AUTOMATIC COMPUTER GENERATION OF STIPPLING ILLUSTRATIONS WITH FELT-TIP COLOURED PENS

Germán Arroyo and Domingo Martín
Dep. Lenguajes y Sistemas Informáticos, University of Granada
C/ Periodista Daniel Saucedo Aranda s/n, 18071, Granada, Spain

Keywords: Rendering, Non-photorealistic rendering, Painting-like rendering, Drawing.

Abstract: Nowadays, non-photorealistic rendering is an area that not only focuses on simulates what artist do or the tools they use, but also on generates new expressive tools for digital art. In this paper we present a new algorithm to generate beautiful stippling illustrations with felt-tipped colour pen from a photograph or an image. This technique is not actually used by artists because technical limitations, therefore this new algorithm might be helpful. We introduce a novel stochastic approach to place coloured dots in a specific order based on the information of the contrast, borders and histogram of the input image. The system is able to generate an unlimited number of non regular synthetic colour dots without the necessity of being scanned. These dots will be composed in a specific order to generate the final illustration.

1 INTRODUCTION

Stippling is the technique of drawing using dots, which are composed of pigment in a single colour applied with a pen or a brush, changing the density to obtain different shades. The stippling technique can be altered to use colours. This technique must not be confused with pointillism, which uses small distinct dots of colour to create the impression of a wide selection of other colours and blending. The technique of coloured stippling overlaps the dots to shade the illustration whereas it is not allowed in pointillism.

Several problems happen when an artist try to stipple an illustration using felt-tip pens. The first problem is that the number of different colours of these pens are very limitted. A second problem is that the amount of ink and the porosity of the tip makes expensive to stipple a complete illustration, specially with medium-tip markers. The paper is also a problem because a thin paper cannot admit a large amount of ink, and it brokes; on the other hand, a thick paper spread out the ink too much, in such a way that the shapes become blurred and undefined.

This paper is structured as follows: In Section 2 we discuss related work. Section 3 we present an overview of our system. In Section 4 we explain the algorithms in detail. Section 5 discusses the results of our approach. The paper is concluded in Section 6.

2 PREVIOUS WORKS

Abstract representation of still images was introduced by Haeberli(Haeberly, 1990) using image color gradient and user interactivity for painting. Hertzmann(Hertzmann, 1998) places curved brush strokes of multiple sizes on images for painterly rendering. The technique fills color by using big strokes in the middle of a region and uses progressively smaller strokes as one approaches the edges of the region. Shiraishi and Yamaguchi(Shiraishi and Yamaguchi, 2000) improves the performance of above method by approximating the continuous strokes by placement of rectangular strokes discreetly along the edges to create painterly appearance. Santella and DeCarlo(Santella and DeCarlo, 2002) uses eye tracking data to get points of focus on images and create painterly rendering with focus information. There good works for painting terrains(Coconu et al., 2006; Bhattacharjee and Narayanan, 2008), but they do not work with general models. Rudolf el al. proposes a physically-inspired model to simulate wax crayons(Rudolf et al., 2003). All these techniques work well on single images but do not simulate colour

Arroyo G. and Martín D. (2010).
AUTOMATIC COMPUTER GENERATION OF STIPPLING ILLUSTRATIONS WITH FELT-TIP COLOURED PENS.
In *Proceedings of the International Conference on Computer Graphics Theory and Applications*, pages 263-266
DOI: 10.5220/0002821002630266

stippling.

Most of the related research on stippling is focused on generating dots according to the shading of a photograph, paying almost no attention to the shape of dots or the techniques that artists use(Secord et al., 2002). Some methods propose using circles instead of realistic dots(Mould, 2007)(Gooch and Gooch, 2001)(Schlechtweg et al., 2005)(Yuan et al.,)(Lu et al., 2003). This differs noticeably from the illustrations created by artists because natural dots have a gradient. Other methods focus on distributing the dots correctly along a surface according to the shading(Secord, 2002)(Pastor et al., 2004). In these cases, the use of Central Voronoy diagrams produce easily recognisable patterns. Renderbots is based on the idea of particles but the results, when they simulate stippling, have the same problems with patterns too(Schlechtweg et al., 2005). None of them uses coloured dots because it is a less usual technique.

In the next Section, we will expose our solution to stipple with simulated felt-tip pens.

3 OVERVIEW

First, we will discuss the proposed system. It is based on the scheme presented in Figure 1. As we can see, the algorithm has the following steps:

1: The information is obtained from the input image.
2: A matrix with probability is generated from this information.
3: The algorithm enters in a loop:
4: **loop**
5: The new place and size of the dot is computed.
6: A new dot is generated with a simplified colour of the region.
7: The dot is composed in the output image.
8: **end loop**

This algorithm finishes when the stop condition is reached or the user stops it. A matrix of probability will guide all the process to place the dots, in such a way that the order of the dots placement is determined by it.

4 ALGORITHMS

The algorithm can be divided in several subalgorithms that takes an input and returns an output to the next step. These algorithms are explained in the following subsections.

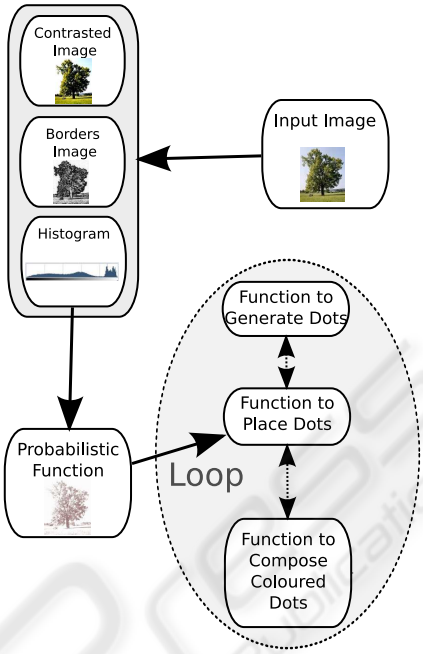

Figure 1: An overview of the algorithm.

4.1 Obtaining Information

First of all, it is necessary to filter the photograph to obtain relevant information before using the algorithm to place dots. This filtering process is based on what artists do before they draw a stipple illustration. The process may vary from artist to artist, but in general, they begin marking silhouettes and stippling them. The next step is to stipple on the darkest areas of the image. It is also important to stipple in areas where details can be enhanced and where there are different or relevant elements of the photograph. The background is usually irrelevant, so, the most repeated tone in the photograph is ignored by the artist. Colour stippling is not an exception to this rule.

4.2 Generating the Function of Probability

A discrete representation of a probability density function (PDF) is constructed with the information obtained in the previous step. The user adjusts the relevance of the histogram, the contrasted image and the borders in the PDF by hand. Hence, a 2d array with the values of the PDF is initialized.

Once the array has been generated, the next step is iterate until all the dots have placed.

4.3 Algorithm for Positioning Dots

The algorithm for dots placement removes the patterns in the final image. We use a method that is also based on a stchocastic algorithm but including some parameters that are controlled by the user. These parameters handle the number of placed dots and the size of the dots.

The algorithm places the dot randomly according to the values of the array 2d, in such a way that highest values has more probability of being chosed to place a dot.

The dots can be placed in the output image using the following formula for every pixel of the dot:

$$C_d = (1-A)\cdot C_d + (A)\cdot C_s \quad (1)$$

Where C_d is the colour destiny, C_s is the colour source, and A is the value of the intensity of the dot.

4.4 Algorithm for Generating Colour Dots

The automatic generation of dots removes the artificial appearance of the illustration. Before generating a dot, the algorithm must decide what the colour will be for all the colour. We must take in consideration that each individual dot cannot contain more than one color.

The colour is determined by a simple equation:

$$C_{dot} = \frac{C_{pixel}\cdot 3.0}{(C_{pixel}.red + C_{pixel}.green + C_{pixel}.blue)} \quad (2)$$

This normalization of the colour produces a flattening and a reduction of the brightness, which is used in computer vision algorithms to simplify the illumination of the scene(Finlayson et al., 1998). Additionaly if the colour is almost black, the luminosity can be increased, specially if the intention is to print the image. The gray colours are substituted by only two tones of grey. The reason is that grey colours do not contribute with too much detail to the scenery and can be obtained easily by stippling on the same place repeatedly.

The problem is reduced to generate a grey scaled dot, and then multiply the obtained colour by the levels of intensity of the dot. White values are not a problem because if a white colour is found, the algorithm does not stipple there.

The proposed algorithm to obtain grey dots is based on Monte Carlo methods. In these kinds of methods the algorithms decide at every step when the solution is false, but do not know when the solution is true. Therefore, the algorithm iterates a certain number of steps until it reaches an approximate solution.

The algorithm is as follows:

1: Create a matrix (P) of local probabilities
2: Create a matrix (I) of intensity values
3: Create a matrix (A) of absorption values
4: Generate a seed
5: **for** i = 0 **to** $i <$ N_DROPS **do**
6: $F_b(n)$ returns a position (p_x, p_y)
7: Deposit the ink in the returned position
8: **end for**
9: Apply a Gaussian filter to smooth the result

N_DROPS is the number of iterations of the algorithm, it can be estimated from the desired size of the dot. The size of the matrices P, I and A is the double of the size of the dot. The matrix P is a probability density function (PDF). I is the final image of intensities, whereas A is the abosorption of the paper, and it can be obtained from a gradient image.

The seed can be simply a small circle, this circle writes the matrix I at its center with a dark value. The matrix P satisfies the discrete condition $\sum_{i,j} P(i,j) = 1$ because it is a PDF. Therefore, the values of the matrix P are initialized to 0 but in the places where is the seed. These cells are initialized to an uniform value scaled according to the number of used cells.

The final dot is used in the algorithm described at previous Section.

5 RESULTS

The algorithm produces a stippling illustration with felt-tip pens that degrade the colours in an aesthetic way. The algorithm also can detect how many dots place in the final illustration by using the information of the contrasted image. If the background of the image is more or less plain, it is detected and removed from the photograph. The algorithm can also take as input a rendered image from a 3D scene or painted illustrations, it also works with complex photographies, as shown in Figure2. It can be appreciated how the algorithm produce good results even with very compex images or photograph because the shape and colour of the dots.

All these images has been generated with felt-tip pens between 40mm and 60mm.

6 CONCLUSIONS AND FUTURE WORKS

We have presented an algorithm to automatically draw coloured illustrations and simulates felt-tip stippling. We have developed a fast and direct equilibra-

Figure 2: A illustration generated by our algorithm.

tion technique that is based on a probabilistic model. This algorithm draws the dots in a natural order, just like artists do. We have introduced an automatic control to stop the algorithm detecting when the illustration is finished. Our system provides both interactivity and high-quality output. It can take as input both photographs, 3D rendered images and illustrations.

Future research is proposed into how more artistic knowledge can be included automatically within the application. A study of how to print this kind of images should be also interesting. Another important issue of future work is the introduction of temporal coherence when applied to frames in a video, specially when try to use the same colours when objects which are moving in a scene.

ACKNOWLEDGEMENTS

We thank the Ministerio de Educación y Ciencia of Spain for funding part of this work under the project TIN2007-67474-C03-02.

REFERENCES

Bhattacharjee, S. and Narayanan, P. J. (2008). Real-time painterly rendering of terrains. In *Proceedings of Sixth Indian Conference on Computer Vision, Graphics & Image Processing*.

Coconu, L., Deussen, O., and Hege, H. C. (2006). Real-time pen-and-ink illustration of landscapes. In *NPAR 06: Proceedings of the 2nd symposium on Non-photorealistic animation and rendering*, pages 27–36.

Finlayson, G. D., Schiele, B., and Crowley, J. L. (1998). Comprehensive colour image normalization. In *ECCV '98 : European conference on computer vision*, volume 1407, pages 475–490.

Gooch, B. and Gooch, A. (2001). *Non-photorealistic Rendering*. A. K. Peters.

Haeberly, P. (1990). Paint by numbers: abstract image representations. In *SIGGRAPH'90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 207–214.

Hertzmann, A. (1998). Painterly rendering with curved brush strokes of multiple sizes. In *Computer Graphics (Annual Conferences)*, number 32, pages 453–460.

Lu, A., Morris, C., Taylor, J., Ebert, D., Rheingans, P., Hansen, C., and Hartner, M. (2003). Illustrative interactive stipple rendering. In *IEEE Transactions on Visualization and Computer Graphics*, volume 9, pages 127–138.

Mould, D. (2007). Stipple placement using distance in a weighted graph. In *Proceedings of Computational Aesthetics in Graphics*, number 3, page unknown.

Pastor, O. M., Freudenberg, B., and Strohthotte, T. (2004). Real-time animated stippling. In *Proceedings of NPAR 2004*, volume 23, pages 62–68.

Rudolf, D., Mould, D., and Neufeld, E. (2003). Simulating wax crayons. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, page 163.

Santella, A. and DeCarlo, D. (2002). Abstracted painterly renderings using eye-tracking data. In *NPAR 02: Proceedings of the 2nd symposium on Non-photorealistic animation and rendering*, pages 53–58.

Schlechtweg, S., Germer, T., and Strothotte, T. (2005). Renderbots: Multi agent systems for direct image generation. *Computer Graphics Forum*, 24:283–290.

Secord, A. (2002). Weighted voronoi stippling. In *Proceedings of NPAR*, pages 37–43. ACM Press.

Secord, A., Heidrich, W., and Streit, L. (2002). Fast primitive distribution for illustration. In *Thirteenth Eurographics Workshop on Rendering*, pages 215–226.

Shiraishi, M. and Yamaguchi, Y. (2000). An algorithm for automatic painterly rendering based on local source image approximation. In *NPAR 00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, pages 53–58.

Yuan, X., Nguyen, M. X., Zhang, N., and Chen, B. Stippling and silhouettes rendering in geometry-image space. In *Proceedings of Eurographics Symposium on Rendering*, pages 193–200.