Video-Based Running Water Animation in Chinese Painting Style

Song-Hai Zhang
Tsinghua University
zhangsh@gmail.com

Tao Chen
Tsinghua University
generalmilk@gmail.com

Yi-Fei Zhang
Tsinghua University
macsyz@gmail.com

Shi-Min Hu
Tsinghua University
shimin@tsinghua.edu.cn

Ralph R. Martin
Cardiff University
ralph@cs.cf.ac.uk

Abstract

This paper presents a novel algorithm for synthesizing *animations* of *running water*, such as waterfalls and rivers, in the style of *Chinese paintings*, for applications such as cartoon making. All video frames are first registered in a common coordinate system, simultaneously segmenting the water from background and computing optical flow of the water. Taking artists' advice into account, we produce a painting structure to guide painting of brush strokes. Flow lines are placed in the water following an analysis of variance of optical flow, to cause strokes to be drawn where the water is flowing smoothly, rather than in turbulent areas: this allows a few moving strokes to depict the trends of the water flows. A variety of brush strokes is then drawn using a template determined from real Chinese paintings. The novel contributions of this paper are: a method for painting structure generation for flows in videos, and a method for stroke placement, with the necessary temporal coherence.

Keywords: Non-photorealistic rendering, Chinese painting, stroke based rendering, video rendering, optical flow

1 Introduction

Chinese painting is an ancient art form, in which natural objects are often depicted with few, yet expressive, brush strokes. It is of quite a different character to e.g. traditional Western art, where the goal of realism plays an important role, and to modern art, too. While image filtering [1] can be used to produce interesting non-photorealistic rendering effects, and can make images look similar to certain styles of Western art, filtering is not well suited to emulating all kinds of painting (see e.g. [2]). This is especially true where sparse brush strokes are used, as in Chinese painting, and different kinds of objects are drawn with different styles of stroke. Rendering in the style of Chinese paintings has mainly focused on physical simulations of water, ink, paper and Chinese brushes, leading to methods which can render various simple objects, such as rocks, leaves, and trees [3, 4, 5, 6, 7].

Animation in the same style began to appear in 1960 [8]. These differ markedly from cartoons produced in the West, and as a result their production often requires several times the amount of work needed for Western cartoons. For example, Chinese brush strokes have diffuse edges, while Western cartoons are generally drawn with sharp edges. Thus, simple interpolation of keyframes as used in Western animation

is not appropriate to Chinese animation. Xu [9] quotes [8] as saying that the famous 18-minute video, "Shan Shui Qing" (Love of Mountains and Rivers) probably took several tens of man-years to make.

Being able to produce animations, or even parts of scenes, directly from real video, could save much work. In turn, the reduced production costs could widen their scope of application to further areas, such as advertising. Many Chinese paintings contain landscapes, depicting mountains, stones, trees, and water, providing a backdrop to the characters who give the plot. Automatically creating such scenery in a reasonably interesting manner would free artists' time to apply their creative skills to the more important aspects of the animation.

When making such a backdrop, moving water must be treated differently from other, static, background elements. Rendering animations of waterfalls, rivers, and so on is thus particularly relevant when devising computer tools to assist animation making in a Chinese painting style.

Chinese painting often portrays nature. Objects in such pictures are generally faithful to their real shapes and appearances, although rendered using sparse brush strokes. This justifies an attempt to use *real* images or videos as a basis for Chinese-painting style rendering. Use of geometric models is an alternative for generating Chinese painting animations [5], but it is tricky to construct and animate complex scenes and objects in this way. Such an approach is inappropriate for flowing water due to the rapidly and constantly changing water shapes, and the need for tricky physical simulations to generate appropriate water behaviour.

We present an efficient framework for rendering videos of waterfalls and similar rapid water flows in a Chinese painting style, based on modeling the water flows in an existing video, and using them to choose and place brush strokes. Very little user interaction is needed. The user must provide a stroke template representing the range of brush strokes to be used to render the water in a chosen style. In principle this could be learnt by user presentation of example brush strokes from an existing painting, and their automated analysis by image processing methods.

In our method, we assume that the flow is generally in a steady state, i.e. the gross motion of the flow is relatively unchanging in any given area, over the time of the video, although there may also be e.g. splashes and ripples. The camera motion may include both rotation and panning.

To generate the painting structure and model individual flows of water, the system initially registers each frame of the video into a common frame of reference, simultaneously removing any camera motion, and estimating optical flow at each pixel of each frame. Flow lines are placed statically in the water following an analysis of variance of optical flow, to cause strokes to be drawn where the water is flowing smoothly, rather than in turbulent areas. A few moving strokes are then used to depict the trends of the water flows; their exact parameters come from an analysis of the input video, and the stroke template. The strokes always move smoothly along the predetermined flow lines, as the video progresses, guaranteeing temporal coherence of the output.

The static background may be rendered with any desired image NPR method before being added back to the rendered result. The output video is generated by adding back the camera motion.

Section 2 discusses related work. Details of our method are given in Section 3. Results are provided in Section 4, and Section 5 concludes the paper.

2 Related work

This section discusses related research concerning rendering, mainly in a Chinese style, and video texture generation.

2.1 Stroke based rendering

Stroke placement for non-photorealistic rendering has been extensively considered. Hertzmann [10] gave a greedy algorithm for placing oil-painting strokes. Salisbury [11] discussed how to place strokes in pencil sketches. Many such methods are based on edge maps. Instead, for the moving water, we use a *flow variance* map.

Producing results with spatio-temporal coherence is an important requirement for video, to

avoid flickering. Both [12] and [13] use optical flow to guide the movement of brush strokes from frame to frame. We also use optical flow, but in a different way. We firstly determine steady-state water flows over the whole video, giving *unchanging* flow lines. These are used as the underlying trajectories of *moving* brush strokes. Parameters determining the appearance and speed of the brush strokes are extracted from the video, and from a template representing a given brush style. This allows us to generate consistent and coherent visual effects for our particular problem, running waters.

An alternative method [14] produces successive frames of video by warping the current frame to account for changes determined by optical flow, and then overpainting areas of the new frame that differ significantly from their predecessors. This idea is extended in [15] to place strokes using an energy term depending on both pixel color differences and optical flow. However, this work paints dense brush strokes, rather than the few, carefully chosen, strokes used in Chinese painting.

Suzuki [16] uses an edge detection method to extract running water flow lines in order to paint strokes. However, his method is unsuited to rapidly running water: we do not believe it desirable in such cases to to draw strokes where there are edges in the image (see later). It is also difficult to enforce spatio-temporal coherence on strokes determined solely by edge detection on a frame-by-frame basis.

2.2 Chinese painting simulation

Research on rendering in a Chinese painting style has mainly considered simulation of the interaction of water, ink, paper and brushes. Strassmann [3] proposed a *hairy brush* model aimed at reproducing Japanese *sumi-e*; Chinese painting is similar. Position and pressure are sampled to give control points determining the stroke location and width. Much subsequent work has used a similar approach to modeling strokes.For example, [5] discussed how to draw trees in a Chinese style given an input skeleton. Such methods model the brush as having various shapes, sizes and patterns in 2D to form the desired stroke. Lee [17] gave a more sophisticated 3D bristle model for oriental brush painting, which considers the bristles as having one end fixed in a handle perpendicular to the paper. Their bending is modelled as the brush is applied to the paper. Baxter [18], on the other hand, used springs to model 3D brush deformation.

Most work on brush models concentrates on how to *model* the brush, rather than how to *control* the brush. The user still needs to be skillful to make a painting. In contrast, we concentrate on the problem of automatically determining from the video *where* to place the brush strokes, as well as the other parameters needed to *control* e.g. their width and opacity. In this respect, [9] is much more similar to our work, as it also uses knowledge of existing strokes to create animated Chinese paintings. However their input is an image, not a video, and they base segmentation on similarly coloured regions, which is not appropriate for flowing water.

Interaction of brushes with the special Xuan paper used for Chinese painting is particularly important. Nelson [19] gave a three-layer paper model and used a lattice Boltzmann equation to simulate ink flow and deposition. Using a more complex paper model can provide more realistic brush stroke appearance. However, as a tradeoff between efficiency and effect, we do not model the paper directly, but rather learn the characteristics of *existing* strokes already drawn on paper, i.e. so that we model the *results* produced jointly by brush and paper in existing drawings. This model also includes, for example, random variations in the stroke width, further simulating interaction of brush and paper.

2.3 Video textures

Video textures [20] are a constantly varying sequence of images representing some random pattern over time, such as smoke, or a waterfall for example. Various authors have proposed synthesis and editing tools for video textures; some of these tools are also relevant to our problem. For example, Bhat [21] analyses the speed of flowing textures in video frames. However, unlike our method, his needs laborious user assistance to help mark the flow. We know of no previous attempts to render video textures in a Chinese-brush-stroke style.

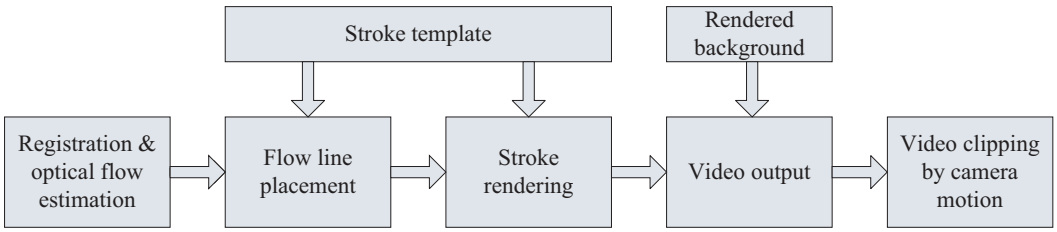

Figure 1: Processing pipeline

3 Method

We now explain our method for making an animated Chinese drawing of flowing water.

3.1 Overview

An overview of our pipeline for converting a real video of running water, such as a waterfall or fast flowing river, into a video cartoon in Chinese painting style, is shown in Fig 1.

Many input videos of interest have camera motion—often the camera is panned or rotated to capture the whole scene. It is necessary to eliminate the camera motion to successfully model the water flow—we need to separate optical flow due to water flow, from that due to camera motion. It is also important to take into account that optical flow computations suffer from large errors near the edges of the image, which is made more problematic due to camera motion (see [12]). Various work has considered video registration of dynamic textures. We follow the approach used in [22] to perform this task, and to place all frames of the video into a common coordinate system, free of camera motion.

We then compute a dense optical flow, for each pixel of the water for each frame, to control stroke generation. This is then analysed to determine the painting structure, that is, where to *place flow lines*, representing water flows, in the output video. Various brush strokes are placed on the flow lines to *render* them in a specific artistic style.

Although different painting styles may use different types of brush strokes, the underlying flow lines upon which the brush strokes are placed are determined by patterns in the water itself. The exact placement of each brush stroke on a given flow line, and its width, opacity, and other parameters are determined by the water flows, and by information in a *stroke template* based on real Chinese paintings. Different templates allow different output styles: typical Chinese painting style uses a few, long, smooth curves, but e.g. a style using shorter, straighter strokes could also be used.

We finally *composite* the rendered water flow with the overall background spanning all frames. The background can be rendered by an artist by hand (at relatively little expense as only one drawing is needed for all frames) or it can be rendered using some NPR technique. Each frame in the resulting panoramic video is then *clipped* using the location of the corresponding original video frame, and the camera motion *restored* to give the final output video.

We now discuss details of our approach.

3.2 Painting structure generation

The process of register the video into a common frame of reference simultaneously provides the optical flow at each pixel in each frame. We use the latter information to determine the painting structure upon which output strokes are to be placed. This structure is composed of a discrete set of flow lines that depict the trends of water motion. Simply using edge detection and placing flow lines at or near edges in the flow leads to unsatisfactory results.

Discussions with an artist indicate that brush strokes should illustrate the important flow motions of the water, rather than local turbulence and splashes. Such motions correspond to places where the flow has relatively consistent direction from frame to frame. This observation is independent of how the flow is to be rendered. We thus detect areas of steady flow, and turbulent areas, and then lay down a sparse set of flow lines which remain *fixed in location* during the entire video. Nevertheless, the placement

of brush strokes *on* these flow lines *varies* from frame to frame. The brush strokes move at the rate of water flow, as if carried along by the flow.

We first *compute optical flow spatial variance* at each pixel within the water. We define this as the variance of the angle between the optical flow, and the optical flows taken over a window of fixed size (we use 5×5) around the given pixel. For stability, each optical flow used is a temporal average over all frames for which it is available: we assume the water motion does not change in a significant manner over time. A similar assumption is made in [21] concerning stable flow lines in water, although in that case the flow lines are constructed manually. High flow variance indicates an area of turbulent water, whereas low variance indicates a steady flow.

We next decide the layout of the flow lines to use. Each flow line should be in an area of smooth flow, and go between (but not into) areas of turbulence. We thus first locate the point with *maximum* value in the flow variance map—a turbulent area—and then follow the flow until we are just outside the turbulent area, i.e. the variance falls below a threshold value. We then start making a flow line, and trace it along the optical flow, until one of three conditions occurs, terminating the flow line: (i) a variance threshold is exceeded—the flow line has entered another area of turbulence, or (ii) the flow line meets the boundary of the video, or (iii) the flow line meets a flag value of $-\infty$, which indicates another flow line has already been drawn close to this point. The last condition prevents flow lines from being drawn too close together. From the same starting point, we also construct a flow line in the opposite direction, following the flow backwards. Fig. 2 shows a waterfall image with superimposed optical flow, while Fig. 3 shows the corresponding variance map together with the first two flow lines drawn, going forward and backwards from the maximum value in the variance map. The threshold used above to decide what is a large variance is not critical. In practice, most pixels have a small variance, and relatively few pixels have a much larger variance. A threshold of 3 times the median value of variance works well in practice.

We need to draw many flow lines, not just one, and we must prevent flow lines from being placed too densely. Thus, after a flow line has been placed in the image, we compute a region of influence around it by growing it outwards by a certain amount. The width of this region is given by the brush template, and determines the spacing between flow lines. All pixels in the flow variance map corresponding to the region of influence are set to $-\infty$, which prevents any subsequent flow line from being drawn in this region. After placing these flag values in the variance map, we repeat the process, seeking the remaining maximum value to place the next flow line, and so on, until the maximum value in the flow variance map is below a threshold. Finally, any flow lines which are less than the minimum brush stroke length allowed by the template are discarded. Fig. 6(left) shows the

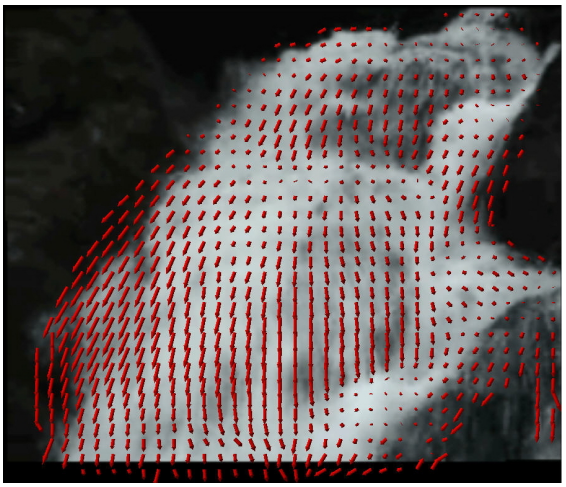

Figure 2: Smoothed optical flow in a waterfall.

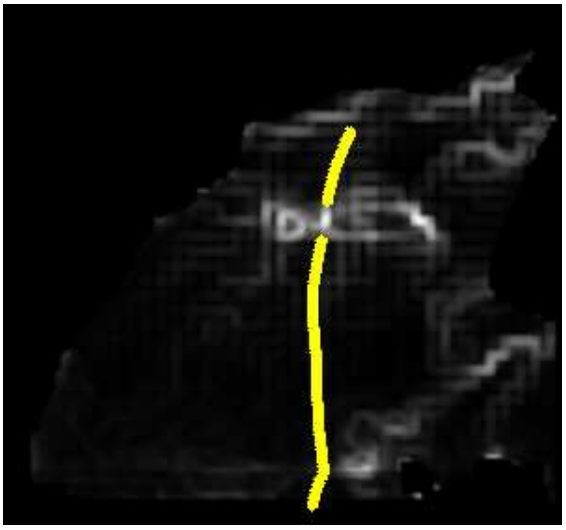

Figure 3: Optical flow variance map, with first forward and backward flow lines.

Figure 4: Above: a frame from the film 'Mudi'. Below: brush strokes sampled from this scene.

final painting structure for this example, which has 22 flow lines.

3.3 Brush stroke modeling

The video may be rendered in a variety of styles, which is achieved based on a template describing typical and permissible strokes for a given artistic style. We use a template constructed by analysing existing Chinese animations or drawings, but clearly brush stroke templates could be formed by analysing any other kind of drawing, or created at an artist's whim. Currently, we construct these templates by analysing about 10 selected example strokes by hand. In principle, automated tools could be written to perform much of this analysis. Some example strokes are shown in Fig. 4(below). As well as analysing the strokes themselves, we also determine lengths of gaps between strokes.

Each stroke is represented by a parameterised spline model which controls the stroke's details: for water in Chinese painting, relatively simple brush strokes are typical. Certain parameter values come from the template, while others are extracted from the video itself. Our stroke model (see Figure 5) is a strip based on a cardinal spline skeleton, plus fade-in and fade-out regions at each end. The strip is defined by $C(u) = (p(u), w(u), o(u), \alpha(u))$ where $u \in [0,1]$.

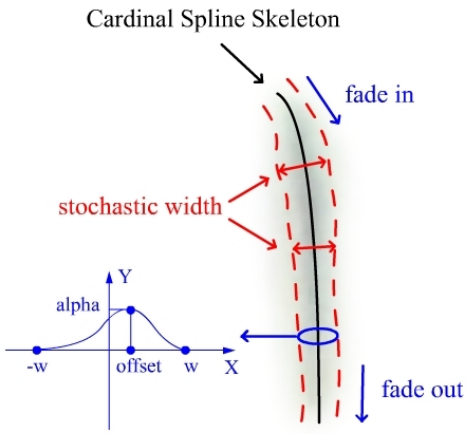

Figure 5: Cardinal spline skeleton model.

$p = x, y$ denotes the skeleton curve's coordinates relative to a local origin. w is the varying width of the stroke. α models the opacity of the ink; $\alpha \in [0,1]$, where 1 indicates total opacity. The opacity of the ink varies across the stroke; o represents an offset between the skeleton curve, and the transverse position of maximum opacity.

The strip itself is represented by:

$$S(u,v) = p(u) + v(w(u) + w_r(u))\hat{N}(u) / \left\|\hat{N}(u)\right\|$$

where $v \in [-1,1]$ and $\hat{N}(u)$ is the normal to $p(u)$. $w_r(u)$ is a random perturbation in the range $[0, \delta w(u)]$, used to simulate irregularity at the edges of the brush stroke arising from interaction between brush and paper due to friction. δ, the edge effect width, is a small constant. The ink opacity is given by $\alpha(u,v) = \alpha(u)\tilde{N}(v)$. $\tilde{N}(v)$ is used to vary opacity across the stroke:

$$\tilde{N}(v) = \begin{cases} N(\frac{v-o(u)}{1-o(u)}) & \text{if } v \geq o(u) \\ N(\frac{o(u)-v}{o(u)+1}) & \text{otherwise} \end{cases},$$

where $N(x)$ is a Gaussian function, of unit height and width: $N(x) = \exp(-9x^2/2)$. The fade in and fade out regions at the ends of the stroke simply continue the model at the ends of the strip, linearly reducing the opacity to zero over a short length at each end.

Certain parameters of the model, e.g. stroke intensity, are determined by information from the video frames, as explained later. Other parameters, e.g. the width of the strokes, come from the template. Further parameters depend

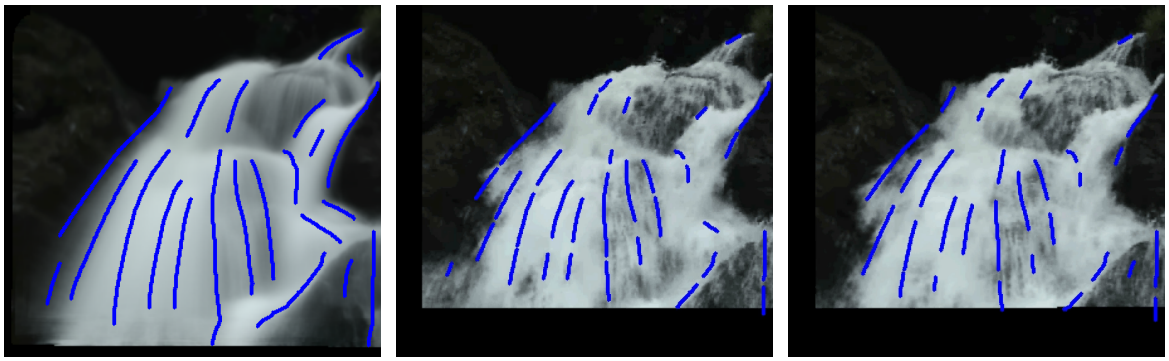

Figure 6: Left: flow lines. Middle: brush stroke locations, frame 0. Right: stroke locations, frame 3.

on both video and template. In the stroke template, for certain quantities, minimum, maximum and average values are stored, to allow variability in the output strokes. When a value of this kind is required, a value is generated at random, within the limits, with greater probability of obtaining a value nearer the average. Values of this kind are length of stroke; length of gaps between strokes along a flow line; width and opacity offset at start, middle, and end of the stroke; and region of influence size (used to prevent strokes being drawn too close together). Other values in the template are modelled as simple values. These are the fade-in and fade-out lengths, edge effect width, and maximum permissible turning angle for the stroke.

3.4 Brush stroke placement

For each frame, we must now decide on the strokes to draw along each flow line: generally, multiple strokes are rendered for each flow line, and while the flow lines are located at fixed positions in the water throughout the video, the strokes move along the flow lines from frame to frame. For each stroke, we must determine its control points, including those for position, opacity, and width. This must be done so that strokes are placed coherently from frame to frame. Coherence is further achieved by the flow lines being located at static positions.

Most flow lines are painted with several brush strokes, as most flow lines are longer than the maximum stroke length. In the first frame we randomly split long flow lines into several brush strokes, favouring the average length, and ensuring that all stroke lengths are between the minimum and maximum, and that no stroke turns through more than the maximum permissible turning angle. Lengths of gaps between the strokes are also determined by the template. The control points locating the stroke are placed on the flow line; use of cardinal splines has the advantage of avoiding the need for any fitting process to determine control points. The fade-in and -out lengths come directly from the template, as do the width of the stroke, and edge effect width. The average intensity of the stroke comes from the average intensity of the corresponding pixels in the input video. The opacity offset comes from differences in optical flow values from the average across the stroke, constrained by the maximum offset value. With these parameters, the stroke can now be drawn.

In order to achieve interframe coherence, when computing strokes for successive frames, we move the position of each stroke in the previous frame according to the average optical flow of its position control points. Each stroke is clipped to the ends of the flow line to which it belongs. Any strokes which as a result become too short are discarded; if a gap exists at the start of a flow line longer than the minimum brush stroke length, a new brush stroke is placed there. After moving each stroke, we recompute its intensity and intensity offset values. Fig. 6(left) shows flow lines placed in the scene in Fig. 3; Figs. 6(middle, right) show the locations of brush strokes placed in the first frame, and 3 frames later. Note that the individual frames are smaller than the overall registered video size after camera motion compensation. For expressing the rapid water flows, we place two groups of brush strokes along the flow line if the flow speed is larger than a threshold. and

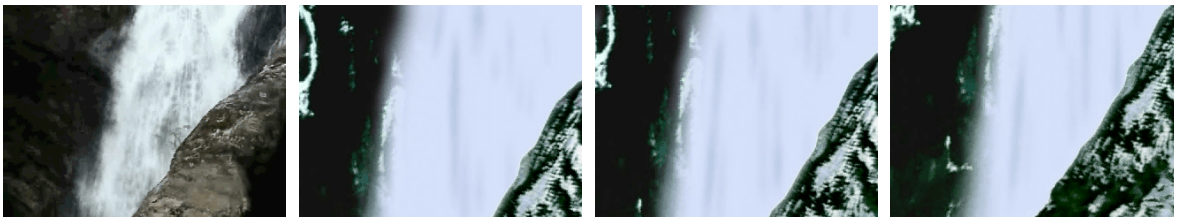

Figure 7: Left: an input video frame showing a waterfall. Others: frames generated by our algorithm.

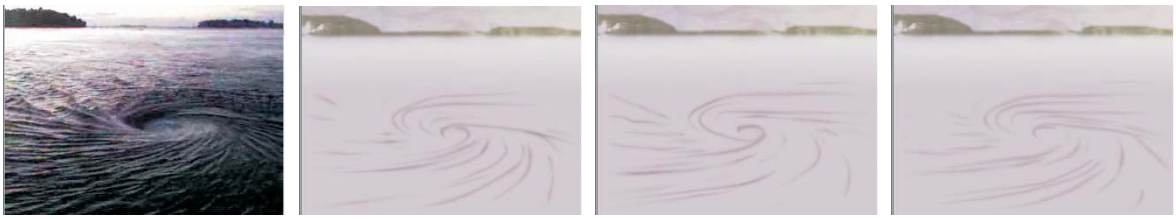

Figure 8: Left: an input video frame showing a whirlpool. Others: frames generated by our algorithm.

randomly extend the flow lines(see Fig. 9).

4 Results

We now present example results generated by our method. In all tests we used a PC with a Pentium 4 CPU running at 2.4GHz with 512MB memory.

Firstly, in Fig. 7, we show frames from a rendering of a waterfall in Chinese painting style, using a brush template manually extracted from the prize-winning film "Mudi" (see Fig. 4). We believe we have successfully recreated its style to a fair degree in our results. Secondly, in Fig. 8, we show rendered output frames from a whirlpool, using the same rendering style. Note that this has very different flow patterns. Thirdly, in Fig. 9, we show rendered output frames from a second waterfall using the same rendering style. Note that this has both almost vertical flows, and almost horizontal flows, in different parts of scene. Finally, we show rendered output of a frame from a third waterfall again in the same style (see Fig. 10(left)), and two frames from the rendering of a river, in two different brush styles, (i) using the same brush template as the waterfall examples, and (ii) using a synthetic brush style (see Fig. 10(middle, right)).

During this work, we solicited guidance from the artist. In his opinion, the results produced are of an acceptable quality for commercial production of background animations for cartoons.

Table 1 gives the processing times, per frame, required to produce these animations. Overall, these times are acceptable for real use. They allow a short sequence of frames to be rendered quickly while trying different input videos, or brush templates. On the other hand, the time taken to render a whole video is much shorter than would be needed for hand animation.

5 Conclusions

Overall, our method produces results of running water animations comparable those seen in real Chinese video animations, and does so with acceptable speed and quality. The main novel ideas are the steps for painting structure generation based on flow variance map and chinese brush stroke placement. Nevetheless, our method currently has certain limitations, which we intend to address in our future work.

Our approach to stroke placement is quite simplistic, even if reasonably effective. Local statistics of water flows are used to place strokes, but more sophisticated rules could be used. For example, an artist has indicated to us that strokes might be drawn differently in different parts of the flow, e.g. at the top and the middle of the waterfall, and might be drawn differently according to the speed of the flow as well as its direction. One way to do this would be to store the strokes in the template, and let the user simply indicate

Figure 9: Rendered output of a second waterfall, with almost vertical and horizontal flows.

Figure 10: Rendered output of a third waterfall, and a river using two different brush templates.

Video	Fig.7	Fig.8	Fig.9	Fig.10(left)	Fig.10(right)
Resolution	320×240	320×240	640×272	320×240	400×224
Panoramic video size	360×400	360×280	640×272	320×280	1096×512
Number of flow lines	10	24	69	22	73
Video registration	3s	3s	4s	3s	5s
Stroke placement	1s	1s	1.5s	1s	1.5s
Stroke rendering	1s	1s	1s	1s	2s

Table 1: Processing times (seconds per frame of rendered output).

which stroke or strokes to use from the template in different regions of the waterfall, or which strokes to use where the water is moving at different speeds. Such an approach would require detailed investigation to achieve the best effects, considering both artistic and scientific aspects of the problem.

There is also scope to replace the camera motion of the source video by a new, user-defined camera motion. Filling in the background in all frames is straightforward, but synthesis of unrecorded spatiotemporal regions of the waterfall would require generalisation of the strokes in a spirit akin to video textures [20]. This approach could also be used generalise the water flows to longer, or even endless, animations. Such extensions would allow greater editorial control over the animation produced, even to the extent of viewing the waterfall in a quite different way to that in which it was originally filmed.

Acknowledgements

This work was partially supported by the National Basic Research Project of China (Project 2006CB303105), the Specialized Research Fund for the Doctoral Program of Higher Education of China (Project 20060003057), and an EPSRC UK travel grant.

References

[1] H. Winnemöller, S. C. Olsen, and B. Gooch. Real-time video abstraction. *ACM Trans. Graph.*, 25(3):1221–1226, 2006.

[2] J. P. Collomosse and P. M. Hall. Cubist style rendering from photographs. *IEEE Trans. Vis. and Comp. Graphics*, 9(4):443–453, 2003.

[3] S. Strassmann. Hairy brushes. In *SIGGRAPH '86*, pages 225–232, New York, NY, USA, 1986. ACM Press.

[4] H. T. F. Wong and H. H. S. Ip. Virtual brush: A model-based synthesis of Chinese calligraphy. *Computers & Graphics*, 24:99–113, 2000.

[5] D.-L. Way, Y.-R. Lin, and Z.-C. Shih. The synthesis of trees in Chinese landscape painting using silhouette and texture strokes. *WSCG*, 10(3):499–507, 2002.

[6] D.-L. Way and Z.-C. Shih. The synthesis of rock textures in Chinese landscape painting. *Computer Graphics Forum*, 20(3):123–131, 2001.

[7] N. S. H. Chu and C.-L. Tai. Real-time painting with an expressive virtual Chinese brush. *IEEE Comput. Graph. Appl.*, 24(5):76–85, 2004.

[8] J. Chen and J. Zhang, editors. *Dictionary of Chinese Films (in Chinese)*. Shanghai Dictionary Press, 1995.

[9] S.-H. Xu, Y.-Q. Xu, S.-B. Kang, D. H. Salesin, Y.-H. Pan, and H.-Y. Shum. Animating Chinese paintings through stroke-based decomposition. *ACM Trans. Graph.*, 25(2):239–267, 2006.

[10] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98*, pages 453–460, New York, NY, USA, 1998. ACM Press.

[11] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *SIGGRAPH '97*, pages 401–406, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[12] J. Hays and I. Essa. Image and video based painterly animation. In *NPAR '04: Proc. 3rd Int. Symp. Non-photorealistic animation and rendering*, pages 113–120, New York, NY, USA, 2004. ACM Press.

[13] P. Litwinowicz. Processing images and video for an impressionist effect. In *SIGGRAPH '97*, pages 407–414, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[14] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *NPAR '00: Proc. 1st Int. Symp. Non-photorealistic animation and rendering*, pages 7–12, New York, NY, USA, 2000. ACM Press.

[15] A. Hertzmann. Paint by relaxation. In *Proc. Computer Graphics Int. 2001*, pages 47–54, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[16] J. Suzuki, Y. Dobashi, and T. Yamamoto. A method for painterly rendering of water surface. *J. Inst. Image Elect. Eng. of Japan*, 34(4):302–310, 2005.

[17] J.-T. Lee. Simulating oriental black-ink painting. *IEEE Comput. Graph. Appl.*, 19(3):74–81, 1999.

[18] B. Baxter, V. Scheib, M. C. Lin, and D. Manocha. Dab: interactive haptic painting with 3d virtual brushes. In *SIGGRAPH '01*, pages 461–468, New York, NY, USA, 2001. ACM Press.

[19] N. S. H. Chu and C.-L. Tai. Moxi: real-time ink dispersion in absorbent paper. In *SIGGRAPH '05*, pages 504–511, New York, NY, USA, 2005. ACM Press.

[20] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *SIGGRAPH '00*, pages 489–498, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[21] K. S. Bhat, S. M. Seitz, J. K. Hodgins, and P. K. Khosla. Flow-based video synthesis and editing. In *SIGGRAPH '04*, pages 360–363, New York, NY, USA, 2004. ACM Press.

[22] H. Pashley, J. H. Hays, Y. X. Liu, J. Z. Huang, X. L. Huang, and D. Metaxas. Optimization and learning for registration of moving dynamic textures. pages 1–8, 2007.