Coding of Stroke-Based Animations

Levente Kovács
University of Veszprém
Dept. of Img. Proc.&Neurocomp.
Veszprém, Egyetem u. 10
H-8200, Hungary

levente.kovacs@freemail.hu

Tamás Szirányi
Hungarian Academy of Sciences, Analogical
Comp. Lab. Comp. & Aut. Res. Inst.
Budapest, POBox 63
H-1518, Hungary

sziranyi@sztaki.hu

ABSTRACT

We present a way of rendering painting-like images and sequences by our Stochastic Painting-based SBR technique. We avoid disturbing artifacts by coding the images in a painting-like way. If we code the resulting stroke sequence, than the level of error comes not from the artifacts but from the painting process. For this reason, if we generate high quality paintings, we can transfer the image without disturbing coding errors. The painting method incorporates some novel properties like dynamic Monte Carlo Markov Chain optimization, multiscale edge gradient following or grayscale stroke templates. The painting technique inherits the properties of the Paintbrush Transformation, like well-defined contours, acceptable distortion and a painting-like view with no fine details below a limit. Our goal is to produce a painting -like output, which contains the stroke-series and the motion data in a losslessly compressed form. This way the painted video contains no compression artifacts (while the painting-like impression remains). The compression scheme of the stroke-series with motion data could also be suitable for compressing painting-like image sequences produced with other painting techniques.

Keywords
stroke based rendering, painting, Stochastic Paintbrush Transformation, animation coding

1 INTRODUCTION

Our Stochastic Paintbrush Transformation (SPT) [Szi00] is a combination of two approaches (greedy and optimized): while being stochastic in stroke generation (position and orientation), it has some built-in optimizations for the stroke acceptance (Monte Carlo Markov optimization) [Szi01]. We extended the original SPT technique onto image sequences with the goal of generating animation/cartoon-like videos. The method presented herein contains motion estimation and cut detection, has the ability to handle region of interest (ROI) areas, allows the usage of user-drawn stroke-templates, stores the processed frames with describing the stroke-series and motion data and losslessly compressing the output stroke-stream. If we generate high-quality painting/stroke-sequences,

we can transfer the images without disturbing artifacts possibly induced by standard block-coders.

2 PAINTBRUSH TRANSFORMATION

The Stochastic Paintbrush Transformation method proposed by [Szi99] uses multiple sizes of rectangular-shaped strokes for describing the input image. We had the goal of achieving an automated method which simulates a real painting process, to obtain a picture similar to a real painting, where the purpose of the painter is to portray something which looks like to be real scenery. There were/are several painting methods which do not produce "better" output (which is a hard to define term in itself in this case, knowing the nature of the painting techniques in general) than others, but with a different technique.

2.1 NPR/SBR – Related Work
Haeberli [Hae90] has introduced painting with an ordered collection of strokes, controlling shape, size, color, orientation. Painting is mainly done by following the cursor and randomly sampling the color data. There's also the possibility to use images and gradient data to control the brush direction. In [Lit97] strokes with given center, length, radius and orientation with color bilinearly interpolated are

generated, adding random variation and perturbation, gradient-based orientation. The author also uses stroke clipping for retaining edges, and anti-aliasing for the same reason. [Mei96] uses particle sets to describe and follow surfaces, tessellating surfaces into triangle sets and geometric and lighting properties of surfaces to control the appearance of strokes. Objects are rendered as different layers. The notion of "shower door" effect was introduced, which is produced when the position and color data of the strokes remain constant. In In [Her98] painting is done with a series of spline strokes aligned to a grid, with a series of layers, in a coarse-to-fine way, presenting a framework for describing a wide range of styles. Brush strokes are anti-aliased cubic B-splines, with constant color and thickness. [Kal02] lets the designer directly annotate a 3D model with strokes, imparting a personal aesthetic to the non-photorealistic rendering of the object. The artist chooses a brush-style then draws strokes over the model from one or more viewpoints. In [Kap00] the author presents an algorithm for rendering subdivision surface models of complex scenes using particle systems and graftals introducing geograftals, extending the work of [Mei96,Kow99].

The present work, which is an extension of [Kov02], is a companion of [Her00] and [Lit97] on the path of painterly animation creation, storing and representation of videos (real life and cartoon). We also introduce some novel features in our painting technique, which make it differ from other similar methods in the field, like the following: unrestricted stroke templates, no stroke grid, stroke positions are stochastic, region of interest areas, optionally following only the edges on the frames after a specified painting step, multiscale gradient-based edge orientation process, storing the output as a compressed stream of stroke-series.

2.2   Stochastic Painting (SPT)

An extended version of the original SPT method is used. An addition is the possibility to use any 60x60 pixel grayscale image as a stroke template for the painting process (e.g. Figure 1). Another is the support for region of interest areas, where the user can define a specific region (or a sequence of ROIs in the case of videos) which will be painted with the finest scale of strokes for better approximation of the model image. A consequence is that a single image can be painted by using multiple strokes (for refinement, or just for another effect).

Any pixel position can be a possible stroke target. This way stroke placement, identification and handling remain a simple and fast task. Good results in segmentation and image classification have also been achieved using this technique [Szi02].

During the coarse-to-fine painting process a stroke with given size (iterated through stroke-sets), orientation (eight possible, rotated by $pi/8$) and color (interpolated from the model image) is placed on the canvas, if its placement makes the painting converge to the model. The resulting image is described by the stroke-series used (e.g. Figure 2).
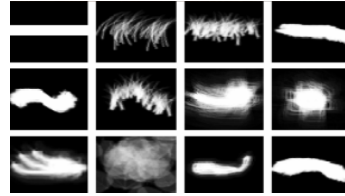

Figure 1. Samples of user-defined stroke-shapes.
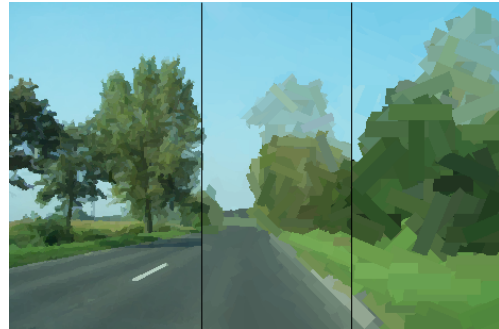

Figure 2. Sample painted image (final, fifth and first step of a ten-step painting process).

Besides imitating motion direction when setting a stroke-direction, sometimes following edge directions can produce the same painted quality but faster convergence (because it produces smoother edges on the painted image). We implemented a version of Lindeberg's [Lin99] multiscale edge detection which finds main edges searching for edge positions which produce an accentuate curve in scale-space (e.g. Figure 5). From the edge-map we calculate a direction for each image pixel as follows: from a pixel location we search for the nearest edge, calculate its gradient, and let the stroke take this orientation. This way compressed outputs get reduced by an average of 20% and transformation times also have a slight decrease.

3   TRANSFORMING VIDEO

Briefly, our video painting technique works as follows: take a keyframe and paint it fully. Then obtain optical flow do partial repainting on the areas where there was motion, using the optical flow data or edge gradients to specify the stroke orientations. When repainting the motion areas and do the post-processing which eliminates over-painted strokes the stroke density becomes generally uniform. Then we compress both the motion and the stroke data, and go to the next frame. We also take into account the ROIs, the option to follow the edges on the frames

(for better contours) and the position of the cuts. ROIs can be defined for a single frame, for a series of frames or for the whole sequence.

3.1 Transformation Details

We tried multiple motion detection algorithms to calculate optical flow (block matching, gradient-based [Bar97,Hee98,Sim93] and hierarchical gradient-based [Ber92] methods) to obtain the areas where motion occurs.

We also use our simple and fast cut-detection and treat the first frame after the cut as key frames. It works as follows: calculate the summarized pixel differences between two following frame intensities, for the whole input video or on a frame-window. At the same time we calculate the edge-maps of the frames and the differences between the edge-maps of consecutive frames. We say we have a transition when the product of pixel- and edge-differences reaches a threshold (which is the average of the difference-products along the frames multiplied by a constant). This approach results in about 3-5% of cut false-positives and has moderate ability for fade-in/out and dissolve detection. We also tested methods like edge-based contrast feature tracking and YUV-histogram-based methods [Lie99] for transition detection, and the RGB-histogram-based methods in [Han03].

The only drawback of this detection shows at longer fades/dissolves. The errors induced by possible false transition detection only get corrected at the first upcoming keyframe. ROI data (selected rectangular frame regions) are stored as distinct files and get associated with the video at the beginning of the transformation process.

The main steps of the video painting algorithm are as follows:

1. *Setting of parameters*
2. *Cut detection.*
3. *Edge detection and SPT of frame F(i) to F'(i), compressing and storing F'(i).*
4. *Edge detection and optical flow calculation.*
5. *Transformation of motion areas.*
6. *Writing the partially transformed frame data onto F'(i), generating F'(i+1) and store.*
7. *If the next frame will be a key-frame then increase i and jump to step 3, otherwise to step 4.*

Frame data gets Huffman-encoded [Wel99] before storing into the output file and the motion data is run-length encoded [Wel99]. Using these data the painted video can be reconstructed stroke-by-stroke, frame-by-frame. The average compression ratios of the output range from 5 to 20%. We have achieved considerable results when transforming cartoons in noise reduction and better representation (avoiding the typical artifacts of DCT), with good compression ratios.

We cannot provide any simple bit rate controlling possibility at this point, for many reasons. Ways of controlling the process are through the number of stroke-sets used for painting (usually 3 to 10), the number of the placed strokes (upper limit), the target PSNR, keyframe density (keyframes are larger than others) or the relative error threshold (controls switching to a smaller stroke-set). When using more of the smaller strokes painted quality can be raised but output size raises also because the amount of data needed to store a stroke is the same for big or small strokes, and we need more strokes to cover the canvas if the stroke's size is small. When changing the error threshold we indirectly tell the algorithm to which extent it should allow a stroke-set to manipulate the painting. With a greater threshold the set-changes will occur earlier and the strokes remain more visible. With a lower threshold the quality will be higher but it will contain more strokes. A placed stroke can be described with its parameters (identification - stroke's set and type, color, position). This representation (6 bytes for a stroke) stays the same for every type of strokes.

We have to state here, that decoding (playing) the painted, stroke-series-based output is not a fast process yet (there's no real-time 24 fps decoding). This is mainly because we have not concentrated on building an optimized fast decoder yet.

4 RESULTS

Figure 3 shows how transformation times behave when keyframe density is changed. In Figure 4 we provide some compression data showing that if we compress the painted frame treated as stroke-series in a lossless way, then we achieve 3-4 times better compression ratios than usual lossless image coders (JPEG-LS, HuffYUV, etc.) which code the painted frame as a bitmap. If we compress these bitmaps in lossy mode to achieve the same ratio as our lossless coding, then we get high compression errors, which – measured in JND scales - show that in this case around 95% of the viewers could easily notice the degradations induced by the lossy coding. One of the interesting properties of our painting technique is that it smoothes surfaces but preserves contours and main edges producing a visually pleasant output.

5 CONCLUSION

The method we are developing is a way of generating animations from real-life scenes, and a method for representing and storing cartoons in a way that still has a pleasant visual quality. We provide the option of compression in a way that reflects the artistical

methods of creating such images. The coding scheme proposed can be a good way of storing painted images in losslessly, more effectively as usual raster coders would do. For more samples: www.knt.vein .hu/~kovacs/mpv/.
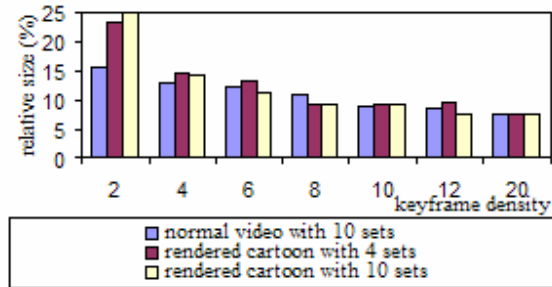
6   Acknowledgements

Figure 3. Variation of output size relative to uncompressed input with changing keyframe density.
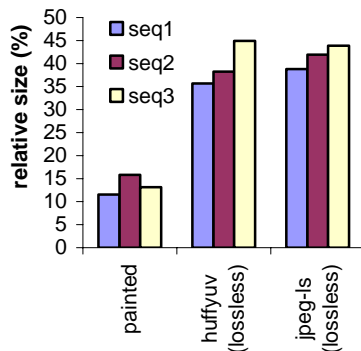

Figure 4. Lossless coding ratios of output stroke-series relative to lossless coded bitmaps of painted images.

7   REFERENCES

[Bar97] Barron. J., Fleet, D.J., Beauchemin, S.S.: Performance of Optical Flow Techniques, IJCV 12(1):43-77, 1997.

[Ber92] Bergen, J.R., Anandan, P., Hanna, K.J., Hingorani, R.: Hierarchical Model-Based Motion Estimation, Proc. Of ECCV, Santa Margherita, Italy, pp.237-252, Springer-Verlag, 1992.

[Hae90] Haeberli, P.: Paint by Numbers: Abstract Image Representation, Siggraph, Computer Graphics vol. 24, number 4, 1990

[Han03] Hanis, A., Szirányi, T.: Measuring The Motion Similarity in Video Indexing, 4th Conference on Video/Image Processing and Multimedia Communications, Zagreb, Croatia, 2003

[Hee98] Heeger, D.J.: Notes on Motion Estimation, Psych. 267/CS3848D/EE 365, 1998

[Her98] Hertzmann, A.: Painterly Rendering With Curved Brush Strokes of Multiple Sizes, Siggraph, pp. 453-460, 1998

[Her00] Hertzmann, A., Perlin, K.: Painterly Rendering for Video and Interaction, NPAR 2000

[Kal02] Kalnins, R.D., Markosian, L., Meier, B.J., Kowalski, M.A., Lee, J.C., Davidson, P.L., Webb, M., Hughes, J.F., Finkelstein, A.: WYSIWYG NPR: Drawing Strokes Directly on 3D Models, Siggraph 2002

[Kap00] Kaplan, M., Gooch, B., Cohen, F: Interactive Artistic Rendering, NPAR 2000, pp.67-74, 2000

[Kov02] Kovacs, L., Sziranyi, T.: Creating Animations Combining Stochastic Paintbrush Transformation and Motion Detection, 16th ICPR, Quebec, Caada, vol. 2, pp. 1090-1093, 2002

[Kow99] Kowalski, M.A., Markosian, L., Northrup, J.D., Bourdeu, L., Holden, .S., Hughes, J.: Art-Based Rednering of Fur, Grass and Trees, Siggraph, pp.433-438, 1999

[Lie99] Lienhart, R.: Comparison of Automatic Shot Boundary detection Algorithms, Proc. Of Image and Video Processing VII, 1999

[Lin99] Lindeberg,T.: Automatic scale selection as a pre-processing stage for interpreting the visual world, Proc. FSPIPA, Budapest, Hungary, September 6-7, 1999. Schriftenreihen der Österreichischen Computer Gesellschaft, volume 130, pp 9--23

[Lit97] Litwinowitcz, P.: Processing Images and Video for an Impressionist Effect, Siggraph 1997, pp.407-414

[Mei96] Meier, B.J.: Painterly Rendering for Animation, Siggraph 1996, pp.477-484

[Sim93] Simoncelli, E.P.: Distributed Representation and Analysis of Visual Motion, Vision and Modeling Group Techincal Report #209, MIT Media Laboratory, 1993

[Szi99] Sziranyi, T., Toth, Z., Kopilovic, I.: Paintbrush Image Transformation, Fundamental Structural Properties in Image ans Pattern Analysis, FSPIPA99, IAPR, pp.157-168, 1999

[Szi00] Sziranyi, T., Toth, Z.: Random Paintbrush Transformation, 15th ICPR, Barcelona, IAPR&IEEE, V.3, pp.155-158, 2000

[Szi01] Sziranyi, T., Toth, Z.: Optimization of Paintbrush Rendering of Images by Dynamic MCMC Methods, Lecture Notes on Computer Science, Springer Verlag, Vol.LNCS 2134, pp.201-215, 2001

[Szi02] Sziranyi,T., Kato,Z., Ji,X., Toth,Z., Czuni,L.: Content-Based Image Retrieval Using Stochastic Paintbrush Transformation, ICIP2002, Multimedia Retrieval and Applications, IEEE, Rochester, 2002

[Wel99] Wells, R.B.: Applied Coding and Information Theory for Engineers, Prentice-Hall, 1999
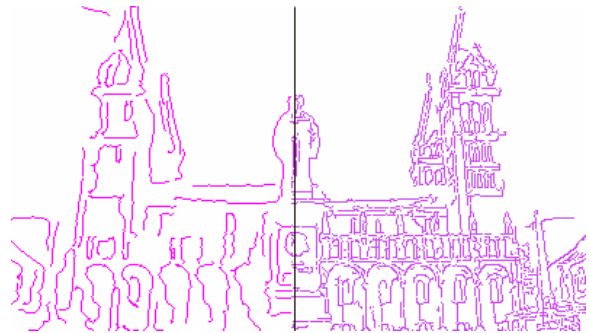
Figure 5. Canny-edges (right, with thresholds at 32 and 200) and Multiscale edges used when painting (left) – mirrored image parts.