SCIENCE CHINA
Information Sciences

Painting patches: Reducing flicker in painterly re-rendering of video

SongHai ZHANG[1,*], Qiang TONG[1], ShiMin HU[1] and Ralph MARTIN[2]

Citation: SCIENCE CHINA Information Sciences **54**, 2592 (2011); doi: 10.1007/s11432-011-4409-2

View online: https://engine.scichina.com/doi/10.1007/s11432-011-4409-2

View Table of Contents: https://engine.scichina.com/publisher/scp/journal/SCIS/54/12

Published by the Science China Press

SCIENCE CHINA
Information Sciences

• RESEARCH PAPERS •

December 2011 Vol. 54 No. 12: 2592–2601
doi: 10.1007/s11432-011-4409-2

Painting patches: Reducing flicker in painterly re-rendering of video

ZHANG SongHai[1]*, TONG Qiang[1], HU ShiMin[1] & MARTIN Ralph[2]

[1]*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China;*
[2]*School of Computer Science and Informatics, Cardiff University, Cardiff, Wales* CF24 3AA, *UK*

Received July 12, 2010; accepted January 5, 2011; published online October 17, 2011

Abstract This paper presents a novel method for re-rendering video in a stroke-based painterly style. Previous methods typically place and adjust strokes on a frame by frame basis, guided by an analysis of motion vectors. Our method constructs painting patches which last for multiple frames, and paints them just once, compositing them after placing and clipping each one in each output frame. Painting patches are constructed by clustering pixels with similar motions, representing moving objects. This is done using a multi-frame window, to take account of objects which are present in consecutive frames, and which may occur a few frames apart with occlusion. The appearance of a given cluster across a sequence of frames is warped to a common reference to produce the painting patch; a global optimization of the warp is used to minimize distortion in the painting strokes. This approach outperforms prior algorithms in problem areas of the image, where flickering typically occurs, while producing comparable results elsewhere. In particular, stable strokes are produced at occlusion boundaries where objects emerge, and at image borders exposed by camera panning. A further advantage is consistent rendering of regions before and after brief occlusion, enhancing temporal stability of the output of discontiguous frames.

Keywords non-photorealistic rendering, painterly rendering, stroke placement, region clustering, video rendering, flicker

Citation Zhang S H, Tong Q, Hu S M, et al. Painting patches: Reducing flicker in painterly re-rendering of video. Sci China Inf Sci, 2011, 54: 2592–2601, doi: 10.1007/s11432-011-4409-2

1 Introduction

Computer-based artistic re-rendering software is of interest for the production of paintings or animations from captured images or videos. Such tools can free artists from tiresome work, and have the potential to make such output readily available for a wide range of application areas such as advertising and marketing, special effects for film and TV, and home entertainment.

Computer generated artistic re-rendering of paintings has typically tried to emulate the huge range of existing artworks; in particular with respect to the placement, shapes and styles of brush strokes. The author of [1] gave a survey on a typical framework for re-rendering images based on modeling physical painting processes. While such emulation is appropriate for images, it is harder to do for video. Due to

*Corresponding author (email: zhangsh@gmail.com)

the huge amount of work needed to produce them, relatively few animations can be found for use as a basis for guiding computer re-rendering of video in a painterly style. Nevertheless, they do exist: "The old man and the sea" is an excellent example of high aesthetic quality, painted on glass, by Alexander Petrov. Studying such examples leads us to conclude that high-quality output depends on *highly coherent brush stroke placement* for each object as it moves about the scene.

Existing methods for painterly re-rendering of video [2–5] are, however, all based on a scheme of placing and adjusting strokes *frame by frame*, guided by motion vectors. Such schemes are straightforward, but are unable to produce coherent brush strokes where motion vectors are ill-defined, most typically where there is occlusion of moving objects. Problems arise as strokes gradually appear in such areas as (i) the frame boundaries, or (ii) where one object emerges from behind another. Furthermore, where an object is occluded briefly for a few frames, motion vectors are not available, and as a result, such objects are not rendered consistently before and after its occlusion. Such problems are independent of the particular stroke placement method used, and significantly decrease the aesthetic quality of the output.

To overcome these problems, we have devised a technique to ensure highly coherent brush placement for moving objects. Using video object segmentation, we construct *painting patches* which last for multiple video frames; each patch corresponds to a set of pixels with a similar motion trajectory within frames. A patch thus represents a moving image which may last for as little as one frame, or at the opposite extreme, for the whole video. To construct the painting patches, we cluster pixels with similar motions, using a window of frames about the current frame: this enables us to follow moving objects which are briefly occluded for a few frames. A random walk model is used to segment each image taking into account the clustering between multiple adjacent frames. The pixels belonging to a given cluster in each frame do not necessarily represent a simple homography, but may contain local deformation from frame to frame, indicated by the optical flow. They must thus be warped and merged into a common coordinate system to produce the painting patch. We paint each patch just once, and then composite the patches after transforming them into each output frame, trimming them as needed. However, as Hertzmann [3] notes, if brush strokes are distorted, this can make the result look unnatural, and unlike a painting. Our warping algorithm is thus designed to minimize distortion of the brush strokes when regions are mapped to the common coordinate system: it minimizes the summed distortion of the regions of the image corresponding to the patch over all frames.

Overall, this approach overcomes the abovementioned shortcomings of previous methods. We achieve much greater stability of strokes between frames, producing animations with less flicker and greater coherence, and which have a natural, painted appearance.

2 Related work

Various papers have been devoted to painterly rendering of images [2,6–9]. The framework of multi-scale stroke based rendering [7] is based on real painting processes, and produces results of high aesthetic quality. Alternative schemes of stroke placement based on this framework differ mainly in terms of stroke starting point selection, the texture or lighting of the stroke, and choice of stroke direction. For example, if stroke direction is based on image gradients, it can produce a van Gogh-like style painting.

Techniques have also been developed to make the paintings move, and to produce animated rendering from video. Optical flow is typically used to determine movement of the strokes' control points between frames, both to make the strokes appear to be attached to the moving objects, and to align the strokes with the direction of movement. Litwinowicz [2] uses Delaunay triangulation of stroke control points to determine when adding new strokes (if existing ones are too far apart), and deleting strokes (if they are too close). Hertzmann [3] uses optical flow to warp one frame to the next, to move control points of strokes into successive frames; overpainting adds strokes where significant differences exist between frames. Hays [4] make strokes gradually reduce in opacity in successive frames to avoid flickering. The above methods assume that the computed optical flow is perfect, but existing methods all compute optical flow using some optimization technique. Motions of pixels are smoothed and the results contain errors [10]. Directly using unreliable optical flows to move control points leads to flickering, and furthermore,

errors are typically accumulated as time goes on. Bousseau [11] also uses optical flow in a watercolor generation pipeline, but relies on bi-directional temporal optical flow interpolation to reduce optical flow errors. However, strokes are still produced and adjusted frame by frame. Processing the video frame by frame, without global video analysis, typically leads to inconsistencies at image and object borders. Furthermore, the same object may disappear and reappear in the video a short time later, and thus may not exist at all successive pairs of frames during its lifetime. Our proposed method takes into account difficulties in optical flow computation, and evaluates the reliability of the motion estimate for each pixel. We use this data to form stable motion clusters of the objects in the video. These are then used to construct painting patches. This approach allows us maintain stroke coherence at edges of objects and the boundaries of each frame, as well as for objects which are present in discontiguous sequences of frames. Ref. [9] also introduces a video painting method. They mainly focus on style parameters and stroke orientation field design, and propagation between frames. However, as they note, sudden visibility changes between neighboring strokes is inevitable and leads to flickering, whereas we are careful to achieve coherence and stability between frames. Ref. [12] proposed a video re-rendering method by analyzing the motion structure of the running water in the video, synthesized coherent animations with Chinese painting style.

Other techniques for non-photorealistic rendering or video editing are based on pixel clustering in video. Stroke surfaces [13] and video tooning [14] use *color* information for video object segmentation, with the goal of producing cartoon-like animation with flat shading of each region, rather than stroke placement. In comparison, we use pixel *motion* for clustering. Unwrap mosaics [15] recovers a texture map for each object and the background from video. However, the video must first be segmented into independently moving objects, which is a difficult problem. Certain methods for motion segmentation or motion layer extraction in video are based on estimating dense optical flow [16–18]. The optical flow field is assumed to be piecewise smooth with discontinuities at object boundaries. As optical flow is inherently unreliable, especially for large disparity motions, other methods are based on tracking and region merging [19,20]. Our method also finds correspondences based on motion, but we aim to find as much as of each correspondence as possible, both between contiguous frames, and unlike these earlier methods, between discontiguous frames. These prior methods cannot easily be extended to handle discontiguous frames. Furthermore, for robustness, we leave pixels with complex motions, or which cannot be tracked, as unlabeled, rather than force them to belong to some region, adversely affecting its stability.

3 Overview

We aim to produce stable painterly animation from video. The novelty of our approach lies in how we achieve stability of painting in the output. We do this by identifying painting patches which exist across multiple frames, and rendering them just once. This improves stability of strokes with respect to previous methods in three cases:

- at image borders, when strokes gradually enter a sequence of frames;
- at object borders, when objects are gradually exposed;
- when objects are briefly occluded during a few frames.

Our pipeline is shown in Figure 1; our novel ideas are contained in the green boxes. Optical flow is computed for each pixel, with a reliability estimate. A random walk model [21] is used for video object segmentation. To achieve interframe consistency, allowing for regions which may disappear and reappear a few frames later, segmentation is performed using a multi-frame video window. A region's appearance in each frame is warped into a common frame of reference to give a moving painting patch, which is used as a basis for stroke based rendering. Warping is done in a way to minimize distortion. The rendered patches are then warped back to appropriate output frames, trimmed where necessary, and composited with other rendered patches in that frame.

4 Motion clustering

We wish to achieve stable rendering of objects between frames. This is difficult or impossible to do if

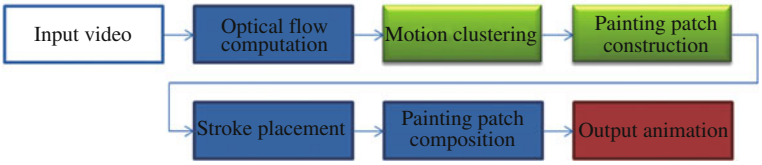

Figure 1 Pipeline: novel contributions of this paper are in the embossed boxes.

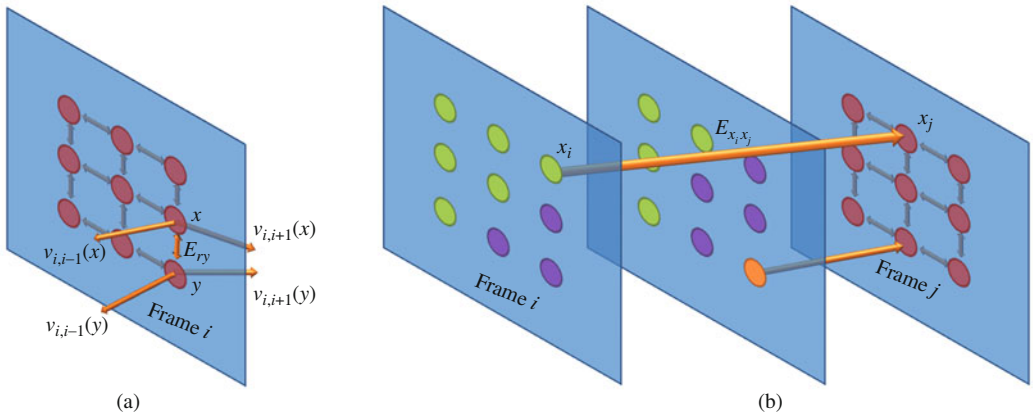

Figure 2 The random walk model for region clustering. (a) Intraframe edges; (b) interframe edges.

Figure 3 Region clustering. Note segmentation coherence at the tongue across discontiguous frames.

rendering strokes are determined simply by the motion information linking pairs of successive frames. Instead, we segment the video using motion information, and cluster pixels on the basis of similarity of motion during a sequence of frames centered at the current frame. Each cluster is then used to give a painting patch (see section 5).

To reduce the complexity of video clustering, we first segment the background into one or more background patches, as well as foreground objects using Video SnapCut [22]. Clustering is used to detect regions which persist between adjacent frames, or persist across multiple frames but are briefly occluded.

During clustering, we wish to find as many maximal clusters exhibiting coherent motion as possible. A random walk motion clustering algorithm is used to segment the foreground information frame by frame based on propagating information forward using a small window of frames up to the current frame (see Figure 2). The length of the window needed depends on the speed of motion of objects in the video—for faster moving objects a smaller window should be used, to avoid using stale information. A longer window may be used for slower objects, but will be more computationally expensive. In practice a window of 5 frames typically works well.

For each frame, we first compute the optical flow between the current frame and every other frame in the window using Zach's method [23] for propagating segmentation information. The reliability of the motion for each pixel is estimated based on the color difference between pixels linked by optical flow. Let the optical flow from pixel x in frame i to the corresponding pixel y in frame j be $v_{ij}(x)$. The reliability factor R_{xy} is then given by

$$R_{xy} = e^{-||c_x - c_y||}, \qquad y = x + v_{ij}(x),$$

where c_x and c_y are the RGB colors of pixels x and y respectively.

The random walk graph is created as follows. We set all pixels of the current frame as graph nodes. We join these to pixels in previous frames in the window linked by optical flow, and set these nodes as seed pixels if the optical flow is reliable enough. Each pixel is connected by an edge to its 4-connection neighbours within the current frame (intraframe edges), and to pixels in other frames corresponding to positions predicted by optical flow (interframe edges). The weight of *Intraframe edge* E_{xy} represents the motion similarity of the two pixels x and y (see Figure 2), defined by

$$w_{x,y} = e^{-(D(v_{i,i+1}(x),v_{i,i+1}(y))+D(v_{i,i-1}(x)-v_{i,i-1}(y)))/2},$$

where $D(v_1, v_2) = ||v_1 - v_2||^2$, and x and y are adjacent pixels within frame i connected by the edge. We consider both forward optical flow $v_{i,i+1}(x)$ to the next frame, and backward optical flow $v_{i,i-1}(x)$ to the previous frame. *Interframe edges* are used to join pixels in *different* frames which are in correspondence according to the optical flow. This weight is defined as

$$w_{x_i,x_j} = \lambda R_{x_i x_j},$$

where λ adjusts the relative importance of interframe and intraframe relationships, and is between 0 and 1. The main contribution of interframe edges is to transport the labels between frames, and very few pixels change their labels if the value of λ is altered. In practice we choose $\lambda = 1$ by default.

We perform the random walk segmentation by using Grady's method [21]. For the first frame, we find good corner points as initial seed pixels instead. Because of the presence of links between non-adjacent frames in the random walk model, the correspondences between discontigous frames (within the frame window) can be detected.

Eventually, after some frames, the resulting segmentation may no longer accurately represent the same information as in the initial frame (referred to *as keyframe* 0). User guidance is used to correct the segmentation in this subsequent frame (*keyframe* 1). A second pass is now used to update the segmentation for frames between these two keyframes, using a variant of the previous algorithm. Suppose we wish to determine the final segmentation for frame F_i. A blending coefficient α is determined as the fractional of the distance in frames of F_i between keyframes 0 and 1. Optical flow is used to propagate the segmentation of frame F_{i-1} forward to F_i, and backwards from frame F_{i+1} to F_i. Pixels in frame F_i which are given the same label by both forward and backwards propagation are used as seeds for the final segmentation. Other pixels are joined with edge weights now modified to be $\alpha w_{x_i,x_j}$ if in correspondence by optical flow with the next frame, or $(1-\alpha)w_{x_i,x_j}$ if in correspondence by optical flow with the previous frame. The pixels of frame F_i resegmented using a random walk. This provides a corrected segmentation for each frame between keyframes 0 and 1. Meanwhile, forward segmentation is reinitialised from keyframe 1, until it too becomes unreliable, where upon user assistance is called upon to segment another keyframe, *keyframe* 2. The video segments between keyframes 1 and 2 are then refined, and so on. This scheme has the advantage that user assistance is only provided as necessary, rather than e.g. at fixed frame intervals.

The properties of random walk segmentation ensure that each cluster detected corresponds to a spatially contiguous region with coherent optical flow, and the motions of these clusters represent the motion of patches of the original video. Figure 3 gives examples of the segmentation produced, which shows an example of correspondence across discontiguous frames: note the tongue of the goose which is present in the first and last frame (which are 5 frames apart), but not in the intermediate frame. This ensures stability of rendering in that area in the final output video.

5 Painting patches for stroke placement

The above method produces clusters of pixels with similar motions, which last for at least two frames, and correspond to moving objects. To produce a coherent and stable animation, these pixels should be rendered by the same strokes in each frame. To do so, we first register and warp the pixels in each frame belonging to the cluster into a single panoramic image [24] (see Figure 4).

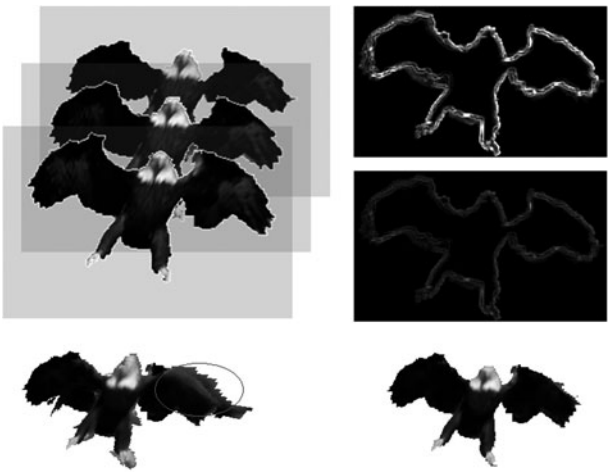

Figure 4 Painting patch construction. Top left: cluster in each frame. Top right: distortion error map before/after warping. Bottom left: Panoramic image without warping: note stretching of right wing. Bottom right: Panoramic image with warping.

As Hertzmann [3] notes, using simple image warping for NPR of video can lead to output which looks rather unnatural: distortion of brush strokes looks unnatural and does not have the desired painted appearance. Thus, we have devised a warping method which minimizes distortion when constructing the panoramic image, so that painting strokes are deformed as little as possible from frame to frame.

Image warping is a classical problem in computer graphics [25]. A standard approach is to estimate parameters for particular mapping functions such as affine transformations or projections. Here, pixels must be warped according to their optical flow, which in general may not admit a simple model. For example, if we were to approximate each region's motion by a simple model, the result could include gaps at the boundaries where clusters with different models meet. Instead, we use optical flow to warp each patch at the pixel level. This avoids the need to perform a further post-processing step for ghost reduction [26].

The warping method works as follows. To start, we find the *master* frame in which the region has maximal area, and initially leave the pixels in their original positions. This is used to define the global coordinate system for the warping process. We then iteratively work both backwards and forwards from that frame, adding the region as it exists in each successive (forward and backwards) frame. At each iterative step, first we use optical flow to find correspondences between the region's pixels in the current frame, and in the previously processed one. Some pixels have correspondences, but as the shape of the region changes, others do not. A warp is calculated to map the pixels with correspondences in this frame to the predecessor frame. This mapping is used to generate a homography, which is used to map the remaining pixels without correspondences to the predecessor frame; in this case their optical flows are ignored. Floating point coordinates are used to avoid error accumulation. We then optimize the warp between the current frame and its predecessor to minimize distortion, this frame is warped to the master frame by concatenating it with the warp for its predecessor. Finally, after all iterative steps have been performed a final global warp optimization is performed over all frames containing the region.

Warp optimization is performed by minimizing a total distortion energy E for each region defined in terms of springs connecting its pixels within a given frame and between frames (after mapping to global coordinates); a similar approach is used both for the inter-frame optimization, and the global optimization finally performed.

$$E = E_t + E_I = \sum_{f \in F, p \in R_f} E_{f,p} + \sum_{f \in F} \sum_{N_{xy}\ inf} E_{xy},$$

where F is the set of the frames, R_f is the region of the frame F, and N_{xy} denotes the pixels x and y are neighbours within frame F.

Linking pixels *between* frames is done by using optical flow. We construct a spring of zero length linking each pixel to its global position on the base frame. If such a spring is stretched to length L, its

energy becomes

$$E_{f,p} = kL^2,$$

where k is the coefficient of elasticity. Each pixel within a single frame is linked to its 8-connected neighbors. Springs of length $L_0 = 1$ join each pixel to its 4-connected neighbours, and springs of length $L_0 = \text{sqrt}(2)$ connect it to its other 8-connected neighbours, to retain a square grid. If the length of the spring changes to L, its energy becomes

$$E_{xy} = (L - L_0)^2.$$

The total energy E is given by $E = E_t + E_{xy}$. We wish to strongly attach the pixels to their global positions on the base patch if the optical flow is stable. Thus, we set $k = 10$ to give this term much greater importance.

Minimizing E can be solved via a linear system with high complexity due to the construction and solving of huge equations. Instead we iteratively find a solution. First, we build a grid on the base patch in global coordinates, in a bounding box surrounding the regions from all frames. Supposing all springs to be static, we compute the spring force at each pixel and use bilinear interpolation to find the force at each corner C of the grid from forces on pixels whose centers are adjacent to this corner. We now move all grid corners simultaneously by a distance equal to one-quarter of the force at each corner, and then update all pixel positions by bilinear interpolation of the corner coordinates. We iterate these two steps until the overall force on the grid no longer decreases (typically less than ten iterations are needed). This gives us the final, distortion minimizing global coordinates for each pixel in each of its frames. Figure 4 shows an example of painting patch construction. At the top left, we see three frame of an extracted eagle. Most errors occur at the borders of the eagle (see the error map at the top right). If this is not corrected during registration, pixels at the boundary will stretch or shrink due to error accumulation (see the bottom left). To avoid this we perform inter-frame warp optimization followed by global warp optimization, as discussed, giving a result with much less distortion (see the bottom right).

After registering each frame's warped image patch to global coordinates, we determine the color of each pixel by averaging the color of the pixel over every frame which contains it. Figure 4 shows the warped panoramic image from a cluster and its distortion error map. The resulting image is the *painting patch* for stroke placement.

We follow Hertzmann's framework [7] to render each painting patch; other rendering styles could also be used. We unwarp the painting patch back to each frame, again using the optical flow. Next, the unwarped rendered painting patch is clipped by the mask formed by the cluster boundary in each frame, and the patches for different regions are combined to build the output frames in painterly rendering style. Unwarping using the optical flow can ensure the smoothness of the scale of strokes of the boundaries of adjacent patches, where the optical flow smoothly change. However, this is not the final desired result, as, in each frame, some pixels are not clustered. These pixels typically have fast or complex motions, and are hard to track: the optical flow has large errors, and we have already decided we cannot reliably find a correspondence for them with pixels of other frames. Such unclustered pixels are painterly rendered, again using Hertzmann's method, simply on a per-frame basis, without further consideration of interframe relations. The final painterly animations are achieved by compositing the rendering results for the painting patches and the unclustered pixels. The unlabeled regions are untrackable pixels or pixels with complex motions, so flickers at these pixels on a per-frame basis is also unnoticeble.

6 Results and discussion

Figure 5 shows various painterly rendering results for videos. Coherence of stroke placement between frames can be seen. Comparing with [4] and [9], our method can produce brushes more stable along with the frames (see our results in the attached videos). The eagle swooping (Figure 5(a)) and turning head (Figure 5(b)) show the deformation of the objects in 3D space, in which the patch's area are changing along frames, as well as the scale of the strokes. Figure 5(d) shows coherent images borders due to the

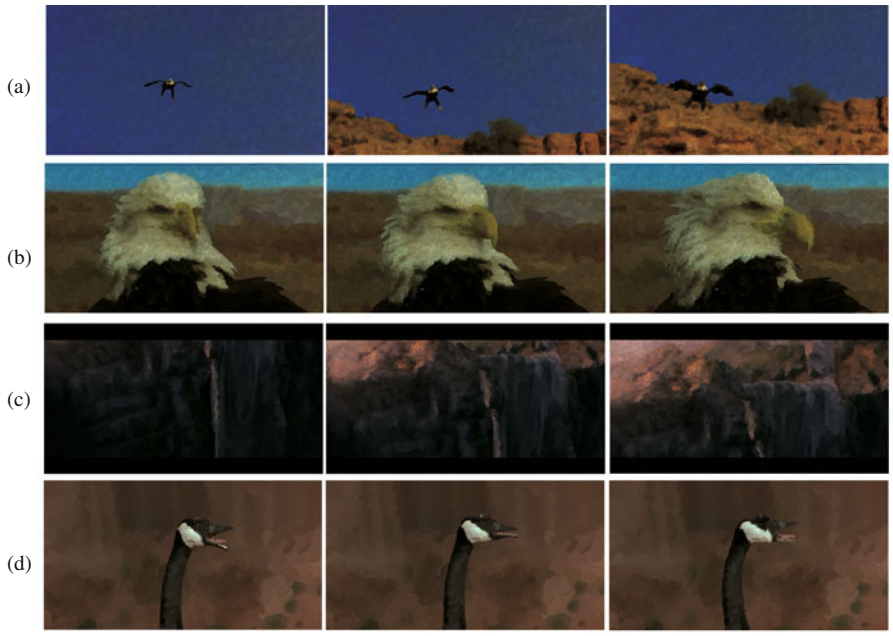

Figure 5 Painterly rendering output for several videos. (a) Eagle swooping with camera panning; (b) eagle turning head, with the deforming object in the video; (c) the great statue with camera panning; (d) goose singing, showing shows non-contiguous frame correspondence for the tongue of a goose.

Figure 6 Painterly rendering output using another style parameters.

Table 1 Brush parameters (in pixel units)[a)]

Region area	$r_{\max}$	$l_{\min}$	$l_{\max}$
$\leqslant 64$	2	1	16
65–512	4	2	32
$\geqslant 513$	8	4	128

a) $r_{\max}$: maximum brush radius, $l_{\min}$: minimum stroke length, $l_{\max}$: maximum stroke length.

panoramic painting patch construction in the large camera panning. And Figure 5(d) shows non-contiguous frame correspondence for the tongue of a goose.

We have presented a framework for painterly rendering of video, which is independent of rendering style. We now illustrate the results with some sample output which use curved brushes for rendering. The biggest brush radius, minimal stroke length, and maximal stroke length depend on the area of the region (over all its frames) is given Table 1. We use image gradients to guide the brush directions, so we can ensure coherence of stroke direction in neighboring patches. We observe that videos may be divided into more than one hundred clusters, and typically fewer than ten clusters comprise the main body of the video, while the other smaller clusters generally represent complex motions. Thus, regions with complex motions are rendered in more detail, in accordance with perceived importance of scene elements.

As noted, we can also apply any other painting style to the painting patches (and even use different styles or different style parameters for different patches). A further example is shown in Figure 6 which uses circle strokes and bigger color jitters. The result will be a temporally coherent painted video whatever the rendering style.

Our prototype system runs on an Intel Core 2 Duo 2.4 GHz based computer with 4 Gb of memory and an nVidia GTX 8600 video card. It takes about 1 s per frame for video object clustering, and less

than 10 s during the warping and composition step to process video of resolution 640×344 pixels. While it would not be too difficult to optimise our algorithm to make it faster, there is little motivation for doing so: user interaction is needed to check the segmentation of each frame, and insert a new keyframe if necessary. Checking each frame will take at least one second per frame, and while the user is adding a new keyframe, the system can catch up on warping and composition.

One issue that we have noticed is that if the rendered output has a 30 Hz frame result, the results can look too realistic, destroying the painterly effect, even though we achieve coherent results. We have found that 10 frames per second works better for our animations, which on the one hand preserves the underlying motion but on the other still gives the impression of a painting effect.

The main challenge in video rendering is to preserve temporal coherence between frames. Due to occlusion in video, some pixels may disappear, other pixels may appear, and some may do both with a few frames between. This typically causes significant flickering in prior approaches to painterly rendering of video. In our framework, we construct painting patches to handle this problem. However, some pixels remain unclustered after clustering pixels with similar motions, due to complex motions or smoothing of motion vectors near object boundaries. Currently, we simply render these pixels from the source frames on a per-frame basis, without further consideration of any interframe relations. Further work is needed to find better ways of handling these pixels. Scenes with motion blur and flashing can limit motion vector computation, reducing the amount of clustering, adversely affecting results.

7 Conclusions

This paper has presented a novel basis for re-rendering video in a painterly style. Instead of using motion vectors to place and adjust strokes on a frame by frame basis, as in prior methods, or segmenting the video volume according to color, we construct moving regions which capture coherent motion spanning multiple, possibly discontiguous, frames of the input video. Region clustering considers a window of frames about the current frame; it builds a random walk model which can detect the correspondences over several frames. Painting patches are produced by warping and registering a region's pixels in each frame into a common image. These are rendered in a stroke-based style (or any other style). The final output is achieved by compositing the rendered painting patches after warping and trimming them into each frame. Unlike the per-frame strategy, the proposed method produces stable strokes at frame boundaries exposed by camera panning, and at boundaries where objects are newly exposed. Temporal coherence of rendering is also achieved when objects are momentarily occluded. The results are the outputed videos with much less flicker than those produced by previous methods.

Further work could enhance our system. It would be desirable to produce fewer, larger regions: our approach leads to overly detailed segmentation and rendering where the motion is complex. A higher-level understanding of salient objects and complex motions would help to further achieve this. We have used a warping method to minimize the distortion error and make the results more painting-like. The construction of a hierarchical stroke placement could be an approach to solve the large scaling problem. And currently the painting patches are rendered independently according to the image gradient. To achieve better results, the correlation of stroke placement of the neighbouring painting patches, including the stroke size and direction, needs further considerations.

Acknowledgements

This work was supported by the National Basic Research Program of China (Grant No. 2011CB302205), and the National Natural Science Foundation of China (Grant Nos. 60970100, 61033012).

References

1 Hertzmann A. A survey of stroke-based rendering. IEEE Comput Graph Appl, 2003, 23: 70–81

2 Litwinowicz P C. Processing images and video for an impressionist effect. In: Proceedings of SIGGRAPH'97, Los Angeles, USA, 1997. 407–414
3 Hertzmann A, Perlin K. Painterly rendering for video and interaction. In: Proceedings of NPAR'00, Annecy, France, 2000. 7–12
4 Hays J, Essa I. Image and video based painterly animation. In: Proceedings of NPAR'04, Annecy, France, 2004. 113–120
5 Park Y, Yoon K. Painterly animation using motion maps. Graph Models, 2008, 70: 1–15
6 Haeberli P. Paint by numbers: abstract image representations. In: Proceedings of SIGGRAPH'90, New York, NY, USA, 1990. 207–214
7 Hertzmann A. Painterly rendering with curved brush strokes of multiple sizes. In: Proceedings of SIGGRAPH'98, Orlando, USA, 1998. 453–460
8 Zeng K, Zhao M T, Xiong C M, et al. From image parsing to painterly rendering. ACM Trans Graph, 2009, 29: 2:1–2:11
9 Kagaya M, Brendel W, Deng Q Q, et al. Video painting with space-time-varying style parameters. IEEE Trans Visual Comput Graph, 2011, 17: 74–87
10 Baker S, Roth S, Scharstein D, et al. A database and evaluation methodology for optical flow. In: Proceedings of ICCV'07, Rio de Janeiro, Brazil, 2007. 1–31
11 Bousseau A, Neyret F, Thollot J, et al. Video watercolorization using bidirectional texture advection. ACM Trans Graph, 2007, 26: 104
12 Zhang S H, Chen T, Zhang Y F, et al. Video-based running water animation in Chinese painting style. Sci China Ser F-Inf Sci, 2009, 52: 162–171
13 Collomosse J P, Rowntree D, Hall P M. Stroke surfaces: Temporally coherent artistic animations from video. IEEE Trans Vis Comput Graph, 2005, 11: 540–549
14 Wang J, Xu Y Q, Shum H Y, et al. Video tooning. ACM Trans Graph, 2004, 23: 574–583
15 Rav-Acha A, Kohli P, Rother C, et al. Unwrap mosaics: a new representation for video editing. ACM Trans Graph, 2008, 27: 1–11
16 Ayer S, Sawhney H S. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In: Proceedings of ICCV '95, Washington, DC, USA, 1995. 777–785
17 Black M J, Jepson A D. Estimating optical flow in segmented images using variable-order parametric models with local deformations. IEEE Trans Patt Anal Mach Intell, 1996, 18: 972–986
18 Odobez J M, Bouthemy P. Direct incremental model-based image motion segmentation for video analysis. Signal Process, 1998, 66: 143–155
19 Xiao J J, Shah M. Motion layer extraction in the presence of occlusion using graph cuts. IEEE Trans Patt Anal Mach Intell, 2005, 27: 1644–1659
20 Wills J, Agarwal S, Belongie S. A feature-based approach for dense segmentation and estimation of large disparity motion. Int J Comput Vision, 2006, 68: 125–143
21 Grady L. Random walks for image segmentation. IEEE Trans Patt Anal Mach Intell, 2006, 28: 1768–1783
22 Bai X, Wang J, Simons D, et al. Video snapcut: robust video object cutout using localized classifiers. In: SIGGRAPH'09: ACM SIGGRAPH 2009, New York, NY, USA, 2009. 1–11
23 Zach C, Pock T, Bischof H. A duality based approach for realtime tv-l1 optical flow. In: Pattern Recognition (Proc DAGM), Heidelberg, Germany, 2007. 214–223
24 Wolberg G. Digital Image Warping. New York: John Wiley & Sons, 1990
25 Zitova B, Flusser J. Image registration methods: a survey. Image Vision Comput, 2003, 21: 977–1000
26 Rebiere N, Auclair-Fortier M F, Deschenes F. Image mosaicing using local optical flow registration. In: Proceedings of ICPR'08, Tampa, FL, USA, 2008. 1–5