Computational Aesthetics in Graphics, Visualization, and Imaging (2010)
O. Deussen and P. Jepp (Editors)

Stroke Matching for Paint Dances

Simon Colton

Computational Creativity Group
Department of Computing
Imperial College London
http://www.doc.ic.ac.uk/ccg

Abstract

We have implemented a non-photorealistic rendering system which simulates the placement of paint/pencil/pastel strokes to produce representational artworks from digital images. The system is able to record an image of each paint stroke independent of the overall picture, in addition to some details about each stroke. Working with sets of paint strokes from paintings of different images, we investigate how to determine which stroke from one picture most closely resembles a given stroke from another picture. This enables the paint strokes from one picture to be used to paint a different painting. This further enables the animation of one picture morphing into another, as the paint strokes move and rotate into new positions and orientations. Using a K-means clustering approach, we can extract a set of representative strokes from a series of paintings/drawings, and animate the same set of strokes moving around a picture in order to represent different scenes at different times. We call such animations "paint dances". We apply this technique to sets of portraits and we present the resulting paint dances in an artistic context as video art. We describe here the various methods we experimented with in order to determine an optimal stroke matching and extraction approach.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

The Painting Fool (www.thepaintingfool.com) is a computer program that we hope will one day be taken seriously as a creative artist in its own right. Leaving aside the computational creativity issues involved with this project, The Painting Fool is at heart a non-photorealistic rendering system which is able to produce painterly renditions of digital images, similar to a plethora of academic and commercial systems, such as those described in [GG01] or [SS02]. The Painting Fool places simulated paint/pencil/pastel strokes on a canvas, and as described in section 2, we have implemented the ability to record both a bitmap image of each stroke and certain details about the stroke. This facilitates the reconstruction of pictures at a later date (possibly with alterations); the comparison of pictures in terms of the strokes that they contain; and the animation of strokes moving on a canvas, which is the focus of the work here. We define a *paint dance* to be an animation depicting the movements of a set of simulated paint/pencil/pastel strokes. In particular, we are interested in the production of paint dances where the strokes periodically come together in a representational way, i.e., to produce a painterly rendition of a digital image. For our particular aesthetic considerations, we have restricted our attention to paint dances where strokes can only translate and rotate on the canvas. However, in future, we may enable changing the colours or performing non-linear transformations of the strokes during the paint dances.

There are two major hurdles to producing paint dances. The first difficulty is to determine how to paint pictorial representations of multiple images using the same set of paint strokes. In section 3, we describe a method for mapping one set of strokes onto another such that the mapping pairs up visually similar pairs of strokes. Given a set of pictures described in terms of the strokes they contain, the second difficulty comes in extracting from the overall set of paint strokes a representative sample which can be used to reconstruct each picture to an acceptable level of fidelity. In section 4, we describe methods which use K-means clustering to identify

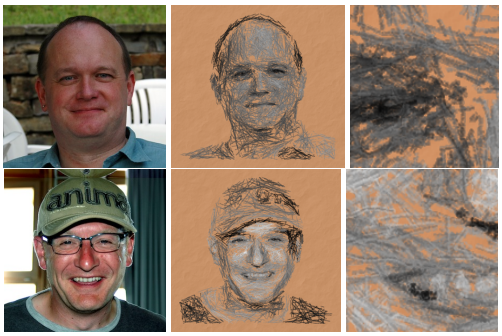

Figure 1: *Example drawings produced with setup 1.*

Figure 2: *Example paintings produced with setup 2.*

a set of exemplar strokes from which multiple pictures can be painted with relatively good fidelity. In section 5, we describe various experiments with stroke matching and stroke set extraction in order to determine optimal settings for these methods. In section 6, we describe how we constructed the animated paint dances. To conclude, we place this approach in context with non-photorealistic rendering projects involving animation, and we discuss future directions for this work.

2. Generating and Recording Rendered Strokes

The Painting Fool's non-photorealistic rendering system is straightforward in conception, and is described in detail in [CVP08]. The process starts with hand-annotated images, where regions of interest are drawn around and labeled with various tags, e.g., eye, mouth, etc. For each labeled region, the software segments the image using a neighbourhood-growing algorithm followed by a rationalisation of the boundary of each region. For each region, the user can specify different parameters for the segmentation process. The parameters control the number required and the nature of the regions, and the edge smoothing process. The segments from a region are tagged with the labels for that region, and the segment colour is set to the average of the RGB values over the pixels in the original neighbourhood from which it was derived. Segment colours can be mapped to the hue and/or saturation and/or brightness of the closest colour from a user-supplied colour palette.

Given the set of segments from all the regions, The Painting Fool simulates the usage of various natural art materials to colour in the body and/or edge of each segment. Each segment can be rendered repeatedly, and the entire set of segments can be rendered repeatedly in multiple layers. As described in [CVP08], the software can simulate acrylic paints, pencils, pastels, charcoals, felt-tip pens, watercolour paints and chalks with differing levels of visual success. For the experiments described here, we employ only the simulation of graphite pencil strokes and acrylic paint strokes. The Painting Fool has various methods for filling in a segment with strokes, but we only employ one here, namely the random fill method. This chooses two points randomly within the area defined by the segment and attempts to paint a stroke starting at one and ending at the other. The stroke stops if it leaves the region area, and the user can specify the length of the smallest stroke allowed, the percentage of the area covered by the region that should be filled, and the amount that the stroke can wobble along a line as it is drawn (leading to curved strokes). The user is also able to specify the maximum and minimum stroke width allowed (denoted by *brush size*). For a particular segment, the brush size is chosen in proportion to the size of the segment, hence larger brushes are used to fill in segments covering a larger area.

For the experiments described below, we produced drawings and paintings from 32 images of faces, as per the following two rendering setups. For setup 1, we hand-annotated each image by drawing a region around the whole 'person', the 'face', and the facial features: the 'eyes', 'nose', 'mouth', 'ears' and 'eyebrows'. We specified that the whole person should be segmented into 50 regions, the face into 50 regions, the eyes and mouth into 50 regions, the nose into 25 regions (with the outer regions removed) and the eyebrows and ears into 10 regions. Each drawing was rendered onto an approximately 1200×1200 pixel image, and graphite pencil strokes with a fixed width were simulated to produce the pictures. Regions were filled using the random fill approach. In particular, regions annotated with the 'person' label were filled until 50% of the region was covered, as were the facial feature regions. Regions annotated with the 'face' label were filled until 40% of the region was covered. Two example drawings are given in figure 1.

For setup 2, we used the same segmentation scheme as in setup 1. However, in order to give the portraits something of a fauvist look, we mapped the hue and saturation of segments to the those of the closest colour from a palette of primary and secondary colours, as prescribed in [Kra02], without changing the segment's brightness. (*Les Fauves* were a group of artists from the early 1900s, called *wild beasts* by critics because of their use of bold colours [Whi91]). Regions were randomly filled as before (with the same percentages), but using simulated acrylic paints strokes. For regions tagged with the 'person' label, wide brush strokes of between 30 and 50 pixels were used. For 'face' regions,

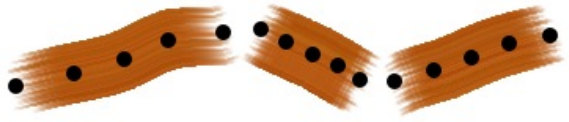

Figure 3: *Extract from an example strokes file, along with the anchor points recorded in the XML file for each stroke..*

medium brush strokes of between 20 and 30 pixels were used, and for facial feature regions, thin strokes of between 5 and 15 pixels were used. Two example paintings are given in figure 2.

To facilitate the kinds of paint dances mentioned above, during the rendering session, The Painting Fool records the following details about each paint stroke:

- The co-ordinates for the top left corner of the stroke's bounding box
- The average colour of the pixels rendered for the stroke
- The width of the brush that was used for the stroke
- The length of the curve that the stroke was rendered on
- Five anchor points distributed evenly along the curve that the stroke was rendered on

At the end of the rendering session, an XML file is generated and saved which contains all the stroke information. In addition, a bitmap file with the stroke images delineated from each other is recorded, as in figure 3. Using these stroke images with the co-ordinate information from the XML file enables the faithful reconstruction of the picture at a later date.

3. Matching Stroke Sets

The starting point for our paint dances is a set of paintings/drawings (pictures) like those in figures 1 and 2 above. The dance is performed by sequentially morphing one picture into another, with the paint strokes from the first becoming those of the next, and so on. Moreover, as mentioned previously, we do not allow the paint strokes to be transformed other than with rotations and translations. To facilitate the animation, we start with a generic set of strokes (calculated using techniques described in section 4), and work out how best to use them to re-paint each picture in the series. In particular, we suggest below a way in which one set of strokes may be equated with another set, so that the first set can be used to paint a facsimile of the picture originally produced by the second set.

Suppose we have two lists of strokes, S_1 and S_2, of equal size, which have been produced via the rendering and recording of pictures P_1 and P_2 respectively, using the processes described in section 2 above. For any stroke s, let $centre(s)$ be the third point in the set of five recorded for it, i.e., the central anchor point for the strokes in figure 3. We define a *stroke-mapping*, $\langle M,R\rangle$, to be composed of a bijective mapping $M : S_1 \rightarrow S_2$, and a set of rotation angles $R = \{r_1,...,r_{|S_1|}\}$, where $\forall r, r \in R$ and $0 \leq r < 360$. Let $pos(s)$ denote the position of stroke s in S_1, let $M(s)$ denote the stroke in S_2 that M maps $s \in S_1$ onto, and let $R(s)$ denote the rotation angle in R at position $pos(s)$. We define the *repainting* of P_2 using $\langle M,R\rangle$, denoted $P_2(M,R)$, as the image produced by taking each $s \in S_1$ and translating it so that $centre(s) = centre(M(s))$, then rotating it around $centre(M(s))$ by $R(s)$ degrees.

Note that images P_2 and $P_2(M,R)$ may have different dimensions, and suppose that w is the largest of the two widths, and h is the largest of the two heights, and that the two pictures have been registered within an image, Im, of dimension $w \times h$ pixels, so that the centre points of each original stroke, s, and each mapped stroke, $M(s)$, co-incide. Let $sp(Im)$ denote the set (without repetitions) of positions of the stroke pixels in either P_2 or $P_2(M,R)$, i.e., the position of pixels which the placement of any stroke changes. We define the *fidelity*, $f(M,R)$ of the stroke mapping $\langle M,R\rangle$ as follows:

$$1 - \frac{\sum_{x=1}^{w}\sum_{y=1}^{h}(||rgb(P_2,x,y),rgb(P_2(M,R),x,y)||)}{|sp(Im)| * \sqrt{3*256^2}}$$

where $||rgb(P_2,x,y),rgb(P_2(M,R),x,y)||$ is the Euclidean distance in RGB-space between the colour of pixel P_2 at co-ordinate (x,y) and the colour of pixel $P_2(M,R)$ at co-ordinate (x,y). Note that $\sqrt{3*256^2}$ is a normalisation factor: it is the distance between black and white in RGB space.

In algorithmic terms, given a stroke-mapping $\langle M,R\rangle$, to produce a repainting of P_2 using the strokes from S_1, each stroke in S_1 is translated so that its centre point is at the same co-ordinate of the centre point of a corresponding (according to M) point from S_2, and then the stroke is rotated according to R. We can approximate how closely the repainted image looks like the original picture by first registering one image over the other so that every pair of stroke centres co-incide properly, and then calculating the average distance between the RGB values of corresponding non-background pixels.

In order to determine methods for generating stroke mappings which achieve acceptably high fidelity, we start with a random method, which assigns the strokes from S_1 to those from S_2 randomly to construct the bijective mapping M, and then randomly assigns rotation values to R. This gives us a base-line for fidelity on which to improve, and some values determined experimentally for this base-line are given in section 5. At the other end of the spectrum, the best stroke map generating algorithm would work out the RGB distance between each pair of centre-registered strokes under every possible rotation. However, such a method is computationally expensive. In order to find a computationally sensible middle ground, upon examination of the repaintings achieved using the random approach, we noted that there are four factors which effect fidelity. In particular, strokes s and $M(s)$ will be a poor match if (i) their colours differ too much (ii) their lengths differ too much (iii) their brush sizes differ too much or (iv) there is no suitable rotation which can move

the anchor points of $M(s)$ into positions close to the anchor points of s.

Based on these observations, we can suggest a weighted sum estimate for how visually different stroke $s_1 \in S_1$ is to stroke $s_2 \in S_2$, which does not involve calculating an RGB distance between the two strokes. Before doing so, we first need to define $length\_distance(s_1, s_2)$ as:

$$\frac{|length(s_1) - length(s_2)|}{max_{s_a \in S_1 \cup S_2, s_b \in S_1 \cup S_2} |length(s_a) - length(s_b)|}$$

where $length(s)$ is the stroke length recorded for stroke s. The length distance is therefore the difference between the stroke length of s_1 and s_2, normalised via division by the difference between the longest and shortest strokes in either picture. We define $brush\_distance(s_1, s_2)$ similarly (with $length(s_i)$ substituted by $brush\_size(s_i)$, i.e., the size of the brush recorded for s_i). We similarly define $colour\_distance(s_1, s_2)$ as the Euclidean distance between the recorded colour of stroke s_1 and the recorded colour of stroke s_2, normalised by division by the maximum colour distance between any pair of strokes from either picture. Given weights w_1, w_2 and w_3, the weighted sum estimate of stroke difference (sd) for strokes s_1 and s_2 can now be defined as:

$$\begin{aligned} sd(s_1, s_2, w_1, w_2, w_3) \quad = \quad & w_1 * length\_distance(s_1, s_2) + \\ & w_2 * brush\_distance(s_1, s_2) + \\ & w_3 * colour\_distance(s_1, s_2) \end{aligned}$$

Given stroke lists S_1 and S_2 as above, and a prescribed set of weights w_1, w_2 and w_3, we use this estimation in a greedy algorithm to determine a suitable stroke-mapping as follows. Firstly, we order the strokes in S_2 by descending stroke area, where stroke area is calculated as the stroke's length multiplied by the stroke's brush width. We then take each $s_2 \in S_2$ in this order, and find the $s_1 \in S_1$ which minimises $sd(s_1, s_2, w_1, w_2, w_3)$. We then remove s_1 from S_1, so that it cannot be paired with any other stroke from S_2. In this manner, we can build up a bijective mapping M which finds the best matches for the strokes from S_2, with the largest strokes being matched earlier than the smaller ones. Then, for each pair $(s, M(s))$, once s has been registered so that its centre co-incides with that of $M(s)$, we calculate the rotation angle $R(s)$ as the integer angle r between 0 and 360 which minimises $\sum_{i=1}^{5} ||p_i(s), p_i(M(s))||$, where $p_i(s)$ is the i-th anchor point recorded for stroke s, and $||x, y||$ denotes the straight line distance between points x and y in a 2D plane. To summarise, we determine a stroke mapping $\langle M, R \rangle$ of strokes S_1 onto picture P_2 by looking at each stroke of S_2 (in decreasing stroke area), and estimating which stroke, s, of S_1 looks the most like it, in terms of a weighted sum of stroke length difference, brush size difference and stroke colour difference. For each match, we work out how best to rotate s so that its anchor points are as close as possible to the anchor points of the original stroke from S_2.

In practice, we do not prescribe the weights for the weighted sum, as we have found experimentally that it is difficult to know in advance what weightings will produce the highest fidelity repainting for any pair of pictures. Hence, we perform an exhaustive search over the sets of weights (w_1, w_2, w_3) in:

$$\{(w_1, w_2, w_3) \in \{0, 0.1, \ldots, 1.0\}^3 : w_1 + w_2 + w_3 = 1\}$$

in order to find the optimal set of weights. That is, for each set of weights, we produce the map $\langle M, R \rangle$ using the greedy algorithm above, and then calculate the fidelity $f(M, R)$. We record the weights which maximise $f(M, R)$. We call this the *exhaustive-greedy* method. While it is fairly computationally expensive, time is not particularly an issue in the production of the paint dances. We compare the results of the random versus the exhaustive-greedy search in section 5. In figure 4, we present the results of the random generation of a stroke mapping against the exhaustive greedy method for rendering setup 1. We see that the increase in fidelity is clearly marked.

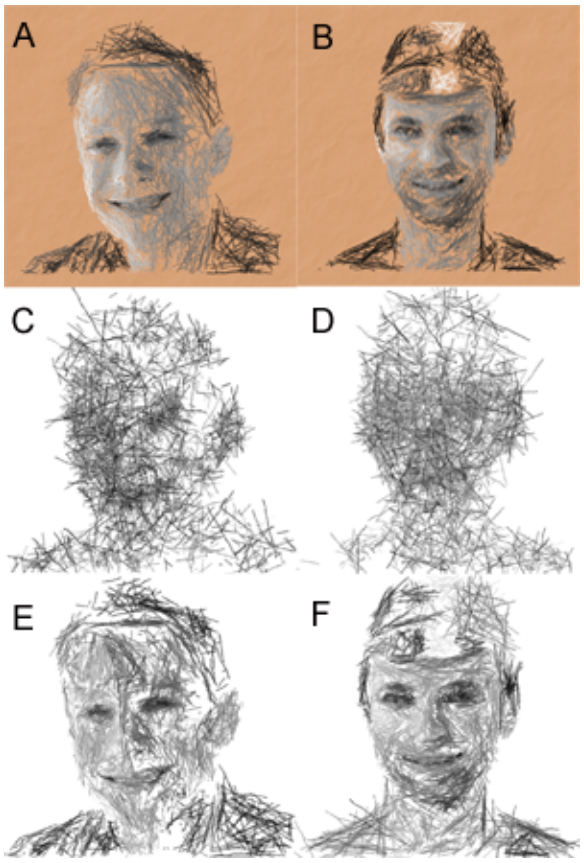

Figure 4: *Images A and B are the original drawings. Image C was produced by mapping the strokes of B onto A with a random stroke mapping (vice-versa for image D). Likewise for images E and F, but the stroke maps were produced using the exhaustive greedy method. The fidelity of the maps producing the images are: C = D = 0.8, E = F = 0.86.*

4. Extracting Generic Stroke Sets

As mentioned previously, stroke dances require the sequential rendering of multiple pictures, $P = P_1, \ldots, P_n$ by a single set of paint strokes. If we label the entire set of strokes $E = \{s : s \in P_i \text{ for some } i\}$, then we need to extract a generic set, G, from E in such a way that the fidelity of the stroke mappings from G to each picture P_i is as high as possible. We chose to do this using a K-means clustering approach to cluster the strokes in E, and then to extract exemplars from the clusters to form G. K-means clustering [Mac67] is a well known method which groups together objects represented by a vector of numerical values into a given number, k, of clusters. It works by randomly choosing a set of k objects as the initial means for the clusters. Then, each object is assigned to the cluster it is closest to (in terms of the Euclidean distance between it and the mean of the cluster). The mean of each cluster is then replaced by the centroid of the cluster, and each object is re-assigned to (possibly different) clusters, etc. This continues until no object is re-assigned. The K-means++ clustering algorithm [AV07] more intelligently chooses the initial means, and converges twice as fast on average as K-means, while consistently producing lower error clusterings (in terms of the average distance between the objects and the centroid of the cluster to which they belong).

We represent each stroke with a vector of its length, brush size and the R,G,B values of its average colour. Letting k be the maximum number of strokes in any P_i of P, and $n = |E|$ be the total number of strokes to cluster, we produce k clusters using either K-means or K-means++ clustering. We then extract strokes from the clusters in one of three ways:

- *simple method:* a random cluster is chosen, and the closest stroke to its centroid is extracted. Each cluster supplies a stroke before the second closest stroke from a previous cluster is extracted, etc. (which doesn't happen when the required number of strokes, q, is the same as the number of clusters, but does happen when q increases – see later).
- *proportional method:* the t closest strokes to the centroid of each cluster are extracted, where t is proportional to the the size, c, of the cluster, i.e., $t = c * k/n$.
- *distribution method:* a cluster is chosen at random with probability, p, proportional to the size, c, of the cluster, i.e., $p = c/n$. Then, the closest stroke to the centroid is extracted. If a cluster is revisited, the second closest stroke to the centroid is chosen, and so on. The method repeats until k strokes have been extracted.

In section 5, we describe experiments where the clustering and extraction techniques are altered, and we compare the fidelity of the resulting stroke mappings. As an example, in figure 5, there are 16 drawings comprising 59,983 pencil strokes overall, with lengths from 12 to 280 pixels. The most strokes in any drawing is 4,373. We used K-means++ with simple extraction to extract this number of generic strokes. Placing the extracted strokes in their original positions/orientations gives the composite image of figure 5.

Figure 5: *16 drawings and the composite image of the generic strokes extracted using the clustering method.*

Figure 6: *Paintings used in the experiments.*

5. Experiments and Results

We work here with a set of 32 drawings, half of which are shown in figure 5, and a set of 32 paintings, as portrayed in figure 6. The original portraits capture a likeness of the sitters which we wanted to maintain during the paint dances. Hence, we concentrated on increasing the fidelity of the stroke mappings, so that during the paint dances, the people being portrayed appear with an acceptable likeness.

We first compared different stroke mapping generators, using a base-line of the random generator as described above. We compared the exhaustive-greedy generator with fixed weight schemes where one of the stroke properties (brush size, colour, length) were fully weighted with the others being zero-weighted, and where the three properties were equally weighted. To perform the comparison, we took a pair of random pictures and the one with the most strokes in had strokes randomly removed until it had exactly the same number as the other. For each stroke map generating method, we produced a stroke mapping so that the strokes of the first picture were used to repaint the second picture, and we recorded the fidelity of the mapping. We repeated this 100 times for both drawings (rendering setup 1) and paintings (setup 2). Note that the brush size in rendering setup 1 was fixed, hence we fixed the brush size weight for the drawings experiments at zero, and we did not experiment with a weighting scheme where only brush size has a non-zero weight. Moreover, in each case, the exhaustive-greedy method determined a best set of weights for the weighted sum estimate. We recorded the average of these over the

100 runs, and found that the average best set of (brush size, colour, stroke length) weights for setups 1 and 2 were: $(0, 0.424, 0.573)$ and $(0.251, 0.226, 0.523)$ respectively. We generated stroke mappings for a further 100 random pairs of pictures as before, but using these fixed best weights. The results are recorded in table 1(a). We see that achieving fidelity is more difficult for the paintings (setup 2) than for the drawings, and that in both cases, the average fidelity of the mappings produced randomly is lower than for all but one non-random methods, with the exception being colour only weights for the pencil drawings, where colour is the least important aspect. We further see that the exhaustive-greedy, equal weighting and best weightings methods perform better than the others on average, which was expected. For our aesthetic purposes, the fidelity of every picture in the paint dance series is important, i.e., every portrait needs to have a good likeness. Hence, we are most interested in the worst case scenario with a mapping generator. For the worst mappings, we see in table 1(a) that the exhaustive, equal and best methods again outperform the others. Interestingly, for paints, the brush, colour and length methods perform worse than random with respect to worst-case mappings.

We next addressed the question of being given a set of pictures, and extracting a subset of strokes from the totality of strokes which achieves acceptably high fidelity when used to repaint all the pictures. In practice for a given clustering/extraction method pairing, we determine the largest number of strokes needed to render any single picture in the set, extract this many from the overall set, and use the exhaustive-greedy method to calculate a stroke mapping from a subset of the extracted strokes to the strokes in the picture. We varied both the clustering approach (K-means and K-means++) and the exemplar extraction method (simple, proportional and distribution), by generating a clustering, then using the three different extraction methods on it. We then checked the fidelity of the stroke mapping generated for each extraction method, when using the exhaustive-greedy mapping generator. The results are recorded in table 1(b), and we see that there is no discernible difference in fidelity for any of the clustering and extraction setups, with the possible exception of the pairing of K-means++ and the simple extraction method outperforming the other pairings for the paints setup. These results possibly highlight the value of the clustering method, i.e., however the exemplars are extracted, the clustering method itself has the most effect on the fidelity of the mappings which result.

Concentrating on the repaintings produced using the K-means++ and simple extraction method, on visual inspection of the drawings, the majority of the repaintings retained enough fidelity for an acceptable likeness of the person in the portrait, with the example provided in figure 8(a) exhibiting the highest fidelity (0.901). The same was not true for the paintings. For instance, the example in figure 8(a) has a high fidelity of 0.912, but does not sufficiently capture the facial features of the sitter. Moreover, as mentioned previously, the

Mapping Generator	Setup 1 (Pencils) worst	av.	best	Setup 2 (Paints) worst	av.	best
Random	0.744	0.802	0.840	0.627	0.694	0.794
Exhaustive	0.799	0.859	0.894	0.653	0.788	0.889
Equal	0.798	0.857	0.894	0.635	0.773	0.889
Brush		n/a		0.626	0.702	0.805
Colour	0.777	0.740	0.877	0.610	0.744	0.858
Length	0.754	0.819	0.864	0.625	0.711	0.812
Best	0.775	0.867	0.894	0.656	0.792	0.871

Clustering Method	Extraction Method	Setup 1 (Pencils) worst	av.	best	Setup 2 (Paints) worst	av.	best
k-means	Simple	0.860	0.890	0.901	0.784	0.876	0.919
k-means	Dist.	0.865	0.891	0.903	0.782	0.877	0.922
k-means	Prop.	0.862	0.891	0.902	0.786	0.878	0.924
k-means++	Simple	0.865	0.892	0.904	0.814	0.889	0.918
k-means++	Dist.	0.862	0.890	0.901	0.782	0.877	0.921
k-means++	Prop.	0.863	0.891	0.901	0.780	0.877	0.919

Table 1: *(a) Fidelity of generated stroke mappings (b) Fidelity of different stroke extraction methods.*

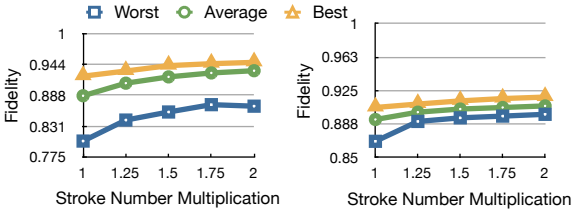

Figure 7: *Fidelity versus stroke number multiplication. First graph: setup 1 (pencils). Second graph: setup 2 (paints).*

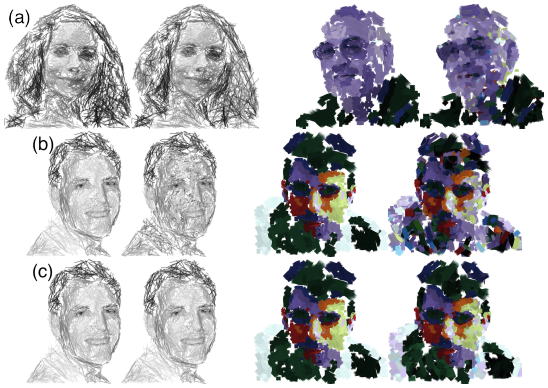

Figure 8: *(a) best repaintings (b) worst repaintings (c) improved results achieved with 2.0 stroke multiplications. The original pictures are on the left of each pair.*

paint dance will be judged as a whole, and the worst fidelity needs to be taken seriously. The worst repaintings for both paintings and drawings are given in figure 8(b), along with the original picture. Subjectively, we found that these were not of sufficiently quality to be included in the paint dances.

To improve matters, we experimented with extracting more than the minimum number of strokes required, i.e., if the largest number of strokes required to paint any one picture was n, we increased the number of strokes extracted to

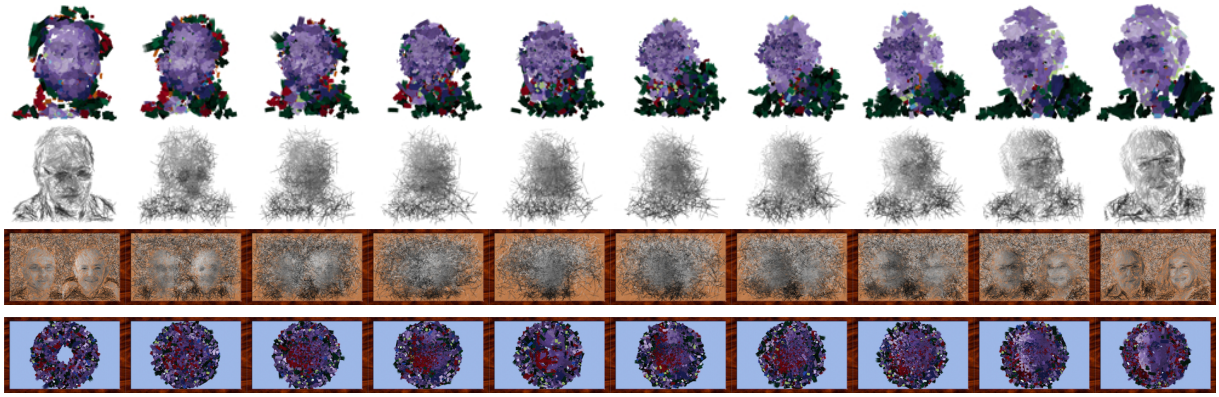

Figure 9: *Example paint dances using simulated acrylic paints; example paint dance using simulated graphite pencils; stills from the 'Meeting of Minds' video piece; stills from the 'Eye to Eye' video piece.*

$1.25n, 1.5n, 1.75n$ and $2n$ to see how much fidelity would be improved over the set of pictures. The improvement in fidelity – as recorded in the charts of figure 7 – is due to having more strokes to choose from for each picture, with the payoff being the difficulty of handling the surplus strokes during the paint dances (as described in section 6). We see that in both rendering setups, there is a marked increase in the worst fidelity when the number of extracted strokes is increased to $1.25n$. The increases for the larger multiplicands are less marked, but overall, the quality of the repaintings improves. In figure 8(c), the improved versions of the worst fidelity repaintings are supplied, and we can see a marked improvement visually. For stroke multiplication factor 2.0, we were satisfied with all the repainted drawings, but still somewhat unsatisfied with a few of the repainted paintings. We will address this via further experimentation, and by producing different original paintings, because the unsatisfying repaintings align with outliers from the original paintings. In particular, the two portraits in figure 6 with orange faces have particularly bad repaintings, even with stroke multiplication factor 2.

6. Paint Dance Animations

To recap, we have implemented methods for (a) rendering paint/pencil strokes and storing them independently of the overall picture (b) extracting a representative set of strokes from those of a set of pictures, and (c) determining how to repaint each of the pictures using this representative set of strokes. Given a set of strokes and a set of stroke mappings to a set of n pictures, we specify an overall number of clock ticks, t, for the animation. The Painting Fool then constructs a series of still images of the strokes which at regular intervals (of t/n clock ticks), come together at the correct coordinates and at the correct orientations to repaint the next in the series of pictures. We then use the `ffmpeg` software (http://ffmpeg.org) to compile the stills into an animation.

There are numerous ways to specify the paths that the strokes take between repaintings. The simplest of these is to move each stroke in equal increments at each clock tick directly from its current position to the position required of it in the next repainting, and to similarly rotate the stroke in equal angles at each clock tick so that it ends in the correct orientation when it gets to its destination. Starting with the centre of the first and second pictures registered, figures 9(a) and 9(b) show two paint dances constructed in this way. We have enabled the user to specify delay ranges for the strokes leaving and joining a repainting, hence the paint strokes arrive and leave at slightly different times to each other, which appears more natural. Similarly, the user is able to specify a trail (semi-transparent renderings of the previous positions/orientations of the stroke) which gives a motion effect. The correct layering of strokes on top of each other is key to achieving high fidelity repaintings. This means that the render ordering of strokes has to change during a paint dance. Doing so all at once causes a jarring animation artefact, hence we move each stroke from its current list position to its target one in random intervals as it travels from one repainting to the next. This smoothes the transition, but is not an entirely satisfactory solution, as there are still occasional artefacts where one stroke bubbles up past another unnaturally. We are investigating ways to improve this situation.

As mentioned above, for our purposes, we chose to use a stroke multiplication factor of 2.0, in order to increase the fidelity of the portraits. This means that at any stage, only half the paint strokes are being used for a repainting, which begs the question of where to place the non-critical strokes. For the two examples in figures 9(a) and 9(b), the non-critical strokes are placed in the centre of the image, at the back, hence conveniently hidden. For our final paint dances – entitled "Meeting of Minds" and "Eye to Eye", with extracts in figure 9 – we have chosen to make a feature of the non-critical strokes. In particular, in the former case, the strokes are placed around the portraits randomly, and similarly move around randomly between repaintings. In the latter case, the strokes maintain a continually moving orbit around the portraits, moving into the centre when needed for a repainting, and back out when not.

7. Related Work

Our work is a form of morphing, as described in [GDCV98], where one image is transformed into another in a seamless transition, which has its origins in film art, dating back more than a century [BT97]. Our approach differs in that the digital images are first represented as a series of paint/pencil strokes in an NPR manner before the morphing occurs. Other researchers have mixed NPR techniques with animations, e.g., Hertzmann and Perlin use NPR methods to paint over successive video frames to produce animated pictures [HP00]. Their approach only applies paint to the areas in frames where the source video is changing, and uses optical flow techniques to morph the strokes. In more recent work, the authors of [Col05] describe a video paintbox for transforming video clips into artistically stylised animations using saliency-adaptive painterly renditions [CH06] of images to mitigate issues of artistically rendering video, such as inherent flickering. Our approach could possibly be approximated by recording a video while morphing one digital image into another, then applying the techniques of [HP00] or [Col05] to the resulting video frames. Other researchers have used 3D models to produce NPR renderings which are animated in real-time while the model is moved and rotated [MKT97]. For instance, in [LD06], the authors employ generative software normally used to produce photorealistic trees, plants and landscapes. This has an advantage over video because the software provides detailed geometric descriptions which support the automatic simplification and segmentation of the 2D images, hence allowing efficient application of NPR techniques, for instance to produce realistic watercolour renditions of animated scenes, as in [LD06].

8. Conclusions and Future Work

We have developed and tested methods for matching sets of paint/pencil strokes so that a series of pictures can be repainted using only one set of strokes. This has facilitated the production of so-called *paint dances* as animations of paint strokes which occasionally form portraits in a sequence. The main difficulty lay in maintaining an acceptable level of fidelity with the repaintings, and we showed that the exhaustive-greedy approach performed well alongside a K-means clustering of the strokes, especially with an increase in the number of strokes available. We showed that the way in which strokes are sampled from the clusters does not particularly effect the fidelity of the repaintings. The 'Eye to Eye' and 'Meeting of Minds' video pieces mentioned above have been produced at full resolution (1920×1200) to be played on an iMac computer, with samples available at www.thepaintingfool.com/galleries/paint_dances. However, there are many other approaches we can envisage which might improve upon the stroke matching and extraction processes. In particular, we plan more experimentation with variations of the methods here, for instance, changing the order of the stroke matching, so that strokes in more salient regions are matched first, and including rotation in the weighted sum estimate of stroke likeness. As suggested by a reviewer, it may be better to find a stroke set for producing all portraits at the start of the pipeline, eliminating much work later on. However, this would mean automatically determining how good the original portraits are, which raises difficult technical issues of its own. As described in [Col08], our ethos is to continually *climb the meta-mountain* by handing over creative responsibility to the software. Hence, part of our future work will involve enabling The Painting Fool to create its own animations with no aesthetic guidance.

Acknowledgements

We would like to thank the anonymous reviewers for their very valuable comments, and the organisers and participants of the 2009 Computational Creativity seminar at Dagstuhl, for their inspiring research, conversations and images.

References

[AV07] ARTHUR D., VASSILVITSKII S.: K-means++: the advantages of careful seeding. In *Proc. of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007).

[BT97] BORDWELL D., THOMPSON K.: *The power of Mise-en-scene in Film Art, an Introduction*. McGraw Hill, 1997.

[CH06] COLLOMOSSE J., HALL P.: Salience-adaptive painterly rendering using genetic search. *Intl. Journal on Artificial Intelligence Tools (IJAIT) 15(4)* (2006), 551–576.

[Col05] COLLOMOSSE J., ROWNTREE D., HALL P.: Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics 11(5)* (2005).

[Col08] COLTON S.: Creativity versus the perception of creativity in computational systems. In *Proc. of the AAAI Spring Symposium on Creative Systems* (2008).

[CVP08] COLTON S., VALSTAR M., PANTIC M.: Emotionally aware automated portrait painting. In *Proc. of the 3rd Int. Conf. on Digital Interactive Media in Entertainment and Arts* (2008).

[GDCV98] GOMES J., DARSA L., COSTA B., VELHO L.: *Warping and Morphing of Graphical Objects*. Morg. Kauf., 1998.

[GG01] GOOCH A., GOOCH B.: *Non-photorealistic Rendering*. AK Peters, 2001.

[HP00] HERTZMANN A., PERLIN K.: Painterly rendering for video and interaction. In *Proc. of the International Symposium on Non-Photorealistic Animation and Rendering* (2000).

[Kra02] KRAUSE J.: *Colour Index*. HOW Design Books, 2002.

[LD06] LUFT T., DEUSSEN O.: Real-time watercolor illustrations of plants using a blurred depth test. *Proc. of the Int. Symp. on Non-Photorealistic Animation and Rendering* (2006).

[Mac67] MACQUEEN J.: Some methods for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability* (1967).

[MKT97] MARKOSIAN L., KOWALSKI A., TRYCHIN S., BOURDEV D., GOLDSTEIN D., HUGHES J.: Real-time non-photorealistic rendering. In *Proc. of SIGGRAPH* (1997).

[SS02] STROTHOTTE T., SCHLECHTWEG S.: *Non-Photorealistic Computer Graphics*. Morgan Kaufmann, 2002.

[Whi91] WHITFIELD S.: *Fauvism*. Thames and Hudson, 1991.