Co-Evolving Line Drawings with Hierarchical Evolution

Darwin Vickers, Jacob Soderlund and Alan Blair

School of Computer Science and Engineering
University of New South Wales
Sydney, 2052 Australia
{z5018493,jnzs770,blair}@cse.unsw.edu.au

Abstract. We use an approach inspired by biological coevolution to generate complex line drawings without human guidance. Artificial artists and critics work against each other in an iterative competitive framework, forcing each to become increasingly sophisticated to outplay the other. Both the artists and critics are implemented in HERCL, a framework combining linear and stack-based Genetic Programming, which is well suited to coevolution because the number of competing agents is kept small while still preserving diversity. The aesthetic quality of the resulting images lies in the ability of the evolved HERCL programs, making judicious use of register adjustments and loops, to produce repeated substructures with subtle variations, in the spirit of low-complexity art.

Keywords: artist-critic coevolution, artificial creativity

1 Introduction

Several recent papers have explored the development of aesthetically pleasing images using an evolutionary approach [6–9, 11]. In most cases, human interaction is required to guide evolution towards pleasing images, However, Machado et al. [9] use an experimental approach inspired by coevolution in nature, which allows novel imagery to be generated without human interaction. The core of the coevolution process is the adversarial relationship between an *artist* and a *critic*. The aim of the critic is to distinguish *real* art from artificial art (produced by the coevolving artist). The aim of the artist is to produce images that the critic will believe to be real.

At the beginning of the process, a set of critics are trained on a predetermined set of *real* and *fake* images. Then, a set of artists are evolved to produce art which *fools* the previously-evolved critics. The images produced by the artists are then added to the fake dataset that is used to train critics in the next generation, and the process is iterated indefinitely. Both the artist and the critic must increase in sophistication at each generation in order to surpass the adversary. The hope is that the ongoing increase in sophistication will produce imagery that is appealing, or at least interesting, from a human perspective.

Table 1: HERCL Commands

Input and Output

`i` fetch INPUT to input buffer
`s` SCAN item from input buffer to stack
`w` WRITE item from stack to output buffer
`o` flush OUTPUT buffer

Registers and Memory

`<` GET value from register
`>` PUT value into register
`^` INCREMENT register
`v` DECREMENT register
`{` LOAD from memory location
`}` STORE to memory location

Jump, Test, Branch and Logic

`j` JUMP to specified cell (subroutine)
`|` BAR line (RETURN on `.|` HALT on `8|`)
`=` register is EQUAL to top of stack
`g` register is GREATER than top of stack
`:` if TRUE, branch FORWARD
`;` if TRUE, branch BACK
`&` logical AND
`/` logical OR
`~` logical NOT

Stack Manipulation and Arithmetic

`#` PUSH new item to stack ..... $\mapsto$ x
`!` POP top item from stack ..... $x \mapsto$ .....
`c` COPY top item on stack ..... $x \mapsto$ .....x, x
`x` SWAP top two items ... $y, x \mapsto$ ... x, y
`y` ROTATE top three items $z, y, x \mapsto x, z, y$
`-` NEGATE top item ..... $x \mapsto$ .....$(-x)$
`+` ADD top two items ... $y, x \mapsto$ ...$(y+x)$
`*` MULTIPLY top two items ... $y, x \mapsto$ ...$(y * x)$

Mathematical Functions

`r` RECIPROCAL .. $x \rightarrow$ .. $1/x$
`q` SQUARE ROOT .. $x \rightarrow$ .. $\sqrt{x}$
`e` EXPONENTIAL .. $x \mapsto$ .. e^x
`n` (natural) LOGARITHM .. $x \mapsto$ .. $\log_e(x)$
`a` ARCSINE .. $x \mapsto$ .. $\sin^{-1}(x)$
`h` TANH .. $x \mapsto$ .. $\tanh(x)$
`z` ROUND to nearest integer
`?` push RANDOM value to stack

Double-Item Functions

`%` DIVIDE/MODULO .. $y, x \mapsto$ .. $(y/x), (y \bmod x)$
`t` TRIG functions .. $\theta, r \mapsto$ .. $r \sin\theta, r \cos\theta$
`p` POLAR coords .. $y, x \mapsto$ .. atan2(y,x),$\sqrt{x^2+y^2}$

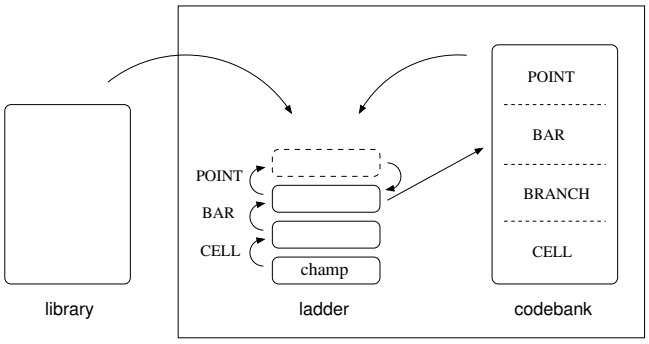

Fig. 1: Hierarchical evolutionary re-combination. If the top agent on the ladder becomes fitter than the one below it, the top agent will move down to replace the lower agent (which is transferred to the codebank). If the top agent exceeds its maximum number of allowable offspring without ever becoming fitter than the one below it, the top agent is removed from the ladder (and transferred to the codebank).

Table 2: Line Drawing Commands

0	TOGGLE	lift pen on/off page
1	MOVE x	move pen forward by x pixels ($0 \leq x \leq 15$)
2	TURN x	turn x degrees clockwise
3	SIZE p	set pen radius to p pixels ($1 \leq p \leq 4$)
4	COLOUR v	set greyscale value [in greyscale mode]
	COLOUR $l\ h\ s$	set colour in HSV colour space [in full-colour mode]

In previous approaches, images were typically generated by artists using pixel-based methods, where a shade or colour is assigned to each pixel in the image [7], often based on its x and y co-ordinates, using either a tree-based Genetic Program [9, 8] or a neural network [11]. The critic used either a tree-based GP [8] or a neural network [9] for classification, based on certain statistical features of the image.

In the present work, we adapt this coevolution approach to the framework of hierarchical evolutionary re-combination (HERCL) as introduced in [2], which combines features from linear GP and stack-based GP. Each agent (artist or critic) is a program written in a simple imperative language with instructions for manipulating a stack, registers and memory. The full list of HERCL commands is given in [2].

HERCL does not use a population as such, but instead maintains a stack or *ladder* of candidate solutions (agents), and a *codebank* of potential mates. At each step of the algorithm, the agent at the top rung of the ladder is either mutated or crossed over with a randomly chosen agent from the codebank, or from an external library. Crossovers and mutations are classified into different levels according to what portion of code is modified. A large crossover at the lowest rung of the ladder is followed up by a series of progressively smaller crossovers and mutations at higher rungs, concentrated in the vicinity of the large crossover (see [2] for further details).

In the present work, both the artists and the critics are HERCL programs. The artist's output is interpreted as a sequence of commands in a simple drawing environment. The critic takes as input a set of features computed from the image, and returns a number between 0 and 1 representing its confidence that the image is real.

The advantages of using the HERCL framework in this context are as follows:

(a) It enables the artist to work at the level of line drawing rather than per-pixel manipulation, thus allowing the exploration of a different artistic modality which is arguably closer to the way humans draw with pen and paper.

(b) The functionality of the artist is extended with programming constructs such as loops, stack manipulation and incrementing or decrementing of registers, as well as various basic math functions.

(c) The ladder and codebank arrangement of the HERCL framework keeps the number of competing artists and critics relatively small, thus allowing computationally efficient coevolution.

The use of a stroke-based model for drawing is similar to Simon Colton's system The Painting Fool [4], whose output is more visually complex. Colton's software aims to produce a painterly rendering of a single image, whereas we are more interested in creating a system capable of generating novel, non-representational imagery based on a collection of images.

2 Methodology

Each artist is a HERCL program, which when executed outputs a sequence of messages that are interpreted as commands in a simple graphics environment. This environment gives the artist control over a virtual pen, which it can rotate and move to draw lines. The artists take no input and behave deterministically, and therefore each artist produces only a single image.

Each message is a sequence of integers; the first integer (modulo 5) specifies the command, and the subsequent integers specify parameters as appropriate (see Table 2). The artist is limited to a maximum of 900 commands, but it can also halt of its own accord before this point.

Typically the artist is allowed to use any instruction in the HERCL syntax, but in some experiments we disallowed the use of the branch-back instruction, preventing the artist from implementing loops.

The critics are also HERCL programs, which take as input a set of features computed from an image, and are required to output a single number between 0 and 1, similar to the classification tasks of [3]. The set of features extracted from the image is primarily based on those used by Datta et al. [5] in their work on computationally assessing photograph quality, as well as some from [9]. We also add some features based on corner detection as this pertains specifically to the line-based drawing method used by our system.

The full list of features is shown in Table 3. For colour images, certain properties are calculated independently across the Hue, Saturation and Value channels, resulting in three features. In greyscale mode, the image effectively consists only of a Value channel, and so the features with superscript H or S are not provided to the critic. All features except N_P and N_C are scaled to within $[0, 1]$. Corner weight and number are computed using the Harris & Stephens corner detection algorithm, as implemented in OpenCV 2.4. In total, there are 19 features in greyscale mode and 33 in full-colour mode.

When evolving an artist, its fitness is determined by how well it fools the set of critics. The image produced by an artist is first converted into a feature vector, and the critics each give a score based on this vector. The cost function for the artist is a weighted sum of critics from all previous generations, with older critics contributing less. For an artist in generation n,

$$\text{cost} = 1 - \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{n-i-1} c_i$$

Table 3: Image Features

Feature	Abbreviation	Source
Mean	M^H, M^S, M^V	[5], [9]
Standard deviation	S^H, S^S, S^V	[5], [9]
Greyscale entropy	H	[9]
Mean edge weight	M_E	[9]
Standard deviation of edge weight	S_E	[9]
Number of homogenous patches	N_P	[5]
Mean of largest patch	P_1^H, P_1^S, P_1^V	[5]
Mean of 2nd-largest patch	P_2^H, P_2^S, P_2^V	[5]
Mean of 3rd-largest patch	P_3^H, P_3^S, P_3^V	[5]
Mean of 4th-largest patch	P_4^H, P_4^S, P_4^V	[5]
Mean of 5th-largest patch	P_5^H, P_5^S, P_5^V	[5]
Size of largest patch	A_1	[5]
Size of 2nd-largest patch	A_2	[5]
Size of 3rd-largest patch	A_3	[5]
Size of 4th-largest patch	A_4	[5]
Size of 5th-largest patch	A_5	[5]
Convexity factor	C	[5]
Mean corner weight	M_C	-
Number of corners	N_C	-

where c_i is the average score across all critics in generation i. Artists are considered successful when they have achieved a cost below 0.1 . The main reason for using critics from all previous generations is that sometimes the most recent critics will be good enough that the artist is initially unable to produce an image which gets a non-zero score. This leaves the artist with a flat fitness landscape and prevents it from evolving successfully. Including critics from previous generations helps the artists to improve, particularly in the early generations, and could perhaps also help prevent suboptimal cycles of forgetting and re-learning in the coevolutionary dynamics [1].

Fig. 2: One example image from each dataset: Chinese characters and coloured circles.

A library is maintained of the successful artist code from all previous generations. Code from this library is made available for crossovers and mutations when evolving the next generation of artists. This enables the artists to evolve by adapting and re-using code from earlier artists, and encourages them to build on the artistic "styles" developed in previous generations. However, this may also limit diversity by encouraging similar code throughout the run.

Critics are evolved against a labeled dataset consisting of feature vectors computed from the real and fake image datasets. The target value is 1 for real images and 0 for fake images. The cost for the critic is the cross-entropy function:

$$\text{cost} = -t \log(z) - (1 - t) \log(1 - z)$$

where t is the target label (0 or 1) and z is the value produced by the critic. Any critic producing a value ≤ 0 for a real image or ≥ 1 for a fake image receives a *penalty* and is heavily punished. This, combined with the logarithmic divergence in the cost function, strongly encourages critics to produce values inside the interval $(0, 1)$ rather than at the extremes. This makes it easier for the artists to evolve because the fitness landscape has a continuous gradient.

As in [9], the real and fake datasets are given equal weight, so that the relative sizes of the two sets do not affect the critic's evolution. Critics are accepted when they achieve a cost below 0.1 .

3 Experiments

We focus primarily on the results of two runs of the coevolution process, using a starting set of images of Chinese characters. We call the two runs **Loops** and **No Loops**. The difference between them is that for the latter, we disabled the branch-back instruction, preventing loop structures from appearing in the artist code. Both these runs operate in greyscale mode.

The starting dataset consists of 290 greyscale images of Chinese characters from Wiktionary, each 80×80. There is no particular reason for using this dataset other than that it contains images that could be feasibly generated within our drawing framework.

We also conducted one additional run, **Colour**, to demonstrate the colour-mode capability. This run used a dataset of 67 colour images taken from a Google

Images search for "circle", with search settings restricting size to 128×128 and specifying 'Full colour'.

For each run, there were three critics and ten artists evolved independently in each generation (each with its own ladder and codebank).

4 Results and Discussion

A selection of generated images from each run is shown in Figures 3 to 5. Aesthetic qualities of the images are difficult to quantitatively assess, but it is clear that the system is capable of generating quite a diverse range of complex imagery with no human guidance.

It is clear from the results, and Figure 3 in particular, that the complexity of the images is increasing across phases. This demonstrates that the adversarial artist-critic relationship is successful in encouraging development of complexity.

The difference between the images in Figure 4 and Figure 3 is notable. We see that allowing loops results in highly structured patterns in the images, whereas disallowing loops results in images with a more freehand appearance.

With loops turned on, there are many cases where the evolved code is quite short but manages to generate surprisingly elaborate images. For example, the left image in the 2nd bottom row of Figure 4 was generated by this code:

```
[<wx<*23.#-!cw8v{.v<wwcv<wwow.v<*vwo;:]
```

The possibility of creating complex images from short programs draws parallels with Jürgen Schmidhuber's theory of low-complexity art [10]. What makes these images particularly interesting (and perhaps aesthetically pleasing) is that they often contain repeated structures which are similar but not quite identical. The evolved HERCL programs make use of loops and register adjustments to introduce subtle differences in these substructures, thus mimicking certain developmental processes in the natural world. This phenomenon can be compared with the pixel-based approach of [7], where local agents following evolved rules were used to create an overall "natural-looking" image.

Greyscale entropy is the feature most commonly used by the critics (see Figures 6 to 8). Other important features include the mean and standard deviation of the overall image (M^V, S^V) and the edge weight (M_E, S_E), the convexity (C) and the size and mean value of the 2nd largest patch (A_2, P_2^V). For the Colour task, which was trained on "circle" images, the mean corner weight and mean of the largest and 3rd largest patch in the H and S channels are also used.

5 Conclusion and Future Work

We have successfully adapted the coevolutionary art paradigm to a natural line-drawing environment, with HERCL programs acting as both artist and critic. Discrimination based on statistical features of the image, combined with the

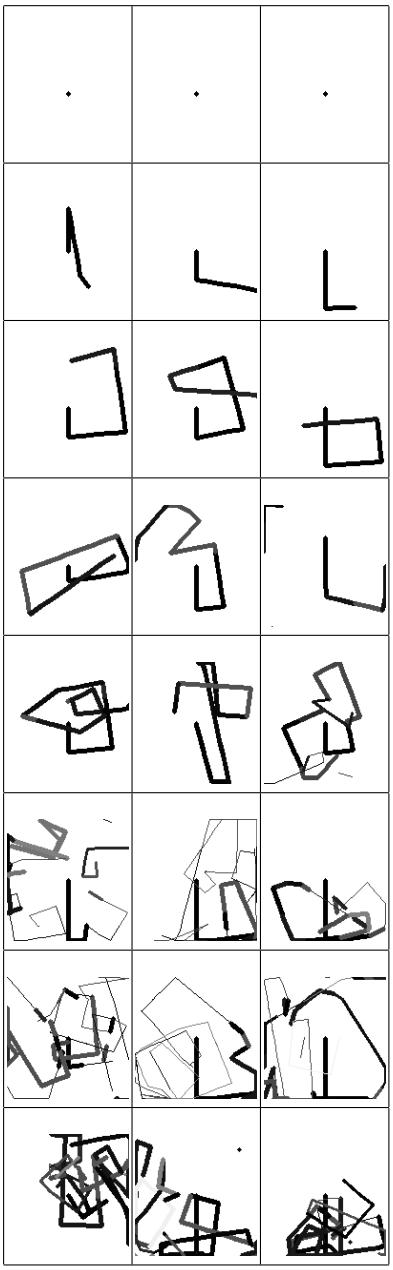

Fig. 3: Three images each from generations 1-10 of the No Loops run.

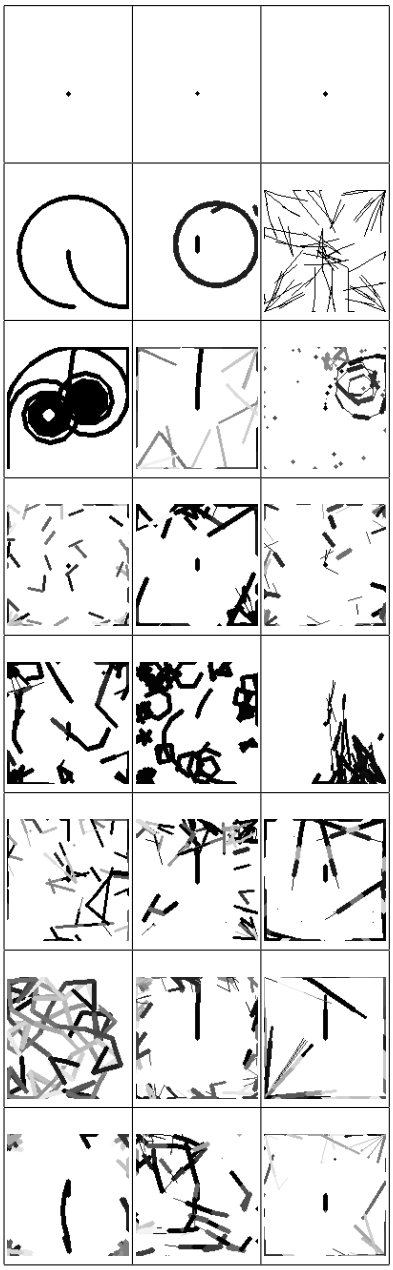

Fig. 4: Three images each from generations 1-10 of the Loops run.

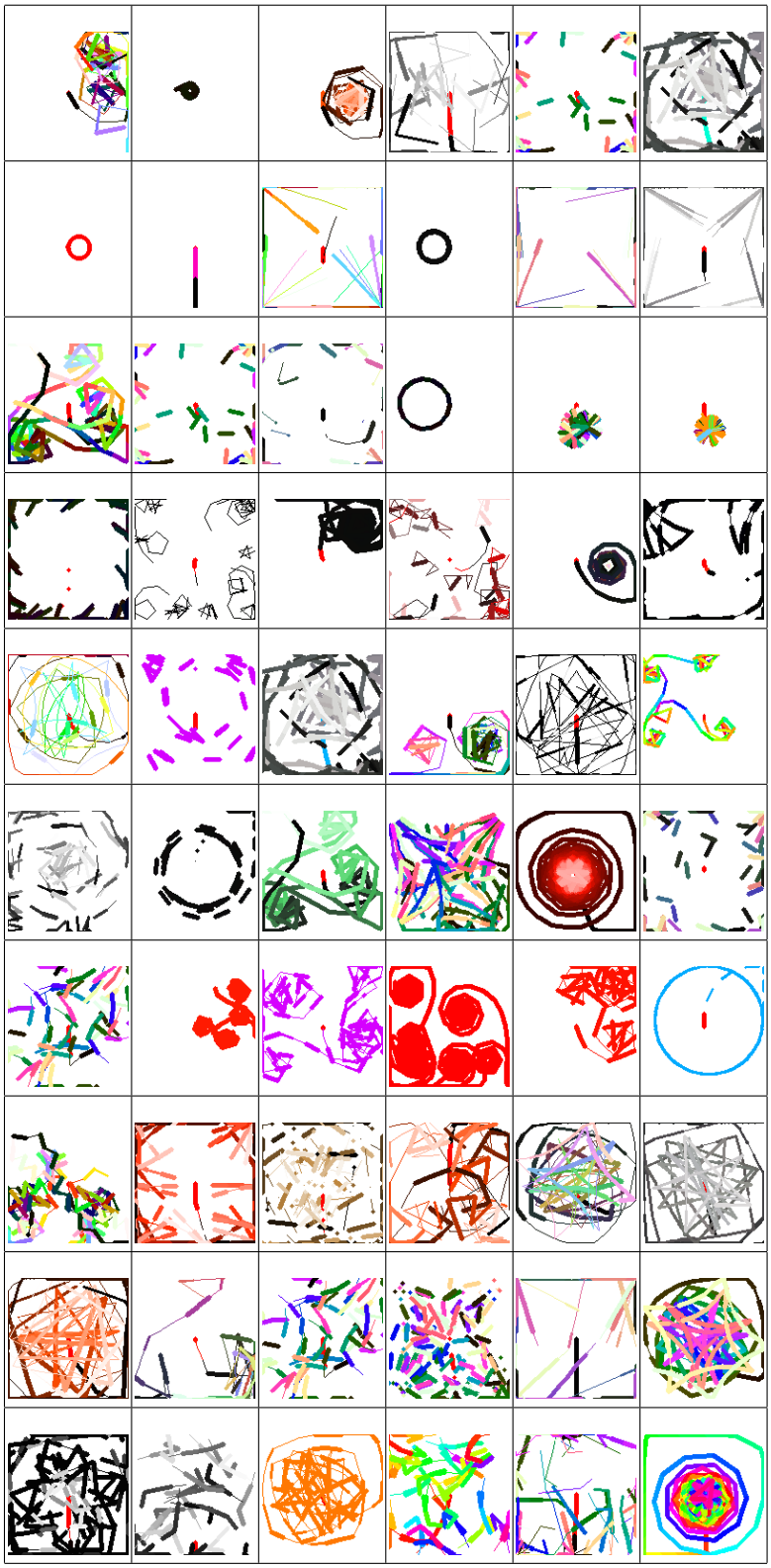

Fig. 5: Six images each from generations 3, 5, 7, 8, 9, 11, 13, 15, 17, 18 of the Colour task.

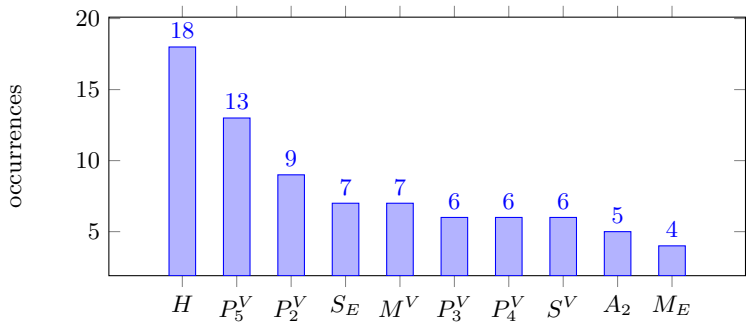

Fig. 6: Image features most used by critics for the No Loops run.

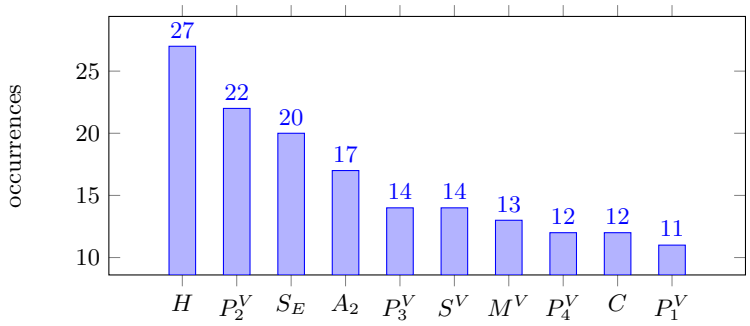

Fig. 7: Image features most used by critics for the Loops run.

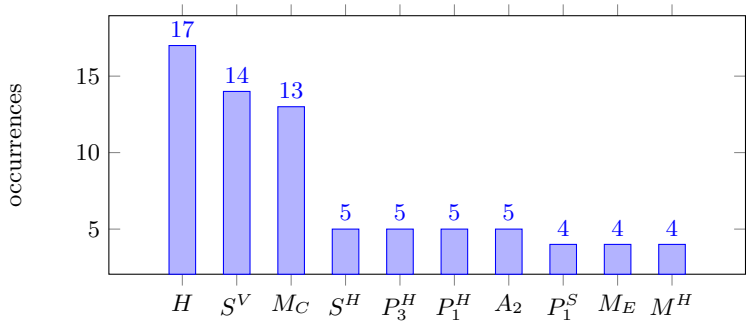

Fig. 8: Image features most used by critics for the Colour run.

inherent preference for shorter programs, are enough to drive complexity and produce aesthetically pleasing images.

However, there are clearly certain local and global properties of the images which are not being captured by these statistical features. In future work, we plan to explore the use of deep convolutional neural networks in the role of the critic. Another avenue of investigation would be the use of multi-cell HERCL programs (which allow jumping to subroutines), or additional line drawing commands for moving to a specified (or previously stored) location on the canvas, to see whether the paradigm can be scaled up to larger and more complex images.

References

1. Axelrod, R., The evolution of cooperation, Basic Books, New York (1984)
2. Blair, A., Learning the Caesar and Vigenere Cipher by Hierarchical Evolutionary Re-Combination, Congress on Evolutionary Computation, 605–612 (2013)
3. Blair, A., Transgenic Evolution for Classification Tasks with HERCL, Artificial Life and Computational Intelligence, Springer, 185-195 (2015)
4. Colton, C. Stroke Matching for Paint Dances, 2010 International Symposium on Computational Aesthetics in Graphics, Visualization and Imaging.
5. Datta, R., Joshi, D., Li, J. & Wang, J.Z., Studying Aesthetics in Photographics Images Using a Computational Approach, Proc. European Conference on Computer Vision, 288-301 (2006)
6. Galanter, P., Computational aesthetic evaluation: Past and future, Computers and Creativity, Springer, 255-293 (2012)
7. Kowaliw, T., Dorin, A. & McCormack, J., 2012, "Promoting creative design in interactive evolutionary computation", IEEE Transactions on Evolutionary Computation 16(4), 523 (2012)
8. Li, Y. & Hu, C., Aesthetic learning in an interactive evolutionary art system, Applications of Evolutionary Computation, Springer, 301-310 (2010)
9. Machado, P., Romero, J. & Manaris, B., Experiments in computational aesthetics: An Iterative Approach to Stylistic Change in Evolutionary Art, The art of artificial evolution, Springer, 381-415 (2008)
10. Schmidhuber, J., Low-complexity Art, Leonardo, Journal of the International Society for the Arts, Sciences, and Technology 30(2), 97-103 (1997)
11. Secretan, J., Beato, N., D'Ambrosio, D.B., Rodriguez, A., Campbell, A., Folsom-Kovarik, J.T. & Stanley, K.O., Picbreeder: A case study in collaborative evolutionary exploration of design space. Evolutionary Computation, 19(3), 373-403 (2011)