Evolutionary search for the artistic rendering of photographs

J. P. Collomosse

Department of Computer Science, University of Bath, Bath, U.K.
jpc@cs.bath.ac.uk

This chapter explores algorithms for the artistic stylisation (transformation) of photographs into digital artwork, complementing techniques discussed so far in this book that focus on image generation. Most artistic stylisation algorithms operate by placing atomic rendering primitives "strokes" on a virtual canvas, guided by automated artistic heuristics. In many cases the stroke placement process can be phrased as an optimisation problem, demanding guided exploration of a high dimensional and turbulent search space to produce aesthetically pleasing renderings. Evolutionary search algorithms can offer attractive solutions to such problems.

This chapter begins with a brief review of artistic stylisation algorithms, in particular algorithms for producing painterly renderings from two-dimensional sources. It then discusses how genetic algorithms may be harnessed both to increase control over level of detail when painting (so improving aesthetics) and to enhance usability of parameterised painterly rendering algorithms.

1 Introduction to Artistic Rendering

Research in Computer Graphics has traditionally focused on attaining *photorealism*; simulating physical interactions between light and modelled objects to produce scenes lit in an ostensibly natural manner. Over the past decade the development of rendering styles outside the bounds of photorealism has gathered significant momentum — so called *non-photorealistic rendering* or *NPR*. In particular, algorithms for generating artistic renderings for the purpose of aesthetics (for example pen-and-ink hatchings [1, 2], or paintings [3, 4]) have received considerable attention from NPR researchers. Artists typically draw not only to convey scene content, but also to convey a sense of how that content is to be perceived. Artistic renderings therefore offer numerous advantages over photorealistic imagery, including the ability to stylise presentation, abstract away unimportant detail and focus the viewer's attention.

The development of automated artistic rendering algorithms arguably began in the early nineties with Paul Haeberli's machine assisted painting environments [5]. Despite the advances in artistic media modelling made during the late eighties, Haeberli observed that convincing impressionist style paintings remained unobtainable in digital paint systems. Blaming the time overhead required to select new colours from the palette, Haeberli devised a novel stroke based rendering system based on source image point sampling.

In Haeberli's system the user starts with a source photograph that they wish to paint, and is supplied with a virtual canvas on which to produce the painting. The user then moves a cursor over the virtual canvas creating brush strokes as is common in typical digital paint systems. The colour of the brush stroke is determined by point sampling the source image at the location of the cursor. Stroke orientation may also be determined automatically by point sampling intensity edge gradient (computed using a Sobel operator) while other brush attributes, such as brush size and style, can be varied interactively by the user. A painting is thus represented as an ordered list of strokes, each stroke exhibiting a mixture of user and system defined attributes. This semi-automatic painting approach was judged highly effective by users who were able to construct renderings quickly and easily. Thereafter, Haeberli's combination of source image point sampling and stroke based rendering became something of an NPR paradigm and has been incorporated in to most painterly rendering algorithms subsequently developed.

1.1 Fully automatic painting algorithms

The novelty and high aesthetic quality of Haeberli's renderings prompted research efforts into both alternative artistic styles for NPR, and further automation of the painting process (for example, to facilitate painterly animations). In this chapter we focus on the topic of painterly rendering, and so review only the latter body of work.

The earliest attempts to derive fully automate the stroke placement process simply substituted the user interactive components of Haeberli's system (e.g. brush stroke size, or stroke painting order) with pseudo-random processes [6]. This led to a disappointing loss of detail in paintings, as brush strokes from visually important (often termed "salient") image regions tended to become overlapped and obscured by strokes from unimportant (non-salient) regions. To mitigate against this problem, various image data driven algorithms were developed that place strokes and modulate their attributes according to image content. These fully automatic algorithms encode procedural heuristics designed to emulate the human artistic process.

The first heuristics appearing in the literature were driven by low-level image processing operators. Litwinowicz presented an algorithm for painterly rendering using rectangular brush strokes [3] which were clipped against thresholded Sobel edges detected in the image. As with Haeberli's system, strokes were oriented tangential to edge gradient. Litwinowicz's system was

Fig. 1. Paintings produced interactively using Haeberli's impressionist system. The user clicks on the canvas to create strokes, the colour and orientation of which are point sampled from a reference photograph (inset). Reproduced from [5]. Courtesy Paul Haeberli. Copyright 1990 ACM.

thus prevented from painting "outside the lines" of edge features in images, so avoiding the problematic loss of detail caused by earlier pseudo-random systems. Statistical image measures were employed by Treavett and Chen [7], and later by Shirashi and Yamaguchi [8]. These systems compute statistical moments of pixel intensity within a small pixel window local to a stroke's intended position. Strokes are then painted tangential to the axis of least variance within that window.

In these early automated painterly algorithms, the generated "brush strokes" were little more than textured stamps centred, rotated and composited on canvas as directed by the painterly rendering algorithm. The late nineties saw the development of a new generation of algorithms that paid greater attention to the complexities of stroke placement. Hertzmann developed a painting algorithm that made use of curved spline brush strokes fitted to strong Sobel edge detected in the source image. Stroke control points were formed by modelling a particle that "hopped" between pixels along stroke image edges in a similar fashion to the algorithm described later in subsection 2.2.2. Hertzmann also adopted a multi-resolution approach to producing his paintings, initially producing a coarse scale painting using large strokes painted on a heavily sub-sampled image. The painting was then refined iteratively by painting increasingly finer strokes on canvas using decreasingly sub-sampled versions of the source image. Curved strokes were also used in Curtis *et al.*'s watercolour painting system [9], accompanied by a sophisticated model of watercolour pigment and substrate.

1.2 More sophisticated approaches to painterly rendering

Most of the painterly rendering algorithms developed in the nineties can be characterised as spatially local, non-linear filtering operations. Strokes are positioned on canvas solely on the basis of information extracted from a small pixel neighbourhood centred upon the stroke's location. This observation was

made explicit by Hertzmann et al.'s "Image Analogies" system [10] which, when presented with a photograph and a painterly rendering of that photograph, was able to learn painterly rendering transformations by example. This was accomplished by modelling the mapping between corresponding pixel windows in each image. Simple linear transformations such as sharpening or blurring could also be learnt in this manner.

At that stage of development, painterly rendering heuristics focused on the preservation of high frequency detail (e.g. edges or fine texture) to mitigate against loss of salient image content during the painting process. For example in Hertzmann's "Paint by Relaxation" systems [11] (discussed in subsection 2.1), paintings were created via an optimisation process in which the objective function minimised discrepancies between detail in the original and painted images. This tends to produce a machine-generated signature in the resulting painterly renderings; human painters do not seek to preserve all content when rendering a scene, but rather paint to emphasise the perceptually *salient* detail in that a scene whilst abstracting away non-salient details. The emphasis placed on a particular region within a painting is therefore a function of the relative importance of that region to the artist. Such observations have most recently motivated a trend away from use of local low-level image processing operators towards the incorporation of mid-level computer vision techniques in stroke placement heuristics (in particular, image salience measures [12] and colour segmentation algorithms [13, 14]). Notably, DeCarlo and Santella produced a segmentation based painting system [14] in which an eye tracker was used to monitor a user's interest in regions of a source photograph, so producing an salience map of the image interactively. Presenting a user with a photograph would cause his or her gaze to automatically fixate upon perceptually salient regions of the image. The location and duration of these fixations governed level of detail in DeCarlo and Santella's painting process. In Section 2.2 we discuss a fully automatic approach to producing salience adaptive painterly renderings [12].

We conclude this review by observing that various attempts have been made to produce painterly animations from video. This is a challenging problem; the presence of video noise or process non-determinism in painterly rendering algorithms frequently induces a distracting flickering or in the painted animation that prohibits the independent rendering of video frames. A naive solution to this problem is to fix the positions of brush strokes, allowing only their colours to change. This gives the impression of video moving "behind a shower door" [15] and is aesthetically poor in most cases. Litwinowicz was the first to produce a solution, making use of *optical flow* algorithms to estimate the motion of pixels from one video frame to the next [3]. The idea is straightforward, paint strokes are generated on the first frame of video and translated to subsequent frames based on the inter-frame motion estimate. This technique was applied in parts of the movie "What dreams may come", which won an Academy Award for "Best Visual Effects" in 1998. Unfortunately optical flow estimates are often inaccurate in general video, and the errors that

Fig. 2. Curved B-spline stroke paintings produced by Hertzmann's original, single-pass algorithm [4] (left) and by Hertzmann's active contour (snake) optimisation process [11] (right). Note the improvements in stroke placement precision using the latter. Reproduced from [10]. Courtesy Aaron Hertzmann. Copyright 2001 IEEE.

accumulate and propagate throughout the animation require extensive manual correction to prevent flicker in videos of a practical size. Other temporally local algorithms, making use of inter-frame differencing rather than optical flow, were presented in [16]. Most recently, researchers have begun to explore the avenue of spatio-temporal optimisation in video for painterly rendering, with some promising results addressing the problem of stroke flicker [17, 18].

2 Painting as an Optimisation Problem

Most painting rendering algorithms treat painting as a "single-pass" process; pixels in the image are examined in turn, and strokes placed according to various artistic heuristics. Although the image might be processed repeatedly at different scales (producing successive coarse to fine layers of strokes [4, 8]), once a particular stroke is placed there is no subsequent adjustment of its position or attributes to improve the painting in a more global sense. Recently researchers have begun to address this shortcoming by phrasing the painterly rendering process as a global goal-directed search (optimisation) problem.

2.1 Paint by Relaxation

Although optimisation approaches to painting were first suggested by Haeberli [5] it was not until a decade later that the first algorithmic solution was presented by Hertzmann [10]. Hertzmann extended his single-pass curved stroke painterly algorithm [4] by treating each stroke as an active contour or "snake". Snakes are a computer vision innovation developed in the late

eighties by Kass *et al.* for the purpose fitting of curves to edges detected in images [19]. A snake is typically a piecewise curve whose control points are iteratively updated to minimise an energy function; a process termed *relaxation*. Ideally, this has the effect of moving the curve incrementally closer to the edge over time. The energy function for a snake is a weighted sum of "internal" energy parameters, guarding against sharp discontinuities or "kinks" appearing along the curve, and "external" energy parameters which serve to minimise distance between the curve and edges detected in the image.

In Hertzmann's optimisation system, a single painting is created from the source photograph and iteratively updated to converge toward an aesthetic ideal. Strokes are placed in their initial positions on canvas using an existing curved stroke painting algorithm [4]. An iterative optimisation phase then begins in which snake strokes are added, deleted or moved (relaxed) over the canvas to minimise an objective function that evaluates the quality of the painting. Hertzmann deems a high quality painting to be one that matches the source image as closely as possible, using a minimal number of strokes but covering the maximum area of canvas in paint. His objective function is a summation of four weighted scalar functions which assess the current painting 'P' with respect to these attributes.

$$E_{app}(P) = \omega_1 \sum_{x=1}^{\text{width}} \sum_{y=1}^{\text{height}} |\underline{P}(x,y) - \underline{G}(x,y)| \tag{1}$$

$$E_{area}(P) = \omega_2 \sum_{S \in P} \text{Area}(S) \tag{2}$$

$$E_{nstr}(P) = \omega_3 \cdot (\text{number of strokes in } P) \tag{3}$$

$$E_{cov}(P) = \omega_4 \cdot (\text{number of unpainted pixels in } P) \tag{4}$$

Weights $\omega_{1..4}$ control the influence of each quality attribute and are determined empirically. Vector functions $P(x,y)$ and $G(x,y)$ refer to RGB pixel colour at point $[x,y]^T$ in the painting and source photograph respectively. Expression $S \in P$ refers to all strokes comprising painting P. Summing the areas of strokes in equation (3) yields a value analogous to the quantity of paint used in producing the painting.

Figure 2 demonstrates the significant improvements in accuracy that can be obtained using an optimisation approach to painting. However the consistently high level of detail returned within the painting can cause the rendering to tend back toward photorealism, and is arguably inconsistent with the selective process of abstraction with which artists paint. Furthermore the snake relaxation process is prone to falling into local minima. Artifacts may appear in paintings where snake relaxation falls into local minima causing strokes to be painted erroneously across salient features in the image. In most cases this can be resolved by tweaking weights $\omega_{1..4}$ and re-running the optimisation. Snake relaxation is also computationally expensive, and this places significant limitations on the usability of the system for experimentation and exploration

of potential artistic styles. A typical painting containing tens of thousands of snake strokes can take many hours to optimise.

2.2 A Search for Salient Paintings

Art historian E.H. Gombrich writes that "works of art are not mirrors" [20] — rather, artists commonly paint to capture the structure and elements of a scene that they consider to be visually important (salient). In an addendum to Hertzmann's "Paint by relaxation" paper (subsection 2.1) users could draw masks over regions of the image to reduce the influence of equation (2), so interactively attenuating non-salient detail in these areas (a precursor to DeCarlo and Santella's gaze driven painting system, section 1.2). In this section we describe an algorithm for rendering images in an impasto oil painted style, *automatically* identifying salient regions in the source image and concentrating painting detail there. The algorithm was developed by Collomosse and Hall [12] and extends an earlier pilot study [21] which demonstrated that ordering the placement of virtual brush stroke with respect to image salience can enhance both accuracy and sense of composition within a painterly rendering.

Collomosse and Hall's algorithm makes use of a Genetic Algorithm (GA) to search the space of possible paintings, so locating the optimal painting for a given photograph. Their *optimality criterion* is a measure of the strength of correlation between the level of detail in a painting and the salience map of its source image (later defined more formally — equation 14). Their GA approach was motivated through consideration of Haeberli's abstraction of a painting; an ordered list of strokes [5] (comprising control points, thickness, etc. with colour as a data dependent function of these). Under this representation the space of possible paintings for a given source image is very high dimensional, and the aforementioned optimality criterion makes this space extremely turbulent. Stochastic searches that model evolutionary processes, such as GAs [22], are often cited among the best search strategies in such situations; large regions of problem space can be covered quickly, and local minima more likely to be avoided [23, 24]. Furthermore the GA approach adopted allows different regions within a painting to be optimised independently, and later combined to produce improved solutions in later generations.

We now outline the trainable salience measure that forms the basis of the fitness function for the GA. The GA is then described in subsection 2.2.2.

2.2.1 A Trainable Image Salience Measure

Image salience measures map a colour image to a scalar field, in which the value of any point is directly proportional to the perceived salience of the corresponding image point. This scalar field describes the importance (salience magnitude) of image regions and is referred to as a "salience map". Producing a salience map is a subjective task; for example, different faces photographed in a crowd will hold different levels of salience to friends or strangers. A priori

user training is one way of incorporating this notion of subjectivity into an automated salience measure. The salience measure used by Collomosse and Hall [25] is trainable, and combines three operators which compute the *rarity*, *visibility* and the *classification* of local image artifacts.

The first operator (P_{rare}) performs unsupervised global statistical analysis to evaluate the relative rarity (P_{rare}) of image artifacts, a technique inspired by Walker *et al.* [26] who observe that salient features are uncommon in an image. However, not all rare artifacts should be considered "salient". In particular, salient artifacts should also be visible, and the salience measure incorporates a second, perceptually trained, operator which estimates the visibility (P_{visible}) of image artifacts. Finally, the user may perceive certain classes of artifact, for example edges or corners, to be more salient than others. The salience measure incorporates a third operator which users train by highlighting salient artifacts in photographs. Signals corresponding to these artifacts are clustered to produce a classifier which may be applied to artifacts in novel images in order to estimate their potential salience (P_{class}). The three operators are computed independently, yielding three probabilities which are combined to estimate the final probability of an image artifact being salient:

$$P_{\text{salient}} = P_{\text{rare}} \cdot P_{\text{visible}} \cdot P_{\text{class}} \tag{5}$$

The three operators are computed for each pixel location $\underline{p} = (i, j)^T$ in the source image by analysing a signal (hereafter written $\underline{x}(.)$) obtained via a circular sampling technique. Image data is sampled under a series of rings of radius ρ, each centred at $\underline{p}$. The image is uniformly sampled around each ring's circumference at angular positions θ, hence obtaining a discrete "circular" signal $\underline{x}(\rho, \theta; \underline{p}) \in \Re^3$; colours are in RGB space. Suitable sampling rates for ρ and θ are cited as 0.5 in range $[1, 3]$, and $\pi/16$ in range $[0, 2\pi]$ respectively.

Operator 1: Feature Rarity

Image rarity is computed by modelling the statistical distribution of a set of measures locally associated with each pixel, and isolating the outliers of this distribution. For each pixel location $\underline{p} = (i, j)^T$ the discrete image signal $\underline{x}(\rho, \theta; \underline{p})$ is rewritten as a column vector. An eigenmodel is created from the collection of vectors $\underline{x}(.)$ sampled over from each pixel within the image. The Mahalanobis distance $d(.)$ is then computed for all pixels $\mathcal{P}$ in the image.

$$d^2(\underline{x}(.)) = (\underline{x}(.) - \underline{\mu})^T \underline{\underline{U}}\, \underline{\underline{\Lambda}}\, \underline{\underline{U}}^T (\underline{x}(.) - \underline{\mu}) \tag{6}$$

The probability of an individual pixel $\underline{q} \in \mathcal{P}$ being rare is given by a quotient measuring the fraction of the sample density which is less rare than the pixel $\underline{q}$:

$$\mathcal{Q} = \{\underline{r} : d(\underline{x}(\underline{r})) \leq d(\underline{x}(\underline{q})) \wedge \underline{r}, \underline{q} \in \mathcal{P}\} \tag{7}$$

$$P_{rare}(\underline{q}) = \frac{\sum_{\underline{p} \in \mathcal{Q}} d(\underline{x}(\underline{p}))}{\sum_{\forall \underline{p} \in \mathcal{P}} d(\underline{x}(\underline{p}))} \tag{8}$$

Operator 2: Feature Visibility

The second operator is perceptually visibility measure, calibrated as a one-off process prior to processing. It is simplistically assumed that for each RGB colour $\underline{r}$ there is distance $\tau(\underline{r})$, also in RGB space beyond which it is possible to perceptually distinguish colours from $\underline{r}$. This unit of distance is termed the "just noticeable difference" (JND). Together the colour and the distance specify a sphere of RGB colours $(\underline{r}, \tau(\underline{r}))$. No colour interior to the surface of the sphere can be perceptually discriminated from the centre colour, whilst all exterior colours can be so discriminated. The distance $\tau(\underline{r})$ is one JND at the colour $\underline{r}$, and is measured experimentally using a process described in [25]. The sphere radius can vary depending on experimental conditions, and after several experimental trials τ emerges as the mean radius accompanied by an associated standard deviation σ.

To evaluate the visibility of artifacts local to a point $\underline{p}$, we compute the differential magnitude of circular signal $\underline{x}(\rho, \theta; \underline{p})$ as:

$$d(\rho, \theta; \underline{p}) = \left| \left| \frac{\delta \underline{x}(\rho, \theta; \underline{p})}{\delta \rho} \right|^2 + \left| \frac{\delta \underline{x}(\rho, \theta; \underline{p})}{\delta \theta} \right|^2 \right|^{1/2} \tag{9}$$

The probability $\phi(.)$ that this change is visible is computed as:

$$\phi(\rho, \theta; \underline{p}) = erf((d(\rho, \theta; \underline{p}) - \tau)/\sigma) \tag{10}$$

where τ and σ are the JND and its deviation for the colour sample at $\underline{x}(\rho, \theta; \underline{p})$ in the local window. Ideally, if a signal grazes the edge of the disc it should register as visible, but not strongly because it will not pass through all rings. The probability of the region local to location $\underline{p}$ being visible is thus:

$$P_{\text{visible}} = \sum_{\rho=1}^{\rho_{\max}} \max(\phi(\rho, \theta; \underline{p})) \tag{11}$$

Operator 3: Feature Classification

The final operator enables users to train the system to identify certain classes of low-level artifact as potentially salient. This not only introduces a subjective property to the salience measure but also enables the salience measure to classify the type of salient feature it encounters. This additional functionality is useful later (subsection 2.2.2) where it enables different types of stroke to be painted for different features such as edges or ridges.

Each ring in circular signal $\underline{x}(\rho, \theta; \underline{p})$ is processed separately, and later combined to produce a value P_{class} for location $\underline{p}$. Here we outline the classification process for a single ring (i.e. ρ is constant), and refer the reader to [25] for details, including the method for combining signals from multiple rings.

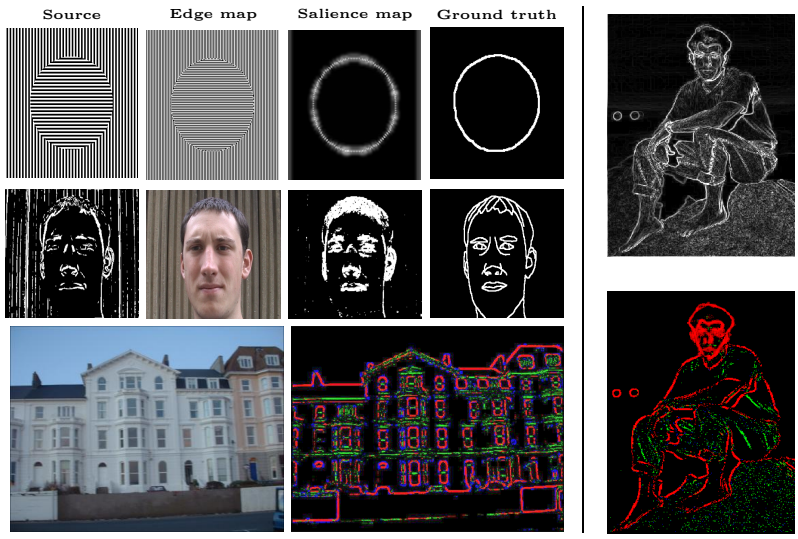

Fig. 3. Left: Comparison of images edge detected and salience mapped [25], with a hand-sketched ground truth. The salience maps are qualitatively closer to sketches, and can "pick out" the circle and face where local methods such as edge detection fail. Such examples motivate the use of global salience maps rather than local edge detection in the production of artistic renderings. The salience measure described in subsection 2.2.1 both estimates salience magnitude and classifies salient artifacts into trained categories (bottom row). In this example edges are drawn in red, ridges green, and corners blue. Right: Sobel edges (top) and salience map (bottom), corresponding to the MODEL painting included on the DVD. Salient edges are discriminated from non-salient high frequency texture, which allows the GA fitness function in subsection 2.2.2 direct level of painterly detail correctly.

The operator begins by transforming the circular signal $\underline{x}(\rho, \theta; \underline{p})$ into a useful invariant form amenable to feature classification. We simplify notation here by writing the periodic signal (at location $\underline{p}$ with constant ρ) as $\underline{y}(\theta)$. To obtain the invariant form, $\underline{y}(\theta)$ is transformed into the spectral domain using a Fourier transform. The magnitude (absolute value) of the Fourier components $|F[\underline{y}(\theta)]|$ are computed, normalised to unit power, and the d.c. component dropped to yield a feature vector $\underline{f}(\omega)$:

$$\underline{f}(\omega) = \frac{|F[\underline{y}(\theta)]|}{\left(\sum_\theta |\underline{y}(\theta)|^2\right)^{\frac{1}{2}}} \tag{12}$$

$$\underline{f}(\omega) \leftarrow \underline{f}(\omega) \setminus \underline{f}(0) \tag{13}$$

By disregarding phase the feature vector $\underline{f}(\omega)$ becomes rotationally invariant. In addition, dropping the d.c. component and normalising offers some invariance to changes of brightness and contrast over the image.

The classification operator requires the user to specify training examples of salient and non-salient features $\underline{f}(\omega)$, in order to build an *a priori* model of the features the user finds subjectively important. Precise details of the modelling process are beyond the scope of this chapter, but details may be found in [25]. In summary, this "training" occurs over several images, and requires the user to interactively highlight artifacts they regard as salient. The user may choose a number of classes of artifacts (such as edge, ridge, or corner), and identify a class label with each artifact they highlight.

During painting, classification of location $\underline{p}$ begins by sampling to obtain $\underline{f}(\omega)$. A probability vector is computed (one element per class of feature trained) to determine if the region local to $\underline{p}$ contains a potential salient feature. P_{class} is computed as the maximum value in this probability vector.

2.2.2 Initialising the Painting Population

Collomosse and Hall's system accepts as input a source image I; paintings derived from I are points in the search space. The algorithm begins by applying the salience measure to I; obtaining both a salience map and a classification probability for each pixel. An intensity gradient image is also computed using Gaussian derivatives, from which a gradient direction field is obtained. Once this pre-processing is complete, a fixed size population of 50 individuals is initialised. Each individual is a point in the search space, represented by an ordered list of strokes that, when rendered, produces a painting from I. Each individual (painting) is derived from the source image via a stochastic process.

Stochastic Stroke Placement and Growth

Seed points are scattered over the canvas, from which brush strokes will be subsequently "grown". Seeds are scattered stochastically, with a bias toward placement of seeds in more salient regions of source image I. A painting is formed by compositing curved Catmull-Rom spline brush strokes on a canvas of identical size to the source image. Brush strokes are grown to extend bi-directionally from each seed point; each end grows independently until halted by one or more preset criteria. Growth proceeds in a manner similar to Hertzmann's algorithm [4]. Starting from the pixel at given seed point, the algorithm "hops" between pixels in the direction tangential to their intensity gradient (Figure 4). The list of visited pixels forms the control points for the stroke.

Recognising that noise forms a component of any real image, Collomosse and Hall treat hop directions as samples from a stochastic distribution. Given a locally optimal direction estimate θ a hop direction is selected by adding Gaussian noise $G(0, \sigma)$. The value σ is an estimate for the level of noise in the image, and is calibrated using a procedure outlined elsewhere [12] (typical σ values vary between 2 and 5 degrees, depending on the imaging device that acquired the photograph). The magnitude of the hop is also Gaussian distributed, but inversely proportional to the local value of the salience map

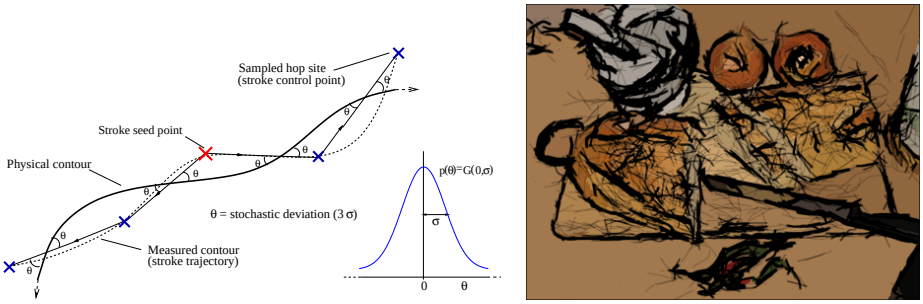

Fig. 4. Left: Stochastic growth of strokes from a seed pixel. The departure angle of a "hop" is drawn from distribution $p(\theta) = G(0, \sigma)$. Right: A "loose and sketchy" painting in the style of Matisse, rendered prior to the GA optimisation step of subsection 2.2.3. Note the variation in stroke style when rendering edges and ridges.

measured at the seed point of the stroke. The growth of a stroke end is halted when either the curvature between adjacent pixels, or the difference between the colour of the pixel to be appended and the mean colour of visited pixels exceeds a threshold. This method initially yields a sub-optimal trajectory for the stroke with respect to our optimality criterion. However, for a "loose and sketchy" paintings this is often desirable (see Figure 4, right).

Rendering an individual painting

At this stage it is possible to either render one of the paintings in the initial population (to produce a single-pass "loose and sketchy" painting), or proceed to subsection (2.2.3) to optimise the painting — each iteration of the latter process also requires paintings to be rendered to evaluate fitness.

Rendering a painting is a straightforward process of scan-converting and compositing its list of curved spline brush strokes. Stroke thickness is set inversely proportional to *stroke salience*; taken as the mean salience over each control point. Stroke colour is uniform and set according to the mean of all pixels encompassed in the footprint of the thick paint stroke. During rendering, strokes of least salience are laid down first, with more salient strokes being painted later. This prevents strokes from non-salient regions encroaching upon salient areas of the painting. The ability of our salience measure to differentiate between classes of salient feature (e.g. edge, ridge) also enables us to vary brush style styles. Figure 4 shows a painting where the classification probability of a feature has been used as a parameter to interpolate between three stroke rendering styles *flat*, *edge* and *ridge*.

2.2.3 Iterative search step of the GA

Genetic algorithms (GAs) simulate the process of natural selection by breeding successive generations of individuals through crossover, mutation and fitness-

proportionate selection [24]. In Collomosse and Hall's system such individuals are paintings; their genomes are ordered lists of strokes. A description of a single iteration (generation) of their GA search now follows. Iteration continues until the improvements gained over previous generations are marginal (the change in both average and maximum population fitness over a sliding time window falls below a threshold).

Evaluation and Fitness Function

The entire population is first rendered, and edge maps of each painting are produced by convolution with Gaussian derivatives, which serve as a quantitative measure of local fine detail. The generated maps are then compared to a precomputed salience map of the source image. The mean squared error (MSE) between maps is used as the basis for evaluating the fitness quality $F(.)$ of a particular painting; the lower the MSE, the better the painting:

$$F(I,\psi) = 1 - \frac{1}{N}\sum \left|S(I) - E(\Psi(I,\psi))\right|^2 \tag{14}$$

The summation is computed over all N pixels in source image I. $\Psi(.)$ is our painterly process, which produces a rendering from I and a particular ordered list of strokes ψ corresponding to an individual in the population. Function $S(.)$ signifies the salience mapping process described in subsection 2.2.1, and $E(.)$ the process of convolution with Gaussian derivatives.

The population is evaluated according to equation (14) and individuals are ranked according to fitness. The bottom 10% are culled, and the best 10% of the population pass to the next generation. The middle 80% are used to produce the remainder of the next generation — two individuals are selected stochastically using roulette wheel selection. These individuals are bred via crossover to produce a novel offspring for the successive generation.

Crossover and mutation

Two difference images, A and B, are produced by subtracting the edge maps of the parents from the salience map of the original image, then taking the absolute value of the result. Large values in these difference images (A and B) indicate large discrepancies between level of detail in the painting, and salient detail detected in the source image. These discrepancies are undesirable, given the fitness criterion defined in equation (14).

We define the $>$ (greater than) operator to act on images, outputting a binary image mask that indicates where pixels in one image hold larger values than those in corresponding locations in a second image. By computing the binary image $A > B$, and likewise $B > A$, it is easy to determine which pixels in one parent contribute toward the fitness criterion to a greater degree than those in the other. Since the primitives of paintings are thick brush

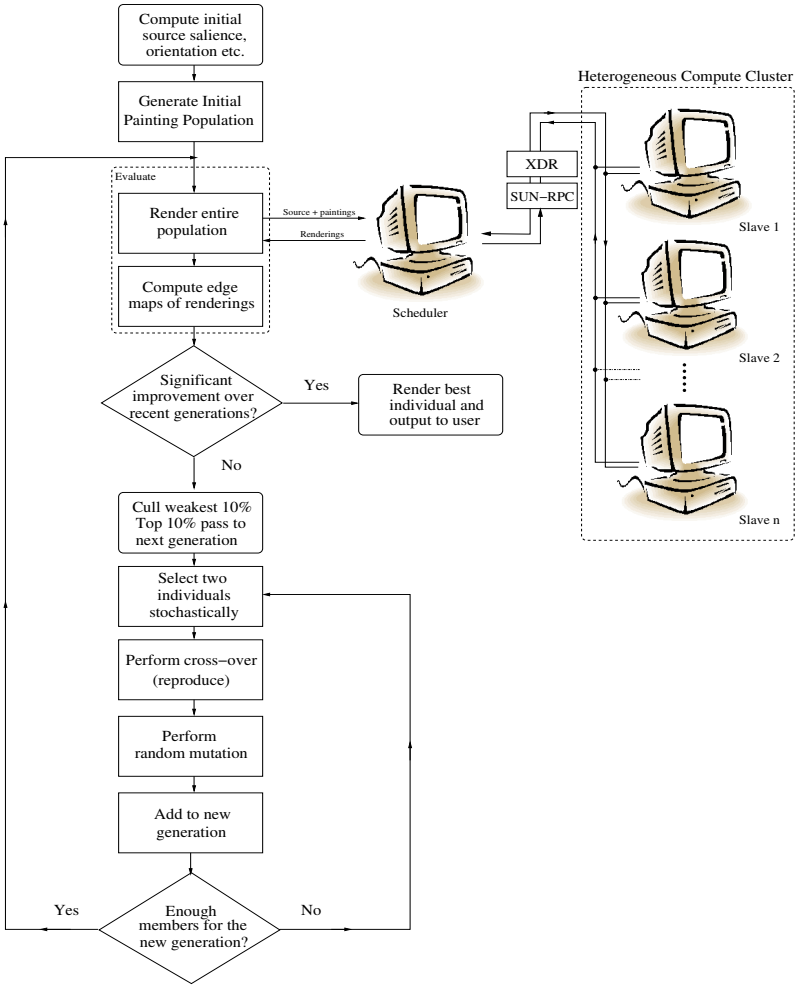

Fig. 5. Control flow in the genetic painting algorithm. Population evaluation is performed in parallel over a distributed computing cluster.

strokes rather than single pixels, we perform several binary dilations to both images to mark small regions local to these "fitter" pixels as desirable. A binary AND operation between the dilated images yields mutually preferred regions (i.e. where $A = B$). These conflicting regions are masked with a coarse chequerboard texture (of random scale and phase offset) to decide between parents in an arbitrary fashion. Strokes seeded within the set regions in each parent's mask are cloned to create the offspring individual (Figure 6).

When a bred individual passes to a successive generation it is subjected to a random mutation. A new "temporary" painting is synthesised (though never rendered), and a binary mask produced containing several small discs

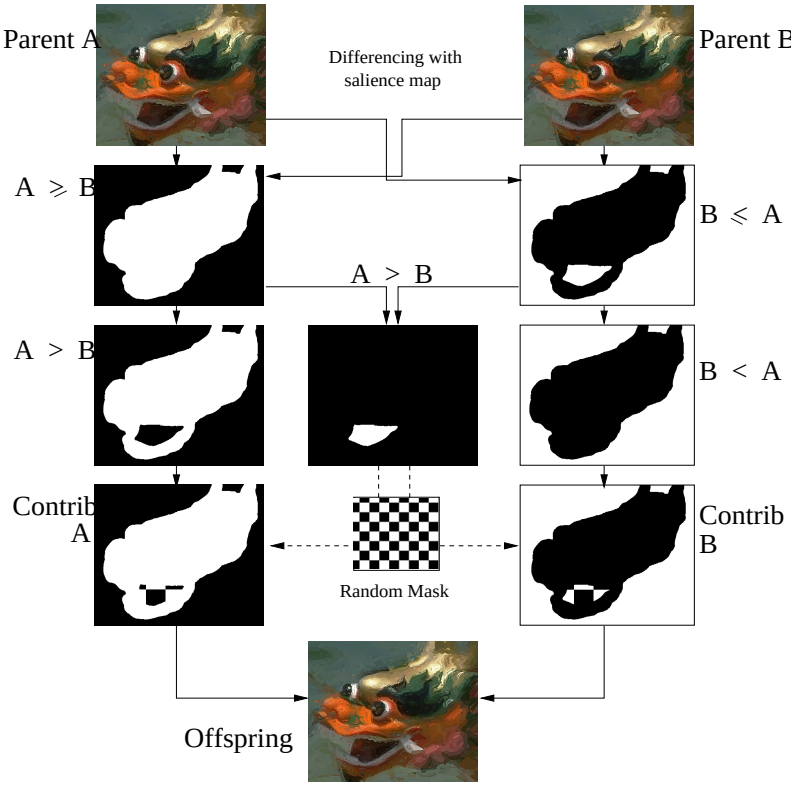

Fig. 6. Visualisation of crossover (using the DRAGON from Figure 8, manually marked up for illustration). Two parent paintings, A and B, are rendered and compared to the pre-computed salience map. If a region from one parent preserves salient detail to a greater degree than the corresponding region from the other parent, the former region's strokes are adopted by the new offspring — see subsection 2.2.3.

scattered within it. The number, location and radius of the discs are governed by random variates. Strokes seeded within set regions of the binary mask are substituted for those in the temporary painting; the temporary painting is then discarded. Mutation occurs over approximately 4% of the canvas.

2.2.4 Parallel Implementation

In practice, evaluation is the most lengthly part of the process and the rendering step is farmed out to several machines concurrently. In Collomosse and Hall's implementation, paintings are evaluated in parallel using the Sun RPC/XDR interface to communicate over a small heterogeneous (Pentium III/UltraSPARC) compute cluster. The typical time to render a 50 painting generation is cited as approximately 15 minutes over 6 workstations. Optimisation of the painting can therefore take in the order of hours, but significant

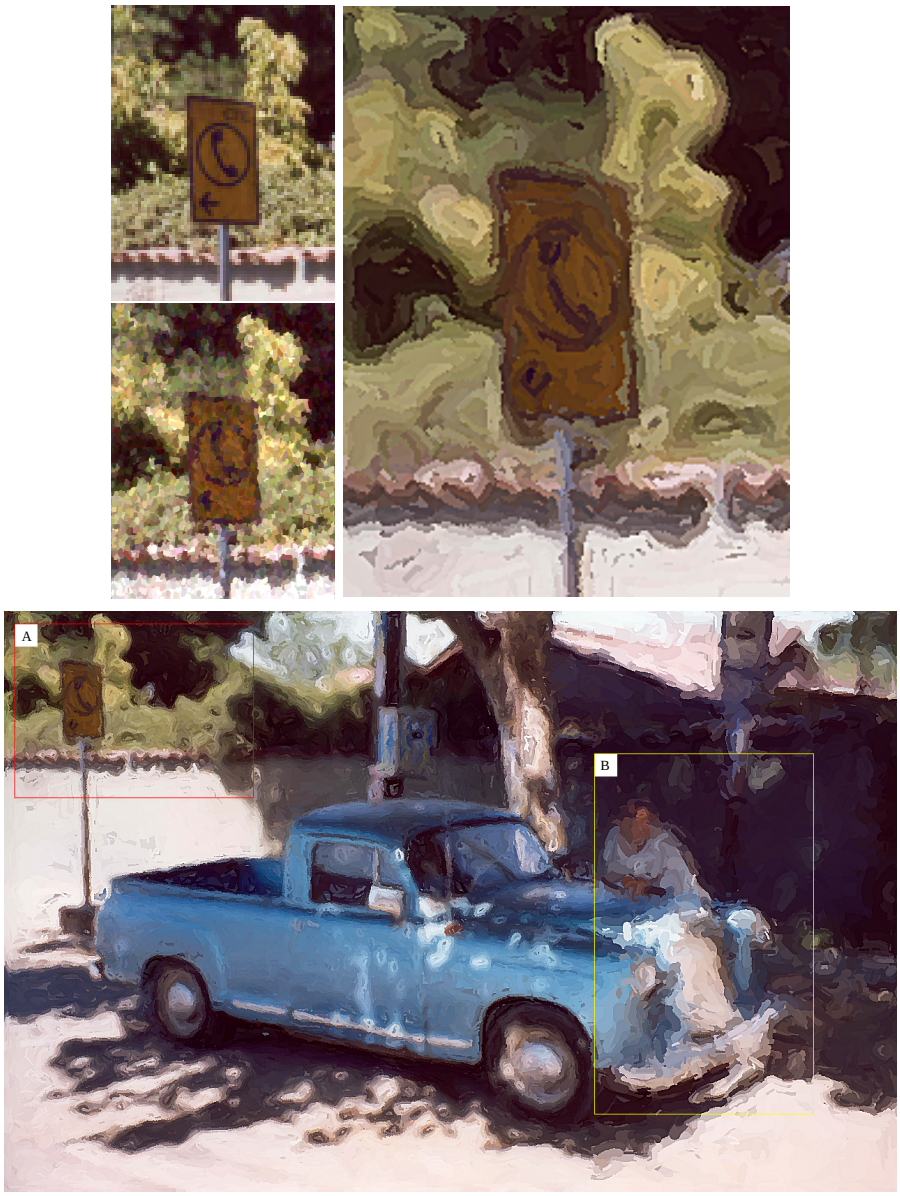

Fig. 7. Salience adaptive painting of the TRUCK image. Bottom row: salience map values within region **B** have been artificially reduced to demonstrate visual abstraction of detail. Top-left: region **A** source image exhibiting salient detail (sign) against non-salient detail (shrubbery). Top-middle: Litwinowicz's painterly rendering algorithm [3] affords equal emphasis to all features. Top-right: Collomosse and Hall's algorithm abstracts away non-salient detail with coarse brush strokes whilst preserving salient detail on the sign.

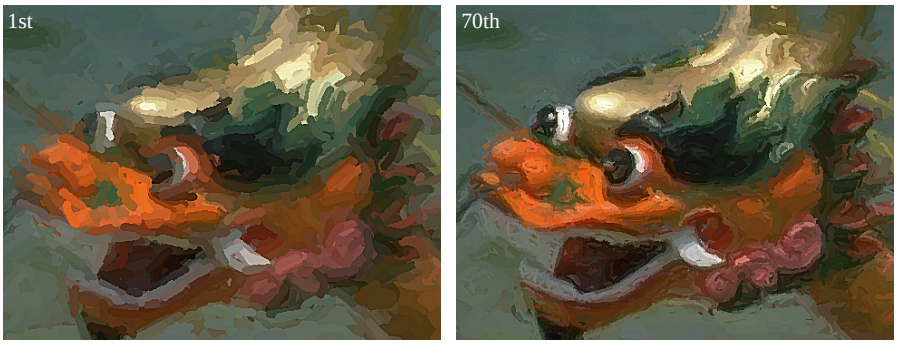

Fig. 8. Detail in the salient region of the DRAGON painting sampled from the fittest individual in the 1^{st}, and 70^{th} generation of the GA search.

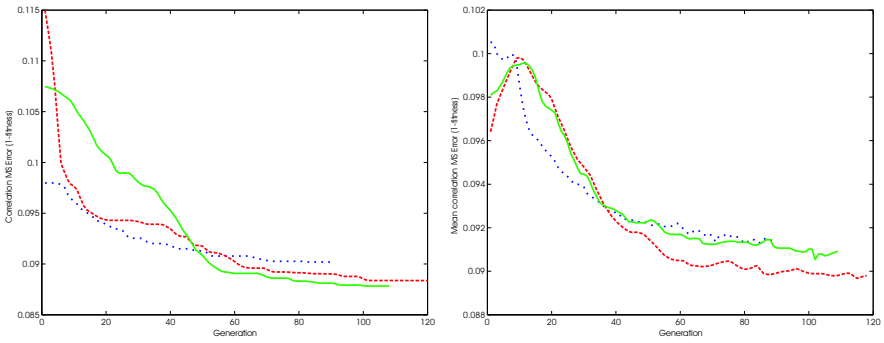

Fig. 9. Population fitness graphed during optimisation. Left: MSE of the fittest individual plotted against time. Right: MSE averaged over each generation. Red-dashed line indicates the DRAGON (Figure 8, video of optimisation on DVD), green-solid line the TRUCK (Figure 7), and blue-dotted line the MODEL (included on the DVD).

improvements in stroke placement can be achieved — see Figures 7,8. In particular note the variation in level of detail present in 7 as a function of the salience map defined in subsection 2.2.1. Additional paintings and a video of the optimisation process for Figure 8 are included on the DVD.

3 Interactive Genetic Search for Parameter Selection

Artistic rendering algorithms are commonly motivated as creative tools, for example helping to improve accessibility to art (perhaps allowing young children to produce digital paintings or create cartoon animations from videos of their toys [27, 18]), or permitting experimentation with new forms of dynamic art (such as artistic stylisation of digital video [17, 18]). In fully automated painterly rendering algorithms, creative control is expressed through setting

various user parameters governing internal operation of the algorithm. For example, Hertzmann's curved stroke painterly algorithm [4] contains many stylistic parameters controlling properties such as maximum stroke length or random colour jitter, as well as data dependent parameters controlling the scale at which images are filtered to detect edges. By manipulating these parameters it is possible to emulate a wide gamut of styles including "pointillist", "impressionist", "expressionist" and "abstract" paintings.

Unfortunately the large number of parameters that accompany many painterly rendering algorithms can be time consuming to set — both due to their number, but also due to their low-level nature, which can make them non-intuitive for inexperienced users to manipulate when aiming for a conceptually high level effect (for example, a dark, gloomy painting or an energetic, cheerful composition). Moreover, parameters can interact in complex ways leading to emergent behaviour within the painting that the user may not expect or understand. The end result is often a slow, iterative trial and error process before the user is able to instantiate their desired results.

This parameter selection problem can be overcome, as before, by framing the painterly rendering problem as a goal-centred evolutionary search [28], and resorting to a form of IEC system (see, e.g. Chap. **??**). A population of paintings is iteratively evolved towards a user's aesthetic ideal using a GA. The user is presented with a sample of the population for fitness evaluation in each evolutionary cycle. Through this interface the user affects selection in the GA, and so the composition of paintings in subsequent generations.

3.1 A fast segmentation-based painterly algorithm

Here we illustrate an interactive GA system for parameter selection using a simple, but fast, single-pass painterly rendering algorithm (from [28]). The algorithm operates by segmentation alone. A source image is first segmented into regions of homogeneous colour using the EDISON [29] algorithm. Each segmented region is rendered using a combination of "interior" and "boundary" strokes; each stroke is curved and takes the form of a spline, textured and bump mapped to give a 3D relief effect reminiscent of oil paintings [30].

Interior strokes are used to fill the interiors of regions, and are painted tangential to the principal axis running through the region (except for very large regions, which are rendered using horizontal straight strokes). Boundary strokes are painted around the exterior perimeter of a region vectorised using a standard contour walking technique (chain codes). The colour of brush strokes is sampled from the source image as with other painterly rendering algorithms. Care is taken when placing each stroke to prevent a) the stroke spanning pixel regions of greatly differing colour b) the stroke bending at too acute an angle. Both problems cause distortion and smearing of detail, and are avoided by fragmenting the stroke into multiple, smaller strokes.

Algorithm Parameterisation

There are eight parameters $p_{1..8} = [0, 1]$ governing this algorithm which, as with [4], are capable of creating significant stylistic variation in the output of the algorithm. A full description of the mathematical function of each parameter is beyond the scope of this text (but may be found in [28]). Instead we now summarise the function of these parameters. Figure 11 gives an indication of the various painterly effects attainable.

p_1 **Colour jitter:** The maximum distance in RGB space by which the colour of strokes may be randomly offset from their "true" colour sampled from the source image.

p_2 **Maximum stroke angle:** The maximum angle a spline stroke may bend during placement, before it becomes fragmented into smaller strokes. This tends to govern stroke size and expressiveness.

p_3 **Region turbulence:** Boundary stroke placement may be repeatedly performed, each time shrinking the area (and so the perimeter) of the region being painted. Under few repetitions, the interior of a region is comprised mainly of strokes oriented in a common direction. Under many repetitions, the interior becomes more chaotic, formed of series of concentric boundary strokes with reduced visual structure.

$p_{4,5}$ **Colour mood:** The colour of strokes is subjected to a transformation process, according to vector (p_4, p_5) which represents a point in Russell's 2D pleasure-arousal space [31]. Russell's space is used to represent emotional state; an emotion such as anger would be positioned low on the "pleasure" axis and high on the "arousal" axis, for example. A complex series of colour transformations, derived from the colour psychology literature, are performed on the original stroke colour according to the value of point (p_4, p_5). Interested readers can find further details in [28].

p_6 **Stroke jaggedness:** Brush strokes are splines, created by interpolating control points generated during the stroke placement process. In one possible extreme ($p_6 = 0$), interpolation is performed smoothly using Catmull-Rom splines; in the other ($p_6 = 1$) using only linear interpolation (so creating jagged strokes).

p_7 **Stroke undulation:** Sinusoidal variation can be introduced along each brush stroke, causing inaccuracy in stroke placement and conveying a different visual aesthetic to the painting. p_7 controls the magnitude of this variation.

p_8 **Region dampening:** The effects of stroke undulation parameter p_7 can affect either the interior or boundary strokes, depending on this parameter.

3.2 Interactive Evolutionary Search

Within this parameterised rendering framework, the painting process is reduced to a search for the point in parameter space $[p_1, p_2, ..., p_8] \in \Re^8$. The

genome of an individual in the GA search is thus represented as eight normalised scalar values. The system creates a population of 1000 such individuals, initially with random genome. As is explained in the next subsection, the user will explicitly evaluate only a fraction of this population on each generation (the remainder will be evaluated by extrapolation). The algorithm then enters an iterative stage in which individuals are bred to produce improved paintings. When successive improvements fall below a threshold, the algorithm terminates; in practice this usually takes only 20 to 25 iterations.

Interactive Evaluation

The first step in each iterative cycle is population evaluation, in which the proximity of each individual's phenotype to the user's "ideal" aesthetic in measured. Specifically we require a mapping $M([p_1 p_2 ... p_8]) \mapsto f \in \Re$ where f is a normalised fitness score; higher values correspond to aesthetically superior paintings. As our aim is to assist the user creatively in style specification it is not possible to write an automatic function for $M(.)$. Our objective is therefore twofold. First, to estimate the mapping function $M(.)$ through user interaction. Second, to search for the point $\underline{p} \in \Re^8$ that maximises $M(\underline{p})$.

The first of these problems can be addressed by sparsely evaluating $M(.)$ over a subset of the population, and use this data to extrapolate the behaviour of $M(.)$ over the entire population. A simple user interface, allowing the user to be prompted for the fitness of a given individual drawn from the population (so obtaining a sparse domain sample of $M(.)$). The user is supplied with a graduated colour bar, and asked to rate the aesthetics of the painting rendered from a given individual on a continuous scale spanning red (bad), amber (neutral) and green (excellent). The user is presented with the nine fittest individuals from the previous iteration, and asked to rate the individual that they feel most strongly about.

The sparse set of user interactions are transformed into a continuous estimate for $M(.)$. Each time a user evaluates an individual we obtain a point $\underline{q}$ and a user fitness rating $U(\underline{q}) = [-1, 1]$. These data are encoded by adding a Gaussian to a cumulative model, built up over successive user evaluations. Each Gaussian distribution is centred at point $\underline{q}$, and multiplied by the factor $U(\underline{q})$. The integral under the Gaussian is assumed to be well approximated by unity in space $\Re^8 \in [0, 1]$, and so the continuous function $M(.)$ is written:

$$M(\underline{p}) = 0.5 + \begin{cases} 0 & \text{if } N = 0, \\ \frac{1}{2N} \sum_{i=1}^{N} U(\underline{q}_i) G(\underline{p}, \underline{q}_i, \sigma) & \text{otherwise} \end{cases} \tag{15}$$

where $\underline{p}$ is an individual to be evaluated in the current generation, $\underline{q}_i$ are individuals evaluated by the user in the previous N iterative cycles, and $U(\underline{x})$ is the user's score of a given genotype $\underline{x}$. The function $G(\underline{x}, \underline{\mu}, \sigma)$ denotes a Gaussian distribution with mean $\underline{\mu}$ and standard deviation σ, evaluated at $\underline{x}$. The standard deviation σ governs the locality in problem space over which a single user evaluation holds influence; $\sigma = 0.1$ for typical results

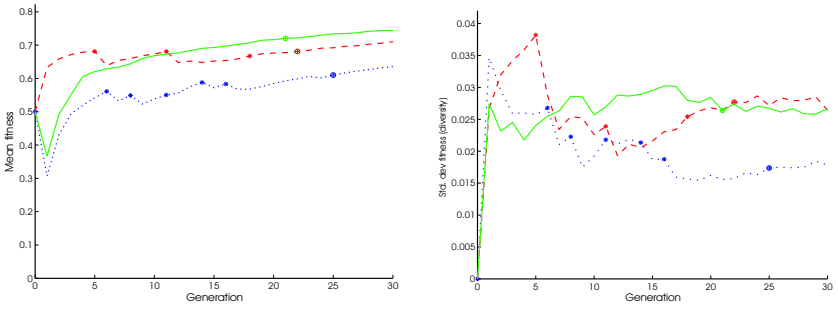

Fig. 10. Population statistics from three runs of the GA system, corresponding to Figure 11 top left (blue, dotted), top right (red, dashed), and bottom left (green, solid). Left: mean fitness over time. Right: fitness diversity over time. The + symbol indicates algorithm termination. $*$ indicates a negative fitness rating from the user.

Selection and Propagation

Once the current population has been evaluated, pairs of individuals are selected and bred to produce the next generation of painting solutions. Parent individuals are selected with replacement, using a stochastic process biased toward fitter individuals. A single offspring is produced from two parents via stochastic crossover and mutation operations. Each of the eight parameters that comprise the genome of the offspring has an equal chance of being drawn from either parent. Mutation is implemented by adding a random normal variate to each parameter. These variates have standard deviations of 0.1; 97% of mutations will produce less than ± 0.3 variation in a rendering parameter.

Improvements in Usability

The population statistics gathered during several runs of the GA search (Figure 10) show a steady improvement in fitness over time, punctuated by short-term dips. These correspond to the occasions when the model $M(.)$ does not tally with the user's aesthetic ideal, requiring correction, in the form of a negative ratings issued by the user. These dips become less pronounced over time as $M(.)$ more closely matches the users expectations.

In small scale usability studies conducted with this system, non-expert users were given a target aesthetic objective, for example "produce a happy, cheerful composition". Using the GA system users were able to manifest their desired aesthetic using approximately 20-25 mouse clicks (one click per generation), in an average of one minute. The same users were asked to re-create the results using a bank of 8 sliders (each controlling an independent painting parameter). The results were reproducible but usually took around 4-5 minutes due to the number of parameters and unexpected emergent visual properties caused by interactions between the parameters. The number of

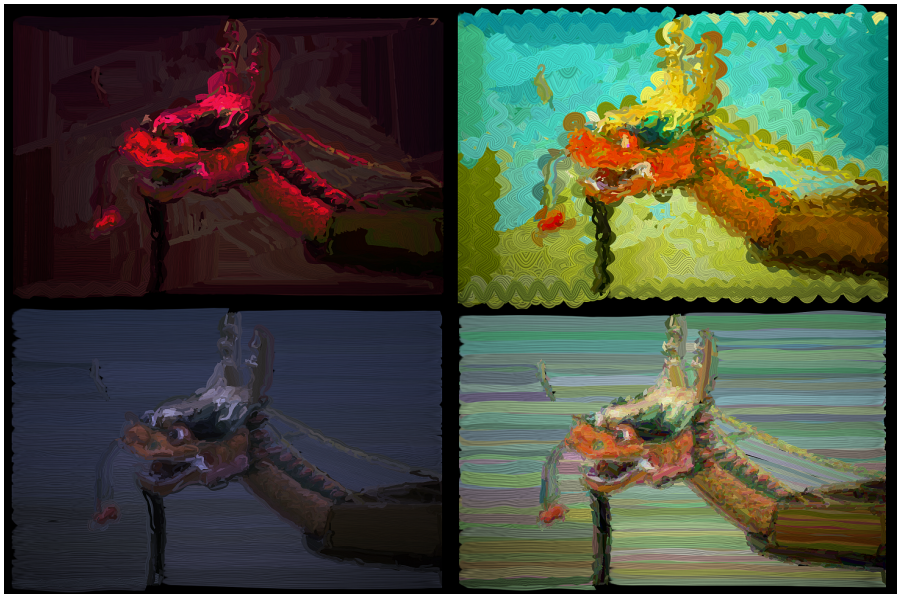

Fig. 11. A variety of paintings rendered from the DRAGON image using the algorithm outlined in subsection 3.1, parameterised using the interactive GA.

mouse clicks required to use the sliders was significantly larger than the GA system; between 100 to 200 interactions in all cases.

4 Summary and Conclusions

In this chapter we have explored the topic of artistic stylisation, presenting two algorithms that harness evolutionary search algorithms to produce synthetic oil paintings from photographs. We began by reviewing a brief history of Artistic Rendering algorithms, which evolved from artistic media emulation work in the mid-1980s, to semi-automated paint packages, to fully automatic stylisation algorithms in the late 1990s (Section 1). We described how these automatic painting processes can be phrased as optimisation problems; as a search to identify the "best" configuration and arrangement of paint strokes for a given source image (Section 2). We then provided a detailed exposition of Collomosse and Hall's evolutionary search algorithm that develops paintings where emphasis (expressed through level of brush detail) is focused upon the areas of visual importance or "image salience" (subsection 2.2).

Collomosse and Hall's algorithm [12] presented two key technical contributions: (i) a perceptually based measure of image salience; (ii) a genetic algorithm driven relaxation process that automatically produces "optimal" synthetic oil paintings under a definition derived from (i). Adopting a salience

adaptive approach to painting was shown to improve the aesthetics of renderings; abstracting away non-salient detail with coarse brush strokes and emphasising salient detail with fine strokes (Figure 7, 8). The ability of the salience measure to classify image artifacts (for example edges, ridges, or corners) was also harnessed to parameterise stroke style, yielding attractive artistic effects.

In addition to directly manipulating strokes during painting, we also discussed how genetic algorithms can be applied to aid user parameter selection for painting algorithms (Section 3.2). Often artistic rendering algorithms are capable of a wide gamut of styles, but the parameterisations of that gamut is counter-intuitive for non-expert users. By framing the painting problem as a goal-centred evolutionary search with interactive aesthetic evaluation (i.e. an IEC system) we described how users can efficiently control a painting algorithm without detailed knowledge of its underlying parameterisation.

Artistic Rendering is a comparatively young field within Computer Graphics, and much of the groundwork has been laid down only within the last 10-15 years. However over this period we have already observed an marked increase in the complexity of stroke placement algorithms; from simple random stroke placement [6], to image filtering and edge detection based techniques [3, 4, 8], to approaches drawing on sophisticated mid-level computer vision [13] and models of perceptual salience [14, 12]. Strong convergence trends are now emerging between Artistic Rendering and fields such as Computer Vision and Cognitive Science. As this cross-pollination of ideas yields increasingly complex stroke placement heuristics, contributing to new algorithms with diverse artistic capabilities, it is likely that evolutionary algorithms will continue to find application in the design and control of Artistic Rendering software.

References

1. Salisbury, M.P., Anderson, S.E., Barzel, R., Salesin, D.H.: Interactive pen-and-ink illustration. In: Proc. ACM SIGGRAPH. (1994) 101–108
2. Salisbury, M.P., Wong, M.T., Hughes, J.F., Salesin, D.H.: Orientable textures for image-based pen-and-ink illustration. In: Proc. ACM SIGGRAPH. (1997) 401–406
3. Litwinowicz, P.: Processing images and video for an impressionist effect. In: Proc. ACM SIGGRAPH. (1997) 407–414
4. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Proc. ACM SIGGRAPH. (1998) 453–460
5. Haeberli, P.: Paint by numbers: abstract image representations. In: Proc. ACM SIGGRAPH. Volume 4. (1990) 207–214
6. Haggerty, M.: Almost automatic computer painting. IEEE Comp. Graphics and Apps. **11** (1991) 11–12
7. Treavett, S., Chen, M.: Statistical techniques for the automated synthesis of non-photorealistic images. In: Proc. Eurographics UK. (1997) 201–210
8. Shiraishi, M., Yamaguchi, Y.: An algorithm for automatic painterly rendering based on local source image approximation. In: Proc. ACM NPAR Sympos. (2000) 53–58

9. Curtis, C., Anderson, S., Seims, J., Fleischer, K., Salesin, D.H.: Computer-generated watercolor. In: Proc. ACM SIGGRAPH. (1997) 421–430
10. Hertzmann, A., Jacobs, C., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: Proc. ACM SIGGRAPH. (2001) 327–340
11. Hertzmann, A.: Paint by relaxation. In: Proc. Computer Graphics Intl. (CGI). (2001) 47–54
12. Collomosse, J.P., Hall, P.M.: Genetic paint: A search for salient paintings. In: Apps. of Evolutionary Comp.. Proc. EvoMUSART, Springer LNCS. Volume 3449. (2005) 437–447
13. Gooch, B., Coombe, G., Shirley, P.: Artistic vision: Painterly rendering using computer vision techniques. In: Proc. ACM NPAR Sympos. (2002) 83–90
14. DeCarlo, D., Santella, A.: Abstracted painterly renderings using eye-tracking data. In: Proc. ACM SIGGRAPH. (2002) 769–776
15. Meier, B.: Painterly rendering for animation. In: Proc. ACM SIGGRAPH. (1996) 447–484
16. Hertzmann, A., Perlin, K.: Painterly rendering for video and interaction. In: Proc. ACM NPAR Sympos. (2000) 7–12
17. Wang, J., Xu, Y., Shum, H.Y., Cohen, M.: Video tooning. In: Proc. ACM SIGGRAPH. (2004) 574–583
18. Collomosse, J.P., Rowntree, D., Hall, P.M.: Stroke surfaces: Temporally coherent non-photorealistic animations from video. IEEE Trans. Visualization and Comp. Graphics **11** (2005) 540–549
19. Kass, M., Witkin, A., Terzopoulos, D.: Active contour models. Intl. Journal of Computer Vision (IJCV) **1** (1987) 321–331
20. Gombrich, E.H.: Art and Illusion. Phaidon Press Ltd., Oxford (1960)
21. Collomosse, J.P., Hall, P.M.: Painterly rendering using image salience. In: Proc. Eurographics UK Conf. (2002) 122–128
22. Holland, J.: Adaptation in Natural and Artificial Systems. An introductory analysis with applications to biology, control, and artificial intelligence. Univ. Michigan Press (1975) ISBN: 0-472-08460-7.
23. de Jong, K.: Learning with genetic algorithms. Machine Learning **3** (1988) 121–138
24. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA (1989) ISBN: 0-201-15767-5.
25. Hall, P.M., Owen, M., Collomosse, J.P.: A trainable low-level feature detector. In: Proc. Intl. Conf. on Pattern Recognition (ICPR). Volume 1. (2004) 708–711
26. Walker, K.N., Cootes, T.F., Taylor, C.J.: Locating salient object features. In: Proc. British Machine Vision Conf. (BMVC). Volume 2. (1998) 557–567
27. Agarwala, A.: Snaketoonz: A semi-automatic approach to creating cel animation from video. In: Proc. ACM NPAR Sympos. (2002) 139–147
28. Collomosse, J.P.: Supervised genetic search for parameter selection in painterly rendering. In: Apps. of Evolutionary Comp. Proc. EvoMUSART, Springer LNCS. Volume 3907. (2006) 599–610
29. Christoudias, C., Georgescu, B., Meer, P.: Synergism in low level vision. In: Proc. Intl. Conf. on Pattern Recognition (ICPR). Volume 4. (2002) 150–155
30. Hertzmann, A.: Fast paint texture. In: Proc. ACM NPAR Sympos. (2002) 91–96
31. Russell, J.A.: Reading emotion from and into faces: Resurrecting a dimensional-contextual perspective. In Russel, J.A., Fernández-Dols, J.M., eds.: The Psychology of Facial Expression. Cambridge Univ. Press (1997) 295–320