User-defined texture synthesis

Francesca Taponecco
Interactive Graphics Systems Group
Department of Computer Science
Technische Universität Darmstadt
Fraunhoferstr. 5,
64283 Darmstadt, Germany
ftapone@gris.informatik.tu-darmstadt.de

ABSTRACT

Synthesis of textures is a very popular and active area of research; the applications and the areas of interest are various and significant. In the last years, much work has been done in order to optimize the synthesis process, speeding up the methods and minimizing the processing errors. Recent efforts in this area are concentrated in producing flexible algorithms and in introducing tools, which augment the texture with artistic effects. In this work we present a novel approach for flexible texture synthesis: starting from an input sample the process generates an arbitrary resolution output image; the characteristics of this output texture are user-defined, as the user can freely choose a force field, determine color variations and add further features. The synthesis process is fully automatic and does not require additional intervention. The resulting outputs can be interpreted as filtered versions of a texture or as being obtained through transfer functions.

Keywords

Personalized texture synthesis, user-dependent settings, force fields, data visualization, artistic effects, image-processing filters.

1 INTRODUCTION

Image generation and image handling are very fascinating fields of research, lots of current works deal with producing special effects and variations of an original image; filter synthesis and analysis also offer many possibilities for image manipulation. Texture synthesis as well is a major area of study: texture mapping is often utilized to add more variety to computer-generated scenes and to provide objects with complex materials appearance (*e.g.* color, bump mapping, transparency, shading) and patterns: this improves the perception of shape and geometry and increases the realism on surfaces.
In this work we present a new idea for personalized texture synthesis. We start by introducing motivation and applications and then we present some previous work done in this field. Next, we explain our algorithm: a novel flexible and straightforward process, which is easy to implement and adapt to several conditions. We show some of our results and summarize the features of the approach; at last, we conclude with discussion and address some possible future work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WSCG Short Communication papers Proceedings
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.
Copyright UNION Agency – Science Press

2 MOTIVATION AND PREVIOUS WORK

Recently, lots of effort has been put in texture synthesis and image processing. The intention of texture synthesis is the following: given a sample of a texture, synthesize an arbitrary resolution output texture that is perceptually similar to the input. Regarding image processing, the objective is in general to produce algorithms and methods, which are as flexible and interactive as possible. In this chapter we will briefly summarize some of the preceding work done in these fields, then we refer to the works that are more directly related to our study. Regarding the research for texture synthesis applications and image processing, many novel ideas are notable; some of them are directly based on

texture synthesis techniques, some others are inspired by the *texture synthesis* theory.

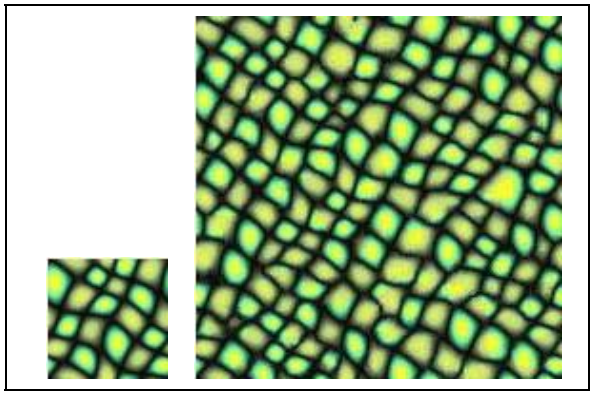


Figure 1. Synthesizing an arbitrary resolution output texture from an input sample.

Drori at al. [Dro03] propose an interesting method for fragment-based *completing* missing parts after their removal from the original image. Bertalmio et al. [Ber00], [Ber03] also fill in missing regions of a picture, for example for restoration and reconstitution - *inpainting* - of images. Their approach works for the structure and for the texture (image details) of the picture as well. Hertzmann et al. [Her01] process images by examples: their method involves two stages: a *design phase*, in which a pair of images (one is a filtered version of the other) is presented as training data, and an *application phase*, in which the learned filter is applied to a new target image in order to create an analogous filtered result. Brooks et al. [Bro02] proposes *self-similarity texture warping*, Freeman et al. [Fre02] performs *super-resolution* by example: they train using different pairs of images with low and high resolution. Ashikhmin [Ash01] produces the effect of rendering a given image with the texture appearance - *texture transfer* - of a sample image. Applications of the texture synthesis theory can also be found in scientific visualization [Tap03], where a synthesis approach based on Markov Random Fields model is used to generate *vector fields*, providing the visualization with various appearances and smooth transitions. Regarding color effects, Welsh et al. [Wel02] introduce a technique for *colorizing* grayscale images by transferring color between a source color-image and a destination grayscale-image. Greenfield et al. [Gre03] present a method for *recoloring* a destination image according to the color scheme found in a source image. The painting's palette can be then inverted or applied to a different image. Interesting 3d approaches for synthesizing textures on surfaces are somehow similar to our idea, but they are mostly used to texture a manifold surface, adapting the direction field for the texturing. Praun et al. [Pra00] propose „Lapped Textures", an approach that uses overlapping sample patches and copies them over a surface. Zhang et al. [Zha03] realize a progressively variant synthesis for texture mapping on surfaces. Turk [Tur01] proposes the use of oriented patches for surface texture synthesis. Another significant approach that synthesizes textures directly on surfaces is from [Wei01].

2.1 Texture synthesis

The intention of texture synthesis is fundamentally the following: given a sample of a texture, synthesize a new texture of arbitrary resolution that appears to be generated by the same underlying process (see Figure 1). That is, two texture images are perceived by human observers to be the same if some appropriate statistics of these images match: this has to be achieved by the synthesis algorithm.

2.1.1 Patch-based texture synthesis

One way to synthesize textures in a fast way is the block-by-block method of Efros and Freeman (See [Efr01] for more details). This process - *image quilting* - is a simple and fast image-based method: the block substitution is here optimized by stitching together small patches of existing images and minimizes the error on the boundary cut where the patches join. Interesting alternative methods that generate images from examples are the fast and patch-based technique of [Xuy00], [Lia02] and more recently [Nea03], [Kwa03], [Coh03]. For patches-based synthesis methods, the size of a block depends on the structure of the texture we want to synthesize.

2.1.2 Pixel-based texture synthesis

Many approaches use Markov Random Fields (MRF) theory to model a texture, and they generate the output pixel-by-pixel [Efr99]. The large texture is produced in scan-line order, where each pixel is set after comparing its neighborhood to all similar shaped neighborhoods in the sample texture, which is both stationary and local [Wei00]. This comparison leads to a distance function, which corresponds to the probability needed to choose the best fitting pixel (the most similar one). This operation is naturally a time consuming process, which can be speeded up using optimization algorithms such as *tree-structured vector quantization* [Ger92], [Wei00] and *image pyramids*[1]. We have implemented this multi-resolution analysis using *Gaussian* pyramids (each level is here obtained via successive filtering and

[1] The image pyramids is a multi-resolution synthesis process based on sub-band transforms: the pyramid is a multi-scale set of image levels. In the lower levels are comprised the coarse large-scale features of a picture, while in the higher levels - high resolution - the details are to find (see also [Hee95], [Bon97], [Por00]).

down-sampling operations by a factor of 2 – *i.e.* the images result to be blurred and decimated versions of the original one, through low pass filtering). For pixel-by-pixel synthesis methods, the size of the neighborhood depends on the structure of the texture we want to synthesize: the neighborhood has to comprise enough information in order to be able to reconstruct the texture pattern in the output image. Using multi-resolution, smaller neighborhoods produce analogous results to larger ones in single-resolution.

3 OUR APPROACH

Our method has been inspired by a previous work [Tap03], which visualizes precise and smooth vector fields, using texture synthesis theory. The target of our technique is to offer a flexible user-dependent tool to intervene in the texture synthesis and modify patterns and images. Basically, we generalize and extend the texture synthesis process: starting from a sample image or, here more generally, from a various set of sample images, the user can additionally decide how to complete the process, choosing between variation options, which affect the original input sample at a per-pixel level. For every point in the output image during the scan-line synthesis process, the information, which depends on the parameters set by the user, is mapped in the pixel. The choice of a particular image-processing filter influences the appearance of the sample, for instance to gradually change the color of the output texture in the desired way (Figure 11, Figure 12, Figure 13); the choice of a direction field determines scaling variations - *magnitude or norm* - (Figure 8), which transform the original pattern, and new directions - *phase* - (Figure 7), along which the sample image has to be rotated.

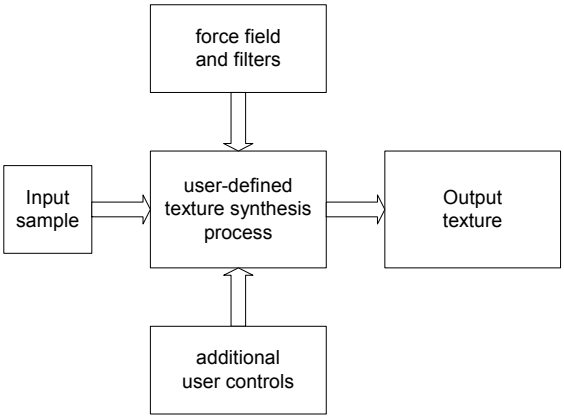


Figure 2. Blocks' scheme of the process flow for a standard application case (single input).

To perform this, the starting sample is rotated, amplified or minimized, distorted or sheared. We explain the algorithm in more detail in the following section. For our work we use a pixel-by-pixel texture synthesis[2], which is slow, but guaranties smoother results and preserves continuity, especially in critical points.

4 ALGORITHM

4.1 Standard Input sample

In Figure 2, we exemplify the process flow in case of a single input image. The user chooses the sample, hence defines a force field (for instance the one of Figure 3) and may set additional variable parameters. Modified texture synthesis is then executed. In Figure 4 and Figure 5 one can observe the resulting textures.

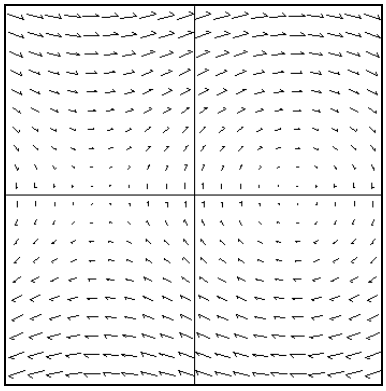


Figure 3. An example of force field.

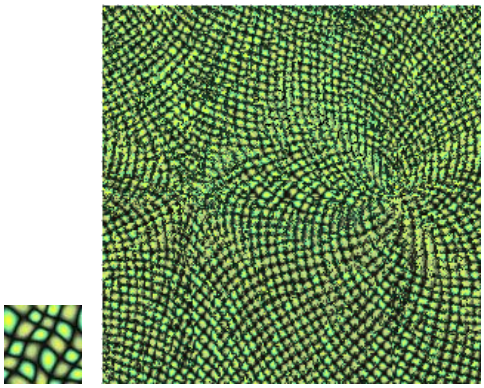
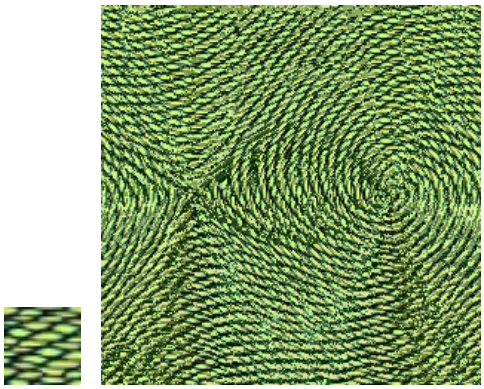


Figure 4. Modified texture synthesis, obtained using the force field of Figure 3.



[2] Based on the work of Wei and Levoy [Wei00].

Figure 5. Accentuating a main direction in the input sample of Figure 1 produces a stronger spiraling effect in the output texture.

4.2 Input set

In addition to the section above, we want here to use a set of input samples in place of an only one: the set can be pre-selected or also user-defined. This can be particularly useful in case the user wants to emphasize or differentiate regions of the output.

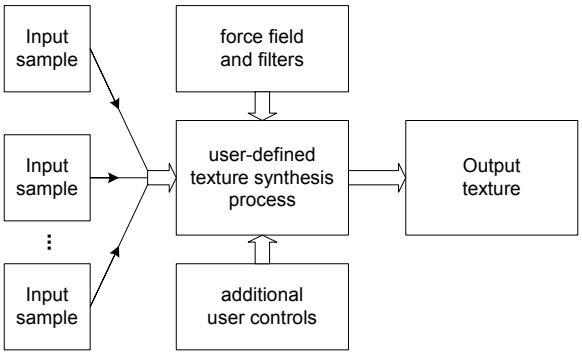


Figure 6. Blocks' scheme in case of multiple input samples (input set).

4.3 Input set variations

The input sample or sample set is modified as follows: as briefly described in Section 3, the user defines a force field and this influences the output textures in amplitude and rotation: see some results in Figure 10 and from Figure 15 to Figure 20.

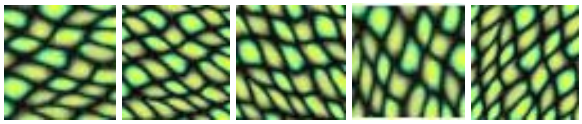


Figure 7. Rotating the input sample of Figure 1.

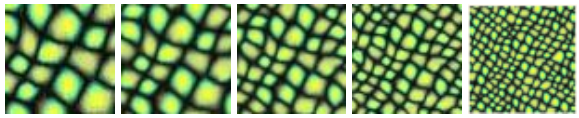


Figure 8. Scaling the input sample of Figure 1.

To achieve this, every input sample has been rotated (see Figure 7) and scaled (see Figure 8) according to the specific field. That is, for each sample a number of modified versions is calculated, or, better, pre-computed. Two correspondent results are in Figure 4, Figure 5. Naturally the direction influence of the field is stronger for input samples that are characterized by a more accentuated major direction of anisotropy. The way the system chooses a particular sample from the whole set depends on the information contained in the corresponding output pixel that has to be drawn. In detail, the output texture is being synthesized per-pixel in scan line order; at every step a pixel value has to be set: the algorithm looks for the best matching pixel, by comparing the neighborhood of the searched pixel with all the similar ones in the input sample; this input sample in turn is being chosen from the complete input set, depending on the user-provided information contained in the point at the current position in the output. Concluding, each pixel in the output texture can communicate the decided characteristics of color and pattern, as it univocally corresponds to a specific sample image.

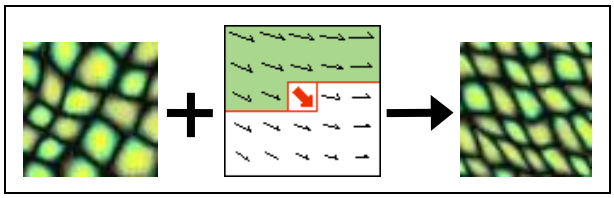


Figure 9. For each pixel in the output, the sample texture is transformed through the vector value (thick arrow) at the pixel position.

More formally, and for a standard case with single input texture T, the algorithm steps are the following. In the output texture, every pixel at the generic position (x, y) is characterized by a vector value $\vec{v}$:

$$(x, y) \Rightarrow \vec{v} \in \Re^d$$

where d is the vector dimension.

Every vector $\vec{v}$, depending on the set force field, comprises scaling and rotating information; hence it corresponds to a particular modified (scaled and/or rotated) version, $T_{(x,y)}$, of the original sample texture T (this is illustrated in Figure 9):

$$\vec{v}(x, y) \Leftrightarrow T_{(x,y)}$$

More generally, when further user-defined parameters have been set, we can connect a specific transformation f (e.g. through filtering operation) to every texture sample $T_{(x,y)}$, which belongs to the input set:

$$f_{\vec{v}(x,y)} \Leftrightarrow T_{(x,y)}$$

Here, the transformation f respectively corresponds to scaling, rotating, coloring, blurring, sharpening, etc. operations:

- $f_{\vec{v}(x,y)} = Scale(\|\vec{v}(x, y)\|)$
- $f_{\vec{v}(x,y)} = Rotate(\angle\vec{v}(x, y))$
- $f_{\vec{v}(x,y)} = Blur(g_{Blur}(\vec{v}(x, y)))$
- $f_{\vec{v}(x,y)} = Sharpen(g_{Sharpen}(\vec{v}(x, y)))$
- …

Figure 10. We used here different direction fields to influence the output texture. The input sample is the previous one from Figure 20.

4.4 Appearance

In Figure 10, we show some results we obtained for the sample "cane pattern" (from Figure 20), changing the direction field each time. Afterward (Figure 15 to Figure 20), we show further results obtained using a variety of patterns, which correspond to common material appearances, obtained this time always choosing the force field illustrated in Figure 3.

4.5 Areas of interest

Particular points of interest or critical points can be highlighted in the output, just by choosing a special sample or a significant color or filter to match the properties on that point. Figure 11 shows how color transitions map different amplitudes or areas of interest.

4.6 Transformations

We also tried out creating transitions between different images: using multiple input samples, it is possible to generate an output image, which combines multiple appearances. Therefore, transformations occur in the output texture, which are smooth, due to the features of the pixel-based method. We are still perfectioning this technique and we refer our preliminary results as future work.

5 RESULTS

The main contribution of our work is to offer personalized options for the generation of user-dependent textures and to combine together several artistic effects in the synthesis process.

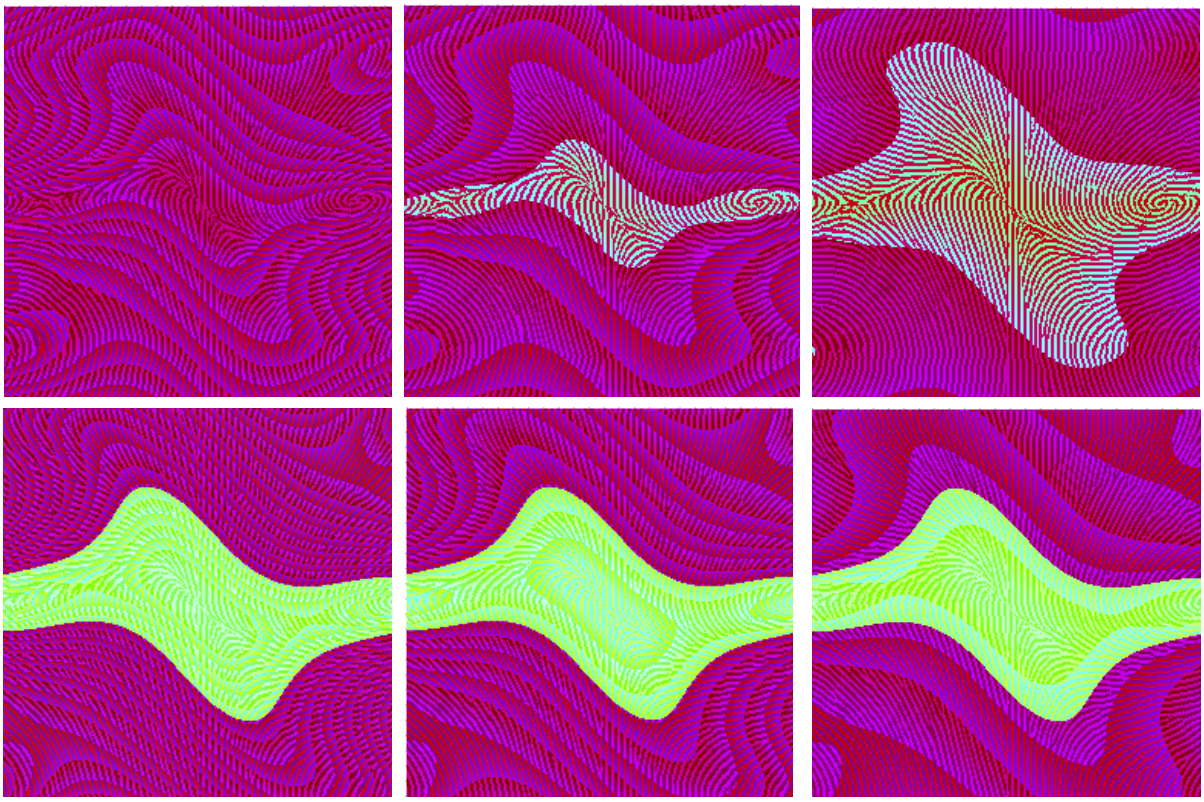


Figure 11. Starting from a simple line pattern, we curved these lines along the field direction, and applied color masks to different regions; this generates color gradient transitions.

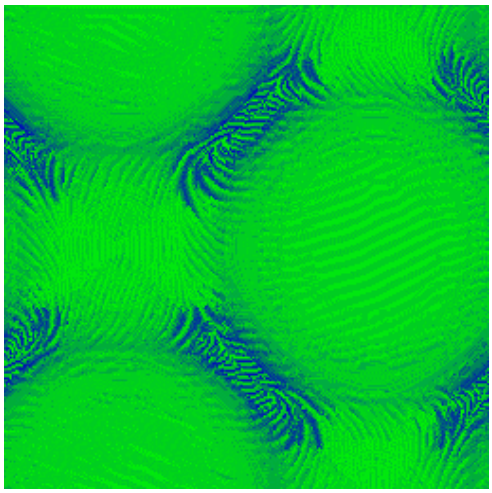


Figure 12. Color effects.

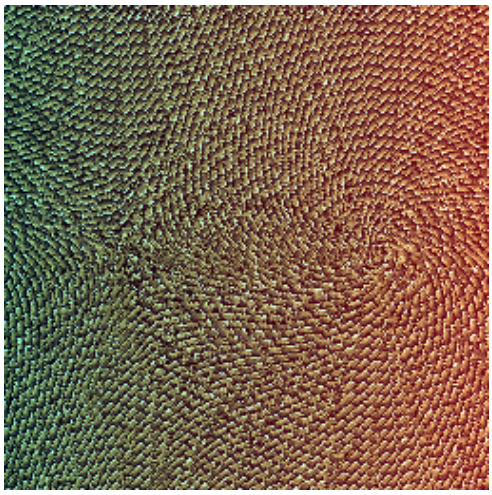


Figure 13. Color effects.

The starting sample pattern (or set of patterns) is freely modified and filtered, and then reproduced in an arbitrary resolution personalized output texture. We applied our idea to different sets of sample images, obtaining the following results. In the Figures above (Figure 4 to Figure 13 and from Figure 14 to Figure 20), it is possible to observe some of the possibilities of this algorithm. Note that, independently of the input sample's pattern, the method guarantees a smooth output texture. Our technique works well for stochastic textures as well as for structured textures, however we applied the technique mainly using structured anisotropic samples, as directional textures better highlight how the orientation of the image follows the user-specified direction field (see for example the difference between Figure 4 and Figure 5); in particular fine-structured sample textures are particular suited to produce precise output also in case of strong curvature changes in the output image. In case of a coarser structure in the sample, instead, a bigger sized neighboring area around each pixel is required to learn and reproduce the sample pattern. Furthermore, the user can better highlight a flow or a specific area using color filtering in addition. Regarding the sampling rate of the superimposed force field, it is defined by the pixel count in the output image, *i.e.* the output dimension. Textures are important for numerous applications, for example the use of textures is fundamental for large scenes and terrains, therefore operating with a force field formula would add relevant features for better visualizing streams of winds, gas or fluids. Texturing 3d surfaces also needs curving the texture and adapting it to fit the 3d object, thus, again force fields and color or light effects are fundamental to modify the appearance and to add more realism to the scene.

Figure 14. Bricks texture.

6 CONCLUSIONS

We have presented a new and versatile technique that allows a user to synthesize a texture in a flexible way. Thanks to user-dependent options and to the possibility of influencing the output image through a free choice of input parameters and values, a variety of desired outputs can be obtained and promising extensions are possible. There is no limitation in the choice of the force fields; this makes this method general. Our examples have shown that this approach provides smooth transitions between different color or pattern areas and generates good outputs also in presence of critical points in particular orientation fields. The pixel-by-pixel process guaranties high-quality texture results, as it selects, at every step, the best choice fitting its neighbors. Consequently, the accuracy that is achieved by this approach is high and provides good results also in scientific visualization. Using different sample images to generate a single output texture also leads to interesting image transformation effects. Using several kinds of filters, painting and artistic

effects can be attained. Furthermore the method is general and easy to use. In general, the main applications of our approach are in the fields of image manipulations, in the generation of artistic effects on images and in the production of user-defined patterns. Our methods allow a user-defined filter application at a per-pixel level, although also pre- and post-processing filtering is possible.

7 FUTURE WORK

Possible extensions can be achieved by investigating new kinds of filters and various artistic effects in order to further differentiate and map particular samples to different regions of the output. Transitions and transformations between different images also need to be further explored. Improvements may be possible by speeding up the software or combining algorithms together to improve the performances.

8 REFERENCES

[Ash01] Michael Ashikhmin. Synthesizing natural textures. In *2001 ACM Symposium on Interactive 3D Graphic*s, pages 217–226, March 2001. ISBN 1-58113-292-1.

[Ber00] M. Bertalmio, G. Sapiro, V. Caselles and C. Ballester. Image Inpainting. In Proceedings of ACM Siggaph 2000, ACM Press, pp. 417 – 424.

[Ber03] M. Bertalmio, L. Vese, G. Sapiro and S. Osher. Simultaneous structure and texture image inpainting. In IEEE Conference on Computer Vision and Pattern Recognition. Vol. 12, No. 8, August 2003.

[Bon97] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. *Proceedings of SIGGRAPH 9*7, pages 361–368, August 1997. ISBN 0-89791-896-7.

[Bro02] Stephen Brooks and Neil Dodgson. Self-similarity based texture editing. *ACM Transactions on Graphic*s, 21(3): 653–656, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[Coh03] M. F. Cohen, J. Shade, S. Hiller, O. Deussen. Wang Tiles for image and texture generation. ACM Siggraph 2003.

[Cri03] A. Criminisi, P. Perez, K. Toyama. Object Removal by Exemplar-based Inpainting. Proceedings CVPR 2003, Madison, US, June 2003.

[Dro03] Iddo Drori, D. Cohen-Or and H. Yeshuun. Fragment-Based Image Completion. 2003 Proceedings of ACM SIGGRAPH 2003.

[Efr99] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Visio*n, pages 1033–1038, 1999.

[Efr01] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 200*1, Computer Graphics Proceedings, pp. 341–346. ACM SIGGRAPH, August 2001

[Fre02] W. Freeman, T. R. Jones and E. C. Pasztor. Example-based super-resolution. IEEE Computer Graphics and Applications 2002, pp. 56 – 65.

[Ger92] A. Gersho and R. M. Gray. Vector Quantization and Signal Compression. Kluwer Academic Publisher, 1992

[Gre03] Gary G. Greenfield and Donald D. House. Image Recoloring Induced by Palette Color Associations, WSCG 2003, February 2003.

[Hee95] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. *Proceedings of SIGGRAPH 9*5, pages 229–238, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[Her01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of ACM SIGGRAPH 200*1, Computer Graphics Proceedings, pp. 327–340. ACM Press / ACM SIGGRAPH, August 2001.

[Kwa03] V. Kwatra, A. Scoedl, I. Essa, G. Turk, A. Bobick. Graphcut textures: Image and Video Texture Synthesis using graph cut. ACM Siggraph 2003.

[Lia02]L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and Heung Shum. *Real-time texture synthesis by patch-based sampling*. ACM Transactions on Graphics, 20(3):127--150, July 2001. 4

[Nea03] A. Nealen and M. Alexa. Hybrid Texture. Eurographics Symposium of Rendering 2003. Leuven , Belgium, June 2003.

[Por00] Javier Portilla and Eero P. Simoncelli. A parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. International Journal of Computer Vision, October 2000.

[Pra00] Emil Praun, Adam Finkelstein, and Hugues Hoppe. Lapped textures. *Proceedings of SIGGRAPH 200*0, pages 465–470, July 2000. ISBN 1-58113-208-5.

[Tap03] F. Taponecco and M. Alexa. Vector Field Visualization using Markov Random Field Texture Synthesis. Eurographics / IEEE TCVG Visualization Symposium Proceedings, pp.195 – 202, Grenoble, France, May 2003.

[Tur01] Greg Turk. Texture synthesis on surfaces. In *Proceedings of ACM SIGGRAPH 200*1, Computer Graphics Proceedings, Annual Conference Series, pages 347– 354. ACM SIGGRAPH, August 2001.

[Wei00] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 200*0, pages 479–488, July 2000. ISBN 1-58113-208-5.

[Wei01] Li-Yi Wei and Marc Levoy. Texture synthesis over arbitrary manifold surfaces. In ACM, editor, *SIGGRAPH 2001 Conference Proceedings, August 2001, Los Angeles, C*A, pages 355–360, New York, 2001.

[Wel02] T. Welsh, M. Ashikhmin and K. Mueller. Transferring color to greyscale images. ACM Siggraph 2002. San Antonio, July 2002, pp. 277 – 280

[Wit91] Andrew Witkin and Michael Kass. Reaction-diffusion textures. *Computer Graphics (Proceedings of SIGGRAPH 91*), 25(4): 299–308, July 1991. ISBN 0-201- 56291-X. Held in Las Vegas, Nevada.

[Xuy00] Y. Xu, B. Guo and H.-Y. Shum. Chaos mosaic: Fast and mamory efficient texture synthesis. Tech. Rep. MSR-TR2000 -32, Microsoft Research, 2000.

[Zha03]J. Zhang, K. Zhou, L. Velho, B. Guo, H.-Y. Shum. Synthesis of progressively-variant textures on arbitrary surfaces. ACM Siggraph ’03, August 2003.

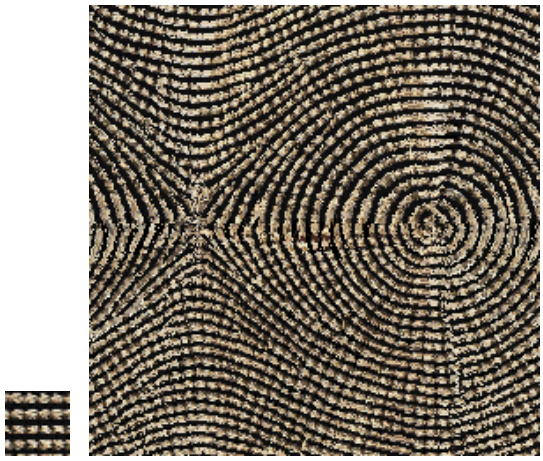


Figure 15. Obtaining the same output pattern using different input samples: wool tricot pattern.

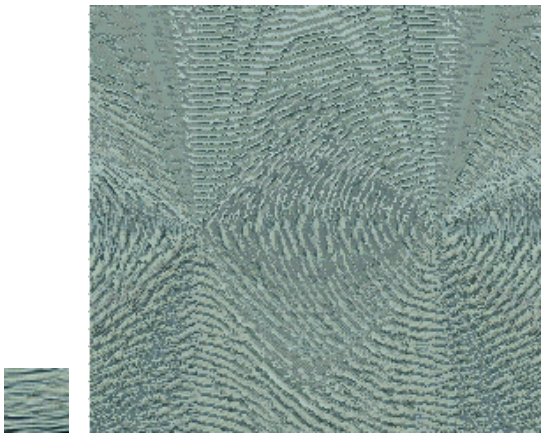


Figure 16. Waves pattern.

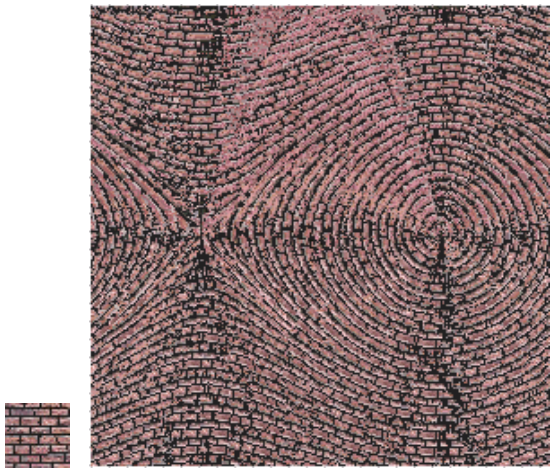


Figure 17. Bricks pattern.

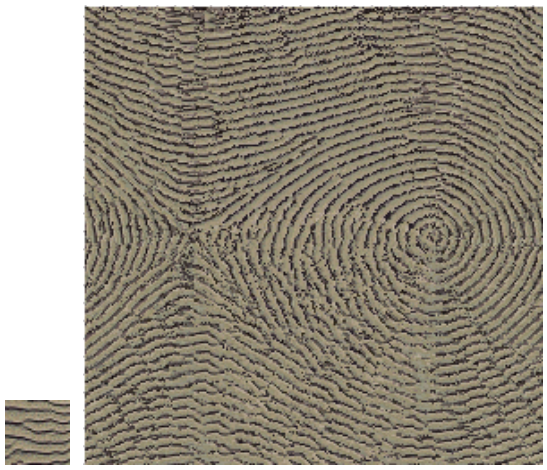


Figure 18. Sand pattern.

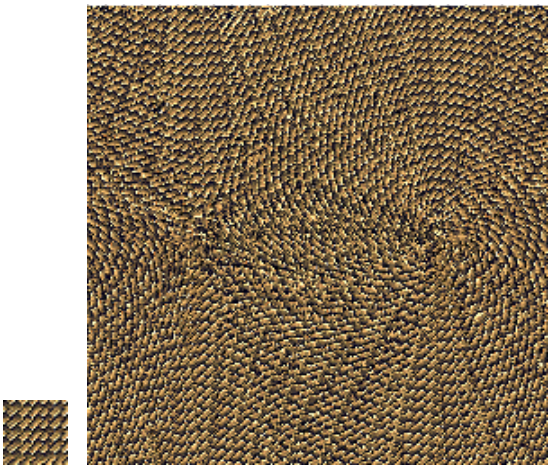


Figure 19. Fabric pattern.

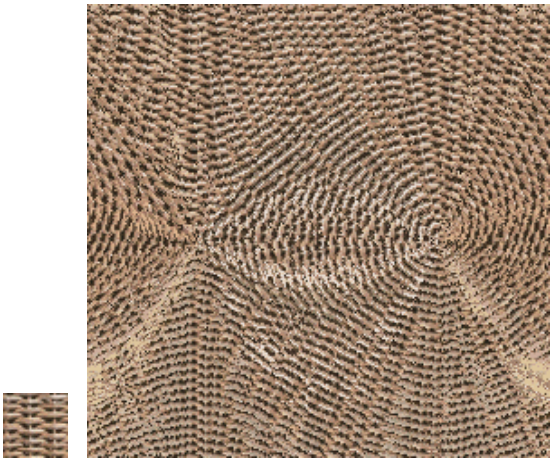


Figure 20. Cane pattern.