Evolving a Library of Artistic Scene Descriptors

Simon Colton

Computational Creativity Group, Dept. of Computing, Imperial College, London
ccg.doc.ic.ac.uk

Abstract. We describe the building of a library of 10,000 distinct abstract art images, and how these can be interpreted as describing the placement of objects in a scene for generative painting projects. Building the library to contain only markedly distinct images necessitated a machine learning approach, whereby two decision trees were derived to predict visual similarity in pairs of images. The first tree uses genotypical information to predict before image generation whether two images will be too similar. The second tree uses phenotypical information, namely how pairs of images differ when segmented using various distance thresholds. Taken together, the trees are highly effective at quickly predicting when two images are similar, and we used this in an evolutionary search where non-unique individuals are pruned, to build up the library. We show how the pruning approach can be used alongside a fitness function to increase the yield of images with certain properties, such as low/high colour variety, symmetry and contrast.

1 Introduction and Motivation

We are building a software system called The Painting Fool, which we hope will one day be taken seriously a creative artist in its own right [5]. We believe that for software to be seen as creative, it must first exhibit behaviours deemed as skillful, appreciative and imaginative [4]. To implement possibly imaginative behaviours, we have looked at automatic scene generation, with the scenes ultimately being rendered in a painterly fashion. This is similar to Cohen's AARON system, which designs and then renders figurative scenes involving people, plants and furniture [16]. To improve the intelligence and flexibility of the scene generation in The Painting Fool, it has been enabled to employ various generative approaches, including context free design grammars [17], constraint solving [3] and evolutionary methods [2]. These address utilitarian aspects of scene generation, namely producing content (context free design grammars) and placing content (constraint solving, evolutionary methods) in the scene. However, none of the methods consider the decorative aspects of scene generation, i.e., that – if required – the scene should be generated to have aesthetically appealing properties, and we address this shortcoming here.

Evolutionary art approaches can produce highly appealing artworks, either through user-centric approaches where an artist acts as the fitness function; semi-automated approaches, where aesthetic values are learned [9], [12]; or fully automated approaches, where aesthetic measures such as those defined in [8] are used in fitness functions. With this in mind, to supplement The Painting Fool's scene generation abilities, we used an evolutionary approach to build up a library of 10,000 distinct abstract art pieces, each of which can be interpreted as scene descriptors that dictate the colouring of objects in the scene.

To start to build the library, we undertook some evolutionary image generation sessions, as described in section 2, but these converged too quickly on images of a very similar visual nature. To combat this, we used a machine learning approach to derive decision trees based on phenotype and genotype information, which can predict whether two images have high structural similarity visually, as described in section 3. This was embedded into a new evolutionary search strategy which prunes new individuals which are not visually unique, and the results were considerable improved. In section 4, we show how the evolutionary search can still be driven by a fitness function favouring aesthetic values such as symmetry, colour variety and contrast. We describe some aspects of the library built up in this way, and how The Painting Fool can select images with certain properties from the library, and use these to place object in scenes. In section 5, we present the first artistic application arising from this method. In section 6, we place this work in context and describe future work involving the automatic generation of meaningful, aesthetically pleasing, artworks.

2 Preliminary Sessions

The aim of building up a large library of visually distinct, aesthetically pleasing, abstract art images suggests a straightforward random generation approach over a space containing large numbers of visually distinct pieces. We have previously worked with a particle based approach, described in [6], which generates images with great visual distinctiveness, and hence we chose this method. The genotypes of the images describe (i) three HSB values for the background colour (ii) six *init* functions dictating the initial (x,y) placement and HSBA values of the particles, and (iii) six *update* functions dictating how the particles move and change colour over a series of time steps. Images are generated by a certain number of particles being placed and then altered in location and colour over a certain number of timesteps. For each particle, a line is drawn from the previous position to the new position in the new colour, and the whole image is blurred at each timestep (with details given in [6]). We experimented with 50 particles and 50 timesteps on a 200×200 pixel image, and found the resulting images to vary sufficiently. Sample images are given in figures 1, 2, 5 and 7 below.

We randomly generated 1,250 images and inspected the results. While there was visual variety, we wondered whether it would be possible to use an evolutionary approach to produce a higher yield of images with certain visual properties. In particular, for the artistic purposes described below, it would be advantageous for the library to contain large numbers of images with low/high colour variety, low/high contrast and low/high symmetry. We implemented a method to estimate the colour variety of an image by scaling the image to a 24×24 pixel version via colour averaging, mapping each pixel to a colour palette of 100 named colours such as Sienna, RoyalBlue, LightSeaGreen, etc, and counting the number of colours exhibited. This value is normalised by dividing by the number of pixels, i.e., 576. To assess contrast, we calculate the average RGB distance between each pixel in the 200×200 pixel image and its neighbours, and normalise by dividing by $\sqrt{256^3}$. For symmetry, we return to the 24×24 pixel

version and calculate the normalised average RGB distance between this and 7 transformations of it (three 90^o rotations plus four reflections).

We implemented a fitness-proportionate (FP) evolutionary search with a mutation rate of 0.01 and five-point crossover of either init or update functions, i.e., children are produced by swapping five function trees in the parents. We ran six sessions where populations of 50 individuals were evolved over 25 generations. We denote the six sessions as: $FP_{v+}, FP_{v-}, FP_{c+}, FP_{c-}, FP_{s+}$ and FP_{s-} and they differed in the fitness function used. For FP_{v+}, the fitness function was the normalised colour variety measure described above, but for FP_{v-}, this value was taken from 1 to give higher fitness to individuals exhibiting fewer colours. Similarly, FP_{c+} and FP_{c-} used the contrast function, with FP_{s+} and FP_{s-} using the symmetry function. To hopefully encourage a wide search of the space, 5 out of the 50 children in each generation were produced randomly. Moreover, a simple check for genetic equality and/or phenotypic equality on each child was undertaken, so that exact copies of previously seen individuals were pruned.

We also implemented a cluster-based (CB) evolutionary search, to hopefully further encourage a broad search of the space. In this method, the individuals in a generation are partitioned into 15 clusters using a K-means++ approach [1], with the values from the RGB-histogram of images being fed into the clustering process. Each cluster contains one or more images of roughly similar colour, and one member of each cluster was chosen for reproduction. Over six sessions denoted $CB_{v+}, CB_{v-}, CB_{c+}, CB_{c-}, CB_{s+}$ and CB_{s-}, the cluster member chosen was the highest according to the respective fitness function. For instance, in the CB_{s-} session, fifteen individuals were chosen as the ones with least symmetry in each cluster. From the fifteen, pairs were randomly chosen for reproduction. As before, we ran sessions with population size 50 for 25 generations, with five random children in each generation, and equality pruning as above.

In both sets of sessions, we found a very high yield of individuals with high value with respect to the chosen fitness function. However, on inspection of the images produced, we found that this had been achieved at the expense of visual variety, and our efforts to encourage a search over a broad space had largely failed. Some triples of very similar examples from the final generation of three FP and three CB sessions are given in figure 1. We see that while the images in each triple are clearly different, they contain the same visual structure, whether it be a clearly identified shape, colour, or texture. We found that as the populations evolved, the number of such structural repetitions increased, and each session yielded a disappointingly low number of visually different pieces. We had planned to experiment with various evolutionary setups (especially increasing the mutation rate), to see which ones increased both the visual variety and the fitness of individuals. However, given that in some generations we were getting as many as 30 visually similar copies of the same image (as quantified later in section 4), we felt that this might be largely wasted effort, and what was needed instead was a check on structural similarity, that could be used to strengthen the pruning aspect of the search, i.e., so that only visually unique individuals are kept. We describe how we derived such a check in the next section.

3 Learning Structural Similarity Predictors

In light of the disappointingly similar images resulting from the first sessions, we decided to enable the software to predict when two evolved images have similar shapes or other structural elements (like lots of cross-hatching lines) in them. There were a number of ways in which to attempt this, and we opted for an approach based on segmenting images into colour regions. To construct a segmentation of an image efficiently, we first scale it down to a 24×24 array of pixels, using a straightforward colour averaging technique. We then use a neighbourhood growing method whereby the pixel, p, at location $(0, 0)$ forms the first member of the first neighbourhood, N_0, and any adjacent pixels are added to the neighbourhood if their colour is within a threshold distance d in RGB-space from the colour of p. This iterates until no new pixel is added to N_0, at which stage a new neighbourhood, N_1, is added to the segmentation, with the first pixel being the closest to the origin $(0, 0)$. Once all the pixels have been assigned to neighbourhoods, we call the final neighbourhoods *segments*, and each segment is given the average colour of the pixels in it. Different values of the distance threshold d naturally give different segmentations which highlight different structures in images. Example segmentations are given in figure 2, with RGB distance thresholds, d, of $0, 25, 50, 75$ and 100.

We can use the segmentations to compare a pair of images as follows. Given an image, A, we denote $seg_d(A)$ to be the set of segments produced using the above method with a distance threshold of d, where a segment is a set of pixel positions in a 24×24 array. Then, given images A and B, we define the *segmentation similarity* of the pair (A, B), denoted by $segsim_d(A, B)$, as:

$$segsim_d(A, B) = \frac{|\{(p, q) : \exists\, s \in seg_d(A) \wedge \exists\, s' \in seg_d(B) \text{ s.t. } p, q \in s \wedge p, q \in s'\}|}{576}$$

Informally, $segsim_d(A, B)$ is the proportion of pairs of pixel positions which are in the same segment for both the segmentation of A and the segmentation of B, when a segmentation of each image using threshold d is produced.

The $segsim_x$ values for three pairs of images are given in figure 2, for five different values of x. We experimented with some simple thresholding methods to try to predict when two images have similar structural compositions. However, the examples in figure 2 highlight the difficulty in a such a simplistic approach. In particular, the two images marked 'positive' – which are clearly structurally similar – have a $segsim_{25}$ value of 0.582, while the two clearly distinct images marked 'random negative', have a much higher $segsim_{25}$ value of 0.806. Hence, using a classifier based on only a threshold for $segsim_{25}$ would predict that the distinct pair are more alike than the clearly similar pair. We were equally able to find examples which ruin a simple threshold approach using $segsim_x$ for any x, hence we turned to a machine learning approach.

Our methodology to derive a classifier was as follows. We first collated 100 pairs of *positive* images which we deemed to be visually similar at a structural level, such as the first two images in figure 2. To populate a negative set, we randomly generated 1000 pairs of images. For each pair of images A and B,

we calculated the values of $segsim_d(A, B)$ for $d = 1, \ldots, 100$ and used these as the attributes of image pairs to learn over. We employed the WEKA machine learning software [10], as this provides a remarkable range of possible learning techniques to experiment with. For this initial data – which we call the 'random negatives' data set – we found that almost all WEKA's tree learning and rule learning approaches produced a very high predictive accuracy. We rather arbitrarily chose to use the output from the BFTree method, and implemented it as executable Java code. The tree from this application is the first in figure 4, and it achieves a 97.9% predictive accuracy on the training set, with the learning method achieving 94.9% under ten-fold cross validation.

To test the ability of the decision tree, we randomly sampled pairs of images from the sessions described in section 2. For any pair that the decision tree predicted had similar structures, we visually checked whether the prediction was correct. We found that around 1 in 3 pairs should in fact have been classified as non-similar. We collected 250 examples of such pairs and added them to the negative data, calling the resulting data set 'intermediate'. We then ran the BFTree method in WEKA again, to produce decision tree (ii) in figure 4, which achieved a predictive accuracy of 97.4%, with a cross validation result of 95.6%. We again used the decision tree to highlight non-similar pairs from the preliminary session which were predicted incorrectly to be similar, and found that the frequency had dropped to around 1 in 10. We collected 250 of these false positives and added them to the negatives again to produce a data set of 1500 'difficult' negative pairs of images, with an example supplied in figure 2. We used the BFTree method to produce decision tree (iii) in figure 4. This achieved a predictive accuracy of 97.8%, with a ten-fold cross validation result of 93.8%. When we tested this, we found that it was very unlikely to falsely predict a positive, with such a case occurring around 1 in 100 times.

We also derived a decision tree to predict whether two genotypes would produce images with similar structural properties. Recall that the genotype of particle-based images includes six *init* functions to initialise the placement and colour of the particles, and six *update* functions to dictate how the colour and position of the particles change at each time step. To derive a decision tree using the genotype, for each of the pairs in the difficult data set, we recorded where the 12 genotype functions differed, in addition to the number of init and update functions and the overall number of functions that differ. Using this data, the BFTree method produced decision tree (iv) in figure 4, which scores 97.9% for predictive accuracy, with a ten-fold cross validation result of 97.3%.

The high predictive accuracy results from the ten fold cross validation exercise give us some confidence that the decision trees produced are not overfitting the data. In addition, the relatively large number of negatives compared to the number of positives in the data set ensures that the trees learned are far more likely to incorrectly classify a pair of similar images as non-similar than vice versa. In particular, looking at the confusion matrices for decision trees (iii) and (iv) in figure 4, we see that each will mis-classify about a third of truly similar images as non-similar. Hence the decision trees can be seen as highly cautious

about predicting that two images have similar structures. The trees were used via a serial two tier approach within evolutionary sessions. That is, to be classified as unique, new individuals have to first be predicted as different to every other in the current and every previous generation by the fast genotype tree method. If they pass this test, then images are produced and tested for uniqueness using the phenotype tree. Having both tests in series caters for the situation where a pair of individuals with quite different genotypes have converged on structurally similar images, e.g., the third pair in figure 2.

We have found that the cautious nature of the decision trees in predicting similarity is balanced by the fact that a new individual is tested for similarity against so many others, i.e., against *every* other image ever seen. In practice, in evolutionary sessions, this means that the two tier approach is very effective at forcing the uniqueness of images, yet it allows a large enough variety of images to be produced for an evolutionary session to proceed. To test whether the stricter pruning stifles attempts to increase the yield of images maximising certain aesthetic fitness criteria, we ran the FP sessions again, but this time pruning any individual which fails the two-tier (TT) uniqueness test (including the fifty random individuals in the first population, and the five random individuals introduced in each generation). In the next section, we describe the results from these sessions, which we denoted $TT_{v+}, TT_{v-}, TT_{c+}, TT_{c-}, TT_{s+}$ and TT_{s-}

4 Session Analyses

We first assess the *multiplicity* of the images in each generation of the sessions described in section 2. This is defined as the average number of similar images – as predicted by the two-tier decision tree approach – per individual in each generation. The results are given in figure 3. We note that this automated analysis matches very well our subjective analysis, i.e., that far too many repeated images were being produced. For instance, in the final generation of the FP_{c+} session, the multiplicity was 29.24, and on visual inspection, that generation only contains 2 different image types (other than the five random individuals). In the worst session, FP_{s+}, the multiplicity was 17.43 on average, which is clearly a wasted effort, and even the best session, CB_{s+}, had an average multiplicity of 2.48. The average multiplicity over the six FP and six CB sessions was 6.43 and 4.96 respectively, so the clustering improved the search coverage somewhat.

Of course, multiplicity was ruled out in the TT sessions. We noted that, on average, the first tier (genotype decision tree) was rejecting around 100 individuals per population, i.e., around 2 individuals for every one that it allowed through, while the second tier (segsim decision tree) was rejecting much less than this, at a rate of around 10 rejections per population. This was roughly the same for each TT session. The strict pruning employed in the TT sessions naturally raises the question of whether the fitness functions could still drive the search and yield larger numbers of fit individuals than random generation. In figure 6, we show how the fitness – relative to the respective measure – of individuals changes on average as the sessions progress. For instance, in the TT_{v+} session, we compare how the average colour variety of individuals in each population changes as the generations progress. Note that the values in the graphs

have been normalised with respect to the lowest and highest fitness ever seen, to bring the values into the whole range between 0 and 1, for better visualisation.

In figure 6, we compare the average fitness over the same number (1250) of random individuals (yellow dashed line) and the average fitness of the evolved individuals (green dotted line). In all but the TT_{s+} session, there is a clear improvement in the average fitness of the evolved images over the randomly generated individuals. We note that it was probably a mistake to average over all seven symmetries in the symmetry measure, as this tends to produce images that have mild symmetry in all dimensions, or strong symmetry in one dimension, which may cause fitness conflicts in the search, and explain the relatively poor performance in the TT_{s+} session. We plan to remedy this with a symmetry measure which takes the maximum symmetry in any one dimension, and run further tests. In the TT_{v+}, TT_{c+} and TT_{s-} sessions, there is a clear trend over the generations to higher fitness, but in the other sessions, the populations vary more wildly from generation to generation. Overall, we can conclude that the fitness functions do indeed drive the TT search, and produce higher yields of fitter individuals than random generation. The TT sessions took twice as long as the FP and CB sessions, but this wasn't an issue for us.

In terms of populating the library of scene descriptors, the TT searches have been very successful. By collating all the images from all the sessions described in sections 2 and 3, deleting any repetitions predicted by the decision trees, we have added around 10,000 unique images to the library. The TT sessions contributed more than 70% of these images. Moreover, the library is searchable using ranges of colour variety, contrast and symmetry, which is a very valuable tool. Some examples of search results where two of these dimensions were varied (in quartile ranges) are given in figure 5. The library will return a hit 100% of the time for any triple of 25% percent ranges for colour variety, contrast and symmetry, and returns a hit 80% of the time for any triple of 10% ranges. This is a good start, but we plan to increase this statistic through further TT sessions.

5 An Artistic Application

There are many possibilities for using the library within The Painting Fool's painting generation processes, and we describe here our first attempt to use the images as scene descriptors. Figure 7 portrays a graphics pipeline for how painterly images with recognisable scene elements can be produced from an abstract art image. Firstly, the scene descriptor library is queried for an image with low colour variety but medium contrast. Then, points from the image are sampled, and turned into non-overlapping 2D boxes, with each box being given the colour of the sampled point (no averaging is carried out). Next, a perspective transform is applied to the set of boxes, and the resulting quadrilaterals are ordered into a list so that the ones at the bottom are last in the list, hence will be rendered last. Each quadrilateral is then stretched to provide the final placeholders, and each placeholder is replaced by a human figure generated by the ContextFree software (www.contextfreeart.org), which has been pre-segmented into 17 colour regions. Some random inaccuracy in the replacement process is

intentionally introduced to produce a more naturalistic look (but this randomness is not present in the second example of figure 7). Finally, each paint region is rendered by the simulation of acrylic paints, used to paint around the region border. The resulting painting has retained much of the aesthetic qualities of the original scene descriptor, yet includes representational (figurative) elements, hence is more worthy of audience interpretation.

6 Conclusions and Future Work

The kind of abstract art pieces we have generated for the library have many decorative, aesthetically appealing, qualities, but few representational qualities. On the other hand, the kind of art produced by The Painting Fool is representational, hence can possibly convey meaning, but does not explicitly appeal to any aesthetic considerations. It therefore seems sensible to combine the relative qualities of the two approaches, and the building of the scene descriptor library is the first step towards such a fruitful combination. We have described here an evolutionary approach informed by machine learning methods which can build a large library of such scene descriptors which differ markedly in their structural content. We have further shown that this can be driven by fitness functions which reward aesthetic qualities, with the resulting sessions producing higher yields of valuable images than random generation.

There are many improvements to the method that we could make, including employing other segmenting techniques such as active contours (snakes) [11], or appealing to image retrieval research for detecting image similarity [7]. However, given that the method appears to work well, we plan to first concentrate on experimentation with other fitness functions. In particular, while most of the pieces in the library have a certain aesthetic appeal (judged subjectively), there are many for which this is not the case (also judged subjectively). We do not want to discard the less appealing instances, as they may of course be useful in producing different styles. However, we plan to implement aesthetic evaluation methods such as those described in [8] and [14], in order to generate more varied images for the library, and for it to have more information about the images it contains. We will also compare and contrast our approach with that described in [15], where the evolutionary system was forced to look for different styles in each different session – this may be an equally fruitful approach to generating a database of images. We similarly plan to experiment with image generation techniques other than the particle based approach, again to build up a more interesting and varied library. We expect the two-tier similarity prediction method to work on images generated in different ways, but this has to be tested.

The value of having such a large library of images is that the software can use sets of scene descriptors that have not been seen before, hopefully leading to novelty and surprise, which are maxims in computational creativity projects [13]. We plan to harness this by investigating different methods for interpreting the images as instructions for scene construction. In particular, we intend to implement methods whereby the software can retrieve images from the library which will emphasise a message to convey through the artwork. For instance, if

the message has a note of discourse, then the software will know that this might be emphasised by a scene with a great deal of colour variance and contrast. Similarly, notes of harmony might be better conveyed with symmetric rather than non-symmetric scenes. We believe that the coupling of the decorative aspects of evolved abstract art and the representational aspect of non-photorealistic rendering approaches will drive forward automated painting to produce culturally interesting and meaningful pictures worthy of proper consideration as artworks.

7 Acknowledgments

We would like to thank the anonymous reviewers for their interesting comments. This work has been supported by EPSRC grant EP/J004049/1.

References

1. D Arthur and S Vassilvitskii. K-means++: The advantages of careful seeding. In *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.
2. S Colton. Automatic invention of fitness functions, with application to scene generation. In *Proceedings of the EvoMusArt Workshop*, 2008.
3. S Colton. Experiments in constraint-based automated scene generation. In *Proceedings of the 5th Int. Joint Workshop on Computational Creativity*, 2008.
4. S Colton. Seven catchy phrases for computational creativity research. In *Proceedings of the Dagstuhl Seminar on Computational Creativity*, 2009.
5. S Colton. The Painting Fool: Stories from building an automated artist. In J McCormack and M d'Inverno, editors, *Computers and Creativity*. Springer, 2012.
6. S Colton, M Cook, and A Raad. Ludic considerations of tablet-based Evo-art. In *Proceedings of the EvoMusArt workshop*, 2011.
7. R Datta, D Joshi, J Li, and J Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2), 2008.
8. E den Heijer and A Eiben. Comparing aesthetic measures for evolutionary art. In *Proceedings of the EvoMusArt workshop*, 2010.
9. A Ekárt, D Sharma, and S Chalakov. Modelling human preference in evolutionary art. In *Proceedings of the EvoMusArt workshop*, 2011.
10. M Hall, E Frank, G Holmes, B Pfahringer, P Reutemann, and I Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.
11. M Kass, A Witkin, and D Terzopoulos. Snakes - Active contour models. *International Journal of Computer Vision*, 1(4), 1987.
12. Y Li and C Hu. Aesthetic learning in an interactive evolutionary art system. In *Proceedings of the EvoMusArt workshop*, 2010.
13. L Macedo and A Cardoso. Assessing creativity: The importance of unexpected novelty. In *Proceedings of the 2nd Workshop on Creative Systems*, 2002.
14. P Machado and A Cardoso. Computing aesthetics. In *Proceedings of the Brazilian Symposium on Artificial Intelligence*, 1998.
15. P Machado, J Romero, and B Manaris. Experiments in computational aesthetics – an iterative approach to stylistic change in evolutionary art. In *The Art of Artificial Evolution*. Springer, 2007.
16. P McCorduck. *AARON's Code: Meta-Art, Artificial Intelligence, and the Work of Harold Cohen*. W. H. Freeman and Company, 1991.
17. R Saunders and K Grace. Extending context free to teach interactive evolutionary design systems. In *Proceedings of the EvoMusArt workshop*, 2009.

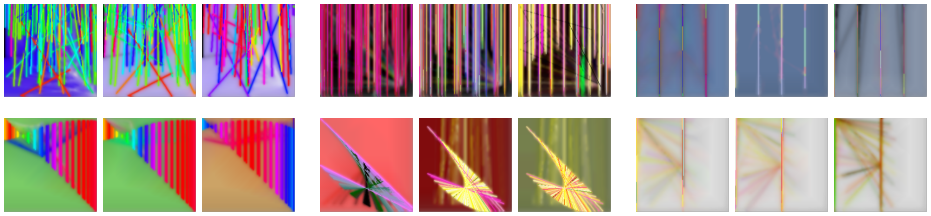

Fig. 1. Sample images from the final generations of the $FP_{v+}, FP_{c+}, FP_{s+}$ sessions (top row) and the $CB_{v+}, CB_{c+}, CB_{s+}$ session (bottom row).

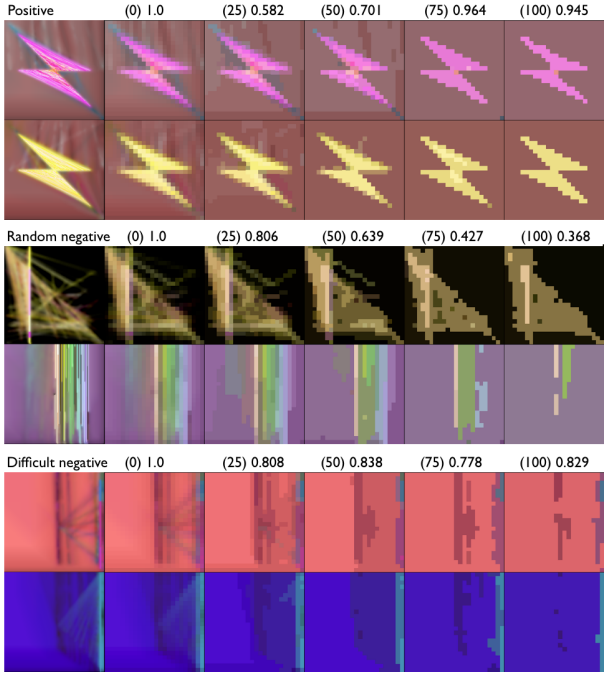

Fig. 2. Segmentation similarity – *segsim* – values for three pairs of images, given along with the distance thresholds (in brackets) and segmentation images

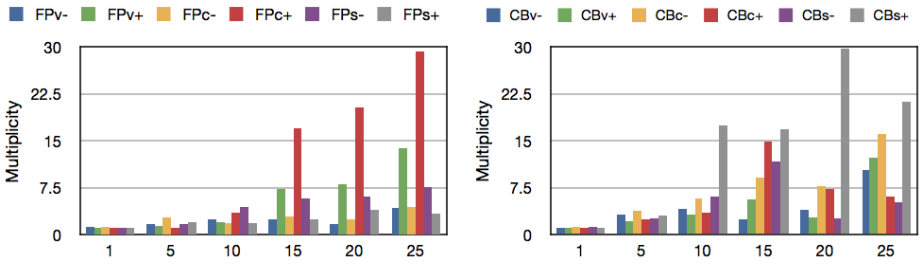

Fig. 3. Multiplicity analysis of preliminary sessions. Generation no. is on the x-axis.

(i) 97.9% 94.9%

```
segsim46 < 0.8019
| segsim15 < 0.88334
| |  segsim4 < 0.8957
| | | segsim40 < 0.86446: NonSimilar
| | | segsim40 >= 0.86446: Similar
| | segsim4 >= 0.8957
| | | segsim47 < 0.40903: NonSimilar
| | | segsim47 >= 0.40903: Similar
| segsim15 >= 0.88334
| | segsim2 < 0.59781: NonSimilar
| | segsim2 >= 0.59781: Similar
segsim46 >= 0.8019
| segsim1 < 0.54291: NonSimilar
| segsim1 >= 0.54291
| | segsim7 < 0.56875: NonSimilar
| | segsim7 >= 0.56875
| | | segsim1 < 0.64654: Similar
| | | segsim1 >= 0.64654: Similar
```

(ii) 97.4% 95.6%

```
segsim19 < 0.89323
| segsim1 < 0.76698: NonSimilar
| segsim1 >= 0.76698
| | segsim13 < 0.80198
| | | segsim9 < 0.88004: NonSimilar
| | | segsim9 >= 0.88004: Similar
| | segsim13 >= 0.80198
| | | segsim54 < 0.6175
| | | | segsim1 < 0.8572: NonSimilar
| | | | segsim1 >= 0.8572: Similar
| | | segsim54 >= 0.6175: Similar
segsim19 >= 0.89323
| segsim5 < 0.65812: NonSimilar
| segsim5 >= 0.65812: Similar
```

(iii) 97.8% 93.8%

```
strisegsim17 < 0.92551
| segsim64 < 0.81387
| | segsim28 < 0.91632
| | | segsim2 < 0.78682
| | | |  segsim20 < 0.91481
| | | | | segsim98 < 0.21473: NonSimilar
| | | | | segsim98 >= 0.21473
| | | | | | segsim40 < 0.86694
| | | | | | | segsim15 < 0.85294
| | | | | | | | segsim18 < 0.86048: NonSimilar
| | | | | | | | segsim18 >= 0.86048: NonSimilar
| | | | | | | segsim15 >= 0.85294: NonSimilar
| | | | | | segsim40 >= 0.86694: NonSimilar
| | | | segsim20 >= 0.91481: NonSimilar
| | | segsim2 >= 0.78682
| | | | segsim16 < 0.88643: NonSimilar
| | | | segsim16 >= 0.88643
| | | | | segsim50 < 0.67126: NonSimilar
| | | | | segsim50 >= 0.67126: Similar
| | segsim28 >= 0.91632: Similar
| segsim64 >= 0.81387
| | segsim23 < 0.81288
| | | segsim24 < 0.64992
| | | | segsim41 < 0.94183: NonSimilar
| | | | segsim41 >= 0.94183: Similar
| | | segsim24 >= 0.64992
| | | | segsim6 < 0.85528
| | | | | segsim42 < 0.8993: NonSimilar
| | | | | segsim42 >= 0.8993: Similar
| | | | segsim6 >= 0.85528
| | | | | segsim21 < 0.68543: NonSimilar
| | | | | segsim21 >= 0.68543: Similar
| | segsim23 >= 0.81288
| | | segsim80 < 0.86131
| | | | segsim51 < 0.89162
| | | | | segsim1 < 0.8291: NonSimilar
| | | | | segsim1 >= 0.8291
| | | | | | segsim59 < 0.84314: Similar
| | | | | | segsim59 >= 0.84314: NonSimilar
| | | | segsim51 >= 0.89162: Similar
| | | segsim80 >= 0.86131: Similar
segsim17 >= 0.92551: Similar
```

(iv) 97.9% 97.3%

```
overall_trees_different < 4.5
| update_trees_different < 1.5
| | update_tree_different0 < 0.5
| | | init_tree_different1 < 0.5: Similar
| | | init_tree_different1 >= 0.5
| | | | init_tree_different3 < 0.5: NonSimilar
| | | | init_tree_different3 >= 0.5: NonSimilar
| | update_tree_different0 >= 0.5: Similar
| update_trees_different >= 1.5
| | init_tree_different5 < 0.5: NonSimilar
| | init_tree_different5 >= 0.5
| | | update_tree_different4 < 0.5: NonSimilar
| | | update_tree_different4 >= 0.5
| | | | init_tree_different0 < 0.5: Similar
| | | | init_tree_different0 >= 0.5
| | | | | update_tree_different0 < 0.5
| | | | | | update_tree_different1 < 0.5: Similar
| | | | | update_tree_different1 >= 0.5
| | | | | | init_tree_different4 < 0.5: Similar
| | | | | | init_tree_different4 >= 0.5: NonSimilar
| | | | update_tree_different0 >= 0.5: Similar
overall_trees_different >= 4.5: NonSimilar
```

(i)

	ss	ns
ss	80	20
ns	3	997

(ii)

	ss	ns
ss	69	31
ns	4	1246

(iii)

	ss	ns
ss	70	30
ns	5	1495

(iv)

	ss	ns
ss	74	26
ns	7	1493

Fig. 4. Segsim-based decision trees learned for the (i) random (ii) intermediate, and (iii) difficult negative data sets. (iv) Genotype-based decision tree for the difficult negative data set. The segsimX notation in the tree refers to the output from the $segsim_X(A, B)$ calculation for a given pair of images A and B. The percentages given are the predictive accuracy of the tree on the training set, and the predictive accuracy of the learning method under ten-fold cross validation. The confusion matrices for each tree are also given, with *ss* denoting similar structure and *ns* denoting non-similar structures.

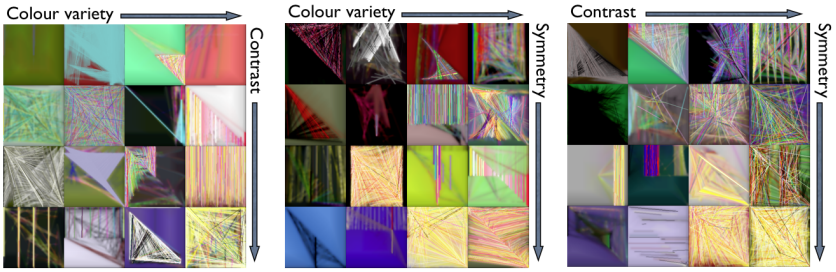

Fig. 5. Example images with varying aesthetic properties ranging over quartiles.

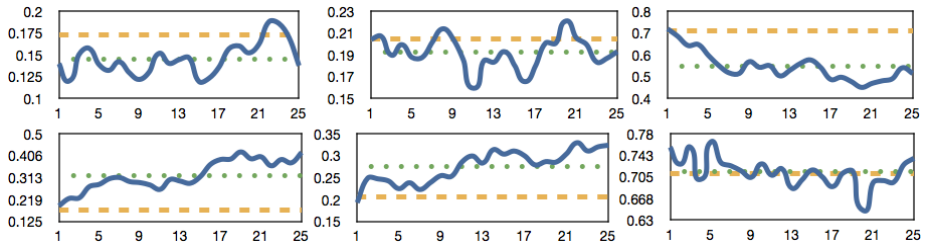

Fig. 6. Evolution overview for the sessions implementing the two tier pruning method. Top row: $TT_{v-}, TT_{c-}, TT_{s-}$ sessions. Bottom row: $TT_{v+}, TT_{c+}, TT_{s+}$ sessions. The generation number is on the x-axis. The y-axis represents the average colour variety, contrast and symmetry values in each generation, which is not the same as the fitness in the TT_{v-}, TT_{c-} and TT_{s-} sessions. The values have been normalised with respect to the lowest/highest values ever seen. The yellow dashed lines represent the average of 1250 randomly generated images, while the green dotted lines represent the average of the evolved images in the session.

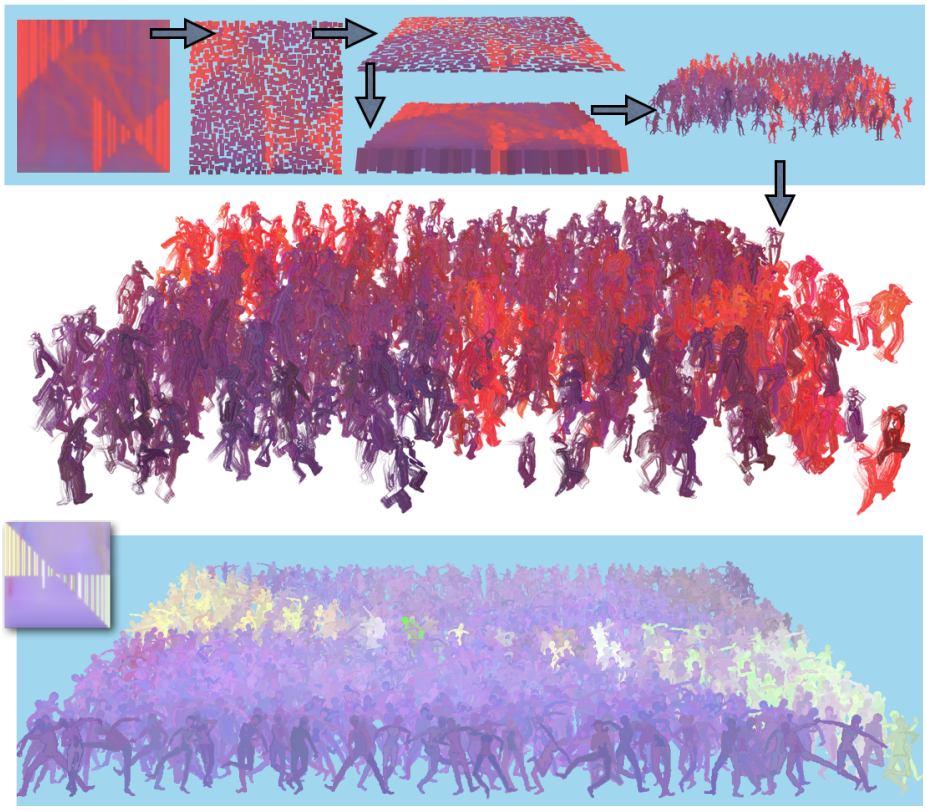

Fig. 7. Top: a pipeline for producing pictures starting with an abstract art scene descriptor. Bottom: another example of scene generation from an abstract art image.