

Painterly Rendering Controlled by Multiscale Image Features

Levente Kovács,[†]
Tamás Szirányi^{‡§}

University of Veszprém, MTA SzTAKI

Abstract

What we present in this paper is a 2D non-photorealistic, stroke-based image rendering technique, which is a fully automatic template-based painting method controlled by the structure of the model image. Namely it is based on dominant, weighted image edges and ridges extracted by a multiscale edge/ridge detection method. Stroke sizes (scales) and painting directions are both controlled by edge and ridge weights and orientations. This way the painting process becomes more "natural" in the sense that all of its parameters are guided by image features. We also present different painting variations based on this technique.

Keywords: stroke based rendering, automatic painting, stochastic painting, image decomposition

1 Introduction

Non-photorealistic stroke-based rendering (NPR/SBR) methods generally have the goal of simulating the painting process of a real painter to obtain a picture which would resemble paintings in overall (at least some impression of the original model image). There are also painterly transformations which produce sketches or some other form of simplistic representation of the model e.g. for illustration purposes.

Abstract representations of usual imagery also serve a special purpose of image/data visualization, when rough, sketchy or specially focused representations are desired, and even provide some artistic imprint (being able to simulate different painting styles). When an image gets represented by a sequence of over- or non-overlapping brush-strokes, other aspects of possible utilization come to attention, e.g. alternative ways of compression, storing and retrieval. For instance a special sequence describing an image area can be used for indexing [5]. At the same time compactly designed stroke parameter series can be effectively compressed [6].

The first major work on stroke-based rendering was the work of Haeberli [2] who introduced painting with an ordered collection of strokes described by shape, size, color and orientation. Painting was done by cursor following and randomly sampling color data. Arbitrary images and gradient data could also be used for painting control. Litwinowitz [9] used strokes with a given center, length, radius and orientation, with bilinearly interpolated color data, adding random perturbation and gradient-based orientation. Meier [10] and Kalnins et al. [4] gave methods for painting 3D surfaces. The former used particle sets and the stroke parameters were controlled by geometric and lighting properties of the surfaces. The latter used silhouette lines, decal strokes and hatching to draw strokes on an object from multiple viewpoints.

Hertzmann [3] used painting with a series of b-spline strokes aligned to a grid mapped on the image, on a series of layers in a coarse-to-fine way, with multiple styles. Brush strokes were modelled as antialiased cubic b-splines with constant color and thickness. Santella et al. [12] presented a promising new extension of traditional automatic painterly rendering methods, namely guiding the painting process by eye movement, focusing on areas where the user is looking. This way they generate an abstract representation of the input image which has those areas emphasized which the user thinks are important.

In [11] Northrup et al. present a way of rendering silhouette outlines of 3D meshes by edge extraction, drawing long connected paths corresponding to visible parts of the silhouettes. In Toth et al. [13, 14] a stochastic painterly rendering method is introduced which uses stochastic placement of rectangular strokes with different sizes on multiple layers. Strokes are described by position, color and orientation which are also random and the process is controlled by constant quality checking and Monte Carlo Markov Chain optimization at the stroke acceptance step. In [7] a painting method is presented where user defined grayscale templates can be used in a stochastic stroke placement painting process, where the orientation of the strokes is controlled by nearby edge orientations.

In the following section a new 2D painting method is introduced. It is based on the approach that as many painting parameters and options should be controlled by the model image data, as possible. We try to eliminate as much randomness and guessing from the painting process as we can

^{*}Dept. of Image Processing and Neurocomputing, University of Veszprém, Hungary

[†]kla@vision.vein.hu

[‡]Hungarian Academy of Sciences, Analogical Comp. Lab., Comp. and Automation Research Institute, Budapest, Hungary

[§]sziranyi@sztaki.hu

to reduce coding times on one hand and to introduce as much "naturalness" to the painting as possible. We do this by extracting weighted edges and ridges from the model image using a variant of Lindeberg's [8] multiscale approach and use the edge/ridge data to control stroke scales and orientation. In our most recent painting technique stroke positions are sequential in the sense that possible painting positions are checked sequentially.

2 Painting

In [6] we used stochastic stroke position and orientation generation, and sequential scale cycling, painting in a multilayer coarse-to-fine way. The images generated this way were usually visually pleasing and could be compressed with high ratios. One of the main drawbacks was coding time. This was due to the highly stochastic nature of the algorithm. Figure 1 shows a sample painted image using this technique.



Figure 1: Example of image painted using simple 10 layer stochastic painting.

We started to search for new ways of painting which would be faster and more closely related to image contents and less stochastic in nature than our previous method. In [7] we introduced an extension which gave some improvement in this direction, which was multiscale edge extraction, and using the extracted edge map to determine stroke orientation. But automatic stroke scale (size) selection still lacked and painting still proceeded through multiple stroke layers.

The method presented in the next sections in detail uses one layer of painting, sequential stroke position selection, edge/ridge orientation and weight to determine stroke scales and orientations automatically. This way the whole painting process is controlled by the extracted main image structures. In Section 2.1 the multiscale edge and ridge extraction method is presented while in Section 2.2 we present painting algorithms and styles using this technique.

2.1 Multiscale Structure Extraction

In [13] strokes are placed at random positions with random orientation. This can lead to situations when the placement of a stroke makes some improvement to the image in general, but it causes such local errors, which can only be corrected with multiple smaller strokes at a later step. This led to the conclusion that the random orientation is not always an acceptable solution.

We tried to place strokes with the orientation of nearby edges. The idea behind this approach was that if the orientation of larger strokes follows the orientation of edges on the image, than smaller strokes need only to cover the remaining areas well. This approach would also eliminate such induced errors when a stroke gets placed normal to an image edge.

We needed a way to get the main edges of the image, and the least of the small/short edges which in our case are considered noise. We do not want to get into unnecessary detail on edge detection methods here, there are many papers and works that deal with this question. Our first choice was the Canny's edge detector [1], which can produce quite good results for most of the cases, but our final choice became an implementation of Lindeberg's [8] multiscale edge detection method. This method tries to find main edges by searching for edge positions which produce an accentuate curve in scale-space. Besides this, it gives much more than Canny's edge detector can, which is weighting the resulting edges and ridges where the weights reflect how relevant the respective curve is. A comparison just for one sample image to show how an edge map looks like which is used for painting is shown on Figure 3.

Given an image

$$f : R^2 \rightarrow R$$

its scale-space representation

$$L : R^2 \times R_{+-} \rightarrow R$$

defined by

$$L(\cdot; t) = g(\cdot; t) * f$$

where

$$g : R^2 \times R_{+-} \rightarrow R$$

denotes the Gaussian kernel, which will be used to generate the scales used in the detection process:

$$g(x; t) = \frac{1}{2\pi t} e^{-\frac{(x^2+y^2)}{2t}}$$

where "t" is the scale parameter. At each scale level, edges are defined from points at which the gradient magnitude assumes a local maximum in the gradient direction. We use the edge definition when at a given point, if it is an edge-point then the second order derivate is zero and the third order derivative is negative (for each respective direction):

$$L_{pp} = 0, L_{ppp} < 0$$

For ridge definition we use the following geometrical definition:

$$L_p = 0, L_{pp} < 0, |L_{pp}| \geq |L_{qq}| \text{ or}$$

$$L_q = 0, L_{qq} < 0, |L_{qq}| \geq |L_{pp}|$$

which is the formulation for 2 directions (horizontal and vertical). We extended this formulation with 2 more directions, the two diagonals. Using 10 scales generated with Gaussian convolution with different scale parameters we calculate scale-space derivatives. Then we search for all curves connected on the same (maximal) scale and calculate the edge strength formulated as:

$$E = t^\gamma (L_p^2 + L_q^2)$$

and the ridge strength as:

$$R = t^{2\gamma} ((L_{pp} - L_{qq})^2 + 4L_{pq}^2)$$

Then significancy measures, which we call edge/ridge weights are calculated, which is a line integral of weighted derivatives along the actual curve. We store all connected curves on the image and sort them in descending order of their summed weights. At last we display the first N curves on an edge/ridge map. For an example of extracted edges and ridges see Figure 2.

When we have this edge map, we calculate a direction for each image pixel as follows: from a pixel location we search for the nearest edge, sample some neighboring points from it, calculate its gradient (from the directional tangent obtained from the sampled point locations), and let the stroke take this orientation. When producing images this way, the length of compressed stroke-series is reduced by an average of 15-20%, and transformation times also decrease. For an example see Figure 4.

The ridge map is used for obtaining stroke size (scale) automatically. This is done in the following way. We find the maximal and minimal ridge weight on the ridge map, and map the available stroke sizes over the weight interval in such a way that lowest weight corresponds to smallest stroke scale and highest weight to highest stroke scale. During the painting process, when a stroke needs to be

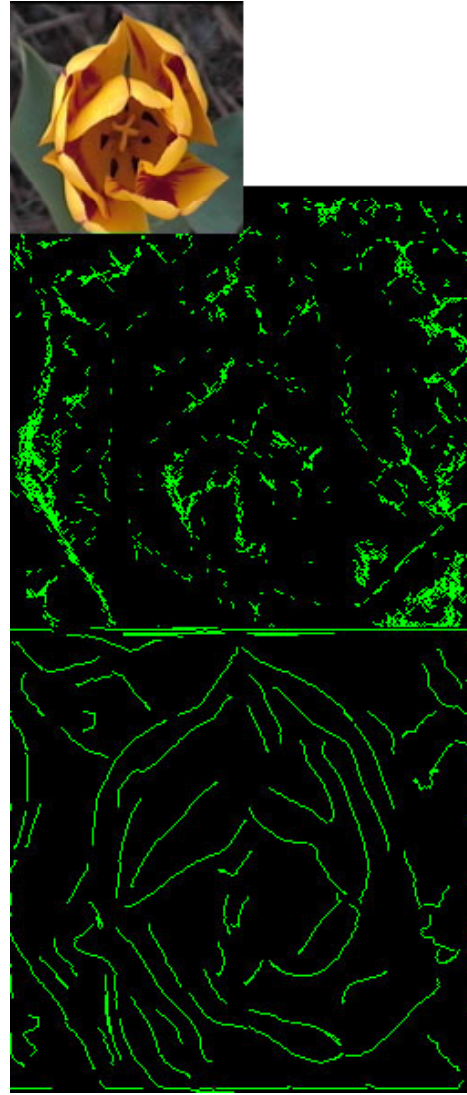


Figure 2: Example of edge and ridge maps extracted from a same image (top to bottom: image, ridges, edges).

placed its scale is determined by searching for the nearest 8 ridges and calculating the stroke scale by weighting the scales associated to the found ridge weights with the distances they are from the current stroke position.

Painting methods using extracted weighted edge and ridge maps are presented in the following section.

2.2 Image Generation

The painterly rendering process starts with edge and ridge extraction as shown above. In this section we present three methods of painting, each using the information obtained this way to some extent.

Before the painting process starts, color sampling for the different possible stroke sizes and orientations is performed as a preprocessing step. This is done by convolving the model image with all possible stroke patterns with all possible orientations, for obtaining a blurred image se-



Figure 3: Example of Canny (3x3, hysteresis parameters 30 and 200, up) and multiscale-detected edges which is used for painting (down).

ries from which color sampling will be performed. Convolution is done in Fourier space (by multiplying the FFT of the templates with the FFT of the model image) for less computation time, exploiting a very important property of the Fourier transform, that convolution is equivalent to multiplication in Fourier or frequency space:

$$f(x) * g(x) = F(w)G(w)$$

Later during the painting process, color sampling will be performed using these pre-generated images as follows: If we wish to place a stroke on a given position, the blurred image version is taken which belongs to the stroke template, size and orientation in question, colors are counted from under the possible stroke position, and the most frequently occurring color is given to the stroke.

The main steps of the first painting approach are shown on the block diagram on Figure 5. In this process only the edge information is used, for stroke orientation selection. The steps are as follows:

- select a stroke set,
- select random stroke positions,
- get stroke orientation from the obtained weighted edge map,
- obtain stroke color,
- if the placement of the stroke does not induce errors in the image (the placement of the stroke lessens the image error in PSNR between the painting and the model over a threshold) then place the stroke,
- repeat stroke placement until no improvement over a threshold is obtained, then switch to a smaller stroke set

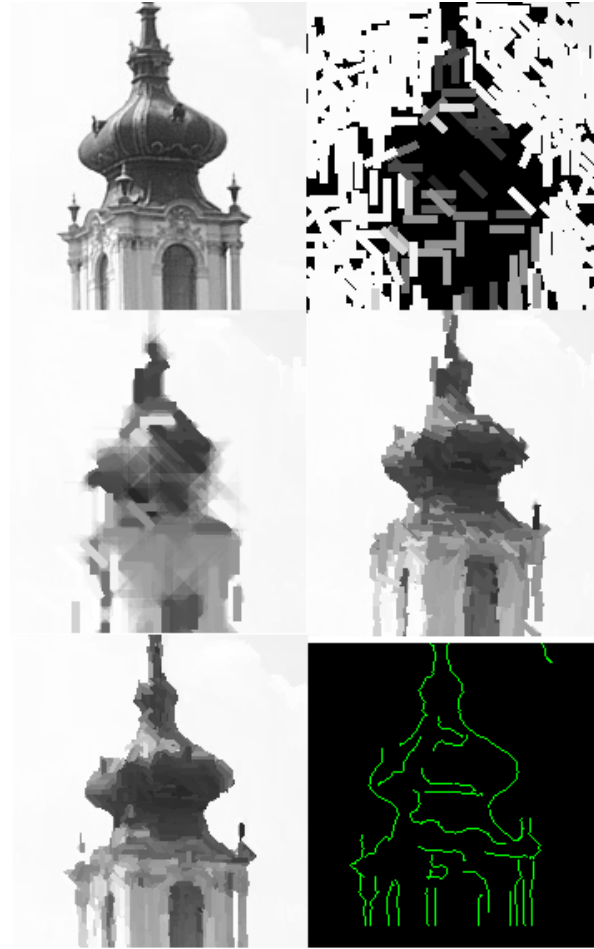


Figure 4: Example of image painted following multiscale edges (top-left: original, bottom: painted, and multiscale edges used when painting).

and repeat the whole process until there are no more sets.

Stroke orientations are calculated from the extracted edge maps as described in the previous section (by weighting the calculated orientations of nearby edges).

An image generated using this method is shown on Figure 6. When using this type of painting, the speedup related to fully stochastic 10 layer painting (which means 10 possible stroke sizes/scales) is shown on Figure 7.

The second method is stochastic in nature by the means that possible stroke positions are still randomly chosen, but both stroke scale (size) and stroke orientation are obtained from the extracted edge and ridge maps. The diagram of the algorithm is very similar to the algorithm on Figure 5, the only difference being in the "Generate strokes" step. The main steps are:

- select random stroke position,
- get stroke scale from the weighted ridge map,
- get stroke orientation from the weighted ridge map,
- obtain stroke color,

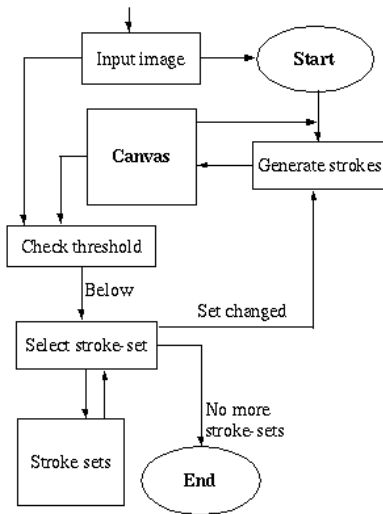


Figure 5: Main steps of the 1st painting approach.



Figure 6: Example of image painted following edge orientations (random stroke positions, sequential 10 layer stroke scales).

- place the stroke and repeat the process until there is no improvement in error between the model and the painting over a threshold,
- do post-processing: correct possible errors by a final overpainting with the finest stroke size (size will be constant, orientation taken from the edge map).

Stroke scales are calculated from the extracted ridge maps as described in the previous section (by proportionally mapping the available stroke sizes over the obtained ridge weight minimum-maximum values).

An image generated using this way of painting is shown on Figure 8.

This method is a bridge between stochastic painting and the following method which has no stochastic steps. While it still searches for stroke positions in a random way, placed strokes now follow the main image structure,

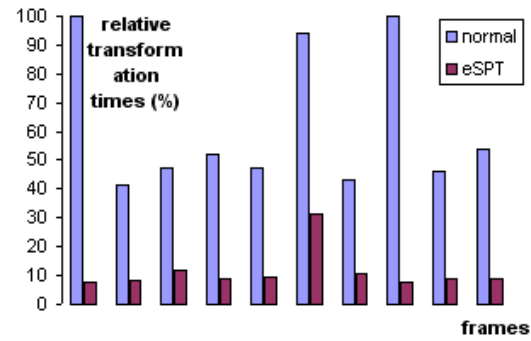


Figure 7: "normal" is the fully stochastic painting method with 10 layers, "eSPT" is a new method with random stroke positions but following edge orientations when placing the strokes. It shows (for consecutive frames of the Mother and Son qcif video) that for achieving similar PSNR the new method needs much less time.

thus processing time decreases further more, and the number of strokes needed to cover the canvas also drops.

The third method we present is totally based and controlled by the edge and ridge maps obtained in the previous section. Its block diagram is shown on Figure 9. The main steps are as follows:

- sequentially search for the next ridge point on the ridge map,
- (a) from that position search for the nearest edge on the edge map,
- (b) obtain stroke color,
- get stroke width from the weighted ridge map,
- (c) draw a longish stroke line with width obtained previously till the position of the found edge point,
- repeat this process until there are no more ridge points,
- go over the painted image and if there are areas left out (still blank) then do and repeat steps marked a,b,c with a constant stroke width to fill in the missing areas.

When painting with this variant a circular stroke template is used with variable radius.

In the case of this third algorithm the ridge scales are used when determining the width of the stroke line drawn from the actual ridge point till the nearest edge point. The smallest of the available strokes is assigned to the minimum ridge weight, the largest to the maximum ridge weight and the rest are proportionally mapped between the two values.

Images generated this technique are shown on Figures 10, 11 and 12.

When painting with the last method, we achieve multiple goals: the painting process gets a reasonable speedup, because there is no need for stochastic generation of positions, scales or orientations, all of these parameters are automatically obtained from the edge/ridge maps; we get a painting process that stands a bit closer to being "natural"



Figure 8: Example of image painted following edge orientation and scaling by ridge scales (with random stroke positions).

in the way that it follows the main image structures. Also, this technique gives space for further development, like adding shading to "hills" (which are spaces between ridges and edges) and "valleys" (which are negative ridges).

3 Conclusion

We presented a new painterly rendering method that is based on multiscale edge and ridge extraction which controls both the size and the scale of placed strokes on the canvas. Images generated this way are more closely connected to image contents and structure than previous

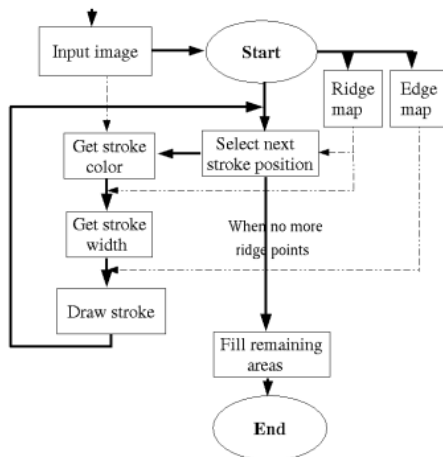


Figure 9: Main steps of the 3rd painting approach.



Figure 10: Example of image painted following edge orientations, scales determined from ridge scales, stroke positions are sequential. Strokes perform hue rotation proportionally to ridge weights.

painterly rendering algorithms were. By following edge directions and scaling the strokes related to ridge strengths painterly image representation becomes more compact and also provides a painting framework which can be the basis of further stroke-based rendering techniques.

References

- [1] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.
- [2] P. Haeberli. Paint by numbers: Abstract image representations. *In Proceedings of ACM SIGGRAPH 90*, pages 207–214, 1990.
- [3] A. Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. *In Proceedings of ACM SIGGRAPH 98*, pages 453–460, 1998.
- [4] R.D. Kalnins, L. Markosian, B.J. Meier, M.A. Kowalski, J.C. Lee, P.L. Davidson, M. Webb, J.F. Hughes, and A. Finkelstein. Wysiwyg npr: Drawing strokes directly on 3d models. *ACM Transactions on Graphics*, (21), 2002.
- [5] Z. Kato, T. Sziranyi, X. Ji, Z. Toth, and L. Czuni. Content-based image retrieval using stochastic paintbrush transformation. *In Proceedings of ICIP*, 2002.
- [6] L. Kovacs and T. Sziranyi. Creating animations combining stochastic paintbrush transformation and



Figure 11: Example of image painted following edge orientations, scales determined from ridge scales, stroke positions are sequential. Main edge contours are also overlaid.

motion detection. *In Proceedings of 16th ICPR, IAPR&IEEE*, 2:1090–1093, 2002.

- [7] L. Kovacs and T. Sziranyi. Coding of stroke-based animations. *WSCG 2004 Posters Papers Proceedings*, pages 81–84, 2004.
- [8] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 2(30):117–154, 1998.
- [9] P. Litwinowicz. Processing images and video for an impressionist effect. *In Proceedings of ACM SIGGRAPH 97*, pages 407–414, 1997.
- [10] B.J. Meier. Painterly rendering for animation. *In Proceedings of ACM SIGGRAPH 96*, pages 477–484, 1996.
- [11] J.D. Northrup and L. Markosian. Artistic silhouettes: A hybrid approach. *In Proceedings of NPAR*, pages 31–37, 2000.



Figure 12: Yet another example of image painted with the third method.

- [12] A. Santella and D. DeCarlo. Abstracted painterly renderings using eye-tracking data. *In Proceedings of NPAR*, pages 75–82, December 2002.
- [13] T. Sziranyi and Z. Toth. Random paintbrush transformation. *In Proceedings of 15th ICPR, IAPR&IEEE*, pages 155–158, 2000.
- [14] T. Sziranyi and Z. Toth. Optimization of paintbrush rendering of images by dynamic mcmc methods. *In Lecture Notes on Computer Science, Springer Verlag*, 2134:201–215, 2001.