Compositional Pattern Producing Networks: A Novel Abstraction of Development

Kenneth O. Stanley (kstanley@cs.ucf.edu)
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816 USA

To appear in:
Genetic Programming and Evolvable Machines
Special Issue on Developmental Systems, New York, NY: Springer, 2007

Abstract

Natural DNA can encode complexity on an enormous scale. Researchers are attempting to achieve the same representational efficiency in computers by implementing *developmental encodings,* i.e. encodings that map the genotype to the phenotype through a process of growth from a small starting point to a mature form. A major challenge in in this effort is to find the right level of abstraction of biological development to capture its essential properties without introducing unnecessary inefficiencies. In this paper, a novel abstraction of natural development, called Compositional Pattern Producing Networks (CPPNs), is proposed. Unlike currently accepted abstractions such as iterative rewrite systems and cellular growth simulations, CPPNs map to the phenotype without local interaction, that is, each individual component of the phenotype is determined independently of every other component. Results produced with CPPNs through interactive evolution of two-dimensional images show that such an encoding can nevertheless produce structural motifs often attributed to more conventional developmental abstractions, suggesting that local interaction may not be essential to the desirable properties of natural encoding in the way that is usually assumed.

1 Introduction

Representation is at the heart of artificial intelligence. Particularly when a problem involves search, a good representation of the solution space can mean the difference between success and failure. In evolutionary computation, as traditional fixed-length genomes and direct encodings reach their limits, the importance of representation has moved to the forefront of research. Meanwhile, biology remains a potent reminder of how far there is to go. In biology, the genes in DNA represent astronomically complex structures with trillions of interconnecting parts, such as the human brain [21, 40, 85]. Yet DNA does not contain trillions of genes; rather, somehow only 30 thousand genes encode the entire human body [23].

There is thus reason to believe that the discovery of systems as complex as humans is only possible through extraordinarily efficient encoding, as seen in nature. A key process in natural encoding is *development*, in which DNA maps to the mature phenotype through a process of growth that builds the phenotype over time. Development facilitates the reuse of genes because the same gene can be activated at any location and any time during the development process. Thus a small set of genes can encode a much larger set of structural components.

This observation has inspired an active field of research in artificial *developmental encodings* [4, 10, 35, 74]. The aim is to find the right abstraction of natural development for a computer running an evolutionary algorithm, so that it can begin to discover complexity on a natural scale. Abstractions range from low-level cell chemistry simulations to high-level grammatical rewrite systems [74]. A common facet of such abstractions is that the state of an individual phenotypic component at one moment in development affects the states of components in the same vicinity in the future, that is, development unfolds through local interactions. Because no abstraction so far has come close to discovering the level of complexity seen in nature, much interest remains in identifying the properties of abstractions that give rise to efficient encoding.

This paper introduces a novel abstraction of natural development that breaks the strong tradition of local interaction and temporal unfolding in developmental encoding research. Yet it is nevertheless demonstrated that it captures several essential characteristics of such encoding. The fundamental insight behind this new encoding, called Compositional Pattern Producing Networks (CPPNs), is that it is possible to directly *describe* the structural relationships that result from a process of development without simulating the process itself. Instead, the description is encoded through a composition of functions, each of which is based on observed gradient patterns in natural embryos.

Because CPPNs are structurally similar to artificial neural networks, they can right away take advantage of existing effective methods for evolving neural networks. In particular, the Neuroevolution of Augmenting Topologies (NEAT) method evolves increasingly complex neural networks over generations [72, 73, 75]. In this paper, with only slight adjustment, NEAT is modified to create CPPN-NEAT, which evolves increasingly complex CPPNs. In this way, it is possible to evolve increasingly complex phenotype expression patterns, complete with symmetries and regularities that are elaborated and refined over generations.

The main contribution of this paper is to establish CPPNs as a legitimate abstraction of natural developmental encoding. To provide evidence for this claim, a novel experimental approach to analyzing the properties of genetic encodings is introduced: CPPN-NEAT-based interactive evolutionary computation (IEC), in which a human select the parents of each new generation, is used to produce examples of several fundamental structural motifs and elaborations of nature. Symmetry, reuse, reuse with variation, preservation of regularities, and elaboration of existing regularities all are demonstrated and capitalized on by CPPN-NEAT, validating it as a promising new abstraction of natural developmental encoding, and one that is able to evolve increasingly complex patterns.

While the primary conclusion is simply that CPPNs are legitimate abstractions of developmental encoding, the ramifications of this conclusion are significant. Admitting a new class of encoding that lacks classic temporal unfolding and local interaction expands the scope of developmental encoding research and suggests possible new directions for the field. Perhaps most significantly, if similar motifs can be produced with and without local interaction, the question arises whether local interaction is, at a high level, essential to efficient encoding, or whether it is rather only an artifact of the constraints imposed by physics in nature.

The next section provides background on the role of development in both natural evolution and artificial evolutionary algorithms. Section 3 then details the CPPN approach, and how CPPN-NEAT evolves increasingly complex patterns. Finally, Section 4 explains the experimental approach and discloses results, followed by a discussion in Section 5.

2 Background

This section introduces important concepts in developmental encoding, starting with a review of prior computational abstractions. Following this review, typical patterns produced by natural development are discussed, providing context for evaluating the validity of different abstractions. Finally, an important process in natural evolution called *complexification*, which also occurs in the experiments in this paper, is described.

2.1 Artificial Developmental Encodings

The apparent connection between development and complexity in biology has inspired significant research into artificial developmental encoding in recent years [4, 6, 10, 11, 12, 13, 22, 24, 25, 26, 27, 32, 33, 34, 35, 38, 41, 42, 43, 44, 47, 48, 54, 55, 57, 60, 67, 69, 80]. Just as a biological embryo starts from a single cell and through a series of genetic instructions achieves structures of astronomical complexity, artificial developmental encodings strive to map an artificial genome to a comparatively more complex phenotype through a series of steps that grow a small starting structure into a mature form.

Development makes such representational efficiency possible by allowing genes to be reused in the process of assembling the phenotype. Without such reuse, the genotype would need to be as complex as the phenotype, potentially including millions or even trillions of separate genes, an intractable search space. Through reuse the number of genes can be orders of magnitude fewer than the number of structural units in the phenotype, making search for extremely complex structures practical. For example, there are numerous receptive fields in the human visual cortex, each of which is encoded from the *same* genes. Thus, a small number of genes can encode for a large number of structures. As the brain develops during embryogenesis, each time a receptive field is built, a similar set of genes is activated [29, 36]. In this way, development and reuse are complementary partners in the representation of complexity. While factors such as the environment and its interaction with development affect the success of evolutionary search as well, the encoding's efficiency provides a significant advantage.

Artificial developmental encodings attempt to exploit the relationship between reuse and complexity in a similar way to nature. However, because computers differ from the natural world and because it is not necessary to simulate every facet of a biological process in order to capture its essential properties, finding the right level of abstraction remains an open problem. Therefore, several levels of abstraction have been explored in recent years. These range from low-level *cell chemistry* simulations to higher-level *grammatical* approaches.

Cell chemistry methods simulate the local interactions among chemicals and genes inside and between cells during embryogenesis [6, 8, 10, 11, 12, 13, 14, 22, 24, 25, 26, 27, 38, 41, 42, 44, 54, 55, 57, 80]. These methods follow the philosophy that the essential functions that allow development to yield significant complexity are located in the physical interactions that occur among proteins and cells in a developing embryo. Thus, cell chemistry approaches evolve genes that produce simulated proteins that diffuse and react as genotype maps to phenotype through a process of growth. The networks formed by genes that send signals back and forth through their protein products are called *gene regulatory networks* (GRNs). GRNs are often explicitly simulated in cell chemistry methods [12, 22, 24].

In contrast, grammatical approaches simulate development at a higher level of abstraction by evolving sets of rewrite (i.e. symbol replacement) rules [9, 32, 33, 34, 35, 43, 47, 48, 60, 67, 69]. When applied iteratively these rules grow a final structure from a single starting symbol. Each replacement occurs at the locality of the symbol to be replaced within the growing phenotype. Thus, local interaction and temporal unfolding also control the growth of phenotypes through grammars.

Both approaches have produced significant results, including simulated three-dimensional body morphologies and locomotion [13, 35], neural networks with repeating structures [32, 43], and physical structures with self-similarity [24, 35]. However, no approach has achieved complexity even close to that seen in nature. Thus, the search for powerful developmental encodings continues [74]. Expanding the scope of developmental encoding to include novel abstractions opens up new opportunities for such exploration.

The next section discusses important characteristics of developmental patterns.

2.2 Patterns of Development

Development produces patterns and evolution produces sequences of patterns over generations. Patterns include regularities, and regularities are what make reuse possible. Without regularity, the same information could not produce different parts of the same phenotype, removing much of the advantage of development. This section enumerates a set of general characteristics of patterns observed in natural organisms that can also be sought in artificially evolved phenotypes.

Identifying the general characteristics of natural patterns is an important prerequisite to describing how such patterns can be generated algorithmically. Both Turing [80] and Lindenmayer [47] were initially inspired by the patterns they observed in nature before they attempted to describe how those patterns could be generated. Turing eventually proposed his *reaction diffusion model*, which successfully produces patterns similar to those seen on sea shells and animal pelts, and Lindenmeyer invented L-systems, which grow accurate plant-like morphologies. Thus, by identifying the common general properties of patterns in nature, it becomes more clear for what phenomena artificial systems must account. The following list describes both patterns present in individual organisms, and the way those patterns change over generations.

- **Repetition**: Multiple instances of the same substructure is a hallmark of biological organisms. From cells throughout the body to neurons in the brain, the same motifs occur over and over again in a single organism. Repetition in the phenotype is also called *self-similarity* [10].

- **Repetition with Variation**: Frequently, motifs are repeated yet not entirely identical. Each vertebrae in the spine is similar, yet they each have slightly different proportions and morphologies [85]. Similarly, human fingers repeat a regular pattern, yet no two fingers on the same hand are identical. Repetition with variation is abundant throughout all of natural life.

- **Symmetry**: Often repetition occurs through symmetry, as when the left and right sides of a body are identical mirror images in classic bilateral symmetry.

- **Imperfect Symmetry**: While an overall symmetric theme is observable in many biological structures, they are nevertheless generally not perfectly symmetric. Such imperfect symmetry is a common feature of repitition with variation. The human body, while overall symmetric, is not equivalent on both sides; some organs appear only on one side and one hand is usually dominant over the other.
- **Elaborated Regularity**: Over many generations, regularities are often elaborated and exploited further [76]. For example, the bilaterally symmetric fins of early fish eventually became the arms and hands of mammals, displaying some of the same regularities. [63].
- **Preservation of Regularity**: Over generations, established regularities are often strictly preserved. Bilateral symmetry does not easily produce three-way symmetry, and four-limbed animals rarely produce offspring with a different number of limbs, even as the limb design itself is elaborated.

Using this list, phenotypes and lineages produced by artificial encodings can be analyzed according to their natural characteristics, giving an indication whether a particular encoding is capturing essential properties and capabilities of natural development. The characteristics in this section are employed to interpret experimental results reported in Section 4.

The next section describes a process through which the patterns represented by a set of genes can become increasingly complex.

2.3 Complexification

The process of complexification allows evolution to discover more complex phenotypes than would be possible through optimizing a fixed set of genes. This section explains complexification, which is demonstrated in the experiments in Section 4, and how it can be implemented.

In searching for the solution to a particular problem whose dimensionality is unknown *a priori*, the more dimensions there are in a selected solution space, the harder it is to discover. In other words, complex solutions are more difficult to evolve than simple ones. Developmental encodings attempt to reduce the effective complexity of the search space by encoding a complex phenotype in a significantly lower-dimensional genotype.

Yet even this reduction does not remove the curse of dimensionality. Even with a developmental mapping, it still takes many thousands of genes to represent genuinely complex phenotypes. For example, the human genome has 30,000 genes (3 billion base pairs), which is a significantly high-dimensional search space. Although the compression between genotype and phenotype is indeed substantial (encoding hundreds of trillions of phenotypic components), the search space of the *genotype* still remains prohibitively complex for direct search.

The reason evolution can overcome this prohibitive complexity is that it does not start searching in a space of the same complexity as the final solution. New genes are occasionally added to the the genome, allowing evolution to perform a complexifying function over and above optimization. Complexification allows evolution to begin with simple phenotypes by starting search in a low-dimensional space, and elaborate on them incrementally, as opposed to evolving elaborate systems from the start.

This process of gradually adding new genes has been confirmed in natural evolution [2, 18, 28, 50, 81] and shown to improve adaptation [1]. New genes commonly appear through *gene duplication*, which is a special kind of mutation in which one or more parental genes are copied into an offspring's genome more than once. Gene duplication has been responsible for key innovations in overall body morphology over the course of natural evolution [2, 15, 28, 50]. The main effect of gene duplication is to increase the dimensionality of the genotype, thereby allowing it to represent increasingly complex phenotypic patterns. Thus, complexification and developmental encoding work together to produce complex phenotypes.

In artificial evolution, genes can be added in many ways, including but not limited to duplication. They can also simply be added as new dimensions with random values or through a more domain-specific process based on properties of the genotype. Regardless of how they are added, the new space they afford builds on the information that is already present and allows a process of unbounded elaboration. Through complexification, major animal body plans can be established early in evolution and then elaborated and refined as new genes are added. Thus, it is important that any encoding abstracted from biology be able to complexify in addition to efficiently encode regularities. The encoding proposed in Section 3.4 satisfies these criteria.

The next section presents a novel perspective on developmental encoding.

3 Patterns Without Local Interaction

Much of evolutionary computation, and artificial intelligence as a whole, is concerned with finding the right level of abstraction. A strong assumption behind much recent work in developmental encoding is that local interaction and temporal unfolding in a simulation are abstractions that capture the essential properties of development in nature. Although this assumption is powerfully intuitive, this section shows how it is possible to abstract away even these seemingly fundamental features yet still capture the same essential

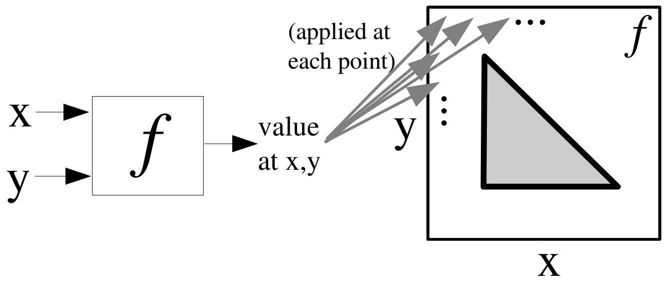


Figure 1: **A Function Produces a Phenotype.** The function f takes arguments x and y, which are coordinates in a two-dimensional space. When all the coordinates are drawn with an intensity corresponding to the output of f at that coordinate, the result is a pattern, which can be conceived as a phenotype whose genotype is f. In this example, f produces a triangular phenotype.

generative properties. The approach is motivated by the characteristics of patterns produced by natural evolution discussed in Section 2.2. The central insight is that local interaction and unfolding over time are prerequisites only to encoding in the physical universe. When the constraints of physics are lifted, they can be abstracted away.

3.1 Eliminating Temporal Unfolding

There is a compact *description* of the phenotype that does not involve explicit temporal unfolding. In order to see that such a description must exist, it helps to conceive the phenotype as a distribution of points in a multidimensional Cartesian space. Viewed this way, a phenotype can be described as a function of n dimensions, where n is the number of dimensions in the physical world. For each coordinate, the presence or absence of a point, or its level of expression, is an output of the function that describes the phenotype. Figure 1 shows how a two-dimensional phenotype can be generated by a function of two parameters.

This observation implies that any phenotype produced through a temporal progression is also possible to represent through a functional description. In fact, Cybenko [17] showed that any function can be approximated by a neural network with two hidden layers. The more neurons in the network, the more accurate the approximation can be. Thus, any morphology, when viewed as a distribution of particles in space, is possible to represent as a function without the notion of time. A developmental chronology is only *one* way of producing a particular constellation of particles.

It follows that encoding an unfolding process may be unnecessary to produce complex phenotypes because a functional description could alternatively have been evolved. Such a description would not necessitate a long unfolding chain of interactions and productions. Thus, at least theoretically, it is possible that the essential properties of developmental encoding can be captured by an encoding without the notion of time. The next section strengthens this perspective by showing how functional representations can include the same essential pattern-generating information as developmental encodings, and in similarly compact ways.

3.2 Functions and Regularities Without Local Interaction

The process of temporal unfolding is predicated on local interaction. Phenotypic development over time requires that local states interact to produce successor states at the same loci. Thus, if temporal unfolding is to be replaced with functional description, then that description must subsume the role of local interaction to remain a valid abstraction of development. This section shows how a class of functional descriptions can in fact mimic the effects of local interaction.

A significant function at early stages of development in natural embryogeny is to define a *coordinate frame*, i.e. a set of virtual coordinate axes, upon which future stages of development will be based [52, 63]. The simplest and most basic of these coordinates frames are the main axes of the body, which are defined at the very beginning of development, inside the egg itself [63, p.188]. These axes include the *anterior-posterior* axis (i.e. head to feet), and the *dorsal-ventral* axis (i.e. back and front). Just like the x and y axes in a simple Cartesian coordinate system, these two main axes are orthogonal and roughly straight. The initial axes are determined through a cascade of local gene interactions defined by a GRN, that is, a signaling pattern that develops over time culminates in a chemical gradient along each initial axis.

Once the initial axes are defined, new coordinate frames for subregions of the body can then be appropriately spatially situated [52]. In this way, the gradients that define earlier coordinate frames are *inputs* to future cascades that produce more localized coordinate frames. For example, the body plan of the *Drosophila* fly includes a series of segments, each with its own internal coordinate axes. Each segment is the origin of a specific body part or region, such as legs, wings, and the head. (Each of these regions in turn have their own subregions with their own coordinate frames as well.) Importantly, the particular coordinate frame of each body segment must be determined by where that segment lies along the anterior-posterior axis of the complete body. In fact, the genes that determine the segmentation of the body by defining their coordinate frames (called *HOX genes*) are activated from the initial axial gradients [16, 46]. Figure 2 shows the anterior-posterior axis of a simplified insect model, and the segments along its axis.

Meinhardt [52, p.81] gives a succinct summary of the phenomenon of layering coordinate frames (which he calls *subfields*) one upon another:

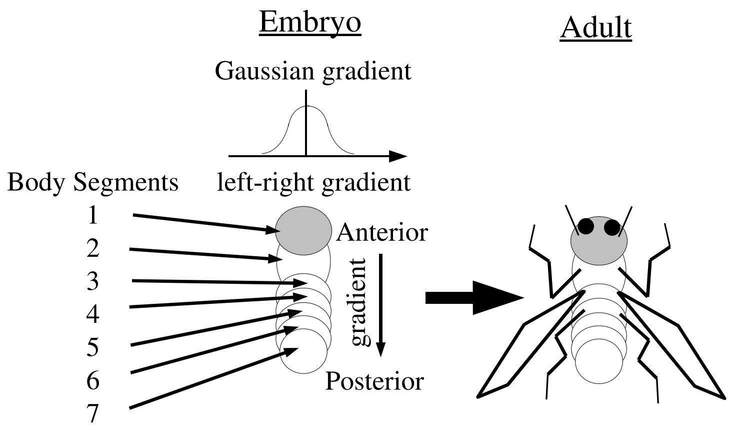


Figure 2: **Gradients Define the Body Plan.** A simplified insect model with discrete body segments and bilateral symmetry is depicted. Gradients along the major axes define the the overall body plan. The anterior-posterior gradient allows segments to situate in the embryo along that axis, each one destined to grow its own associated body parts such as wings or legs. The Gaussian gradient, a function of the original left-right axis gradient, is distributed symmetrically about that axis, allowing bilateral symmetry to develop from it.

> It will be shown that many of these experiments are explicable under the assumption that the boundaries between patches of differently determined cells, determined under the influence of the primary gradient(s), become the organizing regions for the developmental control of subfields. Since the boundaries of existing structures give rise to the new structures, the existing and the new structures have necessarily the correct spatial relationship to each other. This allows a very reliable finer subdivision of a developing embryo.

Interestingly, although the coordinate frames in natural development are defined through local GRNs that unfold over time, in principle the same axes can be described functionally instead, simply by composing the functions that describe them. Through function composition, it is simple to demonstrate that biologically plausible new coordinate frames are derivable from preexisting ones without a notion of time. Consider the *left-right* axis, along which bilateral symmetry is present in many organisms. Coordinates along this axis can be defined simply as $f(x) = x$. However, in order to become bilaterally symmetric along this axis, a developing embryo must organize a gradient that peaks at the center and diminishes towards both the left and right sides. The organizing process to achieve such a gradient may be complex and involve many local interactions, yet functionally it can be described simply as a Gaussian: $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$ (figure 2). In fact, a similar effect could be achieved even more simply through the absolute value function, assuming the midline is $x = 0$: $h(x) = |x|$. It is important to note also that a bilaterally-symmetric coordinate frame does not preclude simultaneous bilateral *asymmetries*; the original left-right gradient is no less present.

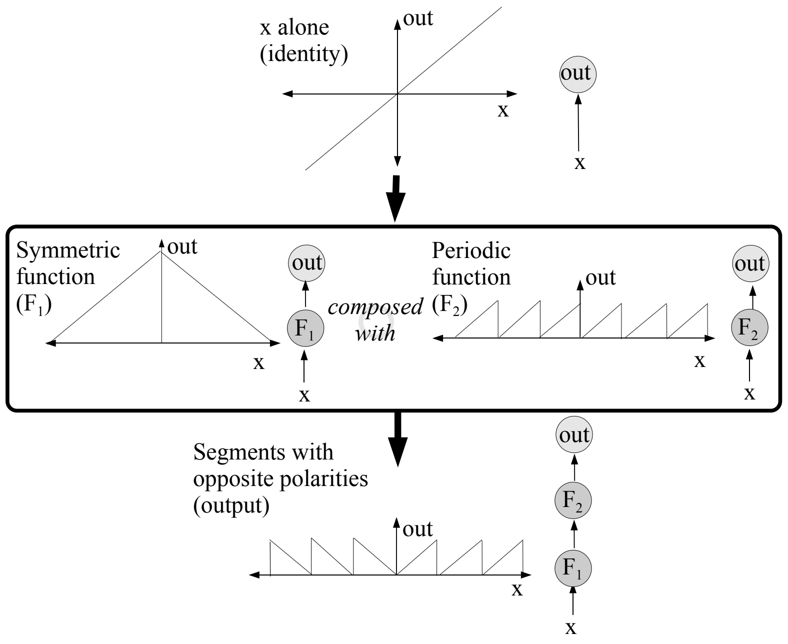


Figure 3: **Composing Gradient Functions.** This example illustrates how a simple composition of functions in one dimension can produce a pattern with multiple regularities. A network representation of each composition is depicted to the right of each function graph. The initial asymmetric gradient x is composed with both a symmetric function (F_1) and a periodic function (F_2), resulting in two sets of segments with opposite polarity. Thus, new coordinate frames can be built upon preexisting ones without local interaction.

Just as a symmetric function can yield bilateral symmetry, segmentation can be represented through periodic functions. The function $j(y) = \sin(y)$ effectively assigns a repeating coordinate frame (i.e. a set of segments) laid along the anterior-posterior axis. In the case of sine, each coordinate frame is itself symmetric about its peak value. Repeating asymmetric frames can be created analogously using the modulus function as in $k(y) = y \bmod 1$.

Coordinate frames created through a developmental process *interact* with each other to produce complex patterns with regularities. In the same way, functionally-represented frames can be *composed* to create complex regularities. In this way, composition replaces interaction. For example, bilateral symmetry and segmentation along the left-right axis can produce two sets of segments with opposite polarities. This phenomenon is easily achieved by feeding the output of a symmetric function into a periodic function (figure 3). This simple composition creates a new set of coordinate frames that can now be used for further refinement, all without explicit local interaction.

The preceding argument establishes that fundamental interactions over development are in fact analogous to a series of function compositions that each in turn create a new, more refined coordinate frame. Eventually, these frames can combine to form patterns much like those observed in nature. This analogy raises the intriguing possibility that function composition is a valid abstraction for development, even though

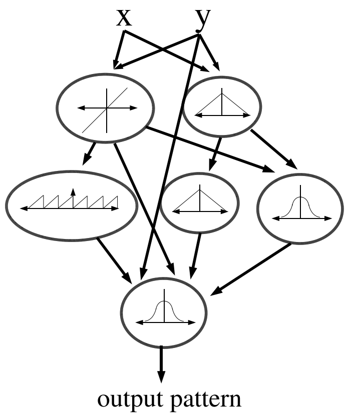


Figure 4: **Composition of Functions as a Graph.** The graph determines which functions connect to which. The connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection. If multiple connections feed into the same function, it means that the downstream function takes the sum of their weighted outputs. Note that the topology is unconstrained and can represent any possible relationships. This representation is similar to the formalism of artificial neural networks with arbitrary activation functions and topologies. Because the absolute coordinate frame (x, y) is input to the network, local interaction can be eliminated from the representation.

function composition is not intrinsically an unfolding process. The main idea is that the *order* in which functions are composed is an abstraction for the chronology of events over the course of development, without the need for simulating such events locally.

A natural way to represent a composition of many functions is as a connected graph (figure 4). The initial coordinate axes (i.e. the most basic coordinate frame) can be provided as inputs to the graph. The next level of nodes can be thought of as descriptions of the first stages of development, such as establishing bilateral symmetry. Higher level nodes then establish increasingly refined coordinate frames. The final outputs are thus informed by each transformation that takes place before them. In this way, the entire graph is like a diagram of the sequence of steps that happen over a developmental chronology.

Providing the initial coordinate axes as inputs to the graph is what allows local interaction to be eliminated: In physical space there are no intrinsic coordinates that an individual cell can access to determine its location (and hence its identity). Therefore, local interaction becomes a way of asking, “where am I?” That is, through the collective negotiation of adjacent cells that interact with each other, it is possible to derive a coordinate frame. However, by composing functions that take as arguments an absolute frame of reference, the need for such negotiation is eliminated and all identities and relative locations can be determined completely independently.

Interestingly, a graph of such compositions is very similar to an artificial neural network with arbitrary topology. The only difference between the two is that artificial neural networks generally use sigmoid functions (and sometimes Gaussian functions) as activation functions in each node, whereas the function composition graph may use any of a variety of canonical functions at each node.

The analogy between a function composition graph and an artificial neural network (ANN) is so strong, in fact, that it is tempting to equate the two. However, while from an external objective standpoint they are clearly related, using the term *artificial neural network* would be misleading in the context of this discussion because artificial neural networks were so named in order to establish a metaphor with a *different* biological phenomenon, i.e. the brain. The terminology should avoid making the implication that biological, thinking brains are in effect the same as developing embryos. Therefore, this paper uses the term *Compositional Pattern Producing Network* (CPPN) to refer to graph constructs that describe compositions of functions intended to produce regular patterns.[1]

By querying a CPPN separately for each point in the Cartesian coordinate space *independently*, it can describe an entire phenotype without local interaction or temporal unfolding. The coordinates of each point are input into the CPPN and the output of the CPPN designates what should exist at that point in space, as depicted in figure 1.

The next section explains how CPPNs can be evolved in a way that maintains the analogy with development: Major body-plan features can be established first and then refined in future generations, just as happens over the course of natural evolution.

3.3 Evolving Compositional Pattern Producing Networks

It is fortuitous that CPPNs and ANNs are virtually the same from a structural perspective because many methods already exist for evolving ANN topology and weights [5, 32, 58, 61, 73, 83, 84]. These methods can be easily extended to CPPNs with little change. There are also methods for evolving genetic programs, which can be similar to CPPNs [39, 45].

Given so many choices, what is the best method for evolving a CPPN? Evolving CPPNs should maintain the essential properties of evolution in nature. One of these properties, described in Section 2.3, is the gradual complexification of the genome: Evolution did not begin search in the space of human beings. Rather, there was initially a smaller genome and new genes were added over the course of evolution. In a CPPN, adding a new gene means adding a new connection or function node to the network. Such an

[1]Potential confusion notwithstanding, the relationship between development and ANNs is not accidental; rather, it shows that the same kinds of abstractions underly diverse complex phenomena.

addition is analogous to adding a new gene in a GRN. Thus, if CPPNs are allowed to complexify in this way, then they too can establish fundamental regularities early in evolution and then elaborate on them over generations.

The NeuroEvolution of Augmenting Topologies method [NEAT; 73, 75] was designed with this process in mind. NEAT evolves increasingly complex ANNs over generations, and addresses the challenges that come with evolving a population of diverse complexifying topologies by using historical markings. Because NEAT starts with simple networks and expands the search space only when beneficial, it is able to find significantly more complex ANNs than fixed-topology evolution [75].

Although NEAT was originally designed to evolve ANNs, it requires only trivial modification to evolve CPPNs, since they are so similar. The next section describes CPPN-NEAT.

3.4 CPPN-NEAT

This section explains how NEAT can evolve CPPNs. See Stanley et al. [72], Stanley and Miikkulainen [73, 75] for detailed descriptions of original NEAT.

The NEAT method was originally developed to solve difficult control and sequential decision tasks. The neural networks control agents that select actions based on their sensory inputs. While previous methods that evolved neural networks, i.e. *neuroevolution* methods, evolved either fixed-topology networks [30, 66], or arbitrary random-topology networks [5, 32, 82], NEAT is the first to begin evolution with a population of small, simple networks and *complexify* the network topology over generations, leading to increasingly sophisticated behavior.

Before describing the CPPN extension, let us review the three key ideas on which the basic NEAT method is based. First, in order to allow network structures to increase in complexity over generations, a method is needed to keep track of which gene is which. Otherwise, it is not clear in later generations which individual is compatible with which, or how their genes should be combined to produce offspring. NEAT solves this problem by assigning a unique *historical marking* to every new piece of network structure that appears through a structural mutation. The historical marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited during crossover unchanged, and allow NEAT to perform crossover without the need for expensive topological analysis. That way, genomes of different organizations and sizes stay compatible throughout evolution, solving the previously open problem of matching different topologies [62] in an evolving population.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before competing with other niches in the population. NEAT uses the historical markings on genes to determine to which species different individuals belong.

Third, other systems that evolve network topologies and weights begin evolution with a population of random topologies [32, 82]. In contrast, NEAT begins with a uniform population of simple networks with no hidden nodes, differing only in their initial random weights. Diverse topologies gradually accumulate over evolution, thus allowing diverse and complex phenotype patterns to be represented. No limit is placed on the size to which topologies can grow. Thus, NEAT can start minimally, and grow the necessary structure over generations.

New structures are introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. In this way, another advantage of complexification is that NEAT searches through a minimal number of weight dimensions, significantly reducing the number of generations necessary to find a solution, and ensuring that networks become no more complex than necessary. In effect, then, NEAT searches for a compact, appropriate topology by incrementally complexifying existing structure.

CPPN-NEAT is an extension of NEAT that allows it to evolve CPPNs. While networks in original NEAT only include hidden nodes with sigmoid functions, node genes in CPPN-NEAT include a field for specifying the activation function. When a new node is created, it is assigned a random activation function from a canonical set (e.g. including Gaussian, sigmoid, and periodic functions). Furthermore, the compatibility distance function that is utilized to determine whether two networks belong in the same species includes an additional argument that counts how many activation functions differ between the two individuals. This feature allows speciation to take activation function differences into account. Because CPPN-NEAT is an enhancement of an effective preexisting method, it provides a reliable platform for studying the evolution of increasingly complex forms. The next section discusses the computational complexity of CPPN representation.

3.5 CPPN Computational Complexity

The CPPN must fully activate *once* for each coordinate in the phenotype, making its complexity $O(N)$, where N is the number of coordinates. Thus, for an $n \times n$ two-dimensional grid, the CPPN is activated $O(n^2)$ times. Unlike other developmental abstractions, CPPNs can produce the same phenotype at any resolution because increasing resolution only requires increasing the granularity of coordinates. A CPPN is in effect an infinite-resolution mathematical structure. Therefore, because the phenotype resolution is not an intrinsic property of the CPPN, computational complexity can be minimized for a particular task by only activating at the minimal-necessary resolution. Furthermore, because the CPPN is activated independently for each coordinate in the phenotype, the complete computation of every coordinate is trivial to parallelize.

Because they are both developmental abstractions, it is informative to compare CPPNs with GRNs. The computational expense in a GRN results from every active cell reacting to all incoming signals on every timestep. For a cell to react to incoming chemicals, it must process them with its internal GRN, which necessitates significant computation. In effect, such a procedure must activate on the order of every cell for every developmental step, resulting in $O(Ct)$ complete GRN activations, where C is the maximum number of cells and t is the number of timesteps. If the phenotype is roughly a $c \times c$ grid of cells, then its time complexity for complete development is $O(c^2t)$. Furthermore, GRNs must compute cell-cell signaling, and sometimes diffusion and reaction over long distances, adding to the computational burden. In fact, the need to simulate an entire interacting physical system of possibly thousands or more cells over hundreds or thousands of developmental steps is a significant long-term challenge for the field of developmental encoding. Thus, CPPNs may save time on some problems by cutting out t.

It is important to note however that the number of individual cells or structural components may differ for CPPNs versus GRNs on the same task. Therefore $O(n^2)$ and $O(c^2t)$ should not be compared directly. However, the complexity bounds do give a rough sense of how the different methods contrast that may be helpful in deciding which abstraction to choose for a particular task in the future.

The next section demonstrates through several experiments that CPPN-NEAT produces phenotypes that exhibit key natural characteristics.

4 Experiments

Developmental encodings are traditionally evaluated for their performance on benchmark tasks [9, 10, 12, 13, 22, 24, 25, 26, 32, 33, 34, 35, 43, 44, 54, 65]. These tasks range from body-brain evolution [13, 22, 34, 44], to target object morphologies [10, 24, 33], to target patterns such as flags [25, 26, 54]. It is also common to compare developmental encodings to direct encodings, usually demonstrating that reuse can be advantageous when the target is sufficiently regular and complex to make direct encoding prohibitive [10, 33, 34, 64, 65].

For three reasons, this paper takes a significantly different experimental approach. First, the primary aim of this paper is to demonstrate that CPPNs abstract the essential properties of natural development even without local interaction or unfolding over time. Because removing these features significantly distances CPPNs from more traditional abstractions, the aim in this section to establish that CPPNs deserve to be categorized in the same class. Other developmental encodings generally need not establish this link because their relationship to natural development is more explicit.

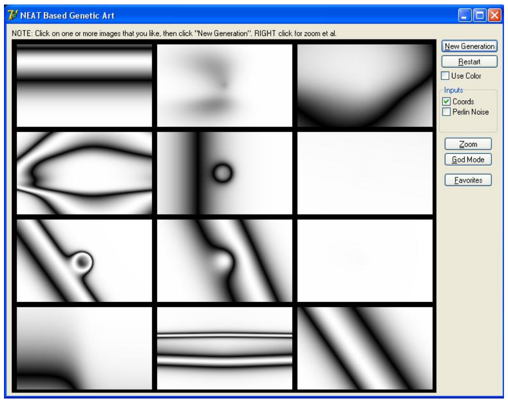
Second, natural evolution does not evolve to specified targets. Rather, it exploits established regularities in creative ways. Thus, evaluating against a specific target or task does not necessarily reveal CPPNs' ability to similarly establish and then exploit regularities. In the future, it will be informative to measure CPPN performance in a range of domains; however, first it is important to establish that the *way* evolving CPPNs solve tasks indeed matches with the important mechanisms of natural evolution and development. The major contribution of this paper is thus to establish that a new kind of abstraction of development is admissible.

Third, natural organisms do not appear to be able to take on any arbitrary form. For example, no macroscopic organism has any appendage analogous to a wheel, even though a wheel is a fundamental engineering structure. Thus, if DNA was tasked with representing a car, it would likely fail. Nevertheless, such failure would not be an indictment against its potential for encoding unparalleled complexity. Therefore, rather than focus on whether a CPPN can represent a certain *class* of object, the following experiments focus on its ability to establish and exploit regularities in general.

4.1 CPPN-NEAT-based Interactive Pattern Evolution

To explore both the patterns that CPPNs produce and the way evolution varies those patterns, an interactive evolutionary approach was taken. Interactive evolution [59, 77] allows a user to perform the selection step that determines which individuals will be parents of the next generation. By introducing this interactive role for the experimenter, it is possible to intentionally explore the space of generated patterns in ways that would otherwise be impossible.

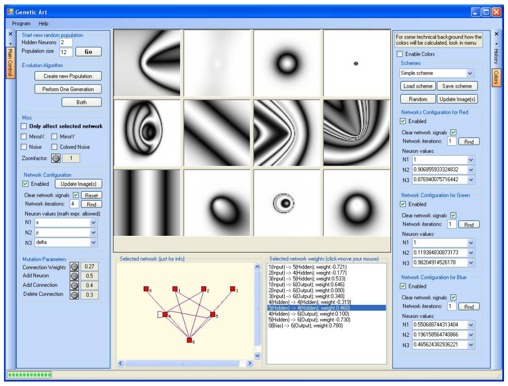
The interactive approach works by visually presenting the experimenter with all the patterns in the current generation on the screen at once (figure 5 introduces the two programs used in the experiments). The experimenter then picks the parents of the next generation by clicking on them. Thus, the selection step is entirely determined by the human experimenter. The aim is to intentionally search for certain types of patterns in order to ascertain whether they can be elaborated and maintained in the same ways as in natural evolution.



NOTE: Click on one or more images that you like, then click "New Generation". RIGHT click for zoom et al.

Use Color [ ]
Coords [✓]
Perlin Noise [ ]

(a) DelphiNEAT-based Genetic Art (DNGA)



Hidden Neurons: 2
Population size: 12

Only affect selected network [ ]
MirrorX [ ]
MirrorY [ ]
Noise [ ]
Colored Noise [ ]
Zoomfactor: 1

Network Configuration
Enabled [✓]
Clear network signals [✓]
Network iterations: 4
Neuron values (math expr. allowed):
N1: x
N2: y
N3: delta

Mutation Parameters
Connection Weights: 0.27
Add Neuron: 0.5
Add Connection: 0.4
Delete Connection: 0.3

Selected network weights (click+move your mouse)

```
1(Input) --> 5(Hidden); weight:-0.721)
2(Input) --> 4(Hidden); weight:-0.177)
3(Input) --> 5(Hidden); weight:0.533)
1(Input) --> 6(Output); weight:0.646)
2(Input) --> 6(Output); weight:0.000)
3(Input) --> 6(Output); weight:0.348)
4(Hidden) --> 4(Hidden); weight:-0.319)
5(Hidden) --> 4(Hidden); weight:0.460)
4(Hidden) --> 6(Output); weight:0.100)
5(Hidden) --> 6(Output); weight:-0.730)
0(Bias) --> 6(Output); weight:0.780)
```

For some technical background how the colors will be calculated, look in menu
Enable Colors [ ]
Schemes: Simple.scheme

Networks Configuration for Red
Enabled [✓]
Clear network signals [✓]
Network iterations: 1
Neuron values:
N1: 1
N2: 0.906855933324832
N3: 0.876940075716442

Network Configuration for Green
Enabled [✓]
Clear network signals [✓]
Network iterations: 1
Neuron values:
N1: 1
N2: 0.119384830873173
N3: 0.98204914526178

Network Configuration for Blue
Enabled [✓]
Clear network signals [✓]
Network iterations: 1
Neuron values:
N1: 0.550688744313404
N2: 0.196158564740866
N3: 0.465624382936221

(b) SharpNEAT-based Genetic Art (SNGA)

Figure 5: **Interactive Evolution Experimental Interfaces.** The two platforms shown above allow an experimenter to evolve CPPN-generated patterns interactively. The set of patterns shown are the current population, and the experimenter can select the parents of the next generation (i.e. fitnesses are not assigned). Both platforms were used in the experiments in this section. Random initial generations are shown for each. From initial populations such as these, all of the regular and complex forms that follow were evolved. (a) DNGA is by Mattias Fagerlund, and uses his underlying DelphiNEAT platform to evolve CPPNs. (b) SNGA, which uses Colin Green's SharpNEAT platform, is by Holger Ferstl. Both platforms are freely available, including source code (see Acknowledgements). The interactive interface makes it possible to explore the way evolution establishes and varies regularities, as opposed to attempting to solve a particular benchmark.

This approach resembles what are commonly called *genetic art* programs, which are traditionally used to generate aesthetically pleasing abstract pictures that would otherwise be unlikely to be conceived by a human [7, 20, 31, 49, 51, 68, 70, 78]. Genetic art programs closely follow the original *Blind Watchmaker* idea from Dawkins [19], in which simple genetically-encoded patterns are evolved through an interactive interface.

Interactive evolution is a novel approach to exploring the properties of a developmental encoding and therefore requires justification. While the interactive genetic art concept is often considered an entertainment application or artistic tool, it is also a rich and underutilized experimental medium for exploring the characteristics of various encodings, as this section will show. Using genetic art, an experimenter can volitionally explore implicit capabilities such as elaboration and preservation of regularity that target-based or benchmark-based evolution cannot easily reward explicitly. For example, in target-based evolution, even if bilateral symmetry is integral to the final design, it may not be preserved from one generation to the next when an early bilaterally-symmetric candidate is far from the target. In contrast, the value of symmetry would be immediately apparent to a human experimenter, who can thus select for it explicitly. Because just such regular properties define the capabilities of an encoding, interactive evolution makes a powerful experimental platform.

Introducing interactive evolution as an experimental tool in developmental encoding raises the inevitable question whether past genetic art systems are in effect experiments in developmental encoding as well. In such systems, it is important to distinguish the process of interactive evolution from the particular encoding being explored. In principle, interactive evolution can be used to explore the properties of *any* encoding, including those with no relationship to development. In general, the encodings used in genetic art systems are not motivated by or based upon natural development. Rather, they are often designed to include intuitive aesthetic features such as shape descriptors [56]. Nevertheless, many such systems do in effect implement what can be described in retrospect as a rough abstraction of development.

In fact some features of CPPNs appear as early as Sims [68], who interactively evolves images encoded as symbolic LISP expressions. These expressions can include any combination of numerous operations including periodic and symmetric functions in addition to noise generators, image processing functions, and others. These expressions produce complex, aesthetically pleasing results.

On the other hand, other approaches to genetic art such as MUTATOR [78, 79] employ encodings more reminiscent of traditional developmental encodings such as L-systems, which include a period of iterative growth that repeatedly places the same primitives in different locations and with different orientations and scales.

In general, the distinctions among different genetic art approaches with respect to developmental encoding are often ill-defined because they were not designed to address the same considerations. Husbands et al. [37] describe a system for interactively evolving three-dimensional shapes that includes both assembling primitives and transform functions such as scaling and twisting that are combined like CPPN function compositions. Nishino et al. [56] introduces a similar system that combines deformable three-dimensional primitives, which effectively encodes visual artifacts.

Thus, overall, individual features of CPPNs have appeared in disparate prior genetic art systems, mainly as a means of convenient deformation. However, before it is pointed out that establishing coordinate frames through local interactions in development can be abstracted as composing canonical functions, it is not surprising that many such systems may go unrecognized by the developmental encoding community. Rather than competing with existing genetic art approaches, CPPNs aim to explicitly exploit the abstraction with development. This paper does not assert that CPPNs are an entirely novel mathematical formalism; function composition is a highly general procedure employed across the disciplines of science. Rather, the main contribution is to highlight the relationship between CPPNs and development, a crucial link that heretofore has not been established. Genetic art in this section acts only as a vehicle for making this connection. From

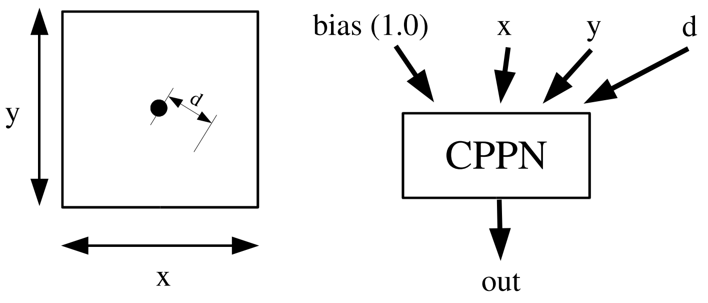


Figure 6: **CPPN Inputs.** Three values, the coordinate on the horizontal axis (x), the coordinate on the vertical axis (y), and the distance of the current coordinate from the center (d) are input into the CPPNs in the experiments in Sections 4.2 and 4.3. Inputting d provides a bias towards symmetry. However, since d is radially symmetric, it does not automatically provide a bilaterally-symmetric coordinate frame.

this novel perspective, it will be instructive in the future to explore many encodings, including both existing genetic art encodings and traditional developmental encodings, in a similar manner.

In this paper, the interactive evolution procedure is used to analyze CPPN-NEAT. Both DNGA and SNGA begin with a population of randomly initialized networks with one or two hidden nodes. The choice to start this way (as opposed to with no hidden nodes) was made so that that patterns in the initial generation would be nontrivial. The user picks one or two parents of the next generation, which are then displayed. The networks utilized by both programs are CPPNs that can represent arbitrary compositions of Gaussian and sigmoid functions. Furthermore, special coordinate frames are provided at the inputs in addition to x and y. DNGA (figure 5a) provides the CPPN with a *distance from the center* coordinate (d), while SNGA (figure 5b) allows the experimenter to provide any arbitrary coordinates frames, including sine waves of varying period. Thus, these programs allow the user to both explore and observe explicit examples of how CPPNs compose functions and derive patterns from provided coordinate frames. Figure 6 depicts the coordinates input into the networks.

Each new generation is created by mutating and mating the chosen parents. Thus, in these experiments, because the user performs selection, speciation (Section 3.4) is not utilized.[2] However, complexification still occurs, with mutations occasionally adding more functions and connections in the CPPN. In SNGA, the probability of adding a neuron to a child genome is 0.5 and the probability of adding a connection is

[2]Even though speciation is not used in these experiments, it is still a crucial component of CPPN-NEAT for any experiment that is not interactive. Speciation in original NEAT has been shown to protect innovative topologies long enough to reach their potential [73]. Since CPPN-NEAT will be used in the future to evolve complex phenotypes without user interaction, speciation is still important and therefore described in Section 3.4.

0.4. In DNGA, the probabilities are 0.06 for both. Extensive experimentation verified that both systems consistently evolve complex regular phenotypes, indicating that interactive CPPN-NEAT is robust to a wide range of these parameters. In both systems, function outputs range between $[-1, 1]$. However, ink level is darker the closer the output is to zero. Therefore, an output of either -1 or 1 produces white. This approach was found to produce the most interesting patterns in early generations.

The following experiments evaluate evolved patterns against the characteristics of natural biological patterns (Section 2.2).

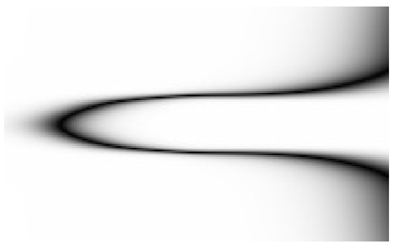
4.2 Elaboration on Pure Symmetry

Hundreds of millions of years ago during a period of evolution known as the *Cambrian Explosion*, when numerous new animal body plans were appearing, the *bilateria* appeared. The simplest of the bilateria were a kind of flatworm [63, pp.38-41]. This original body plan has been elaborated for hundreds of millions of years, all the while preserving its fundamental design. Can a CPPN analogously capture a bilateral body plan and maintain and elaborate on bilateral symmetry for generations?

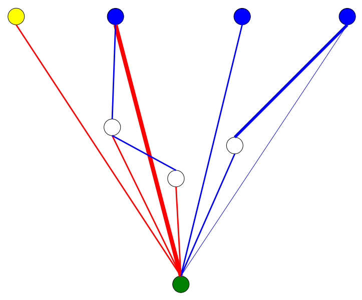
To address this question, a spaceship morphology was interactively evolved with DNGA. The first step was to determine whether the CPPN can discover and establish bilateral symmetry. As figure 5 shows, symmetry is by no means ubiquitous in random initial populations. Many individuals lack recognizable symmetry completely. However, in 20 separate runs, bilateral symmetry arises within the first three generations 100% of the time. Evolution quickly finds a CPPN structure that defines symmetry across a single axis, thereby creating a framework for bilateral symmetry. Figure 7a shows a typical early bilaterally-symmetric pattern.

The CPPN that generates this pattern has twelve connections, two hidden sigmoid functions, and one hidden Gaussian function (figure 7b). Interestingly, only one of the twelve connections can perturb the symmetry of the pattern. Therefore, there is only a one-in-twelve chance that any single gene mutation breaks symmetry.

The first individual with a recognizable spaceship-like pattern is shown in figure 8a. It has seventeen connections and four hidden functions that together produce a tail, wings, and a nose. These features will persist throughout evolution. A succession of descendants that elaborate on this initial theme is shown in figure 8. What is significant about these elaborations is how they genuinely preserve the underlying regularities but also become more complex.

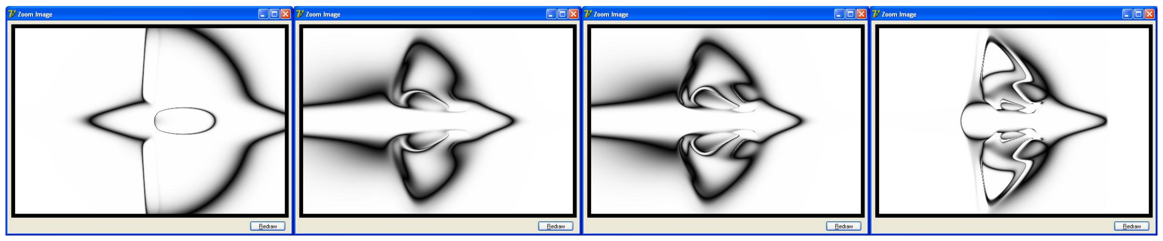


(a) First Bilateral Symmetry

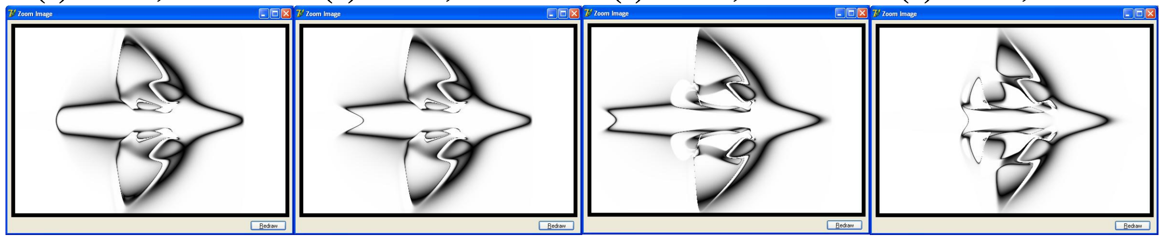


(b) CPPN

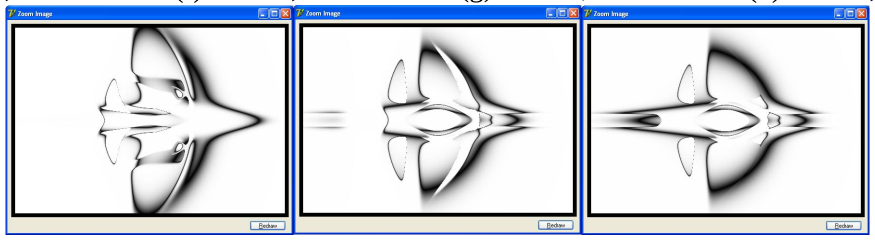
Figure 7: **Initial Discovery of Bilateral Symmetry.** This individual appeared in the third generation. (a) Its pattern forms a nose-like structure with symmetry around the horizontal midline. (b) Its CPPN contains three hidden node functions and twelve connections. The inputs (at top) start with the bias, followed by x, y, and d. The single ink-level output is at the bottom. Connection thickness is proportional to strength. The symmetry property is strongly ingrained into the genotype: Only one of its connections can cause it to lose symmetry.



(a) 4 func., 17 conn. (b) 5 func., 24 conn. (c) 6 func., 25 conn. (d) 8 func., 28 conn.



(e) 8 func., 30 conn. (f) 8 func., 31 conn. (g) 8 func., 32 conn. (h) 8 func., 34 conn.



(i) 8 func., 36 conn. (j) 9 func., 36 conn. (k) 9 func., 38 conn.

Figure 8: **Sequence of Descendant Spaceship Patterns.** The chronological sequence (a)–(k) displays successive progeny evolved with interactive CPPN-NEAT. The number of hidden functions and connections in the generating CPPN is shown below each pattern. The sequence exhibits a continual elaboration of regular form in part analogous to natural elaboration.

Figure 8h shows a remarkable and surprising subsequent variation on the tail in which evolution invented tail fins all at once in a single generation. The tail fins respect the underlying symmetry and at the same time introduce entirely new and elegant features. This kind of elaboration suggests that the complexifying CPPNs in effect implement the fundamental process of continual elaboration of form. This level of meaningful (from a human perspective) elaboration is a significant goal for artificial developmental encodings. Figure 8i displays a descendant of the original tail-fin pattern. The descendant significantly varies the wing regularity yet still preserves its integrity. The presence of wings has become in effect a “module” of the phenotype.

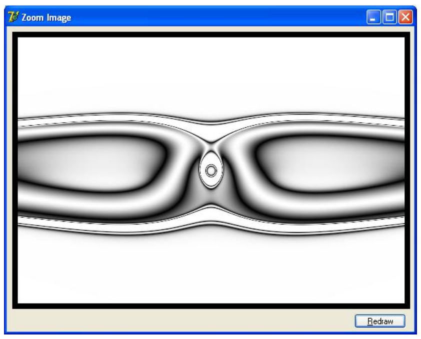
Evolution later creates a “cockpit” elaboration (figure 8j). This variation is accompanied by in an overall change in style. The wings are more rounded and the tail fins less ornate. Yet the main features still persist, showing the powerful capacity of the representation to facilitate meaningful change by capturing the regularities as functional relationships.

Whereas the initial spaceship CPPN contains four hidden functions and 16 connections (figure 8a), the final, significantly more complex spaceship CPPN includes nine hidden functions and 38 connections (figure 8k). Thus, the size of the genotype slightly more than doubled due to CPPN-NEAT’s complexification. The added structure allows CPPN-NEAT to lay new features on top of established coordinate frames. At the same time, it is notable that only 38 connections code for the complex pattern shown in figure 8k, indicating that CPPNs have the potential to reuse information and represent patterns in a way that enhances representational efficiency in analogy with natural development.

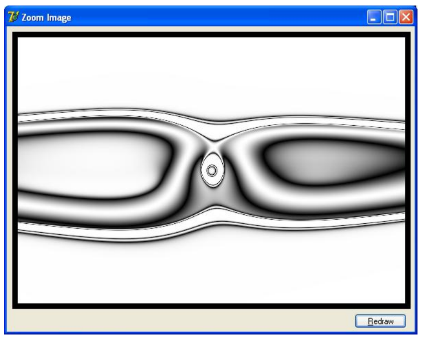
The succession of spaceship phenotypes in this section exhibits several of the fundamental characteristics of developmental encoding in nature. Symmetry is established early and then elaborated. Regularity is preserved and expanded. Mutations respect the underlying body plan as it elaborates through complexification. The next section addresses whether CPPNs can also create regularity within imperfect symmetry.

4.3 Representing Imperfect Symmetry

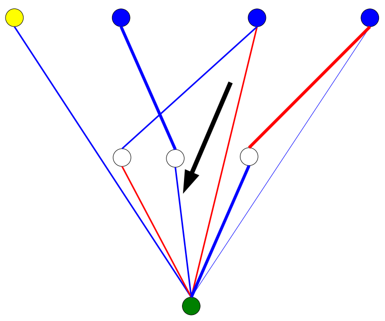
Symmetry is frequently discussed in the developmental encoding literature [10, 33, 64]. However, symmetry with variation receives less attention even though it is perhaps more essential and fundamental to the patterns of nature than perfect symmetry. The placement of organs like the heart, subtle differences between cerebral hemispheres, and even externally observable asymmetry, such as the lobster’s claw, are the norm rather than the exception. Perhaps most remarkable is that patterns can acquire asymmetries over generations while still preserving their overriding symmetrical regularities. This capability implies that encoding in nature is richly amorphous rather than strictly hierarchical or modular. Established regularities can be warped and partially



(a) Symmetric Sunglasses



(b) Warped Symmetry



(c) CPPN

Figure 9: **Varying Symmetry.** A single connection gene controls the level of symmetry in the pattern. (a) The original pattern displays perfect bilateral symmetry. (b) By manipulating a single gene, the symmetry can be varied, yet without breaking the overall regularity of the structure. (c) An arrow points to the single gene responsible for asymmetry. Since overall regularity is preserved, the representation of the sunglasses by the CPPN cannot be strictly hierarchical.

broken without breaking the entire mechanism. A good abstraction of development should exhibit the same capability.

Figure 9a shows a pair of bilaterally symmetric "sunglasses." The bilateral symmetry is an example of genetic reuse because the CPPN functions that describe the left shade are the same as the ones that describe the right. However, varying the weight of a single connection gene (figure 9c) actually varies the *asymmetry* of the shades, even while maintaining the overall regularity (figure 9b) of the pattern. In fact, the same genetic information is *still* generating both sides of the pattern, but one side is a variation of the other. Evolution is able to elaborate selectively without breaking the overall pattern.

Such elaboration is possible because a coordinate frame in a CPPN can be varied without destroying the pattern that is generated on top of it. Thus, the overall symmetry of the underlying coordinates can be broken through a single gene, but the complex pattern drawn within those coordinates still exists uncorrupted. This phenomenon is an abstraction of two gradients formed through local interaction within a natural embryo influencing the fate of cells at different positions. Figure 9 demonstrates that the abstraction is valid.

Figure 10 shows a similar phenomenon in an "eye" pattern evolved interactively with SNGA in which the eyeball can be rotated to one side or another by varying a single gene. The modular elements that compose the image survive even as their underlying coordinate system warps.

These examples demonstrate CPPNs' ability to encode imperfect symmetry.

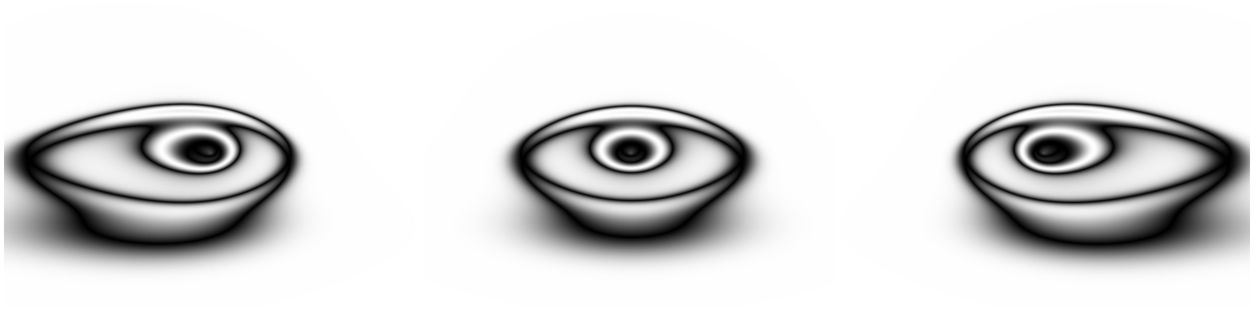

(a) Eye warped left (b) Symmetric eye (c) Eye warped right

Figure 10: **Eye Asymmetries.** The eye-generating CPPN can produce all three eyes above by varying only one connection weight. Interestingly, the eye turns as if in a three-dimensional space, showing that the composition of functions can result in complex new coordinate frames, sometimes having meaningful real-world interpretations, such as rotational variance in a three-dimensional space.

4.4 Repetition with Variation

Reuse through repetition with variation is crucial in natural organisms. Fingers and toes clearly share fundamental regularities yet also differ in size and proportion. Cortical columns in the brain also share structure yet are not identical [85]. If every slight variation on a theme required an entirely different set of genes, the representational space of many organisms would be prohibitively complex. Can a CPPN capture this important kind of regularity?

To answer this question, instead of inputting the x,y, and d coordinates into the CPPN in SNGA as normal, $\sin(10x)$, $\sin(10y)$, and d were input instead (figure 11). By inputting sine waves, the same coordinate frame repeats over and over again. However, the d coordinate, i.e. *distance from center*, does not repeat. Therefore, functions higher in the CPPN can utilize both the repeating frame of reference and the absolute reference *at the same time*. If indeed CPPNs can use this information to produce complex regularities with variation, then it is likely that CPPNs can evolve compositions of such functions to produce such patterns when necessary.

Figure 12a shows a repeating pattern generated by the CPPN with sine inputs. The period of the sine wave is apparent in the image, but more significant is that the design at the center of each repeated instance is slightly different (figure 12b). Figure 12c depicts another example of this phenomenon. The repeating structures are so clearly delineated that they are almost discrete, yet all related to one another.

Figure 13a shows how different coordinate frames can become more or less salient in the overall composition. In the pattern, the effect of the d coordinate is more apparent than sine than in the patterns in figure

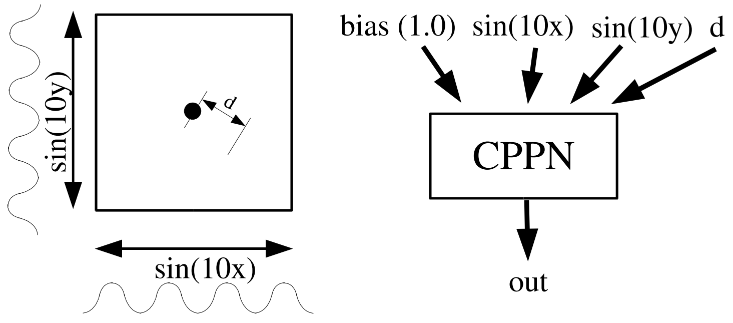


Figure 11: **CPPN Periodic Inputs.** Instead of inputting x and y as usual, $\sin(10x)$ and $\sin(10y)$ are input in the experiments in Section 4.4. These inputs create a novel, repeating coordinate frame on which the CPPN can now elaborate. The purpose of this experiment is to show the powerful effect that coordinate transformation can have on phenotypic patterning.

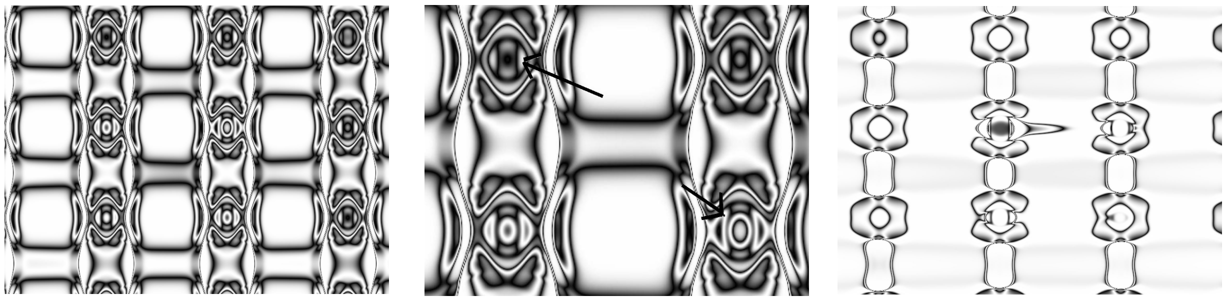
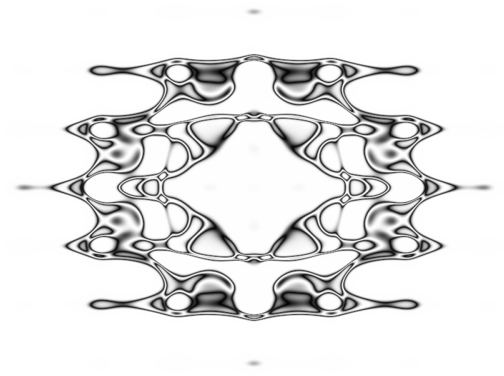


Figure 12: **Repetition with Variation.** The pattern in (b) is a closeup of the upper-left portion of (a). Arrows in (b) point to significant difference between motifs that are otherwise similar. By mixing inputs from separate coordinate frames, one repeating and one not, a CPPN can easily generate a large variety of patterns with this property. This phenomenon is similar to the interaction of two chemical gradients in a developing embryo, one periodic and one not. In (c), the object at the center of the pattern exhibits a strong dissimilarity from its repeated relatives, showing that the effect of interacting coordinate frames can be highly pronounced. In these patterns, the CPPN makes a strong analogy with nature and displays some of its essential capabilities.

12. In figure 13b, d is even more salient.

Finally, it is important to note that such patterns are effectively ubiquitous when a periodic coordinate frame is combined with an aperiodic one. Under such conditions, almost every pattern generated displays repetition with variation (figure 14). Thus, it turns out that such patterns are easily produced by such a combination.

These examples demonstrate how a CPPN can generate repetition with variation, a phenomenon commonly attributed to natural developmental encodings. This phenomenon is an abstraction of the local interaction of two chemical gradients in a developing embryo, one periodic and one not. Figures 12, 13, and 14


(a) Circular cell pattern

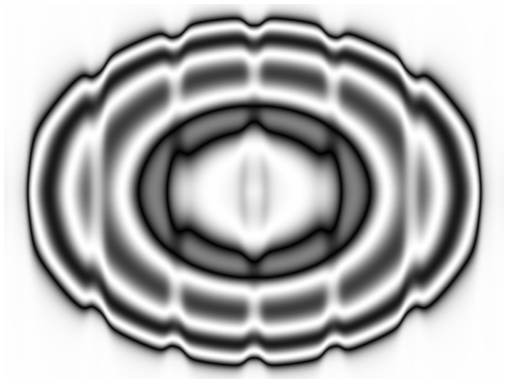

(b) Circularity selected for further

Figure 13: **Distance from Center Becomes the Salient Coordinate Frame.** By selecting patterns for displaying radial symmetry instead of repetition (from the sine waves), it is possible to change which coordinate frame becomes more salient in the overall pattern. While pattern (a) still shows its underlying sinusoidal structure, pattern (b) is becoming increasingly dominated by its radial frame. With continued selection, the sine wave coordinate frame completely disappears. This experiment shows that evolution can select which of the available frames in a CPPN is most important, and furthermore mix and match them with precise ratios.

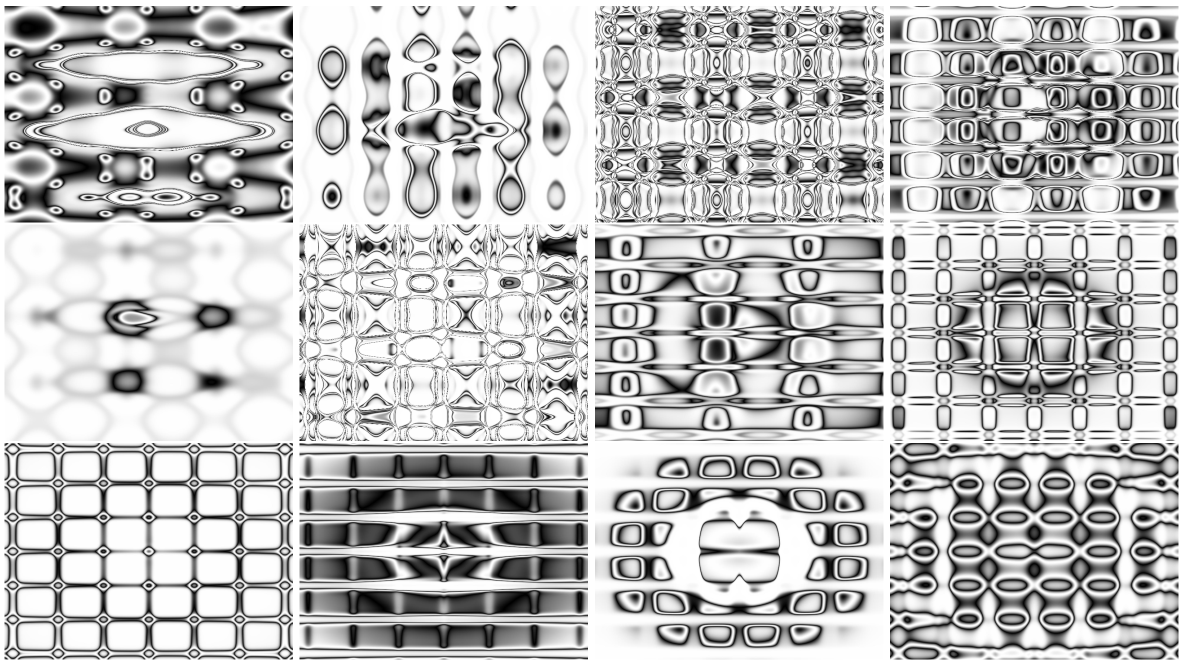


Figure 14: **Ubiquitous Repetition with Variation.** CPPNs easily produce such patterns when the right coordinate frames are introduced. As the figure shows, there are a great variety of such images.

demonstrate that the abstraction works in practice.

While in these examples sine is input to the CPPN to demonstrate its effects explicitly, in practice the CPPN can discover $sin(x)$ or $sin(y)$ on its own by simply by adding nodes with $sine$ activation function internally. In this way, repeating patterns of diverse frequencies can be discovered without provided input.

In summary, even without local interaction or unfolding over time, CPPNs exhibit fundamental properties of natural development, including symmetry, imperfect symmetry, repetition, repetition with variation, elaboration of existing regularities through complexification, and preservation of regularities. In traditional developmental encodings, these motifs are often taken for granted and thus not tested for explicitly; for CPPNs they demonstrate that function composition has more in common with such encodings than is generally acknowledged. For these reasons, CPPNs are admissible into the class of valid abstractions of biological developmental encoding.

5 Discussion and Future Work

Because CPPNs are unlike traditional developmental encodings, it was necessary to establish that they nevertheless belong in the same class of abstractions. Establishing that link is the main contribution of this paper. In doing so, a new and promising research direction is revealed. This section focuses on the promise of this new direction, and where it may lead.

5.1 A Novel Abstraction

A CPPN is not a *simulation* of natural development. Rather, it is an abstraction. The long term goal of such work is not to emulate nature but to succeed in automatically generating complexity on the scale of natural organisms. Thus, nature is inspiration insofar as nature is useful towards this goal.

Nevertheless it is important to identify the extent to which CPPNs capture natural phenomena to solidify their place as a valid abstraction. This paper made an analogy between the local interaction of gradient patterns unfolding over time and simple canonical functions composing with one another to create evermore elaborate patterns. This analogy and the subsequent demonstrations of natural motifs and elaborations underly the argument that CPPNs in effect abstract essential properties of natural encoding.

Yet it may appear that the abstraction ends at encoding while missing other fundamental features of development in nature. For example, developmental is often cited for its connection to embodiment, adaptation, and dynamic processes such as self-repair [3, 53, 65]. The phenotype is viewed as a body inextricably tied to its environment that changes and adapts in concert with and in reaction to its surroundings. All organisms change and adapt over their lifetimes, both cognitively and otherwise. By foregoing local interaction and the normal sequence of temporal unfolding, CPPNs appear to perilously abstract away such crucial phenomena.

However, it turns out that the abstraction naturally encompasses these phenomena as well, further supporting its validity. They key observation is that *position*, which is provided as input to the CPPN, is in effect a property of the environment. That is, each point in the environment has a position, and that position is what is input into the CPPN when it is queried for that point. Thus, in this view, the CPPN is directly a function of the environment. To extend this notion of environmental embodiment further only requires that other position-dependent environmental features are input to the CPPN as well. For example, if the CPPN is being queried for (x, y), it can also be provided chemical concentrations, temperatures, visual stimulus, auditory stimulus, or any other feature of the environment in addition to x and y at the same location. In this way, the spatial pattern generated by the CPPN is as much a product of its environment as any other developmental encoding.

Furthermore, adaptation can then be achieved by iterating the CPPN as the environment changes. That is, the CPPN can be re-queried at every point (or at every point where change has occurred). Interestingly, such iteration reintroduces time, showing that CPPNs can indeed function temporally *when necessary*. In fact, change over time, or adaptation, therefore presents no special problem. This capability supports further that CPPNs belong in the same class as other developmental encodings.

5.2 The Role of Local Interaction

Symmetries and regularities were presented in this paper that in another context might be attributed to a local interactive process. If local interaction can be removed without loss of expressivity, what does it imply about its apparent central role in development?

One interpretation is that local interaction is only necessary in nature because of constraints imposed by physics. In particular, physical space contains no explicit coordinate system, necessitating that developing embryos build their own. In effect, to assume its proper role, every cell in the body must at some point answer the question, “where am I?” Thus, an interesting hypothesis is that local interaction primarily serves to create coordinate systems, i.e. gradients, that signal positional information to cells. In a world where such information is explicit in every point in space (such as CPPNs), the role of each cell can be computed directly as a function of the local coordinate, precluding the need for local interaction. Thus the importance of local interaction to developmental *abstractions* is an open question.

5.3 Not Just Pictures

If CPPNs were only used to draw pictures, their utility would be limited. However, patterns in this paper are depicted visually only for illustration; CPPN output can be interpreted for any objective in any substrate. If the substrate is neural, the CPPN can describe the neural pathways, including all the symmetries, repeating patterns, and elegant group-wise connectivity. If the substrate is physical, the CPPN can encode the building blocks that compose the body, which can then be run in simulation, and later manufactured in the real world. Patterns underly almost all natural and artificial structures; thus a general pattern-generating mechanism can potentially represent any of them.

Just as methods for mapping other developmental encodings to useful phenotypes have motivated much research to date, methods for CPPN interpretation are fruitful avenues for future research as well. Though such research is at an early stage, a few preliminary observations can point in promising directions. While it is natural to interpret multidimensional distributions of particles as straightforward spatial patterns (e.g. as the human eye interprets the two-dimensional patterns in this paper), such interpretation may not most effectively exploit the encoding. Spatial patterns intuitively evoke the idea of contiguous morphology, and it may indeed prove useful to generate morphologies based directly on CPPN-output spatial patterns. However, contiguous morphologies have limited application.

A significant problem with such structures is that they lack discrete divisions among their parts. For example, it would be difficult in robot design to decide where the arm ends and where the torso begins. Similarly, spatial patterns are ambiguous with respect to connectivity in e.g. a neural network. Yet this problem does not imply that CPPNs lack the apparatus to effectively represent such structures; rather, it suggests that some phenotypes are best represented in a higher-dimensional space than their observable physical morphology. That is, such structures are characterized crucially by their *connectivity*, and connectivity includes hidden spatial dimensions. For example, a connection between two points on a two-dimensional plane actually requires *four* dimensions to describe: (x_1, y_1) and (x_2, y_2). Thus a four-dimensional CPPN (i.e. one that takes four inputs rather than two) creates a pattern within a hypercube that can be projected back into two dimensions as a connectivity pattern.[3].

The broader implication is that while many kinds of structures may appear to require information that

[3]Our research group is currently experimenting with four-dimensional CPPNs that produce two-dimensional connectivity patterns, with promising preliminary results

is not inherent within spatial patterns, this problem can be solved by embedding solutions in a higher-dimensional space wherein such details can now fit; CPPNs can produce patterns in any number of dimensions simply by providing the right number of orthogonal axes as inputs. Thus, CPPNs can potentially be employed as general pattern generators for any substrate in any domain.

5.4 Complexification and CPPNs

A final important result, evident in the spaceship experiment, is that CPPNs can complexify in a similar way to natural organisms. Fundamental regularities can be established early in evolution, and then expanded and exploited through a process of elaboration. This elaboration is possible because the *size* of the genome is expanding, thereby expanding the range of complexity that the CPPN can represent. Thus, a significant benefit of the CPPN abstraction is that a principled method for complexifying network structures, NEAT, already exists. With little modification NEAT is extended to CPPN-NEAT, making possible experiments in continual elaboration of form.

6 Conclusion

Research in developmental encoding to this date can be characterized as a search for the right abstraction. Computational abstractions of natural development generally have included, not surprisingly, local interaction and unfolding over time. In this paper, a novel abstraction was proposed that breaks this tradition. Nevertheless, this abstraction, called Compositional Pattern Producing Networks (CPPNs), still captures essential properties of natural developmental encoding. A series of interactive evolutionary experiments with CPPNs demonstrated that they indeed produce patterns and sequences of patterns with properties analogous to those seen in natural evolution and development. Furthermore, an algorithm for evolving increasingly complex CPPNs, called CPPN-NEAT, was introduced. Thus, a promising new encoding that is demonstrably capable of discovering symmetry and repetition with variation is revealed. Future research will focus on converting the patterns output by CPPNs into important substrates such as large-scale neural networks, physical morphologies, and building architectures.

Acknowledgments

Special thanks to Mattias Fagerlund for creating the first NEAT-based genetic art program based on his DelphiNEAT implementation, to Holger Ferstl for creating the second NEAT-based genetic art program based on SharpNEAT, and to Colin Green for creating SharpNEAT. All the software and source code used in this paper, including DNGA, SNGA, DelphiNEAT, and SharpNEAT, is available through `http://www.cs.ucf.edu/∼kstanley`.

References

[1] Altenberg, L. (1994). Evolving better representations through selective genome growth. In *Proceedings of the IEEE World Congress on Computational Intelligence*, 182–187. Piscataway, NJ: IEEE Press.

[2] Amores, A., Force, A., Yan, Y.-L., Joly, L., Amemiya, C., Fritz, A., Ho, R. K., Langeland, J., Prince, V., Wang, Y.-L., Westerfield, M., Ekker, M., and Postlethwait, J. H. (1998). Zebrafish HOX clusters and vertebrate genome evolution. *Science*, 282:1711–1784.

[3] Andersen, T., Otter, C., Petschulat, C., Eoff, U., Menten, T., Davis, R., and Crowley, B. (2005). A biologically-derived approach to tissue modeling. In et al, J. W., editor, *Technology and Informatics*, 15–21. Amsterdam: IOS Press.

[4] Angeline, P. J. (1995). Morphogenic evolutionary computations: Introduction, issues and examples. In McDonnell, J. R., Reynolds, R. G., and Fogel, D. B., editors, *Evolutionary Programming IV: The Fourth Annual Conference on Evolutionary Programming*, 387–401. MIT Press.

[5] Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1993). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5:54–65.

[6] Astor, J. S., and Adami, C. (2000). A developmental model for the evolution of artificial neural networks. *Artificial Life*, 6(3):189–218.

[7] Baluja, S., Pomerleau, D., and Jochem, T. (1994). Towards automated artificial evolution for computer-generated images. *Connection Science*, 6(2 and 3):325–354.

[8] Behravan, R., and Bentley, P. J. (2003). Exploring reaction-diffusion and pattern formation. In *Proceedings of the First Australian Conference on Artificial Life (ACAL 2003)*.

[9] Belew, R. K., and Kammeyer, T. E. (1993). Proceedings of the fifth international conference on genetic algorithms. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Francisco: Kaufmann.

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

[11] Boers, E. J. W., and Kuiper, H. (1992). *Biological Metaphors and the Design of Modular Artificial Neural Networks*. Master's thesis, Departments of Computer Science and Experimental and Theoretical Psychology at Leiden University, The Netherlands.

[12] Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation*.

[13] Bongard, J. C., and Pfeifer, R. (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In [71], 829–836.

[14] Cangelosi, A., Parisi, D., and Nolfi, S. (1993). Cell division and migration in a genotype for neural networks. Technical Report PCIA-93, Institute of Psychology, C.N.R., Rome.

[15] Carroll, S. B. (1995). Homeotic genes and the evolution of arthropods and chordates. *Nature*, 376:479–485.

[16] Curtis, D., Apfeld, J., and Lehmann, R. (1995). Nanos is an evolutionarily conserved organizer of anterior-posterior polarity. *Development*, 121:1899–1910.

[17] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

[18] Darnell, J. E., and Doolittle, W. F. (1986). Speculations on the early course of evolution. *Proceedings of the National Academy of Sciences, USA*, 83:1271–1275.

[19] Dawkins, R. (1986). *The Blind Watchmaker.* Essex, U.K.: Longman.

[20] Dawkins, R. (1989). The evolution of evolvability. In Langton, C. G., editor, *Artificial Life*, 201–220. Reading, MA: Addison-Wesley.

[21] Dellaert, F. (1995). *Toward a Biologically Defensible Model of Development*. Master's thesis, Case Western Reserve University, Clevekand, OH.

[22] Dellaert, F., and Beer, R. D. (1996). A developmental model for the evolution of complete autonomous agents. In Maes, P., Mataric, M. J., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.

[23] Deloukas, P., Schuler, G. D., Gyapay, G., Beasley, E. M., Soderlund, C., Rodriguez-Tome, P., Hui, L., Matise, T. C., McKusick, K. B., Beckmann, J. S., Bentolila, S., Bihoreau, M., Birren, B. B., Browne, J., Butler, A., Castle, A. B., Chiannilkulchai, N., Clee, C., Day, P. J., Dehejia, A., Dibling, T., Drouot, N., Duprat, S., Fizames, C., and Bentley, D. R. (1998). A physical map of 30,000 human genes. *Science*, 282(5389):744–746.

[24] Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. In Husbands, P., and Harvey, I., editors, *Proceedings of the Fourth European Conference on Artificial Life*, 205–213. Cambridge, MA: MIT Press.

[25] Federici, D. (2004). Evolving a neurocontroller through a process of embryogeny. In Schaal, S., Ijspeert, A. J., Billard, A., Vijayakumar, S., Hallam, J., and Jean-Arcady, editors, *Proceedings of the Eighth International Conference on Simulation and Adaptive Behavior (SAB-2004)*, 373–384. Cambridge, MA: MIT Press.

[26] Federici, D. (2004). Using embryonic stages to increase the evolvability of development. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) Workshop Program*. Berlin: Springer Verlag.

[27] Fleischer, K., and Barr, A. H. (1993). A simulation testbed for the study of multicellular development: The multiple mechanisms of morphogenesis. In Langton, C. G., editor, *Artificial Life III*, 389–416. Addison-Wesley.

[28] Force, A., Lynch, M., Pickett, F. B., Amores, A., lin Yan, Y., and Postlethwait, J. (1999). Preservation of duplicate genes by complementary, degenerative mutations. *Genetics*, 151:1531–1545.

[29] Gilbert, C. D., and Wiesel, T. N. (1992). Receptive field dynamics in adult primary visual cortex. *Nature*, 356:150–152.

[30] Gomez, F., and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 1356–1361. San Francisco: Kaufmann.

[31] Greenfield, G. R. (2000). Evolving expressions and art by choice. *Leonard*, 33(2):93–99.

[32] Gruau, F., Whitley, D., and Pyeatt, L. (1996). A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Cambridge, MA: MIT Press.

[33] Hornby, G. S., and Pollack, J. B. (2001). The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2002 Congress on Evolutionary Computation*.

[34] Hornby, G. S., and Pollack, J. B. (2001). Body-brain co-evolution using L-systems as a generative encoding. In [71].

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

[36] Hubel, D. H., and Wiesel, T. N. (1965). Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of Neurophysiology*, 28:229–289.

[37] Husbands, P., Germy, G., McIlhagga, M., and Ives, R. (1996). Two applications of genetic algorithms to component design. *Evolutionary Computing. LNCS 1143*, 50–61.

[38] Jakobi, N. (1995). Harnessing morphogenesis. In *Proceedings of Information Processing in Cells and Tissues*, 29–41. University of Liverpool.

[39] Julian F. Miller, P. T. (2000). Cartesian genetic programming. In *Proceedings of the Third European Conference on Genetic Programming Published as Lecture Notes in Computer Science, Vol. 1802*, 121–132.

[40] Kandel, E. R., Schwartz, J. H., and Jessell, T. M., editors (1991). *Principles of Neural Science*. Amsterdam: Elsevier. Third edition.

[41] Kaneko, K., and Furusawa, C. (1998). Emergence of multicellular organisms with dynamic differentiation and spatial pattern. *Artificial Life*, 4(1).

[42] Kauffman, S. A. (1993). *The Origins of Order*. New York: Oxford University Press.

[43] Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, 4:461–476.

[44] Komosinski, M., and Rotaru-Varga, A. (2001). Comparison of different genotype encodings for simulated 3D agents. *Artificial Life*, 7(4):395–418.

[45] Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

[46] Lall, S., and Patel, N. (2001). Conservation and divergence in molecular mechanisms of axis formation. *Annual Review of Genetics*, 35:407–447.

[47] Lindenmayer, A. (1968). Mathematical models for cellular interaction in development parts I and II. *Journal of Theoretical Biology*, 18:280–299 and 300–315.

[48] Lindenmayer, A. (1974). Adding continuous components to L-systems. In Rozenberg, G., and Salomaa, A., editors, *L Systems, Lecture Notes in Computer Science 15*, 53–68. Heidelberg, Germany: Springer-Verlag.

[49] Lund, H. H., Pagliarini, L., and Miglino, P. (1995). Artistic design with GA and NN. In *Proceedings of the 1st Nordic Workshop on Genetic Algorithms and Their Applications (1NWGA)*, 97–105.

[50] Martin, A. P. (1999). Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, 154(2):111–128.

[51] McCormack, J. P. (1993). Interactive evolution of L-system grammars for computer graphics modelling. In Green, D. G., and Bossomaier, T., editors, *Complex Systems: From Biology to Computation*, 118–130. Amsterdam: IOS Press.

[52] Meinhardt, H. (1982). *Models of Biological Pattern Formation*. London: Academic Press.

[53] Miller, J. F. (2003). Evolving developmental programs for adaptation, morphogenesis, and self-repair. In Banzhaf, W., Christaller, T., Dittrich, P., Kim, J. T., and Ziegler, J., editors, *Advances in Artificial Life. 7th European Conference on Artificial Life*, vol. 2801 of *Lecture Notes in Artificial Intelligence*, 256–265. Dortmund, Germany: Springer.

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

[55] Mjolsness, E., Sharp, D. H., and Reinitz, J. (1991). A connectionist model of development. *Journal of Theoretical Biology*, 152:429–453.

[56] Nishino, H., Takagi, H., Cho, S.-B., and Utsumiya, K. (2001). A 3D modeling system for creative design. In *15th International Conference on Information Networking*, 479–486.

[57] Nolfi, S., and Parisi, D. (1991). Growing neural networks. Technical Report PCIA-91- 15, Institute of Psychology, C.N.R., Rome.

[58] Opitz, D. W., and Shavlik, J. W. (1997). Connectionist theory refinement: Genetically searching the space of network topologies. *Journal of Artificial Intelligence Research*, 6:177–209.

[59] Parmee, I. C. (2002). Improving problem definition through interactive evolutionary computation. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacture - Special Issue: Human-computer Interaction in Engineering*, 16(3):185–202.

[60] Prusinkiewicz, P., and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Heidelberg, Germany: Springer-Verlag.

[61] Pujol, J. C. F., and Poli, R. (1997). Evolution of the topology and the weights of neural networks using genetic programming with a dual representation. Technical Report CSRP-97-7, School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK.

[62] Radcliffe, N. J. (1993). Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90.

[63] Raff, R. A. (1996). *The Shape of Life: Genes, Development, and the Evolution of Animal Form*. Chicago: The University of Chicago Press.

[64] Reisinger, J., Stanley, K. O., and Miikkulainen, R. (2005). Towards an empirical measure of evolvability. In *Genetic and Evolutionary Computation Conference (GECCO2005) Workshop Program*, 257–264. Washington, D.C.: ACM Press.

[65] Roggen, D., and Federici, D. (2004). Multi-cellular development: Is there scalability and robustness to gain. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, 391–400.

[66] Saravanan, N., and Fogel, D. B. (1995). Evolving neural control systems. *IEEE Expert*, 23–27.

[67] Schnier, T. (1998). *Evolved Representations and Their Use in Computational Creativity*. PhD thesis, University of Sydney Department of Architectural and Design Science, Australia.

[68] Sims, K. (1991). Artificial evolution for computer graphics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*, 319–328. New York, NY: ACM Press.

[69] Sims, K. (1994). Evolving 3D morphology and behavior by competition. In Brooks, R. A., and Maes, P., editors, *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV)*, 28–39. Cambridge, MA: MIT Press.

[70] Smith, J. R. (1991). Designing biomorphs with an interactive genetic algorithm. In Belew, R. K., and Booker, L. B., editors, *Proceedings of the 4th International Conference on Genetic Algorithms (ICGA-91)*, 535–538. San Mateo, CA: Morgan Kaufmann.

[71] Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors (2001). *Proceedings of the Genetic and Evolutionary Computation Conference*. San Francisco: Kaufmann.

[72] Stanley, K. O., Bryant, B. D., and Miikkulainen, R. (2005). Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation Special Issue on Evolutionary Computation and Games*, 9(6):653–668.

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

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

[75] Stanley, K. O., and Miikkulainen, R. (2004). Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100.

[76] Stanley, K. O., Reisinger, J., and Miikkulainen, R. (2004). Exploiting morphological conventions for genetic reuse. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) Workshop Program*. Berlin: Springer Verlag.

[77] Takagi, H. (2001). Interactive evolutionary computation: Fusion of the capacities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296.

[78] Todd, S., and Latham, W. (1992). *Evolutionary Art and Computers*. London: Academic Press.

[79] Todd, S., and Latham, W. (1999). *The Mutation and Growth of Art by Computers*, chapter 9, 221–250. Morgan Kaufmann.

[80] Turing, A. (1952). The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72.

[81] Watson, J. D., Hopkins, N. H., Roberts, J. W., Steitz, J. A., and Weiner, A. M. (1987). *Molecular Biology of the Gene*. Menlo Park, CA: The Benjamin Cummings Publishing Company, Inc. Fourth edition.

[82] Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

[83] Yao, X., and Liu, Y. (1996). Towards designing artificial neural networks by evolution. *Applied Mathematics and Computation*, 91(1):83–90.

[84] Zhang, B.-T., and Muhlenbein, H. (1993). Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, 7:199–220.

[85] Zigmond, M. J., Bloom, F. E., Landis, S. C., Roberts, J. L., and Squire, L. R., editors (1999). *Fundamental Neuroscience*. London: Academic Press.