http://dx.doi.org/10.14236/ewic/EVA2016.23

Species Explorer: An interface for artistic exploration of multi-dimensional parameter spaces

Andy Lomas
Independent Artist
20 Kingsley Avenue, West Ealing,
London W13 0EF, UK
andylomas@yahoo.com

This paper describes Species Explorer, an interface to allow creative exploration of generative systems with multi-dimensional parameter spaces. The system combines both evolutionary and machine learning approaches. It was originally designed to assist creating work for the author's 'Cellular Forms' and 'Hybrid Forms' series, where a large number of parameters are used to yield emergent results, but is a general framework that could be applied to many other systems.

Generative art. Evolutionary design. Machine learning. Computationally assisted design.

1. INTRODUCTION

Species Explorer is a system that was developed out of necessity: how to deal with increasingly large numbers of parameters in systems for computer based generative art while retaining creative influence.

Typically generative systems are based on an algorithmic process that is controlled by a number of parameters. Given a set of parameter values the process can be run to create an output. Classic examples include Conway's Game of Life (Conway 1970) and reaction diffusion equations (Turing 1952).

The most interesting systems are generally those that create emergent results. For these systems the relationship between the input parameters and the output is typically complex and non-linear, with effects such as sensitive dependence on initial conditions.

With a small number of dimensions, such as up to 3 parameters, the space of results can be relatively easily explored by simply varying individual parameter values and plotting the effects of different parameter combinations. However, as more parameters are added it becomes increasingly difficult to use this type of approach to explore the space and influence the results in a creative direction.

Species Explorer is designed to allow the user to work with such systems, guiding the search with a creative intent without being overwhelmed by having to deal directly with all the individual parameters. The system uses a hybrid approach, mixing both evolutionary search methods, such as mutation and cross-breeding, with lazy machine learning techniques. The system acts as a framework that allows different methods to be used to create 'populations' of 'individuals', where each population can utilize a different technique for the selection of the parameters to generate individuals. In particular, different methods may be appropriate depending on the artist's intent. For example:

- Initial exploration, where the user is trying to get a basic understanding of how a system works and what sort of result may be possible.
- Secondary exploration where the search is steered into broad areas of the parameter space that appear to be potentially fruitful, and away from regions that have been found to yield invalid results or are otherwise undesirable.
- Refined focus within a small range of parameter values that appear to produce interesting results, often to create final artistic artefacts.
- Looking for novelty: searching the space for results unlike those seen previously that may take the exploration in a new direction.

The essential idea is that the computer acts as an assistant in the process: taking feedback from the user, such as what they like or do not like, and proposing new parameter values to try that will hopefully generate interesting new results and aid exploring the potential of the generative system.

2. PREVIOUS WORK

With a small number of parameters, the space of possibilities can usually be explored quite effectively by simply trying out combinations of parameter values. One technique that is common is to create a chart where all the parameters are sampled independently at regularly spaced values and results are plotted to show the results.

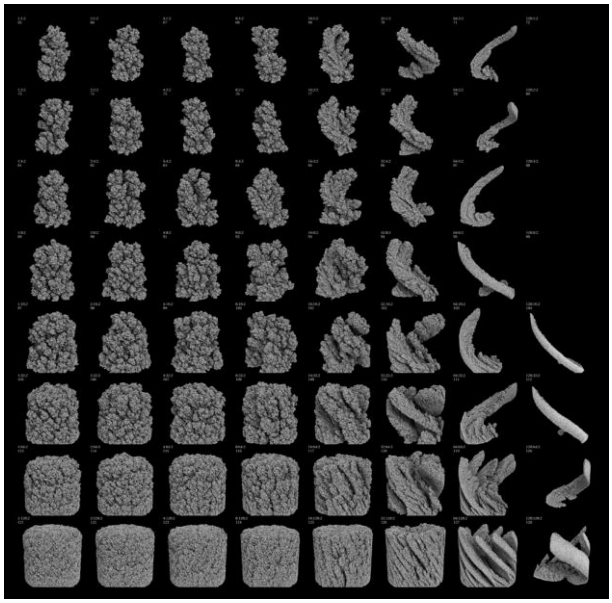

Figure 1: *Chart exploring the effect of varying parameter values from the Aggregation series*

This method of parameter exploration can be effective, and was used by the author for earlier work such as for his 'Aggregation' and 'Flow' series, but only typically works with up to 3 parameters.

A number of authors have proposed using evolutionary methods to explore larger numbers or parameters. Examples include Dawkin's Biomorphs (Dawkins 1986) and Mutator (Todd & Latham 1992). A number of systems that use evolutionary selection for design are described in (Bentley 1999).

As demonstrated by natural processes, evolutionary methods can be effective even with extremely large numbers of parameters. One problem though can be that these methods generally lead to exploring a small number of paths within the space of available possibilities. Children are typically created with parameter values similar to parents, which can bias the search towards areas of the parameter space that have already been highly sampled.

In more recent years a number of authors have proposed using machine learning techniques to assist human designers. In general these are for domain specific applications, such as for architectural space frame structures (Hanna 2007), structurally valid furniture (Umetani, Igarashi & Mitra 2012) or aircraft designs (Oberhauser, *et al.* 2015). In these systems machine learning is typically used to learn about specific properties of the system. This is then used to provide interactive feedback for the user about whether an object designed by them is likely to have desired properties, such as being structurally feasible, without having to do computationally prohibitive tasks such as full finite element analysis.

3. AIMS

The key intent is for the computer to act as an active assistant, helping guide the user as they explore a system to discover its potential capabilities and making the best use of all the input the user has made. The user should be able to steer the search with a creative intent, refining particularly interesting results, with the computer assisting them in exploring the space for novel rich behaviour.

The specific need for such a system came from the number of parameters that the author found he needed when he was developing the simulation engine for 'Cellular Forms' (Lomas 2014).

Figure 2: *Image from the Cellular Forms series*

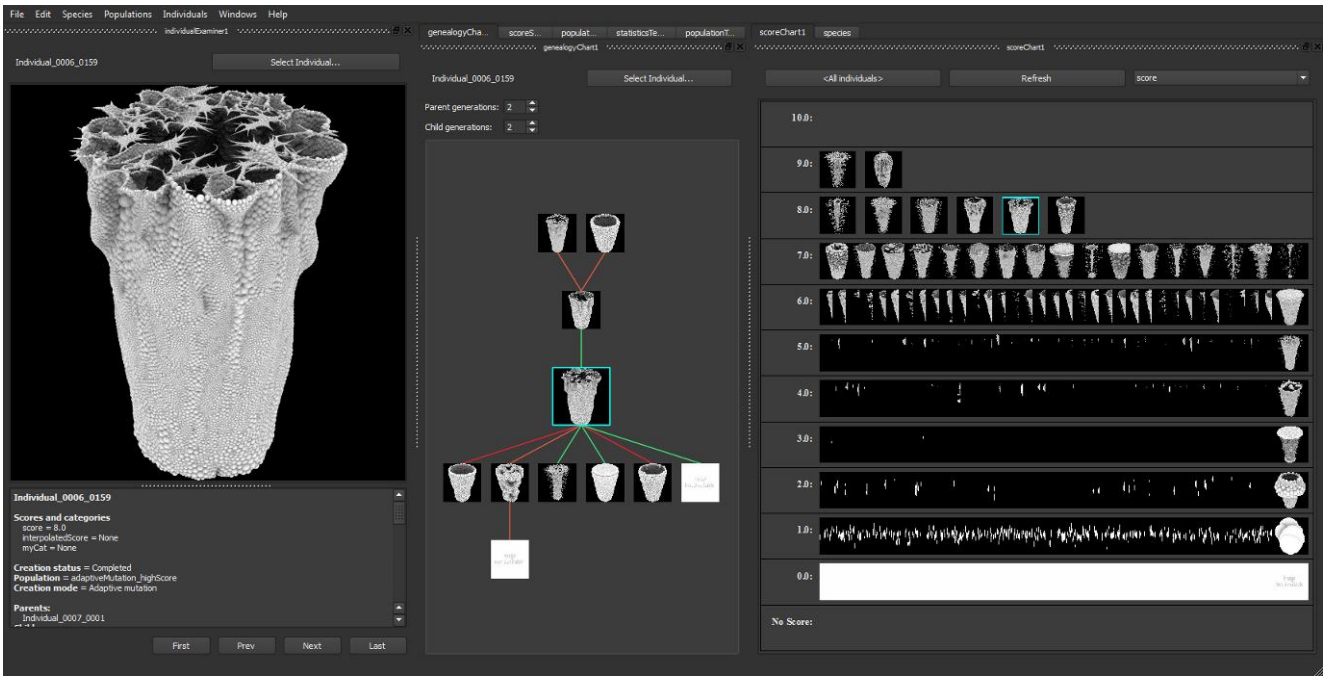

Figure 3: *Species Explorer user interface showing the Individual Examiner, Genealogy View and Score Chart*

This system is designed to explore how complex forms can be emergently created from growth processes using a simplified model of morphogenesis at the level of individual cells. Parameters control a wide number of simulations settings, such as to control how nutrient is generated, which cells are selected to split, the plane of cleavage for a split, and forces between adjacent cells. The system is designed as an extensible framework to explore the effects of different influences, with early versions of the simulation engine having 12 parameters, and more recent versions having over 30.

One of the intuitions from earlier work, such as the Aggregation series, was that the most interesting results often occurs at 'transition zones' between regions of relatively homogeneous behaviour, as can be seen in Figure 1. It is also expected that when dealing with dynamical systems, interesting emergent behaviour will often occur in small transition regions between regularity and chaos. All of these considerations can make the exploration of a multi-dimensional parameter space to find the most interesting complex results particularly challenging.

A system that is based on aesthetic judgements from a human also needs to be tolerant of effects such as inconsistent ratings from the user as they change their opinion about what they consider or do not consider to be interesting.

4. IMPLEMENTATION

Species Explorer has been designed as a general framework to explore any system that is driven by a fixed number of parameter values. It is implemented in Python together with Qt, using the PySide Qt bindings. This has allowed rapid development and experimentation.

It is implemented as a single document interface application using nested tabbed dock widgets. The main components of the user interface are handled using a number of panels, such as the 'Individual Examiner', 'Species Panel' and 'Score Chart', any of which can be docked within tabs. This allows a flexible customizable layout for the user interface.

The current implementation uses Windows, but everything has been written in an operating system agnostic manner that should facilitate support for other operating systems.

4.1 Individuals, Populations and the Species

Species Explorer uses the concepts of 'individuals', 'populations' and the 'species' to help to provide organizational structure.

4.1.1 Individuals

An individual is created for every position sampled in the parameter space. Species Explorer selects the parameter values for the individual, then generates a 'creation script'. This is a Windows batch file or Python script that is executed as a

command line process to run the generative system and create output that the user can review and evaluate.

The generation process is expected to produce at least one image file that can be displayed in the user interface as a representative image for the individual, and a log file that can be parsed to extract additional data such as how long it took to run the generative process, if there were any errors, or any other meta data that should be recorded for that individual.

In the user interface the 'Individuals Examiner' panel can be used to view the representative image for each individual, the parameter values that were used to generate the individual, and additional data such as if the individual has any 'parents' or 'children' through use of evolutionary creation methods.

The user can review each individual and give them score values and place them into categories as described in Section 4.2.

4.1.2 Populations
Populations are used to group together individuals and define which creation method should be used to select the parameters to create new individuals for that population. Examples of creation methods include random selection of parameter values using probability distributions, mutation of the values of existing individuals, or cross-breeding pairs of individuals by choosing parameters that blend between the values from two parents.

Populations can also be used to group together individuals into ‘Selection Pools’ for other purposes. For example, a selection pool could be created for a group of individuals that appear to be particularly interesting so that those individuals can be used as the set of potential parents that will be selected from when using cross-breeding.

4.1.3 The Species
The Species defines the parameters that are needed to generate each individual, as well as the template used to generate the creation scripts.

Each species has a fixed number of named parameters. These parameters can be floating point, integer or boolean values. Numeric values can also be specified as being 'logarithmic' meaning that they are strictly positive valued and that when performing operations such as selecting them from a random distribution those operations should be done using the natural logarithm of the parameter value. This is typically advantageous for values where the ratio between different parameter values is more relevant than absolute values.

In the template creation script tokens with the name of each parameter, such as '<springStrength>', are used. These are replaced by an individual's parameter values when the creation script for that individual is generated. There are also a number of additional tokens, such as '${imageFileStem}' and '${speciesName}', that can be used in the template script to represent values of useful variables that will filled in with the appropriate values.

```
C:\bin\grayScott.exe <Da> <Db> <f> <k> ${imageFileStem}
```

Figure 2: *Example of a simple template creation script*

Since Windows batch files or Python scripts are used for creation scripts, the creation script can describe a series of commands that need to be executed. For example, this can be used to do additional image processing after the main executable for the generative system has completed, or to include more complex control logic as part of the creation process.

4.2 Score sets and Categorization

Score sets are used by the user to rate and categorize individuals. The values assigned to individuals can then be used for purposes such as selecting which individuals should be chosen as parents for cross-breeding, or to estimate the likely score value at new positions in parameter space.

Score values can be integers, floating point numbers, or category values. By default there is a single floating point score set called 'score' which takes values in the range 0 to 10, but the user can modify this and create their own additional score sets.

Categories are a special type of score set that allow each individual to be assigned a value from a set of specified names instead of numeric values. They are typically used to divide individuals into groups that appear to have common properties. For example you could have a category called 'formType' which could take values such as ‘brain’ or ‘reptile skin’ depending on the appearance of the individual.

The value for any score set can also be set to ‘None’. This means that the individual has not been assigned a score, and that individual will be ignored when performing operations such as estimating the score value at new positions in parameter space. By default any newly created individual has all its score set values set to 'None'.
The 'Score Chart' panel in the user interface can be used to assist visualizing score values and assigning scores to individuals. This panel uses thumbnails for each individual, displaying them in horizontal bars for each score value, and allows

assignments of scores by using keyboard short cuts or drag and drop.

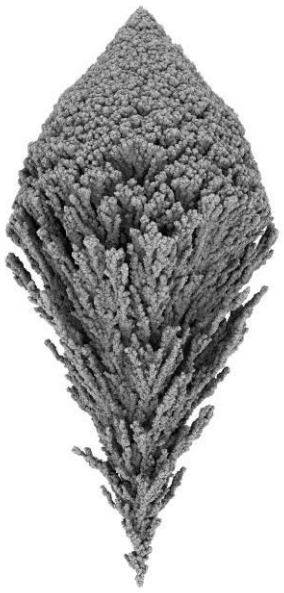

Figure 4: *Image from the Plant-like Cellular Forms series*

4.3 Score expressions and selection tests

The selection of individuals, such as choosing parents to breed from, is done using score expressions and selection tests.

The score expression is a way of giving any individual a fitness score. This can use values set using score sets, but can also use full Python mathematical syntax. This means that more complex fitness values can be expressed than simply using single score values. For example, in addition to using 'score' to give an overall rating to individuals you could have a score set called 'bushiness' to give a rating for how bush-like an object is. You could then use:

```
score * bushiness
```

as a score function to select individuals that have a high value for both 'score' and 'bushiness'.

Category values can also be used in score functions. For example, if you have a category called 'formType' and you want to select objects that have a high value for 'score' and have had the 'formType' set to 'brain' you could use:

```
score * formType_values['brain']
```

The selection test is a boolean test used to specify whether a given individual should be considered or not. Simple selection tests are just lists of individuals or populations. More advanced selection tests can be specified by using Python expressions that return boolean values. For example, to only consider individuals whose 'formType' is 'brain' and whose value for 'bushiness' is greater than 5 you could use the selection test:

```
(formType == 'brain') and (bushiness > 5)
```

4.4 Creation methods

A variety of different methods are provided that can be used to select the parameter values for new individuals. These govern how the software suggests new sample points in the parameter space where the generative system should be run to produce results that the user can then evaluate.

The system is extensible, allowing new creation methods to be easily implemented. The existing creation methods can be divided into 3 types: simple methods, evolutionary methods and machine learning methods.

4.4.1 Simple creation methods
These methods calculate parameter values for new individuals without needing any data from existing individuals. These methods are typically used as a first step to create an initial set of individuals.

Currently three methods are implemented:

- Random Population: parameter values are created randomly using specified probability distributions.
- Fixed Value: individuals are generated using parameter values explicitly specified by the user.
- Wedge Test: parameter values are generated with regular spacing, similar to test chart shown in Figure 1. This is generally only useful for varying a small number of parameter values.

4.4.2 Evolutionary creation methods
These methods involve selection of individuals to act as 'parents', with the parameter values for new child individuals being based on those of the parents.

Currently the following methods are implemented:

- Standard Mutation: for each new individual a single parent is selected, whose parameters are randomly modified according to specified distributions to create the child.
- Adaptive Mutation: similar to Standard Mutation, except the amount of mutation for each parameter is based on the parameter values of the nearest neighbours.
- Cross Blend: for each new individual two parents are selected, with the parameters

for the child being randomly selected blends of the parent values.
- Cross Blend Neighbours: similar to Cross Blend except the second parent is chosen from the closest neighbours of the first.

4.4.3 Machine learning creation methods
These methods use lazy machine learning techniques to estimate the values of score expressions at new positions in the parameter space. New individuals are chosen based on these estimate values using a Monte Carlo method to estimate the score expression at a number of candidate points, and choose one of the candidates with a probability proportional to the estimated values. This means that parameter combinations that are expected to have high values for the score expression will be preferentially selected.

Two methods for estimating values of score expressions are currently implemented:

- K-Nearest Neighbours: values are estimated by finding the nearest neighbours to the sample point that have a valid value for the score expression and averaging them using a weight function based on the distance from the sample point to each of the neighbours.
- Radial Basis Function: values are estimated using a radial basis function to interpolate the values from a given number of neighbours.

When estimating values from a category score set separate floating point valued functions are created for each name value that the category can take, These functions have the value 1 if an individual has been assigned the corresponding name value and 0 if they do not. This allows interpolation methods to be used, with the results interpreted as a measure of the probability that an individual generated at the specified position in parameter space will take that name value. For example, the estimated value could be used to measure the probability that a form generated with a given set of parameter values will be 'reptile skin' like.

4.5 Additional Features

A number of panels are provided in the UI for special purposes such as:

- Genealogy Chart: shows a graphical representation of the parents and children of the currently selected individual.
- Statistics Tests: allows implementation of Python plug-ins to run statistic tests on data such as correlation between parameters and scores. Also allows display of data in graphical form such as histograms or scatter charts.

The system also provides the ability to specify species specific extensions using plug-ins. These allow customizations such as special widgets to modify how individuals are shown in the Individual Examiner, actions to be triggered based on global key-press events, and calculation of additional meta-data that will be held for each individual after the generation process for that individual has completed.

5. RESULTS

5.1 Different search phases

One of the main strengths of the system is to provide a variety of different techniques for generating parameter values, each of which may be appropriate depending on the user's intent.

5.1.1 Initial exploration
During the first phase of exploring a new generative system, when the user typically does not have a clear idea of what regions of the parameter space may be interesting, simple creation methods such as random parameter generation as described in subsection 4.4.1 are generally appropriate.

In general, the author has found that simple random parameter generation is useful for this phase. If the number of parameters is small (typically 3 or less), then a Wedge Test that samples all the parameters with regular spacing may be used.

5.1.2 Secondary exploration
Once the user has got some initial results from random sampling the next phase is generally rating those results using score sets and using creation methods that preferentially create more samples in areas that are likely to be productive without overly narrowing the search.

During this phase machine learning methods that can estimate score functions at any position in parameter space appear to work well. As long as the estimated score is better than random, the search should get directed towards more productive regions.

5.1.3 Refined focus
If the user finds some particularly interesting individuals that they want to refine further then other methods are generally preferable. In particular the intent is to focus the search within a limited range of parameter values around interesting individuals rather than search a wide region of the parameter space.

During this phase the author generally creates a selection pool of the individuals that he is particularly interested in and uses adaptive mutation or cross-breeding between neighbours with the first parent chosen from the selection pool.

5.1.4 Looking for novelty
In contrast to the previous section, if the user's intent is to find new interesting regions of the parameter space then other methods may be more appropriate.

Since cross breeding and mutation methods generally bias the search towards regions that have already been highly sampled, the author typically utilizes machine learning methods when looking for novel results.

During this phase carefully constructed score expressions can be beneficial. For example, it may be useful to try to focus the search on regions where the score for individuals is significantly different from what would be expected using estimation. One way of expressing this is to calculate the difference between the actual score given to each individual and the estimated score that would have been given based on all the other individuals, and then using a score expression such as:

```
abs(estimatedScore – score)
```

5.2 Machine learning methods

While searching for appropriate machine learning methods, the author has run tests to try to evaluate which methods appear to work best with the type of data created. In particular we want methods that can estimate arbitrary score expressions for new parameter values, do not require the user to fine tune values or specify features, can make use of all the data the user has supplied as soon as the user has rated any individuals, and are not adversely affected by non-uniform distribution of samples. This has led the author towards lazy machine learning techniques such as k-nearest neighbours and interpolation using radial basis functions.

To help determine which methods produce the most useful estimates, the author has tested a number of data sets from his work on the Cellular Forms and Hybrid Forms series using variations on k-nearest neighbours and radial basis functions. Presented here are the results from 3 data sets. To provide a measure of how well each method estimates the score the tests use the root mean square error value of the difference between the score the author assigned to each individual and the score that would be estimated by each method using the set of previously generated individuals. Data was only used from the previously generated individuals to make sure that the tests account for early stages in the search when there are only a small number of data points available to create the estimates.

When using k-nearest neighbours, weight functions of the form $1/d^n$, where d is the distance in parameter space between two points, were tested with various exponent values n. For the radial basis functions all the standard interpolation basis functions provided by the Python SciPy library were tested.

To provide a baseline comparison of whether these methods are providing better than random estimates, the RMS error was also calculated using the average score of all previous individuals as the score estimator.

Table 1: *Data sets used to evaluated score estimation methods*

Data set	No. of individuals	No. of species parameters
Set 1 (Cellular Forms)	*1776*	*12*
Set 2 (Plant-like Forms)	*2200*	*19*
Set 3 (Hybrid Forms)	*6926*	*28*

Table 2: *RMS error values for different score estimation methods. The best (smallest) results for each data set are highlighted*

Score estimation method	**RMS score error**		
	Set 1	*Set 2*	*Set 3*
Average (baseline)	*3.268*	*2.141*	*2.120*
kNN weight n=0	*2.946*	*1.944*	*1.717*
kNN weight n=1	*2.904*	*1.939*	*1.711*
kNN weight n=2	*2.869*	*1.935*	*1.704*
kNN weight n=3	*2.868*	*1.931*	*1.697*
kNN weight n=4	*2.886*	*1.929*	*1.691*
RBF Linear	***2.779***	***1.910***	***1.404***
RBF Inverse	*7.663*	*1.963*	*1.421*
RBF Gaussian	*10.26*	*1.989*	*1.537*
RBF Multiquadratic	*11.07*	*2.182*	*1.693*
RBF Thin-plate	*15.14*	*166.5*	*172.8*
RBF Cubic	*95.13*	*3117*	*527.3*
RBF Quintic	*8107*	*337.6*	*2587*

As can be seen, estimation using a linear radial basis function consistently gave the best (lowest) RMS score error in all cases. Different basis functions give very different results. In particular, thin-plate, cubic and quintic basis functions were the worst performing, with RMS errors that were worse than using a simple average as the estimator, and often significantly higher than the maximum score range up to 10. This is probably

due to these methods over-fitting the data and extrapolating. This may be exacerbated by the nature of the data, with non-uniform sampling and potentially inconsistent assignment of score values by the user when making subjective assessments.

The RMS score errors using k-nearest neighbours are all better than the baseline method using the average score, with an inverse weight exponent of 4 giving the best from test data, though the results for all the different inverse weight exponents are quite similar.

Using linear radial basis functions appear to generally give the lowest RMS error scores, but k-nearest neighbours can be computationally faster to calculate. The default options in Species Explorer are set to offer both radial basis functions using linear or k-nearest neighbours using an inverse weight exponent of 4.0.

6. FUTURE WORK

Species Explorer is very much a work in progress, with the author actively adding new features as he gains experience using it.

There are a number of areas for future work, in particular to explore alternative methods of estimating score functions for new parameters. Since they are based on distance measures and proximity to neighbours, the current use of lazy machine learning methods is likely to be less effective if significantly more parameters are used.

7. CONCLUSION

Species Explorer was born out of necessity when dealing with generative systems with more than 3 parameters. Allowing a number of different methods to determine the parameter values to next sample appears to work well, with different methods being appropriate depending on the intent of the user.

In many ways the proof of a system like this is whether it produces good results. As the author has been the only user to date this is somewhat difficult to evaluate, but all the exhibited work from his Cellular Forms and Hybrid Forms series has been generated using samples suggested by Species Explorer. With ten or more parameters, testing the effects of all parameter combinations was impossible and complex inter-dependence between parameters meant that varying one parameter at a time was not creating the desired results. The author felt he has a rich simulation engine but was frustrated by how difficult it was to explore its possibilities.

With Species Explorer the author feels that he has been able to much more effectively explore the space of possibilities, and does not believe that he would have been able to create most of his recent results without it acting as a loyal but honest assistant in the creative process.

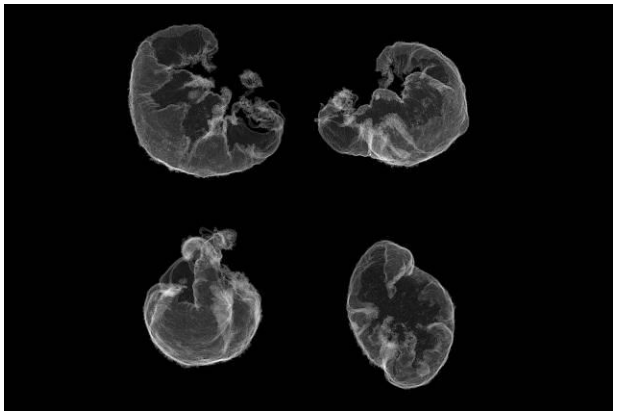

Figure 5: *Image from the Hybrid Forms series*

8. REFERENCES

Conway, J. (1970) The game of life. *Scientific American*, *223*(4), p.4.

Turing, A.M. (1952) The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *237*(641), pp.37–72.

Dawkins, R. (1986). *The blind watchmaker: Why the evidence of evolution reveals a universe without design*. W. W. Norton & Company.

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

Bentley, P. (1999) *Evolutionary design by computers*. Morgan Kaufmann.

Hanna, S. (2007) Inductive machine learning of optimal modular structures: Estimating solutions using support vector machines. *AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, *21*(4), pp.351–366.

Umetani, N., Igarashi, T., and Mitra, N.J. (2012) Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, *31*(4), pp.86:1–86:11.

Oberhauser, M., Sartorius, S., Gmeiner, T. and Shea, K. (2015) Computational Design Synthesis of Aircraft Configurations with Shape Grammars. In *Design Computing and Cognition'14.* Springer International Publishing, pp. 21–39.

Lomas, A. (2014) Cellular Forms: an Artistic Exploration of Morphogenesis. *AISB-50*, London, UK, 1–4 April 2014. Goldsmiths, University of London.