Confronting the Challenge of Learning a Flexible Neural Controller for a Diversity of Morphologies

In: *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO 2013). New York, NY: ACM

Sebastian Risi
Sibley School of Mechanical and Aerospace Engineering
Cornell University
Ithaca, NY 14850, USA
sebastian.risi@cornell.edu

Kenneth O. Stanley
Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816, USA
kstanley@eecs.ucf.edu

ABSTRACT

The ambulatory capabilities of legged robots offer the potential for access to dangerous and uneven terrain without a risk to human life. However, while machine learning has proven effective at training such robots to walk, a significant limitation of such approaches is that controllers trained for a specific robot are likely to fail when transferred to a robot with a slightly different morphology. This paper confronts this challenge with a novel strategy: Instead of training a controller for a particular quadruped morphology, it evolves a special function (through a method called *HyperNEAT*) that takes morphology as input and outputs an entire neural network controller fitted to the specific morphology. Once such a relationship is learned the output controllers are able to work on a diversity of different morphologies. Highlighting the unique potential of such an approach, in this paper a neural controller evolved for three different robot morphologies, which differ in the length of their legs, can interpolate to never-seen intermediate morphologies *without any further training*. Thus this work suggests a new research path towards learning controllers for whole ranges of morphologies: Instead of learning controllers themselves, it is possible to learn the relationship between morphology and control.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning – connectionism and neural nets

General Terms

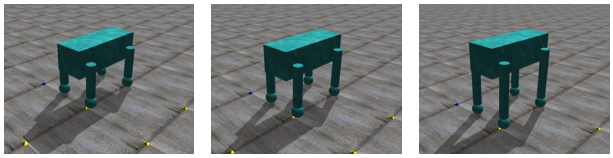
Algorithms

Keywords

HyperNEAT, NEAT, Neuroevolution, Legged Robots

1. INTRODUCTION

Legged robots have long captured the interest of researchers in robotics [12, 13, 15, 27]. Unlike their wheeled counterparts, they offer a high degree of mobility and excel on rugged terrain. However, designing a controller for a legged robot is challenging because of the high number of degrees of freedom within each leg and the need for tight coordination and balance.



(a) Small (b) Medium (c) Large

Figure 1: Quadruped Training Morphologies. The flexible neural controller trained on three quadrupeds with different leg segment lengths (0.25, 0.30, and 0.37 meters) learns to interpolate to never-seen intermediate morphologies without further training.

Controllers for legged robots are often crafted and tuned for a specific morphology, which can be a difficult and time-consuming task [27]. Therefore, interest has increased in recent years in *automatically* learning gaits for particular robots. In one seminal work, Hornby et al. [12] trained dynamic gaits (i.e. gaits that require transient instability during ambulation) for the AIBO robot dog with an evolutionary technique. The trained controller was actually included in the commercial release of the robot. In fact, a range of techniques have shown promise [5, 13, 15].

However, learned and hand-designed controllers both are likely to fail when the robot's morphology is altered, even if only slightly [12]. This brittleness contrasts starkly with the capabilities of legged animals in nature. For example, a foal can quickly learn to walk a short time after birth. As noted by Lewis and Bekey [15], it is unlikely that the locomotion controller in the foal is determined completely before birth or that any "*learning algorithm could program a nervous system ab initio with so few training epochs*". Thus rather than learning a controller for a particular morphology, an intriguing unexplored possibility is instead to learn a *relationship* between the morphology of a robot and its control. In fact, the ability to exploit such a relationship is a major goal for the field of robotics [15].

In this spirit, the main idea in this paper is to train a special function that takes morphology as input and outputs a neural network controller fitted to the specific morphology. The implementation of this idea is enabled by the *Hypercube-based NeuroEvolution of Augmenting Topologies* (HyperNEAT) method [23], which is well-suited to such an approach because it introduced the idea that a function can be evolved that takes domain information as input and outputs a controller. In particular, HyperNEAT evolves the pattern of weights across the geometry of an artificial neural network

GECCO'13, July 6-10, 2013, Amsterdam, The Netherlands.

(ANNs) as a function of the positions of its nodes in space. This capability allows large ANNs with regularities in connectivity to evolve for high-dimensional problems [9, 10, 23, 26]. Here it is extended to the idea that the pattern of connectivity can also depend on an indicator of the *morphology* of the robot being controlled.

In particular, the approach presented in this paper augments HyperNEAT not only to create ANNs as a function of the domain geometry but also based on the particular quadruped morphology. By evolving a function that can output controllers for three different morphologies (figure 1) with different leg lengths, HyperNEAT can actually learn the relationship between morphologies and control policies. In this paper, attempting to train in this way is shown to have the potential to interpolate to many intermediate morphologies *never seen in training*. No such interpolation is observed possible with the baseline static controllers, which work on almost no morphologies other than the ones seen in training.

Thus the main conclusion is that instead of learning controllers themselves for a particular morphology, it does appear feasible to learn the relationship between morphology and control, which can lead to less model-specific controllers.

2. BACKGROUND AND RELATED WORK

This section reviews prior working in machine learning for legged locomotion and then transitions to the Neuroevolution of Augmenting Topologies (NEAT) and HyperNEAT methods that underpin the approach in this paper.

2.1 Legged Locomotion

A range of techniques have shown promise in automating the achievement of effective gaits. Kimura et al. [13] trained a four-legged robot with a reinforcement learning approach. A more biologically inspired approach by Lewis and Bekey [15] allowed a specific quadruped robot to learn to walk in a short amount of time based on a system of distributed adaptive modules. In the context of evolutionary computation, some of the first experiments in legged locomotion were performed by Beer [2], who generated static gaits for a simulated hexapod. Recently Clune et al. [5] evolved coordinated quadruped gaits with HyperNEAT. However, prior approaches in general focused on training a robot for a *particular* morphology. Instead, the work presented here tries to learn the relationship between morphology and control, which has not been attempted so far.

2.2 Neuroevolution of Augmenting Topologies (NEAT)

NEAT starts with a population of simple neural networks and then *adds complexity* over generations by adding new nodes and connections through mutations. By evolving networks in this way, the topology of the network does not need to be known a priori; NEAT searches through increasingly complex networks to find a suitable level of complexity. Because it starts simply and gradually adds complexity, it tends to find a solution network close to the minimal necessary size. However, as explained next, it turns out that directly representing connections and nodes as explicit genes in the genome cannot scale up to very large networks. For a complete overview of NEAT see Stanley and Miikkulainen [21].

2.3 HyperNEAT

NEAT is called a *direct encoding* because each part of the solution's representation (i.e. each connection weight in the genome) maps to a single piece of structure in the final solution network [8]. The significant disadvantage of this approach is that even when different parts of the solution are similar, they must be encoded and therefore discovered separately. This challenge is related to the problem of learning controllers for multiple robot morphologies: After all, if individual controllers are encoded by separate genetic code, even if a component of their capabilities is shared, the learner has no way to exploit such a regularity. Thus HyperNEAT introduces an *indirect* encoding instead, which means that the description of the solution is compressed such that information can be reused, allowing the final solution to contain more components than the description itself. Indirect encodings allow solutions to be represented as a *pattern* of parameters, rather than requiring each parameter to be represented individually [3, 10, 20, 22]. HyperNEAT, reviewed in this section, is an indirect encoding extension of NEAT that is proven in a number of challenging domains that require discovering regularities [5, 6, 9, 10, 23]. For a full description see Stanley et al. [23] and Gauci and Stanley[10].

In HyperNEAT, NEAT is altered to evolve an indirect encoding called *compositional pattern producing networks* (CPPNs [20]) *instead* of ANNs. The CPPN in HyperNEAT plays the role of DNA in nature, but at a much higher level of abstraction; in effect it encodes a pattern of weights that is painted across the geometry of a network. Yet the convenient trick in HyperNEAT is that this encoding is *itself* a network, which means that CPPNs can be evolved by NEAT. A CPPN is a *composition of functions*, wherein each function loosely corresponds to a useful regularity. For example, a Gaussian function induces symmetry and a periodic function such as sine creates segmentation through repetition. In effect, the indirect CPPN encoding can compactly encode patterns with regularities such as symmetry, repetition, and repetition with variation [19, 20]. Thus as the CPPN increases in complexity, it encodes increasingly complex amalgamations of regularities and symmetries that are projected across the connectivity of a network [9, 10].

Unlike in many common ANN formalisms, in HyperNEAT neurons exist at *locations* in space. That way, connectivity is expressed across a geometry, like in a natural brain. Formally, CPPNs are *functions* that input the locations of nodes (i.e. the *geometry* of a network) and output weights between those locations. That way, when queried for many pairs of nodes situated in n dimensions, the result is a topographic connectivity pattern in that space. Consider a CPPN that takes four inputs labeled x_1, y_1, x_2, and y_2; this point in four-dimensional space *also* denotes the connection between the two-dimensional points (x_1, y_1) and (x_2, y_2), and the output of the CPPN for that input thereby represents the weight of that connection (figure 2). Because the connections are produced by a function of their endpoints, the final structure is produced with *knowledge* of its geometry. In effect, the CPPN is painting a pattern on the inside of a four-dimensional hypercube that is interpreted as the isomorphic connectivity pattern, which explains the origin of the name *hypercube-based NEAT* (HyperNEAT). Connectivity patterns produced by a CPPN in this way are called *substrates* so that they can be verbally distinguished from the CPPN itself.

Each queried point in the substrate is a node in an ANN. The experimenter defines both the location and role (i.e. hidden, input, or output) of each such node. As a rule of thumb, nodes are placed on the substrate to reflect the geometry of the task [5, 23]. That way, the connectivity of the substrate is a function of the task structure. For example, the sensors of a robot can be placed from left to right on the substrate in the same order that they exist on the robot. Outputs for moving left or right can also be placed in the same order, allowing HyperNEAT to understand from the outset the correlation of sensors to effectors. In this way, knowledge about the problem geometry can be injected into the search and HyperNEAT can exploit the regularities of a problem that are invisible to traditional neural network encodings.

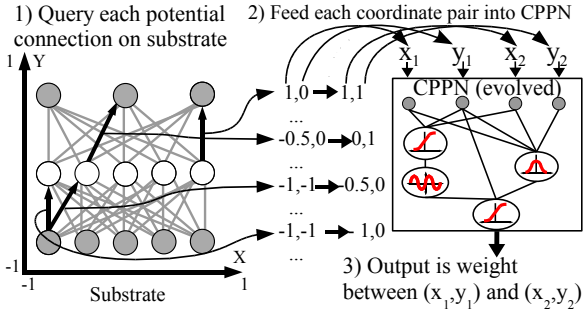


Figure 2: HyperNEAT Geometric Connectivity Pattern Interpretation. A collection of nodes, called the *substrate*, is assigned coordinates that range from -1 to 1 in all dimensions. (1) Every potential connection in the substrate is queried to determine its presence and weight; the dark directed lines in the substrate depicted in the figure represent a sample of connections that are queried. (2) Internally, the CPPN (which is evolved) is a graph that determines which activation functions are connected. As in an ANN, the connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection. For each query, the CPPN takes as input the positions of the two endpoints and (3) outputs the weight of the connection between them. Thus, CPPNs can produce regular patterns of connections in space.

In summary, the capabilities of HyperNEAT are important for the approach presented in this paper because they provide a formalism for producing policies (i.e. the output of the CPPN) as a function of geometry (i.e. the inputs to the CPPN). As explained next, not only can such an approach produce a single network but it can also produce a set of networks that are each generated not only as a function of the the domain geometry (as in the original HyperNEAT) but also of the robot's morphology.

3. APPROACH: LEARNING A RELATIONSHIP BETWEEN MORPHOLOGY AND CONTROL

The main idea in this work is that instead of training a particular controller for a particular morphology, evolution will learn a CPPN that takes morphology as input and outputs an entire neural network controller fitted to the specific morphology. The hope for such an approach, which has not been demonstrated previously, is that once such a relationship is learned the output controllers are able to work on a diversity of different quadruped morphologies.

To understand how a diversity of morphologies will be encoded by the same CPPN, it helps to begin by considering how a single CPPN encodes a single quadruped controller.

3.1 Static Quadruped Neural Controller

In this paper, a three-dimensional quadruped robot (figure 1) in a realistic physics simulation using the freely available Open Dynamics Engine (see `http://www.ode.org`), is controlled by a type of ANN called a *continuous-time recurrent neural network* (CTRNN) that is able to express the non-linear dynamics found in natural gaits and is common in other legged robot experiments [16, 18]. The quadruped robot (figure 1) has a total of twelve degrees of freedom (DOF): two degrees in each hip joint (pitch and roll) and one degree in each knee joint (pitch). Table 1 gives some of the physical parameters of the quadruped model that are the same for all approaches.

Table 1: Quadruped Simulation Parameter Settings

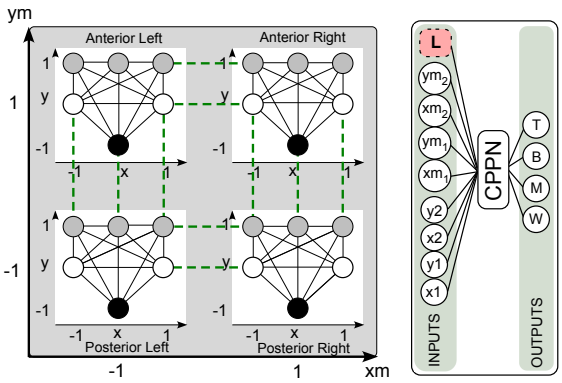
Parameter	Value
Maximum Torque	5.0 newton meters
Proportional Constant	9.0
Torso Length	0.4 meters
Torso Width	0.2 meters
Torso Density	1.0 kilograms per cubic meter

The neural architecture for the quadruped is distributed into separate substrate modules for each of the four legs (figure 3). Similar decentralized architectures have been applied by Téllez et al. [24] to generate the walking behavior of an Aibo robotic dog, are observed in walking biological organisms [17], and also have inspired hand-designed control architectures for hexapod robots [4].

Each leg module has one input, two hidden, and three output nodes. The inputs provide the current angle of the anterior-posterior hip joint. The ANN outputs movement requests for each degree of freedom in the model, i.e. for each independent axis of rotation for all joints in the model. The outputs are scaled to match the angular range of the corresponding DOF, which is interpreted as the angle that the neural network is requesting. The difference between the requested angle and the current orientation of the DOF denotes the disparity between the state the network is requesting and the current state of the model. A proportional controller applies torque to reduce this disparity. In other words, the neural network directs the low-level controllers towards a particular state. This method of control are similar to those in Reil and Husbands [18] and Lehman and Stanley [14].

To generate a controller for the quadruped, a CPPN (figure 3b) with inputs x_1, y_1, x_2, y_2, x_m_1, y_m_1, x_m_2, and y_m_2 first queries each of the four substrates shown in figure 3a to determine the intra-module connection weights between the input/hidden, hidden/hidden, hidden/output, and output/output nodes of each module (determined by CPPN output W). That is, inputs x and y describe the internal position of each node inside their corresponding module, whereas the x_m and y_m inputs determine the position of the module itself (e.g. the module controlling the anterior left leg is located at ($x\_m = -1$, $y\_m = 1$)) within the larger substrate. The CPPN can thus emphasize the influence of x_m and y_m for increasing heterogeneity or minimize it to produce greater homogeneity between different leg modules. Subsequently, the inter-module connections between corresponding neighboring hidden and output neurons are determined (dotted lines in figure 3a) by CPPN output M. This approach to encoding separate modules within a single substrate is inspired by the multiagent HyperNEAT approach [7], which showed how several similar ANNs can be encoded by the same CPPN. However, an added idea here is also to encode connections *between* such modules, to help them synchronize in time.

Additionally the bias and time constant values for each *hidden* node are determined by CPPN outputs T and B. By convention those values are determined by a node-centric query at (x_1, y_1, x_m_1, y_m_1, 0.0, 0.0, 0.0, 0.0), where x_2, y_2, x_m_2, y_m_2 are simply set to zero. The time constants and bias values for the *output* nodes are fixed for all approaches to 0.1 and 3.0, respectively. These values have been found to work well for a variety of different morphologies. Most importantly, the key idea in this section is that the CPPN takes a set of geometric parameters and outputs a connectivity pattern for a network.



(a) Substrate (b) CPPN

Figure 3: Distributed Substrate Architecture and CPPN. The quadruped architecture is divided into four different substrate modules (a). Each module is responsible for controlling a single leg, where the inter-module connections allow neighboring nodes in different modules to communicate (dotted line). Each leg module has one input (black), two hidden (white), and three output nodes (gray). The CPPN (b) takes the internal position of each node inside their corresponding module $(x.y)$ and the the position of the module itself (x_m, y_m) as input to generate the intra-module connections (determined by output W), inter-module connections (output M), time constants (output T), and node biases (output B). The flexible neural controller additionally receive the current leg length L as input when determining the time constant and bias values of the hidden nodes.

Following Beer [1] the activation of each neuron in the CTRNN is determined by the following equation:

$$y_i^{t+1} = y_i^t + \frac{T}{\tau_i}(-y_i^t + \sum_{j=1}^{N} w_{ji}\sigma(y_j^t + \theta_j)), \qquad (1)$$

where:

- y_i^{t+1} is the activation of node i at time step $t+1$.
- y_i^t is the activation of node i at time step t.
- τ_i is the time constant of node i determined by the CPPN and scaled into the range $[0.1, 2.0]$.
- θ_j is the bias for node i determined by the CPPN and scaled into the range $[-3.0, 3.0]$.
- T is the time slice and set to 0.01 for the experiments in this paper.
- w_{ji} is the weight between node i and j where $w \in [-5.0, 5.0]$.
- σ is the sigmoid activation function.

3.2 Flexible Neural Controller

A flexible neural controller poses a greater challenge; how can a single CPPN encode a set of different architectures for different quadruped morphologies, all related but requiring slightly different gaits to locomote?

The main idea is that a single CPPN is able to create different ANNs based on both the robot's internal geometry and its current leg length. While the connectivity pattern of the network is generated as described in the previous section, now the *leg length* is provided as an additional input to the CPPN for the node-centric queries of the time constants and biases of the hidden nodes. These neurons function as the main Central Pattern Generators (CPGs) of the quadruped, controlling the rhythmic movement of the legs of the robot. CPGs have been suggested as important functional units of the central nervous system [11] The hypothesis in this work is that given the right neural connectivity, a change in the time constants can modulate the gait of the robot to fit a certain morphology. Additionally, by supplying the CPPN with a morphology-dependent input (e.g. leg length) and evaluating it on a variety of different morphologies, it should be able to learn a relationship between morphology and control.

While the CPPN in this paper is only augmented with a single additional input, in principle more inputs could be added (i.e. joint ranges, maximum torque, torso densities, etc.), allowing the CPPN to produce an even wider range of different controllers for different robot morphologies.

A key question about such a flexible neural controller is whether the CPPN can also in principle generate controllers for intermediate leg scales on which it was not trained. This capability would allow one CPPN to produce neural networks for a variety of different morphologies without further training. The controllers would be able to interpolate between the policies of the learned morphologies, thereby allowing less model-specific controllers than have heretofore been possible.

3.3 Experimental Setup

Because initial random controllers for legged robots and all of their immediate perturbations tend to fall they provide a bad gradient for an objective-driven evolutionary search [14, 25]. Therefore, this paper employs a variant of evolutionary search called *novelty search* that has been shown to overcome such deception in locomotion-based problems in the past [14]. Novelty search is rewarded when it finds *novel* behaviors and whenever a significantly novel gait is discovered, it is recorded into a permanent archive that characterizes the distribution of prior solutions in behavior space. A detailed description of novelty search can be found in Lehman and Stanley [14]. For the purposes of this study, it is the chosen learning approach for both static and flexible controllers. In the future, when more is known about the proper incentives to encourage flexible controllers to evolve consistently, a more targeted methodology may be warranted. However, at present novelty search, which is agnostic about the specific objective of the search, is a good choice for an initial exploration of what is possible.

To investigate the effect of flexible evolving controllers, both static and flexible neural architectures are trained and tested. Each neural controller is evaluated on three robot morphologies (figure 1) with leg segment lengths of 0.25, 0.30, and 0.37 meters. Each evaluation lasts for the duration of 15 simulated seconds and is terminated if the robot falls.

Creating controllers for morphologies not seen in training is challenging because the new quadruped robots must be assigned gaits automatically. Yet such a capability could be important to learning controllers for whole ranges of morphologies. To test this capability, the best CPPN of each generation of training is tested on several different quadrupeds with intermediate leg sizes without further learning.

3.3.1 Experimental Parameters

The size of each HyperNEAT population was 300 with 10% elitism and a termination criterion of 600 generations. Sexual offspring (75%) did not undergo mutation. Asexual offspring (25%) had 0.6 probability of weight mutation, 0.06 chance of link addi-

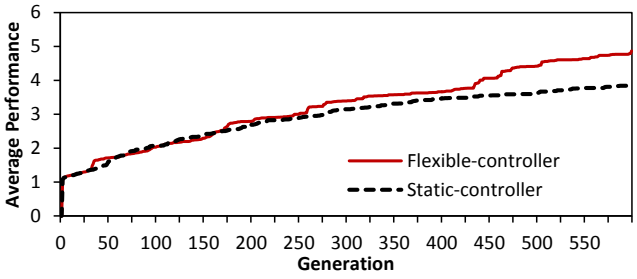


Figure 4: Average Training Performance. The average best fitness over generations is shown for the static and flexible neural controller. The flexible methods performs slightly (though not significantly) better than the static approach.

tion, and 0.005 chance of node addition. The available CPPN activation functions were sigmoid, Gaussian, absolute value, cosine, and sine, all with equal probability of being added.

4. RESULTS

Results were collected from 25 runs of static-controller evolution and 25 runs of flexible-controller evolution. The main question is whether the flexible neural controller's ability to potentially learn a relationship between morphology and control benefits its performance when transferred to never-seen intermediate morphologies.

In training on the three morphologies shown in figure 1 both methods perform similarly (figure 4). The final performance on these three cases with the flexible method is 4.88 meters traveled on average (σ = 3.15) in the allocated time, which is slightly (though not significantly according to the Student's T-test) better than the static method, which travels 3.84 meters on average (σ = 1.21). Interestingly, a significant difference between the two methods is their standard deviation ($p < 0.0001$ according to the F-test). This result suggests that the *best* controllers possible are more likely to be found at the tail of the distribution of flexible controllers, which is where the potential for genuine interpolation may be confirmed.

Indeed, the *best gait* discovered by the dynamic method traveled 12.39 meters (averaged over the three training morphologies), while the best gait discovered by the static method traveled only 6.24 meters. Figure 5 shows the neural dynamics of the best evolved gait from the flexible setup for the medium sized robot (leg segment length = 0.30 meters). Evolution discovered a dynamic trot gait, which is characterized by the motor outputs of the opposing legs being in antiphase with each other.

4.1 Interpolation Performance

More interestingly, the key question is whether any such controllers can work on morphologies never seen in training. Recall that the flexible neural approach encodes multiple controllers as a function of their morphology. To test the capability to interpolate to unseen morphologies, leg segment increments of 0.01 meters in the interval from 0.25 (small training morphology) to 0.37 meters (large training morphology) were attempted. The threshold for passing the interpolation test is based on the qualitative behavior of the evolved solutions and defined as traveling at least 6.00 meters in the allotted amount of time.

An interesting discovery is that two flexible controllers are indeed able to pass this threshold in 12 and 11 out of 13 morphologies, respectively. In contrast, the best discovered static controller

Table 2: Training and Interpolation Results

Training performance	**Static**	**Flexible**
Average Distance Traveled	3.84 (σ = 1.21)	4.88 (σ = 3.15)
Max Distance Traveled	6.24	12.39
Interpolation performance	**Static**	**Flexible**
# Solved Average	0.52 (σ = 0.71)	2.16 (σ = 3.68)
# Solved Best	2	12

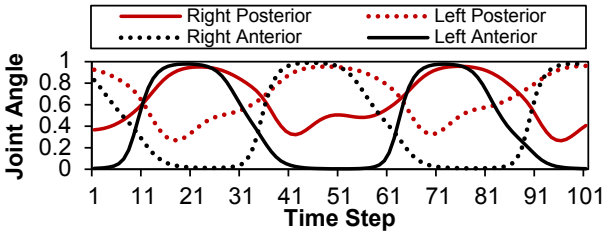


Figure 5: Motor Neuron Dynamics. The neural dynamics of the four hip outputs (pitch) controlling a medium quadruped with 0.30 meters leg segment length are shown. The dynamics of the opposing legs (e.g. anterior left and anterior right) are directly in antiphase with each other, which results in a dynamic trot-like gait.

fails for 11 out of the 13 morphologies. Reflecting the high variance in solutions, the flexible method solves on average 2.16 intermediate morphologies (σ = 3.68), while the static approach only solves 0.52 on average (σ = 0.71), which is statistically significant ($p < 0.05$).

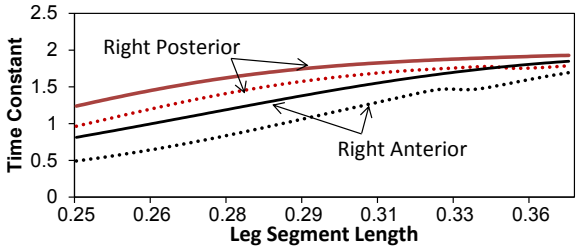
Thus the results (summarized in table 2) do confirm that such relationships exist and can be discovered. The CPPN that outputs these controllers for different leg lengths is the first network interpolator of its kind of which the authors are aware. Videos of this best evolved neural network controller are available at: `http://youtu.be/oLSSt5GyHNk`.

An important question is why the best discovered flexible controller that works on 12 out of 13 morphologies, fails on an intermediate leg segment length of 0.35 meters. Interestingly, varying the *leg length* CPPN input slightly (from 0.35 to 0.355), increases performance on this intermediate segment length from 2.11 to 11.18 meters traveled. This result indicates that a working controller for this intermediate leg length exist, although it might be sometimes expressed at a slightly different position in the space of the CPPN encoded ANNs.

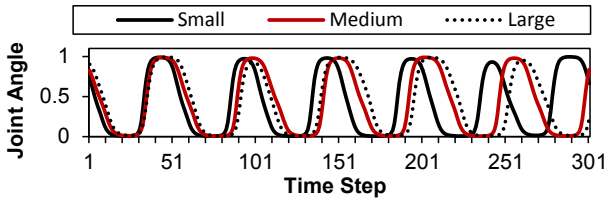
4.2 Relationship Between Morphology and Control

The better performance of the flexible approach stems from its ability to generate a neural controller fitted to a specific morphology. How is this relationship represented in the underlying architecture? Figure 6a shows a correlation between time constant values of the hidden nodes, which are determined through by the CPPN (see Section 3.2), and the change of the *leg length* input.

In particular, the time constant values of the hidden nodes exhibit a significant positive correlation (r = 0.97, $p < 0.01$) with the increased leg segment length. This change in time constant values results in an increased period of the hip (pitch) oscillation (figure 6b), thereby prolonging the leg stride of the robot (i.e. the larger the robot's legs, the larger the leg strides it has to make). This correlation suggests that the CPPN has actually learned a functional relationship between different robot morphologies and the network



(a) Relationship Between Time Constant And Leg Size



(b) Hip Output Pattern for Three Different Morphologies

Figure 6: Morphology-Dependant Change. The relationship between the leg length of the robot and the time constant values of the hidden nodes are shown in (a). An increase in leg sizes correlates clearly with an increase of the time constant values controlling the two right legs. The right anterior hip (pitch) output generated by the three training quadrupeds are shown in (b). The period of the oscillation increases with the length of the robot's legs, resulting in a prolonged leg stride.

Figure 7: Interpolated Locomotion Example. The composite image shows a quadruped robot with an interpolated leg segment length of 0.28 meters (which was never seen in training) successfully locomoting from left to right. In this example evolution discovered a dynamic trot gait, which is characterized by the motor outputs of the opposing legs being in antiphase with each other.

architectures that are necessary to control them. Figure 7 shows the successful interpolation to an intermediate leg length.

5. DISCUSSION AND FUTURE WORK

The major contribution of this work is conceptual. It introduces the idea that learning a relationship between morphology and control may ultimately be more practical than learning individual controllers. Yet to learn such a relationship requires a representation that *outputs* controllers, which is what a CPPN can do. An interesting further challenge is to design different representations than CPPNs that would enable alternative learning methods to HyperNEAT to learn such relationships as well.

While training a flexible CPPN-based controller-generator on three different morphologies might be expected to generate controllers on those three morphologies, the surprise in this paper was the discovery that the best such CPPNs actually can generate intermediate controllers as well for morphologies never seen in training. This discovery suggests that there is a tangible relationship between morphology and controller architectures that can be discovered and exploited, at least with quadrupeds. While the reward system in the experiment (i.e. novelty search) was not set up explicitly to encourage such interpolation, now that we know it is possible to do it, a further important task is to craft reward functions that indeed produce interpolating controllers consistently.

This task is more complex than it may sound; it is likely that simply rewarding several intermediate morphologies would present a deceptive landscape to the search algorithm, which is one reason novelty search is a good initial choice. However, now that such solutions are confirmed to exist, the prospect of devising a better reward scheme to find them consistently is more promising.

Perhaps in the future it will be possible to vary the morphology of certain canonical types of robots such as quadrupeds or bipeds and simply pop in a generic controller-generator that fashions an appropriate controller for the particular variant. Such a capability could one day liberate roboticists from the minutia of designing morphology-specific controllers to concentrate instead only on the best morphology for the job.

6. CONCLUSION

This paper contemplated the possibility that a relationship can be learned automatically by the computer between robot morphology and the appropriate controller architecture. To investigate this question, a special encoding called a CPPN was provided as input information about robot morphology and in turn output a morphology-specific controller. A significant discovery was that in some cases such CPPNs actually can output effective controllers for almost every intermediate morphology that had not been seen in training. Thus it appears that achieving such a capability is feasible and the technique for doing so will likely continue to be refined. Ultimately such flexible neural controllers promise to impact the field of robotics by moving away from body-specific controllers.

7. ACKNOWLEDGEMENTS

This work was supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD) and through grants from the US Army Research Office (Award No. W911NF-11-1-0489) and DARPA Computer Science Study Group Phase III (Award No. N11AP20003). This paper does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred.

References

[1] R. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509, 1995.

[2] R. Beer. The dynamics of adaptive behavior: A research program. *Robotics and Autonomous Systems*, 20(2):257–289, 1997.

[3] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proc. of the 2002 Congress on Evolutionary Computation*, 2002.

[4] R. Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Neural computation*, 1(2):253–262, 1989.

[5] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2009) Special Session on Evolutionary Robotics*, Piscataway, NJ, USA, 2009. IEEE Press.

[6] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 15(3):346–367, 2011.

[7] D. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley. Evolving policy geometry for scalable multiagent learning. In *Proc. of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 731–738, Richland, SC, 2010.

[8] D. Floreano, P. Dürr, and C. Mattiussi. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62, 2008.

[9] J. Gauci and K. O. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, Menlo Park, CA, 2008. AAAI Press.

[10] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, 22: 1860–1898, 2010.

[11] S. Grillner and P. Wallen. Central pattern generators for locomotion, with special reference to vertebrates. *Annual review of neuroscience*, 8(1):233–261, 1985.

[12] G. Hornby, S. Takamura, J. Yokono, O. Hanagata, T. Yamamoto, and M. Fujita. Evolving robust gaits with AIBO. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 3040–3045, 2000.

[13] H. Kimura, T. Yamashita, and S. Kobayashi. Reinforcement learning of walking behavior for a four-legged robot. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, pages 411–416, 2001.

[14] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19 (2):189–223, 2011.

[15] M. A. Lewis and G. A. Bekey. Gait adaptation in a quadruped robot. *Autonomous Robots*, 12:301–312, 2002.

[16] G. McHale and P. Husbands. Gasnets and other evolvable neural networks applied to bipedal locomotion. *From Animals to Animats 8*, 2004.

[17] K. Pearson et al. The control of walking. *Scientific American*, 235(6): 72, 1976.

[18] T. Reil and P. Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168, April 2002.

[19] J. Secretan, N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: Evolving pictures collaboratively online. In *CHI '08: Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1759–1768, New York, NY, USA, 2008. ACM.

[20] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[21] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.

[22] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[23] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[24] R. Téllez, C. Angulo, and D. Pardo. Evolving the walking behaviour of a 12 dof quadruped using a distributed neural architecture. *Biologically Inspired Approaches to Advanced Information Technology*, pages 5–19, 2006.

[25] M. Van De Panne and A. Lamouret. Guided Optimization for Balanced Locomotion. In *6th Eurographics Workshop on Animation and Simulation*, Maastricht, Pays-Bas, 1995.

[26] P. Verbancsics and K. O. Stanley. Evolving static representations for task transfer. *Journal of Machine Learning Research*, 99:1737–1769, August 2010.

[27] D. Wettergreen and C. Thorpe. Gait generation for legged robots. In *IEEE International Conference on Intelligent Robots and Systems*, 1992.