Evolutionary Design via Indirect Encoding of Non-Uniform Rational Basis Splines

Adam Gaier
Bonn-Rhein-Sieg University of Applied Sciences
Grantham-Allee 20
Sankt Augustin, Germany
adam.gaier@h-brs.de

ABSTRACT

A novel approach to produce 2D designs by adapting the HyperNEAT algorithm to evolve non-uniform rational basis splines (NURBS) is presented. This representation is proposed as an alternative to previous pixel-based approaches primarily motivated by aesthetic interests, and not designed for optimization tasks. This spline representation outperforms previous pixel-based approaches on target matching tasks, performing well even in matching irregular target shapes. In addition to improved evolvability in the face of a well-defined fitness metric, a NURBS representation has the added virtues of being continuous rather than discrete, as well as being intuitive and easily modified by graphic and industrial designers.

CCS Concepts

•**Computing methodologies** → **Genetic algorithms;** *Parametric curve and surface models;* Neural networks;

Keywords

Design Optimization; HyperNEAT; NURBS; Indirect Encodings; Compositional Pattern Producing Networks

1. INTRODUCTION

Increased availability of high performance computing resources, coupled with the growing sophistication of surrogate modeling techniques [4], is rapidly increasing the capabilities of evolutionary design optimization from toy problems to real world engineering applications. For the field to truly progress strides must also be made in the development of flexible design representations which can not only express complex objects but can be effectively evolved.

Designers of evolutionary design optimization algorithms often find that it is necessary to reduce the number of design parameters to make the problem tractable. This can be done by crafting domain specific representations, for example in aerodynamics optimization PARSEC parameters which define the geometry of an airfoil [9], or by narrowing the design focus, such as reducing the aerodynamics of an automobile body to the angle of the rear section while keeping the rest of the body the same [3].

In many cases however such a reduction is either not feasible or not desired. If, again using the aerodynamics case as an example, we would like to optimize the entire body of the car for low wind resistance we are confronted with an overwhelming number of parameters which define the shape in its entirety. Not only are there a large number of parameters, but these parameters also interact; the shape of the front of a car has a large effect on the air flow at the back. In order to simultaneously optimize a large number of interacting parameters we look to tools used in neuroevolution, a field which faces the same difficulty.

2. RELATED WORK

HyperNEAT evolves Compositional Pattern Producing Networks (CPPNs), which encode spatial patterns as a composition of simple functions [10, 12]. Each node in the network is assigned a geometric position, and the positions of the two nodes are used as input to the CPPN. The resulting output of the CPPN is then assigned as the weight of the connection between the two nodes. This indirect representation allows for neural networks, regardless of the number of nodes and weights, to be effectively optimized. CPPNs take the form of a directed graph with weighted edges, and so themselves are similar enough in structure to be evolved using the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [11]. The NEAT algorithm begins with minimal neural networks, which gradually become more complex through mutation. HyperNEAT, the evolution of CPPNs via NEAT to produce neural networks, has proven effective across a variety of domains [2].

The HyperNEAT algorithm has also been applied to the design of both pixels of 2D patterns [8] and voxels of 3D objects [1]. These experiments have primarily been motivated by creating interesting patterns and shapes, and so the majority of work has been limited to evolution based on user preference, emphasizing demonstrations of expressivity over the ability to evolve in response to a well-defined fitness measure. What optimization experiments have been done with these representations have involved matching a form to a preexisting shape, with the hope that an existing image or object could be converted into a CPPN representation and be evolved further based on user input. Experiments

GECCO '15, July 11 - 15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739482.2768478

with both the 2D [13] and 3D case [1] have shown that the HyperNEAT algorithm, coupled with a pixel or voxel representation, struggles to match all but the most trivial objects. In an attempt to improve this matching capability, an alternative representation, using ellipses rather than pixels as the basic building block, has been proposed in order to better convert an existing shape to a CPPN representation, though at present it has only been tested in a user-driven manner [6].

Here we present an approach using the HyperNEAT algorithm to produce Non Uniform Rational Basis Splines (NURBS) [5]. NURBS are a widely used design representation both in engineering and computer graphics. This is particularly important in an evolutionary design setting, as evolved designs must be able to be intuitively understood and altered by others. We demonstrate how to adapt the HyperNEAT algorithm to a NURBS representation, and compare it to the pixel approach in basic tests of evolvability.

3. METHODS

We compare two representations used to produce two dimensional shapes, both based on a CPPN encoding evolved through the HyperNEAT algorithm. First we define a substrate. The substrate is composed of geometric coordinates which are used input to the CPPN. In the case of the pixel-based approach we have two layers, which correspond to the x and y coordinates of the image. The values of the matching cells in each layer are paired and used as input values (along with a bias[1]) to the CPPN. The output value, determined by the weighted connections and various activation functions of the CPPN, is then assigned as an intensity value for the pixel associated with the substrate cell.

In order to produce a well-defined shape we use a threshold value (i.e. $intensity > 0$) to determine whether a pixel is black or white. This same technique was used for the evolution of 3D shapes to determine whether a given voxel was filled or empty [1]. Once we have converted the output to a binary image we take the border of the largest shape as our expressed design. This process is illustrated on the left half of Figure 1. While the CPPN is a continuous representation, it is expressed in a discrete space determined by the resolution of the substrate.

To use the HyperNEAT approach to produce splines we return to the concept of a pattern of connectivity. A single NURBS curve in isolation can be understood as a start point, an end point, and an intermediate control point which determines the form of the curve. We interpret this intermediate control point as defining the connection between the start and end points. In the neural network case a source node and a destination node are taken as input, and the output as the resulting connection weight. Here we take a starting point and an end point as the input, and the output is the resulting intermediate control point. The two substrate layers in this case are arranged in a ring, where the values of the second layer are identical to the first but shifted two places in the clockwise direction, creating pairings of start and end points. Each of these pairings are then given to the CPPN which outputs the x and y coordinates, weight value,

[1]not shown in Figure 1

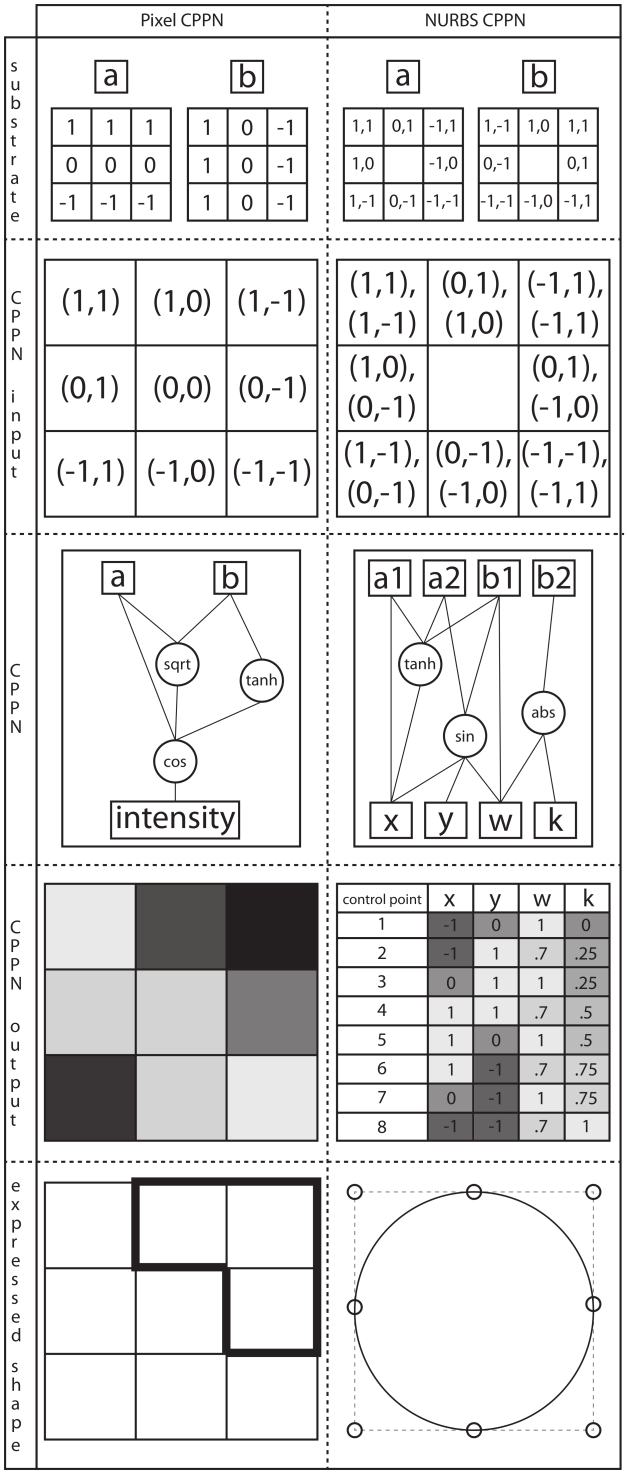

Figure 1: Shape expression using a pixel(left) and NURBS(right) representation. Note that a pixel representation would typically be expressed with far higher resolution, here, for illustrative purposes, only 9 cells are shown.

and corresponding knot of the intermediate point. This process is illustrated in the right half of Figure 1.

The output of the CPPN is mapped directly to the value of the x and y of the control point, as weights cannot be negative they are mapped to a value between 0 and 2. The values of the knot vector of a NURBS curve must be ascending, so the knot value is computed as the sum of the CPPN output and the knot value of the previous control point. As the knot values are locations within line space, they must be scaled to begin at 0 and end at 1.

The use of an indirect encoding allows us to easily adjust the resolution of the output, adding control points in the NURBS case and increasing the number of pixels in the pixel case. This allows us to begin by optimizing simpler shapes and scale up to produce more complex designs. This complexification could be done in an adaptive way, triggered when a solution reaches a certain level of precision, or when the population has stagnated. In order to easily examine the effects of this complexification over multiple runs, we increase the resolution of the output after a set number of generations.

4. RESULTS

To compare the expressibility and ability of the two representations to respond to evolutionary pressure we compare them on a basic target matching task. A target shape is given, and individuals are evaluated based on how closely they match the target. To determine this we take the mean squared distance from 150 points placed along the perimeter of the target shape to the evolved shape. In addition, a fitness penalty is applied according to distance of the farthest point on the evolved shape to the target shape. This penalizes the evolution of shapes which cover the target, but also have extraneous sections which do not match it.

We compare the performance of the representations on two target objects: a circle, which is symmetrical in every axis and does not contain any corners, and a 2D silhouette of a vehicle, which is neither symmetrical nor smooth. A population of 50 CPPNs, with no hidden nodes, and inputs fully connected to outputs, were initialized with random weights. Available CPPN activation functions were: linear, squared, square root[2], absolute value, gaussian[3], hyperbolic tangent, cosine, and sin, all with equal probability.

One run is composed of 1000 generations, with the resolution of the output increased at generations 251, 501, and 751. The PixelCPPN used a 21X21 square grid, which increased to 41X41, 81X81, and 121X121. The hyperNURBS substrate begins as a 3X3 ring, which increases to 5X5, 7X7, and 9X9. All substrate values in both cases were evenly spaced between -1 and 1. The mean squared error of the best performing individuals over 30 runs of each algorithm on each target are compared.

In both the regular and irregular cases the NURBS representation outperforms the pixel representation. Though PixelCPPNs which produce a circular pattern appear, the precision is limited by the discretization of the pixel grid. In contrast the NURBS based representation operates in continuous space, and so has no such limitation. High performing (>0.999 MSE) solutions were found with both 33 control points located roughly on the perimeter of the circle, as well as with only 8 control points in a rhombus surrounding the target.

[2]for negative values -1*(sqrt(abs(x)))
[3]sigma = 1, mean = 0

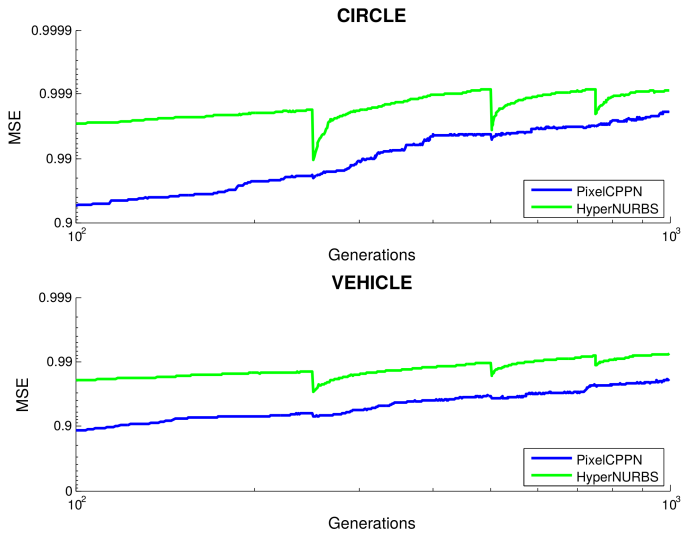

Figure 2: Mean squared error of highest fitness shapes produced by pixel and NURBS representation. Note: Both error and generations are in log scale, with the first 100 generations omitted for axis scaling purposes.

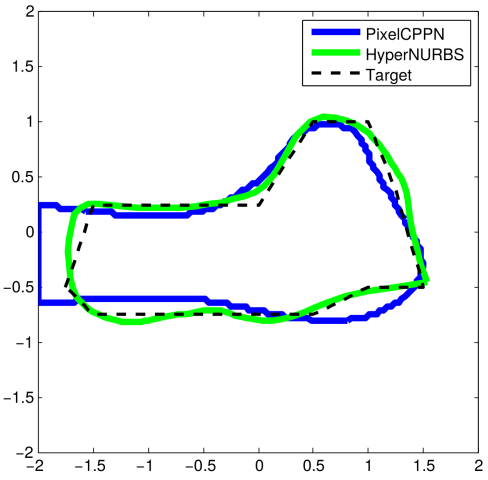

Figure 3: Best performing shapes produced for irregular vehicle target by PixelCPPN and HyperNURBS representation.

The PixelCPPN representation struggled not only to replicate the corners of the shape, but the basic border as well. The majority of even the best performing individuals had at least one border defined by the bounds of the workspace, rather than the CPPN values. In constrast the NURBS

representation quickly arrived at the basic shape, and in further generations only relatively small adjustments were made. The resulting best fit shapes for the irregular target are shown in Figure 3.

Though increasing the resolution of the output only based on the number of iterations is not ideal for optimization, it allows us to clearly see its effect across multiple runs. As can be seen in Figure 2 the PixelCPPN representation suffered little negative effects from the transition, and a greater resolution typically resulted in quickened convergence rate. This is to be expected, as a higher resolution of pixels means a higher discretization level and greater precision, and the underlying pattern is unchanged.

This is not so with a NURBS curve. Thought the underlying pattern is unchanged, adding additional control points, even along an already existing curve, will alter the shape. Adding control points resulted in a shape which was a deformed version of the original. In most cases the original fitness was able to recover quickly, and with the additional flexibility of control points, surpassed. Those individuals whose control point pattern was not robust to this addition of control points, for instance twisted shapes which were able to cover most of the target perimeter without resembling the shape, do not survive long at higher resolution. It can be seen from figure 2, a more sophisticated mechanism for determining the output resolution must be considered; the increase in resolution cut short the optimization process still in progress at the lower resolution.

5. CONCLUSIONS AND FUTURE WORK

We have presented a technique for evolving 2D designs based on NURBS curves using the HyperNEAT algorithm. By using a NURBS representation we produce designs in a continuous space which can be easily understood and altered by human designers. In addition to ease of use we have shown that this alternate formulation responds more readily to evolutionary pressure, and so is a step forward toward the use of HyperNEAT not just for the creation of aesthetically pleasing patterns, but for design optimization problems.

More sophisticated ways of altering the number of control points expressed must be explored. This could be done either with triggers, such as predefined levels of fitness or periods of stagnation, or based on the variance in the expressed values as is done for hidden neurons in ES-HyperNEAT [7]. Though difficult to see from the mean values, the fitness level of the highest performing solutions did increase quickly after a resolution increase, while others had their fitness reduced.

In our naive approach the substrate values were always evenly spaced, and the same along every dimension. An evolvable substrate would allow areas of the shape which need higher precision to contain more detail through a greater number of control points, while maintaining broad curves when delicate forms are not necessary.

One of the great strengths of indirect encodings is their ability to express solutions at various resolutions. That in many cases increasing the resolution resulted in a very different shape is symptomatic of deficiencies in the representation. Preliminary experiments showed that determining the CPPN input based on the neighboring points in the substrate performed better than using the point itself however, in light of the difficulties this introduces in scaling the representation, closer examination is required.

We have presented an approach for spline evolution in 2D, the basic principles of which can be applied to the 3D case. While transferring this to the 3D case may not be as simple as adding another input dimension, as from pixel evolution to voxel evolution, the first steps have been made. The basic building blocks of 3D NURBS surfaces are patches, themselves composed of splines. As progress in developing the best way to represent and evolve the basic spline building block continues, we will find ourselves closer to the goal of detailed, complex, evolutionary design.

6. REFERENCES

[1] J. Clune and H. Lipson. Evolving 3d objects with a generative encoding inspired by developmental biology. *ACM SIGEVOlution*, 5(4):2–12, 2011.

[2] D. D'Ambrosio, J. Gauci, and K. Stanley. HyperNEAT: The First Five Years. *Growing Adaptive Machines*, 557:159–185, 2014.

[3] L. Dumas. CFD-based optimization for automotive aerodynamics. *Optimization and Computational Fluid Dynamics*, 05, 2008.

[4] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 2011.

[5] L. Piegl and W. Tiller. The nurbs book. 1997. *Monographs in Visual Communication*, 1997.

[6] S. Risi. A Compiler for CPPNs: Transforming Phenotypic Descriptions Into Genotypic Representations. *Proceedings of the 2013 AAAI Fall Symposium on How Should Intelligence be Abstracted in AI Research*, 2013.

[7] S. Risi, J. Lehman, and K. O. Stanley. Evolving the placement and density of neurons in the hyperneat substrate. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, page 563, New York, New York, USA, July 2010. ACM Press.

[8] J. Secretan and N. Beato. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation*, 2011.

[9] H. Sobieczky. Parametric airfoils and wings. In *Recent Development of Aerodynamic Design Methodologies*, pages 71–87. Springer, 1999.

[10] K. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 2007.

[11] K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 2002.

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

[13] B. Woolley and K. Stanley. On the deleterious effects of a priori objectives on evolution and representation. *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011.