Evolving CPPNs to Grow Three-Dimensional Physical Structures

Joshua E. Auerbach
Morphology, Evolution and Cognition Lab
Department of Computer Science
University of Vermont
Burlington, VT 05401
joshua.auerbach@uvm.edu

Josh C. Bongard
Morphology, Evolution and Cognition Lab
Department of Computer Science
University of Vermont
Burlington, VT 05401
jbongard@uvm.edu

ABSTRACT

The majority of work in the field of evolutionary robotics concerns itself with evolving control strategies for human designed or bio-mimicked robot morphologies. However, there are reasons why co-evolving morphology along with control may provide a better path towards realizing intelligent agents. Towards this goal, a novel method for evolving three-dimensional physical structures using CPPN-NEAT is introduced which is capable of producing artifacts that capture the non-obvious yet close relationship between function and physical structure. Moreover, it is shown how more fit solutions can be achieved with less computational effort by using growth and environmental CPPN input parameters as well as incremental changes in resolution.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Experimentation

Keywords

Evolutionary robotics, Generative and Developmental Systems

1. INTRODUCTION

Robots that operate in outdoor or other unstructured environments such as the home or office would be of great social utility. But, to date, the vast majority of robots currently in use operate only in structured environments such as factories. If robots are to make the migration from factories into our everyday lives they will need to be adaptive; that is, they must exhibit intelligent behavior.

According to proponents of embodied artificial intelligence such intelligent behavior arises out of the coupled dynamics between an agent's body, brain and environment [9, 2, 25, 4]. One extension of this concept is that the complexity of an agent's controller and morphology must match the complexity of the task or tasks that it is required to perform. However, when extending this idea to more complex agents in more complex environments it is not clear how to distribute responsibility for different behaviors across the agent's controller and morphology. For example, if all a robot needs to do is follow a light source over flat terrain wheels and a direct sensory motor mapping would be an appropriate solution [8], but if the robot must be able to navigate over a variety of terrains and perform more complicated tasks a more complex control strategy and/or morphology are required. This issue of scaling has been one of the major obstacles in developing robots capable of robust and adaptive behavior in unstructured environments.

1.1 Background

Evolutionary robotics [17, 24], in which evolutionary algorithms are employed to optimize the control policy of a robot, has provided one framework for overcoming the limitations of human intuition in designing robust, non-linear, control strategies. However, the majority of work in evolutionary robotics has done only that: optimize control strategy for a human designed or bio-mimicked robot morphology. This methodology has severe limitations: fixing a robot's morphology places limits and biases on the kinds of action that the robot can perform, and therefore also on the more complex behaviors that those actions may eventually support. For example, a robot with legs can only exhibit legged locomotion while a wheeled robot with a rigid gripper can only move over even terrain and grasp objects with a fixed radius.

However, there are ways to overcome these limitations. Evolutionary algorithms may be used to optimize robot morphology as well as the control policy. Sims [28] first introduced an evolutionary framework in which both the morphology and control of simulated machines were optimized in virtual environments to produce adaptive behavior. This work was followed by other studies [14, 21, 1, 22, 20, 18, 19, 30, 15, 5, 7, 6] in which aspects of the machine's morphology and control were evolved in virtual environments. This approach has the advantage of discovering body plans appropriate for the machine's task environment rather than being artifacts of human design biases or copies of animal body plans only appropriate for that animal's ecological niche.

In fact, it is sometimes possible to construct morphologies ideally suited to their task environment such that no active control is necessary to accomplish non-trivial behaviors. One class of such morphologies are passive dynamic walkers [13, 33]. These "robots" are able to stably locomote down an inclined plane without using any sensors or motors. Their stability and momentum conservation are inherent properties of their body plans. If the prevailing view held by robotocists that an appropriate morphology will always be intuitive to design, as summed up by Nelson *et al.*

> Humans are much better at designing physical systems than they are at designing intelligent control systems: complex powered machinery has been in existence for over 150 years, whereas it is safe to say that no truly intelligent autonomous machine has ever been built by a human (p. 22)[23].

were correct then it would be intuitive how to design such structures. However, such body plans are non-intuitive to design because they have subtle dependencies on mass distribution and structural curvatures, making automated methods including evolutionary algorithms good candidates for designing these artifacts.

In order to capture the essence of evolving non-trivial physical structures the current work tackles the problem of evolving solid objects with these important properties. While not incorporating any actuation and therefore not robots, strictly speaking, the structures evolved in this work provide a stepping stone for the eventual evolution of fully articulated robots controlled by closed loop, neural network control policies. This approach is not without precedent in the field of evolutionary robotics. Funes and Pollack [16] first demonstrated evolving solid structures such as bridges, scaffolds, and crane arms constructed of Lego bricks before moving on to the evolution of actuated robots [26].

Specifically, the work presented in this paper makes use of a recently introduced abstraction of development known as compositional pattern producing networks (CPPNs) [31] to grow three-dimensional physical structures. In this paper, these CPPNs are evolved using CPPN-NEAT [31] to produce physical structures capable of conserving momentum to achieve maximum displacement due to gravity. CPPNs are used here because they are a form of indirect encoding that have been shown able to capture geometric symmetries appropriate to the system being evolved, are capable of reproducing outputs at multiple resolutions [29], and have shown promise in producing neural network control policies for legged robots [11, 12]. The combination of these features makes it likely that evolving CPPNs will prove to be a more promising approach to realizing intelligent agents than other approaches.

This paper presents a method for using CPPN-NEAT along with a novel growth procedure to evolve three dimensional structures appropriate for a specific task, presents some advantages of CPPN-NEAT over other methods that may be used for evolving three-dimensional physical structures, and discusses how this method can be extended to co-evolve actuated body plans and control strategies.

2. METHODS

This section presents a brief description of CPPNs and the CPPN-NEAT evolutionary algorithm. This is followed by a description of the methods used for generating three-dimensional physical structures from evolved genotypes. Following this a description is presented of the fitness function used for evaluating these structures.

2.1 CPPNs

Compositional Pattern Producing Networks (CPPNs) are a form of artificial neural network (ANN) where each internal node can have an activation function drawn from a diverse set of functions instead of being limited to a sigmoid function as is the case with classical ANNs. This function set includes functions that are repetitive such as sine or cosine as well as symmetric functions such as gaussian, thus easily allowing for motifs seen in natural systems: symmetry, repetition, and repetition with variation.

2.2 CPPN-NEAT

CPPN-NEAT uses the NeuroEvolution of Augmenting Topologies (NEAT) [32] method of neuro-evolution to evolve increasingly complex CPPNs. An extension of CPPN-NEAT —HyperNEAT— has been used [29, 11, 12] to evolve traditional ANNs, where each node is embedded in a geometric space and whose coordinates are fed to an evolved CPPN. In effect the connections are "painted" on to the network from the output patterns produced by the CPPN. As shown by Stanley *et al.* [29] this has the crucial benefit that a CPPNs evolved to produce the connectivity patterns of small ANNs can be re-queried at a higher resolution to produce the connectivity patterns of larger ANNs without needing to re-evolve these large ANNs. Analogously CPPNs evolved to produce structures at one resolution should be able to also produce structures at a higher resolution.

CPPNs have several properties desirable for generating robot morphologies. It is directly evident that geometry is a key aspect of any artifact existing in a physical or simulated physical environment. Providing the evolutionary process with information about this geometry should be useful in evolving functional structures. Additionally the ability to operate at multiple resolutions should allow for the rapid evolution of coarse grained structures composed of a small number of large components followed by re-querying the generating CPPN to produce qualitatively similar structures composed of a greater number of smaller components without needing to evolve CPPNs for these higher resolution morphologies from scratch.

Importantly, if the evolved structures are to be physically fabricated using a technique like that outlined in [20] the structures' dynamical properties must be retained across the simulation-reality gap [3, 34]. The ability to query the same evolved encoding at different resolutions should be useful in this endeavor since the dimensions and densities with which the evolved structures will be fabricated most likely will not be able to precisely match those of the simulation.

2.3 Growing Three-Dimensional Physical Structures from CPPNs

In this work three-dimensional physical structures are grown from evolved CPPNs. Each structure is composed of many spherical cells which fuse together to make rigid bodies. For an example of a structure produced in this way see Fig. 1.

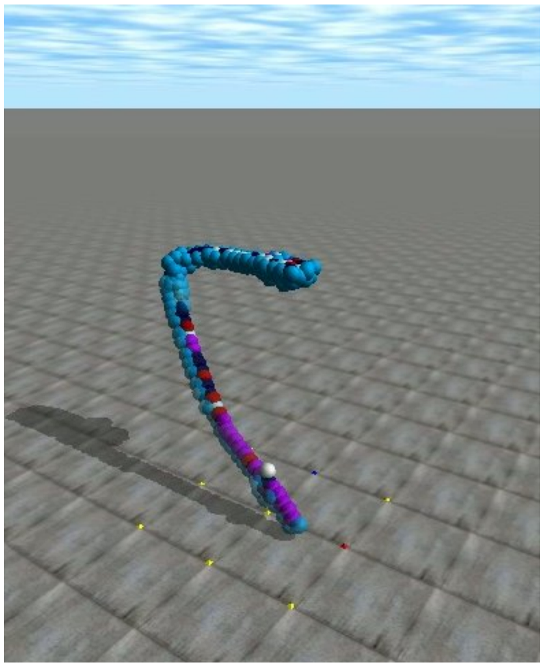


Figure 1: A sample structure evolved for maximum displacement due to gravity.

The growth procedure begins with a single cell, henceforth referred to as the root, located at a designated origin. A cloud composed of n points is cast around this cell with the n points being evenly distributed on the surface of the root sphere (all n points are at distance r from the center of the root). This point cloud is cast using a spiral method that is a variant of the algorithm presented by Saff and Kuijlaars [27]. Specifically the method is the "golden section" modification to the Saff and Kuijlaars algorithm described in [10]. Once this cloud is cast, every point in the cloud is used to query a CPPN to retrieve a single output value. This output value can be thought of as a concentration of matter at that point, such that when over a certain matter threshold, T_{matter}, a cell will be placed at that point. The more the output value exceeds the matter threshold the denser the cell placed at that point will be. This creates a continuum from no cell to very light cells to heavier cells (since the cells are all the same size, density and weight are completely correlated).

The CPPN takes as input the Cartesian coordinates (x, y, z) of the point in question as well as a constant bias input. Additionally information is provided as input about the growth trajectory itself: two angles ϕ_1 and θ_1 describe the direction from the parent cell that this point is located and an additional two angles (ϕ_2 and θ_2) provide directional information about the parent cell relative to its own parent cell. Further knowledge of the growth trajectory is provided via an input that receives the number of cells in the growth tree separating this point from the root (the cell's depth, d). Additional input is provided by a value representing the radius of the cells being considered for addition (r) thus informing the CPPN about the resolution at which the structure is being grown. Resolution will be dependent on the environment in which the physical structure exists, therefore this value may be considered an environmental input.

Once the output values for all n points in the cloud have been computed the points are sorted in order of descending output values. The sorted points are then looped through and the algorithm considers adding a cell centered at each point in turn. Specifically a cell, centered at point p is added to the structure if (a) the output value of point p is above the threshold T_{matter} and (b) no other cell, besides the one to which this new cell will be attached (its parent) has previously been added to the structure with center located at distance $< r$ away from p.

When a cell is added to the structure it gets placed into a priority queue whose priority is based on the output value of the CPPN at that point. When all points from the current cloud have been considered the algorithm takes the cell at the top of the priority queue and casts a point cloud around it, and this process continues either until there are no possi-

```
1.  GrowStructure(CPPN)
2.    Initialize priority queue $q$, with priority based on
          cell density
3.    Create cell $c$ at origin with full density, add to
          structure $S$ and flag its coordinates 'discovered'
4.    Enqueue $c$ in $q$
5.    WHILE $\sim q$.isEmpty
6.      $c \leftarrow q$.front
7.      Cast point cloud $C$ centered at $c$
            using the golden section spiral method
8.      Initialize vector $V$ of neighboring cells
9.      FOR EACH point $p$ in $C$
10.       Query CPPN at $p$ to get output value $v$
11.       Add $p$ with value $v$ to vector $V$
12.     Sort $V$ by descending value
13.     FOR EACH point $p$ with value $v$ in sorted vector $V$
14.       IF coordinates of $p$ not yet 'discovered'
15.         Flag $p$ 'discovered'
16.         IF CanAdd($p$,$v$,$c$)
17.           Add cell centered at $p$ with density
                  proportional to $v$ to structure $S$
18.           Enqueue $(p, v)$ in $q$
-----------------------------------------------------------
19. CanAdd($p$,$v$,$c$)
20.   IF $v > T_{\text{matter}}$ AND
          $\forall$ cells $d \in S, d \neq c$ dist($p$,$d$) $\geq r$ AND
          $p$ is within bounding cube AND
          $S$ has less than $M$ cells
21.     Return true
22.   ELSE
23.     Return false
```

Figure 2: Grow Structure pseudo code. The growth procedure starts with a root node at the origin (line 3). Then, as long as there are nodes in the queue to consider it takes the node at the front of the queue, casts a point cloud around it and considers adding a node at each point in turn (lines 5-18). A node is added at a given point if all of the following hold: it does not conflict with a previously added node, the CPPN outputs a value above the threshold T_{matter} when queried at that point, the point is within the bounding cube, and the maximum number of nodes M has not been reached (lines 19-23).

ble points at which to place cells or a maximum number of cells (M) have been created. One further constraint is that the structure is not allowed to grow outside of a bounding cube with side lengths l. This constraint was imposed so that in the future it will be possible to physically fabricate the entire evolved physical structures within the confines of a 3D-printer. Fig. 2 gives pseudo code for this growth procedure.

There are several reasons why it is desirable to have a growth procedure such as this. Merely querying CPPNs over a sampling of three-dimensional space may lead to disconnected objects. Even if all but one of these objects are thrown out much computational resources will have been wasted querying these regions of space. Additionally imposing a grid over space to determine which points to query restricts the sort of structures that may be produced when compared to the "point cloud" method discussed above. For example curved structures can be constructed using the point cloud method, but only coarsely approximated on a grid. Additionally, having a growth procedure allows for providing the CPPN with knowledge about the structure's growth trajectory and environment which prove to be beneficial (see below) and which may more easily allow for environmental influences to act on this growth trajectory in more complex ways in the future.

2.4 Selecting for dynamical properties of evolved structures

One major purpose of this paper is to demonstrate that CPPN-NEAT coupled with the growth procedure just presented is capable of evolving three-dimensional physical structures with desirable dynamic properties. This is a necessary capability of any evolutionary algorithm to be used for co-evolving robot morphology and control. In particular the property selected for in this work is the maximum displacement of an object due to gravity from a starting position where part of the object begins in contact with the ground.

To select for this property, an evolved virtual object is placed in a physical simulator[1] for a set amount of time. The fitness of this object is then calculated by finding the point of the object nearest to the origin of the space (where the object was touching the ground) in terms of the planar Euclidean distance. This distance becomes the fitness of the object and therefore of the CPPN that produced the object, which CPPN-NEAT attempts to maximize.

3. RESULTS AND DISCUSSION

This section presents the results of several experiments designed to illustrate the capabilities of this methodology as well as study the effects of including the growth and environmental inputs introduced above and the effects of varying the resolution at which structures are grown within a single evolutionary trial. All experiments are conducted with the fitness function described above and each experiment involves running 30 independent evolutionary trials. In all cases CPPN-NEAT is configured to use a population size of 150, and run for 200 generations. Additionally in all experiments the value T_{matter} is fixed at 0.7, and each cell of the structure is restricted to having its center within the bounding cube $([-2,2],[-2,2],[0,4])$ (coordinates all in meters). The CPPN internal nodes are allowed to use the signed cosine, gaussian, and sigmoid activation functions. All other parameters of the evolutionary algorithm are kept at the default values provided with the C++ implementation of HyperNEAT[2].

In the first experiment (**experiment 1**) the radius r of each cell is fixed at 0.1 meters and the growth procedure is limited to $M = 200$ cells. **Experiment 2** is identical to experiment 1 except the growth and environmental CPPN inputs ϕ_1, θ_1, ϕ_2, θ_2, r and d are omitted so that just the basic Cartesian coordinates (x, y, z) along with the constant bias are inputted to each CPPN. Henceforth the inputs used in experiment 1 will be referred to as the full set, while those used in experiment 2 will be referred to as the restricted set.

Experiment 3 uses the full set of CPPN inputs and begins with the cell radius r fixed at 0.15 meters and with $M = 60$ cells maximum per structure. These parameter values remain fixed for the first 100 generations of each evolutionary trial. After the 100th generation the resolution of the structures is increased to that of the first two experiments: $r = 0.1$ meters, $M = 200$. These values, as well as those used in all subsequent changes of resolution are chosen to preserve the volume of the structures. As the radius is changed by a factor x the maximum number of cells is changed by a factor x^3. The next experiment, **Experiment 4**, is identical to experiment 3, except the CPPNs are limited to the restricted set of inputs.

Experiment 5, like experiment 1, uses the full set of CPPN inputs and keeps the resolution fixed for the duration of each evolutionary trial. However, in experiment 5 this resolution is higher. The cell radius r is set to 0.08 meters with a maximum, $M = 391$ cells per structure. **Experiment 6** is identical to experiment 5, but uses the restricted set of CPPN inputs.

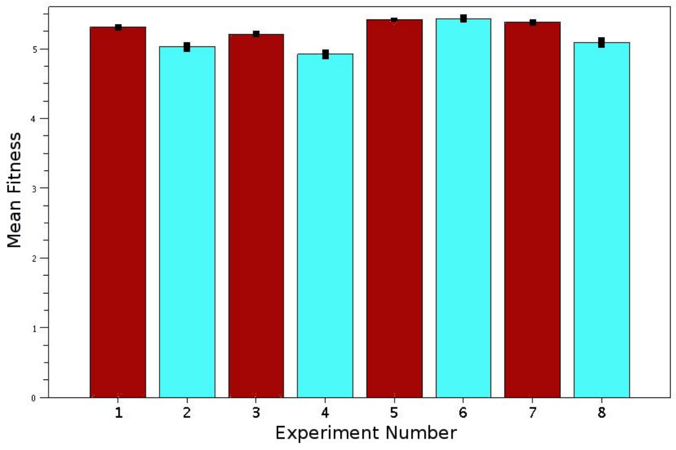


Figure 3: Mean best fitnesses (displacement in meters) in final generation across the 30 independent evolutionary trials with standard error bars for each of the eight experiments. The red bars represent experiments using the full set of CPPN inputs, while the blue bars represent experiments using the restricted set.

[1]Simulations are conducted in the Open Dynamics Engine (http://www.ode.org), a widely used open source, physically realistic, simulation environment

[2]Available at http://eplex.cs.ucf.edu/hyperNEATpage/HyperNEAT.html

	CPPN Input Set	**Resolution: first 100 generations**	**Resolution: second 100 generations**
Experiment 1	Full	$r = 0.1$m, $M = 200$	$r = 0.1$m, $M = 200$
Experiment 2	Restricted	$r = 0.1$m, $M = 200$	$r = 0.1$m, $M = 200$
Experiment 3	Full	$r = 0.15$m, $M = 60$	$r = 0.1$m, $M = 200$
Experiment 4	Restricted	$r = 0.15$m, $M = 60$	$r = 0.1$m, $M = 200$
Experiment 5	Full	$r = 0.08$m, $M = 391$	$r = 0.08$m, $M = 391$
Experiment 6	Restricted	$r = 0.08$m, $M = 391$	$r = 0.08$m, $M = 391$
Experiment 7	Full	$r = 0.1$m, $M = 200$	$r = 0.08$m, $M = 391$
Experiment 8	Restricted	$r = 0.1$m, $M = 200$	$r = 0.08$m, $M = 391$

Table 1: Summary of the parameters used in the eight different experiments. The full set of CPPN inputs includes the growth and environmental inputs while the restricted set does not.

Two final experiments, **experiments 7 and 8** follow the template of experiments 3 and 4. In both these experiments the resolution is increased halfway through each evolutionary trial, but in this case the resolution starts at $r = 0.1$ meters, $M = 200$ and increases to $r = 0.08$ meters, $M = 391$ after 100 generations. Once again these experiments are identical, beside experiment 7 using the full set of CPPN inputs while experiment 8 uses the restricted set. Table 1 summarizes all these experimental setups.

After completion of all evolutionary trials, statistics are computed for each experimental setup. Specifically of interest is how fit the evolved structures are, how long it takes for each evolutionary trial to complete, and how robust evolved CPPNs are to changes in the resolution at which they are queried.

Fig. 3 shows the mean best fitnesses from the final generation for each of the eight experiments. The first thing to notice in this figure is that in three of the four pairs of experiments the structures evolved using the full set of CPPN inputs (shown in red) on average achieve significantly higher fitness when compared with the structures evolved in the equivalent experiment using the restricted set. Moreover in the fourth pair of experiments (experiments 5 and 6) on average there is no significant difference in performance between those evolved with the full set of CPPN inputs and those evolved with the restricted set. This means that including the additional inputs never degrades performance of the evolved structures and more often than not leads to an improvement in performance.

Also noteworthy in this figure is that, when using the full set of CPPN inputs, on average there is no significant difference in the best fitness in the final generation when comparing between experiments that always evaluate structures at the highest resolution and those that spend the first half of their generations evaluating structures at a lower resolution (comparing experiment 1 with experiment 3 and experiment 5 with experiment 7). This is important, because as shown in Fig. 4 the evolutionary trials that spend their first 100 generations evaluating structures at a lower resolution run in significantly less time than those that always evaluate structures at the full resolution. This makes intuitive sense, because evaluating at the lower resolution requires fewer queries of the CPPN and allows for faster physical simulations due to the lower complexity of the structure being evaluated.

The final property of interest is how robust the evolved CPPNs are to changes in resolution. A key benefit of CPPN-NEAT over other evolutionary methods is that the evolved encoding are capable of producing structures at different

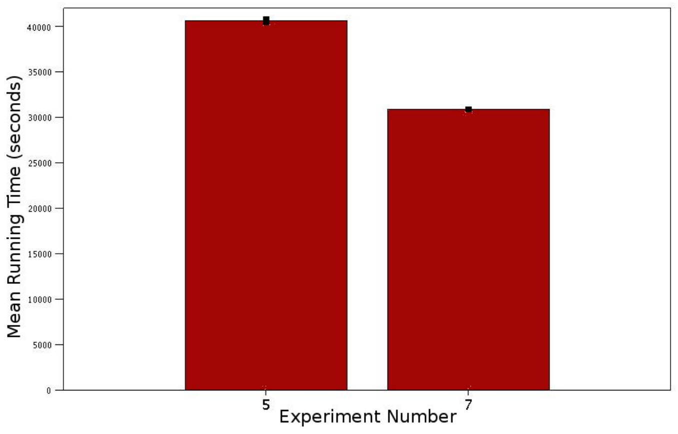


Figure 4: Mean running time in seconds with standard error bars for the two experiments that evolve structures at the highest resolution with the full set of CPPN inputs. Evolutionary trials in experiment 5 which always evaluate structures at the highest resolution take significantly longer than those of experiment 7 which evaluate structures at a reduced resolution for the first 100 generations.

resolutions, and as mentioned above if these structures are going to be physically fabricated it is important that they not be too sensitive to changes in resolution. Fig. 5 demonstrates how some of the evolved structures have similar dynamics and achieve similar performance when grown at different resolutions. This figure shows the dynamics of a single structure that achieves the best fitness in one of the evolutionary trials in experiment 5 first as it was evolved, then regrown at a lower resolution and finally regrown once more at a higher resolution. It is noteworthy how in all three cases a mass distribution is preserved that allows the structure to first fall onto its heavier end, carry its momentum through a horizontal rotation and fall once again away from its starting position.

Unfortunately, not all of the evolved structures preserve their dynamics like this example when grown at different resolutions. Fig. 6 compares the performance of structures grown from CPPNs evolved in experiments 5 and 6 by re-querying at both a lower and higher resolution than that at which they were evolved. One can see how once again using the full set of CPPN inputs improves performance. When growing structures at a resolution lower than that used during evolution those grown from CPPNs using the full set

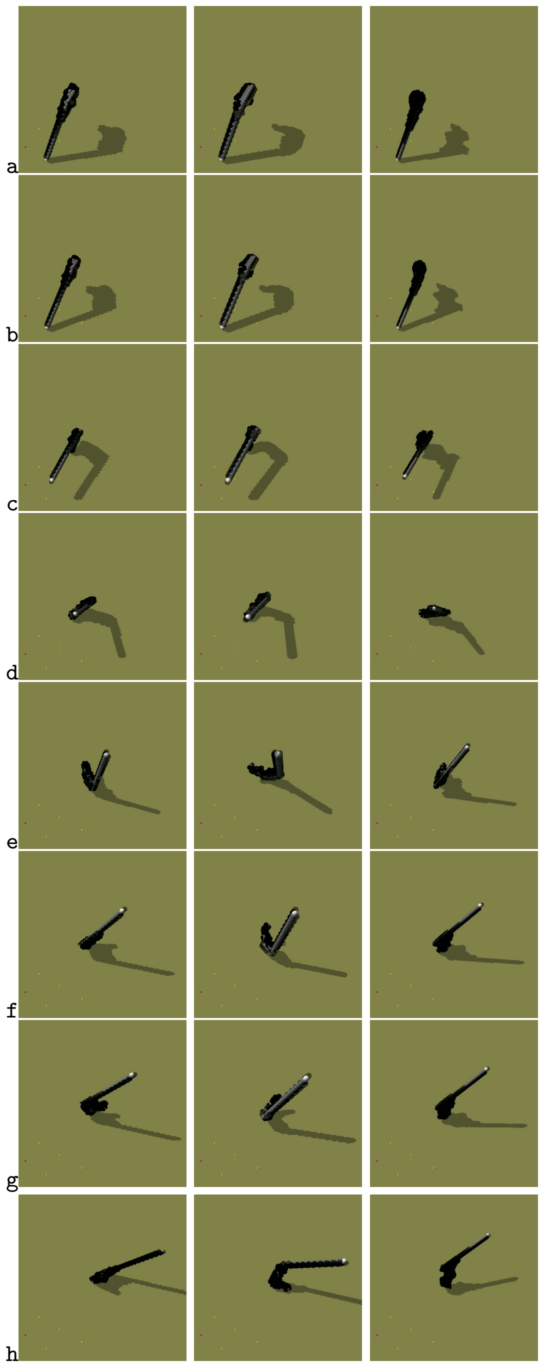
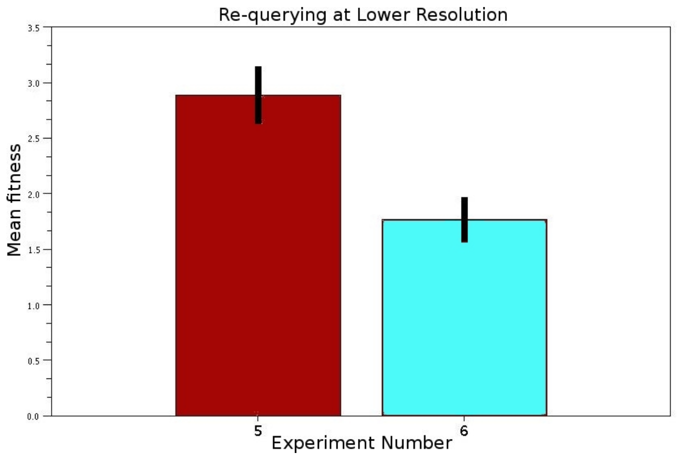


Figure 5: Left: Behavior of an evolved structure where each cell has radius $r = 0.08$ meters and is alloted a maximum $M = 391$ cells. Center: The CPPN used to generate the structure on the left is re-queried to produce a new structure at lower resolution: $r = 0.1$m, $M = 200$. Right: The same CPPN is re-queried again to produce a new structure at higher resolution: $r = 0.07$m, $M = 583$. Note how the three structures achieve similar performance even though structures were only evaluated at the $r = 0.08$m resolution during evolution. See Fig 7 for an enlarged view of these structures.



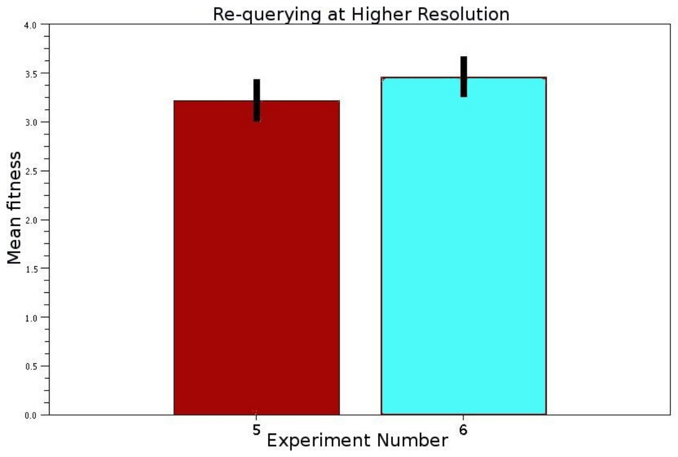


Figure 6: Top: Mean fitnesses of structures grown from CPPNs evolved in experiments 5 and 6 by re-querying at a lower resolution ($r = 0.1$m, $M = 200$) with standard error bars shown. Bottom: mean fitnesses of structures grown from the same CPPNs by re-querying at a higher resolution ($r = 0.07$m, $M = 583$.

of inputs on average significantly outperform those grown from CPPNs which use the restricted set (experiment 5 vs. experiment 6). When growing structures at a higher resolution than what their CPPNs were evolved for no significant difference in performance is observed between those using the full set of inputs and those using the restricted set.

4. CONCLUSIONS

This paper demonstrates that CPPN-NEAT is capable of evolving three dimensional physical structures with non-trivial dynamical properties. Moreover a case has been made for why including additional inputs which recursively provide the CPPN with information about the growth trajectory and environment (those in the full set) is beneficial when using CPPN-NEAT to evolve such structures. Specifically, structures evolved using these inputs on average perform either equivalently or significantly better when compared with those grown from CPPNs that are limited to the basic Cartesian inputs in the restricted set. Moreover including these inputs results in CPPNs that are either as

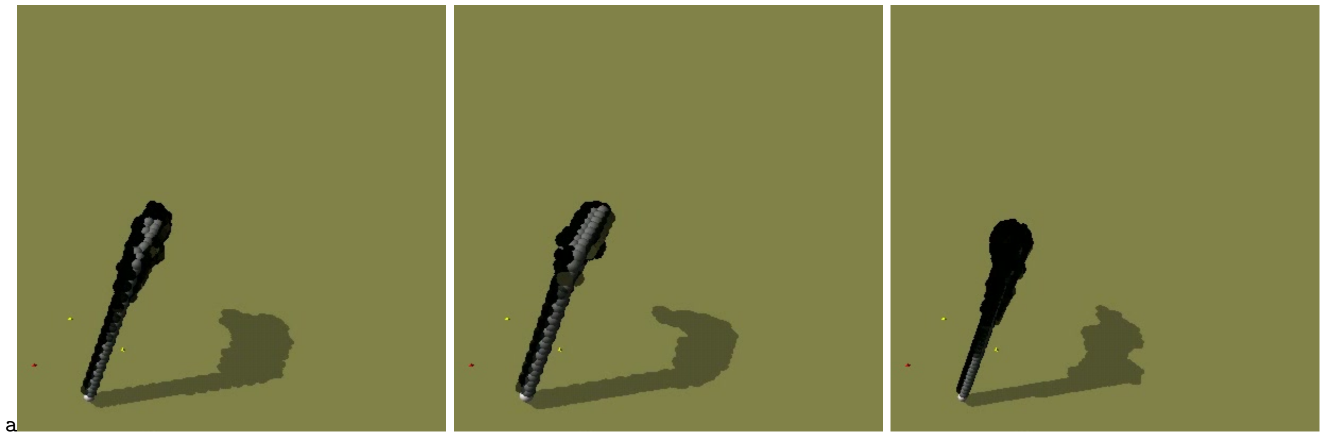


Figure 7: Enlarged snapshots of the structure shown in Fig. 5 grown at three different resolutions. Left to right: radius $r = 0.08$ meters, maximum $M = 391$ cells; $r = 0.1$m, $M = 200$; $r = 0.07$m, $M = 583$.

robust or more so to changes in growth resolution relative to CPPNs taking only the restricted set of inputs.

Additionally, this work demonstrates that it is possible to improve run time performance without significantly degrading the quality of evolved structures by using a lower resolution at the start of an evolutionary trial followed by increasing this resolution partway through. First evaluating structures at a lower resolution allows the evolutionary process to more quickly search the space of possible solutions before switching to a higher resolution to more further refine the shape and mass distribution of the structures.

This work is a first step in a research trajectory that aims to co-evolve articulated body plans and control policies. The results presented here are promising in this endeavor in that CPPN-NEAT is able to find non-intuitive solutions that capture the powerful yet subtle relationship between physical structure and function. Adding in articulation will involve extending the growth procedure presented here to additionally query evolved CPPNs for cell connectivity information. For example when adding a new cell to the morphology instead of always doing so in a rigid manner the CPPN can be queried with information regarding the two cells' geometric positions to produce another value used to determine whether the cells should be connected rigidly or with a rotational joint, and if they are to be connected with a joint, properties of this joint may also be determined from the CPPN output.

More work will be needed to extend the growth procedure to allow for the inclusion of arbitrary numbers of sensors on each cell, but the current results along with previous experiments using CPPNs suggest that additional CPPN outputs and/or input flags will provide the necessary mechanisms for extending the current framework in that direction. Additionally, since the HyperNEAT variant of CPPN-NEAT has shown success in the evolution of ANNs it is reasonable to expect that co-evolving neural network control policies along with the morphology should be possible. The authors intend to tackle these problems in future work.

5. REFERENCES

[1] A. Adamatzky, M. Komosinski, and S. Ulatowski. Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics*, 29(9/10):1344–1351, 2000.

[2] M. Anderson. Embodied Cognition: A field guide. *Artificial Intelligence*, 149(1):91–130, 2003.

[3] R. D. Beer. *Intelligence as adaptive behavior: an experiment in computational neuroethology*. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[4] R. D. Beer. The dynamics of brain-body-environment systems: A status report. In P. Calvo and A. Gomila, editors, *Handbook of Cognitive Science: An Embodied Approach*, pages 99–120. Elsevier, 2008.

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

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

[7] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pages 1872–1877, 2002.

[8] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1986.

[9] R. Brooks. *Cambrian intelligence*. MIT Press Cambridge, Mass, 1999.

[10] CGAFaq. Evenly distributed points on sphere – cgafaq, 2010. [Online; http://cgafaq.info/wiki/Evenly_distributed_points_on_sphere accessed 25-January-2010].

[11] J. Clune, B. Beckmann, C. Ofria, and R. Pennock. Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pages 2764–2771, 2009.

[12] J. Clune, R. T. Pennock, and C. Ofria. The sensitivity of hyperneat to different geometric representations of

a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2009.

[13] S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[14] F. Dellaert and R. Beer. Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.

[15] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, pages 205–213, 1997.

[16] P. Funes and J. Pollack. Computer evolution of buildable objects. *Fourth European Conference on Artificial Life*, pages 358–367, 1997.

[17] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224, 1997.

[18] G. Hornby and J. Pollack. Body-brain co-evolution using l-systems as a generative encoding. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 868–875, 2001.

[19] G. Hornby and J. Pollack. Evolving L-systems to generate virtual creatures. *Computers & Graphics*, 25(6):1041–1048, 2001.

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

[21] H. H. Lund and J. W. P. Lee. Evolving robot morphology. *IEEE International Conference on Evolutionary Computation*, pages 197–202, 1997.

[22] C. Mautner and R. Belew. Evolving robot morphology and control. *Artificial Life and Robotics*, 4(3):130–136, 2000.

[23] A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370, 2009.

[24] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology,Intelligence,and Technology.* MIT Press, Cambridge, MA, USA, 2000.

[25] R. Pfeifer and J. Bongard. *How the Body Shapes the Way We Think: A New View of Intelligence.* MIT Press, 2006.

[26] J. B. Pollack, H. Lipson, G. Hornby, and P. Funes. Three generations of automatically designed robots. *Artif. Life*, 7(3):215–223, 2001.

[27] E. Saff and A. Kuijlaars. Distributing many points on a sphere. *The Mathematical Intelligencer*, 19(1):5–11, December 1997.

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

[29] K. Stanley, D. D'Ambrosio, and J. Gauci. A Hypercube-Based encoding for evolving Large-Scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

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

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

[32] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:2002, 2001.

[33] R. Tedrake, T. Zhang, and H. Seung. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT, 2005.

[34] R. A. Watson, S. G. Ficici, and J. B. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Congress on Evolutionary Computation*, pages 335–342. IEEE Press, 1999.