Steerable Texture Synthesis

Francesca Taponecco and Marc Alexa

Department of Computer Science, TU Darmstadt

Abstract

Texture synthesis is typically concerned with the creation of an arbitrarily sized texture from a small sample, where the pattern of the generated texture should be perceived as resembling the example. Most of the current work follows a Markov model approach. The texture is generated by finding best matching pixels or patches in the sample and then copying them to the target. Here we extend this concept to incorporate arbitrary filters acting on the sample before matching and transferring. The filters may vary over the generated texture. Steering the filters with properties connected to the output image allows generating a variety of effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation Display algorithms

1. Introduction

Texture synthesis from examples seems to have reached a mature state. Given a sample texture with clear pattern, it is possible to create a texture of arbitrary size that is perceived as resulting from the same underlying process. Current methods for the generation of the texture are related to Markov models, i.e. the texture is generated by copying stochastically matched pieces from the sample.

Large textures are found and used everywhere in computer graphics these days. Synthesizing these textures from small samples is a powerful way of saving on storage. However, due to the complexity of processes generating textures in the real world, one sample is typically not sufficient to describe a large texture. Oftentimes, a texture seems to be generated from a few underlying processes, which blend or merge over the surface. In addition, other easily discerned properties such as lightness, saturation, color, orientation, size, etc. change over the surface.

To accommodate these effects we generalize pixel by pixel texture synthesis. The main idea is to allow the texture being generated from a set of texture samples. Each output texel is associated to a sample texture in the set. The set might be generated from a small discrete set of input samples that is enlarged by using steerable filters, which change properties such as orientation or size.

2. Motivation and Related work

Currently, different methods are used for texture synthesis: depending on the performances to achieve, a patch- or a pixel-based approach could be preferable (see [DMLG02]). Our algorithm generates textures one pixel at a time, for this reason we refer to previous work based on *per-pixel* synthesis. Wei & Levoy [WL00] create the output in scan-line order and speed up the fundamental algorithm of Efros & Leung [EL99] using tree-structured vector quantization and multi-scale pyramids [HB95]. Ashikhmin [Ash01] reduces the computation and increases image coherency for highly structured patterns. Hertzmann *et al.* [HJO*01] combine the approaches together, getting the benefits of both.

These methods have been optimized over the last years and are capable of generating high quality results at reasonable computation times; though, most of them only synthesize homogeneous textures starting from single samples. However, it would be interesting to produce *ad hoc* modified outputs, allowing the user to vary features of a chosen pattern, still reproducing its main structure. Recently, a considerable number of techniques, as[Tur01], [WL01], [YHBZ01],[SCA02],[GS] recognize the lack of local control for output textures and the necessity to offer more flexibility and user intervention for a better match with real appearances of natural objects. The early work of Ashikhmin[Ash01] proposes editing textures; still with the scope of adding variation to textures generation, Wei[Wei03] suggests creating solid textures from multiple 2D-views and

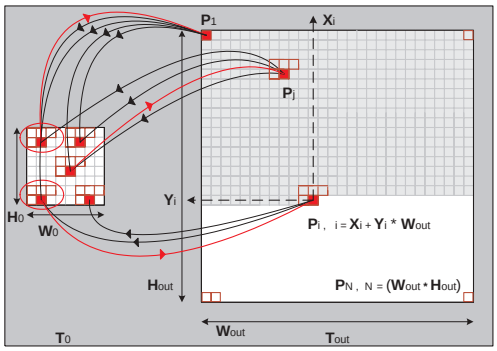

Figure 1: *Standard approach: the pixels P_i of the output T_{out} are set checking the most probable pixels in the input T_0.*

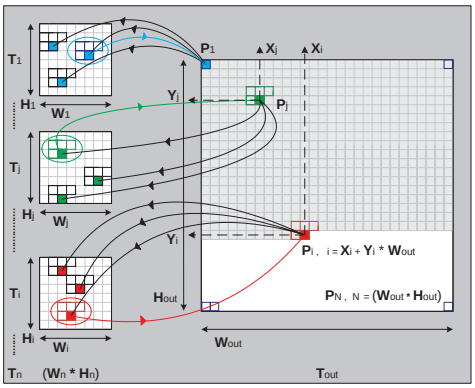

Figure 2: *Our approach: unlike the basic approach, the best matching pixels are derived from diverse samples.*

texture metamorphosis: from a pair of samples, an average one is computed and used for the transition area. Zhang *et al.*[ZZV*03] also investigate metamorphosis by weighted blending and texton masks. Nevertheless, interaction on textures and the generation of non-homogeneous textures is still challenging and promises interesting applications.

In our work, we aim at progressing the idea of adding more control over the texture generation process, proposing a framework that offers users many options.

3. Approach

Our approach is based on the inspiring work of Wei & Levoy [WL00]. Their algorithm uses a small sample as input and a random image as initial output that is transformed, a pixel at a time, into a larger desired texture. The color of each pixel is set based on its neighbors color values. This neighborhood is compared with similarly shaped neighborhoods in the sample. The best matching one yields the color for the output position. In this way the algorithm preserves the local similarity between input and output image. Fig. 1 exemplifies the neighborhood comparison and pixel copy. *Per pixel* synthesis has proved to produce continuous outputs and has been applied also in scientific visualization [TA03].

In our method, instead of only having a single input sample, we facilitate a set of samples. These may be modified versions of a single generator, or, more generally, an arbitrary set of at most one sample per output pixel. This means that for each pixel in the output the algorithm chooses a specific input sample, inside which the most similar neighborhood and, thus, the output pixel are determined. The idea is conceptually simple, but allows a powerful and general way to produce a large variety of output textures. It has been partly exploited by Hertzmann[HJO*01]. Our main observation, however, is that the process of neighborhood matching is the key factor to enforce continuity in the output (as long as the input is continuous). We explain our approach in detail in the next section. Fig. 2 illustrates the basic idea (cf.Fig. 1).

3.1. Algorithm

Observing Fig. 2 for an overview of our algorithm, note how the pixels P_i in the texture T_{out} are set executing a query inside corresponding samples T_i. The crucial point is how the input set $T_{in} = \{T_0, T_1, ..., T_i, ..., T_n\}$ is composed.
A special input T_i may be used to set one or more specific output pixels, i.e. $n \leq N$. In the most generic case, $n = N$, in an individual way the user may control every single output pixel P_i, deriving it from an uniquely corresponding sample. Note that the inputs T_i might also have different pixel counts. This approach is illustrated in Fig. 3. Complex filtering and blending produce transformations between diverse textures. The input set may comprise independent miscellaneous samples, and the desired output transits between the various appearances. We see this as part of our future work, as we intend to further investigate a combination of image-morphing techniques with our algorithm.

Formally, let T_{out} be the desired output texture of dimensions $(W_{out} \times H_{out})$, and P_i an output pixel at position (x,y): every P_i corresponds to an array $\underline{a}$ of dimension n:

$$\forall (x,y)_{out} \in T_{out} \Rightarrow \underline{a} \in R^n \tag{1}$$

The dimensionality n depends on the number of parameters (each of them can in turn be an array) that operate on the input set. Thus, the array $\underline{a}$ comprises information that reflects image characteristics and defines the input samples T_i:

$$\underline{a} \Leftrightarrow T_i \tag{2}$$

More specifically, we have correspondence between the current pixel at (x,y) and the particular input texture T_i:

$$\underline{a}|_{(x,y)} \Leftrightarrow T_{i|_{(x,y)}} \tag{3}$$

Now, let all the samples of T_{in} be filtered versions of an original sample T_0, and $\mathcal{T}$ the transfer function of this filter, then

$$T_{i|_{(x,y)}} = T_0 \cdot \mathcal{T}(\underline{a}|_{(x,y)}) \tag{4}$$

where $\mathcal{T}$ gets as arguments the n components of the array

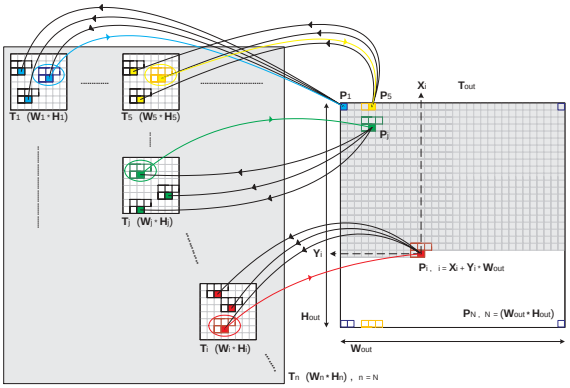

Figure 3: *General approach: the input set is composed by a matrix of samples in correspondence with the output pixels.*

a and may be (see below) a specified operator, or a combination of operators. For a straightforward explanation, a simple case study is described by the results of Fig. 5, where the user intention is to just modify the sample appearance through gradual parameter variation along a specified curve or direction. Consequently, a progressive filter iteratively operates over the input T_0, generating modified versions of it. This guarantees smooth variations and continuous outputs.

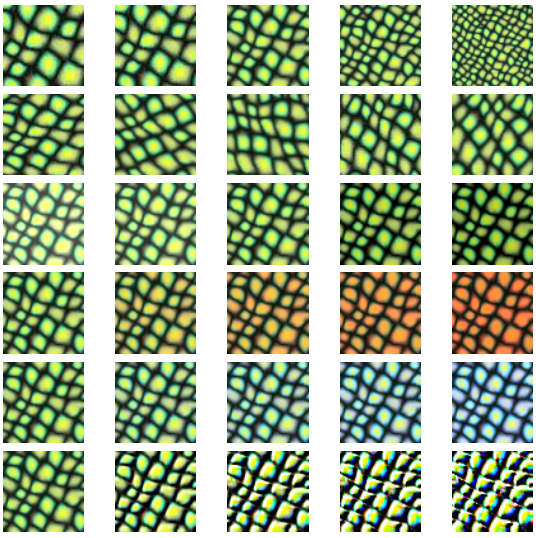

Figure 4: *Scaling, rotating, brightening and darkening, incrementing red and blue component, embossing.*

As our mechanism adds visual effects to each pixel according to user-controlled commands, it is possible to generalize it in a broad way. To modify a starting sample, our system implements image-processing filters, including:

Scaling: to change the resolution of the sample
Rotating: to rotate the sample along specified directions

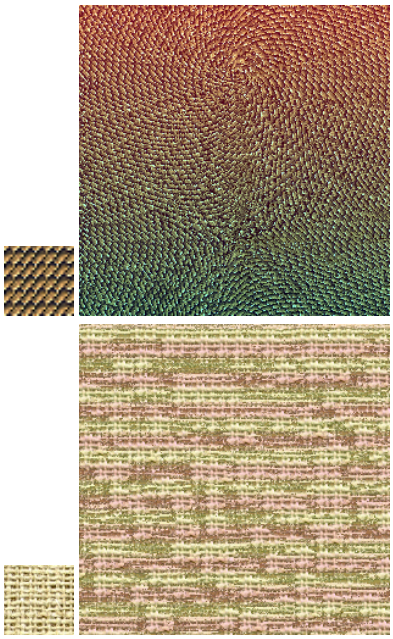

Figure 5: *Variations along y-axis: coloring & changing direction(above); modifying cloth pattern sinusoidally(below).*

Coloring: to control color effects and rgb-transitions
Brightening/Darkening: to modify the sample by increasing/decreasing its luminosity properties
Contrasting: to enhance contrast present in the sample
Embossing/Blurring: to introduce emboss/blur effects

We collected some of the described operations in Fig. 4. The way a filter is then applied to transform the input may be related to several rules, based for instance on:

- the coordinate axis of the output image
- the behavior of a specified function
- the intensity and/or curvature information of a force field
- a mixture of various rules

In general, it is up to the user to choose intuitive and significative mappings. Figures 6 and 7 illustrate several effects.

4. Conclusions

This work proposes a new method for customized texture generation: essentially, the novelty of the approach is a generalization of pixel by pixel texture synthesis that takes into account a mapping from output pixels to input sample. The main observation is that continuity of the output results from the neighborhood matching and, especially, smoothness across the different input samples is not necessary.

The implemented algorithm is intuitive and the system is easy to use: the user is able to freely combine filters and

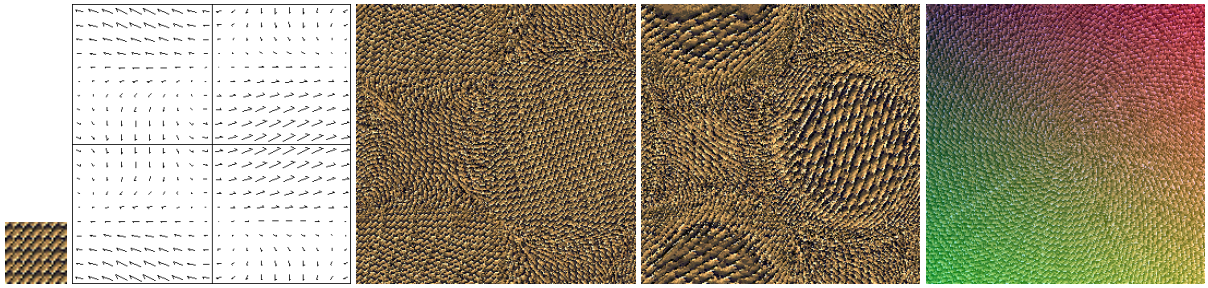

Figure 6: *Controlled Texture Synthesis: the input sample(a) is modified through a superimposed field(b): phase information is used to change the direction of the pattern (c), magnitude information is considered in addition to scale the original pattern (d). In (e), a critical point is visualized: the pattern(a) is modified by rotation and color changes along several directions.*

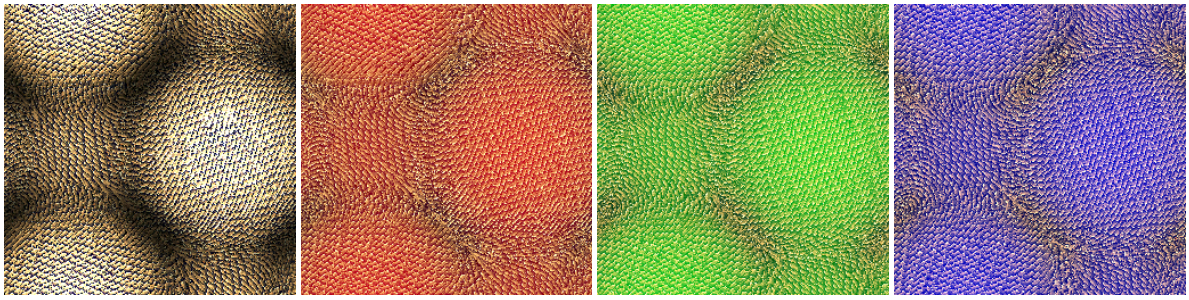

Figure 7: *Further filtering: enhancing brightness (a), increasing red (b), green (c), blue (d) component with amplitude variation.*

parameters yielding easy and flexible control over the generation of the output texture. We implemented the pixel-by-pixel texture synthesis, sped up by the multi-resolution approach based on Gaussian Pyramids. For our examples, we mostly used 16x16 sized samples and the system requires a few minutes to synthesize 256x256 large output textures. As the input sets can be pre-computed or pre-filtered, the processing times are not delayed, regardless of how many variables the user is modifying; consequently, there is no penalty for the added degrees of freedom.

The presented system may be further improved and it would be interesting to combine it with other approaches (*e.g.* [WL00], [ZZV*03]) that optimize texture metamorphosis.

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *2001 ACM Symposium on Interactive 3D Graphics* (2001), pp. 217–226. 1

[DMLG02] DISCHLER J., MARITAUD K., LÉVY B., GHAZANFARPOUR D.: Texture particles. *Computer Graphics Forum 21*, 3 (2002), 401–410. 1

[EL99] EFROS A., LEUNG T.: Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision* (1999), pp. 1033–8. 1

[GS] GORLA G. I. V., SAPIRO G.: Growing fitted textures. In *SIGGRAPH 2001, Appl. and Sketches*, ACM. 1

[HB95] HEEGER D. J., BERGEN J. R.: Pyramid-Based texture analysis/synthesis. In *SIGGRAPH* (1995). 1

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *SIGGRAPH 2001* (2001), pp. 327–340. 1, 2

[SCA02] SOLER C., CANI M.-P., ANGELIDIS A.: Hierarchical pattern mapping. In *Proceedings of SIGGRAPH* (2002), pp. 673–680. 1

[TA03] TAPONECCO F., ALEXA M.: Vector field visualization using markov random field texture synthesis. In *IEEE TCVG Visualisation Symposium* (2003), ACM Press. 2

[Tur01] TURK G.: Texture synthesis on surfaces. In *Proceedings of ACM SIGGRAPH 2001* (2001), ACM Press, pp. 347–354. ISBN 1-58113-292-1. 1

[Wei03] WEI L.-Y.: Texture synthesis from multiple sources. In *SIGGRAPH 2003, Applications and Sketches* (2003), ACM Press. 1

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH* (2000), pp. 479–488. 1, 2, 4

[WL01] WEI L.-Y., LEVOY M.: Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH* (2001). 1

[YHBZ01] YING L., HERTZMANN A., BIERMANN H., ZORIN D.: Texture and shape synthesis on surfaces. In *EG Workshop on Rendering* (2001), pp. 301–312. 1

[ZZV*03] ZHANG J., ZHOU K., VELHO L., GUO B., SHUM H.-Y.: Synthesis of progressively variant textures on arbitrary surfaces. *ACM TOG* (2003). 2, 4