Single-Unit Pattern Generators for Quadruped Locomotion

Gregory Morse
Department of EECS
University of Central Florida
Orlando, FL 32816
gregmorse12@gmail.com

Sebastian Risi
Sibley School of Mechanical
and Aerospace Engineering
Cornell University
Ithaca, USA 14853
sebastian.risi@cornell.edu

Charles R. Snyder
Department of EECS
University of Central Florida
Orlando, FL 32816
charles@knights.ucf.edu

Kenneth O. Stanley
Department of EECS
University of Central Florida
Orlando, FL 32816
kstanley@eecs.ucf.edu

ABSTRACT

Legged robots can potentially venture beyond the limits of wheeled vehicles. While creating controllers for such robots by hand is possible, evolutionary algorithms are an alternative that can reduce the burden of hand-crafting robotic controllers. Although major evolutionary approaches to legged locomotion can generate oscillations through popular techniques such as continuous time recurrent neural networks (CTRNNs) or sinusoidal input, they typically face a challenge in maintaining long-term stability. The aim of this paper is to address this challenge by introducing an effective alternative based on a new type of neuron called a *single-unit pattern generator* (SUPG). The SUPG, which is indirectly encoded by a compositional pattern producing network (CPPN) evolved by HyperNEAT, produces a flexible temporal activation pattern that can be reset and repeated at any time through an explicit trigger input, thereby allowing it to dynamically recalibrate over time to maintain stability. The SUPG approach, which is compared to CTRNNs and sinusoidal input, is shown to produce natural-looking gaits that exhibit superior stability over time, thereby providing a new alternative for evolving oscillatory locomotion.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Performance

Keywords

HyperNEAT, Generative and Developmental Systems, Neuroevolution, Gait Generation, Oscillation

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

1. INTRODUCTION

Legged robots have attracted significant interest in a broad range of applications ranging from consumer toys to military transporters, and even planetary exploration [10, 17, 23]. Legs offer several benefits over wheels, such as increased maneuverability in tight quarters and over rugged terrain. However, the potential advantages of legged robots are tempered by the difficulty of formulating effective methods to control them. For example, they have many degrees of freedom, which adds to the complexity of the task. While controllers for legged robots are often hand crafted [1], the design process is time consuming and would need to be repeated for a different body morphology. Thus a promising alternative to hand-crafting controllers is to train them through machine learning. Popular approaches include neuroevolution (i.e. evolving neural networks) and other evolutionary algorithms, which have achieved some success in approaching the challenge of creating controllers for legged robots [4, 8, 17, 19, 20, 35, 36, 38]. Evolutionary algorithms offer the increased flexibility of being possible to re-run for different body morphologies, alleviating the burden of creating a new controller by hand for each new body morphology.

Among the various neuroevolution-based approaches to solving the problem of legged locomotion, a common theme is to facilitate oscillation within the neural architecture. One popular approach is to enable oscillation by evolving central pattern generators (CPGs) based on continuous time recurrent neural networks (CTRNNs) [3, 4, 35], which are inspired by mechanisms in the nervous systems of many animals [9, 28]. Another approach is to integrate oscillation more directly by inputting a sine wave (or other oscillatory pattern) directly into the artificial neural network (ANN) [8, 20, 38]. Still another is to compute leg motion as a function of the positions of other legs [36]. While these approaches each offer a useful contribution, a general problem for the field is that all approaches tend to trade off between flexibility and stability. For example, oscillators based on recurrent architectures can produce almost unlimited dynamics, but require significant design effort to achieve stable oscillation [6, 25]. Non-recurrent architectures on the other hand may be stable, but often restrict potential dynamics. For example, inputting a sine wave constrains the period from the start, and linking leg movement to other legs' po-

sitions determines a priori the kind of gait that can emerge depending on which legs are functions of which.

The aim in this paper is to provide a novel alternative that offers both flexibility and stability, while also being easy to set up. Rather than creating oscillation indirectly through the complex interactions of CPGs or directly through a sine wave, the proposed strategy is to produce oscillation by introducing a new type of neuron, called a *single-unit pattern generator* (SUPG) that once triggered generates a single cycle of a genetically-encoded temporal activation pattern. Through the repeated triggering of the SUPG, the activation pattern is generated repetitively, producing oscillations. Because the trigger can reset the oscillation at any point in the cycle, the SUPG can continually recalibrate to maintain stability. The SUPG approach takes advantage of the HyperNEAT method [11, 34] for indirectly encoding ANNs by encoding the temporal patterns of SUPGs through a compositional pattern producing network (CPPN) that encapsulates SUPG activation patterns as a function of their location within the network. That way, a single CPPN can indirectly encode the temporal dynamics of every joint in the body without the need for recurrence or for pre-arranged dynamics. Furthermore, the well-established pattern-generating capabilities of CPPNS [7, 15, 29, 30] now can benefit the discovery of *temporal patterns* underlying gaits.

The SUPG approach is compared in this paper to CTRNN and sine-based approaches. Not only do SUPG-based quadrupeds evolve faster and walk farther, but they exhibit significantly greater temporal stability (far beyond the training window), an essential property for any industrial application of evolved controllers that until now had not yet been achieved in bodies that require balance. The hope is that the SUPG approach can thereby open a new research direction in evolving robotic gaits that can provide a viable alternative to today's dominant approaches.

2. BACKGROUND

This section first discusses prior work in neuro-evolved gait generation and then reviews the NEAT and HyperNEAT approaches that enable SUPGs.

2.1 Neuro-evolved Gaits

The main challenge in applying neuroevolution to gait generation is to encourage some form of oscillation to produce regular gaits. A popular approach is to evolve CPGs by evolving a special kind of neural network called a continuous time recurrent neural network (CTRNN) [3, 4, 35]. It is important also to note that while non-neural CPGs [18, 25] are investigated outside evolutionary computation, the focus of this paper is on evolving neural network-based CPGs. In such neural CPGs, CTRNNs often produce regular oscillatory patterns that are suited to gaits in legged robots. However, while CPGs may be the most biologically authentic approach, they can be difficult for an evolutionary algorithm to tune. The oscillatory pattern produced by a CPG is the product of multiple neurons interacting in a complex network, which requires significant care in design to maintain stability [6].

Other approaches avoid this problem by inserting an oscillatory pattern directly into the network [8, 38]. These oscillations can be more easily tuned, alleviating some of the burden faced by the evolutionary algorithm. However, imposing a fixed oscillation period on the network constrains evolution's ability to find viable gaits that may exist outside the chosen period. Furthermore, fixing the period does not guarantee stability because the physical environment itself might gradually degrade a behavior cycle intended to repeat perfectly every time. One interesting alternative approach by Valsalam and Miikkulainen [36] is to compute leg movements as a function of the positions of specific other legs. This idea yielded robust gaits of specific types that correspond to the leg-leg dependencies defined by the experimenter. In contrast to these approaches, the new approach in this paper, based on HyperNEAT (described next), aims for the flexibility of a CTRNN (e.g. to determine period and dynamics) but with stability more like a fixed-period controller. Furthermore, the hope is to let evolution decide the best gait without any explicit enumeration of possible leg-leg dependencies.

2.2 NEAT and HyperNEAT

The new approach in this paper is based on HyperNEAT, which is an extension of the original NEAT algorithm that evolves ANNs with a direct encoding. The NEAT method has proven effective in a variety of control and decision-making domains [21, 31, 33]. NEAT draws much of its strength from the idea of increasing complexity. In particular, it begins with simple ANNs and gradually increases the complexity of their genotypes to produce more sophisticated behavior in their phenotypes. A full description of NEAT is available in Stanley and Miikkulainen [31] or Stanley and Miikkulainen [33].

With direct encodings such as in NEAT, each weight in the network is represented by a single parameter in the genotype. This one-to-one mapping works in simple problems but cannot exploit domains with very large phenotypic solutions that contain self-similar parts. For example, a structure that effectively controls one leg of a quadruped is likely similar to the structure that effectively controls one of the other legs. With a direct encoding, any such regularities in the solution must be discovered separately.

In contrast, HyperNEAT [11, 34] is an extension of NEAT that evolves an indirect encoding called a compositional pattern producing network (CPPN). In effect, the CPPN in HyperNEAT is evolved by NEAT. One common advantage of indirect encodings such as CPPNs is that they can learn regularities all at once by reusing genetic information. The simultaneous discovery of regularities is one reason indirect encodings [2, 5, 13, 16, 22, 32] have become an important area of research. CPPNs are a type of network that serve as an indirect encoding in which each node can contain one of several functions such as sine and Gaussian. The use of such functions allows the CPPN to easily discover natural patterns that contain symmetry, repetition, and repetition with variation [30]. These patterns can then map to a *connectivity pattern* that also contains such regularities. CPPNs thus allow the simultaneous discovery of regularities across the connectivity of an ANN.

By convention, the nodes connected by the CPPN are placed in a geometric space called the *substrate*, as shown in figure 1. Notice that each node in the substrate in figure 1 is placed at an explicit (x, y) location. That way, it is possible to *query* the CPPN for the weight of any connection between two points (x_1, y_1) and (x_2, y_2) by inputting (x_1, y_1, x_2, y_2) into the CPPN, which then outputs the associated weight. The key idea is that the CPPN in effect paints a pattern

across the geometry of the network by querying all the potential connections for their weights. This pattern thereby yields a function of the geometry of the underlying problem domain, allowing HyperNEAT to exploit the structure of the problem. Kodjabachian and Meyer [19] have noted that geometry is important in particular to evolving effective gait controllers. Notice that querying the weight of a connection requires inputting the four dimensions x_1, y_1, x_2, and y_2. In effect, as HyperNEAT queries all such connection weights, it is painting its pattern within a four-dimensional hypercube, which is why it is called HyperNEAT.

Because in this paper HyperNEAT also evolves properties of *nodes* (such as the SUPG), it is also important to note that the CPPN can also be queried to output parameters of individual nodes in the substrate instead of just parameters of connections. The convention for this purpose is to query a node at position (x, y) as $(x, y, 0, 0)$ and to retrieve the corresponding node-specific parameter (such as a bias) from the a special node-specific output on the CPPN (figure 1). In this way, HyperNEAT can also evolve properties of nodes as a function of where they appear in the network.

HyperNEAT has been successful in a variety of challenging domains that require the discovery of regularities [11, 12, 34, 37]. Because effective legged locomotion exhibits regularities, HyperNEAT is also well-suited to the task of discovering controllers that produce regular gaits [8]. The next section describes the new HyperNEAT-based approach proposed for this problem.

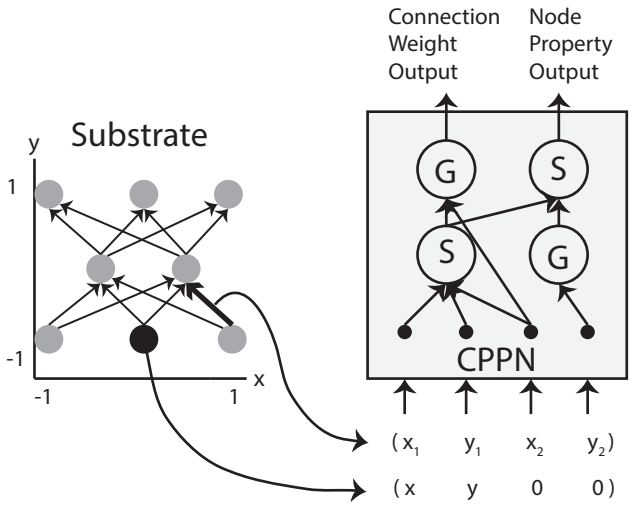

Figure 1: How CPPNs encode connection weights and node properties in HyperNEAT. The *substrate* (left) consists of a collection of nodes and connections, where each node is placed in a geometric location ranging from -1 to 1 in each dimension. Connection weights and node properties are generated by feeding coordinates into the CPPN (right), which is evolved with the NEAT algorithm. *Connection weights* are produced by inputting the coordinates of the connected nodes, such as (x_1, y_1, x_2, y_2). *Node properties*, such as bias and time constant, are produced by inputting the coordinates of the node into one coordinate pair and setting the other coordinate pair to 0, such as $(x, y, 0, 0)$. Although only one node property output is shown, any number of node property outputs can be included depending on the number of node properties that need to be determined. In the depicted CPPN, which is evolved by NEAT, S stands for sigmoid and G stands for Gaussian.

3. SUPG APPROACH

Producing oscillations can be difficult and deliberately tuning such oscillations poses a serious challenge to any neural architecture. The pattern of oscillations may require significant complexity. Because CPPNs have exhibited the ability to encode spatial patterns with a natural appearance [7, 29, 30], an interesting possibility is that they could be successful at encoding patterns across *time* as well. In fact, their use in generating temporal music sequences [14, 15] further supports this idea. After all, a pattern across space or a pattern across time is still just a pattern. Thus CPPNs could serve as the foundation for an oscillatory system. This observation is the motivation behind the introduction of the single-unit pattern generator (SUPG). The SUPG, shown in figure 2, is a type of neuron whose activation pattern over *time* is actually encoded by a CPPN. As is conventional for CPPNs, the CPPN takes as inputs the (x, y) coordinates of the SUPG, but it also takes a novel input: the time since the SUPG began the current cycle. In other words, the CPPN computes the activation level of the SUPG as a function of the time since its last cycle began (in addition to its position in the substrate). That way, all the pattern-generating capabilities of CPPNs are now brought to bear on generating *temporal* patterns that can be activated in cycles, which is a novel approach to generating gaits.

The temporal duration of an SUPG cycle is called the period of the SUPG. During the period of the SUPG, its internal timer climbs linearly at each tick of the clock in simulation from a starting value of zero at the start of the period to an ending value of 1.0 at the end of the period. In effect, the CPPN that encodes the resultant temporal pattern of the SUPG is computing this pattern as a function of the current state of the timer, as shown in figure 3. In-

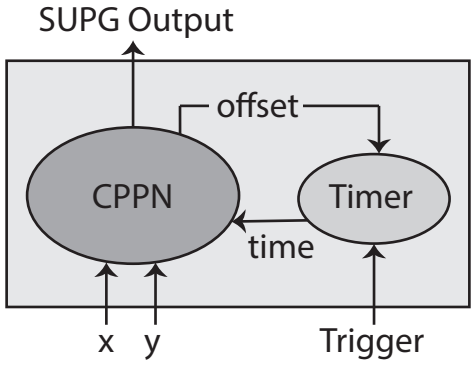

Figure 2: Visualization of a single SUPG. Once triggered, the activation produced is based upon the neuron's coordinates (x, y) in the substrate and the time since it was last triggered. The offset determines the time of the initial trigger. For simplicity, potential x_2 and y_2 inputs to the CPPN, which could allow it also to generate connection weights (in addition to SUPG properties) are not shown.

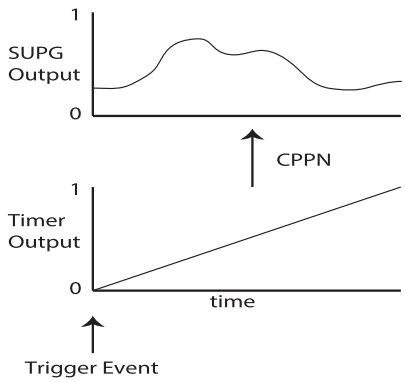

Figure 3: One period of an SUPG. Once triggered, the timer within an SUPG starts at zero and increases linearly until it reaches a maximum of one at the end of the period. This timer output is input into the CPPN that encodes the SUPG (along with the SUPG's x and y coordinates) to determine the output of the SUPG.

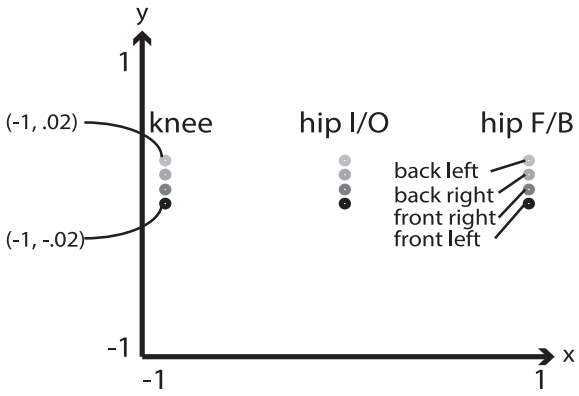

Figure 4: SUPG-based substrate for quadruped locomotion (not drawn to scale). Each depicted node is an SUPG at its designated position in the substrate. The SUPGs for each motor type are grouped tightly together, resulting in each leg producing similar motion trajectories (though with different offsets). At the same time, different types of joints are placed farther apart, reflecting that the trajectories of different types of joints likely differ from each other. The three motors are *knee*, *hip in/out*, and *hip forward/back*.

terestingly, unlike with a sine-based controller, this SUPG period does not need to match the length of the optimal or desired gait. The reason is that the period can be *restarted* at any time by the occurrence of an external event called the *trigger*. That way, instead of simply repeating its pattern at a set interval, the SUPG contains a trigger that when fired resets its activation pattern to the start (i.e. time zero) of a new period. If the trigger is fired again before the period is complete, the period is interrupted and restarts. One logical choice for a trigger in locomotion is a foot touching the ground. That way, when a foot touches the ground, it thereby restarts all SUPGs that are connected to that foot. This triggering mechanism means that the ultimate cycle depends on *events in the world* rather than on a fixed (and thereby brittle) mechanism. It also potentially permits flexible adjustment to variation in terrain, in addition to reducing the accumulation of errors that can occur with a fixed repeated period. Thus the combination of the pattern generating capability of the CPPN with the new trigger introduces a new oscillatory mechanism to ANNs that is naturally suited to evolution by HyperNEAT.

SUPGs must be placed in the HyperNEAT substrate in a principled manner. Because HyperNEAT can take advantage of the geometry of the problem, it is desirable to select a configuration that exploits such an advantage. For example, consider a quadruped with three motors per leg: one knee joint and two hip joints. Because legs in most quadrupeds perform similar motions to each other, the SUPGs can be grouped closely on the substrate to SUPGs for other legs that control motors of the same type (which conveys to HyperNEAT that their patterns will be similar). In this configuration, the different motor types are spread along the x axis, as shown in figure 4. By grouping SUPGs for related joints together, but spreading them far from the joints of other types, HyperNEAT can exploit such geometry to produce gaits in which each of the legs behave in a similar manner (though of course potentially not all in phase), but each joint of a particular leg can yield a different motion trajectory from the other joints on that leg.

Note that if all four legs start touching the ground, then all four would be triggered simultaneously, which would make it difficult to produce any gait except a pronk, a hopping gait common in some quadrupedal animals in which all four legs move in unison. To allow a wider range of gaits, an *offset* can be evolved for the first step. Because SUPGs are temporal, adding this offset is simple. For this purpose, a second output is added to the CPPN, as shown in figure 2, that defines the offset for a given SUPG. When querying the CPPN to obtain this offset, the x value and time value are set to 0, and the y value is set to the y position of the SUPG. This configuration ensures that all SUPGs for the same leg are started at the same time. Note that once the initial offset time elapses, offsets are discarded and leg cycles thereafter depend entirely on triggers.

Taken together, the SUPG approach encompasses three key concepts: (1) the CPPN encodes the *shape* of the cycle, (2) the trigger determines its *timing*, and (3) the offset controls its *initiation*. By encapsulating this system within the geometry of the HyperNEAT substrate, a simple and clean approach (which does not depend on recurrence) is introduced to describe and evolve complex oscillatory gait patterns.

4. EXPERIMENT

The aim of this experiment is to establish the viability of SUPGs as an alternative research direction in legged locomotion. Thus it is compared to two more traditional architectures in the quadruped domain described in the previous section. The hope is that this comparison can help to show that SUPGs are viable without unnecessarily implying that they are the "best" among all possible alternative methods. That is, the aim is to establish SUPGs as a viable alternative.

The quadruped domain is implemented in the Open Dynamics Engine (ODE) physics simulator (`http://www.ode.`

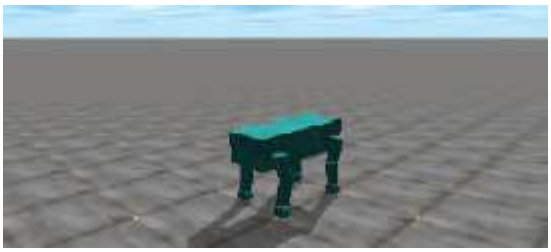

Figure 5: The starting position of the quadruped in the training environment. The environment is an unlimited flat plane with no obstacles. The quadruped starts in a standing position with all feet touching the ground. It is controlled with 12 motors: one for each knee, one for each hip forward and back, and one for each hip in and out.

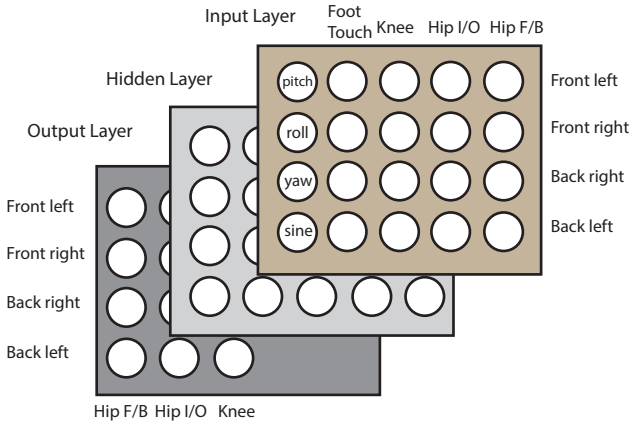

Figure 6: Substrate for the sine wave architecture (inspired by Clune et al. [8]). The substrate consists of three planes: input, hidden, and output. The input layer includes rows of sensors for each of the twelve joints, along with touch sensors for each of the feet. It also inputs the pitch, roll, and yaw of the body, and the sine wave that provides the main fixed period of the gait. The hidden layer contains twenty hidden nodes and the output layer produces twelve outputs for the twelve joint motors. The CPPN takes six (two sets of three) inputs for the x and y position and layer, and has a single output for connection weight.

`org`). The quadruped's body is 1.17m long, 0.43m wide, and 0.40m high. Each leg is 0.7m long. The quadruped starts in a standing position with all legs precisely vertical and knees locked, as shown in figure 5, and attempts to ambulate across an infinite flat plane. It is important to note that unlike some quadruped architectures [8, 38], this quadruped's legs are vertical and relatively long, which necessitates consistent stability to avoid falling. This quadruped physics model is also used in Risi and Stanley [26], which focuses on the problem of training for multiple body morphologies. Each leg has three degrees of freedom: knee, HipFB (foward and backward) and HipIO (inward and outward). Each substrate output is scaled to match the angular range of the corresponding joint and is interpreted as the desired angle. The difference between the desired angle and current angle guides a proportional controller that applies torque to reduce this difference. This method of control is similar to those in Reil and Husbands [24] and Lehman and Stanley [21].

The **SUPG architecture** described in the previous section (shown in figure 4) is compared to architectures that exploit CTRNNs and sine waves to create oscillation. All three architectures are evolved by HyperNEAT with the same parameter settings. The **sine wave architecture**, detailed in figure 6, is inspired by Clune et al. [8], which first demonstrated HyperNEAT effectively applied to quadruped locomotion. In fact, a later test of this approach on a real quadruped robot yielded the fastest gait yet demonstrated by any optimization method for that model [20] (though of course the robot in this paper has a different morphology). The period of the sine wave is set to one second, which matches the period of the SUPGs (though recall that the actual period realized by the SUPG can vary because of the trigger). The **CTRNN architecture**, detailed in figure 7, is inspired by Téllez et al. [35] and Risi and Stanley [26]. In those works, each leg contains an identical recurrent neural structure while connections between the modules provide information needed to coordinate the different legs. The HyperNEAT substrate in the experiment augments the original architecture by incorporating geometric information. It is important to note that the sine wave and CTRNN results are not directly comparable to their inspirations in Clune et al. [8] and Téllez et al. [35] because the physical simulation and quadruped architectures are *not* identical to those works in this experiment. For example, the legs in the model in this paper are relatively long, increasing the challenge of maintaining stability, which is a focus of this study. Also, the CTRNN cannot be directly compared to Risi and Stanley [26] because their architecture receives joint angles as input while the one in this paper receives only foot touch (just like the SUPG), which is less informative. However, these re-implemented approaches do respect the overall designs of the originals and furthermore were both validated to evolve effective walking gaits that can last the duration of the training trial.

Previous work has shown that novelty search [21], which alters the traditional fitness function to search for novelty rather than optimality, is effective for biped locomotion because it avoids deception [27]. Therefore, both novelty search and conventional fitness-driven search are attempted for each of the three variant approaches. In novelty search, individuals are rewarded not for how well they perform in an objective sense, but for how much their behavior is different from that of their peers. For this purpose, the behavior of an individual in the quadruped domain is defined as a vector of coordinates corresponding to the center of mass of the individual at each of the $1,500$ simulated time steps over each 15-second trial. The similarity of individuals for the purpose of measuring novelty is then the Cartesian distance between their behavior vectors. The fitness function for the more traditional fitness-driven runs is the Cartesian distance from the starting point to the endpoint at the end of the 15-second trial.

Each individual is evaluated during training for fifteen seconds of simulation time. The population size is 300 and each run is given 800 generations. The activation functions for the CPPN are sine, Gaussian, bipolar sigmoid, and linear, each with an equal probability. The probability of adding a connection and node to the CPPN is 0.06 and 0.01, respectively. The elitism proportion is set

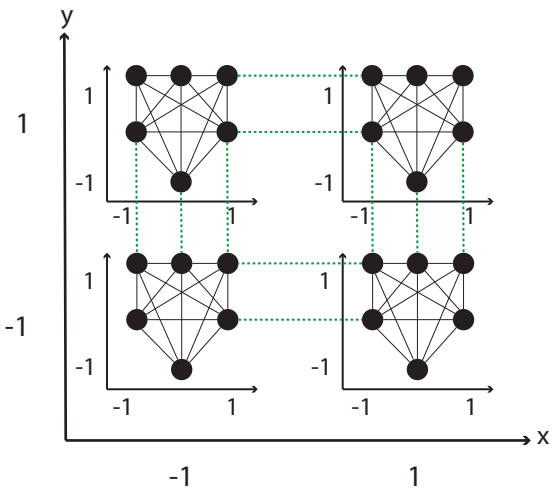

Figure 7: Substrate for the CTRNN architecture (inspired by Téllez et al. [35] and Risi and Stanley [26]). Each leg is controlled by its own recurrent module. The bottom node of each module is the touch sensor, while the top three nodes correspond to its three motor outputs. Each module is connected to neighboring modules with dashed lines. Each node has four points defining its location in space: an x and y value within its module and an x and y value for the location of its module within the overall substrate. The weight between the nodes is determined by sending in the eight (four for each node) coordinates of the connected nodes into the CPPN. There are then separate CPPN outputs for inter-module connections, intra-module connections, node bias, and node time constant. When querying the CPPN for bias and time constant, target node inputs are set to zero, as explained in Section 2.2.

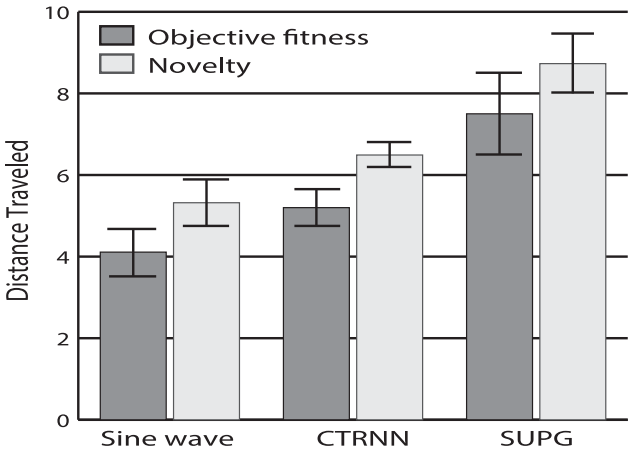

Figure 8: Training performance of the sine wave, CTRNN, and SUPG-based architectures with both objective fitness and novelty search. Each result represents the distance traveled in a 15 second period averaged over 20 runs. Bars indicate 95% confidence intervals. The main result is that the SUPG travels significantly farther with either method of evolution.

to 10%, and elitism is carried out using the objective fitness, to ensure the objective fitness never decreases. The novelty threshold is set to $1,000$ initially. Novelty is calculated as the sum of distances to the 15 nearest neighbors. To test stability, the best individuals from training are later tested in much longer five minute trials. Source code for the experiments in this paper can be found at `http://eplex.cs.ucf.edu/uncategorised/software`.

5. RESULTS

Figure 8 shows the final training performance for the three neural architectures (averaged over 20 runs for each), which is measured as their distance traveled within the 15-second training window. If success in producing legged locomotion is defined as four or more meters traveled (recall the body is just 1.17m long), then all architectures sometimes produce controllers that succeed in traversing the environment. The ability of all variants to succeed in this way validates the viability of all of them. However, the CTRNN architecture performs significantly better ($p < .01$; Student's t-test) than the sine wave architecture, while the SUPG architecture performs significantly better than either alternative ($p < .01$ in both cases). It is important to note that by 800 generations, all methods converge, that is, further evolution would not be expected to significantly alter the final results. Furthermore, the relative results are the same whether training with novelty or fitness, though novelty does tend to outperform fitness for any given approach. The average *CPPN* complexity for SUPG solutions from novelty search is 6.4 hidden nodes (sd = 2.8) and 39.7 connections (sd = 17.0); for the sine wave it is 10.5 hidden nodes (sd = 8.5) and 64.8 connections (sd = 47.4); for the CTRNN it is 2.0 hidden nodes (sd = 3.95) and 48.2 connections (sd = 19.7).

Qualitatively, the SUPG architecture produces steadier and more natural-looking gaits (see videos at `http://eplex.cs.ucf.edu/supg/supgql`). The only standard gait produced by the sine wave and CTRNN architectures is a pace, a gait in which the left legs move in unison and the right legs move in unison. In contrast, the SUPG architecture produces a pace, trot (in which diagonal legs move in unison) and walk (a four-beat gait that is between a pace and a trot). While the CTRNN and sine wave architectures may be capable of producing a wider variety of quadrupedal gaits in some domains, the challenging morphology of the body in this domain likely restricts the range of gaits produced.

To test for long-term stability, a major challenge for evolved walkers, figure 9 shows the average performance of the champions of each architecture on a five minute extended trial, which is many times longer than the 15-second training trials. Because novelty proves superior in training, only walkers evolved with novelty are tested in the extended trial. While the sine wave architecture travels on average 36% farther in the extended trial than in the 15-second training window and the CTRNN architecture travels more than twice as far, the SUPG architecture exhibits a highly significant difference by traveling on average over *eight times* as far in the extended trial than in training. Of the 20 runs performed on each of the other architectures, only a single run with the CTRNN architecture reached the average testing performance of the SUPG architecture. Also, while no individuals evolved from the CTRNN or sine wave architectures walk for more than three minutes, five of the individuals

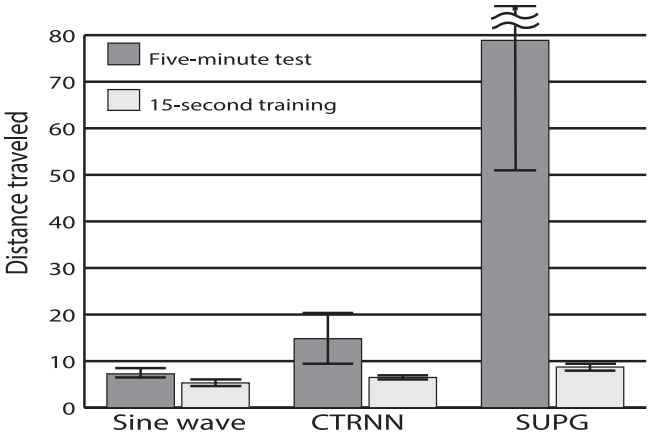

Figure 9: Testing performance of the sine wave, CTRNN, and SUPG-based architectures on the five minute trial. Each result represents the distance traveled in a five-minute period averaged over the 20 champions from each approach. Training performance is also shown for each method for comparison. Bars indicate 95% confidence intervals. The SUPG approach averages over five times farther than the CTRNN and ten times farther than the sine, demonstrating superior long-term stability.

evolved from the SUPG architecture walk for the duration of the five minute testing period. In fact, one individual walks for over *three hours* before falling.

6. DISCUSSION AND FUTURE WORK

The success of the SUPG establishes that CPPNs can produce temporal patterns suited to the task of controlling legged robots. Not only does the SUPG approach yield controllers that can produce locomotion, but the resultant gaits are both natural-looking and robust, that is, they are able to continue walking well beyond the training window. While all three architectures could produce walkers that remain stable during training, only the SUPG produces such performance well beyond the training window. The relative simplicity of the SUPG substrate, detailed in figure 4, is also notable because it is easier to set up. Its robustness is a product of both the ability of CPPNs to encode complex patterns and the novel triggering mechanism for resetting the cycle of each SUPG, which permits imperfections in the timing of the gait to be corrected while the robot is in motion.

The major contribution of this work is not simply a good result, but rather a new research direction for evolving effective locomotion through indirect encoding. The simple walkers evolved in this paper are only a first step. For example, the SUPG architecture could include the ability to *dynamically* modulate either the period of the SUPG cycle or the shape of the temporal pattern produced by the SUPG, based upon other inputs such as leg angles. In this way, the pattern produced by the SUPG could be tuned in response to changes in the environment such as obstacles or inclines. The substrate that incorporates the SUPGs could also be significantly more sophisticated. For example, it could include additional traditional neurons as well as connections among SUPGs and neurons, e.g. to control turning or gait modulation. In addition, the transition from one cycle to another in the current architecture may be abrupt because the SUPG output at the end of the cycle may differ from that at the start. Future research could identify steps to smooth this transition and thereby offer improved stability. Finally, other legged morphologies, such as bipeds or hexapods, may also benefit from SUPGs. In this way, SUPGs open a rich set of future possibilities.

7. CONCLUSION

This paper presented a novel approach called a single-unit pattern generator (SUPG) for harnessing the pattern producing capabilities of CPPNs to encode temporal patterns that can be applied to the domain of legged robot locomotion. Results in a quadruped domain comparing the SUPG to more traditional oscillatory gait generating techniques show that not only does this approach produce controllers that solve the locomotion task, but that it is also sufficiently robust to continue performing well beyond the short window of training. This novel approach provides a new avenue for confronting the enduring challenge of evolving fluid and effective walking robots.

References

[1] A. Azad, N. Cowan, M. Tokhi, G. Virk, and R. Eastman, editors. *Adaptive Mobile Robotics: Proceedings of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, 2012. World Scientific Publishing Company.

[2] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 35–43, 1999.

[3] A. Billard and A. Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, volume 6, pages 637–641. IEEE, 2000.

[4] J. Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108 (4):1234–1239, 2011.

[5] J. Bongard and R. Pfeifer. Evolving complete agents using artificial ontogeny. *Morpho-functional Machines: The New Species*, pages 237–258, 2003.

[6] H. Chiel, R. Beer, and J. Gallagher. Evolution and analysis of model CPGs for walking: I. dynamical modules. *Journal of Computational Neuroscience*, 7 (2):99–118, 1999.

[7] J. Clune and H. Lipson. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In *Proceedings of the 20th European Conference on Artificial Life*, pages 141–148, 2011.

[8] 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.

[9] F. Delcomyn. Neural basis of rhythmic behavior in animals. *Science*, 1980.

[10] P. Fiorini, S. Hayati, M. Heverly, and J. Gensler. A hopping robot for planetary exploration. In *IEEE Aerospace Conference*, volume 2, pages 153–158, 1999.

[11] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, 22(7):1860–1898, 2010.

[12] J. Gauci and K. O. Stanley. Indirect encoding of neural networks for scalable go. In R. Schaefer, C. Cotta, J. Kołodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature – PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 354–363. Springer, 2010. ISBN 978-3-642-15843-8.

[13] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 81–89. MIT Press, 1996.

[14] A. K. Hoover and K. O. Stanley. Exploiting functional relationships in musical composition. *Connection Science Special Issue on Music, Brain, and Cognition*, 21(2 and 3):227–251, 2009.

[15] A. K. Hoover, P. A. Szerlip, and K. O. Stanley. Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In M. L. Maher, K. Hammond, A. Pease, R. P. Y. Perez, D. Ventura, and G. Wiggins, editors, *Proceedings of the 3rd International Conference on Computational Creativity (ICCC-2012)*, 2012.

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

[17] G. S. Hornby, S. Takamura, J. Yokono, O. Hanagata, M. Fujita, and J. Pollack. Evolution of controllers from a high-level simulator to a high DOF robot. In *Evolvable Systems: From Biology to Hardware*, pages 80–89. Springer, Berlin, 2000.

[18] A. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.

[19] J. Kodjabachian and J.-A. Meyer. Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects. *IEEE Transactions on Neural Networks*, 9(5): 796–812, 1998.

[20] S. Lee, J. Yosinski, K. Glette, H. Lipson, and J. Clune. Evolving gaits for physical robots with the HyperNEAT generative encoding: the benefits of simulation. *Applications of Evolutionary Computing*, 2013.

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

[22] J. F. Miller. Evolving a self-repairing, self-regulating, French flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004)*, Berlin, 2004. Springer Verlag.

[23] M. Raibert, K. Blankespoor, G. Nelson, R. Playter, et al. Bigdog, the rough–terrain quadruped robot. In *Proceedings of the 17th International Federation of Automation Control*, pages 10823–10825, 2008.

[24] 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, 2002.

[25] L. Righetti and A. Ijspeert. Design methodologies for central pattern generators: an application to crawling humanoids. In *Proceedings of robotics: Science and systems*, pages 191–198. MIT Press Cambridge, MA, 2006.

[26] S. Risi and K. O. Stanley. 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)*. ACM Press, 2013.

[27] S. Risi, S. D. Vanderbleek, C. E. Hughes, and K. O. Stanley. How novelty search escapes the deceptive trap of learning to learn. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2009)*, New York, NY, USA, 2009. ACM Press.

[28] S. Rossignol. Neural control of stereotypic limb movements. *Comprehensive Physiology*, 1996.

[29] J. Secretan, N. Beato, D. B. D.Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 19(3):345–371, 2011.

[30] 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.

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

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

[33] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *JAIR*, 21:63–100, 2004.

[34] 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.

[35] 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.

[36] V. Valsalam and R. Miikkulainen. Modular neuroevolution for multilegged locomotion. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 265–272. ACM, 2008.

[37] B. G. Woolley and K. O. Stanley. Evolving a single scalable controller for an octopus arm with a variable number of segments. In *Parallel Problem Solving from Nature – PPSN XI*, volume 6239 of *Lecture Notes in Computer Science*, pages 270–279. Springer, 2010.

[38] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. Zagal, and H. Lipson. Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization. In *Proceedings of the 20th European Conference on Artificial Life*, pages 890–897, 2011.