International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 09 | September - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Self-Healing Robot

Aniket Manik Kathe[1]

[1]Student Mechanical Engineering Department, PVGCOE & SSDIOM, Nashik

---

Abstract - When people or animals get hurt, they will usually compensate for minor injuries and keep limping along, except for robots, even slight damage can make them stumble and fall. Now a robot scarcely larger than a person's hand has demonstrated a novel ability: It can recover from damage -- an innovation that could make robots more independent.
The new robot, which seems like a splay-legged, four-footed starfish, deduces the form of its own body by performing a series of playful movements, swiveling its four limbs. By using sensors to record resulting changes within the angle of its body, it gradually generates a computerized image of itself. The robot then uses this to plan out the way to walk forward.
The researchers hope similar robots will someday respond not only to wreck to their own bodies but also to changes in the surrounding environment. Such responsiveness could lend autonomy to robotic explorers on other planets like Mars -- a helpful feature, since such robots can't always be in touch with human controllers on earth. apart from practical value, the robot's abilities suggest a similarity to human thinking because the robot tries out various actions to figure out the shape of its world.

***Key Words*:** Error recovery, self-healing, algorithm implementation, theory to reality.

1.INTRODUCTION

1.1 Robots

A robot may be a mechanical or virtual, artificial agent. it's usually an electromechanical system, which, by its appearance or movements, conveys a way that it has intent or agency of its own.
A typical robot will have several, though not necessarily all of the subsequent properties:
• Is not 'natural' i.e., has been artificially created.
• Can sense its environment.
• Can manipulate things in its environment.
• Has a point of intelligence or ability to make choices based on the environment or automatic control / pre-programmed sequence.
• Is programmable.
• Can travel along one or more translational or rotational axes.
• Can make dexterous coordinated movements.
• Appears to possess intent or agency (reification, anthropomorphisation or Pathetic fallacy).
Robotic systems are of growing interest due to their many practical applications as well as their ability to help understand human and animal behavior, cognition, and physical performance. Although industrial robots have long been used for repetitive tasks in structured environments, one among the long-standing challenges is achieving robust performance under uncertainty. Most robotic systems use a manually constructed mathematical model that captures the robot's dynamics and is then wont to plan actions. Making accurate models for complex machines is challenging, especially when attempting to account for potential topological changes to the body, such as changes brought on by damage. Despite the existence of some parametric identification methods for automatically improving these models, this is especially true for complex machines.

Figure 1: Robot

1.2 Error Recovery

Recovery from error, failure or damage may be a major concern in robotics. A majority of effort in programming automated systems is devoted to error recovery. the necessity for automated error recovery is even more acute in the field of remote robotics, where human operators cannot manually repair or provide compensation for damage or failure.
Here, it's detailed how the four-legged robot, through minimal but self-directed interaction, spontaneously creates a predictive model of its own topology (where and the way its body parts are connected) with its environment, then uses this model to synthesize successful new locomotive behavior before and after damage. These findings may help develop more robust robotics, also as shed light on the relation between curiosity and cognition in animals and humans.

2.LITERATURE REVIEW

1. S Terryn
The intrinsic compliance of sentimental robots provides safety, a natural adaptation to its environment, allows to soak up shocks, and protects them against mechanical impacts. However, a literature study shows that the soft polymers used for his or her construction are susceptible to various types of damage, including fatigue, overloads, interfacial debonding, and cuts, tears and perforations by sharp objects. An economic and ecological solution is to construct future soft robotic systems out of self-healing polymers, incorporating the power to heal damage. This review paper proposes criteria to guage the potential of a self-healing polymer to be used in soft robotic applications. supported these soft robotics requirements and on defined performance parameters of the materials, linked to the mechanical and healing properties, the

various types of self-healing polymers already available in literature are critically assessed and compared.

2. Richard McWilliam
this text reviews the existing work in self-healing and self-repairing technologies, including add software engineering, materials, mechanics, electronics, MEMS, self-reconfigurable robotics, et al. . It suggests a terminology and taxonomy for self-healing and self-repair, and discusses the varied related types of other self-* properties. The mechanisms and methods resulting in self-healing are reviewed, and customary elements across disciplines are identified.

3. R. Adam Bilodeau
Advances in soft robotics are going to be crucial to the next generation of robot–human interfaces. Soft material systems embed safety at the fabric level, providing additional safeguards which will expedite their placement alongside humans and other biological systems. However, so as to function in unpredictable, uncontrolled environments alongside biological systems, soft robotic systems should be as robust in their ability to get over damage as their biological counterparts. There exists an excellent

deal of labor on self-healing materials, particularly polymeric and elastomeric materials which will self-heal through a wide variety of tools and techniques. Fortunately, most emerging soft robotic systems are constructed from polymeric or elastomeric materials, so this work are often of immediate benefit to the soft robotics community. Self-healing and damage robust systems are beginning to be included into the three major support pillars that are enabling the future of soft robotics: actuators, structures, and sensors, despite the fact that the sector of soft robotics is still in its infancy as a whole. This paper examines the most recent advancements in self-healing and damage resilience technologies as they relate to these three pillars. Future uses for soft robots that have self-healing capabilities are also covered in this review.

4. Joost Brancart
Soft robots are nearly entirely constructed of flexible, soft material, making them suited for applications in uncertain, dynamic task contexts, including secure human-robot interactions. They are inspired by the compliance seen in many creatures. They are protected from mechanical collisions and shocks by their inherent compliance. However, the soft materials used for his or her construction are highly susceptible to damage, like cuts and perforations caused by sharp objects present in the uncontrolled and unpredictable environments they operate in. during this research, we propose to construct soft robotics entirely out of self-healing elastomers. On the idea of healing capacities found in nature, these polymers are given the power to heal microscopic and macroscopic damage. Diels-Alder polymers, being thermo reversible covalent networks, were wont to develop three applications of self-healing soft pneumatic actuators (a soft gripper, a soft hand, and artificial muscles). Soft pneumatic actuators commonly experience perforations and leaks thanks to excessive pressures or wear during operation. Finite element modelling was used to create all three prototypes, which were then mechanically characterised. The manufacturing method of the actuators exploits the self-healing behavior of the materials, which may be recycled.

3. SELF MODELLING ROBOTS

When people or animal get injured, they catch up on minor injuries and keep limping along. But within the case of robots, even a small injury can make them stumble and fall. Self-healing robots have a capability to adapt to minor injuries and continue its job. A robot is in a position to indirectly infer its own morphology through self- directed exploration and then use the resulting self-models to synthesize new behaviors. If the robot's topology unexpectedly changes, the identical process restructures its internal self-models, resulting in the generation of qualitatively different, compensatory behavior. In essence, the method enables the robot to continuously diagnose and recover from damage. Unlike other approaches to wreck recovery, the concept introduced here doesn't presuppose built-in redundancy, dedicated sensor arrays, or contingency plans designed for anticipated failures. Instead, our approach is predicated on the concept of multiple competing internal models and generation of actions to maximize disagreement between predictions of these models.

3.1 Researchers

Figure 2: Victor Zykov, Josh Bongard, and Hod Lipson

The Computational Synthesis Lab at Cornell University conducted the study. Hod Lipson, Viktor Zykov, and Josh Bongard are the members of the team. Josh Bongard conducted this research as a postdoctoral scholar at Cornell before moving on to the University of Vermont, where he is currently an assistant professor. Hod Lipson is a professor at Cornell and the director of the Computational Synthesis Lab, while Victor Zykov may be a Ph.D. candidate at CCSL. The National Science Foundation's Engineering Design Program and the NASA Program on Intelligent Systems both provided funding for this study.

3.2 The Starfish Robot

3.2.1 Characterizing the Target System

The target system during this study is a quadrupedal, articulated robot with eight actuated degrees of freedom. The robot consists of an oblong body and four legs attached to it with hinge joints on each of the four sides of the robot's body. Each leg successively is composed of an upper and lower leg, attached along with a hinge joint. Airtronics 94359 high torque servomotors are used to move the robot's eight hinge joints. However, within the current study, the robot was simplified by assuming that the knee joints are frozen: all four legs are held straight when the robot is commanded to perform some action. the subsequent table gives the overall dimensions of the robot's parts.

Table 1: Overall dimensions of robot

Parameter	Value (mm)
Width and length of the body	140
Height of the body	85
Length of the upper leg	95
Height of the upper leg	26
Length of the lower leg	125
Diameter of the foot	12

All eight servomotors are controlled using an on-board PC-104 computer via a serial servo control panel SV-203B, which converts serial commands into pulse-width modulated signals. Servo drives are capable of manufacturing a maximum of 200 ounce- inches of torque and 60 degrees per second of speed. The actuation ranges for all of the robot's joints are summarized within the following table.

Table 2: Actuation ranges

	Lower range bound (degrees)	Upper range bound (degrees)
Hip joint	-96	+74
Knee joint	-96	+94

This four-legged robot can automatically synthesize a predictive model of its own topology (where and the way its body parts are connected), then successfully move around. It also can use this "proprioceptive" sense to determine if a component has been damaged, then model new movements that take the damage into account.

The robot is provided with a suite of different sensors polled by a 16-bit 32- channel PC-104 Diamond MM-32XAT data acquisition board. For the present identification task, three sensor modalities were used: an external sensor was wont to determine the left/right and forward/back tilt of the robot; four binary values indicated whether a foot was touching the ground or not; and one value indicated the clearance distance from the robot's underbelly to the ground, along the traditional to its lower body surface. All sensor readings were conducted manually, however all three sorts of signals will be recorded in future by on-board accelerometers, the strain gauges built into the lower legs, and a belly-mounted optical distance sensor for the robot.

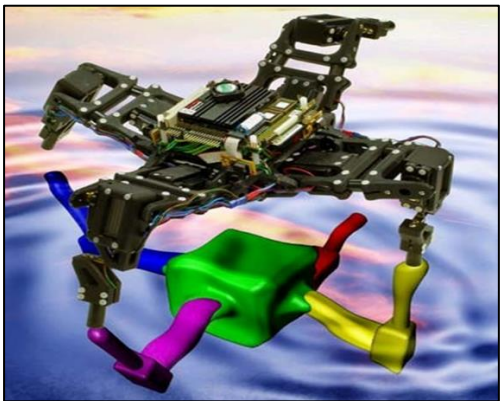

Figure 3: The starfish robot with reflection

3.3 Self Modelling Briefly

Here, its explained how the four-legged robot automatically synthesizes a predictive model of its own topology (where and the way its body parts are connected) through limited yet self-directed interaction with its environment, then uses this model to synthesize successful new locomotive behavior before and after damage. These findings may help develop more robust robotics, also as shed light on the relation between curiosity and cognition in animals and humans

A constantly shifting and unclear environment is a robot's worst nemesis. Typically, robots depend upon internal maps (either provided or learned), and sensory data to orient themselves with reference to that map and to update their location. If the environment is changing or noisy, the robot has got to navigate under uncertainty, and constantly update the possibilities that a particular action will achieve a particular result. things become's even worse if the robot's own shape and configuration can change, that is, if its internal model becomes inaccurate. In most cases, such an occasion constitutes the end of that particular robot's adventure.

The ability of robotic systems to model their surroundings on their own has advanced significantly, but nothing is known about how a robot may learn its own morphology, which cannot be deduced by direct observation or retrieved from a database of previous experiences. Robotic systems without internal models can synthesis increasingly complicated behaviors or recover from damage through physical trial and error, but this requires hundreds or thousands of tests on the physical machine and is typically too slow, expensive to run, or hazardous. Here, we describe a lively process that allows a machine to sustain performance through an autonomous and continuous process of self-modeling. A robot is in a position to indirectly infer its own morphology through self-directed exploration and then use the resulting self-models to synthesize new behaviors. If the robot's topology unexpectedly changes, the identical process restructures its internal self-models, resulting in the generation of qualitatively different, compensatory behavior. In essence, the method enables the robot to continuously diagnose and recover from damage. Unlike other approaches to wreck recovery, the concept introduced here doesn't presuppose built-in redundancy, dedicated sensor arrays, or contingency plans designed for anticipated failures. Instead, our approach is predicated on the concept of multiple competing internal models and generation of actions to maximize disagreement between predictions of these models. the method is composed of three algorithmic components that are executed continuously by the physical robot while moving or at rest (Fig. 2.3): Modeling, testing, and prediction.

The robot first executes an arbitrary motor activity and collects the resulting sensory data as the first phase of self-healing (Fig. 2.3A). The model synthesis component (Fig. 2.3B) then synthesizes a group of 15 candidate self-models using stochastic optimization to explain the observed sensory-actuation causal relationship. The action synthesis component (Fig. 2.3C) then uses these models to seek out a new action most likely to elicit the most information from the robot. this is often accomplished by searching for the actuation pattern that, when executed on each of the candidate self models, causes the foremost disagreement across the predicted sensor signals. This new action is performed by the physical robot (Fig. 2.3A), and therefore the model synthesis component now

reiterates with more available information for assessing model quality. After 16 cycles of this process have terminated, the foremost accurate model is used by the behavior synthesis component to create a desired behavior (Fig. 2.3D) which will then be executed by the robot (Fig. 2.3E).

If the robot detects unexpected sensor-motor patterns or an external signal as a result of unanticipated morphological change, the robot reinitiates the alternating cycle of modeling and exploratory actions to supply new models reflecting the change. The new most accurate model is now wont to generate a new, compensating behavior to recover functionality. an entire sample experiment is shown in Fig. 2.4.

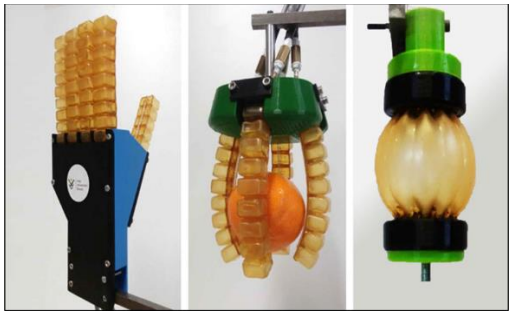

Figure 4: Self-Healing Robot

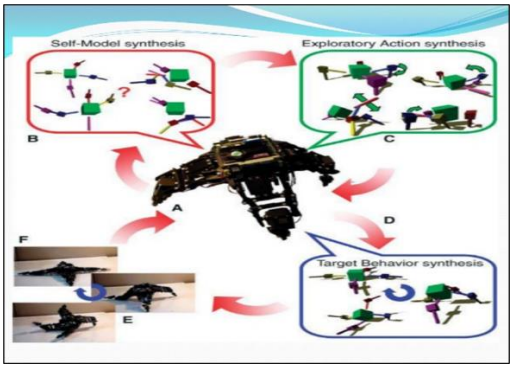

Figure 5: Outline of the Algorithm

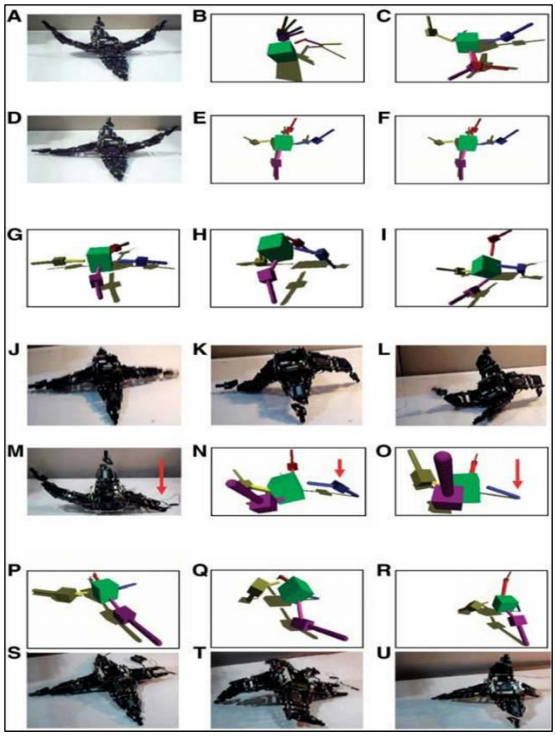

Figure 6: Robot modelling and behavior

A four-legged physical robot with eight motorized joints, eight joint angle sensors, and two tilt sensors was used to evaluate the suggested procedure. Any planar topological arrangement with eight limbs was included in the space of potential models, including chains and trees (for examples, see Figs. 2.3 and 2.4). After damage occurs, the space of topologies is fixed to the previously inferred morphology, but the dimensions of the limbs can be scaled (Fig. 2.4, N and O). The desired angles that the motors were instructed to achieve were included in the space of potential actions. The specific simulations used here might be replaced by a variety of other self-model representations, such as artificial neural or Bayesian networks, and additional sense modalities, such as pressure and acceleration, could be utilised (here the joint angle sensors were used only to verify achievement of desired angles and orientation of the main body was used only for self-model synthesis). Nonetheless, the utilization of implicit representations such as artificial neural networks—although more biologically plausible than explicit simulation—would make the validation of our theory more challenging, because it might be evaluating the model's accuracy is challenging (which can be done by visual inspection for explicit simulations). More important, without a particular representation, it's difficult to reward a model for a task such as forward locomotion (which requires predictions about forward displacement) when the model can only predict orientation data.

Table 3: Results of baseline algorithms

	Baseline 1	Baseline 2	Model-driven algorithm
Before damage			
Independent experiments (n)	30	30	30
Physical actions per experiment	16	16	16
Mean model evaluations (n = 30)	262,080 ± 13,859	246,893 ± 17,469	262,024 ± 13,851
Successful self-models	7	8	13
Success rate	23.3%	26.7%	43.3%
Mean model error (n = 30)	9.62 ± 1.47 cm	9.7 ± 1.45 cm	7.31 ± 1.22 cm
After damage			
Independent experiments (n)	30	30	30
Physical actions per experiment	16	16	16
Mean model evaluations (n = 30)	292,430 ± 44,375	278,140 ± 37,576	296,000 ± 22,351
Mean model error (n = 30)	5.60 ± 2.98 cm	4.55 ± 3.22 cm	2.17 ± 0.55 cm

The proposed process (Model-driven algorithm) was compared with two baseline algorithms, both of which use random instead of self-model–driven data acquisition. All three algorithm variants used an identical amount of computational effort (~250,000 internal model simulations) and therefore the same number (16) of physical actions (Table 2.3). within the first baseline algorithm, 16 random actions were executed by the physical robot (Fig. 1A), and therefore the resulting data were supplied to the model synthesis component for batch training (Fig. 2.3B). within the second baseline algorithm, the action synthesis component output a random action, instead of searching for one that created disagreement among competing candidate self-models. The actions related to Fig. 2.3, A to C, were cycled as within the proposed algorithm, but Fig. 2.3C output a random action, instead of an optimized one.

Before damage, the robot began each experiment with a group of random models; after damage, the robot began with the simplest model produced by the model-driven algorithm (Fig. 2.4F). it had been found that the probability of inferring a topologically correct model was notably higher for the model-

driven algorithm than for either of the random baseline algorithm (Table 2.3), which the final models were more accurate on average in the model-driven algorithm than in either random baseline algorithm (Table 2.3). Similarly, after damage, the robot was better ready to infer that one leg had been reduced in length using the model-driven algorithm than it could, using either baseline algorithm. this means that alternating random actions with modeling, compared with simply performing several actions first then modeling, doesn't improve model synthesis (baseline 2 does not outperform baseline 1), but a robot that actively chooses which action to perform next on the idea of its current set of hypothesized self-models has a better chance of successfully inferring its own morphology than a robot that acts randomly (the model-driven algorithm outperforms baseline algorithms 1 and 2).

Because the robot is assumed to not know its own morphology a priori, there's no way for it to determine whether its current models have captured its body structure correctly. it had been found that disagreement among the current model set (information that is available to the algorithm) is a good indicator of model error (the actual inaccuracy of the model, which isn't available to the algorithm), because a direct correlation exists between model disagreement and model error across the (n = 30) experiments that use the model-driven algorithm (Spearman rank correlation = 0.425, $P < 0.02$). Therefore, the experiment that resulted within the most model agreement (through convergence toward the correct model) was determined to be the most successful from among the 30 experiments performed, and therefore the best model it produced (Fig. 2F) was selected for behavior generation. This was also the starting model that the robot used when it suffered unexpected damage (Table 2.3).

The behavior synthesis component (Fig. 2.3D) was executed repeatedly with this model, starting whenever with a different set of random behaviors. The anticipated distance and actual distance are not exactly in line, but there is a definite upward trend that is missing from the random behaviors. This means that this automatically generated self-model was sufficiently predictive to allow the robot to consistently develop forward motion patterns without further physical trials. The transferal from the self-model to reality wasn't perfect, although the gaits were qualitatively similar; differences between the simulated and physical gait were presumably due to friction and kinematic bifurcations at symmetrical postures, both difficult to predict. Similarly, after damage, the robot was ready to synthesize sufficiently accurate models (an example is given in Fig. 2.4 O) for generating new, compensating behaviors that enabled it to continue moving forward.

4. ALGORITHM

For both robotics and electronic circuits, a number of techniques based on iterative testing for mistake recovery have been developed and demonstrated. Repeated hardware trials take a lot of time, may necessitate quick repair (such as when solar panels are covered in rain), and continuously alter the robot's state, making damage diagnosis challenging. For these reasons, repeated generate-and-test algorithms for robotics are not recommended.

Recent developments in simulation have made it possible to automatically evolve the controller and morphology of simulated robots in tandem to produce certain behaviours. We employ evolutionary algorithms to co-evolve robot bodies and brains in this case, but we do so in reverse: rather than developing a controller given robot morphology, we evolve a root morphology given a controller. Also, instead of evolving to reach a high fitness as a form of design, we evolve towards an observed low fitness (caused by some unknown failure) as a form of diagnosis. The evolutionary algorithm can adjust for damage or failure of the robot's mechanics, its sensory or motor apparatus, the controller itself, or some combination of these failure types by not distinguishing between the robot's morphology and controller. This contrasts with all existing automated recovery methods up to this point, which can only account for a few pre-specified faults.

In addition, qualitatively distinct behaviours (such hopping instead of walking) emerge in reaction to failure when recovery is accomplished via an evolutionary algorithm. The behaviours that are produced in response to minor damage by more conventional analytical methods are only marginally altered.

4.1 Algorithm overview

4.1.1 Estimation-Exploration Algorithm

The estimation-exploration algorithm is actually a co-evolutionary process comprising two populations. One population is of candidate models of the target system, where a model's fitness is decided by its ability to correctly explain observed data from the target system. the opposite population is of candidate unlabelled sentences, each of whose fitness is decided by its ability to cause disagreement among model classifications (thereby elucidating model uncertainties), or by exploiting agreement among models to realize some desired output (thereby capitalizing on model certainties).

The estimation-exploration algorithm performs two tasks: controller evolution and damage hypothesis evolution (the exploration phase). The algorithm also maintains a database, which stores pairs of data: an evolved controller and therefore the fitness produced by the 'physical' robot when that controller is used. Two separate evolutionary algorithms—the estimation EA and therefore the exploration EA–are used to generate hypotheses regarding the failure incurred by the physical robot, also as controllers for the simulated and 'physical' robot, respectively. Figure 3 outlines the flow of the algorithm, together with a comparison against an algorithm for evolving function recovery all on a physical robot.

A hybrid force-motion controller and a cycloidal motion path are proposed to deal with shape exploration. An adaptive exploration algorithm for segmentation of surface features and a predictor-corrector algorithm for exploration of deep features are introduced supported discrete impedance estimates. These estimates are derived from localized excitation of tissue including simultaneous force measurements. Shape estimation is validated in ex-vivo bovine tissue and attains surface estimation errors of but 2.5 mm with force sensing resolutions achievable with current technologies in minimally invasive surgical robots.

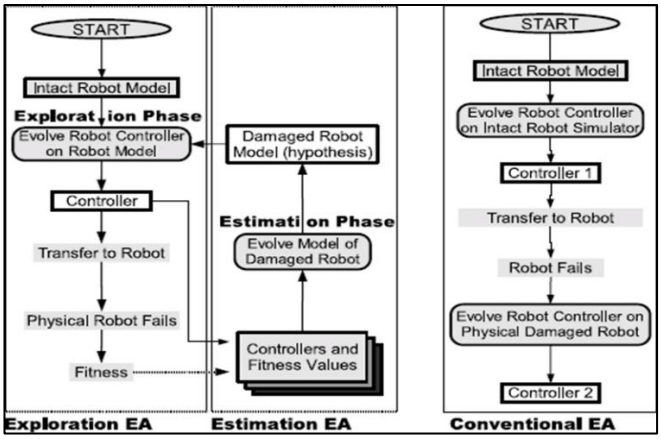

Figure 7: Flow chart of estimation-exploration phases

Exploration Phase: Controller Evolution.
The exploration EA is employed to evolve a controller for the simulated robot, such it is able to perform some task. the primary pass through this phase generates the controller for the intact physical robot: subsequent passes attempt to evolve a compensatory controller for the damaged physical robot, using the present best damage hypothesis generated by the estimation phase. When the exploration EA terminates, the simplest controller from the run is transferred to and used by the physical robot.
Physical Robot Failure: The physical robot uses an evolved controller to steer forwards. An unanticipated failure occurs to the robot, and therefore the broken robot records its own forward displacement for a period of time. The physical robot is then stopped, and therefore the recorded forward displacement (fitness) is inserted into the database along with the evolved controller on-board the robot at that time: these become an Input-output pair wont to reverse engineer the damage suffered by the robot. The damaged robot tries to function utilizing the compensating evolved controller created during the exploration phase throughout subsequent iterations of the algorithm.
Estimation Phase: Damage Hypothesis Evolution.
The estimation EA is employed to evolve a hypothesis about the actual failure incurred by the physical robot. The assessment EA tests the accuracy of each diagnosis represented by the estimation EA's genomes using the forward displacements created by the damaged physical robot and the related controllers operating on the physical robot at the moment. When the estimation EA terminates, the foremost fit damage hypothesis is supplied to the exploration EA. The robot simulator is updated to model this damage hypothesis: for instance if the hypothesis is that one of the legs has fallen off, that leg is broken off of the simulated robot. The investigation EA uses this revised approach to develop a compensatory controller.

4.2 Experimental Setup
The recovery of locomotion of badly damaged legged robots was accomplished using the suggested technique. A robot simulator is employed to evolve controllers for the 'physical' robot: here the 'physical' robot is also simulated. The physical robot receives evolved controllers from the simulation, and the simulation receives performance data from the physical robot. The robot simulator is predicated on Open Dynamics Engine, an open-source 3D dynamics simulation package. The simulated robot consists of a series of three-dimensional objects, connected with one degree-of freedom rotational joints.

4.2.1 The Robots
The two hypothetical robots tested in this preliminary work—a quadrupedal and hexapedal robot—is shown in Figure 4.2. There are eight mechanical degrees of freedom in the quadrupedal robot. Per leg, there are two rotational joints with one degree of freedom each: one at the shoulder and one at the knee. Four binary touch sensors, one in each of the lower legs of the quadrupedal robot, are present. The touch sensor returns, 1.0 if the lower leg is on the bottom and −1.0 otherwise. There also are four angle sensors in the shoulder joints, which return a sign commensurate with the flex or extension of that joint (−1.0 for max flexure up to 1.0 for max extension). Eight joints are controlled by a torsional motor. The joints have a maximum flex of 30 degrees from their original setting (shown in Figure 3), and a maximum extension of 30 degrees. The hexapedal robot has 18 mechanical degrees of freedom: each leg features a one degree-of-freedom rotational joint at the knee, and one two degree-of-freedom rotational joints connecting the leg to the spine. Each joint is actuated by a torsional motor, and therefore the joint ranges are the same as for the quadrupedal robot. The hexapedal robot contains six touch sensors, one per lower leg, and 6 angle sensors, placed on the joints connecting the legs to the spine.

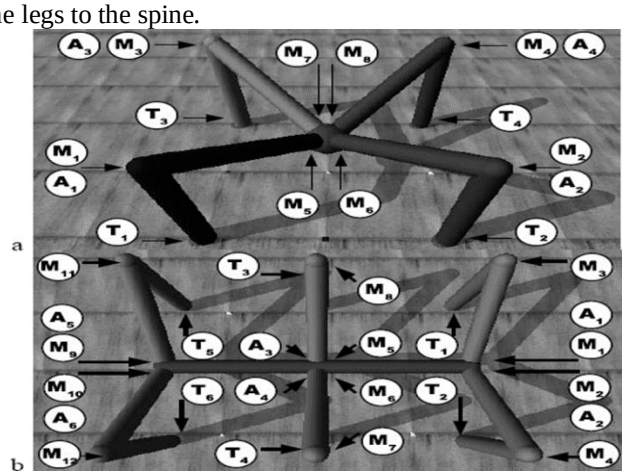

Figure 8: The simulated robots used for experimentation.

4.2.2 The Controllers
The robots are controlled by a neural network, which receives sensor data from the robot at the start of each time step of the simulation into its input layer, propagates those signals to a hidden layer, and eventually propagates the signals to an output layer. The neural specification and connectivity is shown in Figure 3.3. Neuron values and synaptic weights are scaled to dwell the range [−1.00, 1.00]. The neurons receive a threshold activation function. There's one output neuron for each of the motors actuating the robot: the values arriving at the output neurons are scaled to desired angles for the joint corresponding to that motor. For both robots here, joints can flex or reach π/ 4 away from their default starting rotation, π/ 2. The angles are translated into torques employing a PID controller, and therefore the simulated motors then apply the resultant torques. The physical simulator then updates the position, orientation and velocity of the robot supported these torques, together with external forces such as gravity, friction, momentum and collision with the bottom plane.

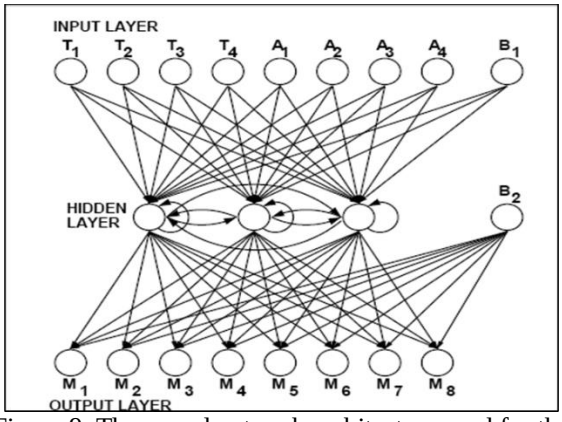

Figure 9: The neural network architecture used for the quadrupedal robot.

4.3 Algorithm Implementation

The Exploration EA: The exploration EA is employed to generate sets of synaptic weights for the robot's neural network (Figure 3.3). The fitness function awards robots for advancing as far as they can during the simulation's 1000 time steps. The fitness function is defined as f(gi) = d(t1000) - d(t1), where f(g) is the robot's fitness, expressed in metres, and whose neural network controller is labelled with the values contained in genome g. At the first time step of the simulation, the robot's forward displacement, or d(t1), was measured in metres. At the last time step of the simulation, d(t1000), it was measured in metres once more. The exploratory EA's genomes are a series of floating point data that encode.

The synaptic weights. For the quadrupedal robot, there are a complete of 68 synapses, giving a genome length of 68. For the hexapedal robot, there are a complete of 120 synapses, giving a genome length of 120. The encoded synaptic weights are represented to 2 decimal places, and dwell the range [-1.00, 1.00]. At the start of each run a random population of 100 genomes is generated. If there are any previously evolved controllers stored within the database, these are downloaded into the starting population. A genome is evaluated as follows: the encoded weights are wont to label the controller; the robot is then evaluated in the simulator for 1000 time steps using that controller; and the resulting fitness value is returned. Once all of the genomes within the population have been evaluated, they're sorted in order of decreasing fitness, and therefore the 50 least fit genomes are deleted from the population. Fifty new genomes are selected to exchange them from the remaining 50, using tournament selection, with a tournament size of three . Selected genomes undergo mutation: each floating-point value of the copied genome features a 1 per cent chance of undergoing a point mutation. 12 couples of the 50 newly created genomes are randomly chosen and go through one-point crossover.

The Estimation EA: The assessment EA develops theories for the failure that the physical robot experienced.

The genomes of the estimation EA, just like the exploration EA, are strings of floating-point values. Each genome within the estimation EA is composed of four genes: each gene denotes a possible failure. during this preliminary study, the particular robot can undergo three different types of damage—joint breakage, joint jamming, and sensor failure—and can incur zero to four of those damages simultaneously. In joint breakage, any single joint of the robot can break completely, separating the 2 parts of the robot connected by that joint. In joint jamming, the 2 objects attached by that joint are welded together: actuation has no effect on the joint's angle. In sensor failure, any sensor within the robot (either one among the touch or angle sensors) feeds a zero signal into the neural network during subsequent time steps. Any sort of failure that does not conform to one of these types is referred to henceforth as an unanticipated failure: so as to compensate for such cases, the estimation EA has got to approximate the failure using aggregates of the encoded failure types.

Each of the four genes encoded within the estimation EA genomes is comprised of four floating-point values, giving a complete genome length of 16 values. just like the exploration EA, each of the values is represented to 2 decimal places, and lies in [−1.00, 1.00]. the primary floating-point value of a gene is rounded to an integer in [0, 1] and denotes whether the gene is dormant or active. If the gene is dormant, the damage encoded by this particular gene isn't applied to the simulated robot during evaluation. Which of the three damage kinds should be administered to the simulated robot depends on whether the gene is active; this is indicated by the second floating point value, which is rounded to an integer in the range [0, 2]. The third value is scaled to an integer in the range [0, j 1], where j is the number of mechanical degrees of freedom of the robot (j = 8 for quadrupedal robots and j = 12 for hexapedal robots), depending on the damage type. The fourth value is currently not utilized in this preliminary study, but are going to be used for additional damage types that are not binary, but occur with a lesser or greater magnitude (i.e. a sensor that experiences 80% damage, rather than completely failing). The simulated robot is initially broken for each genome in the estimation EA in line with the failure scenario encoded in the genome. Then, the broken robot is appraised using the controller that was just developed by the exploration EA and tested on the "real" robot. The estimation's fitness function EA aims to reduce the discrepancy between the forward motion achieved by the 'real' robot using that controller and the forward motion achieved by the simulated robot utilising the encoded damage hypothesis. This is often based on the observation that the closer the damage hypothesis encoded in the genome is to the actual damage, the lesser the difference between the 2 behaviors.

During subsequent passes through the estimation phase, there are additional pairs of evolved controllers and forward displacements within the database: the controllers evolved by the exploration EA and therefore the fitness values attained by the 'physical' robot when using those controllers, respectively. In these cases, the simulated robot is evaluated once for every of the evolved controllers, and therefore the fitness of the genome is then the sum of the errors between the forward displacementsWhen the estimation EA terminates, the simplest evolved damage hypothesis is stored in a database: these hypotheses are used to seed the random population at the beginning of the next run of the estimation EA, instead of starting each time with all random hypotheses.

The estimation EA is analogous to the exploration EA, apart from the length of the genomes, what those genomes encode, and therefore the fitness function: in the exploration EA, forward displacement is maximized; within the estimationEA, error between the simulated and 'physical' robots' forward displacements is minimized.

Table 4: Damage scenarios tested

DAMAGE SCENARIOS TESTED	
Scenario	**Explanation**
1	One of the lower legs breaks off.
2	One of the entire legs breaks off.
3	One of the touch sensors fails.
4	One of the entire legs breaks off, and the touch sensor on one of the intact legs fails.
5	One of the angle sensors fails.
6	One of the upper-leg joints jams.
7	One of the entire legs breaks off, and one of the upper-leg joints jams on an intact leg.
8	One of the entire legs breaks off, one of the upper-leg joints jams on an intact leg, and one of the angle sensors fails on another intact leg.
9	Nothing breaks.
10	One of the hidden neurons fails.

4.4 Results of Estimation-Exploration Algorithm

This study looks at techniques that would let surgical slave robots examine the contours and stiffness of surgical fields on their own. The methods for determining the form and impedance properties of tissue are discussed in the paper, as well as techniques for independently examining perceived impedance during tool interaction inside a tissue cleft.

Control experiments were performed which conforms to the algorithm outlined within the right-hand panel of Figure all evolution is performed on the 'physical' robot after damage. during this case, controller evolution is performed by the exploration EA until generation 30 on the quadrupedal robot. The controller is then transferred to the 'physical' robot, which then undergoes separation of 1 of its lower legs (damage case 1). The exploration EA then continues on the 'physical' robot for an extra 70 generations. The algorithm proposed here was then applied several times to the quadrupedal and hexapedal robots. During each application of the algorithm, the robots suffered a special damage scenario: the 10 scenarios are listed in Table 3.1

For each run of the algorithm, the exploration EA is run once to get the initial evolved controller, then both the estimation and exploration EAs are run three times each after physical robot failure. Each EA is run 30 generations, employing a population size of 100 genomes. Twenty runs of the algorithm were performed (10 damage cases for every of the two robots), during which both the exploration and estimation EAs were initialized with independent random starting populations seed with any previously evolved controllers or damage hypotheses. Damage scenarios 1, 2, 3, 5 and 6 are often described by a single gene in the genomes of the estimation EA.

Scenarios 4, 7 and eight represent compound failures, and need more than one gene to represent them. Case 9 represents things when the physical robot signals that it has incurred some damage, when after all no damage has occurred. Case 10 represents an unanticipated failure: hidden neuron failure can't be described by the estimation EA genomes. Figure 6 shows the recovery of the quadrupedal robot after minor damage (scenario 3); after suffering unanticipated damage (scenario 10); and recovery of the hexapedal robot after severe, compound damage (scenario 8). The recovery of both robots for all 10 damage scenarios is shown in Figure.

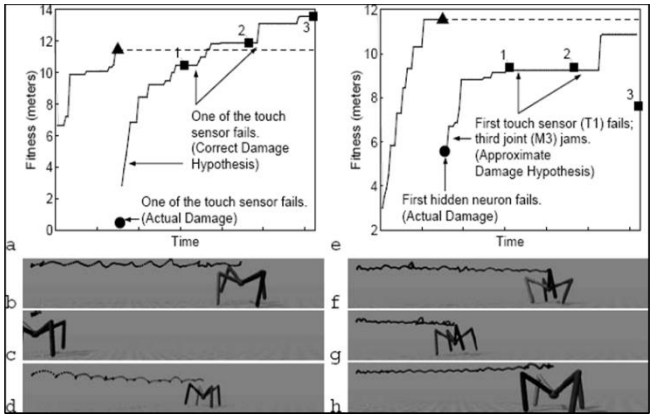

Figure 10: Three typical damage recoveries

4.5 Analysis of Estimation-Exploration Algorithm

It is often seen, that even after several generations have elapsed after the 'physical' robot suffers damage for the experiment , and 3550 hardware evaluations are performed, total function has not been restored. The degree of restoration (about 70%) is about the identical as that achieved by the quadrupedal robot suffering the same type of damage when the proposed algorithm is used to restore function. However the proposed algorithm only requires three hardware evaluations (more than two orders of magnitude fewer hardware trials) to revive function.

Figure 3.4, shows that for 3 sample damage scenarios, much functionality is restored to the physical robot after only three hardware trials. within the case of sensor failure for the quadrupedal robot, the forward displacement of the physical robot after the third hardware trial exceeds its original functionality. Frequently, the compensating controller results in a gait that is very dissimilar from that of the original, uninjured robot. Example: After recovering from a sensor failure, a robot moves in a more steady but irregular gait as opposed to before the failure (notice the distinct arcs in the trajectory of its centre of mass in Figure 3.4d) (Figure 3.4b).

The information-poor way of contrasting the behaviour of the physical robot with that of the simulated robot used in this paper, which compares behaviour only by measuring forward displacement, is thought to be the cause of the algorithm's intermittent failure. This method are going to be replaced in future with a more sophisticated method such as comparing sensor time series or measuring the differences in gait patterns.

The algorithm performs equally well for both morphological and controller damage: function recovery for scenarios 1 and a couple of (morphological damage) and scenarios 3 and 5 (controller damage), for both robots, approaches or exceeds original performance. Because the algorithm evolves the robot simulator itself supported the 'physical' robot's experience, it might be straightforward to generalize this algorithm beyond internal damage: the estimation EA could evolve not only internal damage hypotheses but also hypotheses regarding environmental change, like increased ruggedness of terrain or high winds.

4.6 Estimation-Exploration Algorithm Overview

1. Characterization of the target system

• Define a representation, variation operators and similarity metric for the space of systems.

• Define a representation and variation operators for the space of inputs (tests)
• Define a representation and similarity metric for the space of outputs

2. Initialization

• Create an initial population of candidate models (random, blank, or seeded with prior information)
• Create an initial population of candidate tests (random, or seeded with prior information)

3. Estimation Phase

• Evolve candidate models; encourage diversity
• Fitness of a model is its ability to explain all input-output data in training set

4. Exploration Phase

• Evolve candidate tests (input sets)
• Fitness of a test is the disagreement it causes among good candidate models
• Carry out best test on target system; add input/output data to training set

5. Termination

• Iterate estimation-exploration (steps 3-4) until the population of models converges on a sufficiently accurate solution, or the target system exhibits some desired behavior.
• If no model is found, the search space may be inappropriate, or the target system may be inconsistent
• If no good test is found, then either:
  – All good candidate models are perfect;
  – The search method for finding good tests is failing; or.
  – The target system may be partially unobservable.

6.Validation

• Validate best model(s) using unseen inputs
• If validation fails, add new data to training set and resume estimation phase.

5. FROM THEORY TO REALITY

Here, we present validation of our algorithm on a physical articulated robot (shown in Figure 1a): the robot evolves a particular model of its body using sensor data from different modalities. this is often the first time explicit, predictive robot models are intelligently synthesized based on physical interactions. In previous section, we've demonstrated that the EEA can synthesize both the topology and parameters of a hidden system, during which no model is required a priori (Bongard and Lipson, 2005a).

In order to apply the EEA to a new target system, like the physical robot used in this work, three preparatory steps must be first carried out: characterization of the system to be identified, how models are to be represented and optimized, and the way controllers are to be represented and optimized. Self-repair mechanisms in robots also means they're less of a risk to the humans that may be working with or near them, since they're unlikely to cause any real damage due to their gentle, flexible nature. This makes work environments with increasingly high populations of robotic 'workers' much safer for human workers and therefore the products.

By developing soft, safe materials embedded with polymers which may sense and locate damages and trigger the healing process without human interference, robots traditionally found as automated workers in factories and labs will soon begin to maneuver into more environments, like agriculture, and into the typical household. Users or owners of such robots will avoid the necessity to seek out costly repairs for damages sustained and can instead take advantage of this safe, sustainable alternative, putting this at the forefront of developing a replacement generation of robotics.

The cost of robot repairs and replacements is incredibly high, therefore the introduction of self-healing technology is not only game-changing, but can help to scale back the need for time-consuming, complex, and expensive repairs. It also has implications for sustainability, as robots will need to be replaced with new models less often, since they will perform standard maintenance independently and autonomously.

5.1 Characterizing the Target System

The target system in this study is a quadrupedal. The Details of the target system has been explained in section 2.2.1

5.2 Characterizing the Space of Models

Models are considered to be three-dimensional simulations of the physical robot (see Figure 5 for three model examples). The simulations are created within Open Dynamics Engine, a three-dimensional dynamics simulator. However in the current work only static identification is performed: the physical robot is commanded to achieve a static pose, and then hold still while sensor data is taken. Every candidate model (as well as the target robot) is assumed to start as a planar configuration of parts; when it begins to move, it can assume a three-dimensional configuration. The geometry and physical properties of the main body part is assumed to be known; the eight upper and lower leg parts are represented as solid cylinders. Each model is evaluated for an arbitrarily set time of 300 time steps of the simulator, which is enough time for most models to come to rest given an arbitrary motor program.

Models are encoded as either vectors or matrices, and these data structures are used to construct a possible articulated robot in the simulation environment. In the first set of experiments, it was assumed that everything about the physical robot is known except the lengths of its four legs. Models are therefore encoded as vectors containing eight real-valued parameters in [0, 1], with each value encoding the estimated length of one of the eight leg parts. We constrain the estimation about the minimum and maximum length of a leg to be between 2 and 40 centimeters, so each value is scaled to a real-value in [1,20]cm.

In the second set of results, we assume that less information about the robot is known: how the eight body parts attach to each other or the main body, and how the hinge joints connecting them are oriented. In that case, models are encoded as 8×4 real-valued matrices. Each row corresponds to one of the eight parts. The first value in row i is scaled to an integer in [0, i−1], indicating which of the previous body parts it attaches to; the second value is scaled to an integer in [0,3], indicating whether the current part attaches to the left, front, right, or back side of the parental part. The third value is scaled to an integer in [0,5], and indicates how the hinge joint connecting the current part to its parent operates: 0 and 1 cause the part to rotate leftward or rightward in response to a positive commanded joint angle (and rightward and leftward in response to a negative commanded angle); 2 and 3 cause the joint to rotate upward or downward in response to a positive commanded angle; and 4 and 5 cause the part to

rotate leftward or rightward around its own axis in response to a positive commanded angle. The fourth value is scaled to a value in [1, 20] cm to represent the length of the leg part.
In both types of experiments, a genetic algorithm using deterministic crowding (Mahfoud, 1995) is used to optimize the models. Genomes in the population are simply the vectors or matrices described above. The subjective error of encoded models is minimized by the genetic algorithm. Subjective error is given as the error between the sensor values obtained from the physical robot, and those obtained from the simulated one.
In the first pass through the estimation phase, a random population of models is generated, and optimized for a fixed number of generations. On the second and subsequent passes through the estimation phase, the previously optimized population of models is used as the starting point, but they are re-evaluated according to the new error metric with the additional set of sensor data.

5.3 Characterizing the Space of Controls

This paper presents a procedure for designing a strong controller which is able to stabilize a multivariable system with interval uncertainties in the system parameters. It is assumed that the uncertain parameters fall within predetermined ranges.
In this work a motor program is a set of four joint angles that either the target robot, or a model robot, is commanded to achieve2. Both the target and model robots begin during a planar configuration, with the joint angles at zero. Joint angles during a given motor program are selected randomly from the range [−30, 30] degrees. This constrains the range of motion of the target robot; without a model of itself, it's possible that the robot could perform some action that would be harmful to itself or complicate the inference process.
At the start of an identification run, a random motor program is generated, and sent to the target robot. Its motors are sufficiently strong to succeed in the desired angles. Once it reaches those angles it holds steady, and therefore the sensor data is taken, and fed into the EEA. The estimation phase then begins, as outlined above. When the estimation phase terminates, a replacement random motor program is generated. For this work, the exploration phase isn't used; i.e., a useful motor program isn't sought. Thus, the look for controllers is random.

5.4 Results: Parametric Identification

In the first set of experiments, only the lengths of the eight leg parts were identified: all other aspects of the target robot are assumed to be known. within the estimation phase, a population of 100 random models are created, and in each pass the population is evolved for 10 generations. a complete of four random motor programs are used; the population of models is optimized four times, whenever with an additional motor program and resulting set of sensor data from the target robot.
In total, 30 independent runs were conducted for every of seven experimental variants. Three or fewer of the sensor modalities are presumptively available in each variant for optimising the model. Within the first run, all three sensor modalities— touch, tilt and clearance—were assumed available for identification. within the second run, only touch and tilt information were available. When using simply touch and tilt information, it is frequently seen that the first run was more successful than the second since there was a sizable inaccuracy in the left leg's length estimation.

Across the seven versions, the optimised models' average quality was compared. It had been discovered that only the lean sensor data was necessary to create accurate models, with the mean difference between the model's leg length and the target robot's leg length serving as the criterion for model quality. This is often because for the experiment variants that included tilt information in calculating model quality, evolved models were more accurate than when tilt information was excluded from the calculation. The model quality is decided as the variance across the lengths of a single model's legs; in a good model all four legs should have the same length.

5.5 Results: Topological Identification

In the second set of experiments, the inference algorithm was required not only to spot the length of the robot's legs, but how the legs are attached to at least one another or to the main body, and where they're attached. In these experiments, parametric changes within the genome correspond to topological changes in the body plan of the robot model. during this more difficult task, the population size was expanded to 300, and every pass through the estimation phase was conducted for 40 generations
The figure 4.1, given below shows the results of a successful topological identification. Figures 4.1b-d show the simplest model obtained at the end of the first, fifth and ninth iteration, respectively. The final model achieves identical poses to the physical robot when given the same motor programme, such as the initial random motor programme, which demonstrates that the model is indeed predictive (compare Figures 4.1a and d).

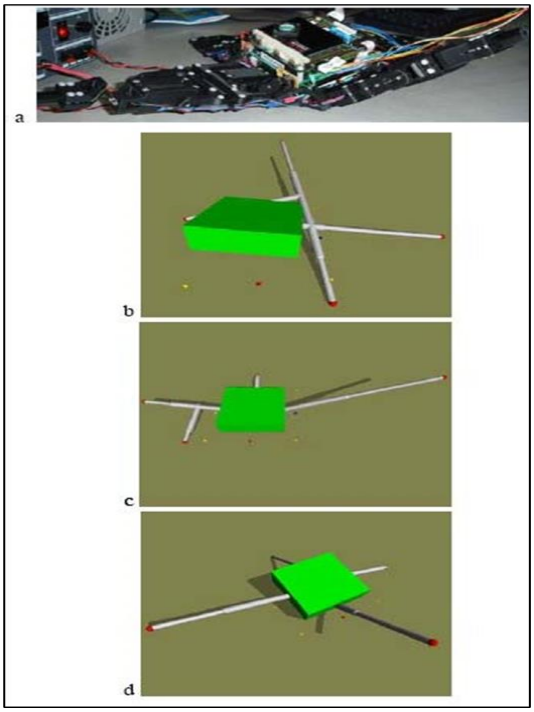

Figure 11: Results from a successful topological identification.

A: The pose produced by the physical robot as a result of running the first random motor program.
B: The best model produced after the first iteration of the run reported in figure 5

6. CONCLUSION

Although the likelihood of autonomous self-modeling has been suggested, here it had been demonstrated for the first time a physical system able to autonomously recover its own topology with little or no prior knowledge, also as optimize the parameters of those resulting self-models after unexpected morphological change. These procedures show topological self-modeling as well as parametric self-modeling. Future machines might be able to continuously recognise changes in their own morphology (such as after damage has occurred or when grasping a replacement tool) or the environment (such as when a robot enters an unfamiliar or modified environment) and use the inferred models to generate compensatory behaviour. Beyond robotics, the power to actively generate and test hypotheses can lead to general nonlinear and topological system identification in other domains, like computational systems, biological networks, damaged structures, and even automated science. apart from practical value, the robot's abilities suggest a similarity to human thinking because the robot tries out various actions to figure out the shape of its world.These findings may help develop more robust robotics, also as shed light on the relation between curiosity and cognition in animals and humans: Creating models through exploration, and using them to make new behaviors through introspection. Someday similar robots will someday respond not only to wreck to their own bodies but also to changes in the surrounding environment. Such responsiveness could lend autonomy to robotic explorers on other planets, a helpful feature, since such robots can't always be in touch with human controllers on earth.

7. BIBLIOGRAPHY

1. Bongard J., Zykov V., Lipson H. (2006), “Resilient Machines through Continuous Self-Modeling".

2. Adami C., (2006) "What Do Robots Dream Of?” Science Vol. 314.

3. Bongard J., Lipson H. (2004), “Automated Damage Diagnosis and Recovery for Remote Robotics”, IEEE International Conference on Robotics and Automation

4. Bongard J., Zykov V., Lipson H. (2006) “Automated Synthesis of Body Schema using Multiple Sensor Modalities”, Proceedings of the 10th Int. Conference on Artificial Life (ALIFE X).

5. Zykov V., Bongard J., Lipson H., (2004) "Evolving Dynamic Gaits on a Physical Robot", proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO'04

6. Lipson, H., Bongard, J., Zykov, V., Malone, E., (2006) “Evolutionary Robotics for Legged Machines: From Simulation to Physical Reality”, Proceedings of the 9th Int. Conference on Intelligent Autonomous Systems, University of Tokyo, Tokyo, Japan.

7. Bongard J., Lipson H. (2005) “Automatic Synthesis of Multiple Internal Models through Active Exploration”, AAAI Fall Symposium: From Reactive to Anticipatory Cognitive Embodied Systems, November 2005.

Websites

http://ccsl.mae.cornell.edu/research/selfmodels/

http://www.mae.cornell.edu/ccsl/papers/Science06_Bongard.pdf

http://www.mae.cornell.edu/ccsl/papers/Nature05_Zykov.pdf