Dynamic Texture Editing

Radek Richtr* Michal Haindl†
The Institute of Information Theory and Automation of the CAS*†,
Faculty of Information Technology, Czech Technical University*

Figure 1: A preview of several edited synthetic dynamic textures.

Abstract

A fast simple method for dynamic textures enlargement and editing is presented. The resulting edited dynamic texture is a mixture of several color dynamic textures that realistically matches the given color textures appearance and respects their original optical flows. The method simultaneously allows to spatially and temporarily enlarge the original dynamic textures to fill any required four dimensional dynamic texture space. The method is based on a generalization of the prominent static double toroid-shaped texture modeling roller method to the dynamic texture domain. The presented method keeps the original static texture roller principle of separated analysis and synthesis parts of the algorithm. In its analytical step, the input textures patches are found by an optimal overlap tiling and the subsequent minimum boundary cut. The optimal toroid-shaped dynamic texture patches are created in each spatial and time dimension, respectively. The spatial dimension tile border is derived by textural features, color-tone, and the minimal overlapping error. The time dimension tile border is detected by minimizing the overlapping error and using the input textures optical flow. The realistic appearance of the dynamic textures mix requires to edit the patch color space and to find border patches which consists from more than one type of the texture. These border patches are found similarly to the multi-texture analysis patch step. Since all time-consuming processing, such as the finding of optimal spatio-temporal triple toroidal patches, are done in the analytical step which is completely separated from synthesis part, the synthesis of the edited and enlarged resulting texture can by done very efficiently by applying simple set of repeating rules for these triple toroidal patches. Thus the presented method is extremely fast and capable to synthesize a learned natural dynamic texture spatially and its time span in real-time.

*e-mail:richtrad@utia.cas.cz
†e-mail:haindl@utia.cas.cz

SCCG '15, April 22 – 24, 2015, Smolenice, Slovakia.
 ISBN 978-1-4503-3693-2/15/04…$15.00
DOI: http://dx.doi.org/10.1145/2788539.2788559

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: Computer Graphics, Dynamic Texture, Dynamic

Texture Editing, Patch-based Texture Synthesis, Dynamic Texture Modeling, Image Generation

1 Introduction

Visual texture modeling is the critical part for any computer-based visualization application because whatever size is the measured texture, it is always inadequate and requires its enlargement to cover the required visualized object's surface area. The available material sample size is either too small for rendering complex and large virtual scenes, if we measure material visual properties of real existing objects, or the measurement technology does not allow us to measure larger material samples. The typical example is the recent most advanced visual surface material representation in the form of the bidirectional texture function [Haindl and Filip 2013]. The amount of such measured data similarly to dynamic textures is immense, e.g., in the range of tera bytes even for such spatio-temporally restricted measurements thus any such texture visualization inevitably requires simultaneously enlargement and some compression capability. Texture editing enables to create photo-realistic synthetic textures, which are either difficult to measure or which even do not exist in nature. Dynamic (DT) texture synthesis and editing aims to create a visual texture which is perceptually similar, ideally visually indiscernible, to the target dynamic texture and has the required spatial and temporal extent. The target texture can be either a measured sample or a set of desired features for the resulting DT. Natural DT modeling is a very challenging and difficult task, due to unlimited variety of possible materials, illumination, and viewing conditions, simultaneously complicated by the strong discriminative functionality of the human visual system. Dynamic texture editing is useful process which allows to synthesize alternative and unmeasured types of DTs. It allows to reach huge compression ratio if numerous DTs can be constructed from several basis small DT samples, or to study relationship between model parameters and their impact on visual DT appearance, etc.

1.1 Dynamic Texture

Similarly to other types of visual textures, there is neither an exact definition of the dynamic (or video, temporal) texture [Haindl and Filip 2013]. DT can be characterized by some spatially invariant statistics like other visual textures [Zhu et al. 1998] but additionally to that it requires also some temporal invariance. DT can be represented as as a realization of a 4D stochastic random field. The two main approaches can by applied to the dynamic texture modeling - the approach based on the measured data sampling and the mathematical model based approach. Typically the modeling by some mathematical probabilistic model is by far less demanding on memory, but sometimes produces lower visual quality results, while the intelligent sampling methods can produce better quality textures but they need to store material patches [Haindl and Richtr 2013]. Three major properties [Doretto et al. 2003] are present in DTs. They intuitively define whole texture - the static texture itself, global dynamics and local dynamics (Fig. 4). The static texture represents a structure of the scene and its objects, the global motion moving of the whole scene (e.g. rotating or sliding camera), and the local motion dynamics in the texture (small motion, oscillation or overlapping and disappearing of small objects).

Some of the early methods were based on simple particle system, where particles were parts of the mapped semi-transparent textures with integrating light from the neighbouring polygons. Perry and Picard [Perry and Picard 1994] modeled flames, Stam [Stam and Fiume 1993; Chiba et al. 1994] or Chiba [Chiba et al. 1994] published similar models simulating simple gas phenomena like clouds or smoke. The Szummer [Szummer and Picard 1996] method is based on (causal) auto-regressive model restricted to gray-scale DT and linear dynamic between pixels only. The Schodl [Schödl et al. 2000] method is restricted to temporal enlargement based on similarity and optical flow to find possible transitions from one frame in the original video clip to another one. Parts of the video with similar textural and motion data are simply played in infinite loops using blur between frames to reduce visually apparent jumps. The statistical method by Bar-joseph [Bar-Joseph et al. 2001; Bar-Joseph 1999] can create DT using wavelet transform and steerable pyramids directly extending the 2D problem into 3D domain. This method produces new data, but the method is time and space consuming and can produce [Bar-Joseph et al. 2001] only spatially and temporally limited data. The larger DT can be generated only by creating several small ones and concatenating them while blending their boundary frames, but this approach often introduces strong optical distraction and excessive blur in the blended frames. This method can also create some type of mix-of-DT output. Typical sampling methods [Schödl et al. 2000; Kwatra et al. 2003; Phillips and Watson 2003] are based on simple repetition of input data with edge blending and adequate morphing techniques. The repetition is done in spatial or temporal way and may be just frame based [Schödl et al. 2000] or it uses time consuming graph-like min-cut methods [Kwatra et al. 2003]. The method [Lizarraga-Morales et al. 2014] selects subsequent patches using iterative greedy-like search based on the modified textural LBP fetures. Its major drawback is unseparated analytical and synthesis steps. The Doretto [Doretto et al. 2003] and Soatto [Soatto et al. 2001] model based method is based on auto-regressive moving average process and SVD. This method allows to edit the synthesized result by changing model parameters, but the parameters have unknown influence on the textural appearance (i.e. speed or direction for wind, or granularity for water textures) and must be interpreted for every given type of a texture. Similarly to static textures, DT can be modeled as a mixture of several dynamic textures [Chan and Vasconcelos 2008; Chan and Vasconcelos 2005]. These types of textures can be created e.g. by layered dynamic textures [Chan and Vasconcelos 2006], mix of dynamic textures [Bar-Joseph et al. 2001; Chan and Vasconcelos 2005; Chan and Vasconcelos 2008], or by our triple toroid- shaped tiles and optical flow based approach which focus on dynamics and human-eye-observation. Filip [Filip et al. 2006] presented much faster way to synthesize color textures. Similar results were presented in [Chan and Vasconcelos 2005; Constantini et al. 2006; Costantini et al. 2008] using tensor decomposition and SVD techniques.

We present a new simple method for synthesizing dynamic textures, mix of DTs and DT editing. The method generalizes our approach [Haindl and Richtr 2013] mainly by its novel editing and mixing capabilities and it is illustrated on the inpainting application. The possibly infinitely large (in spatial and temporal dimensions simultaneously) synthesized texture is created from a noticeably smaller patches of input DTs. The result is seamless, without any strong distortions and does not require any blurring or morphing techniques. These properties is achieved through intelligent sampling and subsequent copying patches abreast using simple adjacency rules. Few triple toroid-shaped tiles can create numerous noticeably larger synthesized DTs with similar stochastic properties. By using toroidal-shaped tile and fitting them by optical flow, the synthesized texture can be recreated from various input textures to generate many types of output textures or even mix-of-textures. The repetition of the spatial and temporal dimension is suppressed both by using overlapping samples and their random placement in the synthesized texture and temporal deformation of the texture dynamics. This intelligent sampling allows to retain a small enough

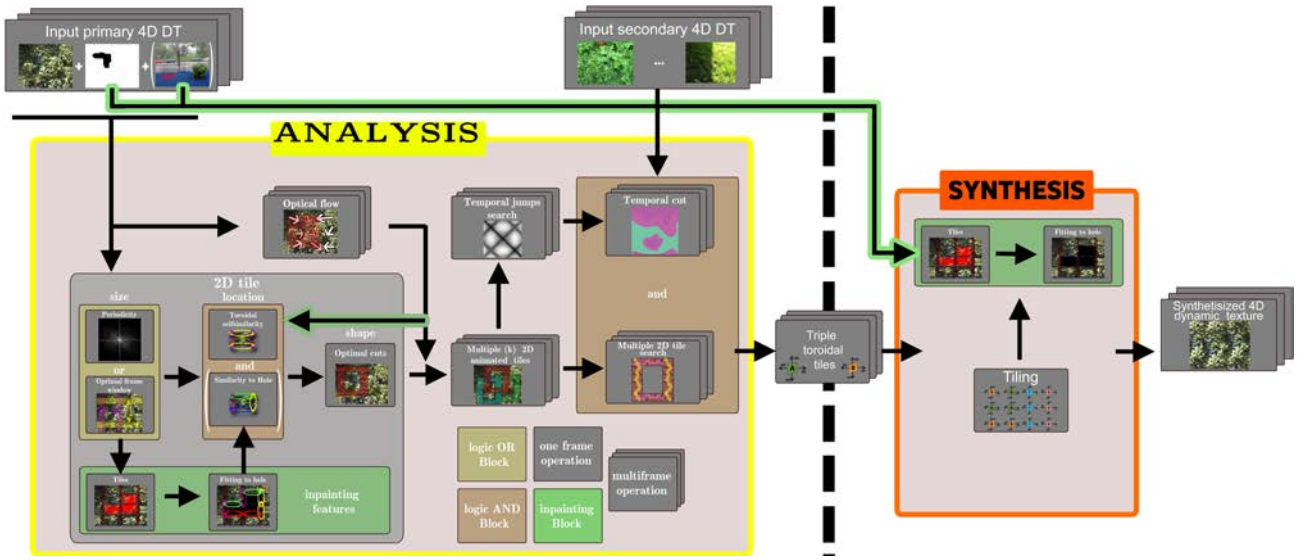


Figure 2: The overall flowchart of the presented method.

representative data samples to represent the whole texture. It also allows the creation of the infinitely large seamless output by repeating sequence of these toroidal samples. Our method can create large range of varios DT types and produce high quality realistic appearance results.

2 Toroidal-Based Texture Model

The presented enlargement method principle is to find optimal triple toroid-shaped dynamic texture tiles which can be subsequently seamlessly repeated in dynamic texture data space spatial and temporal directions. The method estimates several optimal triple toroid-shaped dynamic texture tiles combining estimated optical flow and double toroid-shaped dynamic texture frame tiles as described in detail in [Haindl and Richtr 2013] and it is a part of the presented method. Although the principle of the optimal spatial and temporal cuts detection is similar, the temporal and spatial visual perception differs [Edelman 1992; Fahle and Poggio 1981], thus we have developed two distinct approaches for the spatial and optical flow driven temporal cuts. The overall flowchart of the present method is illustrated on Fig. 2. The input 4D DT's frequencies and the optimal patch size (2D tile block: size, location) are computed and their time properties (Optical flow block) are analyzed in parallel, followed by the search for additional patches (Multiple (k) 3D animated tiles) and their temporal translations (Temporal jumps search). The optimal spatial (Multiple 2D tile search) and temporal (Temporal cut) cut is then computed. The synthesis step avoids damaged or unwanted parts and the results are synthesized by randomly generated DT arrangement.

2.1 Static and Dynamic Double Toroid-Shaped Tiles

The toroid-shaped tile (see Fig. 2) is a patch from the original data with the specific property to have toroidal border condition in each (horizontal, vertical, and temporal) dimension. The textural tile is assumed to be indexed on the regular three-dimensional toroidal lattice. The lattice size depends on the estimated texture periodicity in all dimensions.

Figure 3: The double toroid-shaped tile and its source frame texture, respectively.

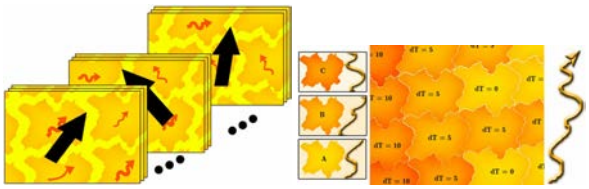


Figure 4: Left: Dynamic overview. Black arrows: Global dynamics (optical flow) of the whole texture cause moving of every patch; Red arrows: Local patch dynamics (optical flow) which can be different each time and derives small patch-shape changes. Right: Optical flow illustration, synthesized result (with varying temporal shift illustrated by the color tone of patches), three patches with different time shift (0, 5 and 10 frames) and different local (arrows) optical flow, respectively.

The optimal tile search algorithm which respects its toroidal border condition produces a tile which can be seamlessly repeated in both spatial and the temporal dimensions, respectively. Thus the boundary between two tiles placed abreast is invisible. The main idea of the triple-toroidal tiling is to find some relevant and sufficiently representative tiles from the measured DT, which can be *directly* seamlessly copied to the output DT, where the offset is driven by the tile shape placed on the three dimensional tile label lattice.

The static double toroid-shaped tile edges (patches) are found by the roller method [Haindl and Hatka 2005] and they are combined using their optical flow controlled propagation throughout the time

(see [Haindl and Richtr 2013]). The tiles are fitted to local dynamics (see Fig. 4) and subsequently expanded into the triple toroid-shaped forms.

Let us denote $r = [r_1, r_2, r_3, r_4]$ a multiindex, where the components are row, column, spectral (color information and vector of optical flow), and frame indices, respectively. $\bullet$ denotes all values of the corresponding index. The optimal horizontal and vertical overlaps are found using the following criterion:

$$\zeta_r^* = min_\zeta \left\{ \frac{1}{\lfloor \zeta \rfloor} \sum_{\forall r \in I_\zeta} \psi_r^\zeta \right\} , \quad (1)$$

where $\lfloor \zeta \rfloor$ denotes the the corresponding overlap area of ζ, I_ζ is the overlap area index set, and ψ_r^ζ is the L_2 norm of the corresponding ($\zeta = h$ horizontal or $\zeta = v$ vertical) overlap error for a multispectral pixel vector Y_r.

The optimal size of the tile is found by minimizing the overlap error ζ_r^* (see (1) or [Haindl and Hatka 2005]) by the Fourier transformation [Haindl and Hatka 2005] to find the dominant texture periodicity (with the weight in every frame, or in one representative frame).

The located single frame double toroid-shaped tile is propagated along the optical flow which represent DT global dynamics to the subsequent frames to model the textural structures movement. The propagated double toroid-shaped tile (see Animated 2D tile block in Fig. 2) in every frame serves as the *initialization* for local modification of such a double toroid-shaped tile to each frame-specific final shape. Multiple animated 2D tiles (with possible overlapping) are found in the source texture by minimizing the overlap error ζ_r^* in border areas of the patches given advantage of the same global optical flow in the same frames of DT. The minimal distance of the patches are given by smaller breadth of up/down of left/right overlaps (see block Multiple (k) 2D animated tiles in Fig. 2). The local dynamics problem is solved by the iterative border elaboration to minimize the overlapping border error ζ_r^* in areas given by approximately animated tiles. The optimal shape for the primary tile is computed by the back-propagation dynamic-programming-like rules which minimize both spatial $(\Psi_r^{h+}, \Psi_r^{v+})$ and temporal $(\Psi_r^{h*}, \Psi_r^{v*})$ cut cost:

$$\begin{aligned}
\psi_r^{h*} &= \psi_r^{h*} + \min\left\{ \psi_{r-[c,0,\bullet,1]}^{h*}, \ldots, \psi_{r+[c,0,\bullet,-1]}^{h*} \right\} , \\
\psi_r^{v*} &= \psi_r^{v*} + \min\left\{ \psi_{r-[0,c,\bullet,1]}^{v*}, \ldots, \psi_{r+[0,c,\bullet,-1]}^{v*} \right\} , \\
\Psi_r^{h+} &= \psi_r^{h+} + \min\left\{ \psi_{r-[c,1,\bullet,0]}^{h*}, \ldots, \psi_{r+[c,-1,\bullet,0]}^{h*} \right\} , \\
\Psi_r^{v+} &= \psi_r^{v+} + \min\left\{ \psi_{r-[1,c,\bullet,0]}^{v*}, \ldots, \psi_{r+[-1,c,\bullet,0]}^{v*} \right\} .
\end{aligned}$$

The parameter c is derived by the average of the optical flow from the actual frame to the next frame plus local optical flow in a given patch (it usually creates large areas *inside* patch to handle small periodic motion like leaves fluttering). Large optical flow values increase this parameter value, which allows greater flexibility and possibly also significantly larger tile border modifications. This allows adaptation for scenes with higher movement dynamics.

This scheme allows the non-greedy-computation of the double (spatially) toroid-shaped patch in the whole 4D textural space, what is advantageous in cases where there are several distinct dynamic textures moving in spatial directions. In this case the back-propagation is used to find parts inconvenient of the texture (i.e., parts of the dynamic texture with not sufficiently long patches). It can also override the problems with errors or textural artifacts.

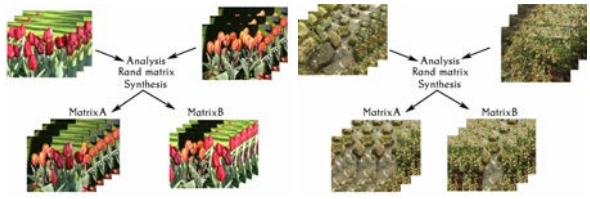


Figure 5: The example of combining two very different types of textures (left - river and shrub, right - different color tone of background and focused objects.

These restricted search areas extremely speed up the computation time needed to find the optimal tile shape in comparison to the whole textural multispectral space search. These areas are simultaneously beneficial in avoiding some typical DT modeling problems such as artifacts introduced by extremely varying speed or problems with highly dynamic textures (e.g., fire or water) or mix of dynamic textures. In these areas optimal fitting planes can be found. In the fitting step the overlap error is computed only from actual spatial data:

$$\begin{aligned}
\Psi_r^{h+} &= \psi_r^{h+} + \min\left\{ \psi_{r-[c,1,\bullet,0]}^{h+}, \ldots, \psi_{r+[c,-1,\bullet,0]}^{h+} \right\} , \\
\Psi_r^{v+} &= \psi_r^{v+} + \min\left\{ \psi_{r-[1,c,\bullet,0]}^{v+}, \ldots, \psi_{r+[-1,c,\bullet,0]}^{v+} \right\} .
\end{aligned}$$

The major advantage of our approach is the complete separation between the analytical and the synthesis steps. Once patches are found, the output can be synthesize on-the-flow simply by copying toroid-shaped data tiles with the corresponding offsets. These tiles are stored in a small tile database which is entirely unrelated to any synthesis application, so the analysis and synthesis steps are completely separated.

2.2 Triple Toroid-Shaped Tile

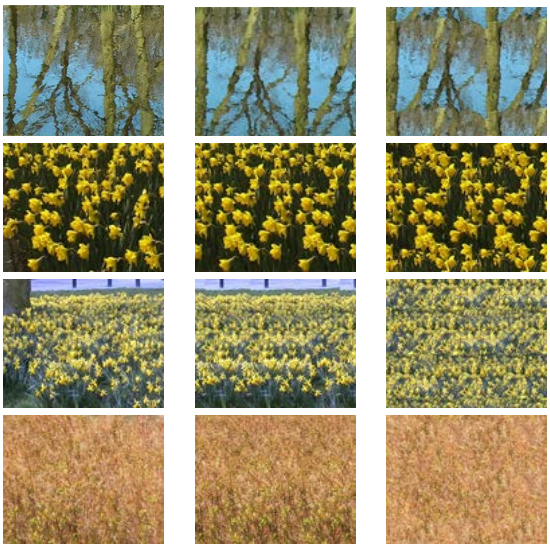
A temporal cut which allows seamless tile repetition in the time dimension can be done by the graph-cut algorithm [Kwatra et al. 2003] or by a frame-to-frame loop [Schödl et al. 2000]. Our algorithm starts by finding the most similar sets of o frames, which is the most time consuming part of our approach (hence the nearly all-to-all distances are computed). For finding similar frames, we adopt the similarity matrix proposed by Schödl [Schödl et al. 2000]. The minimized error can be chosen as the pixel-to-pixel difference of frames or by some other metric [Kwatra et al. 2005] with addition of local optical flow like fourth part of RGB spectral dimension. Since the most similar sets of frames are found, our approach creates transition pixel per pixel or by one consistent part of texture to other given by type of optical flow in the texture (for details see [Haindl and Richtr 2013]).

3 Dynamic Texture Editing

Our approach allows to edit DT in several ways. The system of toroidal samples allows to apply the textural operators to tone colors, change dynamics or differs texture itself without possibility to affect the homogeneity of the whole DT. For example to change the dynamics of one object in the DT it usually requires area segmentation (which must be consistent in all frames), detection of an object and changing its dynamics. Our approach allows to skip the

Table 1: Measured textures (left column), enlarged cutouts in temporal and horizontal dimension (middle), and the enlargement in all three dimensions, respectively.

Original	X,T enlarged	X,Y,T enlarged



segmentation step, since the texture is divided into logically consistent patches (given by spatial similarity and similar dynamics) from previous analysis steps.

Elementary, but not trivial type of texture editing is to synthesize texture consisting of multiple input textures (Fig. 5). The resulting texture is therefore a mix of textures, which can vary in two of the three basic properties of DT - the texture itself (this includes changes in lighting, but also the other conditions - detail, different viewing angle, etc.) and local dynamics (intensity and direction of wind, rain, etc.). It is obvious that the maximal number of possibly synthesized textures is:

$$(N_{in\_tex} * N_{p\_per\_tex})^{(\lceil \frac{w\_tex}{w\_patch} \rceil + 1) * (\lceil \frac{h\_tex}{h\_patch} \rceil + 1)} \quad , \qquad (2)$$

if we assume a constant distribution of patches in input textures. N_{in_tex} is number of input textures, $N_{p\_per\_tex}$ is average numbers of patches per input texture. Similarly, w_tex and h_tex are vertical and horizontal size of the synthesized texture respectively and w_patch and h_patch are horizontal and vertical size of patches, respectively. More generally and because we often use different number of patches in every texture (to ensure better visual appearance of synthesized textures) the (2) can be simplified to:

$$(N_{patches})^{(\lceil \frac{w\_tex}{w\_patch} \rceil + 1) * (\lceil \frac{h\_tex}{h\_patch} \rceil + 1)} \quad , \qquad (3)$$

where the first product in 2 becomes to $N_{patches}$ as the number of all patches found by the analytical part of our algorithm. The basic synthesis of the mix of DTs consist just from finding the same triple toroid-shaped tiles or - more accurately - by finding one (or more) optimal triple toroid-shaped tile in one primary DT and finding the minimal cost of placing this shape to other textures by finding optimal time-spatial shift of the optimal shape. If there are more textures the computational time can by reduced by using a texture pyramid, testing only local optical flow, or testing global optical flow first (alternatively in the backward order).

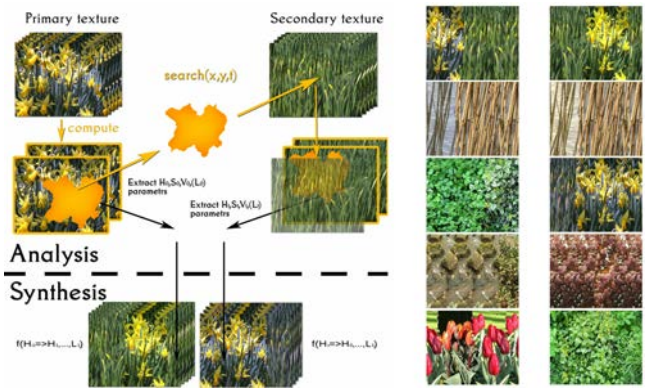


Figure 6: Illustration of the direct mix of DTs with color toning and two synthesized textures with different color tone (left). Examples of synthesized results without color toning and with color toning (right).

3.1 Colortone

Using several input textures can create the problem with slightly different color tone or illumination of particular input DTs. This problem can be overcome, i.e., by tone mapping (see Fig. 7). The color tone of patches can be fitted to the color tone of the selected input texture, or a consensus of all textures. More advanced algorithms and approached can be used for this problem [Tian et al. 2002]. We used very simple but satisfying approach - fitting average R,G,B or H, S, L/V values of one texture to another by per pixel multiplication. This simple approach creates good results in the most of cases (see Fig. 6) of synthesized DTs. Optimal shape is computed in the primary texture and found by spatio-temporal shift in the secondary texture. HSV and L parameters (or RGB) are extracted from all input textures. In the synthesis step the color tone of textures are fitted (per pixel) to one input texture to create similar color appearance. The number of possibly synthesized textures then grows to:

$$N^*_{in\_tex} * (N_{patches})^{(\lceil \frac{w\_tex}{w\_patch} \rceil + 1) * (\lceil \frac{h\_tex}{h\_patch} \rceil + 1)} \quad , \qquad (4)$$

where $N^*_{in\_tex}$ is the number of input textures (color tones) to which the synthesized result can be mapped (plus possibly consensus of all/subset of textures). In this automatic approach we assume that the input DTs are selected manually and that they are visually similar and do not differ significantly in color (i.e. grass and clouds).

3.2 Mix of Dynamic Textures

Our triple toroid-shaped approach can handle the mix of DT (see Fig. 5 and previous section) by random patches placing. Sometimes DTs vary in dynamics or appearance and patches that satisfy the similarity can not be found. In these cases we can use border type of input texture. We assume the existence of the transition (border) between types of DT but also the lack of one (or both) texture itself (see Fig. 7). In this approach the optimal shape is computed in the primary texture and found by spatio-temporal shift in the secondary textures with different type of DT. The random DT arrangement is generated by iterative growing of DT types from two or more arbitrary coordinates. Patches that can't be filled are computed in the second step. For every empty location the best-fit patch are found in the given border textures (marked by violet) by minimizing error cost to *each* already (even another border patch) completed adjacent

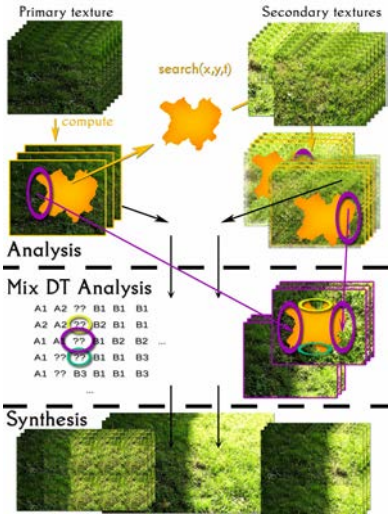


Figure 7: Illustration of the mix of DTs with border patches. Three synthesized textures with different DTs are shown (the color tone of textures change are optional).

patches found before. One (by the types of DT) of four/five (by the exact adjacent patch data) patches are found). Although, such type of the source data is difficult to obtain, it is visually sufficient to have even a small data sample. The dominant component that determines the quality of the appearance of the synthesized result tend to be both major pure DTs.

For each area labeled as a border (labeled '??' in Fig 6, right) the best fitting patch from border texture is found by minimizing the error between this sample itself and each already completed adjacent sample (the one patch for every location or one patch for every type of surroundings DT). Although, the second analytical phase is added, the analytical and synthesis step are still separated. The border patches can be found for each location or every type of location (i.e., B type texture on left, A type texture or right and border texture itself up and down) which drastically reduce computational time.

In this different approach, the number of possibly synthesized textures by one type is due to non rectangular shape at one DT type area

$$N_i = N^*_{in\_tex} * (N_{patches})^{n\_patch} \quad , \tag{5}$$

where n_patch is number of (even incomplete) places for patches of type i type of DT.

$$\prod_{i=1}^{DT_{count}} N_i + N_{border} \quad , \tag{6}$$

where DT_{count} is count of DT type in the synthesized result and N_{border} is total number of possible combinations of textures border which is typically small (one or two patches for one border patch type).

The detection of border texture (see Fig. 7) is similar to finding location of the optimal triple toroid-shaped tile but it needs additional computation. In some cases the minimizing function (1) is not satisfying (imagine two textures which differs in color but are similar in dynamics and structure i.e.textures of leaves or grass with very different color. The borders of leaves are always dark, so by minimizing cut lines error the false result can be found). We add color

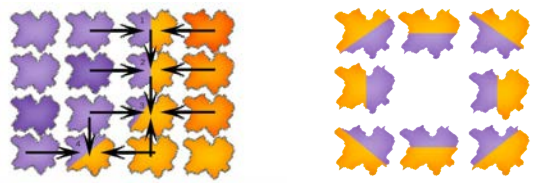


Figure 8: A scheme of a border texture with 8-neighborhood (right). Border texture patches are computed top-down and new border patches are computed from surroundings patches.

pyramid of the whole overlap areas $(1,...,h) \times (1,...,M)$. Many different textural features (like LBP, Haralick and many others) can be used in the similar way.

Finding of border patches can be done either online or offline. First the sufficient numbers of patches for all DT type are found (see labels A1, A2 and B1, B2, B3 in Fig. 7 right) and then the borders textures are applied. In offline version border patch for all possible combination of input DT are found (see Fig. 8 for example of 8-neighborhood border patches). The border patch are fitted to the previously found patches by minimizing (1) and the temporal cut cost (for details see [Haindl and Richtr 2013]) with the set of proper input textures. In the online version a new border patch (see Fig. 8, left) is found for every location in arbitrary order from surrounding patches (even border texture patches found by previous step). Many alternatives of founding the patches are possible, e.g., weighted by number of already know neighborhood patches. This allows greater variability of results, but violates the separation of analysis and synthesis. If the assumption of temporal homogeneity is used, we can presume that temporal optimal overlap can by found in analysis part as preprocessing for border texture (as well for another input textures) and space borders can be found later, in synthesis. If the complete temporal cut is found, the spatial cut and fitting can be found in tens of seconds.

3.3 Temporal Dimension Editing

The DTs mixing and editing can be done directly in temporal dimension too. The input textures can be manually labeled or divided by global and local dynamics (e.g., strenght of wind, rain density, etc.). The patches are found and divided to smaller patches by this temporal labels while preserving the toroidal transition (e.g., NO RAIN ⇒ BEGINS TO RAIN as can be seen in Fig. 9, left) similar to a Markov chain or FSM. This temporal editing and mixing technique can produce high visual quality results, but it is very demanding on the quality of input data.

In order to avoid the typical synthesizing problem - periodicity (given by the small amount of input data) - we use the spatial editing methods: multiple patches, several input DTs, mix of DTs. Besides editing of the spatial dimension to an increase variability, we use also temporal methods. The same samples only slightly time-shifted, but with different dynamics, disrupt the overall impression of their periodicity. Simple random time shift of one part of DT can cause many inconsistencies in synthesized result. The optimal approach would be a segmentation (consistent in all frames), detection of an object and changing only its dynamics. Our approach allows to skip the segmentation step, since the texture is divided into logically consistent patches (given by spatial similarity and similar dynamics) from previous analysis steps and edit the patches separately, but in the border of patch, we change the time factor gradually. We first generate only time indices of pixels and

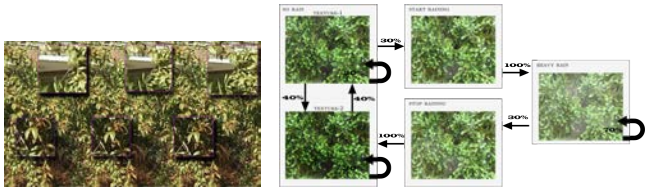


Figure 9: Synthesized texture with three identical, time shifted patches, detail are zoomed. The texture is consistent in spatial and temporal dimensions, but vary in local dynamics (left). Example of one DT labeling with probabilistic state changing rules (right).

then the convolution with the Gaussian kernel is done in order to soften the time change.

Time shift of patches are applied with maximal change to base time by neighbor patch (Fig. 4, right) to satisfy consistency of dynamics of the texture. Of course the temporal shift of patches must satisfy global dynamic condition (which is usually satisfied).

4 Results

The proposed method was extensively tested on DTs from the DynTex [Peteri et al. 2010] database (See some results in Tab. 1). The all used DTs were color with resolution up to HD 720 and varying length (from seconds to minutes). The analysis usually takes from minutes to hours and it strongly depends on the length and the self-similarity of a texture (due to the branch and bound method). The synthesis time is negligible and it is only limited by memory operation and the video storing/coding operations. Our approach can work with various DTs types but it has also some limits. Strong perspective distortions, non periodical color gradient or chaotic local dynamics might degrade the resulting DT quality.

Time dimension edit We have demonstrated the ability to edit the time dimension of the texture to increase its variability and to suppress its repetition impression (see Fig. 9). The time variability can be enhanced by using more time jumps in one type of DT. The output texture can use a graph-like structure for labeled time sections editing.

Video editing (Inpainting) Since our solution is able to create arbitrarily large textures from a relatively small sample of data it can be used to repair corrupted sections of a DT (see Fig. 10). Any synthesized texture can be made artifact free simply by ignoring patches with these artifacts. This can be used for example in editing video containing a DT. The DTs area in a texture is simply replaced by the (different or the same) synthesized DTs or only minimal sufficient number of patches are used to cover the unwanted DT part.

Mix of textures We have demonstrated the capability of creating the mix of DTs from several input DTs. The similarity in the structure of DTs allows usage of the same patch shape and the similarity in the DT type guarantees similar dynamics. Small differences in DTs appearance can be handled by the color tones mapping, which can also be used to intensify the variability of the output (e.g., adding a faint shadow, slightly different flower color, etc.). If the DT types varies strongly in its structure, the border texture (if exists) can be used to connect both DT types in the synthesized results. Tab. 2 shows examples of the DTs mix. The small pictures are input DTs, the large image is the synthesized result. GRASS2 and GRASS3 show random mixing from two and three input DTs, respectively. The SHRUBS and FENCE show two DT types with very different structures and/or color. TREE IN THE RAIN show a mix of DTs with exactly the same scheme on Fig. 8. All input textures are in the HD 720 format, uncompressed and from our own DT database.

Figure 10: Example of simple video editing - the area with 'damaged' grass (top, left) is repaired by filling whole texture by another texture.

Mix of textures in input texture Our method can easily handle input textures containing several DTs types. In these cases, our algorithm selects only one type of DT. This is caused by the toroidal property, which is usually fulfilled only from patches of the same type of DT.

5 Summary

We have proposed a new approach for the fast 4D dynamic multispectral textures synthesis and editing based on the toroid-shaped tiles. The synthesized examples illustrate good performance of our approach for many types of DTs. Our approach can edit temporal and spatial properties of DT and can easily create a mix of several dynamic textures. We have validated our solution on several DT modeling problems. The method is simple. It can generate infinitely long textures in all dimension. Its most important advantage is its complete separation of the analytical and synthesis parts, thus the synthesis can be performed effectively in real-time.

Acknowledgments

This research was supported by the Czech Science Foundation project GAČR 14-10911S.

References

BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2001. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics 7*, 120–135.

BAR-JOSEPH, Z. 1999. *Statistical learning of multi-dimensional textures*. PhD thesis, The Hebrew University.

CHAN, A. B., AND VASCONCELOS, N. 2005. Mixtures of dynamic textures. In *10th ICCV 2005), 17-20 October 2005, Beijing, China*, IEEE Computer Society, 641–647.

Table 2: DT mix: GRASS2 and GRASS3 are synthesized by the random tile arrangement. FENCE, SHRUBS and TREE IN THE RAIN are mix of DTs with border texture. Smaller (scaled) figures are input DT, the larger figures are the synthesized outputs.

GRASS2 GRASS3 FENCE SHRUBS TREE IN THE RAIN

CHAN, A. B., AND VASCONCELOS, N. 2006. Layered dynamic textures. In *Advances in Neural Information Processing Systems 18*, MIT Press.

CHAN, A. B., AND VASCONCELOS, N. 2008. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *IEEE Tr. PAMI*.

CHIBA, N., MURAOKA, K., TAKAHASHI, H., AND MIURA, M. 1994. Two-dimensional visual simulation of flames, smoke and the spread of fire. *The Journal of Visualization and Computer Animation 5*, 1, 37–53.

CONSTANTINI, R., SBAIZ, L., AND SSSTRUNK, S. 2006. Dynamic texture synthesis: Compact models based on luminance-chrominance color representation. In *International Conference on Image Processing*, 2085–2088.

COSTANTINI, R., SBAIZ, L., AND SSSTRUNK, S. 2008. Higher order svd analysis for dynamic texture synthesis. *IEEE Transactions on Image Processing*, 42–52.

DORETTO, G., CHIUSO, A., WU, Y. N., AND SOATTO, S. 2003. Dynamic textures. *Int. Journal of Computer Vision 51*, 2, 91–109.

EDELMAN, S. 1992. Visual perception. *Encyclopedia of Artifical Intelligence*, 2:1655–1663.

FAHLE, M., AND POGGIO, T. 1981. Visual hyperacuity: spatiotemporal interpolation in human vision. *Proceedings of the Royal Society of London. Series B. Biological Sciences 213*, 1193, 451–477.

FILIP, J., HAINDL, M., AND CHETVERIKOV, D. 2006. Fast synthesis of dynamic colour textures. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 4, 25 –28.

HAINDL, M., AND FILIP, J. 2013. *Visual Texture*. Advances in Computer Vision and Pattern Recognition. Springer-Verlag, London.

HAINDL, M., AND HATKA, M. 2005. A roller - fast sampling-based texture synthesis algorithm. In *The 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2005*, University of Western Bohemia, Plzen, V. Skala, Ed., University of Western Bohemia, 80–83.

HAINDL, M., AND RICHTR, R. 2013. Dynamic texture enlargement. In *Spring Conference on Computer Graphics*, ACM, New York, NY, USA, SCCG '13, 005:5–005:12.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. 22*, 3 (July), 277–286.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, N. 2005. Texture optimization for example-based synthesis. *ACM Transactions on Graphics, SIGGRAPH 24*, 795–802.

LIZARRAGA-MORALES, R. A., GUO, Y., ZHAO, G., PIETIKINEN, M., AND SANCHEZ-YANEZ, R. E. 2014. Local spatiotemporal features for dynamic texture synthesis. *EURASIP Journal on Image and Video Processing 2014*, 1.

PERRY, C. H., AND PICARD, R. W. 1994. Synthesizing flames and their spreading. In *Proceedings of the Fifth Eurographics Workshop on Animation and Simulation*, 1–14.

PETERI, R., FAZEKAS, S., AND HUISKES, M. J. 2010. Dyntex: A comprehensive database of dynamic textures. *Pattern Recognition Letters 31*, 12, 1627 – 1632.

PHILLIPS, P. M., AND WATSON, G. 2003. Generalising video textures. In *Theory and Practice of Computer Graphics 2003 (TPCG 2003), 3-5 June 2003, Birmingham, UK*, 8–15.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, SIGGRAPH '00, 489–498.

SOATTO, S., DORETTO, G., AND WU, Y. N. 2001. Dynamic textures. In *iccv*, vol. 2, 439–446.

STAM, J., AND FIUME, E. 1993. Turbulent wind fields for gaseous phenomena. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '93, 369–376.

SZUMMER, M., AND PICARD, R. W. 1996. Temporal texture modeling. In *IEEE Intl. Conf. Image Processing*, vol. 3, 823–826.

TIAN, G. Y., GLEDHILL, D., TAYLOR, D., AND CLARKE, D. 2002. Colour correction for panoramic imaging. In *Information Visualisation, 2002. Proceedings. Sixth International Conference on*, 483–488.

ZHU, S. C., WU, Y., AND MUMFORD, D. 1998. Filters, random fields and maximum entropy (frame) – towards a unified theory for texture modeling. *INTERNATIONAL JOURNAL OF COMPUTER VISION 27*, 2, 1–20.