, the success of the algorithm is determined to be how well the algorithm can reconstruct the topology of the target robot, and not how far the algorithm can get the target robot to move. It was found that only a few time steps of observed sensor data was required in order to correct infer the topology of the robot, so the algorithm was greatly sped up by limiting the evaluation of the target robot (and the model robots) to only 20 time steps. The time series of the six sensors is then fed into the estimation phase.

3) Estimation Phase. The estimation attempts to reconstruct the topology of the target robot, not just some of its physical parameters. This is significantly more difficult be-

[2]The relatively large masses and sizes of the robot's body parts was implemented because of the particular collision detection method of the underlying simulation, which performs more accurate collision detection and resolution for large and heavy objects. It is anticipated that the results acquired using this model would be similar for a smaller and lighter robot.

cause the search space of possible models is much larger and more deceptive.

In this application, the following data is assumed to be **known** about the target robot:

- The robot is composed of five cylindrical body parts.
- The radii of the body parts.
- The masses of the body parts are known.
- All of the four joints are motorized by a one degree-of-freedom rotational motor.
- Each joint contains an angle sensor.
- A body part may or may not contain a touch sensor.
- There are two touch sensors.

The following is assumed to be **unknown:**

- The lengths of the body parts.
- Which body part is attached to which other one.
- The relative position and orientation of a body part attachment.
- The location of the two touch sensors.

The genomes of the estimation phase must encode these missing morphological details in order to produce a candidate model. Genomes are therefore encoded as an $n \times 6$ matrix with real values in $[0, 1]$, where $n = 5$ equals the number of robot body parts. Therefore row i of the genome matrix then encodes the missing data for body part i.

Candidate model construction from a genome proceeds as follows. For each row i, the first value in the row is scaled to an integer in $[0, i-1]$: this value indicates which body part the current body part attaches to. This value in the first row of the matrix—which corresponds to the first body part—is ignored (this body part is considered as the root body part), and this value in the second row is always set to 0 (the second body part can only attach to the root body part). The values of the second column are rounded to binary values: a 0 in this position indicates that the current body part attaches to the tail of its parent body part; a 0 indicates it attaches to the head of its parent body part. The tail of a body part corresponds to its attachment point to its own parent; the head is the tip of the body part away from the attachment part. The head of the root body part is considered to be the end with a positive z-coordinate value[3]. The third value is scaled to a real value in $[l_{\text{min}}, l_{\text{max}}]$, which indicate the minimum and maximum possible lengths of the body parts as set by the user, respectively. The fourth and fifth values are scaled to $[0, \pi]$, and represent the two Euler angles used to calculate the three-dimensional orientation of the current body part relative to its parent body part.

The sixth value is scaled to a binary value, and indicates whether the current body part contains a touch sensor or not. If a genome encodes more than two touch sensors, then only the first two touch sensors are included in the model robot, reading from the top of the six column of the genome matrix downward; the additional touch sensors are ignored and not included in the robot. If less than two touch sensors are encoded in the genome matrix, the unspecified touch sensors are placed randomly in the model robot's body, and are disabled: they output a zero value during each time step of the evaluation. Valid touch sensors provide either a 1 or -1 signal into the controller at each time step, corresponding to whether the body part containing the sensor is touching the ground plane or not; the four angle sensors take the current joint angle (in $[-\pi/6, \pi/6]$), divide the value by $\pi/6$ and input the resulting value in $[-1, 1]$ into the controller.

Using each genome matrix, a fully specified model robot is constructed, and its fitness is evaluated. The model robot is evaluated using each of the n neural network controllers that have been evaluated on the target robot so far, and the resulting sensor signals from these n runs is recorded. The model robot's fitness is then given as the inverse of the subjective error, where subjective error is calculated as:

$$e_s = \frac{\sum_{i=1}^{n} \sum_{j=1}^{s} \sum_{k=1}^{t} |o_{ijk} - a_{ijk}|}{nst}, \quad (1)$$

where s is the number of sensors contained in the robot, $t = 20$ is the number of time steps for which the target and model robots are evaluated for, o_{ijk} is the observed value of sensor j from the target robot using controller i during time step k, a_{ijk} is the actual value of sensor j obtained from the model robot using controller i during time step k.

During the first pass through the estimation phase, a population of 100 random genome matrices are generated. After each encoded model robot has been evaluated, selection and mutation is performed (crossover is not currently used). Seventy-five pairs of genome matrices are chosen in sequence. For each pair, the genome with the higher fitness is copied and overwrites the genome with lower fitness. The copied genome is then mutated as follows. A single element e_{ij} is selected at random in the matrix, and is set to:

$$e_{ij} = \begin{cases} \alpha & : \quad \beta = 0 \\ e_{ij} + \gamma 10^{-\delta} & : \quad \text{otherwise} \end{cases} \quad (2)$$

where α, β, γ and δ are random variables: α is a real value chosen from $[0, 1]$; β is a random binary value; γ is either -1 or 1; δ is a real value chosen in $[1, 7]$. All random values are chosen using a uniform distribution. This mutation operator ensures that half the time a new random value is chosen, while otherwise the current value is nudged up and down by a small randomly-determined amount. Because the most fit genome cannot be overwritten, it is guaranteed to be preserved into the next generation.

The estimation proceeds for 20 generations. At the end of the estimation phase, the most accurate simulator is output to the exploration phase. On the second and subsequent passes through the estimation phase, the initial random population is seeded $i - 1$ genome matrices, which describe the most accurate model robots output by the previous $i - 1$ passes through the estimation phase.

4) Exploration Phase.

[3]The center of the root body part is always considered to be located at $[0, r, 0]$, where r is the known radii of the body parts. The root body part is always oriented horizontally along the sagittal plane. This ensures that no matter the length of the root body part, one end will have a positive z-coordinate.

The exploration phase is currently not enabled for this application: this phase simply outputs a random neural network controller. In this application the goal is not produce a controller that can be transferred from simulation to the target robot, but rather to produce an accurate simulation of the target robot automatically. In previous applications (including the one described in section) we have found that random sensor-based controllers are sufficient to elicit varied behavior from the target robot, and therefore to indirectly elicit morphological information about the target robot through the sensors. The random controller is downloaded on to the target robot, and the target robot is evaluated.

5) Termination. The algorithm was run for only 10 cycles, as in many runs the accuracy of robot models output by the estimation phase did not increase much past that point.

6) Validation. In order to validate the accuracy of the robot models, an objective error metric was formulated:

$$e_o = \frac{\sum_{i=1}^{5} \sqrt{(t_x^i - m_x^i)^2 + (t_y^i - m_y^i)^2 + (t_z^i - m_z^i)^2}}{5} \quad (3)$$

Where $t^i_{x|y|z}$ is the x-, y- or z- coordinate of the center of the ith body part of the target robot and $m^i_{x|y|z}$ is the x-, y- or z- coordinate of the center of the ith body part of the model robot. Computing the mean distance between body parts gives a good approximation of how well the algorithm has inferred the overall body plan of the target robot.

Results of Topological Construction

Three sets of 15 independent runs were performed using the algorithm described in the previous sub-section. In the first set, the target robot returned the six time series of the signals from the two touch and four angle sensors. In the second set of runs, a range sensor was attached to each of the five body parts. A simulated range sensor returns a value at each time indicating the distance from the center of that body part to a fixed external point at $[-1, -1, 0]$; this is a simulated analogue of a beacon, a laser range finder or light sensor in which luminosity is an indirect measure of distance from a light source. In the third set of experiments each body part contained four range sensors, providing distance information of how far the center of that body part is from four external fixed points at $[-1, -1, 0]$, $[-1, 1, 0]$, $[1, -1, 0]$ and $[1, 1, 0]$. In the first set of experiments the touch and angle sensors drive behavior by stimulating the neural network controller as well as provide time series for inference; in the second and third experiment sets the touch and angle sensors continue to be used to drive behavior, but the additional range sensors (which do not provide data to the controller) provide the data for inference.

Figure 7 shows some sample model robots from a typical run of the third experiment set. Four random model robots are shown (**a**-**d**), as well as the best model output by the first pass through the estimation phase (**e**); the best model after the second pass (**f**); the best model after the tenth pass (**g**); and the target robot itself (**h**). As can be seen, the algorithm converges on an approximate description of the target robot's morphology.

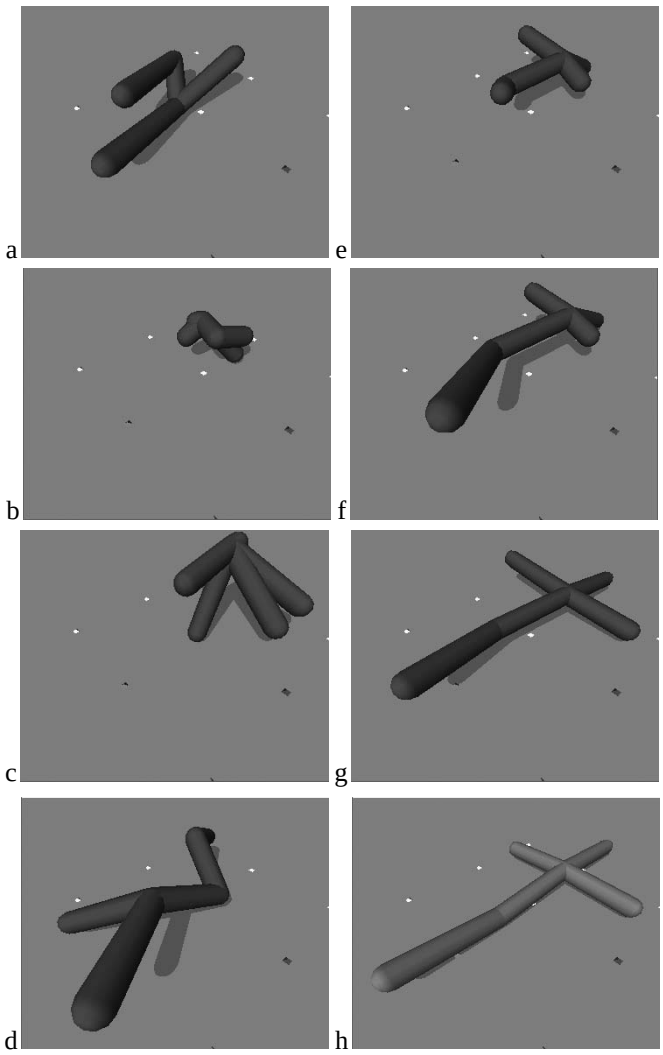

Figure 7: **The experimental enclosure for simulated experimentation with self assembly and reconfiguration. a:** The enclosure, top view. **b:** Enclosure, side view. **c:** Magnification of the seed plate. The cross indicates that the top face of the middle block is magnetized.

Figure 8 reports the progress of one typical run from each of the three experimental regimes. As can be seen in Figure 8a, the touch and angle sensors do not provide sufficient information for morphological inference; this algorithm variant does not discover models more accurate than those generated in the initial random model population. In the case when range sensors are used for inference instead of touch and joint angle information (Figure 8b), some inference takes place; later models tend to be more accurate than earlier models, although not by very much. In the typical run from the third regime (Figure 8c), in which four range sensors are used on each body part, significant inference takes place: later models are much more accurate than those generated in the initial random population.

The most accurate model output by each pass through the

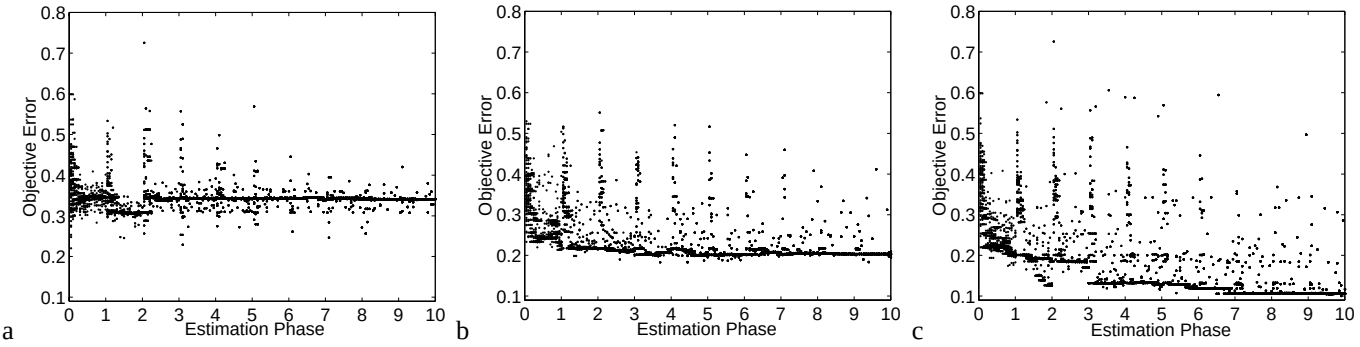

Figure 8: **Evolutionary progress of a typical run from each experimental regime. a:** A typical run from the first regime, in which the two touch and four angle sensors return signal time series for inference. **b:** A typical run from the second regime, in which the five range sensors (one for each body part) return signal time series for inference. **c:** A typical run from the third regime, in which the 20 range sensors (four for each body part) return signal time series for inference. Each point represents the objective error (Eq. 3) of one model robot.

estimation phase was recorded: the objective errors of these 15 models from each run were calculated and averaged. Figure 9 reports these means for all three experimental regimes. As can be seen, the use of range sensors is much more informative for morphological inference than the combined touch and joint angle sensors: both the second and third regimes perform significantly better than the first regime. Additionally, the use of only five range sensors allowed for inference to occur, while six sensors of different modality (touch and angle) did not: this indicates that the amount of sensors used for inference is less important than the type of sensor used: range information yields (indirectly) more morphological data than touch and joint angle information do. It is also clear from Figure 9 that the regime in which 20 range sensors were used for inference far outperformed the other two regimes, indicating that amount of sensor data is also important: range data from four external points yields more information about a body part than range data from only one external point. We hypothesize (although as of yet have not proven) that the multiple range signals for each body part allow the EEA to perform some kind of triangulation and thereby infer more morphological detail than if only a single range datum is available for each body part, per time step.

Figure 9 also indicates that the third regime allows for significant inference to occur using only a single target robot trial; the third regime produces more accurate models ($e_o \approx 0.21$) after only a single pass compared to the first regime ($e_o \approx 0.325$) and the second regime ($e_o \approx 0.275$).

Discussion and Conclusions

In this paper we have described the application of our system identification algorithm—the estimation-exploration algorithm (EEA)—for the automatic generation of robot predictive internal models (simulators). In the first set of experiments we demonstrated that EEA can be used to reconstruct unknown physical parameters of a target robot. We demonstrated that even though the algorithm may not correctly infer all of the unknown parameters (Figure 5), it is sufficient to allow for successful transferal of neural network controllers evolved using the inferred robot simulator to the target robot (Figure 6).

In the second set of experiments we demonstrated that the algorithm can reconstruct the morphology of a hidden target robot 'from scratch'. Even if the detailed topology of the robot's body plan is unknown, it can be reconstructed only from sensor data. Furthermore, we demonstrated that certain sensor modalities are more valuable for inference than others. In this case, range sensors provides more indirect morphological information than a combination of touch and joint angle sensors does. One of the unique properties of the EEA is that sensors serve a dual role: They guide behavior via sensor-based neural network controllers, and they also provide signals for use in the inference process.

Generating a robot simulator automatically, with little prior assumed knowledge, could be valuable in a number of situations. In a previous paper we demonstrated that EEA could be used for damage diagnosis and recovery in remote environments. If a robot's morphology alters as a result of physical damage, it may be necessary to reconstruct a new simulation describing the changed, unknown morphology in order to generate a compensatory controller Bongard & Lipson (2004a). Also, in reality a mobile robot contains many components—electronics, batteries, wires, flexible parts, scientific instrumentation—that make creating a detailed physical model of the robot very difficult and time consuming. By accelerating the simulation generation cycle through automation, the method outlined here could greatly speed the entire process of robot design, manufacture and deployment.

One major question in cognitive science, as it relates to modeling, is whether higher animals are born with models encoded in the brain which are then subsequently refined through experience, or are models synthesized 'from scratch' as the animal learns about its body and the world around it. In this paper we have documented two such approaches to modeling in robotics: the parametric identification of various physical properties of a fixed robot simulator; and the topological identification of the robot's mor-

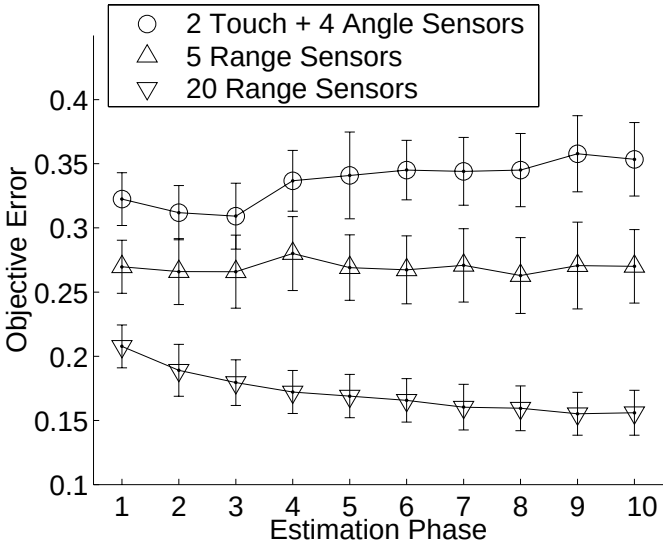

Figure 9: **Mean inferential ability of the three experimental regimes.** The mean objective errors (Eq. 3) of the best models output by the 10 passes through the estimation phase were calculated and averaged for each experimental regime. Error bars indicate standard error with a confidence interval of 95%.

phology. Another major question is whether higher animals maintain one or multiple models of their morphology. The EEA demonstrates how multiple models can be useful for predication and anticipation: synthesizing controllers that cause the models to agree increases the probability that the physical robot will reproduce the desired behavior synthesized in simulation; and synthesizing controllers that cause the models to disagree increases the probability of obtaining novel motor-sensor transformations that allow for further model refinement. Therefore this methodology may ultimately shed light on how models are created and maintained in higher organisms, what role they play in prediction and anticipation, and eventually, in intelligence.

Acknowledgments

This research was supported in part by the NASA Program for Research in Intelligent Systems, under Grant NNA04CL10A.

References

Adamatzky, A.; Komosinski, M.; and Ulatowski, S. 2000. Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics* 29:1344–1351.

Andrew, H. 1996. System identification using genetic programming. In *Proceedings of the Second Intl. Conf. on Adaptive Computing in Engineering Design and Control*, 57–62.

Bongard, J. C., and Lipson, H. 2004a. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In *Proceedings of The 2004 NASA/DoD Conference on Evolvable Hardware*, 169–176.

Bongard, J., and Lipson, H. 2004b. Once more unto the breach: Co-evolving a robot and its simulator. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, 57–62.

Bongard, J. C., and Lipson, H. 2005a. Active coevolutionary learning of deterministic finite automata. *Journal of Machine Learning Research* in review.

Bongard, J. C., and Lipson, H. 2005b. Nonlinear system identification using co-evolution of models and tests. *Evolutionary Computation* in review.

Bongard, J., and Pfeifer, R. 2001. Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO 2001)* 829–836.

Brooks, R. A. 1987. Intelligence without representation. *Artificial Intelligence* 47:139–160.

Chen, Y.; Yang, J.; Zhang, Y.; and Dong, J. 2005. Evolving additive tree models for system identification. *International Journal of Computational Cognition* 3(2):19–26.

DiPaolo, E. A. 2000. Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. In Meyer, J. A.; Berthoz, A.; Floreano, D.; Roitblat, H. L.; and Wilson, S. W., eds., *From Animals to Animats 6*, 440–449. MIT Press.

Floreano, D., and Mondada, F. 1998. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks* 11:1461–1478.

Floreano, D., and Urzelai, J. 2001. Neural morphogenesis, synaptic plasticity, and evolution. *Theory in Bioscience* 120:225–240.

Funes, P., and Pollack, J. 1999. Computer evolution of buildable objects. In Bentley, P., ed., *Evolutionary Design by Computer*. San Francisco: Morgan Kauffman. 387–403.

Gray, G.; Murray-Smith, D.; Li, Y.; Sharman, K.; and Weinbrenner, T. 1998. Nonlinear model structure identification using genetic programming. *Control Engineering Practice* 6:1341–1352.

Hornby, G. S., and Pollack, J. B. 2002. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life* 8(3):223–246.

Imamizu, H.; Kuroda, T.; Miyauchi, S.; Yoshioka, T.; and Kawato, M. 2003. Modular organization of internal models of tools in the human cerebellum. *PNAS* 100(9):5461–5466.

Jakobi, N. 1997. Evolutionary robotics and the radical envelope of noise hypothesis. *Adaptive Behavior* 6(1):131–174.

Koza, J.; Bennett, F.; Andre, D.; and Keane, M. 1999. *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann.

Lipson, H., and Pollack, J. B. 2000. Automatic design and manufacture of artificial lifeforms. *Nature* 406:974–978.

Ljung, L. 1999. *System Identification: Theory for the User*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Llinas, R. R. 2001. *i of the vortex*. Cambridge, MA: The MIT Press.

Macinnes, I. 2003. Visually guided physically simulated agents with evolved morphologies. In Banzhaf, W.; Christaller, T.; Dittrich, P.; Kim, J.; and Ziegler, J., eds., *Advances in Artificial Life—Proceedings of the Seventh European Conference on Artificial Life (ECAL)*, 821–828. Berlin: Springer Verlag.

Mahdavi, S. H., and Bentley, P. J. 2003. An evolutionary approach to damage recovery of robot motion with muscles. In *Seventh European Conference on Artificial Life (ECAL03)*, 248–255. Springer.

Nilsson, N. 1980. *Principles of Artificial Intelligence*. San Francisco: Morgan Kauffman.

Nolfi, S., and Floreano, D. 2000. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Boston, MA: MIT Press.

Penfield, W., and Boldrey, E. 1937. *Brain* 37:389–443.

Pollack, J. B.; Lipson, H.; Ficici, S.; Funes, P.; Hornby, G.; and Watson, R. 2000. Evolutionary techniques in physical robotics. In Miller, J., ed., *Evolvable Systems: from biology to hardware*, 175–186. Springer-Verlag.

Rodriguez-Vazquez, K., and Fleming, P. 1999. Genetic programming for dynamic chaotic systems modelling. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, 22–28.

Sims, K. 1994. Evolving 3D morphology and behaviour by competition. *Artificial Life IV* 28–39.

Snider, R. S., and Eldred, E. 1951. *J. Neurophysiol.* 15:27–40.

Thompson, A. 1997. Artificial evolution in the physical world. In Gomi, T., ed., *Evolutionary Robotics: From intelligent robots to artificial life (ER'97)*, 101–125. AAI Books.

Tokura, S.; Ishiguro, A.; Kawai, H.; and Eggenberger, P. 2001. The effect of neuromodulations on the adaptability of evolved neurocontrollers. In Kelemen, J., and Sosik, P., eds., *Sixth European Conference on Artificial Life*, 292–295.

Wolpert, D. M.; Miall, R. C.; and Kawato, M. 1998. Internal models in the cerebellum. *Trends in Cognitive Sciences* 2(9):338–347.