# Upload Any Object and Evolve it: Injecting Complex Geometric Patterns into CPPNs for Further Evolution

Jeff Clune
Evolving AI Lab
University of Wyoming
jeffclune@uwyo.edu

Anthony Chen
Creative Machines Lab
Cornell University
apc52@cornell.edu

Hod Lipson
Creative Machines Lab
Cornell University
hod.lipson@cornell.edu

***Abstract*—Ongoing, rapid advances in three-dimensional (3D) printing technology are making it inexpensive for lay people to manufacture 3D objects. However, the lack of tools to help non-technical users design interesting, complex objects represents a significant barrier preventing the public from benefiting from 3D printers. Previous work has shown that an evolutionary algorithm with a generative encoding based on developmental biology–a compositional pattern-producing network (CPPN)–can automate the design of interesting 3D shapes, but users collectively had to start each act of creation from a random object, making it difficult to evolve preconceived target shapes. In this paper, we describe how to modify that algorithm to allow the further evolution of any uploaded shape. The technical insight is to inject the distance to the surface of the object as an input to the CPPN. We show that this *seeded-CPPN technique* reproduces the original shape to an arbitrary resolution, yet enables morphing the shape in interesting, complex ways. This technology also raises the possibility of two new, important types of science: (1) It could work equally well for CPPN-encoded neural networks, meaning neural wiring diagrams from nature, such as the mouse or human connectome, could be injected into a neural network and further evolved via the CPPN encoding. (2) The technique could be generalized to recreate any CPPN phenotype, but substituting a *flat* CPPN representation for the *rich*, originally evolved one. Any evolvability extant in the original CPPN genome can be assessed by comparing the two, a project we take first steps toward in this paper. Overall, this paper introduces a method that will enable non-technical users to modify complex, existing 3D shapes and opens new types of scientific inquiry that can catalyze research on bio-inspired artificial intelligence and the evolvability benefits of generative encodings.**

## I. Motivation and Previous Work

Recent advances in 3D printing technology have captivated hobbyists and spurred the desktop fabrication movement [22]. Current applications of desktop 3D printing include the production of tools, jewelry, art and prototypes. While these advances have empowered do-it-yourself communities, the ability to design complex 3D objects still presents a challenge to non-technical users [22]. Because traditional 3D design tools have steep learning curves, simplifying the design process would increase the ability of the public to use and benefit from 3D printers [22].

There is a long history of evolving 3D shapes, but early encodings tend to produce overly regular, blocky or recursive designs, and do not abstract the way that natural animals develop [3], [32], [18], [21]. Clune and Lipson [7] showed that complex, interesting shapes could be evolved with the CPPN generative encoding [26], which is based on principles from developmental biology [4]. Clune et al. then built a website called EndlessForms.com that allowed web visitors to collectively explore the space of CPPN-encoded objects via interactive evolution: users select the objects they are interested in, which serve as parents for the next generation [7]. To date, over 50,000 visitors from over 150 countries have evaluated nearly 4 million objects, demonstrating the sizable public interest in tools that can automate the design of 3D objects. The shapes produced via this crowdsourced experiment are complex and interesting (Fig. 1). They often look natural or engineered because they display regularities such as symmetry and repetition, with and without variation, which are hallmark properties of the CPPN encoding [26], [9], [7]. The results of the EndlessForm.com experiment echo those of Picbreeder.com, which was an earlier website that enabled the crowdsourced exploration of CPPN-encoded images [25].

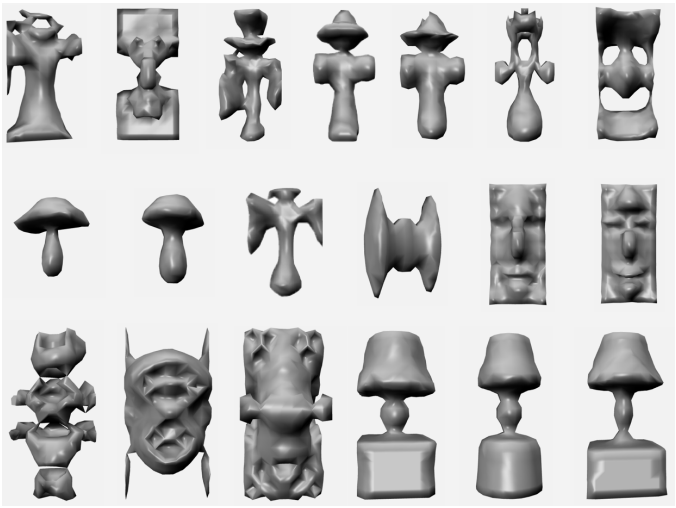

Fig. 1. Example objects evolved on EndlessForms.com.

While EndlessForms.com does allow users to further evolve objects that others have published, ultimately every object on the site started from a random CPPN genotype. By far, the most commonly requested feature by EndlessForms.com users is to be able to upload any shape and further evolve it. One reason that request is so popular is that it is difficult to evolve toward a predefined target because the stepping stones to a given target are not obvious and the path to it is extremely difficult for humans or machines to discover [20], [33]. There is thus a need for a technique that will enable users to upload

an arbitrary object and further evolve it.

It is easy to allow users to further evolve objects that were evolved with CPPNs, since the CPPN genome for such objects is available, but it is not obvious how to allow users to further evolve arbitrary, non-evolved objects. One idea would be set the uploaded object as a target, and select for CPPN-encoded shapes that increasingly resemble the target. That approach works somewhat for simple 3D shapes [7] and 2D images [33], but fails completely once the target is even slightly complex [33]. In the next section we describe an alternate method that allows the creation of a CPPN-encoded object for any shape, not by evolving toward a target, but by directly injecting the geometry of a target into the genome and thus allowing evolution to immediately start from that target.

## II. Methods

### A. Compositional Pattern Producing Networks

Because CPPNs have been repeatedly described in detail [7], [26], [27], [9], here we provide only a brief summary that is adapted from Clune and Lipson 2011 [7]. CPPNs abstract the natural processes of developmental biology without simulating the diffusion of chemicals [26]. In nature, the cells of organisms differentiate into different types, such as eye or skin cells, based on where they are located geometrically in the body [32]. To inform cells of their geometric position, processes unfold during development that create gradients of chemical morphogens that tell cells *where* they are and, therefore, *what* to become [32]. The anterior-posterior and dorsal-ventral axes in animals are often originally described via morphogen gradients provided by the mother. Genes in the embryo use these simpler gradients to create different, often more complex, geometric morphogen patterns, which become the inputs to genes that produce even more complex downstream patterns. This process allows the generation of arbitrarily complex geometric patterns that ultimately specify cell differentiation [32], [4].

CPPNs abstract this process by composing geometric patterns out of other geometric patterns. For computational efficiency, the patterns are represented mathematically instead of by simulating diffusing chemicals. A few simple gradients are provided to represent maternally-provided seed gradients. The final output patterns of a CPPN specify phenotypic attributes, and thus do so as a function of the different geometric locations of those phenotypic components. To encode a two-dimensional picture, for example, the coordinates of each pixel (e.g. $x = 4$, $y = 7$) are iteratively passed into a CPPN genome and the color of each pixel is determined by the corresponding CPPN outputs (Figure 2).

CPPNs are directed graphs where every node is a simple math function (e.g. Gaussian). These node functions can produce desirable phenotypic properties, such as repetition (e.g. a sine function) and symmetry (e.g. a Gaussian function). CPPNs can combine coordinate frames by composing functions of other functions. For example, a sine function early in the network could create a repetitive motif that can be input into a symmetrical Gaussian function to create a symmetrical, repeating pattern (Figure 2). This process can create geometric patterns of arbitrary complexity, as in the natural processes involved in biological development [32].

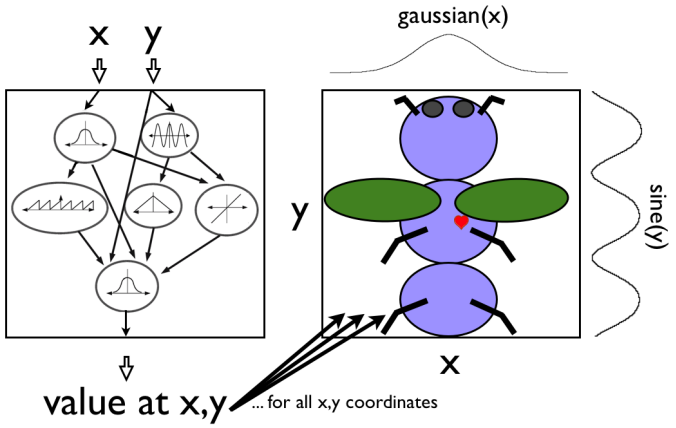


Fig. 2. CPPNs create regularities, such as symmetries and repeated motifs, with and without variation, by composing math functions out of the outputs of upstream functions within the network. Adapted from Stanley (2007).

The weights between nodes within CPPNs increase or decrease the values passed from node to node, just like with neural networks. Mutations altering these weights can thus enhance or diminish the influence of various patterns within the network.

During evolution, CPPNs experience crossover and mutations, the latter of which can add nodes or alter weights. The base functions for CPPN nodes in this study are sine, sigmoid, Gaussian, and linear. The population of CPPN networks are evolved with the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [28]. NEAT consists of three key principles [28]. (1) Genomes start with no hidden nodes and such nodes are added across generations via mutation. This process, called *complexification*, means that the complexity of patterns tends to increase across generations. (2) NEAT encourages diversity in the genotype space, protecting new genotypic innovations. (3) NEAT matches up homologous genotypic structure before crossover.

### B. Encoding 3D Objects with CPPNs

Following [7], to evolve 3D objects, inputs for the $x$, $y$, and $z$ dimensions are provided to a CPPN, along with additional inputs that were experimentally found to be helpful, specifically the Euclidean distance from center and the Euclidean distance from center in each of the $xy$, $yz$, and $xz$ planes. The experimenter defines a *workspace*–the maximum size of an object—and its *resolution* (the number of voxels in each dimension). In this study there are 20 voxels in each of the $x$, $y$ and $z$ dimensions. Voxels are considered full if the CPPN output for that voxel is greater than a threshold (here, 0.2), otherwise the voxel is left empty. The resulting array of voxels is then processed by the Marching Cubes algorithm [23], which smooths the surface. A normal is provided for each vertex, which the WebGL visualization engine uses to further smooth the object's surface. Thus, high-resolution CPPN objects can be visualized without extreme computational costs. All experimental parameters in this paper are identical to those in [7], although the presented technique is independent of, and thus will work with, any evolutionary parameters.

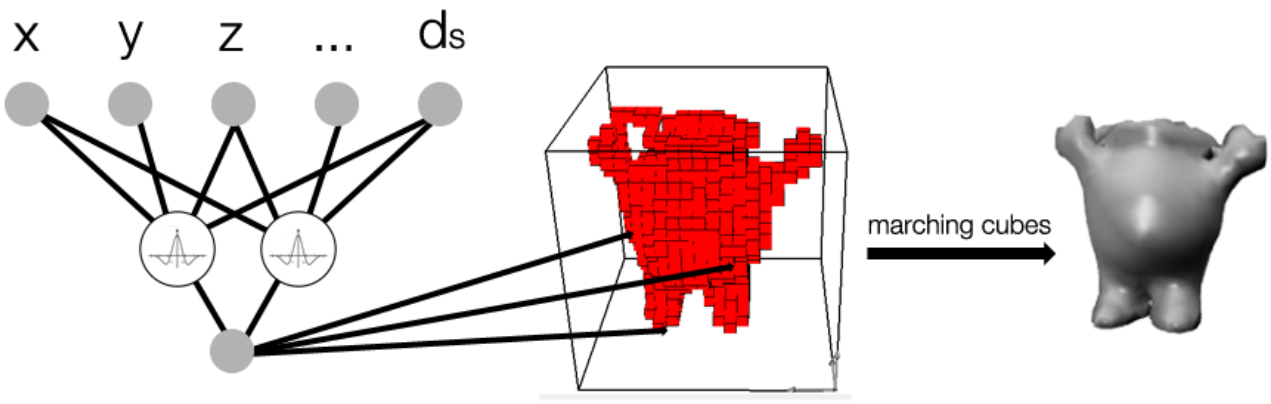


Fig. 3. A CPPN genome specifying an uploaded shape. In addition to the traditional $x$, $y$, $z$, and other inputs (see text), to specify an uploaded shape a distanceFromSurface ($ds$) input is added that informs the CPPN of the proximity of each voxel to the shape surface. Distances are negative outside the shape and positive inside. The CPPN output determines whether a voxel is present at each location based on these inputs. The centermost, contiguous cluster of voxels (shown as red voxels) is then smoothed by Marching Cubes [23] and WebGL. Genomes start with no hidden nodes, but can gain them via mutations.

This method of encoding 3D objects with CPPNs is used in EndlessForms.com [7] and is an extension of how 2D pictures are encoded with CPPNs on Picbreeder.com [26], [25]. It has also been used to evolve soft robots composed of multiple-materials [5]. Like most CPPN-encoded phenotypes, there is no growth process, unlike a different proposal for evolving 3D CPPN objects that produced shapes composed of attached spheres of different sizes [2], [1].

### C. Seeding CPPNs

No previous method exists for *starting* evolution with a generative encoding from a complex shape, although seeding musical motifs has been investigated [17], [10]. Previously, researchers set an object as a target and hoped evolution could produce it, at which point that object could be further evolved: that method fails for all non-trivial targets [33], [7]. One novel attribute of the CPPN encoding is that it is 'geometrically aware', meaning that it specifies phenotypic attributes as a function of their geometric location [26], [8], [9], [15]. That enables generating a CPPN-encoded replica of an arbitrary shape by making the CPPN aware of the geometric shape to be replicated. One way to start from a complex design would be to evolve a CPPN object and then add or subtract it to the desired shape (e.g. an evolved cylinder in the right location could be added to Pinocchio to lengthen his nose). This technique allows additions or ablations to a shape, but does not allow the original shape to be scaled, warped, extended, or deformed in interesting ways. Another approach is to provide an input bit that would be on for every voxel in the shape, but that method similarly does not allow the shape to be scaled, transformed, or warped, since voxels close to the surface of the shape would not be distinguishable from voxels farther away. Instead, the key is to let the CPPN know the distance between each voxel and the surface of the shape, as well as whether the voxel is inside or outside the shape, which enables the CPPN to scale and deform the shape in arbitrary ways.

We define the *seed object* as the object to be replicated and the *seeded-CPPN-object* as the resulting CPPN-based replica object. To produce a seeded-CPPN-object, we add a *distanceFromSurface* CPPN input. For each voxel, this input value, *ds*, is the shortest distance from that voxel to the surface of the seed object, and is negative if outside the shape and positive if inside. Thus, *ds* input values increase closer to an object's surface and are maximal at the object's center.

For each seed object, the *ds* inputs are pre-calculated once for every voxel in the workspace and are stored for use throughout evolution. Objects are scaled to fit into the workspace and are represented as a set of triangles that form the surface of the object, from which an AABB tree is constructed. An AABB tree is a hierarchical decomposition of an object's bounding box into axis-aligned bounding boxes (AABBs), which each contain a triangle [13]. Arranging AABBs in a tree logarithmically reduces the number of intersection tests necessary for calculating the distance from each voxel to the object surface. This process is computationally expensive for high resolutions, but has to be performed only once per uploaded shape and workspace resolution. For example, for a relatively low-resolution shape consisting of 5512 triangles and a $20^3$ workspace, 8000 distance calculations were performed on a modern computer in 87 seconds. Experiments were conducted within the EndlessForms.com code base and viewed in the Chrome browser, which displays and smooths objects via WebGL. Figure 3 illustrates the process.

In addition to the *ds* input, traditional CPPN inputs such as $x$, $y$, and $z$ are included to allow evolution to morph the original shape by incorporating influence from other dimensions. For example, adding $x$ to the *ds* input could cause the original shape to be recognizable, but get larger from left to right.

Initial seeded-CPPN genomes have no hidden nodes, but complexify via the NEAT algorithm. Typically, the initial weights between input and output nodes are randomized [26], [27], [9], [7]. However, we discovered that random weights often led to a few strong weights from input dimensions other than *ds*, which masked the signal coming from the shape and made the result unrecognizable. To ensure that objects would initially resemble their respective seed objects, we set

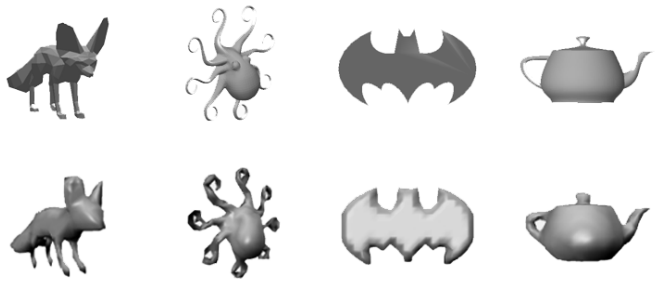

Fig. 4. For any uploaded shape (top row), a seeded-CPPN can be created that produces a similar replica shape (bottom row). Inaccuracies occur due to low workspace resolution, but can be remedied with higher resolutions (Fig. 5). Seed objects in the top row are visualized in an STL viewer; seeded-CPPN-objects in the bottom row are visualized in the Chrome browser.

Fig. 5. Increasing the workspace resolution enables higher fidelity replications of the seed object. Computational costs increase linearly with the number of voxels. Left: the default resolution of $20\times20\times20 = 8000$ voxels (requiring 87 seconds) . Middle: $40^3 = 64000$ voxels (694 seconds) . Right: $80^3 = 512000$ voxels (5453 seconds) . All other images in this paper are at the default $20^3$ resolution due to computational costs.

the weight between the *ds* input and the output node to a high value (experimentally chosen to be 1.16) and initialized all other weights to zero. All weights were free to change via mutation and selection during evolution. Mutations during evolution can mask or even ignore the original shape, but the user can simply not select such individuals.

## III. Results

### A. Ability to generate arbitrary objects

The Seeded-CPPN technique can produce CPPN-encoded objects that resemble the seed object. To demonstrate this result, we uploaded different designs to our software in STL (STereoLithography) format, the standard file type for 3D objects. We found that every shape we tried could be represented by a CPPN to a high degree of accuracy (Fig. 4). It appears that any 3D shape can be created, and we are unaware of any a priori reason why certain shapes would be unreproducible, but future work is needed to test this expectation. For computational reasons, our default resolution was sometimes too low to capture every feature of the seed object, resulting in a lower-resolution representation of the object that lacked some of its features (e.g. the eyes of the octopus in Fig. 4). This appears to be the main, and possibly only, source of inaccuracy.

To test whether seeded-CPPNs could increasingly capture finer features of seed objects as the workspace resolution increased, we applied the technique to the same object, but at increasing workspace resolutions. We found that higher workspace resolutions produce higher fidelity objects that captured increasingly fine details of the seed object (Fig. 5). This result is encouraging because as computers continue to increase in speed, higher fidelity seeded-CPPN-objects will become feasible. The computational cost increases linearly with the number of voxels (Fig. 5).

### B. Evolving seeded-CPPN-objects

We evolved many different seeded-CPPN-objects, and found that evolution with the CPPN encoding can produce a variety of changes to the original shape while still preserving that shape, or at least retaining some of its properties (Figs. 6 and 7). Importantly, the CPPN encoding allows the various input dimensions to have differing influences on the design. This enables a host of *regular* transformations of the seed

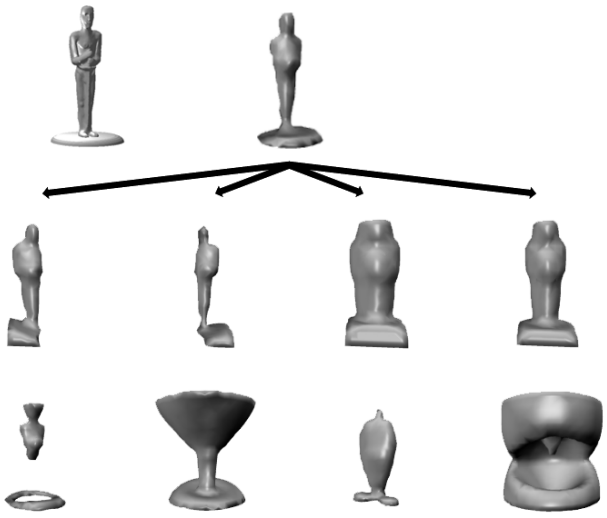

Fig. 6. Evolved descendants of an uploaded object (the statue given out at the Oscars). The seed object, viewed in an STL viewer, is shown in the top left. To its right is the recreated version encoded by a CPPN. The middle and bottom rows contain objects that were evolved from that seeded-CPPN-object. Example descendants here, and elsewhere in the paper, are chosen from the first 35 generations, usually from the first 5-10, with the seeded-CPPN object counting as generation 0. Note the presence of both symmetric and asymmetric alterations, as well as scaling and more complex, unexpected variations. Complexification over time is also evident: the bottom-row objects are from later generations than those in the middle row, and their extra hidden nodes increase their difference from the ancestor.

object, such as the left-to-right and right-to-left deformation seen in the left two columns of the middle row of Fig. 6. The original object can also be scaled, as seen in the two different scales portrayed in the right two columns of the middle row of that figure.

The bottom row of Fig. 6 shows more complex transformations. The left column likely resulted from a reduction in scale, which eliminated the legs of the statue and part of the base. The object to its right demonstrates that CPPNs can alter one shape into a completely different one, which is an important trait for an encoding [29]. Interestingly, the first author was once challenged by an organization that had a martini glass logo to

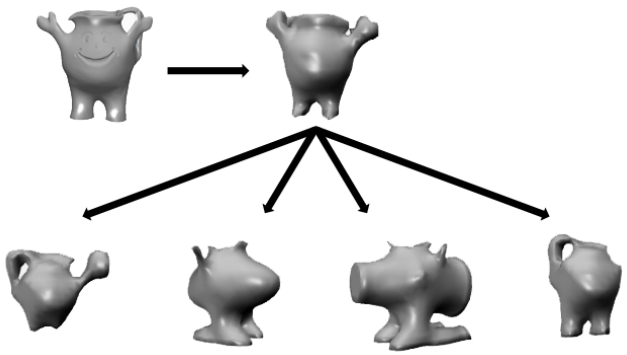

Fig. 7. Evolving the Kool-Aid man. The original STL version is in the top left, with the CPPN recreation to its right. The bottom row shows evolved descendants. These shapes—which resemble watering cans, aliens, piggy banks, and clay jugs—demonstrate that evolution can make non-trivial changes to seeded-CPPN-objects.

evolve a martini glass on EndlessForms. The attempt failed, even when the author used available designs on EndlessForms as starting points. Here, unexpectedly, something resembling a martini glass emerged. This serendipitous discovery underscores that the stepping stones that may lead to a particular target are not obvious ahead of time [33].

The third column of the bottom row of Fig. 6 reveals a mutation that eliminated much of the radial symmetry of the statue, stretching it along one plane. The result makes the statue appear fatter and to have clown shoes. Such interesting and even humorous alterations are the sort of spontaneous creativity that delight users in systems that help automate design. The bottom right object in Fig. 6 also shows evolution's ability to surprise. It uses the mathematical description of the statue to create something entirely different, yet non-trivial.

Fig. 7 also shows complex, unexpected, interesting variants of an uploaded object: the Kool-Aid man. From that iconic starting place, objects that are clearly descendant, but also quite different, evolve. Objects that resemble watering cans, aliens, piggy banks, and clay jugs were just some of the many fascinating designs evolved. Interestingly, some of these deformations are the sort of changes a human would make, such as elongating the Kool-Aid Man's tubular legs into feet, but are not trivial extensions of the original design. Of course, humans are doing the selecting, but the encoding has to generate such variation in the first place. Generating interesting, nuanced variation has historically been difficult, even during interactive evolution, because of less sophisticated encodings or symbolic encodings where one mutation often radically alters the phenotype [26], [29], [7], [25].

Because NEAT increases complexity over time by adding hidden nodes to CPPN genomes, the shapes become more complex, and different, across evolutionary time. This can be seen in Fig. 6. Objects in the middle row are from early generations (generation 10 or before), whereas objects in the bottom row are from later generations (generation 20 or later). These evolved objects validate that many of the key properties of CPPN-NEAT, such as complexification and the generation of complex regularities, occur in these objects. Such properties have previously been shown to be important aesthetically [25], [7] and for performance, such as when CPPNs encode neural networks [26], [27], [9], [8], [14], [15].

## IV. Discussion and Future Work

### A. Qualitative vs. Quantitative Results

The purpose of this paper is to introduce a technique showing that it is possible to start evolution from a complex 3D shape, instead of always starting from random genomes. A common reaction to this type of research is to ask whether the new technique can be quantifiably shown to be better than some previous technique. For example, one could see whether the Seeded-CPPN technique reproduces targets with less error than an alternate method, such as trying to evolve a target via a fitness function based on target-reconstruction error. We do not make such a comparison for a number of reasons. First, there is no other method that we are aware of to start evolution with generative encoding from a specific, complex target shape. Second, because the Seeded-CPPN technique can represent a 3D pattern to an arbitrary degree of precision (Fig. 5), the error between the object and the target can be tuned toward zero by increasing the resolution, which makes quantifying that error uninteresting. Techniques that do involve significant error, such as target-oriented evolution, would clearly not compete. Third, it has been shown that setting a target for evolution and having it try to evolve to produce that target fails for all but the simplest pictures and objects, such as three overlapping spheres [33], [7]. There is thus no reason to believe that target-based evolution can evolve more complex shapes like the fox, octopus, teapot, Kool-aid man, Mario, and the other shapes replicated earlier (Figs. 4,5,6,7). CPPNs even fail to evolve targets that were originally evolved with CPPNs [33], revealing that target-based evolution is unlikely to compete with the Seeded-CPPN technique, even to produce CPPN-evolved objects such as the "Angel with Halo" (Fig. 8). Quantitative comparisons are therefore unlikely to further our understanding of the advance that seeded-CPPNs yield in terms of beginning evolution from a complex starting point.

Another reason quantifiable results are not the best way to understand the Seeded-CPPN technique is because humans are far better than computers at recognizing visual patterns. Scientists can learn a lot from visual domains that cannot be easily quantified. In fact, the CPPN encoding itself, as well as Novelty Search, were invented as a direct result of investigations in visual, non-numerical domains like Picbreeder [26],

Fig. 8. An object originally evolved on EndlessForms.com that, for this study, is further evolved with its original CPPN genome and with a genome created via the Seeded-CPPN technique.

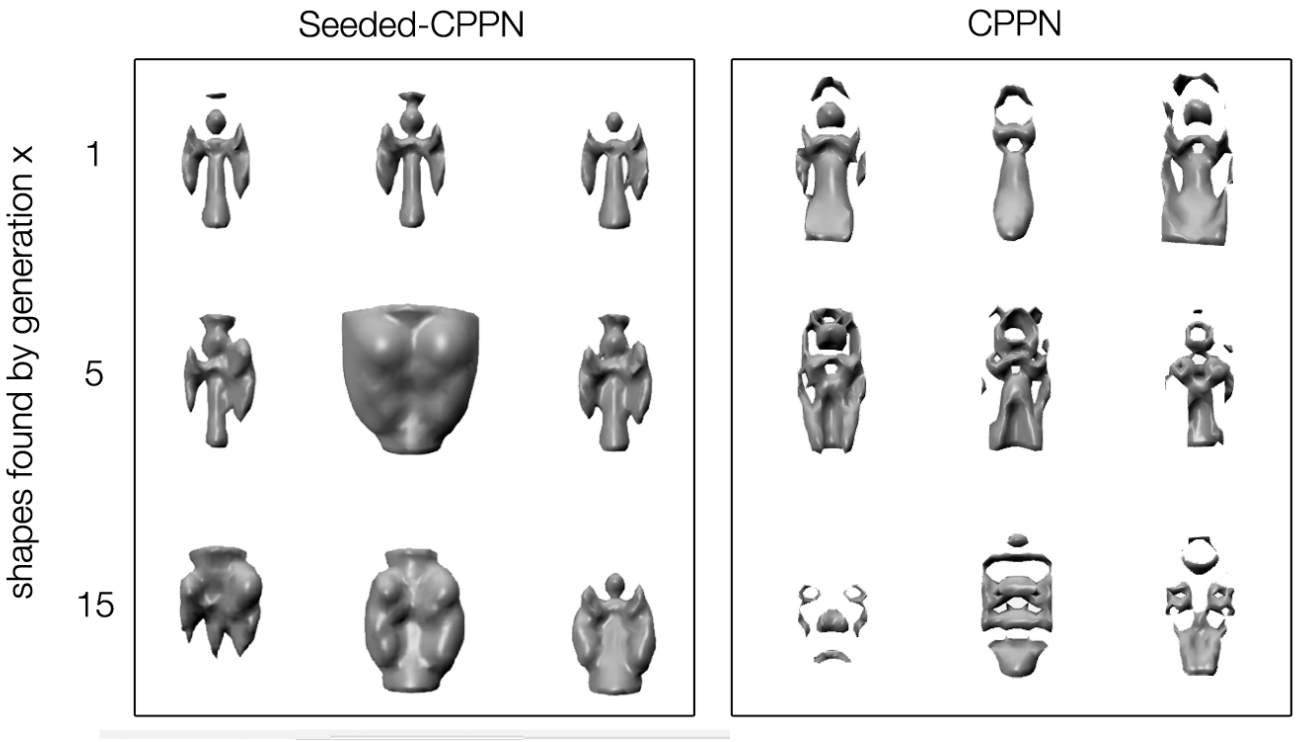


Fig. 9. A comparison of evolutionary variation produced when evolving the "Angel with Halo" object from EndlessForms (Fig. 8) with two different CPPN representations: its original rich CPPN genome and a flat CPPN genome produced by the Seeded-CPPN technique. Overall, the original CPPN genome produces more substantial changes across generations, although its designs are still interesting instead of random. The seeded-CPPN lineages feature less change overall, but still produce interesting variation on the ancestral shape. In some cases, however, the changes produced by the seeded-CPPN encoding are quite striking, as in the center object that resembles a sculpture of a muscular human torso. Note: For space limitations, per generation, only three representative objects of the 15 total are shown that evolved at or before that generation. All 15 objects per generation are visible at the following link: http://dx.doi.org/10.5061/dryad.4qk42

[20], [25]. As scientists, we should not categorically preclude ourselves from the valuable insights visual domains have to offer just because they resist quantification.

In addition to introducing the Seeded-CPPN technique and documenting that it allows evolution to start from complex targets, an additional focus of this paper is to discuss new types of scientific inquiry that the Seeded-CPPN technique enables, which can produce important, quantifiable data in future studies. We discuss those possibilities next.

### *B. Evolvability*

One of the goals of research into generative encodings is to improve evolvability—the speed with which adaptation occurs [24], [29]. It is hypothesized that in nature, evolution discovers types of variation that tend to produce more adapted offspring, and thus evolvability increases across generations [31], [30], [24]. For example, if it is helpful to have the size of limbs be correlated, or even symmetrical, evolution could encode those limbs with the same genes to enforce their similarity. Generative encodings should similarly learn what types of variation are helpful, and produce such variation while avoiding deleterious variants, a process known as *canalization* [29]. It is hypothesized that the CPPN generative encoding does increase evolvability in this way, but such claims have previously been difficult to test [26], [9], [7]. The difficulty in investigating this subject arises because, for any given CPPN-encoded phenotype, it is difficult to further evolve the object with and without the representation that evolved inside the genome. It is thus hard to separate the encoding itself from any evolvability it learns along the way. Some attempts have been made to investigate evolvability by comparing a generative encoding to a direct encoding [9], but such comparisons are imperfect because the different encodings mean that improved evolvability itself is not isolated.

The Seeded-CPPN technique for the first time allows comparisons between phenotypes with identical generative encodings that either have, or do not have, extended evolutionary histories. This is possible because the Seeded-CPPN technique can generate a CPPN without hidden nodes that encodes for an object that was originally evolved with the CPPN encoding. The original may have canalized certain forms of variation, but such biases would not be present in the seeded-CPPN. The Seeded-CPPN technique thus allows studies into whether evolutionary *rich* encodings are more evolvable than the *flat* encodings produced by the Seeded-CPPN-object technique. The seeded-CPPN genomes are initially flat because they have no hidden nodes, and no canalization, in contrast to the evolved CPPN networks that have many hidden nodes that compose different mathematical regularities in particular, ordered ways. If evolvability is documented in the rich encodings vs. the flat encodings in early generations, it will also then be possible to see how evolvability increases in the (initially) flat encodings over evolutionary time as they are further evolved and add hidden nodes.

Protracted studies in this vein are beyond the scope of this paper and will be explored in future studies. However, we do take an initial step in that direction. We select an object evolved with the CPPN encoding on EndlessForms.com, titled "Angel with Halo" (Fig. 8), and then generate a replica of it via the Seeded-CPPN-object technique. The evolved genome for this object from EndlessForms.com has 33 hidden nodes, compared to the 0 hidden nodes for the seeded-CPPN genome. To explore the impact of the evolved vs. injected representation, we further evolve both shapes to see if there are dissimilarities in the way evolution operates. To eliminate the bias of a human performing selection, we randomly choose three objects per generation (out of a possible 15) to be the parents of the next generation. The results show interesting differences in how evolution occurs across generations (Fig. 9). The original CPPN encoding makes larger leaps through phenotype space, quickly leading to shapes that are interesting, but do not resemble the starting object. Even the shapes in the first generation are vastly different than their parent organism and no longer look like an angel, although many retain the humanoid form. By generation 5, the objects from the original CPPN encoding seem like druids or aliens. By generation 15 the objects look like faces or masks, and do not resemble the human body plan.

In contrast, objects evolved with the seeded-CPPN genome are initially quite similar to the ancestor (Fig. 9). By generation 5, differences are apparent in all objects, but many objects are noticeably similar to the angel ancestor. One striking exception is the object that looks like a sculpture of a muscular human torso (Fig. 9, middle row and column). This object demonstrates that the seeded-CPPN encoding can produce substantial, surprising variation. After 15 generations of random selection, some objects evolved with the seeded-CPPN genome still resemble humanoids, such as the object that resembles a monk (Fig. 9, bottom row, right column). Other objects, however, are quite different, such as the object in Fig. 9's lower-left corner.

In addition to facilitating evolvability studies with interactive evolution, evolvability can be studied with non-interactive evolution, such as the evolution of neural networks. Neural networks have been evolved with the CPPN encoding in an algorithm called HyperNEAT [27]. Such networks have generated impressive, high-performing gaits in simulated and real quadruped robots [9], [6], [19], evolved interesting soft-robot morphologies [5], performed board evaluations in games like checkers [16], [15], and effectively controlled robot swarms [11], [12]. Those neural networks could be stripped of their internal CPPN genotypic structure by producing a flat CPPN via the Seeded-CPPN technique. The flat CPPN genome could then be further evolved on the same or similar tasks. Performance comparisons between the flat and rich CPPN genomes would then allow quantifiable, objective experiments into whether any evolvability was acquired in the original evolutionary run. Such objective studies would not have humans in the loop, and would thus complement studies done with humans performing fitness evaluations. If similar results are observed in both types of studies, we could conclude that human evaluators are not the driving cause of evolvability differences. Future work is needed to investigate the degree to which the Seeded-CPPN technique can produce neural networks as well as it can 3D objects, but if it proves possible the ability to investigate evolvability in neural networks would be enhanced.

### C. Seeding Neural Patterns

The benefits of being able to inject a complex geometric pattern into a CPPN open other new doors in the field of artificial intelligence. Developments in the technique may allow any pattern to be seeded into a CPPN, including the neural wiring diagram of animal brains. ANNs based on the connectome of mice, humans, and other animals could be created and then further evolved. It is unlikely that the known connectome would work "out of the box", so techniques for automatically optimizing and adapting it are necessary. CPPNs are an ideal encoding for such research, since they can scale to very large neural networks, including those with millions of connections [27]. It is not currently computationally feasible to evolve human-scale brains, but smaller animals may be possible. Many details remain to be worked out regarding whether and how this technique will work. Instead of providing a distance to the surface of a shape, for example, a CPPN could be provided with the distance to biological neural structures (e.g. neurons, synapses) for each potential neural structure in an artificial neural network.

If such new types of experiments prove fruitful, the research in this paper with 3D objects may help fuel the intuitions of scientists performing neural network investigations with the Seeded-CPPN technique. It is helpful to have a visual testbed with which to understand the types of variation that are possible with various genetic representations. Researchers could gain such intuitions with 3D objects and then utilize that understanding when dealing with high-dimensional spaces that are difficult to visualize, such as the four-dimensional, hypercube-encoded neural networks in HyperNEAT.

### D. Automating Object Design

The Seeded-CPPN technique could greatly expand the number of people that can change the objects that surround them in the world. With 3D printing technology, they can modify such objects and then easily manufacture them. For example, on EndlessForms.com, users evolve objects and then click the "3D print" button, which enables them to have the company Shapeways print the object and ship it to them for a small fee. Objects can be printed in a variety of materials, from gold or glass to silver and sandstone. Many EndlessForms.com users report liking the ability to print evolved objects, but would prefer even more to print slightly modified versions of objects available in online repositories like Thingiverse.com. In future work we plan to add this functionality to EndlessForms.com in order to discover the degree to which users appreciate and utilize this often-requested functionality. Such work will shed light on whether users can better accomplish their goals when starting from a seed, as well as whether different classes of novel designs emerge when interactive evolution can start from complex, human-engineered designs. Additional possibilities are enabled by 3D scanners, which allow non-technical users to scan everyday objects—or even themselves—and further evolve them. Finally, the benefits of automated 3D design are not limited to 3D printing, but can also create avatars and other objects in virtual worlds.

## V. Conclusions

The Seeded-CPPN technique presented in this paper brings closer the day in which non-technical users can create and

manufacture any design they wish. By allowing users to further evolve arbitrary, complex shapes, they can start close to the goal they have in mind. Evolution can then help them morph their seed object into their desired goal, or may surprise them by suggesting a design that is superior or more preferable than the one they had in mind.

We have shown that it is possible to encode any 3D object with a CPPN network. Because the CPPN network has many desirable properties [26], [9], [8], users can harness the encoding to further evolve any object.

The Seeded-CPPN technique also suggests new scientific tools. With it, scientists can study whether CPPNs evolve evolvability. Thus, researchers can learn about the CPPN encoding in particular, and about the nature of evolvability itself, which remains a difficult-to-study, yet important, area of biological research [24], [31], [30]. Additionally, the Seeded-CPPN technique may one day enable the creation of CPPN-encoded replicas of natural wiring diagrams, such as those for animal brains, which could then be further evolved. That possibility may open new avenues of research in artificial intelligence and computational neuroscience. Overall, the Seeded-CPPN technique has immediate practical applications in design automation and introduces new possibilities for research into generative encodings, evolvability, and artificial intelligence.

## VI. Acknowledgments

We thank AV Terekhov, Kenneth O. Stanley, Kristin Soriano, Jason Yosinski, NSF CDI (0941561), DARPA OM (W911NF-12-1-0449), and an NSF Postdoctoral Research Fellowship in Biology to Jeff Clune (DBI-1003220).

## References

[1] J. Auerbach and J. Bongard. Dynamic resolution in the co-evolution of morphology and control. In *Proceedings of Artificial Life XII*, 2010.

[2] J. Auerbach and J. Bongard. Evolving CPPNs to grow three-dimensional physical structures. In *Proceedings of the genetic and evolutionary computation conference*, pages 627–634. ACM, 2010.

[3] P. J. Bentley. *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*. PhD thesis, University of Huddersfield, 1996.

[4] S. Carroll. *Endless forms most beautiful: The new science of evo devo and the making of the animal kingdom*. Norton, 2005.

[5] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2013.

[6] 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 Computation*, pages 2764–2771, 2009.

[7] J. Clune and H. Lipson. Evolving three-dimensional objects with a generative encoding inspired by developmental biology. In *Proceedings of the European Conference on Artificial Life*, pages 144–148, 2011.

[8] J. Clune, C. Ofria, and R. Pennock. The sensitivity of HyperNEAT to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 675–682, 2009.

[9] J. Clune, K. Stanley, R. Pennock, and C. Ofria. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 15(4):346–367, 2011.

[10] P. Dahlstedt. Autonomous evolution of complete piano pieces and performances. In *Proceedings of Music AL Workshop*. Citeseer, 2007.

[11] D. D'Ambrosio, J. Lehman, S. Risi, and K. Stanley. Evolving policy geometry for scalable multiagent learning. In *Proc. of the Conference on Autonomous Agents and Multiagent Systems*, pages 731–738, 2010.

[12] D. D'Ambrosio and K. Stanley. Generative encoding for multiagent learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 819–826. ACM, 2008.

[13] C. Ericson. *Real-time collision detection*. Morgan Kaufmann, 2004.

[14] J. Gauci and K. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 997–1004. ACM, 2007.

[15] J. Gauci and K. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 628–633. AAAI Press, 2008.

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

[17] A. K. Hoover, P. A. Szerlip, M. E. Norton, T. A. Brindle, Z. Merritt, and K. O. Stanley. Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. In *International Conference on Computational Creativity*, page 111, 2012.

[18] G. Hornby, H. Lipson, and J. Pollack. Generative representations for the automated design of modular physical robots. *IEEE Transactions on Robotics and Automation*, 19(4):703–719, 2003.

[19] 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. In *Applications of Evolutionary Computing*. Springer, 2013.

[20] J. Lehman and K. Stanley. Exploiting open-endedness to solve problems through the search for novelty. *Artificial Life*, 11:329, 2008.

[21] A. Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968.

[22] H. Lipson and M. Kurman. *Fabricated: The New World of 3D Printing*. Wiley, 2013.

[23] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, 1987.

[24] M. Pigliucci. Is evolvability evolvable? *Nature Reviews Genetics*, 9(1):75–82, 2008.

[25] J. Secretan, N. Beato, D. D'Ambrosio, A. Rodriguez, A. Campbell, J. Folsom-Kovarik, and K. Stanley. Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evolutionary Computation*, 19(3):373–403, 2011.

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

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

[28] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

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

[30] G. Wagner. Homologues, natural kinds and the evolution of modularity. *Integrative and Comparative Biology*, 36(1):36, 1996.

[31] G. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.

[32] L. Wolpert and C. Tickle. *Principles of Development*. Oxford University Press, 4th edition, 2010.

[33] B. Woolley and K. Stanley. On the deleterious effects of a priori objectives on evolution and representation. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 957–964, 2011.