Pacific Graphics 2010
P. Alliez, K. Bala, and K. Zhou
(Guest Editors)

Volume 29 (2010), Number 7

Image Synthesis for Branching Structures

Dominik Sibbing Darko Pavić Leif Kobbelt

RWTH Aachen University

Abstract

We present a set of techniques for the synthesis of artificial images that depict branching structures like rivers, cracks, lightning, mountain ranges, or blood vessels. The central idea is to build a statistical model that captures the characteristic bending and branching structure from example images. Then a new skeleton structure is synthesized and the final output image is composed from image fragments of the original input images. The synthesis part of our algorithm runs mostly automatic but it optionally allows the user to control the process in order to achieve a specific result. The combination of the statistical bending and branching model with sophisticated fragment-based image synthesis corresponds to a multi-resolution decomposition of the underlying branching structure into the low frequency behavior (captured by the statistical model) and the high frequency detail (captured by the image detail in the fragments). This approach allows for the synthesis of realistic branching structures, while at the same time preserving important textural details from the original image.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

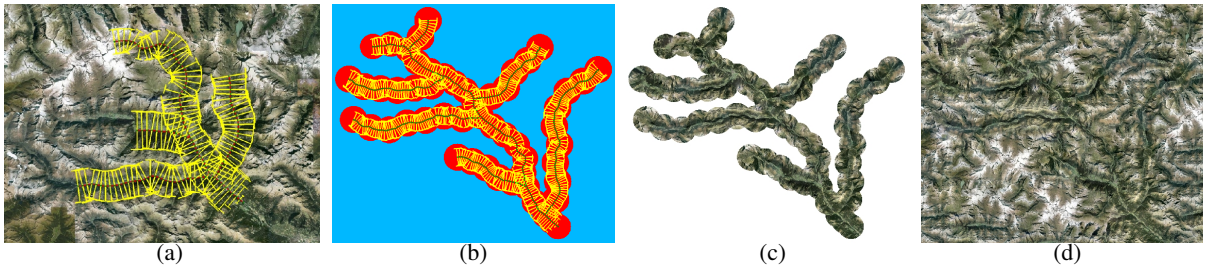
1. Introduction

The generation of images is one of the major activities of human creativity. However, with the existence of large image collections on the internet the necessity to generate new images and illustrations from scratch becomes less and less important since for most objects and scenes some image material is already available. Hence, techniques are coming into focus that support image synthesis by decomposing and recombining existing material. Such techniques find more and more applicability in movies or computer games where the repetition of textures appears unnatural.

In this paper we are presenting an approach to generate images that contain branching structures like rivers, cracks, lightnings, mountain ranges, or blood vessels. Our algorithm is able to geometrically synthesize similar structures as seen in the input image. This can be done automatically by the system (using parameters learned form the input image) but it also provides some control mechanisms to the user to guide the structure synthesis. Hence the automatic generation of multiple similar images, as well as the creation of individual designs guided by a person, becomes possible. Our algorithms for image analysis and synthesis are integrated into a three stage interactive workflow that provides various semi-automatic selection tools in order to effectively control the process.

In the first step an input image is chosen. From this image, the user extracts the branching structures (Sect. 3). The structure is analyzed in the second step in order to derive a statistical model which captures the characteristic style of the branching structure. This model is applied to synthesize new abstract branching structures, *i.e.* polygonal skeletons with associated thickness coefficients (Sect. 4). Technically our model describes a Markov process where new extending segments are selected by matching sub-sequences of segments in the input structure. The selection probabilities are adapted such that locally similar segments and segments which satisfy some global characteristics are more likely to be chosen (Fig. 1). In the final step the output image is generated by a fragment-based image completion technique which first copies matching fragments from the input image to cover the region where the branching structure passes through and then fills the regions in between the branches (Sect. 5). In order to produce high quality images we employ Graph Cuts to optimize seams between fragments, Poisson image editing to reduce the gradient across the seams, and image warping techniques to avoid discontinuities along the polygonal skeletons.

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.



(a) (b) (c) (d)

Figure 1: *Based on the input image and its structure information (a) our system learns a statistical structure model from the given example. It is then able to synthesize images with visually similar looking branching structures (d). The overall process is divided into the geometric structure reconstruction (b) and the fragment-based image synthesis (c).*

2. Related Work

Curve synthesis One of the most prominent methods to generate tree-like structures is to use L-Systems which were introduced by Lindenmayer [Lin68]. In L-Systems a grammar is defined and used to recursively replace modules under certain conditions. These systems are very suitable for tree structures and were, *e.g.*, used to synthesize plants [PH89]. A stochastic model can be added to those systems but the underlying rules of an L-System are designed only for a certain application (*e.g.*, generating plants or trees). Palubicki *et al.* [PHL*09] use an L-System to simulate the growth of self-organizing trees. The creation of a tree is controlled by a set of well defined rules and includes a guidance model which directs the plant away from shadowed regions. In contrast our method does not rely on a predefined set of rules. It rather deduces the necessary parameters to recover a specific structure from an example.

Hertzmann *et al.* [HOCS02] proposed an elegant method to map a transformation given for a source curve to a target curve. For this, an initial target curve is required as input which is then used as a guidance for the synthesis process. This concept of Curve Analogies was *e.g.* used by Wu *et al.* [WZSB08] to edit trajectories in motion data. Our model, together with the space colonization, is able to synthesize arbitrary curves (of varying thickness) automatically without any given target curve, while optionally the user is still allowed to include some degree of guidance by adapting the underlying density field as described in Sect. 4.5. Since we uniformly resample the curves it is possible to define a genuinely scale-free similarity measure between curves by just comparing angles instead of evaluating a positional distance function including least-squares-optimized translation and rotation parameters for the source structure to align with the target structure [HOCS02]. In addition our model includes a branching model which allows for a larger variety of synthesized structures and hence is quite different.

Kim and Lin [KL04] present a specialized framework based on a simplified Helmholz equation to compute the spread of electric charge along a 3D grid. Using a special renderer based on the Atmospheric Point Spread Function they produce impressive images of lightnings. Chen *et al.* [CEW*08] proposed a specialized framework for urban modeling. The user defines a set of directions from which an underlying tensor field is computed which guides the generation of a street graph. Aliaga *et al.* [AVB08] proposed an interactive method to generate images of urban layouts. From exemplary images they extract a street graph and synthesize a new city showing similar street characteristics. In this setting the streets work as separators between image patches such that seams between patches are never an issue. Similar to [CEW*08] these systems focus on a very specialized purpose of rendering urban layouts or physical phenomena, while our method focuses on tree-like structures in general. Tan *et al.* [TZW*07] present a semi-automatic method to generate botanic trees from images. They reconstruct visible 3D branches from 3D points obtained from structure from motion, extend them assuming branches to be self-similar and add leaves to produce 3D models of botanic trees.

Image Synthesis In general texture synthesis aims at creating arbitrary large textures from small input examples. Efros and Leung [EL99] use a Markov random field to compute new colors for single pixels by comparing a small already synthesized local neighborhood with parts of the source image. Ashikhmin [Ash01] took a user defined image to guide the synthesis process and exploit the coherence of neighboring source pixels. In [HJO*01] a labeling of image regions is performed by the user. From another user-defined image, showing the same labels, a similar image can be synthesized by filling regions with pixels of corresponding labels. The resulting images are convincing, but compared to our system which is also able to produce variants of the underlying structure via a statistical model, this method is less flexible, since the user has to provide the afore mentioned labeled regions.

In order to fill large regions one can also employ patch based methods which copy whole fragments instead of pixels of a source image. Brooks and Dodgson [BD02] proposed an approach to generate self-similar textures, which works well for unstructured textures like a field of flowers, unstructured rifts, sand or stones. Fragment based methods often have problems blending to adjacent fragments in order to avoid seams. Kwatra *et al.* [KSE*03] introduced a nice idea to adapt the path of a seam such that

differences in color values of neighboring pixels across the seam are minimized. In our image synthesis we also employ their method of Graph Cuts to produce visually pleasing results. Also very sophisticated methods to generate large (infinite) textures from compact input samples are proposed in [LH05, LH06, KEBK05], where repetition artifacts and seams are nearly invisible. All pixel and fragment based algorithms require rather unstructured input images. Since we are able to synthesize the geometric content of an image by generating a branching curve skeleton, we are focusing on the synthesis of a completely different category of images. Zhou *et al*. [ZSTR07] proposed a method to generate height maps for terrain rendering from exemplary height fields and a user-defined sketch. They detect significant features in the height field and transfer surrounding image content to similar features of the sketch. Combining these specialized textures, which are height fields, with a well designed renderer they produce impressive images of terrains. Compared to our work this method needs a user-defined sketch in order to identify corresponding features for the synthesis of the underlying geometry. Although we allow user guidance, we are also able to synthesize new (more arbitrary) structures in a completely different, learning based, approach. Rosenberger *et al*. [RCOL09] compute different layers of a texture in order to create a control map for the texture synthesis. They focus on computing 2D layers whose boundaries show similarity to boundaries of the input layers. These layers are used as a control map to actually create a final texture. In our work we do not consider inhomogeneous textures showing layered content (like peeling paint on bare metal with areas of rust) for which their algorithm is well suited. Our focus is on branching structures and as such it is a very different problem instance.For a more thorough survey and detailed explanations of example based texture synthesis we would like to refer the interested reader to [WLKT09, KW07].

Fragment based image completion techniques [TCLT07, HE07] fill missing regions by copying whole fragments from a source to a target image position. The original approach [DCOY03] creates impressive results, but is rather slow and also has problems to correctly synthesize structures shown in the image. In [SYJS05] the authors consider structures to be user defined. They distinguish between the structure and texture in the images and apply the completion task in two steps. First, they use belief propagation to propagate image structures along the user defined paths. Then, for the remaining missing parts a fragment-based completion approach is applied [TCPT03]. We use similar ideas, but in our method we can synthesize structures via a statistical model. In contrast to [SYJS05] we apply a greedy variant of structure propagation, which is more suited for the synthesis of tree-like structures, which starts with an empty image. For the remaining missing parts in the image we apply the flexible interactive image completion approach of [PSK06], but this is surely exchangeable by some standard technique reproducing rather homogeneous image content [WLKT09].

Contributions

- Our similarity measurement between curves compares angles and thickness and is a scale-free measure that does not include computationally involved least-squares-optimization of translation and rotation parameters.
- We achieve local as well as global control over the synthesis procedure by modeling local similarity as a Markov process and computing global statistics based on histograms.
- Our branching model is automatically learned for different structures via support vector machines.
- We incorporate advanced image completion techniques into our synthesis: Graph Cuts for seam optimization, Poisson image editing to adapt color values across seams, and warping techniques to avoid discontinuities along structures.
- Our system supports *optional* user interaction during the synthesis, which makes the system a powerful sketch tool for the recombination of existing image material.

3. Generating Input Structures

When describing the characteristic style of a curve-like feature in an image we have to distinguish three different levels of detail (scales). On the coarsest scale the univariate feature follows some global path across the image. On the next finer scale we can identify a local bending behavior which is characteristic for a given phenomenon. For example a lightning has a more jaggy appearance than a river. On the finest scale we can identify small detail on texture structures, *e.g*., houses on the river banks.

The goal of this paper is to synthesize new images with branching structures from examples by separating these three levels of detail. For the newly generated images we define a new shape on the coarsest scale only, *i.e*. we define new paths which should be followed by the branching structure. The characteristic bending and furcation as well as the fine detail from the input, however, should be preserved as faithfully as possible.

In order to achieve this we use a statistical geometric structure model for the mid-scale shape which controls the characteristic bending behavior. The finest level detail is preserved by composing the final output from texture fragments of the input image.

The decision which geometric feature magnitude (kilometer, meters, centimeters) belong to which of the three levels of detail depends very much on the image content and on the specific application. Hence it is not a reasonable approach to try to perform this decomposition fully automatic. We rather let the user perform this decomposition by manually producing the example data which then serves as the input to the structure analysis and synthesis stages. This manual process includes the generation of a polygonal skeleton $[\mathbf{x}_0, \ldots, \mathbf{x}_n]$ for each branch of the structure plus adding a branch thickness r_i to the vertices x_i of the skeleton.

Our interactive system conveniently supports this manual process. After loading an input image, the user simply selects points $\mathbf{x}_i$ along the branches of the structure. In the example of Fig. 1 we selected 65 points which are further refined (Fig. 1a) by the system (Sect. 4.1). Here the precision requirements are not very high. A simple maximum gradient detection operator in the direction perpendicular to the branch orientation provides some initial estimate for the local branch thickness which can be adjusted by the user. After a few minutes of interaction, the branching structure is captured and can be passed to the subsequent, automatic processing stages.

4. Statistical Structure Synthesis

The look and style of a branching structure is determined by a mathematical model for the characteristic bending behavior as well as for the number and distribution of furcations. In order to achieve maximum flexibility, we handle both in separate statistical models that are specifically designed to support the extension of existing branches (growing) and to decide where new branches should start. The models are learned from examples, *i.e.* we expect a polygonal branching structure as input and extract the models from it. In the synthesis phase we can then use these models to generate new branching structures showing the same characteristic behavior as the input.

The synthesis starts by generating one non-furcated main branch, where one, *e.g.*, can initialize the main stem by copying the first elements from the input structure. Then a set of furcation points is determined and at each such point, a new branch is grown. This process is recursively repeated until a stopping criterion is meet, *e.g.*, maximum levels of branch generations, maximum total branch length, or maximum spatial density of branches (see Sect. 4.4).

4.1. Input Structure

A single branch is given by a polygonal skeleton with vertices $[\mathbf{x}_0,\dots,\mathbf{x}_n]$. For each vertex $\mathbf{x}_i$, we add the local thickness r_i as an attribute. In order to obtain an orientation independent representation, we use the angles α_i between successive edges of the polygonal skeleton as an additional attribute. For the sake of simplicity we assume that the polygonal skeleton has been uniformly resampled such that the edge lengths $\|\mathbf{x}_{i+1}-\mathbf{x}_i\|$ are constant. This implies that the angles α_i and thicknesses r_i together with the constant edge length h uniquely and completely define the geometry of a branch up to rigid transformations.

The different (sub-)branches of the input structure are stored as separate edge sequences with the additional information from which vertex in the parent they branch off.

4.2. Bending Model

Markov Model. We model the characteristic bending of a branch by a Markov process [Nor98], *i.e.* when growing a branch by adding the next vector of m segments $(\alpha_{n+1,m},\mathbf{r}_{n+1,m}) = (\alpha_{n+1},r_{n+1}),\dots,(\alpha_{n+m},r_{n+m})$ we consider only the finite history of the $k+1$ previous angles and thicknesses $(\alpha_{n-k,k+1},\mathbf{r}_{n-k,k+1}) = (\alpha_{n-k},r_{n-k}),\dots,(\alpha_n,r_n)$, see Fig. 2. By putting both, the angle and the thickness, into one model we can reflect the observation that fine branches often bend differently from the thicker main branches.

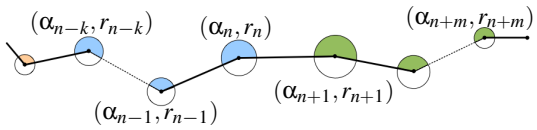


Figure 2: *Illustration of the model parameters. The previous neighborhood depends on the parameter k. The nodes in this neighborhood are colored in blue while the next m nodes, which will be appended, are displayed in green.*

More concretely, we take the angle and thickness history $(\alpha_{n-k,k+1},\mathbf{r}_{n-k,k+1})$ and find the best matching c candidates among all sub-sequences $C_i = (\hat{\alpha}_{i,k+1},\hat{\mathbf{r}}_{i,k+1})$ in the input structure. Here, the quality Q_i of a candidate C_i is measured scale-invariantly by the distance

$$Q_i \;=\; dist(\alpha_{n-k,k+1},\hat{\alpha}_{i,k+1})\cdot dist(\mathbf{r}_{n-k,k+1},\hat{\mathbf{r}}_{i,k+1})$$

where we define

$$dist(\mathbf{X},\mathbf{Y}) \;=\; \frac{1}{\|\mathbf{X}-\mathbf{Y}\|+1}+1$$

such that the quality score always remains >1 and the singularity for $X=Y$ is avoided. Then we randomly select one of the c candidates with highest quality with a probability proportional to their quality scores and the m elements following it are appended to the output branch. The parameters that control the behavior of this model are m, k and c. The parameter m influences the size of the extension that is added in each step and hence the size of what is considered a "typical" feature. If the target feature size is s and the uniform resampling step width is h we set $m = s/h$. The parameter k defines the order of the Markov process and is set to $k = 3m$ in all of our experiments, because it consistently led to most plausible results. The number of candidates c finally controls the variability of the output. For $c = 1$ we deterministically extend the branch with the best matching sub-sequence. By default, we set this parameter to $c = 2$ in our experiments. Fig. 3 illustrates the effect of these parameters in a simple example setting.

Histogram matching. In order to additionally reproduce some global characteristics of the structures we employ histogram matching similar to [KFCO*07]. We adapt the quality scores Q_i such that histograms of angles and thickness of the synthesized match the input structure. For the input structure we calculate normalized histograms $\hat{H}_\alpha$ and $\hat{H}_r$, which reflect its angle and thickness distribution, at the beginning of the synthesis process. Analogously the histograms of the target structure before extending are defined as H_α and H_r,

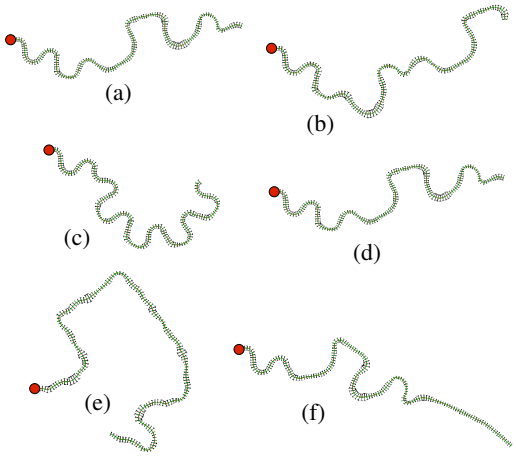


Figure 3: *In this figure we demonstrate the different effects of choosing the size of the neighborhood k, the size of the extension m and the variability c. (a) source structure. (b) k and m are very high but we allowed more variability ($c = 2$). The result is a slight deviation from the source structure. In (c) and (d) we omitted the variability ($c = 1$) and chose a large look-ahead ($m = 15$). In (c) we only considered a small value for $k = 2$ which results in a repeating pattern of the same structure. In (d) we chose k to be large which results in reproducing the source structure exactly. In (e) and (f) we set the size of the extension to be small ($m = 2$) and allowed some variability ($c = 2$). When little is known about the past ($k = 2$) the produced structure looks not at all like the source structure (e). The effect of increasing the size of the neighborhood k with small m is finally shown in (f).*

whereas H^i_α and H^i_r denote the prospective histograms after adding a candidate C_i. In all our experiments each histogram splits the interval between the minimal and maximal value observed in the input structure into $B = 16$ bins.

To achieve global similarity we adapt the quality scores according to

$$Q^*_i = Q_i \cdot \; e^{\,\omega_G\,(\Delta_\alpha(i)+\Delta_r(i))\cdot\frac{n_t}{m}}$$

where $\Delta_x(i) = \frac{1}{B}\|\hat{H}_x - H_x\|^2 - \frac{1}{B}\|\hat{H}_x - H^i_x\|^2$ measures the improvement of the histogram similarity, if the the candidate C_i would be added. The factor $\frac{n_t}{m}$, n_t being the number of nodes of the structure synthesized so far, compensates for the decreasing influence of the m new nodes on the histogram as the number n_t of existing nodes grows. Here ω_G, which we set to 50 by default, is a weight which can be used to control the influence of the histogram matching. In this way the score of a candidate is increased or decreased, depending on the amount of improvement or impairment of the global similarity.

4.3. Furcation Model

For the furcation model, we need to distinguish between the decision *where* to place a furcation and *how* this furcation should look like. Both decisions should, again, depend on the local shape of the branch such that the characteristic structure of the input is preserved.

To determine the orientation and bending behavior of the child branch being attached at some vertex $\mathbf{x}_i$ of the parent branch, we can use a similar Markov process as in Sect. 4.2 with the added feature that we can now even use a non-causal neighborhood $(\alpha_{i-k,2k+1}, \mathbf{r}_{i-k,2k+1}) = (\alpha_{i-k}, r_{i-k}), \ldots, (\alpha_{i+k}, r_{i+k})$ instead of just a causal one because the parent branch is completed before child branches are added. Once the initial segment of the child branch is synthesized, we switch to the causal model described in Sect. 4.2 for further branch growing.

Since the input usually contains only a relatively small number of furcations, it may lead to statistically biased results if the decision of where to place a furcation point is also based on the same non-causal Markov process. This is why we "interpolate" the furcation examples by computing a linear classifier that maps sub-sequences of length $2k+1$ on a parent branch to furcation probabilities. We use a support vector machine (SVM) to find the optimal classifier [Bis07].

For each vertex $\mathbf{x}_i$ on each branch of the input structure we take the non-causal neighborhood $(\alpha_{i-k}, r_{i-k}), \ldots, (\alpha_{i+k}, r_{i+k}) \in R^{2k+2}$ and the index distances to the previous and next furcation points (or the endpoints of the branch), n^- and n^+ respectively. Notice that due to the uniform resampling of the input structure, index distances are equivalent to discretized arc-length distances. These $4k+4$ dimensional samples are labeled with $+1$ if $\mathbf{x}_i$ is a furcation point and -1 if $\mathbf{x}_i$ is no furcation point. The SVM computation will produce the linear classifier L (*i.e.*, the hyperplane in R^{4k+5}) that fits the input with a maximum margin. In our implementation we used the SVM code provided by [CL01].

For a given branch we now pick d vertices with maximum furcation probability and start synthesizing child branches from there. Hence d controls the furcation complexity of the resulting branching structure and can be choosen randomly with a probability reflecting the number of sub-branches observed at branches of the input structure or manually. The coefficients n^- and n^+ effectively prevent furcation points from clustering. If a minimum distance between furcation points should be guaranteed then the furcation points have to be picked sequentially with each furcation blocking further sub-branches in its immediate neighborhood.

4.4. Space Colonization

So far we have treated the growing process of each branch independently. This, however, may lead to collisions and crossings between branches which is considered unnatural in many cases (*e.g.* rivers). This is why we use a technique similar to [PHL*09] in order to promote branch growing directions that conquer empty image regions and avoid growing into regions where other branches already exist.

We model the attractiveness of an image region by computing a density map which assigns to each point $\mathbf{p}$ a value

$$D(\mathbf{p}) = exp(-d(\mathbf{p},\mathbf{B})^2/\rho^2)$$

with $d(\mathbf{p},\mathbf{B})$ measuring the minimum Euclidean distance between $\mathbf{p}$ and the set of previously generated branches $\mathbf{B}$ (= polygonal skeletons with thickness). The standard deviation ρ is a parameter which controls how close the different branches are allowed to approach each other. Everytime a branch has grown by adding m new segments to it, the density map is updated. For efficiency reasons we restrict the update to a local vicinity with radius 3ρ since outside of this radius the change in density can be neglected.

The density map is included into the candidate selection process by computing for each extension candidate (cf. Sect. 4.2) the average growing direction (*e.g.* by principal component analysis) and then finding the maximum of the density map along that direction and within a distance not larger than three times the length of the extension. This density maximum $D_i \in [0,1]$ is then used to adapt the relative candidate selection probabilities by appropriately scaling the quality scores

$$Q_i^{**} = Q_i^*/(D_i+\varepsilon)$$

with some maximum bound of $1/\varepsilon = 10^3$ on the scaling factor. An intersection of that extension with the structure can robustly be detected by looking at the density function along the extension, which then shows a decreasing, increasing and again decreasing behavior. We explicitly set the quality score to zero whenever we detect such intersections.

4.5. Guided Synthesis

We can use another optional density map $G(\mathbf{p})$ in order to guide the synthesized branches along a prescribed path. For this we simply define a map with $G(\mathbf{p}) = 1$ everywhere except for a smooth valley along the prescribed path. Then we scale the selection probabilities of the extension candidates as in the previous section. With the width of the valley we can control how precisely the branch follows the path (Fig. 4). Notice that if the valley is chosen too narrow, *i.e.* the width being smaller than the amplitude of the characteristic bending features, the result will look less realistic since with stronger guidance we are effectively suppressing the mid-scale features of the bending model.

5. Fragment-Based Image Synthesis

In the next step we are generating the final output image by re-combining fragments from the input image. Similarly to [SYJS05] we exploit the synthesized branching structure to first cover the branches by a sequence of overlapping fragments. This is done in the same order as the structure synthesis, *i.e.* we first cover the main branch and then add the sub-branches at the furcation points. Finally the remaining gaps between the branches are filled by an image completion technique like [PSK06].

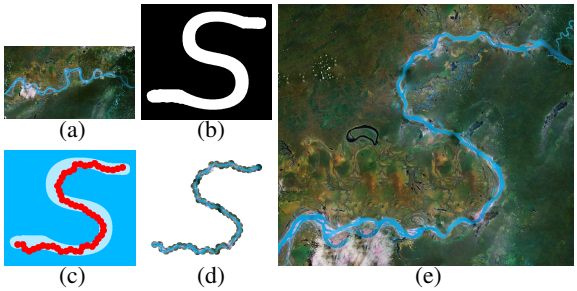


Figure 4: *Guidance along a prescribed path. (a) Shows the source image. In (b) we show the user defined density map, which is used to guide the generation as shown in (c). The missing regions of the synthesized river (d) is filled using interactive image completion and the result is shown in (e).*

The input to the image synthesis stage consist of the original input image, the user-defined input branching structure S (source structure), and the automatically synthesized output branching structure T (target structure).

We first compute a set of image fragments F_i covering the source structure S and then compose the branches of T from sequences of such fragments. In the selection of fragments we optimize for geometric similarity as well as for color consistency, *i.e.* the skeleton and thickness profile of the source branch segment covered by F_i has to match the corresponding skeleton and thickness profile of the target branch segment, and the pixel colors in the overlapping region of neighboring fragments in the target image have to match as well.

5.1. Fragment Definition

A fragment F_i is a disk-shaped image region of the input image, specified by its center position $\mathbf{z}_i$ with the radius s_i. Since the fragments are primarily used to synthesize the branching structure, source fragments have their center $\mathbf{z}_i$ on the polygonal skeleton of a source branch. The radius of a fragment is by default set to three times the (linearly interpolated) thickness of the branch at $\mathbf{z}_i$. In order to guarantee sufficient overlap between neighboring fragments, we set their distance $\|z_{i+1}-z_i\|$ to $\lambda(s_i+s_{i+1})$ with $\lambda = 0.75$ in all our experiments. Additional fragments are placed at the furcation points of S in order to be able to synthesize proper branchings in T. On the target branch structure we define a similar set of (yet empty) fragments G_j with the same radius and distance convention.

Each fragment F_i or G_j is associated with a segment of a polygonal skeleton which passes through the fragment center. We call this segment and the corresponding thickness profile the *shape* of the fragment. In order to simplify the shape comparison between fragments we resample these segments and thickness profiles to a fixed number of k uniformly distributed samples that we collect in a $3k$ dimensional vector $(\mathbf{p}_1,\ldots,\mathbf{p}_k,\mu_1,\ldots,\mu_k)$ or

$(\mathbf{q}_1,\ldots,\mathbf{q}_k,\nu_1,\ldots,\nu_k)$ respectively (*i.e.*, k 2D sample positions and k thickness values). Notice that this is in contrast to Sect. 4.1 where we resampled the skeleton with a constant step width h since here the same number of samples for fragments with different radii leads to different step widths. The fixed number of samples allows us to compare even fragments of different sizes in a meaningful manner. Also notice that we are using 2D sample positions and not angles to describe the geometry of the skeleton.

5.2. Branch Synthesis

The task of branch synthesis consists of finding the best combination of source fragments $F_{i(j)}$ that we copy to the target fragments G_j so that we obtain maximum shape matching between $F_{i(j)}$ and G_j and maximum color consistency between successive target fragments G_{j-1} and G_j. While in principle this is a complex global optimization problem similar to the one discussed in [SYJS05] it turned out in our experiments that in our setting it is sufficient to add one fragment at a time in a greedy fashion to obtain very good results.

The shape similarity between a source fragment F_i and a target fragment G_j is measured by the difference between their shape information $\mathbf{f}_i$ and $\mathbf{g}_j$. However, since the representation of the shape information in terms of 2D positions is not orientation independent, we first have to compute the rigid transform R_{ij} that brings the skeleton segment of F_i in best alignment to the skeleton segment of G_j. We do this in closed form by shifting F_i's center to the center of G_j and rotating F_i so that the square error

$$\sum_{l=1}^{k} \|R_{ij}(\mathbf{p}_l) - \mathbf{q}_l\|^2$$

is minimized. This corresponds to one step of a 2D version of the ICP algorithm [TL94]. Under this rigid transform the shape (non-)similarity between F_i and G_j is measured by

$$S(F_i,G_j) \;=\; \sum_{l=1}^{k} \|R_{ij}(\mathbf{p}_l) - \mathbf{q}_l\|^2 + \omega_S|\mu_l - \nu_l|^2 \qquad (1)$$

with some weight coefficient ω_S which by default we set to $\omega_S = 2$. Notice that even if the positional deviation grows with the size of the fragment (but the thickness difference does not), we still do not need to adjust ω_S since the fragment size is defined as a multiple of the thickness and hence both terms in (1) scale equally.

The color consistency $C(G_{j-1},G_j) = C(R_{i(j-1),j-1}(F_{i(j-1)}),R_{i(j),j}(F_{i(j)}))$ between two overlapping target fragments is simply measured by the sum of squared color differences in the region of overlap.

With these ingredients we can now formulate the greedy branch synthesis procedure which copies in each step that source fragment $F_i(j)$ to the next target fragment G_j which minimizes the objective function

$$i(j) \;=\; \arg\min_i S(F_i,G_j) + \omega_C C(G_{j-1},R_{ij}(F_i)),$$

where $G_{j-1} = R_{i(j-1),j-1}(F_{i(j-1)})$. The parameter ω_C balances shape versus color matching and depends on the contrast in the image and the variability of the branch structure. Just like in Sect. 4.2 we can increase the variability of the output by randomly picking one of the c best matching candidate fragments with a probability inversely proportional to the corresponding value of the objective function.

5.3. Fragment Composition

The greedy branch synthesis selects a new source fragment to be copied in every step. In the target image these fragments need to be composed in a seamless manner. Possible reasons for visible seams in the output can be shape matching errors between $R_{i(j),j}(F_{i(j)})$ and G_j (causing jaggy artifacts) or color matching errors. We compensate shape matching error by smoothly deforming the rigidly transformed source fragments so that the shape deviation is eliminated. The color matching error is eliminated by using photometric correction [PGB03] (Fig. 5).

(a) (b) (c)

Figure 5: *(a): Fragment composition without deformation (b): Deforming fragments avoids visible artifacts caused by shape matching errors. (c): Correcting the global color shift by adding constraints to the photometric correction.*

After the rigid transform $R_{i(j),j}$ the remaining shape deviation between the skeleton polygons of $R_{i(j),j}(F_{i(j)})$ and G_j is described by the displacement vectors $\mathbf{d}_l = R_{i(j),j}(\mathbf{p}_l) - \mathbf{q}_l$. In order to obtain perfect alignment, we compute a smooth displacement field $\mathbf{d}_{uv}$ which assigns a displacement vector to each pixel in G_j such that the energy functional

$$E = \sum \|4\mathbf{d}_{uv} - \mathbf{d}_{u-1,v} - \mathbf{d}_{u+1,v} - \mathbf{d}_{u,v-1} - \mathbf{d}_{u,v+1}\|^2$$

is minimized subject to the constraint that the vectors $\mathbf{d}_l$ are interpolated. This requires the solution of a linear system with the number of unknowns being the number of pixels in the fragment. In order to find this minimizer in a convenient way, we use the freely available constrained solver presented in [BZK09]. Since the shape deviation between $R_{i(j),j}(F_{i(j)})$ and G_j can be expected to be rather small, the deformation field is smooth and injective such that no artifacts are generated in this step (Fig. 5).

In order to remove seams caused by color mismatch between fragments we apply photometric correction [PGB03] where color gradients instead of pixel colors are copied and the final target fragment colors are computed by an integration step guaranteeing zero gradients along common boundary with previous fragments. Applying this technique directly, however, tends to produce significant global color

shifts as we keep on adding new fragments (see Fig. 5). To prevent this color shift, we introduce additional constraints to the photometric correction forcing the colors of the copied fragment to remain unchanged along the boundary where the new fragment is not adjacent to a previous one.

Similar to [KSE*03] we compute new seams which partition the overlapping regions of two adjacent fragments such that color values across that new seam are similar. This can efficiently be done by using Graph Cuts. The graph is constructed by connecting pixels on the boundary of the overlap region to source and sink and connecting every pixel of that region with its neighbors. The cost function between two neighboring pixels is defined such that it is proportional to the color difference between source and target fragment at these positions. The min-cut or max-flow algorithm will then identify the optimal cut through the overlapping region.

A final problem is that when adding sub-branches to the main branch, we need to overwrite previously synthesized fragments to properly introduce the furcation points. This, however, is very easy to achieve since we know exactly where the fragments need to be placed and what their radius is. Hence, we can simple delete that region from the output image and continue with the image synthesis (Fig. 6). One could also use squared shape fragments where the computation of the overlapping region and inside tests can be performed more efficient. Nevertheless in our experiments we observed that round fragments cause less artifacts and accelerate the computation of Graph Cuts and photometric correction by $\sim 20\%$, because of smaller overlapping regions.

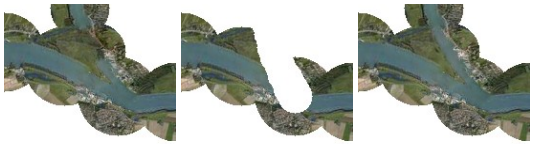


Figure 6: *To generate proper furcations, parts of the parent branch need to be deleted before synthesizing sub-branches.*

5.4. Image Completion

After the branching structure has been synthesized, we complete the output image by filling the remaining gaps using the image completion technique described in [PSK06]. Here we make sure that only those fragments are copied from the input image that do not overlap with branches since otherwise pseudo-branches might be generated.

6. Results

We performed our experiments on an Intel Core i7 with 2.7GHz in a single threaded prototypical application. Drawing a simple *source tree* takes about 2 minutes of user interaction. For, *e.g.*, Fig. 1 we specified only 65 points. The reliability of a maximum gradient detection operator to estimate the radius strongly depends on the input images: In Fig. 1 we re-defined almost all radii while in the river example (Fig. 7) we only corrected less than ten skeleton nodes. Training the SVM takes about one minute. Note that both steps need to be done only once for one source image.

Example	#SF / #TF	R	IS (sec)
River	68/139	67	30.6
Mountain	31/43	144	72.8
Collard	68/71	100	50
Lightning	251/378	20	8.6

Table 1: *Timings for the image synthesis. #SF and #TF denote the number of source and target fragments.*

In Fig. 7 we present some results. For larger and more complex output structures please see the accompanying supplemental material. The first column shows the input image from which the structures are extracted and learned. After synthesizing the structure which is used in the fragment-based image synthesis we obtain images as shown in the second column. The final result produced with an interactive image completion technique is shown in the third column. In order to quantitatively evaluate the result of the structure synthesis we compute histograms from the angle and thickness distribution of input and target structure. In Fig. 7 we show these histograms right to the synthesized image. The corresponding histograms for the input and output images are placed on top of each other for more convenient comparison. The left histograms show the angle and the right histograms the thickness distributions. In Fig. 8 we show an example where we additionally considered a proper perspective rectification for a source image not captured from a frontal view.

To create the target structure the user decides how much he wants to interact. The system provides metaphors to automatically extend existing single branches (point and click), to alternating extend all branches, to place furcations automatically or manually, to manually delete unwanted sub-branches or to edit the density map in order to provide a guidance at the coarsest scale (Sect. 3). Since extending existing branches can be done instantaneously this part is fully interactive. The river and mountain range (Fig. 7) was generated without manual interaction. For the collard and lightning example we deleted several branches in a last step, which took less than two minutes. Synthesizing image content along the structure is again an automated process and involves solving linear systems for the warping, photometric correction and computing new seams. The computation time for the image synthesis (IS) strongly depends on the radius (R) of the fragments (Table 1). Our interactive image completion for filling the gaps between the structure takes the most time (10 minutes) and we plan to replace this by a fully automatic method which adjusts positions and size of the target fragments starting at furcation points. Common texture synthesis methods, *e.g.* [BD02, KSE*03], should be able to handle this task, since in this stage we only need to synthesize rather homogeneous image content. With this the

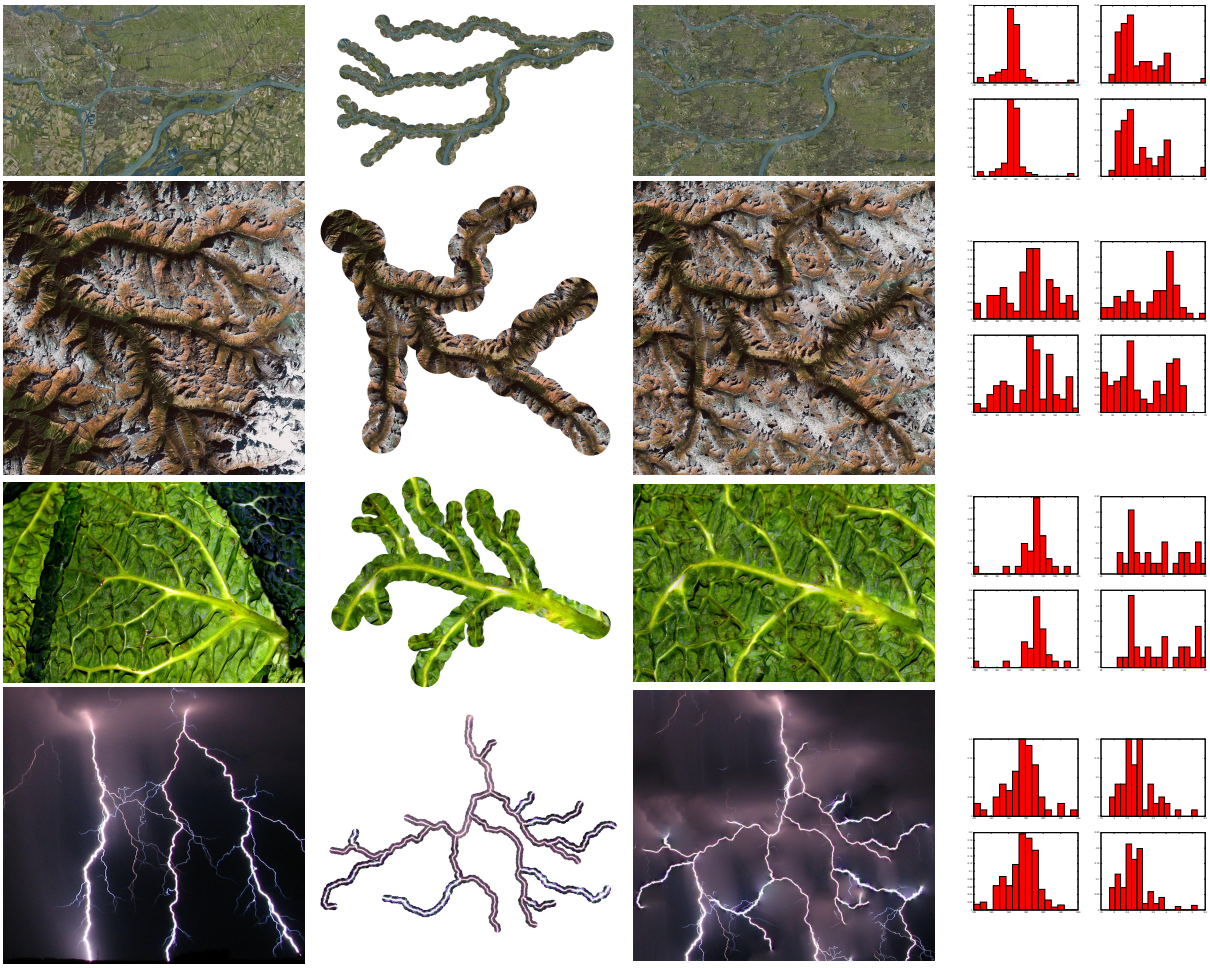
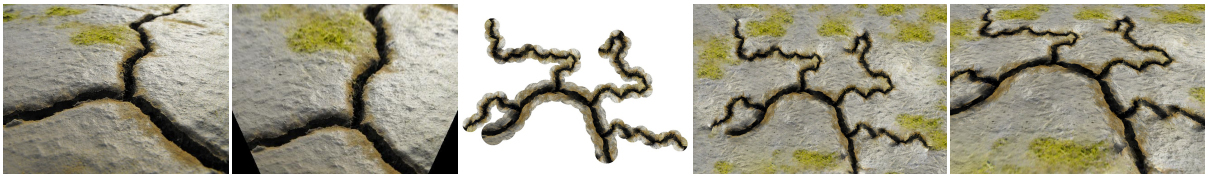


Figure 7: *Results generated by our system. The first column shows the input image. The synthesized structure is seen in the second column, while the third column shows the final result. For each result we computed a histogram for the angles (left) and thickness values (right). The corresponding histograms for the input and output images are placed on top of each other for more convenient comparison. In the last row we allowed that the interactive image completion adds fragments showing the nerve of the lightning, hence some additional small branches appeared.*



(a) (b) (c) (d) (e)

Figure 8: *Crack structure example. The image of a crack (a) is rectified (b) to allow for perspectively correct copy and paste operations during the image synthesis process. (c): Structure propagation only. (d): Initial Solution. (e): Back-distorted solution with similar perspective as the one shown in the input image.*

generation of multiple outputs from a source image becomes more easy.

Limitations The main limitation of our system is, that it can only handle bi-furcation or tree-like structures, *i.e.* it is not possible to synthesize structures that split and merge again as observed at cracks in dry ground or city layouts. For this, methods like [BD02] seem to be more suitable. Our system is purely 2D and cannot produce 3D structures. It is also hard to create images showing 3D structures, like trees or plants in general. A user could identify the structure at a certain depth layer, but the image synthesis will probably fail, since it is very likely that fragments at a wrong depth

layer are selected, which will result in a scrambled looking image. For future work we would like to detect source structures automatically or at least strongly machine guided and automatize the time consuming interactive image completion.

7. Conclusion

In this paper we presented an approach to synthesize images containing branching structures. Our branching model is a Markov model where proceeding angles and thickness parameters are used to find the most similar structures on a source structure and expand a target structure by the successive elements from the source structure. The quality score used in this process is updated due to a global similarity measurement involving histograms over the angle and thickness distribution. As seen in the results these histograms indicate that the synthesized structures have similar global characteristics. Our space colonization approach prevents the structure from self-intersecting and allows for guidance by a user. For each individual structure we learn a branching model by using support vector machines. In the image synthesis part we use multiple techniques to increase the image quality.

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *I3D* (2001), pp. 217–226. 2

[AVB08] ALIAGA D. G., VANEGAS C. A., BENEŠ B.: Interactive example-based urban layout synthesis. In *SIGGRAPH Asia* (2008), ACM, pp. 1–10. 2

[BD02] BROOKS S., DODGSON N.: Self-similarity based texture editing. In *SIGGRAPH* (2002), ACM, pp. 653–656. 2, 8, 9

[Bis07] BISHOP C. M.: *Pattern Recognition and Machine Learning*, 1 ed. Springer, October 2007. 5

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM TOC 28*, 3 (2009), 1–10. 7

[CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. *ACM TOC 27*, 3 (2008), 1–10. 2

[CL01] CHANG C.-C., LIN C.-J.: *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm. 5

[DCOY03] DRORI I., COHEN-OR D., YESHURUN H.: Fragment-based image completion. *ACM TOC 22*, 3 (2003), 303–312. 3

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *ICCV* (1999), p. 1033. 2

[HE07] HAYS J., EFROS A. A.: Scene completion using millions of photographs. *SIGGRAPH 2007 26*, 3 (2007). 3

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

[HOCS02] HERTZMANN A., OLIVER N., CURLESS B., SEITZ S. M.: Curve analogies. In *EGRW* (2002), pp. 233–246. 2

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. In *SIGGRAPH* (2005), pp. 795–802. 3

[KFCO*07] KOPF J., FU C.-W., COHEN-OR D., DEUSSEN O., LISCHINSKI D., WONG T.-T.: Solid texture synthesis from 2d exemplars. *ACM TOG 26*, 3 (2007), 2:1–2:9. 4

[KL04] KIM T., LIN M. C.: Physically based animation and rendering of lightning. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 267–275. 2

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. In *SIGGRAPH* (2003), ACM, pp. 277–286. 2, 8

[KW07] KWATRA V., WEI L.-Y.: Example-based texture synthesis. In *SIGGRAPH Course #15* (2007). 3

[LH05] LEFEBVRE S., HOPPE H.: Parallel controllable texture synthesis. In *SIGGRAPH* (2005), pp. 777–786. 3

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. In *SIGGRAPH* (2006), pp. 541–548. 3

[Lin68] LINDENMAYER A.: Mathematical models for cellular interaction in development i. filaments with one-sided inputs. In *Journal of Theoretical Biology* (1968). 2

[Nor98] NORRIS J.: *Markov Chains (Cambridge Series in Statistical and Probabilistic Mathematics)*. Springer, July 1998. 4

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM TOC 22*, 3 (2003), 313–318. 7

[PH89] PRUSINKIEWICZ P., HANAN J.: *Lindenmayer systems, fractals and plants*. Springer-Verlag New York, Inc., 1989. 2

[PHL*09] PALUBICKI W., HOREL K., LONGAY S., RUNIONS A., LANE B., MĚCH R., PRUSINKIEWICZ P.: Self-organizing tree models for image synthesis. In *TOC* (2009), vol. 28, ACM, pp. 1–10. 2, 5

[PSK06] PAVIĆ D., SCHÖNEFELD V., KOBBELT L.: Interactive image completion with perspective correction. *Vis. Comput. 22*, 9 (2006), 671–681. 3, 6, 8

[RCOL09] ROSENBERGER A., COHEN-OR D., LISCHINSKI D.: Layered shape synthesis: automatic generation of control maps for non-stationary textures. In *SIGGRAPH Asia* (2009). 3

[SYJS05] SUN J., YUAN L., JIA J., SHUM H.-Y.: Image completion with structure propagation. In *SIGGRAPH* (2005), ACM, pp. 861–868. 3, 6, 7

[TCLT07] TING H., CHEN S., LIU J., TANG X.: Image inpainting by global structure and texture propagation. In *Proc. of int. Conf. on Multimedia* (2007), pp. 517–520. 3

[TCPT03] TOYAMA C. E., CRIMINISI A., PÉREZ P., TOYAMA K.: Object removal by exemplar-based inpainting. In *CVPR* (2003), pp. 721–728. 3

[TL94] TURK G., LEVOY M.: Zippered polygon meshes from range images. In *SIGGRAPH* (1994), ACM, pp. 311–318. 7

[TZW*07] TAN P., ZENG G., WANG J., KANG S. B., QUAN L.: Image-based tree modeling. In *SIGGRAPH* (2007), ACM, p. 87. 2

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *EG-STAR* (2009). 3

[WZSB08] WU Y., ZHANG H., SONG C., BAO H.: Space-time curve analogies for motion editing. In *GMP08* (2008). 2

[ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M.: Terrain synthesis from digital elevation models. *IEEE Trans. on Visualization and Computer Graphics 13*, 4 (2007), 834–848. 3