COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2013; **24**:43–64
Published online 11 May 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1435

RESEARCH ARTICLE

Painterly rendering techniques: a state-of-the-art review of current approaches†

Siddharth Hegde*, Christos Gatzidis and Feng Tian

Bournemouth University, School Of Design, Engineering and Computing, Talbot Campus, Poole House, Fern Barrow, Poole, Dorset, BH12 5BB, UK

ABSTRACT

In this publication we will look at the different methods presented over the past few decades which attempt to recreate digital paintings. While previous surveys concentrate on the broader subject of non-photorealistic rendering, the focus of this paper is firmly placed on painterly rendering techniques. We compare different methods used to produce different output painting styles such as abstract, colour pencil, watercolour, oriental, oil and pastel. Whereas some methods demand a high level of interaction using a skilled artist, others require simple parameters provided by a user with little or no artistic experience. Many methods attempt to provide more automation with the use of varying forms of reference data. This reference data can range from still photographs, video, 3D polygonal meshes or even 3D point clouds. The techniques presented here endeavour to provide tools and styles that are not traditionally available to an artist. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

graphics; non-photorealistic rendering; NPR; survey; painterly

***Correspondence**

Siddharth Hegde, Bournemouth University, School Of Design, Engineering and Computing, Talbot Campus, Poole House, Fern Barrow, Poole, Dorset, BH12 5BB, UK.
E-mail: shegde@bournemouth.ac.uk

1. INTRODUCTION

Unlike photographs or photorealistic images, painterly style paintings are traditionally created by artists who try to strategically place their brushstrokes to focus viewer attention on specific regions and create the perception of depth through the use of colours [1] and by changing the size, shape and placement of strokes [2]. Durand [3] suggested that the goal of the artist is to represent and convey properties through the use of a brushstroke and not to recreate them as seen in a photograph or photorealistic rendering. By avoiding too much detail, the artist aims to invoke our imagination [4] and allows us to fill in the gaps. For generations, the act of creating a painting could only be performed by artists whose skills are acquired over a long period of time. Furthermore, creating animated sequences manually in this style, the works in [5] and [6] require hundreds of man hours for every minute of output [7]. This style is distinctively dissimilar to most computer-generated animations we observe today, which generally have sharp boundaries and a mechanical look and feel that is different from hand-created paintings. In this paper, we will look at some techniques that attempt to create animated paintings quickly and easily.

Unlike previous surveys on non-photorealistic rendering (NPR) techniques [8,9], our focus is placed on the derived narrower field of specific methods that are able to produce painterly style output. Hence, techniques that cover NPR styles such as hatching, stippling and pen-and-ink will not be reviewed in this paper.

We start this paper by reviewing low-level techniques that try to simulate traditional artistic tools such as brushes, medium and surface. In the next section, we explore techniques that attempt to determine properties of individual brushstrokes automatically. Following this, we present methods that manage to generate painterly style output without explicitly creating individual strokes. An overview of this paper and a detailed hierarchical structure of the content covered can be seen in Figure 1. Once all stroke-based methods are reviewed, we present a table that summarises the different painterly styles each method is capable of producing. In the conclusion, we explore some open contemporary challenges in creating digital paintings.

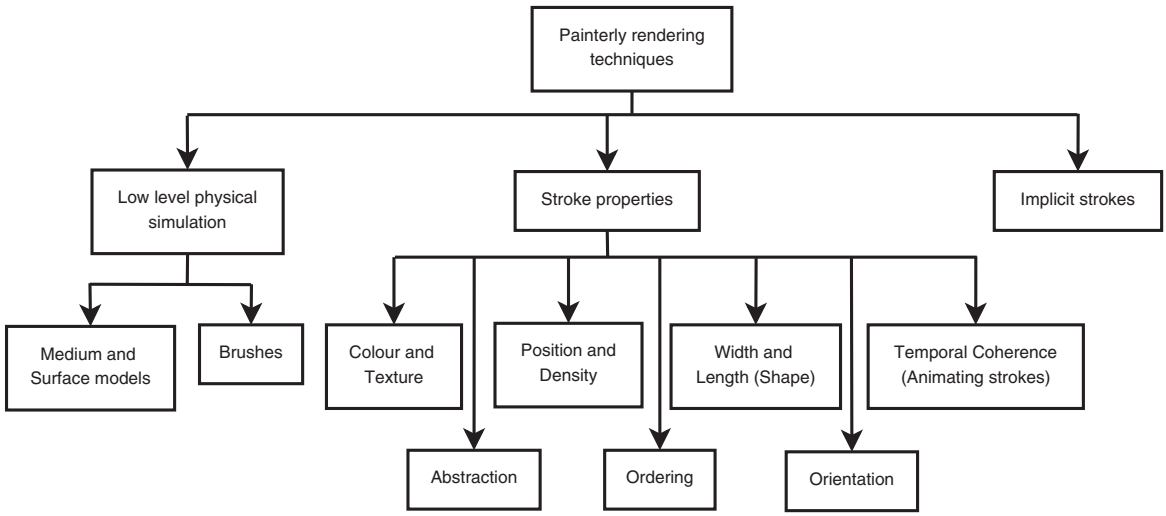

Figure 1. A hierarchical overview of this paper.

2. LOW-LEVEL PHYSICAL SIMULATION

A significant amount of research has been carried out in order to help artists carry their skills over to the digital world. These methods simulate traditional tools used in painting such as brushes, mediums and the interaction of the medium with the brush, canvas or surface.

2.1. Brushes

In this section, we look at different methods that attempt to simulate the physics of a real brush. Some methods focus on specific brush types, whereas others provide a more generic framework.

Strassmann [10] presented a set of techniques to simulate the brush and ink used to create a traditional Japanese sumi-e style painting. The author modelled a brush by using a one-dimensional array of bristles. The work in the paper presents algorithms to approximate the interaction of the individual bristles in a brush with (i) the ink, (ii) the neighbouring bristles and (iii) the paper as a stroke is painted. The bristles are all assigned a shade of grey upon performing the virtual act of dipping. A bristle can randomly obtain ink from neighbouring bristles even after it has run dry. Applying pressure on the brush causes the bristles to spread out. Because of the lack of an appropriate input device at the time, the path that a stroke takes must be created manually by defining a spline from a set of nodes. Every node must then be assigned properties such as pressure and time that the brush will reference when it reaches that node. Assigning properties to these nodes can at best be a tedious trial-and-error process, which a traditional artist is not accustomed to doing. Additionally, the algorithms presented in this research only handle grey ink tones.

Sousa and Buchanan [11,12] modelled the footprints of tools used in pencil drawings (pencil lead, blenders and erasers) as a series of vertices that define a polygon. An additional vertex defines the centre point of the footprint. The final shape of the footprint is scaled along an axis on the basis of the azimuth and elevation the tool makes with the surface (paper). Although each of these tools interact differently with the amount of pigment (lead material) deposited on the paper, the underlying interaction is based on the pressure at discreet points inside this polygon. Pressure coefficients are defined at each vertex and the centre point. The polygon is broken up into a triangle fan and the applied pressure is linearly interpolated to determine the pressure at arbitrary points inside the polygon.

In [13], Lee allowed the user to define the path of a stroke by moving a brush in real time. In addition to this, Lee's brush is based on a 3D model of the bristles. Here, the interaction of each bristle with the paper plane is based on the elastic properties of the individual bristles that make up the brush. The author derived two algorithms on the basis of the amount of force applied on the brush. One is based on a soft brush model to be used if the pressure on the brush is less than a critical force, whereas the second algorithm is used in all other cases. Furthermore, the paper presents a proof of concept system that runs in real time and basic methods to account for the paper texture and the diffusion of ink deposited on the paper. Unfortunately, there is a distinct limitation to shades of grey ink and a single type of brush used in oriental paintings.

Baxter *et al.* [14] also presented a real-time system that is based on 3D models of popular brushes used in acrylic and oil paintings. Moreover, their methods could be extended to other types of brushes as well. Their system is integrated with a haptic device that gives the user a sense of touch and pressure as the brush moves on the paper. In addition to this, the system allows the user to move the virtual brush with six degrees of freedom, allowing artists to

Comp. Anim. Virtual Worlds 2013; **24**:43–64 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/cav

quickly transfer their skills into the digital domain. Their application makes visual deformations to the brush head on the basis of concepts from cloth dynamics, although the sense of pressure is provided by a set of faster piecewise linear functions. The authors also simulated the interaction of paint already deposited on the paper by accounting for common painting techniques such as blending, drying, glazing and bidirectional paint transfer between ink on the paper and on the brush. Finally, the authors also allowed coloured inks to be used, thereby providing significant improvements over previously reviewed methods.

Zwicker *et al.* created an application that can be used to paint on point-based surfaces [15]. They presented a general framework that allows users to create brushes that can be employed to change attributes of the points, such as their position, orientation and colour. In the case of painting, the user initially provides a global parameterisation of the surface by matching points on the surface to a texture map. When a brush interacts with the surface, a local parameterisation is calculated by approximating a plane in the neighbourhood of the points. The points are orthogonally projected onto this local plane and are assigned coordinates defined on the surface of the plane. In order to avoid any loss in detail, the surface is resampled to match the footprint of the brush. The colour is sampled from a texture and transferred to the points. An overview of the process can be seen in Figure 2. Adams *et al.* [16] extended these techniques to create an application specifically designed for painting on point-based surfaces. Similar to [14], they integrated their application with a haptic feedback device with six degrees of freedom. On this occasion, the brush surface itself is modelled using point sets. This is performed so that there is no loss in detail during painting. Brushes are modelled using a mass spring-based skeletal system and points surround this skeleton to define the surface of the brush. Different types of brushes can be modelled by creating and aligning one or more tips. In addition to this, tips may split when painting on surfaces with high curvature (shown in

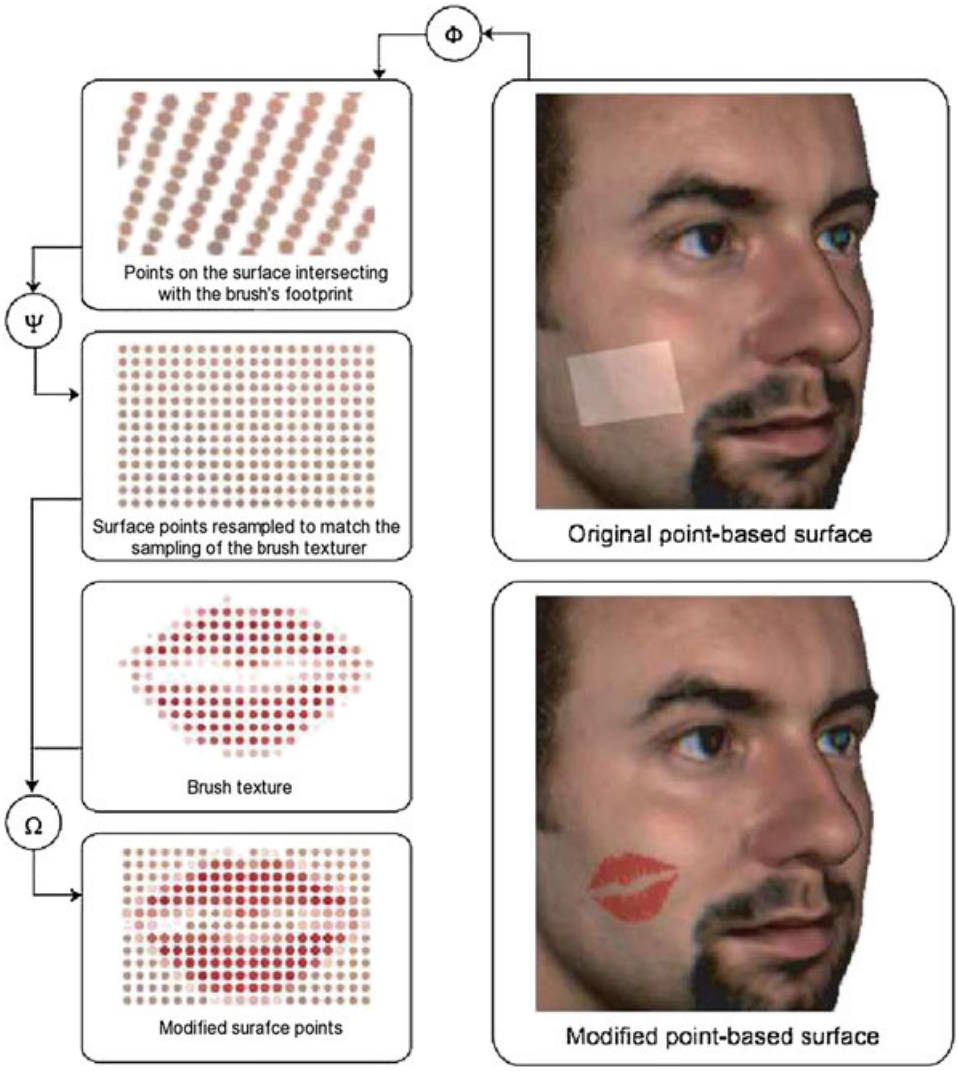

Figure 2. Process flow when painting a point-based surface [15].

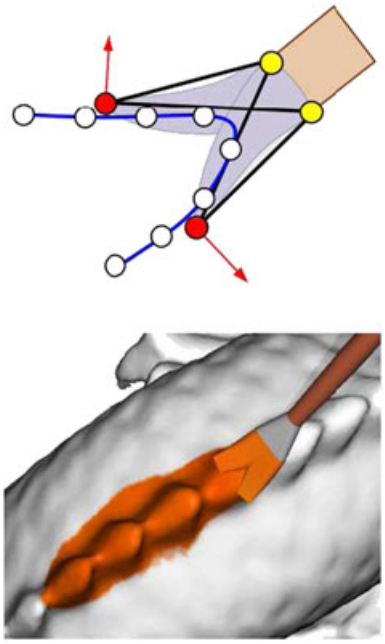

Figure 3. Brush that splits depending on the curvature of the underlying surface [16].

Figure 3). This is accomplished by checking the normals of the points that intersect with the footprints brush tip. If the angle between the normal vectors is greater than a certain threshold, the brush splits into two planes instead of just one. This has the additional advantage of reducing errors in parameterisation when points are projected onto planes in regions of high curvature. Bidirectional paint transfer takes place by calculating the depth of the brush's surface points. Points on the surface may be dynamically upsampled to match the point density of the brush. This makes sure that no details are lost. Finally, after a stroke has been painted, the previously upsampled points will be downsampled depending on the deviation in the stroke colour. This way, an optimal number of points is maintained at all times.

2.2. Medium, Surface Models and Their Interaction

Curtis *et al.* [17] simulated brushstrokes in water colour paintings. Different watercolour effects are simulated on the basis of actual physical observations of real water colour strokes. Some of the simulations made are edge darkening by calculating evaporation of water around the borders of a stroke, granulation using a granulation factor, colour glazing using the Kumbelka-Munk model and backruns by using shallow fluid dynamics. These effects can be seen in Figure 4. The model keeps track of the wetness of the paper, paper height and granularity which affect the absorption and diffusion of pigment at a point.

Takagi *et al.* [18] modelled paper used in pencil drawings as a 3D discretely sampled data set. The authors modelled the fundamental materials of paper such as pulp and loading matter as long deformed cylinders and flat broad cylinders (discs), respectively. The proportion of long cylinders to discs determines the smoothness of paper just as the proportion of pulp to loading matter does in real paper. This volume is discretely sampled at regular intervals to generate the 3D paper volume. When the head of a drawing brush (pencil lead) touches a voxel, a lead voxel is shaved off if the voxel is below other voxels in the direction of the brush movement or the brush is in contact with a loading matter voxel. If the current brush is set to be an eraser or blending tool, the brush head only interacts with lead voxels when they come in contact with each other. In the case of the blending tool, the lead voxels are relocated while erasers remove the lead voxels altogether. This method aims to recreate the paper structure from the most fundamental building blocks. This might be excessive as only the surface voxels are used in all painting operations. Furthermore, maintaining a 3D volume needs considerably more memory and it is computationally inefficient to process and render this volume.

In an attempt to present a general framework in order to define different paper textures, Lee [19] introduced the concept of a paper element (papel). This is analogous to the concept of pixels in a digital image. Each papel contains information on how the different fibres are connected to neighbouring papels. The fibres act as capillary tubes causing water and pigment particles to diffuse. Diffusion is simulated at discreet time steps during which exchanges of ink take place between papels. With the use of the concept of papels, different types of paper textures can be recreated virtually. Although this technique does not directly consider the height of the paper surface, this can be carried out by modifying the properties of individual papels.

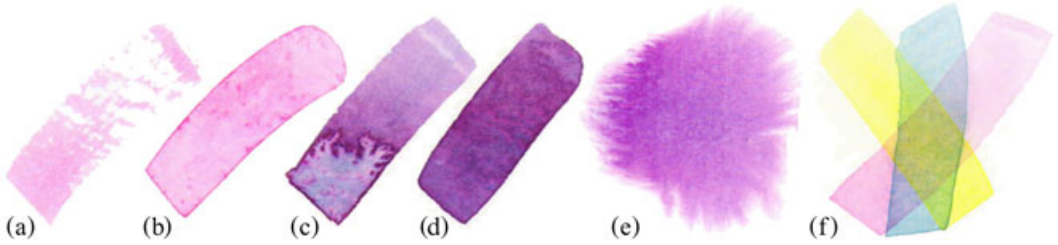

Figure 4. Different watercolour effects: (a) dry brush, (b) edge darkening, (c) backruns, (d) granulation, (e) flow effects and (f) glazing [17].

DOI: 10.1002/cav

Despite the fact that detailed discussions on paper models and diffusion of ink are presented, there are no considerations made for coloured inks and the interaction of ink already deposited on the paper.

In [20], Chu and Tai presented ideas similar to Lee's and extended them to include the absorption and drying of ink. The painting surface is modelled as three layers: (i) the surface layer, where the ink is initially transferred from the brush; (ii) the flow layer, where the physical simulations are made; and (iii) the fixture layer, where the final position of the ink particles is determined. Simulations are made at discreet points on a grid at discreet time steps. Different effects such as dispersion, boundary roughening and edge darkening are approximated by calculating the flow of ink between a point and its immediate and diagonal neighbours by using a modified incompressible Lattice Boltzmann equation. Properties of the ink such as its viscosity and evaporation rate can be defined by the user and can affect the strokes. Finally, the authors also accounted for the density of fibres and substances (alum) applied onto the paper to control the absorption of ink using texture maps. A comparison between real ink flow effects and those recreated using the techniques presented in [20] can be seen in Figure 5.

In addition to providing a brush model, Sousa and Buchanan [11,12] modelled the paper texture as a deformable height field. When a pencil (brush) interacts with the paper, the amount of lead (pigment) deposited is calculated on the basis of the difference in height in the local neighbourhoods, the pencil's grading and pressure at that point. At the same time, the authors proposed deforming the height of the paper, just as a normal pencil would on the basis of the pressure applied and grading. The methods in the publications are based on detailed and careful observations and analysis of the actual physical properties of paper and the different materials that make up the lead in a pencil. Unfortunately, the work presented in these papers does not allow different colours to be used.

Lee *et al.* [21] attempted to create an interactive application to simulate the creation of Jackson Pollock's style of abstract paintings. In order to simulate the flow of paint in 3D and in real time, the authors broke the model in two parts. The first determines the width of the stream by using a one-dimensional Navier-Stokes equation. The second part simulates the 3D flow path of paint using discrete points.

Chu *et al.* [22] presented techniques to accurately simulate the smearing of colour when a stroke is drawn on top of another stroke. The authors argued that previous techniques caused excessive blurring that is not seen in real paintings when the oil or pastel is used as the medium. They identified two reasons for this. (i) Blurring is caused by repeated upsampling and downsampling when the bristles of a brush pick up paint from the canvas and transfer it back. This is solved by creating a pickup texture that is the same resolution as the canvas. This texture is masked by the footprint of the brush. (ii) Blurring is also caused by footprints left that are picked up immediately. This causes a wet look that is not seen when using pastels. This problem is rectified by creating a copy of the canvas that is updated only outside the current pickup texture's boundary. The user is given an option to disable this, so that a wet look can still be generated. Using this method prevents displacement of the ink caused when the bristles are tuned. However, the authors did not account for the amount of ink already deposited on the canvas or picked up by the brush. Additionally, thin translucent layers of paint are not accounted for. An example of several brushstrokes smeared using this method can be seen in Figure 6.

The aforementioned methods aim to provide the user with realistic and fine levels of control over each stroke. These techniques require a very high degree of user interaction and do not use any kind of reference images or other forms of data to assist the user. The user must be a skilled artist and the techniques are designed to retain the artist's skills, which were acquired over a long time. The

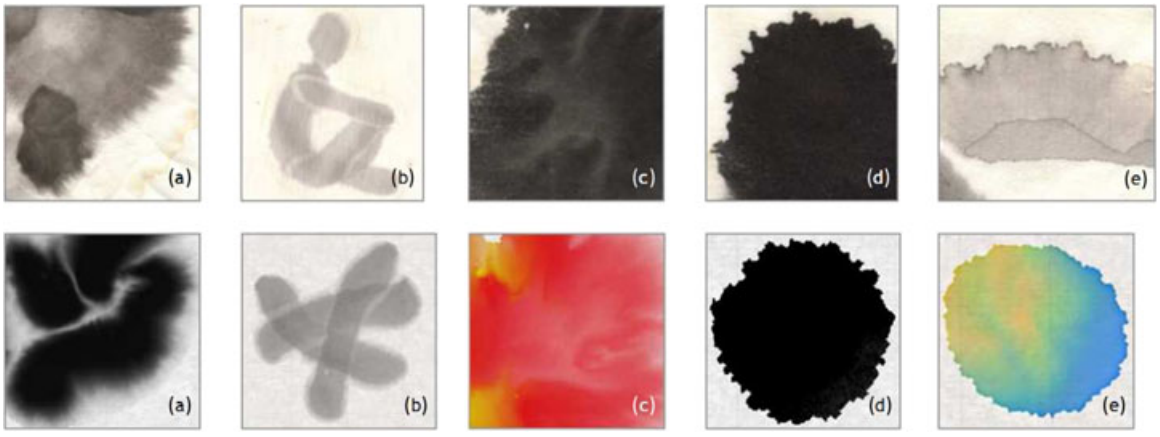

Figure 5. Comparison of different ink flow effects recreated using the technique presented in [20]. Top row shows images of real ink flow effects, whereas bottom row shows images of ink flow effects simulated using the technique presented in [20]: (a) feathery pattern, (b) light fringes, (c) branching pattern, (d) boundary roughening and (e) boundary darkening.

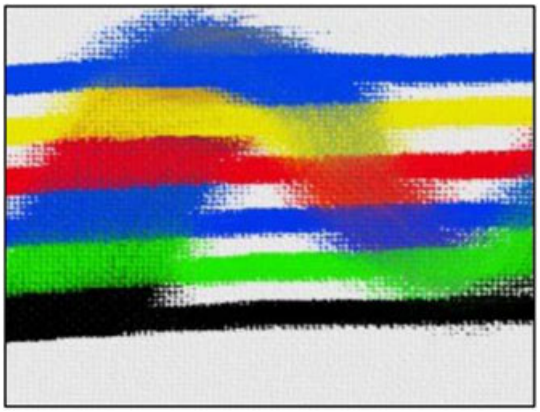

Figure 6. Smearing of overlapping brushstrokes using techniques presented in [22].

direct integration of these methods into a more automated system will present us with a very challenging optimisation problem as there are several dimensions, so specifying the desired result would be a complex task to undertake.

3. STROKE GENERATION

In a painting, a stroke is the fundamental unit just as a pixel is in a digital image. Artists use different mediums, surfaces and brushes to create their own unique styles. In a similar way, authors present techniques that work best for creating output in one or more given styles. Different algorithms require varying degrees of expertise and user intervention over the process of generating the final output. The final output is generally in the form of a static image or animated video sequence that aims to recreate the same properties found in a human-created art piece. Different methods try to achieve this common goal by using different algorithms to determine the properties of a stroke. These properties are generally its (i) position and density, (ii) path and orientation, (iii) length and width, (iv) ordering, (v) colour and (vi) temporal coherence. In the following sections, we will review the different methods employed to achieve this.

3.1. Abstraction

The process of rendering using strokes abstracts the input data to a large extent. However, this method provides a uniform level of abstraction across the output. Artists, on the other hand, vary properties of strokes in order to abstract regions of less interest and focus our attention to certain regions of the painting. We will first review the different techniques used to estimate importance as they will be referred to when we describe different methods used to determine properties of strokes.

Mao *et al.* [23] abstracted an image by segmenting the input image into regions. This is carried out by merging regions when the difference between their histograms is less than a certain threshold. Small regions are merged with larger neighbours to prevent small specks from being generated. Yamamoto *et al.* [24] extended this technique to colour images by checking the L_2 norm between colours in the CIE-LAB colour space. Papari *et al.* [25] proposed an abstraction filter on the basis of the fact that an artist generally discards texture information but preserves the edges present in an image. The authors extended the Kuwahara filter [26] by making it more robust to noise and improving its output in highly textured regions of an image. This is carried out by calculating the Gaussian weighted mean and standard deviations in N sectors of a disc centred at a pixel. An example of the output produced using this technique is shown in Figure 7. Gooch *et al.* [27] made use of depth information to modify properties of the strokes. However, depth information may not always be available. In this case, importance of a region must be estimated in other ways. Santella and DeCarlo [28] determined importance at a point by tracking which parts of an image a user looks at. The authors argued that the more time a user spends looking at a certain part of an image, the more important it must be. A set of locations where the user's eyes fixated

(a) (b)

Figure 7. (a) A reference image and (b) the reference image abstracted using techniques presented in [25].

DOI: 10.1002/cav

on are collected using an eye tracker and an importance map is generated from this. Although this is a very good way of calculating importance, it is subjective and requires human subjects, which can be a severely time-consuming and expensive process. Collomosse and Hall [29] also generated an importance map referred to as a *salience image*. An example of this can be seen in Figure 8(c). The significance of a pixel at a certain location is based on the first and second partial derivatives of the image convolved using a Gaussian kernel. The significance of a pixel's value is computed with respect to its neighbours. This method essentially computes the importance of a pixel based on its variance, suggesting that more important points are located in regions in which there is much change. Xu *et al.* [30] used a similar theory to determine important points in a 3D point cloud. They combined the variation of colour and normals to estimate the *feature degree* (importance) of a point. Zhao and Zhu [31] determined the abstraction level of an image after the user has segmented the image into a tree structure. The scene in the image and the different nodes of the tree are classified manually depending on the materials of the objects within them. An example of this hierarchical classification can be seen in Figure 9. Optionally, this classification may be done automatically using the *texton boost* algorithm as used by Lin *et al.* [32]. Once the image is segmented and classified, the importance of each segment is based on the probability that a certain type of material appears in a particular type of a scene. The probability is based on a database developed by Yao *et al.* [33].

3.2. Position and Density

One of the first challenges with generating strokes is determining their location. One of the key challenges with this method is ensuring that there are no unwanted holes in the output painting. This has to factor in the stroke's coverage area. Because, for many methods, calculating the exact boundaries of a stroke may be computationally prohibitive, methods generally resort to overlapping multiple strokes to ensure there are no holes.

Haeberli [34] avoided this problem by requiring the user to manually position each stroke. Kalnins [35] required the user to place a few strokes. These strokes are used as templates and are then replicated at other locations by calculating the distance of the template strokes from feature lines. Others took a more automated approach. Litwinowicz

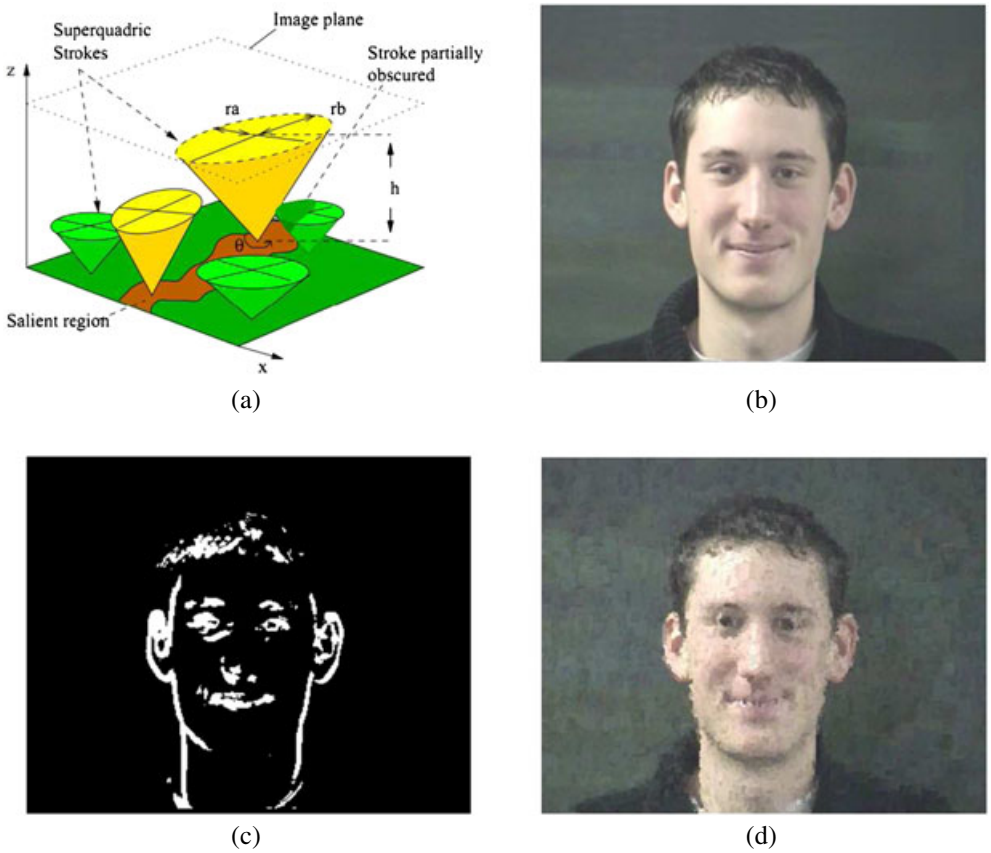

Figure 8. Painterly rendering achieved using salience maps as presented by Collomosse and Hall [29]: (a) superquadric strokes used in this method, (b) a reference image, (c) salience map of the reference image and (d) output produced using this method.

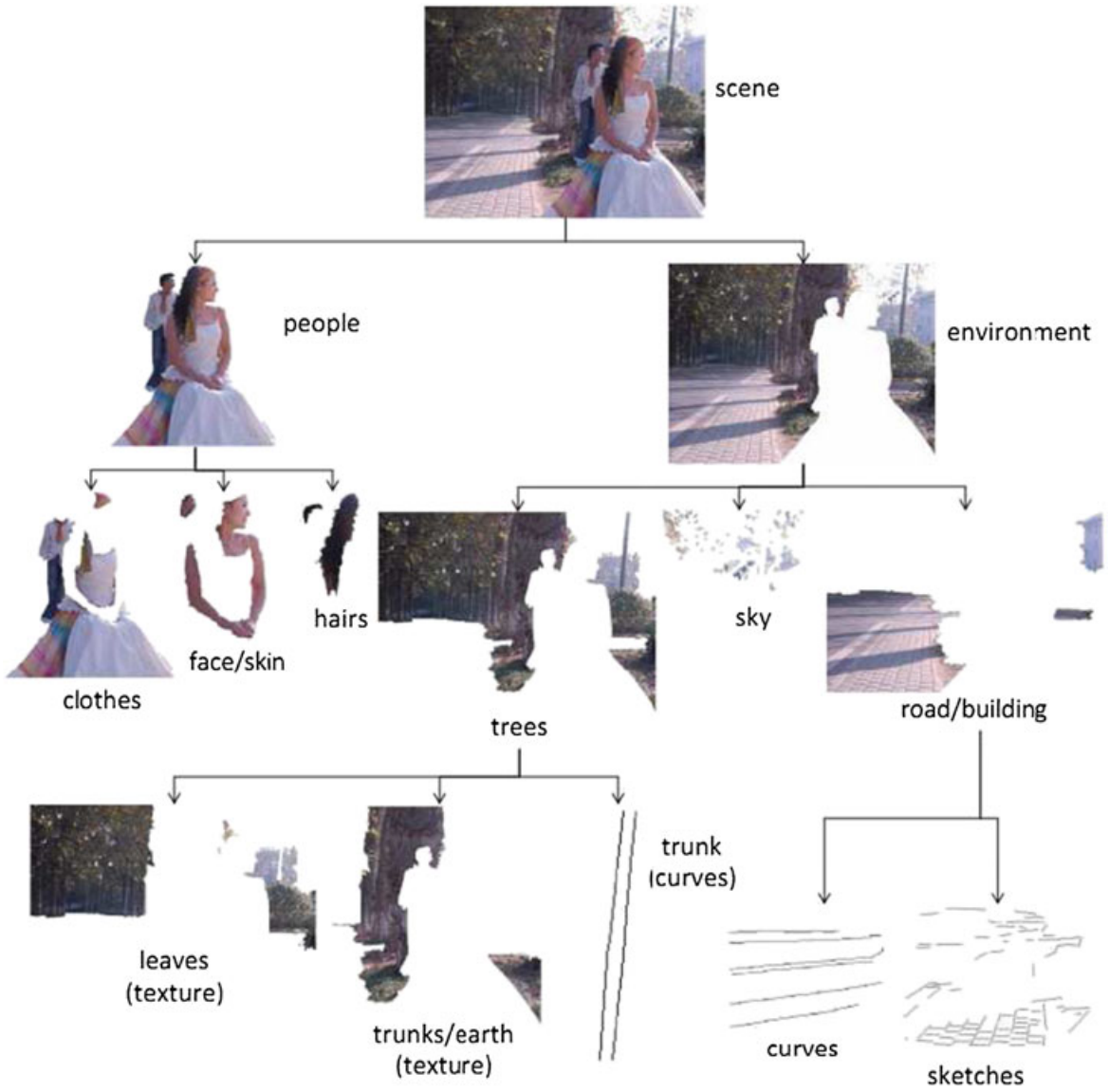

Figure 9. A reference image hierarchically segmented and classified by materials in each region as implemented in [31,50].

[36] placed strokes at alternate pixel locations to ensure there are no holes in the output. Park and Yoon [37] placed strokes at locations where a randomly generated value exceeds the probability threshold. Hertzmann [38] and Schlechtweg *et al.* [39] placed the starting points of a stroke at locations where the intermediate output differs in a greater amount than a certain threshold from a reference image. In a later work, Hertzmann [40] ensured sufficient coverage and avoided adding new strokes by repositioning existing strokes. This is carried out by including an energy term E_{cov} for the coverage and another energy term E_{nstr} for the number of strokes. Shiraishi and Yamaguchi [41] attempted to estimate the area of the strokes by generating an intensity image using the 0^{th} image moments. Initial locations for the strokes are determined from this intensity image by using a halftoning algorithm by Velho and Gomes [42]. Strokes are later iteratively repositioned by first calculating a difference sub-image centred at each stroke's location. The strokes are then repositioned to the centroid of this difference image. Mao *et al.* [23] determined the density of strokes by generating noise at a density based on the tone of the underlying image. The strokes will be implicitly generated by the use of a low-pass filter. Yamamoto *et al.* [24] extended this technique to generate coloured output by determining the two most prominent colours for each region. Implicit strokes for each colour are generated in two layers. The density of strokes in each layer is based on a duotone algorithm that determines the proportion each colour must be used to achieve the target colour. Meier [43] and Kowalski

DOI: 10.1002/cav

et al. [44] attached strokes to the surface of a 3D object randomly on the basis of a user-defined density parameter. Kowalski *et al.* [44] allowed the user to later move individual strokes by parameterizing the 3D surface. Schmid *et al.* [45] allowed the user to place strokes in 3D space on the basis of level sets defined by proxy geometry. With the parameters adjusted in an *energy minimisation* function, a stroke can be optimised to stick to a given level set or span a range. An example of the different tools made possible using this technique can be seen in Figure 10. This gives an artist the freedom to easily paint volumetric effects such as clouds, smoke and fur that would have been difficult to do so previously. Way *et al.* [46] used the dot product of the normalised view vector and surface normal to determine the density of strokes in a certain location. This way, there will be a higher density of strokes along the silhouette of an object, thus highlighting object boundaries.

3.3. Orientation and Path

The next task is to align the strokes to a particular direction. Haeberli [34] allowed the user to decide the orientation of each stroke by calculating the direction of movement. Haeberli also suggested automatically aligning strokes perpendicular to the gradient vector at a point in the image. This is based on observing artists who orient their strokes in the direction where there is minimal change in colour. Gradient vectors can be calculated quickly using a Sobel filter. Litwinowicz [36], Hertzmann [38], Collomosse and Hall [29] and Santella and DeCarlo [28] used this strategy to align their strokes. This method works well in high-frequency regions where there is much variation in colour. In regions where the colour is similar to neighbouring pixels, the gradient vector tends to point in arbitrary directions. Litwinowicz [36] addressed this problem by checking the gradient magnitude. A small gradient magnitude would mean an unpredictable gradient vector direction. In this case, the direction is interpolated using radial basis functions from neighbouring regions with a large gradient magnitude. Lee *et al.* [47] used optical flow to calculate the direction of motion from the reference video to determine the orientation of strokes representing flowing elements such as clouds and water. Shiraishi and Yamaguchi [41] used image moments to determine an angle θ to which their strokes will be aligned to. When a 3D surface is being processed, the orientation can be based on the 3D surface normals as shown by Meier [43]. In [48], Xu and Chen assigned an importance factor referred to as a *feature degree* to 3D points in a point cloud. The method of least squares is then used to fit a line to points with a high *feature degree*. Later in [30], Xu *et al.* used this line to determine the direction of a stroke. Mao *et al.* [23] and Yamamoto *et al.* [24] generated a vector field to be used with the line integral convolution (LIC) algorithm. This is carried out by transforming the pixels in the local neighbourhood of the point into the frequency domain using the Fourier transform. The most prominent direction is determined by comparing the sum of powers at small angles.

Once the initial direction of a stroke is determined, its path must be set. Haeberli [34], Litwinowicz [36], Meier [43] and Shiraishi and Yamaguchi [41] used short straight strokes. Because the strokes are short, there is no need to generate a curved path. Hertzmann [38] generated long curved strokes by using a series of connected short straight lines. Each line segment is oriented perpendicular to the gradient direction at the point where the line starts. This method suffers from the same problem seen when using gradient directions to determine initial orientation of a stroke. Hertzmann [40] later improved upon this technique by iteratively refining the path using energy minimisation. Another option, implemented by Santella and DeCarlo [28], is to continue moving in the same direction when the gradient magnitude at a point is low. Others such as Olsen *et al.* [49] used a higher threshold on the gradient magnitude and used a radial basis function to define a smoother and less varying vector field within regions of a segmented image. The user is given the option of interactively modifying the vector field using fluid dynamics. Gooch *et al.* [27] generated stroke paths by calculating B-splines from the medial axis of a segmented image. Holes within the regions are removed and borders of the segmented regions are smoothed using morphological operations. This method used without any alteration would generate a large number of very short strokes. To avoid this, the authors merged the medial lines of two segments if the difference in colour, direction and distance is within a certain threshold. Zeng *et al.* [50] used the primal sketch algorithm of Guo *et al.* [51] to define a vector field to be used by the strokes. In the case of 3D objects, we have additional information on the objects and strokes can follow surface curvature and feature lines, as demonstrated by Chi [52].

3.4. Width and Length

The width and length of a stroke can determine the style of the output painting and play a particularly important role.

Haeberli [34], Litwinowicz [36] and Meier [43] generated short strokes to give the output an impressionistic style. Litwinowicz [36] suggested a method where strokes are clipped when they encounter an edge so that strokes do not cross over object borders. Hertzmann's methods [38,40] have the capability of generating longer strokes abstracting the input reference image. Collomosse and Hall [29] determined the shape of a stroke by using superquadric equation $(x/a)^{2/\alpha} + (y/b)^{2/\alpha}$. α controls the shape of the stroke, making the stroke rectangular as it approaches 0 and star shaped as it approaches ∞. a and b are normalised and control the eccentricity of the superquadric. The difference between a and b is proportional to the gradient magnitude, resulting in elongated strokes in regions with well-defined edges and rounder strokes in regions with less variation in colour. The superquadric strokes used in the method can be seen

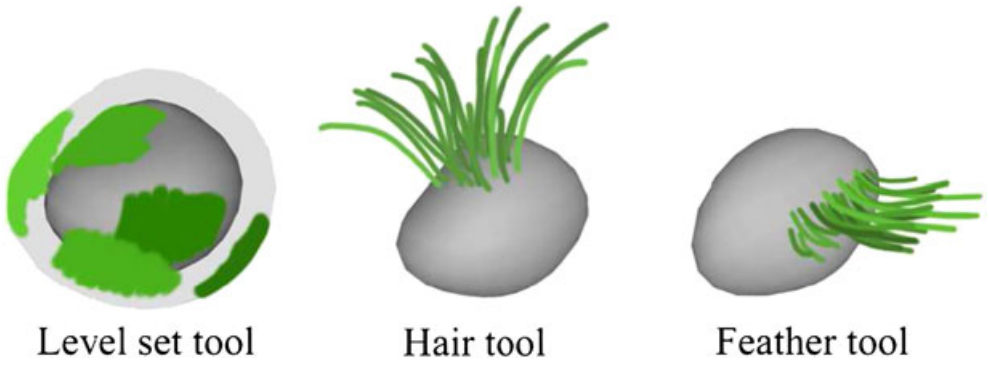

Figure 10. The stroke positioning tools made possible by adjusting the weights of an energy minimisation function in [45].

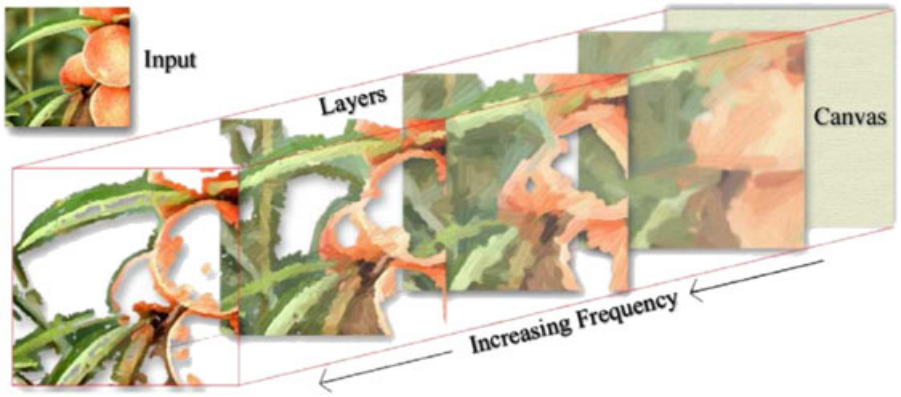

Figure 11. Layering used by composite strokes of different widths into the final output as implemented by Hays and Essa [54].

in Figure 8(a). Mao *et al.* [23] and Yamamoto *et al.* [24] controlled the width of their implicit strokes by varying the granularity of the noise. Length of the stroke is controlled by varying the length of the convolution kernel used in LIC [53].

3.5. Ordering

Hertzmann [38] was one of the first researchers in the area who realised that ordering strokes according to their thickness is a common practice used by artists. The work by him and others such as Park and Yoon [37] and Hays and Essa [54] generates output by using multiple layers. Each layer uses a low-pass filtered version of the reference image used by the upper layer. This way, the amount of detail available to a layer reduces with its depth and the width of the strokes used on each layer is increased. The layers are composited to generate the final output. See Figure 11 for an example. In [27], Gooch *et al.* provided the user an option of compositing different layers generated using completely different reference image source data. This allows us to easily place an object in front of different backgrounds. Ordering of strokes will be randomised within each layer with the use of this method. Shiraishi and Yamaguchi [41] suggested ordering strokes according to their area instead. In this case, calculating the stroke area is computationally efficient as the authors assumed all strokes to be rectangles. A shortcoming of this technique is that the shape of all strokes in a painting is restricted to being short and straight. Collomosse and Hall [29] ordered strokes on the basis of a *salience map* (see Section 3.1).

3.6. Colour and Texture

There are several methods in which the texture and colour of a stroke can be determined. Methods presented by Haeberli [34], Litwinowicz [36], Hertzmann [38] and several other authors sample the reference image's colour at the position where the stroke is placed. In the case of Haeberli [34] and Litwinowicz [36], the stroke colour is blended with a stroke texture to add detail to individual strokes. This is a simple and fast method that works well if there is little variance in the colour under the footprint of the stroke. In an extension of his work, Hertzmann [40] and others such as Schlechtweg *et al.* [39] and Hays and Essa [54] used the average colour under the stroke. Chen *et al.* [55] also used the average colour, but they sampled the average colour from Voronoi cells. The cells acquire

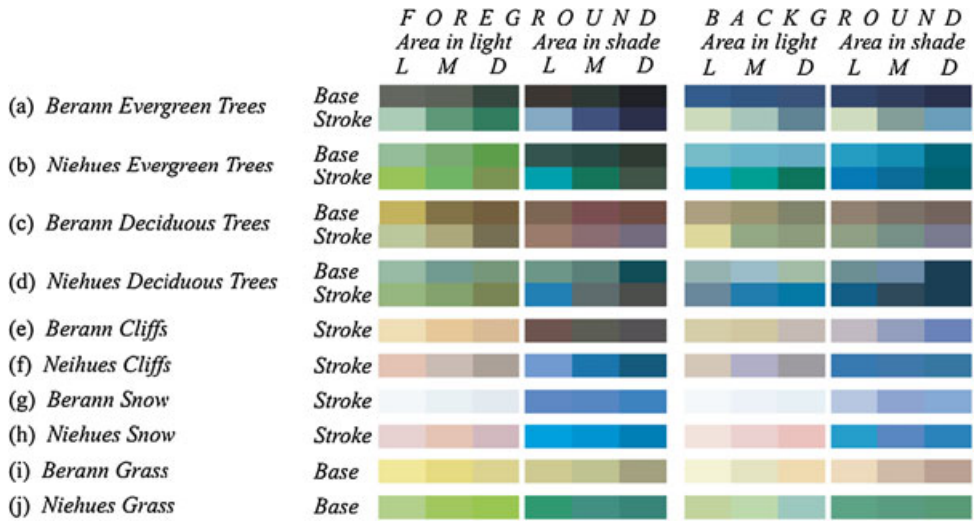

Figure 12. Example of two colour palettes selected in [57] by examining the work of two artists (Berann and Niehues).

the average colour of the underlying pixels of the reference frame. Yamamoto *et al.* [24] decided the two most prominent colours within each region by checking the difference between the colours in the region with a predefined palette. Once the two colours have been determined, the Neugebauer duotone printing algorithm is used to determine proportion of each colour so that strokes may be generated in the same proportion. Luft and Deussen [56] and Braktova *et al.* [57] decided the colours from a carefully selected palette of colours, determined by studying similar types of paintings created by artists. An example of such a palette can be seen in Figure 12. Zeng *et al.* [50], Zhao and Zhu [31] and Lin *et al.* [32] used a brushstroke texture on the basis of the material classification of a region of the painting. The texture is sampled from a database of stroke textures created manually by artists for each type of material. Xie *et al.* [58] generated the stroke texture procedurally. The footprint of the brushstroke is divided into six segments. Each segment is assigned a texture from a sample of six styles (see Figure 13). The textures in the different segments are blended to generate a seamless stroke. Hsu *et al.* [59] provided a detailed description of texturing individual strokes. They included techniques to handle self-overlapping strokes by removing overlapping points, texturing strokes in regions of high curvature by adding additional points and the parameterisation of the stroke footprint so that a texture can be mapped onto it.

Colours of strokes may be modified locally on the basis of different factors. Gooch *et al.* [27] modified the colour of a stroke on the basis of the depth at the point. Santella and DeCarlo [28] increased the contrast along edges and increased saturation on the basis of the importance of a region (see Section 3.1). Zeng *et al.* [50] adjusted the colour temperature of the strokes within a painting depending on the scene. This is carried out by observing the colour temperature shift in paintings created by expert artists.

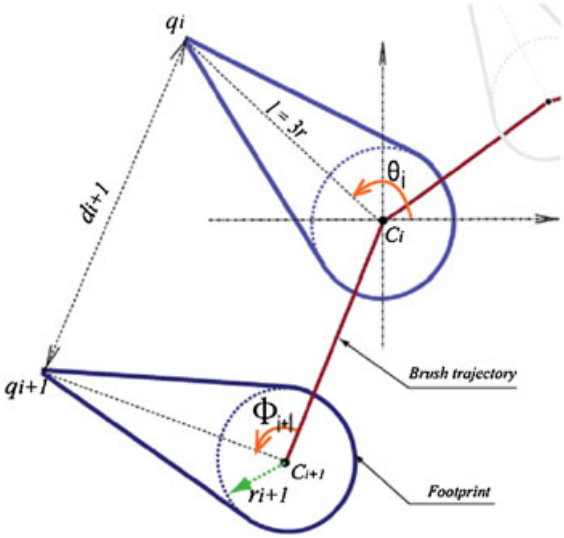

Figure 13. Six different brushstroke styles used by Xie *et al.* [58]. Each stroke is divided into six segments. The texture used in each segment is presented as smaller strokes above each stroke style.

3.7. Temporal Coherence

A considerable amount of recent research has focused on generating temporally coherent strokes that are capable of creating animated paintings. This area of research has the potential of making a new visual style available to animators and directors that previously required hundreds or even thousands of man hours in order to achieve manually. Key challenges within this area are (i) detecting motion of objects, (ii) transforming strokes along with the objects

and (iii) modifying the properties of strokes in a temporally coherent way to avoid popping or scintillation.

3.7.1. Detecting Object Transformation.

When video is used, optical flow is a popular method to track movement between frames. Litwinowicz [36], Hertzmann and Perlin [60], Olsen *et al.* [49] and Lu *et al.* [61] used optical flow to move strokes. Unfortunately, optical flow algorithms fail to accurately track motion between frames. In an attempt to overcome this, Park and Yoon [37] combined optical flow with edge detection. The movement of pixels between consecutive frames is stored in 2D vector fields called *motion maps*. The authors created two *motion maps*. One is a strong motion map, which stores the vectors for pixels on edges. All pixels that lie between the old position and the new position are also stored in this motion map. The other is a weak motion map, where the vectors for regions around the edges are stored. The remaining pixels acquire motion vectors from these pixels. An overview of this process can be seen in Figure 14. Lin *et al.* [32] did not use optical flow. Feature points at locations of high frequency are detected using scale-invariant feature transform. A gradient histogram is generated at these points. In low-frequency regions, the authors used *maximally stable extremal regions* [62] to generate a colour histogram of the pixels within the region. Feature points assigned to two different segments in two frames are mapped to each other by minimizing a difference metric. A snapshot of the different feature tracking techniques used can be seen in Figure 15. Because the techniques employed here are tuned to smooth movement of the underlying objects, they are unable to accurately recreate rapid movement and detect occlusions and sudden appearances of objects.

A side effect from using methods such as these is a build up of stroke density in certain locations, while holes tend to appear at other locations. Hertzmann and Perlin [60] added new strokes at locations where the colour difference between the output and the reference image is greater than a certain threshold. Litwinowicz [36] generated a Delaunay triangulated mesh. New strokes are added at locations where the area of a triangle is greater than a certain threshold. In a similar way, strokes are removed if they come too close to each other.

Hays and Essa [54] added new strokes at locations where gaps form in the output. Collomosse *et al.* [63] suggested an offline approach where the video sequence is treated as 3D volume data in space and time. Individual frames are segmented using the EDISON algorithm [64] and regions are merged between frames. A Catmull-Rom patch is fitted to the boundary of these regions to create a smooth surface in 3D, as can be seen in Figure 16. A 2D tensor field is calculated at each slice of each object in the time domain. The 2D tensor field holds the transformation of 2D points within the object between frames. Zhang *et al.* [7] detected and used flow lines from a video sequence containing water to direct motion of their strokes. The problem becomes simpler when using 3D objects. Lu *et al.* [61] generated motion vectors by using the projection matrix of the previous frame to reproject a vertex's position and compare it with its current position. Others such as Meier [43] and Kaplan *et al.* [65] attached strokes to particles on the surface of a 3D object and transformed it along with the object. In Figure 17, we look at an overview of how the different stroke colouring, orientation and sizing techniques come together to produce the final output as presented in Meier's work.

3.7.2. Modifying Properties of Strokes in a Temporally Coherent Way.

Hays and Essa [54] prevented rapid changes by limiting the rate of change of a stroke's property such as its location and colour. New strokes are gradually faded in, whereas

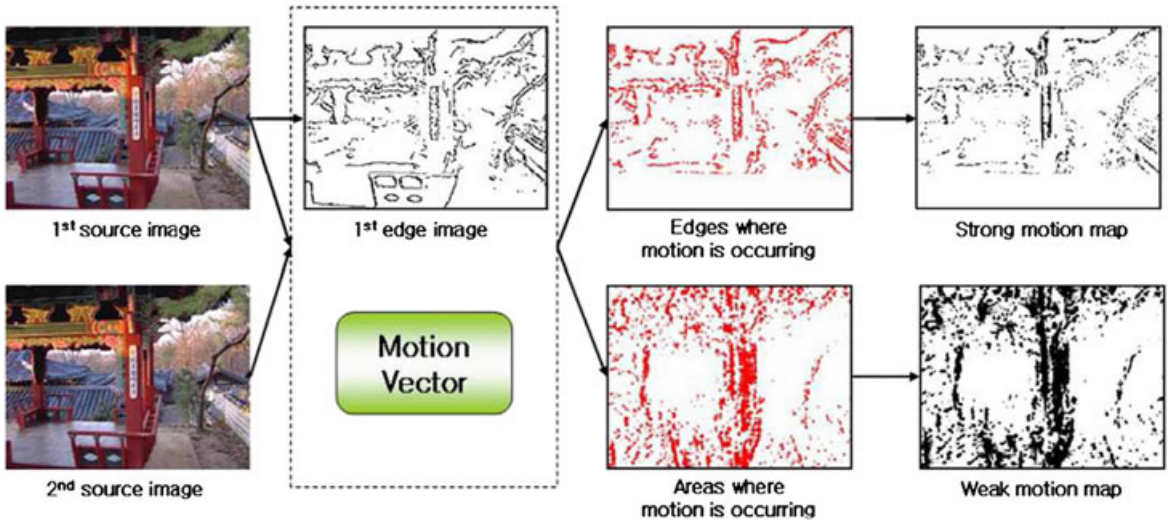

Figure 14. Combining strong and weak motion maps to calculate motion between frames [37].

Comp. Anim. Virtual Worlds 2013; **24**:43–64 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/cav

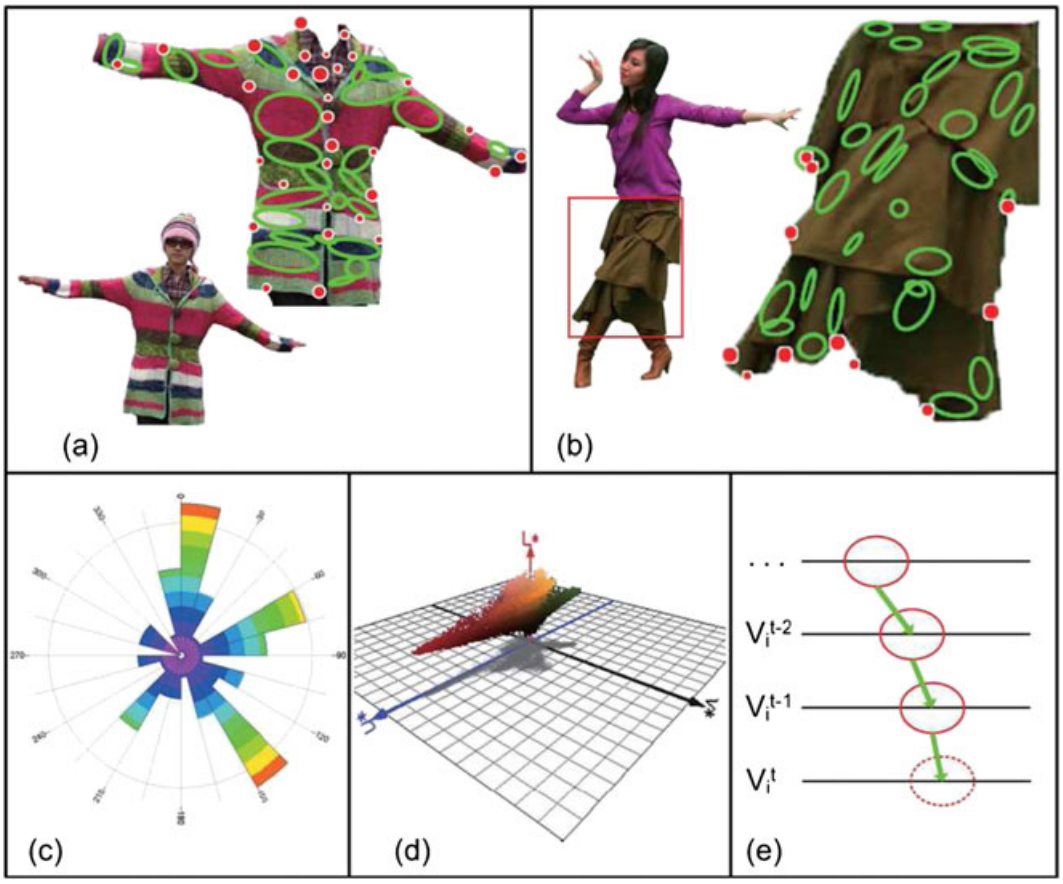

Figure 15. Different methods used to track points between two frames in [32]: (a) features in a texture region, (b) features in a textureless region, (c) gradient histogram, (d) color histogram and (e) motion consistency.

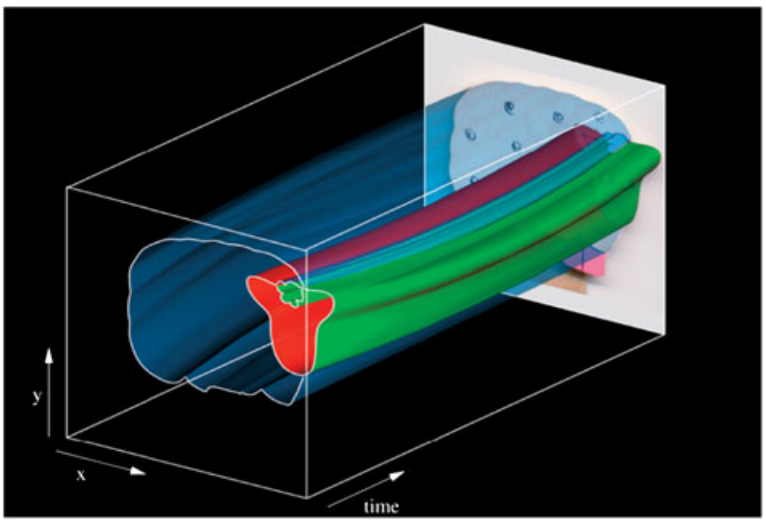

Figure 16. Stroke surfaces generated from a video volume [63].

old ones are faded out to prevent popping. Collomosse *et al.* [63] prevented scintillation by removing regions that do not exist for more than 0.25 seconds. Surrounding objects expand to occupy the void space. Lin *et al.* [32] tried to minimise movement by deforming a stroke with the use of the feature point locations. In addition to this, they

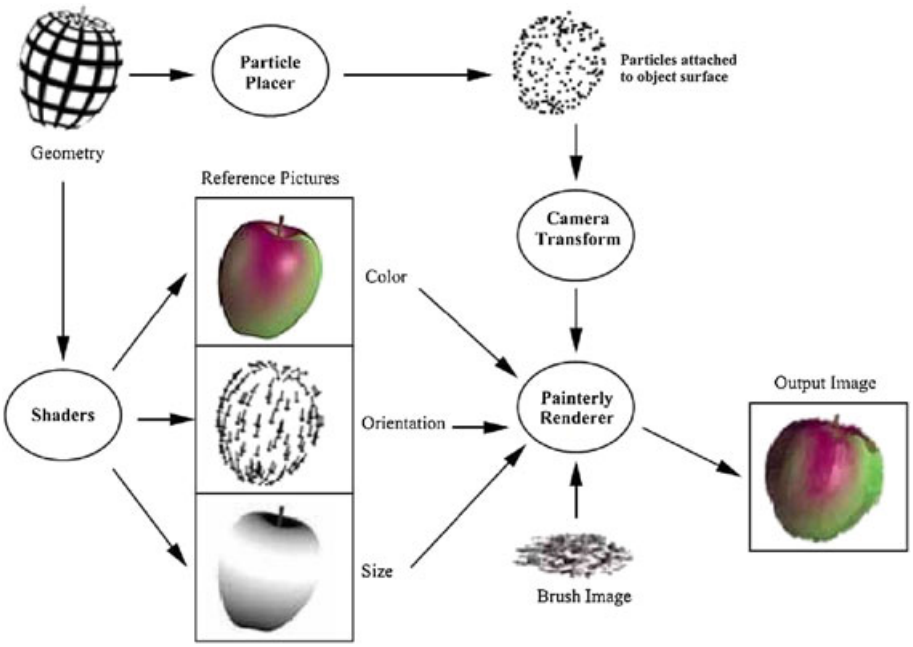

Figure 17. An overview of the 3D painterly system presented by Meier [43].

dampened stroke movement in the space and time to prevent rapid movement of strokes. Unfortunately, this does not work very well when objects move rapidly in a scene.

When rendering strokes on 3D objects, Kaplan *et al.* [65] suggested scaling up the size of strokes when objects move closer to the viewpoint. If stroke density is to be maintained, Xu and Chen's method [48] can be used. Here, points at which strokes are rendered are randomly and continually added or removed depending on the distance of the viewpoint from the object.

4. IMPLICIT STROKES

Instead of rendering individual strokes, some methods manage to achieve painterly style output by using image processing techniques.

Bousseau *et al.* [66] attempted to create watercolour style paintings without explicitly simulating brushstrokes. Instead, base colours are generated by segmenting an image by using a mean shift algorithm. Various phenomena seen in a watercolour painting are simulated through the modification of the base colours. Flow effect is simulated using a Perlin noise layer, granulation using sum of Gaussians and the paper texture is determined by scanning different papers. Edge darkening and rough boundary edges are simulated by using the gradient of the segmented image and the dry brush effect is produced by thresholding the height of the paper texture. The degree to which each of these effects contributes to the final output can be controlled by the user.

In an attempt to generate better results by using optical flow, Bousseau *et al.* [67] extended their previous work [66] to include video. Because this method does not explicitly create strokes, only the underlying textures that simulate the flow, dry brush and granulation effects in a watercolour painting need to be distorted according to the underlying motion, in order to prevent a shower door effect. Segmentation of individual frames are made temporally coherent by using a 3D structuring element for the morphological region boundary smoothing operations. The third dimension is in the time domain and the structuring element tapers as the frames progress away from the current frame. The other watercolour simulations are just image filters that work on a per-pixel basis. The underlying textures are distorted and blended using bidirectional optical flow. One optical flow is calculated as normal from the beginning of the video sequence going forward and the other in reverse from the end of the video. With the use of two motion flow calculations, more robust distortions can be achieved. An example of the distortions produced using this technique can be seen in Figure 18. Because optical flow is not accurate, errors can build up over time. This is handled by resetting back to the original texture after a certain period.

Normally, texture transfer algorithms generate a result image (R) by using colours from a source image (S) while trying to maintain the texture information from a target image (T). These algorithms are based on colour information. Lee *et al.* [68] extended the standard texture transfer algorithms to replicate brushstroke styles from a source painting by using directional information in the target image. An overview of the algorithm can be seen in

DOI: 10.1002/cav

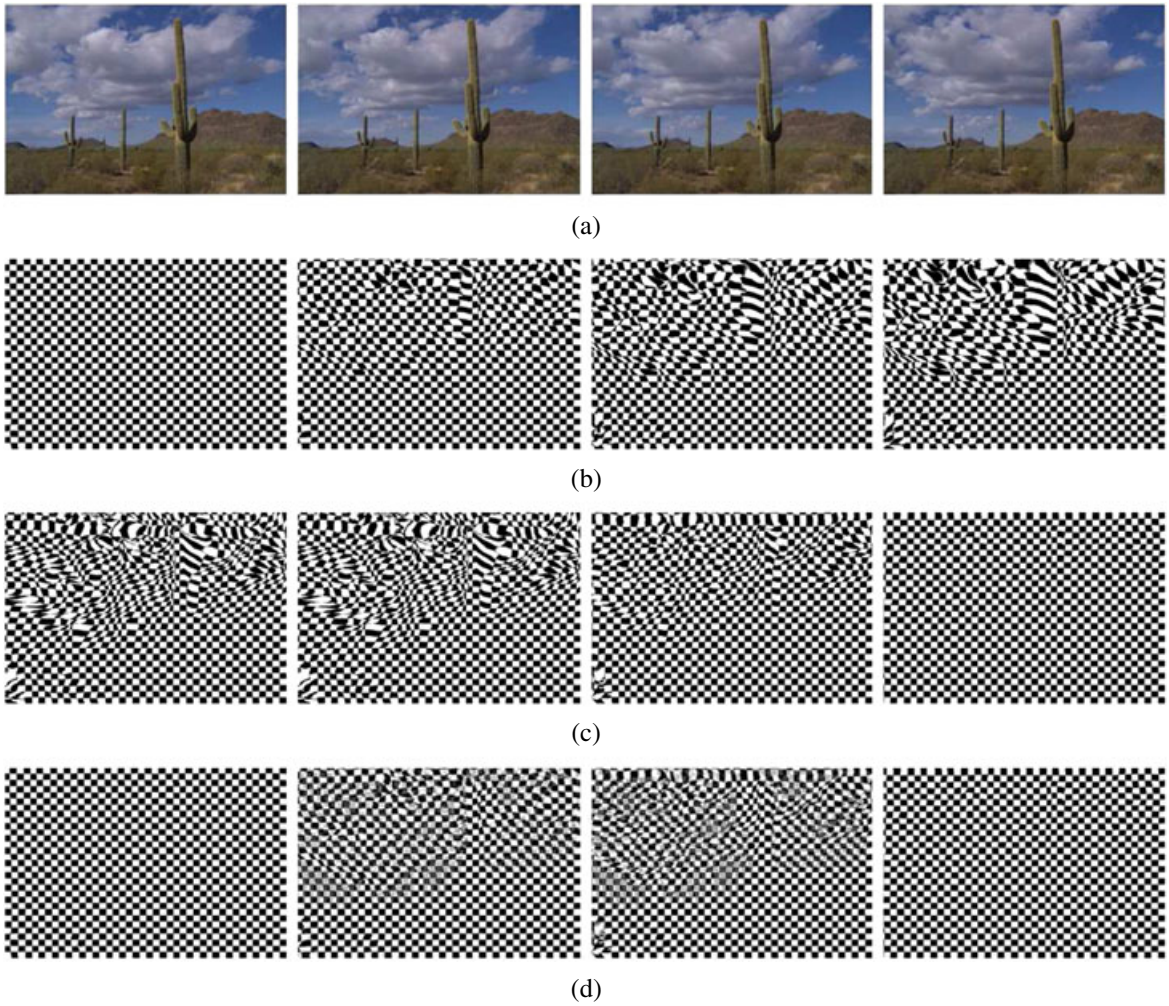

Figure 18. Bidirectional texture distortion as implemented in [67] is demonstrated on a checkerboard texture: (a) the original reference frames, (b) the checkerboard texture distorted in the forward direction, (c) the checkerboard texture distorted in the reverse direction and (d) the checkerboard texture distorted by combining the forward and reverse directions.

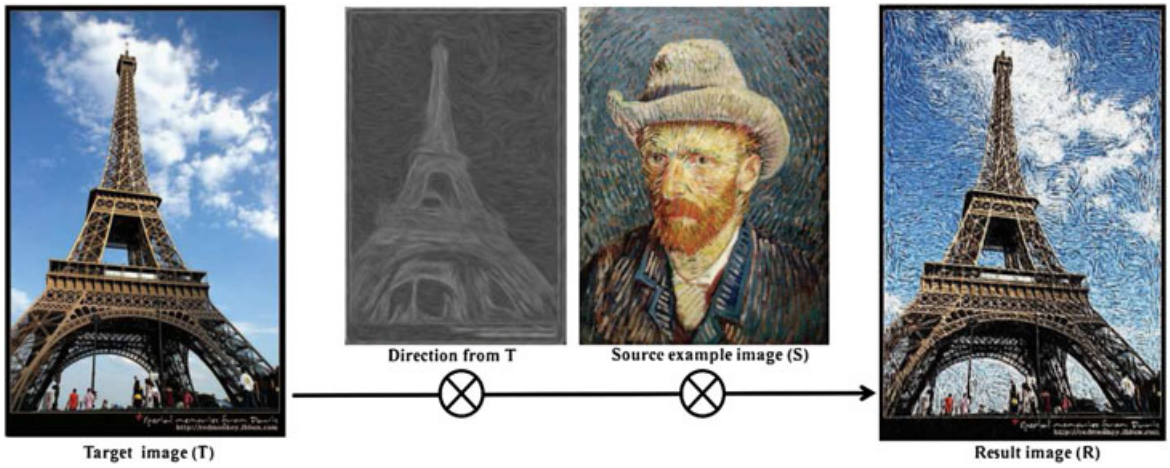

Figure 19. The directional texture transfer algorithm [68].

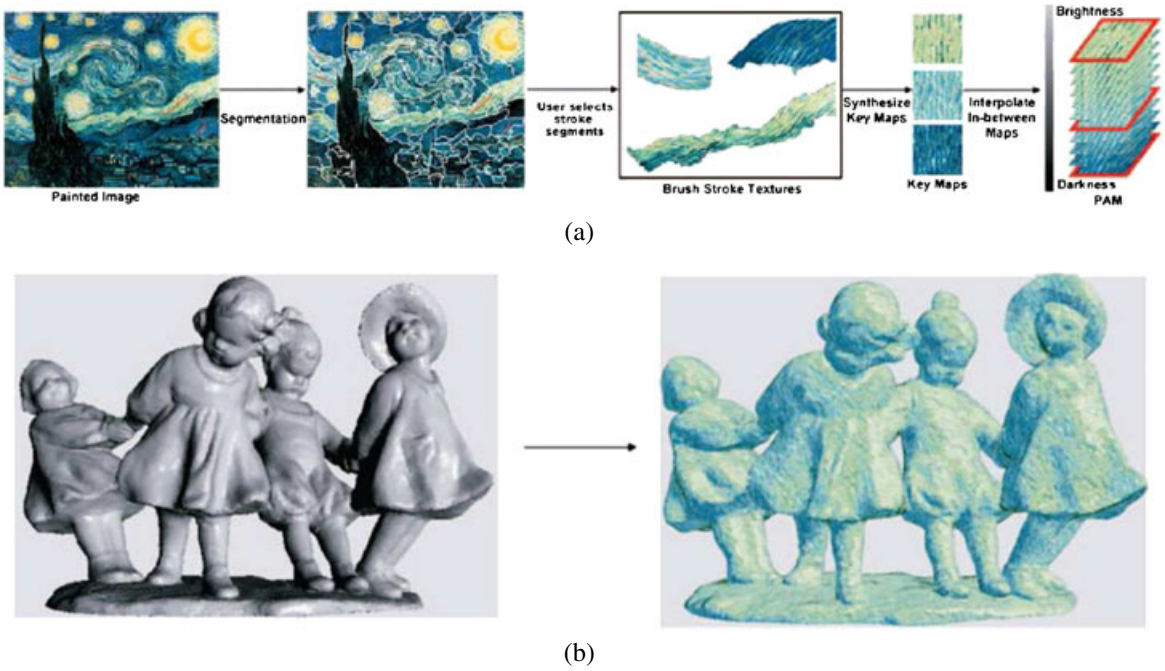

Figure 20. (a) The different stages to generate the tone maps and (b) the output [69].

Figure 19. The algorithm initialises R with random pixels from S. The result R is then iteratively updated by checking for the best candidate in the local neighbourhood of each pixel. In this case, the best candidate is determined by including directional information.

Yen *et al.* [69] also utilised a texture transfer algorithm to paint 3D geometry by using brushstroke styles from a sample painting. The advantage of using this method is that the user can select regions of the painting whose style is to be replicated. This is achieved by segmenting the source painting and selecting particular regions to be used. An overview of their algorithm can be seen in Figure 20(a). The strokes are aligned using vector fields. The authors presented techniques to normalise the width of the strokes that are referenced from the painting. The user draws two parallel lines in each selected region to indicate the number of strokes between them. It is assumed that the number of strokes between these two lines remains the same in all segments. Stroke alignment is carried out automatically using texture synthesis. Key maps are produced for each region selected by the user. The synthesised key maps have uniform stroke width and alignment. A range of intermediate tones are generated automatically using blending because the regions selected by the user would generally only contain a few tones. Finally, the strokes are rendered onto the 3D model with the use of texture splatting. Discontinuities between different tone maps are avoided using hardware interpolation. The strokes can be animated coherently over the surface of the mesh by moving the textures. The final output can be seen in Figure 20(b).

The methods of Yen *et al.* [69] assume that the reference painting contains only straight strokes. Also, there is no distinction made between stroke texture and colour. Kulla *et al.* [70] overcame these shortcomings by using a slightly different approach to painting 3D meshes. They required the user to create an image strip that contains varying texture and colour. The texture and colour is separated during preprocessing and blended back during rendering. This way, a user is offered the flexibility to change only the colours and keep the stroke textures the same or vice versa. A 3D mesh is then rendered in the painterly style using different techniques. One such technique is based on texture quilting. The quilting method is modified to reduce discontinuities between blocks. Another method converts the image strip into a 3D texture. In order to reduce seams within the texture, a texture blending method is used to blend each individual texture block's edges with its centres. Some modifications are also made to make the output temporally coherent. Unlike the methods presented by Yen *et al.* [69], the authors do not make any effort to align the strokes.

Hertzmann *et al.* [71] provided a more specialised framework that can perform texture transfer in addition to other image filtering operations. Given a pair of source images, a source image and a filtered version of the source image, the application tries to approximate the parameters used in different image filters. These filters are then applied to a target image. The filter parameters are approximated using machine learning algorithms. The application is capable of producing painterly style output by learning

DOI: 10.1002/cav

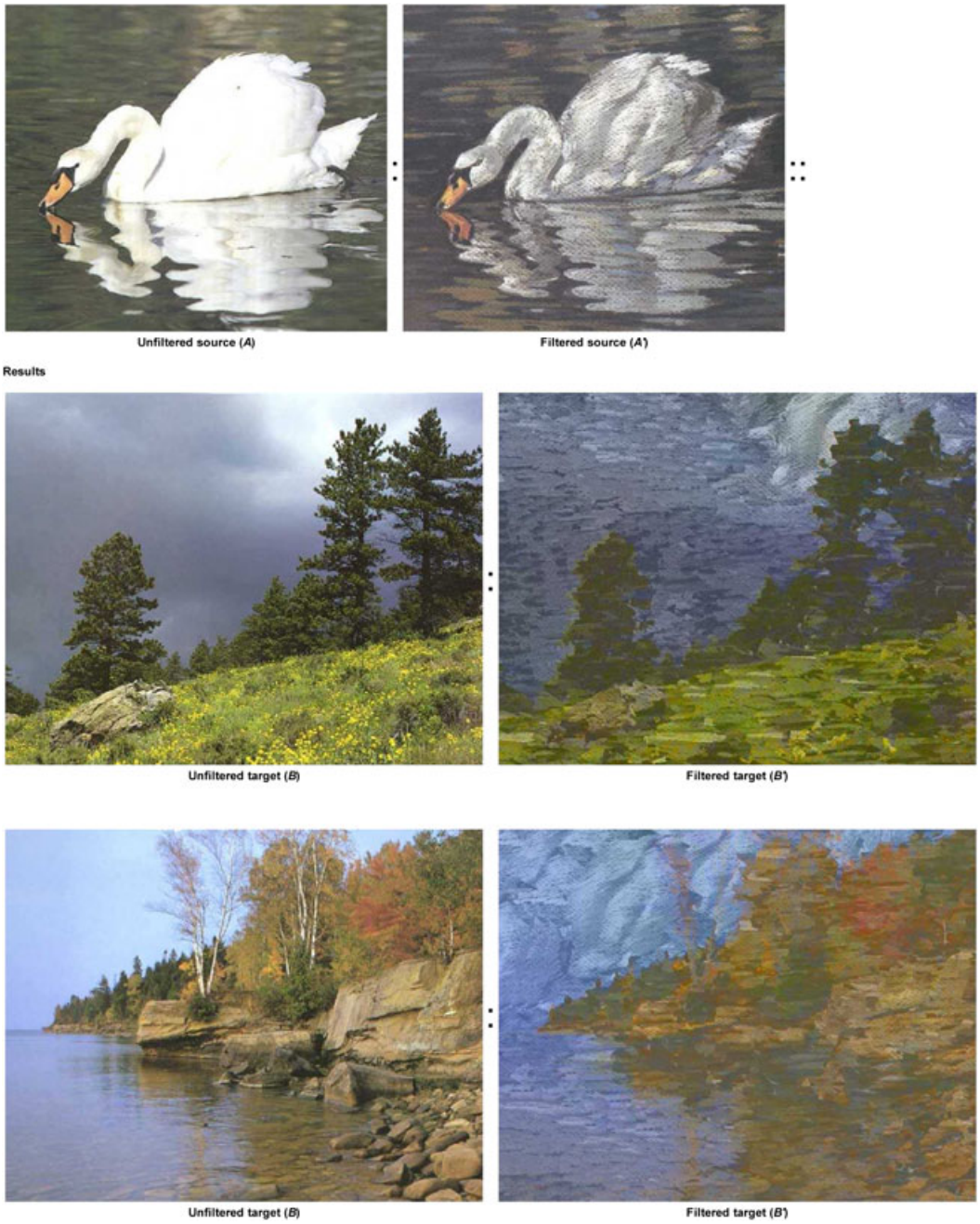

Figure 21. A sample input source images (A and A′) along with the target input images (B) with the result output (B′) produced using techniques presented in [71].

several artistic filters and is capable of producing oil, watercolour and other artistic style images. One drawback of this method is that it needs the original source image used, which may not always be available. An example of the outputs that can be produced can be seen in Figure 21.

Another popular choice for simulating strokes is the LIC algorithm [53]. Originally created to visualise vector fields, it is used by many techniques to create painterly style output.

Mao *et al.* [23] and Yamamoto *et al.* [24] used the LIC algorithm to generate pencil strokes. Properties of the strokes such as the length are controlled by varying the length of the convolution kernel. Width is controlled by varying the granularity of the underlying noise that is used as the source texture. Finally, the direction of the stroke is varied by generating a vector field within each region. Although the methods presented by these authors vary properties only within certain regions of the image,

Table I. Summary of painterly techniques reviewed.

Method	Source data	Output style	AE	Auto	TC
BKTS06 [66]	Img, 3D	Watercolour	None	High	Yes
BNTS07 [67]	Video	Watercolour	None	High	Yes
BST09 [57]	HF	Panorama maps	Novice	High	No
CH02 [29]	Img	Painterly	None	High	No
CRH05 [63]	Video	Painterly, CTN	Medium	Medium	Yes
CTM08 [55]	LV with 3D	Watercolour	None	High	Yes
GCS02 [27]	Img	Painterly	None	High	No
Hae90 [34]	Img	Painterly	None	Medium	No
HAY04 [54]	Img, Video	Painterly	None	High	Yes
Her98 [38]	Img	Painterly	None	High	No
Her02 [40]	Img, Video	Painterly	None	High	Yes
HP00 [60]	Video	Painterly	None	High	Yes
HMIA01 [71]	Img + Ref	SBR	None	Medium	No
KGC00 [65]	3D	SBR	Novice	High	Yes
KMM02 [35]	3D	SBR	Medium	Medium	Yes
KMN99 [44]	3D	SBR	Novice	High	Yes
KUL03 [70]	3D	Painterly	Medium	High	Yes
KYP11 [72]	Img, Video	Painterly	None	High	Yes
LD06 [56]	Plant geom	Watercolour	None	High	Yes
LEE09 [47]	Video	Painterly	None	High	No
Lit97 [36]	Img, Video	Painterly	None	High	Yes
LSF10 [61]	Video, 3D	Painterly	None	High	Yes
LSRY10 [68]	Img + Ref	Painterly	None	High	No
LZL10 [32]	Video	Painterly	None	Medium	Yes
MAO01 [23]	Image	Pencil	None	High	No
Mei96 [43]	3D	Painterly	Novice	Medium	Yes
MGS06 [52]	3D	Painterly	None	High	Yes
OMG05 [49]	Img	Painterly	None	Medium	No
PPC07 [25]	Img	Painterly	None	High	No
PY08 [37]	Video	Painterly	None	High	Yes
SAA02 [28]	Img	Painterly	None	Medium	No
SIMC07 [74]	Img	SBR	None	Medium	No
STRB05 [39]	3D	SBR	None	High	Yes
SY00 [41]	Img	Painterly	None	High	No
WAY02 [46]	Plant geom	Oriental	None	High	Yes
XLSN10 [58]	Img	Sumi-e	Novice	Medium	No
YAM04 [24]	Img	Pencil	None	High	No
YN07 [69]	3D + Ref	Painterly	None	High	Yes
XUPW04 [30]	Point set	Painterly	None	High	Yes
ZCZ09 [7]	Video	Oriental	None	High	Yes
ZZ10 [31]	Img	Abstract	None	Medium	No
ZZXZ09 [50]	Img	Painterly	None	Medium	No

3D: 3D mesh geometry; AE, artistic expertise; Auto, level of automation over the process; CTN, cartoon; HF, height field with land cover data; Img, 2D images; LV, live video; Ref, reference painted image; SBR, stroke-based output (painterly, mosaic, stippling and hatching); TC, temporal coherence.

more variation could be achieved by generating reference maps that determine stroke properties at every point within the image without the need to generate individual strokes. Kyprianidis *et al.* [72] overcame these limitations by estimating the second fundamental tensor at each point. The LIC algorithm uses the minor eigenvector of the tensor to smooth the image without affecting its edges. The length of the LIC convolution kernel is controlled by the curvature at that point. In low-contrast regions, the tensor is interpolated from neighbouring regions as it can be significantly affected by noise. Once smoothing is performed, the authors suggested using a directional sharpening filter to enhance edges that might be smoothed. The smoothing and sharpening are carried out over several iterations until the desired abstraction level is achieved.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have reviewed several techniques that have been proposed to create digital paintings. We

Comp. Anim. Virtual Worlds 2013; **24**:43–64 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/cav

observed a trend where early works such as [34,36,38] address basic issues in painting, whereas later research such as [31,35,50] focused on more challenging topics such as providing a generic framework for stroke-based techniques and abstracting details like a real artist would. Such methods allow us to create and experiment with content in new styles that would be prohibitively expensive using traditional techniques. Another challenge in this area is temporal coherence. From the methods that we reviewed, we found the work by Lin *et al.* [32] to produce comparatively robust results. Even with this, however, several shortcomings, including tracking scene changes, rapid motion between frames, translucent objects and fluid motion, are not addressed. Using 3D geometry can provide more robust results, but none of the methods reviewed address translucent and deformable objects.

In Table I, we present a summary of the different methods reviewed in this paper. The table offers a quick overview as to the reference data used by each method and the output styles generated. None of the low-level techniques are listed, as the output generated by these methods depends on the artistic expertise of the user using the system.

Although automation is desired, no method is perfect; hence, it is always favourable to provide tools so that the user can further define the final output. Methods such as [35,63,65] try to automate the entire process but always provide the user with tools required to tweak the output at any point. This is important because an artist would inevitably like to fine tune the final style of the output. Unfortunately, such methods would not work when real-time output is needed.

Finally, although many methods do provide tests and comparisons, such comparisons can be restricted to metrics such as computational efficiency and ease of use. While some properties of a proposed technique such as temporal coherence are easier to judge, other properties such as abstraction and style of output are more difficult to assess, as these are meant to invoke one's individual imagination. This is particularly pertinent to painterly styles. Gooch *et al.* [73] suggested a few methods to test NPR techniques. Most methods discussed in [73] are more relevant to NPR techniques that attempt to produce output for use in technical illustrations, medical imaging and educational purposes. A final option is to judge the output by conducting Turing tests, but even this might be too restrictive as computer generated painterly output may well define a new style, just as 8-bit or pixel art has done in the past. As long as a proposed method is visually appealing to a large-enough audience, it can be argued that it can potentially be seen as a reasonably valid contribution in this field.

REFERENCES

1. Gooch A, Gooch B, Shirley P, Cohen E. A non-photorealistic lighting model for automatic technical illustration, In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98*, Orlando, Florida, USA, 1998.
2. DeCarlo D, Santella A. Stylization and abstraction of photographs, In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '02*, San Antonio, Texas, USA, 2002; 769–776.
3. Durand F. An invitation to discuss computer depiction, In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering, NPAR '02*, Annecy, France; 111–124.
4. Solso RL. *Cognition and the Visual Arts*. MIT Press, Cambridge, Massachusetts, USA, 1996.
5. Petrov A. The old man and the sea, Montreal, 1999.
6. Te W. Shan shui qing, China, 1988.
7. Zhang SH, Chen T, Zhang YF, Hu SM, Martin R. Video-based running water animation in Chinese painting style. *Science in China Series F: Information Sciences* 2009; **52**(2): 162–171.
8. Lansdown J. Expressive rendering: a review of non-photorealistic techniques. *IEEE Computer Graphics and Applications* 1995; **15**: 29–37.
9. Hertzmann A. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications* 2003; **23**: 70–81.
10. Strassmann S. Hairy brushes. *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* 1986; **20**(4): 225–232.
11. Sousa MC, Buchanan JW. Observational model of blenders and erasers in computer-generated pencil rendering, In *Graphics Interface*, Kingston, Ontario, Canada, 1999; 157–166.
12. Sousa MC, Buchanan JW. Observational models of graphite pencil materials, In *Computer Graphics Forum*, Interlaken, Switzerland, 2000; 27–49.
13. Lee J. Simulating oriental black-ink painting. *IEEE Computer Graphics and Applications* 2002; **19**(3): 74–81.
14. Baxter B, Scheib V, Lin MC, Manocha D. DAB: interactive haptic painting with 3D virtual brushes, In *SIGGRAPH '01*, Los Angeles, California, USA, 2001; 461–468.
15. Zwicker M, Pauly M, Knoll O, Gross M. Pointshop 3D: an interactive system for point-based surface editing. In *ACM Transactions on Graphics (TOG)*, SIGGRAPH. ACM, New York, NY, USA; 322–329.
16. Adams B, Wicke M, Dutré P, Gross M, Pauly M, Teschner M. Interactive 3D painting on point-sampled objects, In *Eurographics Symposium on Point-Based Graphics*, Zurich, Switzerland, 2004; 59–66.
17. Curtis CJ, Anderson SE, Seims JE, Fleischer KW, Salesin DH. Computer-generated watercolor,

In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques,* SIGGRAPH '97, Los Angeles, California, USA, 1997; 421–430.

18. Takagi S, Fujishiro I, Nakajima M. Volumetric modeling of colored pencil drawing, In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society*, Seoul, Korea, 1999; 250–258.
19. Lee J. Diffusion rendering of black ink paintings using new paper and ink models. *Computers & Graphics* 2001; **25**(2): 295–308.
20. Chu NS, Tai CL. Moxi: real-time ink dispersion in absorbent paper. *ACM Transactions on Graphics (TOG)* 2005; **24**(3): 504–511.
21. Lee S, Olsen SC, Gooch B. Interactive 3D fluid jet painting, In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '06*, Annecy, France, 2006; 97–104.
22. Chu N, Baxter W, Wei LY, Govindaraju N. Detail-preserving paint modeling for 3D brushes, In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '10*, Annecy, France, 2010; 27–34.
23. Mao X, Nagasaka Y, Imamiya A. Automatic generation of pencil drawing from 2D images using line integral convolution, In *CAD/Graphics,* volume 9, Kunming, China, 2001; 240–248.
24. Yamamoto S, Mao X, Imamiya A. Colored pencil filter with custom colors, In *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, Seoul, Korea, 2004; 329–338.
25. Papari G, Petkov N, Campisi P. Artistic edge and corner enhancing smoothing. *IEEE Transactions on Image Processing* October 2007; **16**(10): 2449–2462.
26. Kuwahara M, Hachimura K, Eiho S, Kinoshita M. Processing of RI-angiocardiographic images. *Digital Processing of Biomedical Images* 1976: 187–203.
27. Gooch B, Coombe G, Shirley P. Artistic vision: painterly rendering using computer vision techniques, In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering, June, NPAR '02*, Annecy, France, 2002; 83–ff.
28. Santella A, DeCarlo D. Abstracted painterly renderings using eye-tracking data, In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering, NPAR '02*, Annecy, France; 75–ff.
29. Collomosse JP, Hall PM. Painterly rendering using image salience, In *Proceedings of the 20th Eurographics UK Conference*, Leicester, UK, 2002; 122–128.
30. Xu H, Gossett N, Chen B. Pointworks: abstraction and rendering of sparsely scanned outdoor environments, In *Proceedings of the 15th Eurographics Workshop on Rendering Techniques*, Norköping, Sweden, 2004; 21–23.
31. Zhao M, Zhu SC. Sisley the abstract painter, In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '10*, Annecy, France, 2010; 99–107.
32. Lin L, Zeng K, Lv H, Wang Y, Xu Y, Zhu SC. Painterly animation using video semantics and feature correspondence, In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, NPAR '10*, Annecy, France, 2010; 73–80.
33. Yao B, Yang X, Zhu SC. Introduction to a large-scale general purpose ground truth database: methodology, annotation tool and benchmarks, In *Proceedings of the 6th international conference on Energy minimization methods in computer vision and pattern recognition*, Ezhou, China, 2007; 169–183.
34. Haeberli P. Paint by numbers: abstract image representations. *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* 1990; **24**(4): 207–214.
35. Kalnins RD, Markosian L, Meier BJ, Kowalski MA, Lee JC, Davidson PL, Webb M, Hughes JF, Finkelstein A. WYSIWYG NPR: drawing strokes directly on 3D models. *ACM Transactions on Graphics (TOG)* 2002; **21**(3): 755–762.
36. Litwinowicz P. Processing images and video for an impressionist effect, In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97*, Los Angeles, California, USA, 1997; 407–414.
37. Park Y, Yoon K. Painterly animation using motion maps. *Graphical Models* 2008; **70**(1-2): 1–15.
38. Hertzmann A. Painterly rendering with curved brush strokes of multiple sizes, In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '98*, Orlando, Florida, USA, 1998; 453–460.
39. Schlechtweg S, Germer T, Strothotte T. RenderBots–Multi-Agent systems for direct image generation, In *Computer Graphics Forum,* volume 24, Dublin, Ireland, 2005; 137–148.
40. Hertzmann A. Paint by relaxation, In *Computer Graphics International (CGI'01)*, Hong Kong, China, 2001; 47–54.
41. Shiraishi M, Yamaguchi Y. An algorithm for automatic painterly rendering based on local source image approximation, In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering, NPAR '00*, Annecy, France, 2000; 53–58.

42. Velho L, de Miranda Gomes J. Digital halftoning with space filling curves, In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques,* volume 25 of *SIGGRAPH '91*, Los Angeles, California, USA, July 1991; 81–90.
43. Meier BJ. Painterly rendering for animation, In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '96*, New Orleans, Louisiana, USA, 1996; 477–484.
44. Kowalski MA, Markosian L, Northrup JD, Bourdev L, Barzel R, Holden LS, Hughes JF. Art-based rendering of fur, grass, and trees, In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99*, Los Angeles, California, USA, 1999; 433–438.
45. Schmid J, Senn MS, Gross M, Sumner RW. OverCoat: an implicit canvas for 3D painting, In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, ACM, New York, NY, USA, 2011; 28:1–28:10.
46. lor Way D, ru Lin Y, chung Shih Z. The synthesis of trees chinese landscape painting using silhouette and texture strokes. *Journal of WSCG* 2002; **10**: 499–506.
47. Lee H, Lee CH, Yoon K. Motion based painterly rendering, In *Computer Graphics Forum*, volume 28, Munich, Germany, 2009; 1207–1215.
48. Xu H, Chen B. Stylized rendering of 3D scanned real world environments, In *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '04, Annecy, France, 2004; 25–34.
49. Olsen SC, Maxwell BA, Gooch B. Interactive vector fields for painterly rendering, In *Proceedings of Graphics Interface 2005*, Kelowna, British Columbia, Canada, 2005; 241–247.
50. Zeng K, Zhao M, Xiong C, Zhu SC. From image parsing to painterly rendering. *ACM Transactions on Graphics (TOG)* 2009; **29**(1): 1–11.
51. Guo C, Zhu SC, Wu YN. Primal sketch: integrating structure and texture. *Computer Vision and Image Understanding* 2007; **106**(1): 5–19.
52. Ming-Te Chi TYL. Stylized and abstract painterly rendering system using a multiscale segmented sphere hierarchy. *IEEE Transactions on Visualization and Computer Graphics* 2006; **12**: 61–72.
53. Cabral B, Leedom LC. Imaging vector fields using line integral convolution, In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, Anaheim, California, USA, 1993; 263–270.
54. Hays J, Essa I. Image and video based painterly animation, In *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '04, Annecy, France, 2004; 113–120.
55. Chen J, Turk G, MacIntyre B. Watercolor inspired non-photorealistic rendering for augmented reality, In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, Bordeaux, France, 2008; 231–234.
56. Luft T, Deussen O. Real-time watercolor illustrations of plants using a blurred depth test, In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '06, Annecy, France, 2006; 11–20.
57. Bratkova M, Shirley P, Thompson WB. Artistic rendering of mountainous terrain. *ACM Transactions on Graphics (TOG)* 2009; **28**(4): 1–17.
58. Xie N, Laga H, Saito S, Nakajima M. IR2s: interactive real photo to sumi-e, In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, Annecy, France, 2010; 63–71.
59. Hsu SC, Lee IHH, Wiseman NE. Skeletal strokes, In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, UIST '93, Atlanta, Georgia, USA, 1993; 197–206.
60. Hertzmann A, Perlin K. Painterly rendering for video and interaction, In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '00, Annecy, France, 2000; 7–12.
61. Lu J, Sander PV, Finkelstein A. Interactive painterly stylization of images, videos and 3D animations, In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, Washington DC, USA, 2010; 127–134.
62. Matas J, Chum O, Urban M, Pajdla T. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing* 2004; **22**(10): 761–767.
63. Collomosse JP, Rowntree D, Hall PM. Stroke surfaces: temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics* 2005; **11**: 540–549.
64. Christoudias CM, Georgescu B, Meer P. Synergism in low level vision, In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, Quebec, Canada, 2005; 150–155.
65. Kaplan M, Gooch B, Cohen E. Interactive artistic rendering, In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '00, Annecy, France, 2000; 67–74.
66. Bousseau A, Kaplan M, Thollot J, Sillion FX. Interactive watercolor rendering with temporal coherence and abstraction, In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '06, Annecy, France, 2006; 141–149.

67. Bousseau A, Neyret F, Thollot J, Salesin D. Video watercolorization using bidirectional texture advection, In *ACM SIGGRAPH 2007 papers, SIGGRAPH '07*, San Diego, California, USA, 2007.
68. Lee H, Seo S, Ryoo S, Yoon K. Directional texture transfer, In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '10, Annecy, France, 2010; 43–48.
69. Yen CR, Chi TYLM-T, Lin WC. Stylized rendering using samples of a painted image. *IEEE Transactions on Visualization and Computer Graphics* 2007; **14**: 468–480.
70. Kulla CD, Tucek JD, Bailey RJ, Grimm CM. Using texture synthesis for non-photorealistic shading from paint samples, In *11th Pacific Conference on Computer Graphics and Applications (PG'03)*, Canmore, Canada, 2003; 477–481.
71. Hertzmann A, Jacobs CE, Oliver N, Curless B, Salesin DH. Image analogies, In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, Los Angeles, California, USA, 2001; 327–340.
72. Kyprianidis JE, Kang H. Image and video abstraction by coherence-enhancing filtering. *Computer Graphics Forum* 2011; **30**(2): 593–602. Proceedings Eurographics 2011.
73. Gooch AA, Long J, Ji L, Estey A, Gooch BS. *Viewing progress in non-photorealistic rendering through Heinlein's lens*, NPAR '10. ACM, Annecy, France, 2010. 165–171.
74. Schwarz M, Isenberg T, Mason K, Carpendale S. Modeling with rendering primitives: an interactive non-photorealistic canvas, In *Proceedings of the 5th International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '07, San Diego, California, USA, 2007; 15–22.

AUTHORS' BIOGRAPHIES

Siddharth Hegde is a PhD candidate at Bournemouth University, UK, at the School of Design, Engineering and Computing (Creative Technology Research Group). His current area of research is non-photorealistic rendering. He graduated from Kingston University, London, with an MSc in Computer Vision and Image Analysis. He has a keen interest in computer graphics and technology in general.

Christos Gatzidis is a Senior Lecturer in Creative Technology at Bournemouth University, UK. Additionally, he is a Visiting Research Fellow at the School of Informatics, Department of Information Science, City University London, where he completed his PhD, titled 'Evaluating Non-Photorealistic Rendering for 3D Urban Models in the Context of Mobile Navigation'. Furthermore, he has a Masters in Arts in Computer Animation from Teesside University and a BSc in Computer Studies (Visualisation) from the University of Derby. He has contributed to several refereed conference, book and journal publications and is also a member of the advisory board of three journals and various international program conference committees.

Feng Tian is an Associate Professor in the School of Design, Engineering and Computing (DEC) at Bournemouth University, UK. His research focuses on computer graphics, computer animation, NPR and so on. He has published over 50 papers in peer-reviewed international journals and conferences. Prior to joining Bournemouth University, he was an Assistant Professor in the School of Computer Engineering, Nanyang Technological University (NTU), Singapore.

Comp. Anim. Virtual Worlds 2013; **24**:43–64 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/cav