# Parallel Composite Texture Synthesis

Alexey ZALESNY, Vittorio FERRARI, Geert CAENEN, and Luc VAN GOOL

***Abstract*--Many textures require complex models to describe their intricate structures. Their modeling can be simplified if they are considered composites of simpler subtextures. After an initial, unsupervised segmentation of the composite texture into the subtextures, it can be described at two levels. One is a label map texture, which captures the layout of the different subtextures. The other consists of the different subtextures. This scheme has to be refined to also include mutual influences between textures, mainly found near their boundaries. The proposed composite texture model also includes these. The paper describes an improved implementation of this idea. Whereas in a previous implementation subtextures and their interactions were synthesized sequentially, this paper proposes a parallel implementation, which yields better results with simpler models.**

***Index Terms*--texture synthesis, texture analysis, statistical texture modeling, composite textures, hierarchical texture model.**

## I. INTRODUCTION

THE intricate nature of many textures makes it difficult to extract models that are compact and that support high quality synthesis. Often, the problem can be reduced by considering the texture as a composite of simpler subtextures. We propose such hierarchical approach to texture synthesis. We show that this approach can be used to synthesize intricate textures and even complete scenes, and that it also improves the results for "simple" textures. This suggests that hierarchical approaches to texture synthesis hold good promise as a general principle.

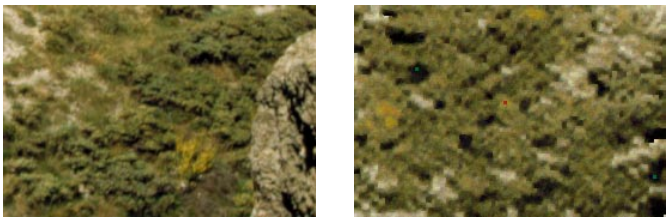
Before explaining the composite texture algorithm, we concisely describe our basic texture model for single textures, in order to make this paper more self-contained. The point of departure of the basic model is the co-occurrence principle. Simple statistics about the colors at pixel pairs are extracted, where the pixels take on carefully selected, relative positions. It differs in this selectivity from more broad-brush co-occurrence methods [3], [4]. Every different type of pair – i.e. every different relative position – is referred at as a *clique type*. The statistics gathered for these cliques are the histograms of the intensity differences between the head and tail pixels of the pairs, and this for all three color bands R, G, and B. Hence, the basic model consists of a selection of cliques (the *neighborhood system*) and their color statistics (the *statistical parameter set*). A more detailed explanation about these basic models and how they are used for texture synthesis is given in [8].

Alexey Zalesny, Vittorio Ferrari, and Luc Van Gool are with the Swiss Federal Institute of Technology Zurich, Switzerland;
{zalesny, ferrari, vangool}@vision.ee.ethz.ch
http://www.vision.ee.ethz.ch/~zales; +41 1 632{5724, 7685, 6578}.

Geert Caenen and Luc Van Gool are with the Catholic University of Leuven, Belgium; {Geert.Caenen, Luc.VanGool}@esat.kuleuven.ac.be
+32 16 32{1079, 1705}

## II. COMPOSITE TEXTURE MODELS

Fig. 1 shows a texture and a synthetic version that was generated using its basic model. The result is of low quality.



(a) original (b) synthesis from basic model

Fig. 1. The image on the left shows a complex landscape texture; the image on the right shows the result of attempting to synthesize similar texture from its basic model, that considers the original as one, single texture.

The example shows that at least in some cases a more sophisticated texture model is needed. As mentioned, the idea explored in this paper is that a prior decomposition of such textures into their subtextures (e.g. grass, sand, bush, rock, etc. in the example image) is useful. The layout of these subtextures (the *label map*) can be considered to be a texture in its own right, which can be modeled using an approach similar to the basic modeling scheme and of which more can then be synthesized. Also the subtextures can be modeled using the basic model. If not, a second decomposition could be used, but we have not explored such recursive decompositions at this point. Texture synthesis then amounts to first synthesizing a label map, and then synthesizing the subtextures at the corresponding places. A similar idea has been propounded independently by Hertzmann et al. [5], but their "texture by numbers" scheme (based on smart copying from the example [2], [6]) did not include the automated extraction or synthesis of the label maps (they were hand-drawn).

A crucial point is how to perform the decomposition, i.e. how to segment the texture into its subtextures. We have devised an unsupervised segmentation scheme, which calculates pixel similarity scores on the basis of color and local image structure and which uses these to group pixels through efficient clique partitioning. This segmentation procedure is explained in a companion paper [1].

This paper focuses on the construction of the composite texture model, on the basis of an example image and its segmentation. A straightforward implementation would model

the subtextures separately. In reality, however, subtextures will not be stationary within their patch boundaries. Typically there are natural processes at work (geological, biological, …) that cause interactions between the subtextures. There are transition zones around some of the subtexture boundaries. Fig. 2 illustrates such transition effect. The image on the left is an original image of zebra fur. The image in the middle is the result of taking the left's image label map (consisting of the black and white stripes) and filling in the black and white subtextures. The boundaries between the two look unnatural. The image on the right has been synthesized taking the subtexture interactions into account, using the algorithm proposed in this paper. The texture looks much better now.

(a) original (b) subtexture synthesis without interactions (c) subtexture synthesis with interactions

Fig. 2. Composite texture synthesis of zebra fur with and without subtexture interactions demonstrates the importance of the latter.

In [7] we have proposed a scheme that orders the subtextures by complexity, and then embarks on a *sequential* synthesis, starting with the simplest. Only interactions with subtextures that have been synthesized already are taking into account. In this paper we propose an alternative, *parallel* approach, where all subtextures and their interactions are taken care of simultaneously, both during modeling and synthesis. The sequential aspect that remains, is that first the label texture is synthesized and only then the subtextures.

The main advantage of the parallel approach is that all bi-directional, pairwise subtexture interactions can be taken into account. This, in general, results in better quality and a more compact model. Additionally, there is a disturbing asymmetry between the model extraction and subsequent texture synthesis procedures with the sequential approach. During modeling the surrounding subtextures are ideal, i.e. taken from the reference image. This is not the case during the synthesis stage, where the sequential method has to build further on the basis of previously synthesized subtextures. There is a risk that the sequentially generated subtextures will be of lower and lower quality, due to error accumulation. The parallel approach, in contrast, is free from these drawbacks, as both the modeling and synthesis stages operate under similar conditions. The advantage of the sequential modeling step (but not the synthesis one!) is that every subtexture can be modeled simultaneously, distributed over different computers. But as speed is more crucial during synthesis, this advantage is limited in practice.

During model extraction, the foremost problem is the clique type selection. This problem is more complicated in the parallel case, as there are many more clique types to choose from. At every iteration of the modeling algorithm – to be described shortly – a choice can be made between all *inter-* and *intra-label cliques*. Intra-label cliques have their head and tail pixels within the same subtexture, whereas inter-label cliques have their heads and tails in different subtextures. The inter-label cliques are needed to model the interactions between the subtextures.

The parallel composite texture modeling takes the following steps:

1. Segment the example image of the composite texture. The image and the resulting label map are the input for the modeling procedure. Let $K$ be the number of subtextures and $B$ – the number of image color bands.
2. Calculate the intensity difference histograms for all inter-label and intra-label clique types that occur in the example image, up to a maximal, user-specified head-tail distance (the *clique length*). They will be referred at as reference histograms. After this step the example image is no longer needed.
3. Construct an initial composite texture model containing the $K\times B$ intensity histograms for each subtexture and each color band and $2K\times B$ clique types and their statistics: for each subtexture/band the shortest horizontal and shortest vertical cliques are added to the model.

***Loop*:**

4. Synthesize a texture using the input label map and the current composite texture model, as discussed further on.
5. Calculate the intensity difference histograms for all inter-label and intra-label clique types from the image synthesized in step 4, up to a maximal, user-specified clique length. They will be referred to as *current histograms*.
6. Measure the histogram distances – a weighted (see below) Euclidean distance between the reference and current histograms.
7. If the maximal histogram distance is less than a threshold go to the step 9.
8. Add the following $2K$ histograms to the composite model: (a) $K$ intra-label ones, for each subtexture the one with the largest histogram distance; (b) $K$ inter-label ones, those with the largest histogram distance for pairs of subtextures $(k,n)$ for all $n$ and one fixed $k$ at a time.

***Loop end.***

9. Model the label map as a normal non-composite texture (i.e. using the basic model), except that instead of intensity difference histograms the label co-occurrences are used.

***Stop.***

The label map generation is driven by label co-occurrences and not differences because the latter are meaningless in that case. Also, there are only a few labels in a typical segmentation and, hence, taking the full co-occurrence matrix rather than only difference histograms into account comes at an affordable cost.

We now concisely describe how texture synthesis is carried out, and how histogram distances are weighted.

### *A. Texture synthesis*

Texture synthesis is organized as an iterative procedure that generates an image sequence, where histogram distances for the clique types in the model decrease with respect to the

corresponding reference histograms. This evolution is based on non-stationary, stochastic relaxation, underpinned by Markov Random Field theory. Non-stationary means that the control parameters of the synthesizer (in our case these are so-called Gibbs parameters of the random field) are changed based on the comparison of the reference and current histograms [8].

A separate note on the synthesis of subtexture interactions is in order here. Even if the modeling procedure selects quite a few inter-label clique types, they still represent a very sparse sample from all possible such clique types, as there are of the order of $K^2$ subtexture pairs, as opposed to $K$ subtextures, for which just as many clique types were selected. Thus, many subtexture pairs do not interact according to the composite texture model (i.e., there are no cliques in the model corresponding to the label pair under consideration). For such pairs, subtexture knitting – a predefined type of interaction – is used. During knitting neighboring pixels outside the subtexture's area are nevertheless treated as if they lay within, and this for all clique types of that subtexture. The intensity difference is calculated and its entry in the histogram for the *given* subtexture is used. Knitting produces smooth transitions between subtextures. In case clique types describing the interaction between a subtexture pair have been included in the model, their statistical data are used instead and knitting is turned off for that pair.

During normal synthesis, the composite texture model is available from the start and all subtexture pairs without modeled interactions are known beforehand. Hence, knitting is always applied to the same subtexture pairs, i.e., it is static. During the texture modeling stage the set of selected clique types will be constantly updated and the knitting is adaptive. Knitting will be automatically switched off for subtexture pairs with a selected inter-label clique type. This will also happen for subtexture pairs that do not interact, e.g. a texture that simply occludes another one. This is because during the composite texture modeling process knitting is turned on for all pairs which are left without an inter-label clique type. Knitting will not give good results for independent pairs though, as it blends the textures near their border. Hence, a clique type will be selected for such pairs, as the statistics near the border are being driven away from reality under the influence of knitting. The selection of this clique type turns off further knitting, and will itself prescribe statistics that are in line with the subtextures' independence. This process may not be very elegant in the case of independent subtextures, but it works.

### B. Histogram distances

The texture modeling algorithm heavily relies on histogram distances. These are weighted Euclidean distances, where the weight is calculated as follows:

$$\mathrm{weight}(k,k,type) = \frac{N(k,k,type)}{N_{\max}}, \qquad (1)$$

where the clique count $N(\cdot)$ is the number of cliques of this type having both the head and the tail inside the label class $k$, and $N_{\max}$ is the maximum clique count reached over all clique types. The rationale behind this weighing is that types with low clique counts must not dominate the model, as the corresponding statistical relevance will be wanting. This weight also reduces the influence of long cliques types, which tend to have lower clique counts.

For the inter-label cliques, this effect is achieved by making the weights dependent on clique length explicitly:

$$\mathrm{weight}(k,n,type) = \left(1 - \frac{l^2(type)}{(l_{\max}+1)^2}\right)\frac{N_{\max}(k,n)}{N_{\max}}, \qquad (2)$$

where $N_{\max}(k,n)$ is the maximal clique count among the types for the given subtexture pair, $l(type)$ is the clique length, and $l_{\max}$ is the maximal clique length taken over all types present in the example texture. Such weighing again increases the statistical stability and gives preference to shorter cliques, which seems natural as the mutual influence of the subtextures can be expected to be stronger near their boundary.

## III. Experimental Results

This section shows some of our experimental results with the parallel composite texture synthesis. Fig. 3 shows three images. From left to right one has an example image of cloth, the result of segmentation, and synthetic cloth, using the label map in the middle. Fig. 4 and Fig. 5 show similar results for a limestone texture and a complete landscape, respectively. In both figures the one on the top is the original image, and the one at the bottom is synthetic. These examples demonstrate that the method is able to capture most of the subtleties of natural textures.

Fig. 3. Parallel composite texture synthesis. Left to right: original image of cloth with four different knitted textures, segmentation map, synthetic image.

Fig. 6 shows three images of a landscape. The one on the top is the original image. A label map was segmented out, and treated as a texture in order to generate similar landscape layouts. The image in the middle shows the result of our previous, sequential texture synthesis method applied to one of these layouts (synthetic label maps). The image on the bottom shows the same experiment, but now with textures synthesized by the parallel approach described in this paper. The overall result looks better. In particular, some unnaturally sharp transitions between the bush and grass subtextures have been eliminated.

Fig. 7 shows another example where texture has been synthesized on the basis of a synthetic label map. The zebra texture has been synthesized completely automatically from the example of Fig. 2 (a). Fig. 8 shows a result for the texture of Fig. 3 left. This synthetic cloth looks awkward to the human eye, as it doesn't respect the horizontal layout.

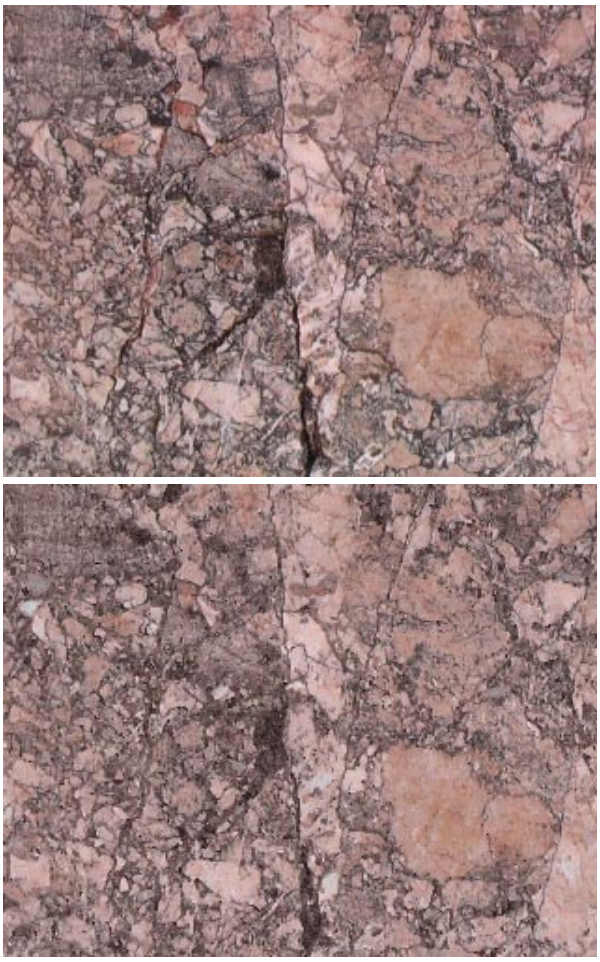


Fig. 4. Parallel composite texture synthesis. Top: original image of limestone, bottom: image synthesized on the segmentation map.

Fig. 5. Parallel composite texture synthesis. Top: original aerial image of landscape, bottom: image synthesized on the segmentation map.

Fig. 6. Landscape texture synthesis. Top to bottom: original image with three different textures, sequential scheme of synthesis, parallel composite synthesis with better, more natural texture transitions.

Fig. 7. Synthetic zebra fur on the synthetic layout label map (cf. Fig. 2).

Fig. 8. Synthetic cloth on the synthetic layout label map (cf. Fig. 3).

Finally, Fig. 9 illustrates that the composite texture approach also holds good promise for "single" textures. The texture on the left is the example. The one in the middle has been generated on the basis of its basic model, which included 59 clique types. The image on the right is the result of a composite texture synthesis, where the model was limited to its first 59 cliques. Not only is this result better, it also took only 100 iterations instead of the 500 needed to get the result with the basic model. Also, despite the same overall clique type number, the computational complexity of the parallel approach is lower, because every pixel has about a half of this number belonging only to one of the two subtexture models and the subtexture interaction model.

(a) original (b) synthesis as a single texture (c) synthesis as a composite texture

Fig. 9. Also patterns that traditionally would be considered to be a single texture can benefit from the composite texture approach.

## IV. Conclusion

We have described a hierarchical texture synthesis approach, that considers textures as composites of simpler subtextures, that are studied in terms of their own statistics, that of their interactions, and that of their layout. The approach supports the fully automated synthesis of complex textures from example images, without verbatim copying.

## Acknowledgment

The authors gratefully acknowledge support by the European IST project "3D-Murale" and "CogViSys".

## References

[1] Geert Caenen, Vittorio Ferrari, Alexey Zalesny, and Luc Van Gool. Analysing the Layout of Composite Textures. Submitted to Texture 2002.

[2] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. *Proc. Int. Conf. Computer Vision (ICCV'99)*, Vol. 2, 1999, pp. 1033-1038.

[3] A. Gagalowicz and S.D. Ma. Sequential Synthesis of Natural Textures. *Computer Vision, Graphics, and Image Processing*, Vol. 30, 1985, pp. 289-315.

[4] G. Gimel'farb. *Image Textures and Gibbs Random Fields.* Kluwer Academic Publishers: Dordrecht, 1999, 250 p.

[5] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. *ACM SIGGRAPH 2001*, pp.327-340, 2001.

[6] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. *SIGGRAPH 00*, pp.479-488.

[7] Alexey Zalesny, Vittorio Ferrari, Geert Caenen, Dominik Auf der Maur, and Luc Van Gool. Composite Texture Descriptions. Submitted to ECCV 2002.

[8] A. Zalesny and L. Van Gool. A Compact Model for Viewpoint Dependent Texture Synthesis. *SMILE 2000, Workshop on 3D Structure from Images, Lecture Notes in Computer Science 2018*, M. Pollefeys et al. (Eds.), 2001, pp. 124-143.