# Dynamic Texture Enlargement

Michal Haindl* Radek Richtr†
Institute of Information Theory and Automation of the ASCR, Prague, Czech Republic
Faculty of Information Technology, Czech Technical University

Figure 1: Synthesis result from several dynamic textures.

## Abstract

A simple fast approach for dynamic texture synthesize that realistically matches given color or multispectral texture appearance and respects its original optic flow is presented. The method generalizes the prominent static double toroid-shaped texture modeling method to the dynamic texture synthesis domain. The analytical part of the method is based on optimal overlapping tiling and subsequent minimum boundary cut. The optimal toroid-shaped dynamic texture patches are created in each spatial and time dimension, respectively. The time dimension tile border is derived from the optical flow of the modeled texture. The toroid-shaped tiles are created in the analytical step which is completely separated from the synthesis part. Thus the presented method is extremely fast and capable to enlarge a learned natural dynamic texture spatially and temporally in real-time.

**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:** dynamic texture, texture enlargement, tile-based texture synthesis

## 1 Introduction

Texture enlargement is crucial for numerous computer-based visualization applications because whatever size of the measured textural data is, it is always insufficient. The available material size is either too small if we measure material of real existing objects for rendering of a complex and large virtual scenes or the measurement technology does not allow 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 (BTF) [Haindl and Filip 2013]. The amount of measured BTF data can be extremely high, e.g., in the range of tera bytes, even for such spatially/dynamically restricted measurements thus any visual texture modeling method inevitably requires simultaneously some compression capability.

Dynamic texture (DT) rendering requires to solve the problem of seamless spatial and temporal enlargement (modeling, synthesis) of measured dynamic textures. Dynamic or static texture editing is another useful application which allows to synthesize alternative and unmeasured types of DTs, recreating sequences for studying of relationship between model parameters and visual appearance of the scene, etc. The texture synthesis aims to create a visual texture which is perceptually similar to the target texture. A synthesized texture should ideally produce such a visual perception so that original measured texture and the synthetic one are visually indiscernible. Unfortunately, natural texture modeling is a very challenging and difficult task, due to unlimited variety of possible surface materials, illumination and viewing conditions, simul-

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

taneously with the strong discriminative functionality of the human visual system. Also observations-nuisance-factors like artifact of unwanted regularity must be avoided.

### 1.1 Dynamic texture

Although, no rigorous dynamic texture definition exists [Haindl and Filip 2013], DT can be characterized by some spatially and temporally invariant statistics [Zhu et al. 1998] (i.e., there must be both an inter-frame spatial and between-frame temporal homogeneity). So the dynamic texture can be defined as as realization of 4D (spectral, temporal, and 2 spatial dimensions) stochastic random field.

Similarly to static texture modeling, also dynamic textures can be synthesized using two main approaches. One approach is based on a mathematical model and requires to store only its parameters, while the other option is to use an intelligent sampling from the original measured DT data part of which have to be stored.

The model-based dynamic texture synthesis using parametric spatiotemporal causal auto-regressive model and restricted to gray-scale DT only was studied Szummer [Szummer 1995; Szummer and Pickard 1996]. Later Doretto [Doretto 2002; Doretto 2005; Doretto et al. 2003] and Soatto [Soatto et al. 2001] presented auto-regressive moving average process based on SVD. This approach contains time consuming iterative gradient method for paremeters estimation, but allow to editing synthesized texture by changing of model parameters. Fast synthesis of dynamic colour textures was presented by Filip [Filip et al. 2006] which overhelmed time consuming previous methods. Chan [Chan and Vasconcelos 2005], Constantini [Costantini et al. 2006] [Costantini et al. 2008] who use tensor decomposition and SVD techniques and others [Vidal and Ravich 2005][Liu et al. 2005] presented similar approaches. Since parametric model are very general and usable for recognition [Saisan et al. 2001] they can not model many type of dynamic and video textures. DT textures can be modeled also as a mix of several dynamic textures[Chan and Vasconcelos 2008; Chan and Vasconcelos 2005]. These types of textures can be created by layered dynamic textures [Chan and Vasconcelos 2006].

Sampling methods [Schodl et al. 2000; Bar-Joseph et al. 2001; Kwatra et al. 2003; Phillips and Watson 2003], which are prevailing in computer graphics applications, are based either on simple dynamic texture repetition with edge blending or on more or less sophisticated video block tiling methods. Different sequences of frames taken from the original video are repeated to form a longer (possibly infinite) sequence, ensuring that transitions between consecutive blocks are not noticeable. This requires to find minimum-error transition cuts and to use adequate morphing techniques to diminish inevitable discontinuities between blocks. The simple temporal enlargement method only [Schodl et al. 2000] uses frames similarity and optical flow to find parts with similar motion which are subsequently combined using a blur technique. The method is applicable to limited number of DT because it is not always possible to find sufficiently similar frames to create a DT loop without optical distraction. The method [Bar-Joseph et al. 2001] can create only time limited sequences.

We present a new approach to generate possibly infinitely large spatially as well as temporally dynamic texture from a fixed size significantly smaller dynamic texture without help of the blurring or morphing techniques. This is achieved by a sophisticated composition of several triple toroid-shaped tiles, which can be seamlessly repeated in every direction. By this way we can create a significantly larger pattern with similar stochastic properties. Our algorithm can finds toroidal patches in the input texture such that after giving patches abreast the borders between patches are visually indiscernible (seamless). This intelligent sampling allows to retain a small enough 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.

## 2 Toroidal-Based Texture Enlargement

The presented method principle is to find optimal triple toroid-shaped dynamic texture tiles which can be subsequently seamlessly repeated in all data space directions spatial as well as temporal. Its scheme is illustrated on Fig.2. The method estimates several optimal triple toroid-shaped dynamic texture tiles combining estimated optical flow and double toroid-shaped dynamic texture frame tiles. Our method can be used to enlarge both 3D (static colour/multispectral textures) or 4D (dynamic multispectral) textures. The principle of spatial and temporal cuts creating is similar, but since the temporal dimension has different visual perception properties both spatial dimensions, we have developed two distinct approaches for the spatial and optical flow driven temporal cuts, respectively. Its justification is easy seen from the spatial and temporal property of dynamic textures - a human perceives and detects differently errors in the space and time domain [Edelman 1992]. Both spatial dimensions have similar property, but the temporal dimension differs [Fahle and Poggio 1981]. While the spatial dimensions may be interchanged and texture still keeps it structure, the temporal dimension has distinct property (unless some exceptional procedural 4D material textures) and can not be swapped with another spatial coordinate.

### 2.1 Double Toroid-Shaped Tile

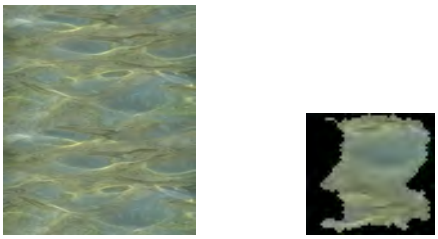


Figure 3: The double toroid-shaped tile (right) and its source frame texture.

The toroid-shaped tile (see Fig. 3) 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.

The optimal tile cut 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 (Fig. 2) is to find some relevant and sufficiently representative tiles from the source 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 lattice of the tile label.

The tile edges (patches) can be found by some of existing algorithms [Boykov et al. 2001; Greig et al. 1989; Kwatra et al. 2003], but these methods do not respect distinctions between spatial and

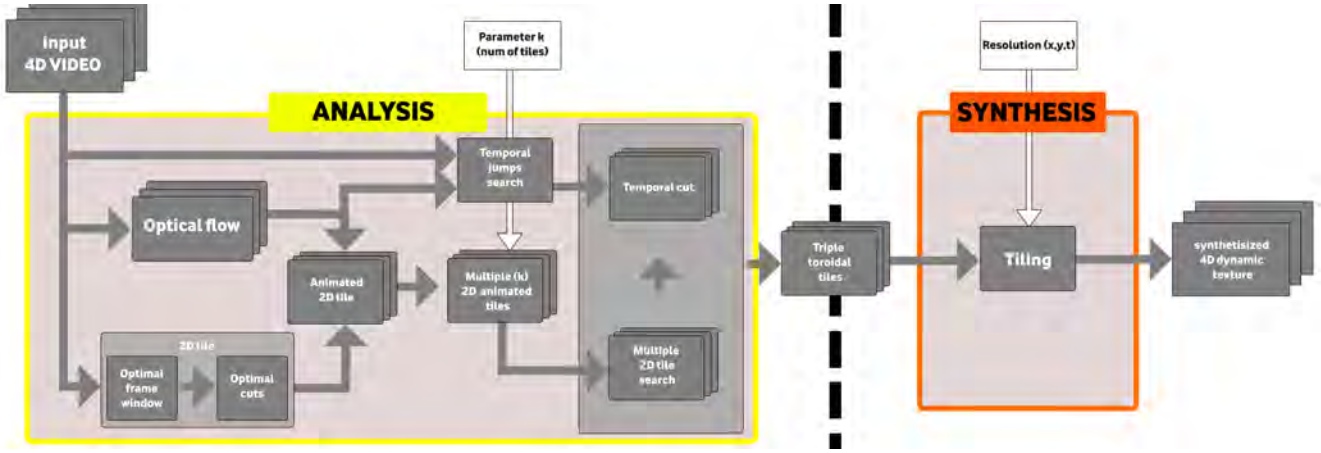



Figure 2: The overall flowchart of the presented method.

temporal dimensions. We use the roller method approach [Haindl and Hatka 2005] to find the static double toroid-shaped tiles (Fig. 3) and to combine them with their optical flow controlled propagation throughout the time. The tiles are fitted to local dynamics and subsequently expanded into the triple toroid-shaped forms. Let us define the overlap error for a multispectral pixel vector $Y_r$ as follows:

$$\psi_r^h = |Y_r - Y_{r+[N-h,0,\bullet,0]}|_2^2 \qquad \forall r \in I_h \ , \qquad (1)$$

$$\psi_r^v = |Y_r - Y_{r+[0,M-v,\bullet,0]}|_2^2 \qquad \forall r \in I_v \ , \qquad (2)$$

where the $p = 2$ vector norm is defined as

$$|X_{r_1,r_2,\bullet,r_4}|_p = \left( \sum_{r_3} |X_{r_1,r_2,r_3,r_4}|^p \right)^{\frac{1}{p}} \ ,$$

the multiindex $r$ has four components $r = [r_1, r_2, r_3, r_4]$, where the components are row, column, spectral, and frame indices, respectively. $\bullet$ denotes all values of the corresponding index, $I_v$ is equal to the frame area $(1,...,h) \times (1,...,M)$, and $I_h$ is equal to $(1,...,N) \times (1,...,v)$. The optimal horizontal and vertical overlaps are found from the following equation:

$$\zeta_r^* = min_\zeta \left\{ \frac{1}{\lfloor \zeta \rfloor} \sum_{\forall r \in I_\zeta} \psi_r^\zeta \right\} \ , \qquad (3)$$

where $\lfloor \zeta \rfloor$ denotes the the corresponding overlap area of $\zeta$.

The optimal size of tile must be first found. This can be done simply by minimizing the overlap error $\zeta_r^*$ (see (3) or [Haindl and Hatka 2005]) by the Fourier transformation [Haindl and Hatka 2009] to find the dominant texture periodicity (with the weight in every frame, or in one representative frame).

## 2.2 Dynamic Double Toroid-Shaped Tile

The located single frame double toroid-shaped tile found is propagated along the optical flow to subsequent frames to model the textural structures movement. The optical flow is used to represent a dynamic texture global dynamics. The propagated double toroid-shaped tile in every frame serves as the initialization for local modification of such a double toroid-shaped tile to each frame-specific final shape.

An optimal location for the tile is computed by back-propagation dynamic-programming-like rules (4)-(7) which minimize both spatial $(\Psi_r^{h+}, \Psi_r^{v+})$ and temporal $(\Psi_r^{h*}, \Psi_r^{v*})$ cut cost:

$$\psi_r^{h*} = \psi_r^{h*} + \min \left\{ \psi_{r-[c,0,\bullet,1]}^{h*}, \ldots, \psi_{r+[c,0,\bullet,-1]}^{h*} \right\} \ , \qquad (4)$$

$$\psi_r^{v*} = \psi_r^{v*} + \min \left\{ \psi_{r-[0,c,\bullet,1]}^{v*}, \ldots, \psi_{r+[0,c,\bullet,-1]}^{v*} \right\} \ , \qquad (5)$$

$$\Psi_r^{h+} = \psi_r^{h+} + \min \left\{ \psi_{r-[c,1,\bullet,0]}^{h*}, \ldots, \psi_{r+[c,-1,\bullet,0]}^{h*} \right\} \ , \qquad (6)$$

$$\Psi_r^{v+} = \psi_r^{v+} + \min \left\{ \psi_{r-[1,c,\bullet,0]}^{v*}, \ldots, \psi_{r+[-1,c,\bullet,0]}^{v*} \right\} \ . \qquad (7)$$

The parameter $c$ is derived by the average of the optical flow from the actual frame to the next frame. 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 (e.g., moving shore video). In this case the back-propagation is used do find parts inconvenient of the texture (i.e., parts of the dynamic texture with not sufficiently long patches). It can also override problems with errors or textural artifacts. A simple soution can be found by a double toroid-shaped tile moving along the optical flow and to replicate the textural structure movement. This solution represents two of the three main scene features. The global dynamics (scene or large objects movement, e.g., whole trees) is represented by the optical flow, and the texture itself by the double toroid-shaped tile. However, local dynamic (like leaves or grass blades shivering) cannot be sufficiently represented this way.

The local dynamics problem can be solved by the iterative border elaboration to minimize the overlapping border error (see Fig. 4). The variable $c$ is used to determine structure size with respect to shape of the original approximate border.

These restricted search areas extremely speed up the computation time needed to find the optimal tile shape in comparison to search the whole textural space. These areas are beneficial also in avoiding some typical DT modeling problems like artifacts introduced by extremely varying speed or problems with highly dynamic texture, e.g., fire or water details. In these areas optimal fitting planes can be found by the algorithm described below. This can be done by whole recreating of tiles, or by building and traversing of the recomputed

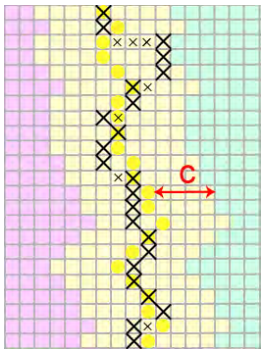



Figure 4: Toroidal boundary location for the simple border model. Yellow dots represent border initialization, while the yellow area is the area where the precise location is searched for. Crosses represent the exact boundary sweaty only spatial errors, commplementary points (small crosses) must be added to make the border continuous. The parameter $c$ is drived by the actual optical flow.

(to overcome blind and unwanted states) structure. In the fitting step the overlap error is computed only by actual spatial data:

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

Spatiotemporal patches construction:

1. Find the initial double toroid-shaped tile.
2. Move the tile by the optical flow.
3. Compute searching areas from the tile and flow.
4. Find exact spatiotemporal planes
   a) By findind sets of independent cuts (for high dynamic data) or
   b) By propagating the tile thought the time dimension with keeping the structure of cuts (reverse rule to building the data structure)

The major advantage of our approach is the complete separation between the analysis 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 step is completely separated.

Using only one double toroid-shaped tile per frame can produce observable false spatial periodicity for some textures. This problem can be easily eliminated by using of multiple per frame tiles (Figs. 7,8). Multiple tiles require identical or very similar optical flow and the temporal shape development. This is can be a challenging problem.

## 2.3 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[Schodl et al. 2000]. In fact, without the spatial enlargement functionality, the temporal cut part of our method resembles the Kwatra[Kwatra et al. 2003] algorithm or in extreme cases (highly blurred/noised texture) the Schodl[Schodl et al. 2000] method.

The triple toriod-shaped tile construction from the double toroid-shaped video tile is not trivial. The spatial border toroid property of the sample must be maintained by this temporal cut (as can be seen in Tab. 1 (b),(c)). Here the temporal and *any* spatial dimensions are represented by the horizontal and vertical direction, respectively. The first two rows show *the same* texture with the cut in it. The third row shows textures synthesize in the temporal dimension. The fourth row presents synthesis in the temporal and spatial dimensions. The red dotted line shows the cut where the spatial continuity is violated. It can be easy seen, that if the temporal cut is not double toroid-shaped the border violates the required spatial homogeneity.

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 Schodl[Schodl et al. 2000]. The minimized error can be chosen as the pixel-to-pixel difference of frames or by some other metric [Doretto 2005; Kwatra et al. 2005].

The cost of the whole (permissible) temporal cut is minimized (see 8) so as to preserve the spatial properties of the toroid. We define the temporal overlap error as follows:

$$\Psi^{t*} = min_{\Psi}\{\sum_{\forall r_1,r_2} \psi_{r,m}^{t*}\} \quad (8)$$

where $t$ is the unknown temporal coordinate of the first most similar frame, $m$ is the distance between the most similar frames.

$$\psi_{x,y,t,m}^{t} = D^m(Y_r^g, Y_{r+[0,0,m]}^g) \quad (9)$$

where $D$ denotes distance operation with metric $m$ between operands of $D$. Maximal change of $t$ between 8-neighborhood of $Y_r$ and $Y_{r_1+i,r_2+j,r_3}$ is $c_t$.

### 2.3.1 Temporal Cut

Rather than finding a single transition time for the whole image our algorithm cuts the pair of the most similar frames to two (or more) parts and overlay them. Next cuts are computed iteratively. In each step the algorithm is looking up to four cuts (every cut generates two half-planes and can have previous and next frame) and use one cut in every half-plane. This result is very similar to determining the time transition on per-pixel basis, but the local spatial properties of the texture are keep. By this, infinitely long sequence of a texture can be found and because the patch satisfy the global texture dynamic this simple approach can work with it.

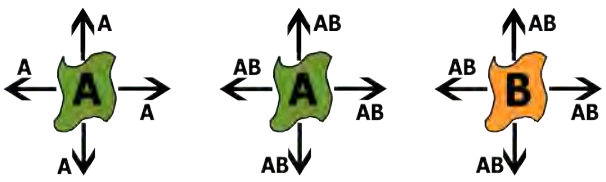
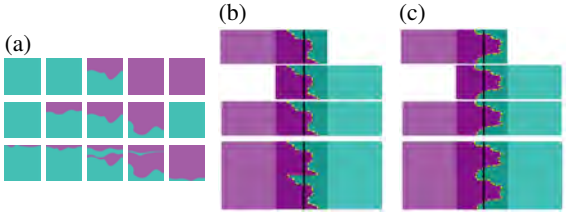



Figure 5: A standalone double toroid-shaped tile (left) and multifitting double toroid-shaped tiles (middle and right) and their possible neighbors.

Table 1: (a) Every row represents one iteration of fitting the transition plane; each column represents five different frames : $t-2, t-1, t, t+1, t+2$; all pictures is one frame of a texture where color represents offseted and original data; the first row - one cut that put together two most similar frames; the second row - in each halfplane of the next and previous frames another soft cut is found; the third row is final result of another iteration. It presents the ability to choose to change slope (b) Illustration of cuts for temporal synthesis without spatial control (c) Illustration of cuts for temporal synthesis with spatial control




The problem of this approach can be viewed on Tab. 1(b). However, it can be easily overcomed by enforcing the start and end coordinates to be the same, to run path search algorithm $o$ times and to find the best possible suboptimal path (Tab. 1(c)). The second toroidal property (top and down border on Tab. 1(a) are minimized by building a tree of all potential cuts and then traversing them to minimize the temporal and this boundary error (denoted as weighted difference of $c_2$ coordinates in the top and bottom row). This problem is not relevant in the case that the texture is synthesized in one spatial dimension only.

### 2.3.2 Plane Fitting

In some cases the previous approach for temporal cut is not satisfying. Every single cut is computed in one frame and derived by only *spatial* data but in fact it drives the *temporal* offset. If the temporal dynamic of the texture are significantly greater than the local spatial dynamic the temporal cut plane can be computed differently. Like in set-of-cuts the problem is to create a cut with no boundary spatial errors. This can be done by using a similar procedure like in propagating the double toroid-shaped tile. The structure for computing the offset error is created, but the time axis is changed to one spatial axis and the plane which minimize the overall error are searched by traversing a line by the structure (with keeping of consistency and the double toroidal property).

This is better to the texture with low local spatial dynamics like leaves or grass in string wind, flowers or leaves. As can be seen in Fig. 6 the planes generated by this approach are more general, but satisfy the toroidal boundary rules and are double toroid-shaped like a spatial patch (this can be easy done to setting error cost of ending not wanted coordinates to infinity before computing the structure).

Figure 6: Each picture represent one frame in texture left-right and top-down respectively; all pictures is one frame of texture where colors represents offseted and original data; you can see the general shape of the plane and its double toroidal property

## 3 Results

The proposed method was extensively tested on dynamic textures from the DynTex[Péteri et al. ] database with exceptions of some dynamic textures used for comparison with alternative methods. These colour textures have spatial resolution $720 \times 576$ pixels and varying length (from seconds to minutes).This database contains a large number of various texture types, including grass, water, animals, smoke and other natural or artificial scenes. Our synthesis output does not depend on a type of input texture, but better results were observed on textures containing sharp edges (trees, leaves). The analysis time varies from minutes to hour given by the length of an input sequence and the size of patches (if patches are large, the time-cut part of algorithm requires more time), while the synthesis time is negligible.

Tab. 4 compares our results with results previously published by Kwatra [Kwatra et al. 2003], Schödl [Schodl et al. 2000], and Bar-Joseph [Bar-Joseph et al. 2001]. All textures are 4D, but have extremely small resolution and artifacts created by using the lossy format (which works like automatic blur of seams) instead of uncompressed format which we used for all other results.

The first example in Tab. 4 is the CLOUDS texture (resolution is 128x128x59 by horizontal, vertical and time axis respectively). This is example of the texture where our algorithm does not work well, the clouds are unique and texture is extremely small to find perfect toroid-shaped tile and the edge is visible in some cases (unless some blending method is used). The GRASS example texture (resolution is 224x114x59 by horizontal, vertical and time axis respectively) is presented. The texture of SPARKLE (resolution is 240x160x150 by horizontal, vertical and time axis respectively) is easy to create spatial cuts, but difficult to create the temporal one (due to uniques of texture frames). You can see mistakes on both our (detailed) and Kwatra result. The last comparision example is the Kwatra and our result of the synthesized WATER texture. The temporal cut is presented in at least one of our presented examples.

In the last row of Tab. 3 it is possible to see the interesting result of our approach. The first row contains synthesized results created with two patches. On this texture CREEK is a stream of water 'springing' from stones. This is due the algorithm used no syntactic information. In second row you can see result with using more patches where you can see two 'rivers' (In such cases it may be a type of sample set manually.).

**Temporal cut** Contrary to the Bar-Joseph [Bar-Joseph et al. 2001] method which can create only time limited sequences, our approach can generate infinitely long dynamic textures. By using of the optical flow, we can create textures which are 'constantly moving' like panoramic slide with the camera. In some cases this can cause unexpected results - for example the SPARK texture is constantly drifting due to optical flow generated by electric flashes (this global move can be of course handled and disabled by the type of texture).

**Local dynamic** If only local dynamics present in the texture our approach works very well and is able to synthesize a wide range of natural DT types (see Tab. 2). Some DT like a smoke from a fire or a river, which have seemingly mostly local dynamic have in fact strong global motion.

**Global dynamics** When some global motion (i.e. moving camera view) is present in the texture the synthesis with the temporal offset is not satisfactory and the entire patch must be spatially moved. This can be detected by the optical flow of the texture.

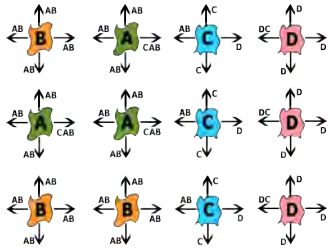


Figure 7: Four types of tiles putting together

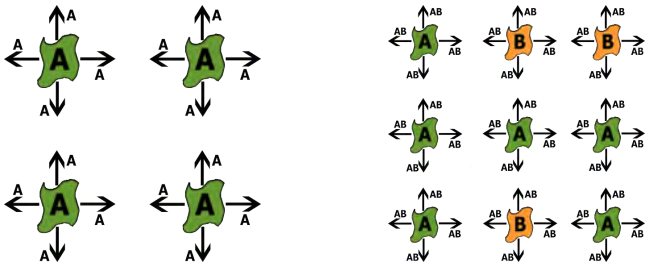


Figure 8: Synthetic texture from one tile type (left) and two tile types (right).

**Textural mixture** Our method can easily handle input textures containing mixtures of several dynamic textures. In these cases, our algorithm selects only one type of texture and create output containg only samples from one texture type. This is caused by minimizing the error at the edge of the patch, which is usually fulfilled only by samples of the same type of DT. This property is illustrated on the SHORE texture. One way how to overcome this property is to create various texture type tiles with different shapes or to compute special boundary types as can be seen on Fig. 7 where *A* and *B* represent water, *D* grass and *C* shore.

**Distortions** Typical problems in the synthesis of the dynamic texture is the perspective distortions or view rotation. The synthesis of such scene types is difficult. Typical manifestation of this problem is not perfect vertical and eye-evident seam. The similar effect is present if smooth color or illumination transition (typically the sky) is present in the texture as can be seen on SMOKE texture. Naturally, there are many dynamic texture types, which have different horizontal consistency and the horizontally cut in it has no real sense (blasting flames, flag, panoramic view) or it will need more patches to maintain it (top part of flag, patches of fluttering flag and its lower part).

**Highly dynamic texture** Some DT changed its structure extremely fast from frame to frame. Typical examples are flames (see FIREA and FIREB on Tab. 3), where the single flash of fire disappear and raised during small number of frames or faster than the capture frequency. This can be often overcomed by manually setting of large areas for finding spatial and temporal cut by set-of-cuts as can be seen on picture below. The automatic setting of our algorithm sometimes fails if there are some continually moving parts (flames going up) and still parts (bottom of flame) in the texture. It can be seen that the synthesized picture looks like very good result, but gradually degrades how flames changed. However, after some optically wrong frames, the algorithm finds new local optimum, changes the cut plane drastically and again starts from better results (and all repeats). Fortunately the "wrong frames" are often only small part of the texture and can be found and removed without disturbing the whole sequence consistency.

# 4 Summary

We have demonstrated a new approach for 4D dynamic multispectral texture synthesis based on the toroid-shaped tiles. The synthesized examples illustrate good performance of our approach for many types of dynamic textures. We have demonstrated also some dynamic textures modeling problems and our solution for them. The method is simple, it can generate infinitely long textures in all dimension and what is the most important it has completely separated analysis and synthesis parts hence the synthesis can be easily done in real-time and using various tile selection scenarios. In the coming work we will improve our algorithm by computing perspective translation to overcome this problem and augment additional types of texture tiles with different shapes and better options for dynamic textures editing applications.

**Acknowledgements** This research was supported by grants GAČR 102/08/0593, 103/11/0335.

# 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*, 2 (Apr-Jun), 120–135.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 2001.

CHAN, A. B., AND VASCONCELOS, N. 2005. Mixtures of dynamic textures. In *ICCV*, IEEE Computer Society, 641–647.

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 Trans. Pattern Analysis and Machine Intelligence 30*, 5 (may), 909–926.

COSTANTINI, R., SBAIZ, L., AND SUSSTRUNK, S. 2006. Dynamic texture analysis and synthesis using tensor decomposition. In *Advances in Visual Computing*, II: 245–254.

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. *International Journal of Computer Vision 51*, 2 (February), 91–109.

DORETTO, G. 2002. *Dynamic texture modeling*. Master's thesis, University of California, Los Angeles, CA.

DORETTO, G. 2005. *DYNAMIC TEXTURES: modeling, learning, synthesis, animation, segmentation, and recognition*. PhD thesis, University of California, Los Angeles, CA. Committee: Adnan Darwiche, Petros Faloutsos, Demetri Terzopoulos, Ying Nian Wu, Stefano Soatto (Chair).

EDELMAN, S., 1992. Visual perception. Cornell University.

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.

Table 2: First column: the original texture; second column: enlarged in temporal and one spatial (horizontal) dimension; third column: Enlarged in both spatial and temporal dimensions

| Original | X,T enlarged | X,Y,T enlarged |
|---|---|---|

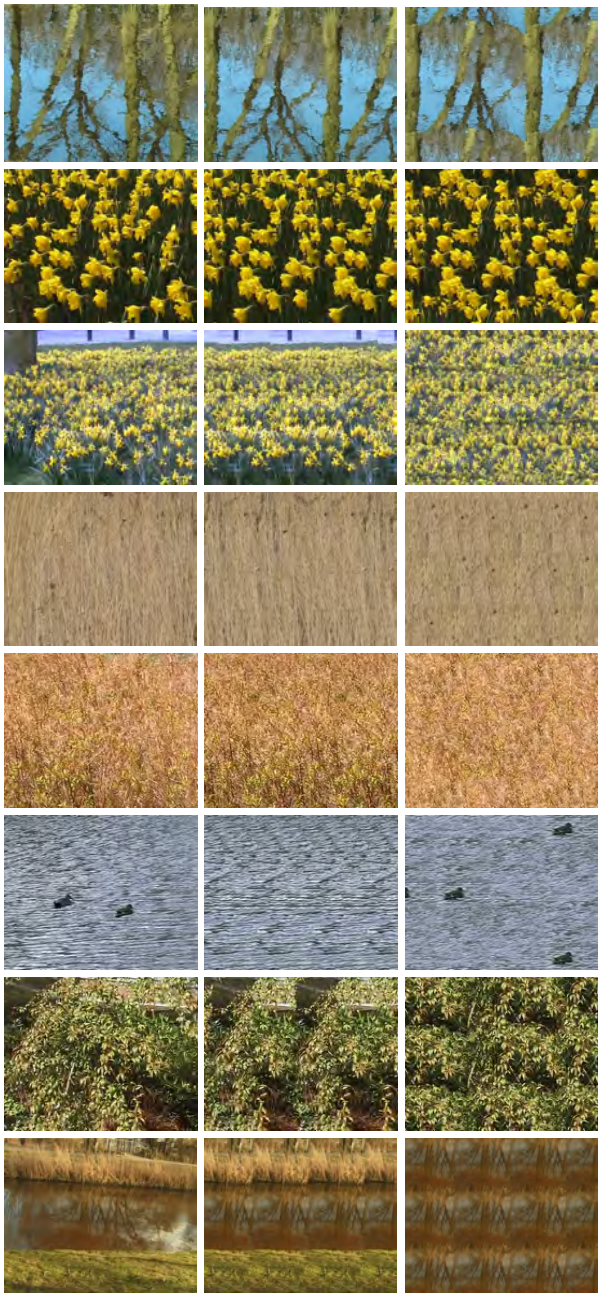
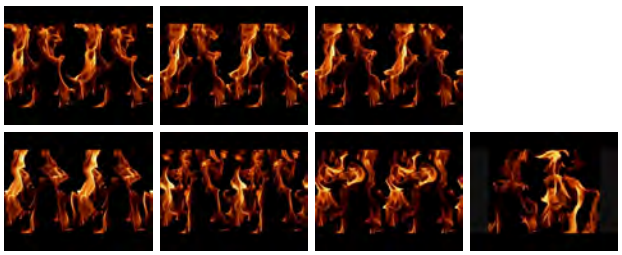
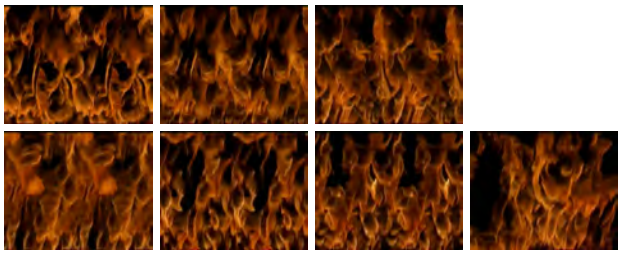


Table 3: Synthesized texture examples. The first three columns show examples of synthesized textures, the fourth column of the original sample texture. First and second rows shows 6 frames of a texture FIREA from left to right and top to down respectivelly, cyclic degrading can be seen. Third and fourth rows shows 6 frames of a texture FIREA from left to right and top to down respectivelly, cyclic degrading can be seen. On sitxth and seventh rows water texture WATERA and WATERB enlargen in all dimensions can be seen. The last two rows shows the CREEK texture with different number of patches.

FIREA

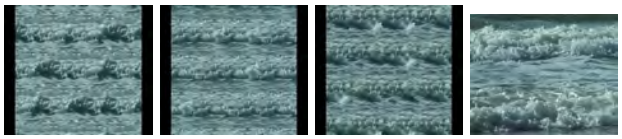


FIREB



WATERA

WATERB



CREEK

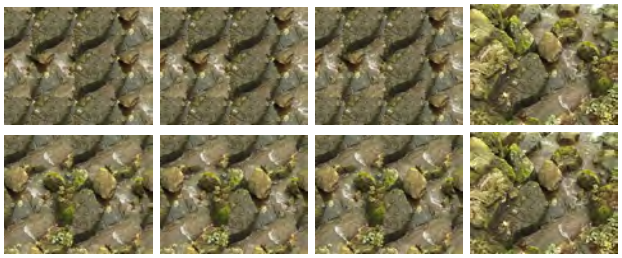
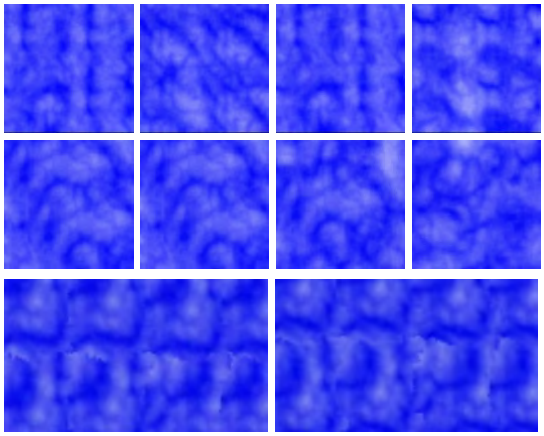


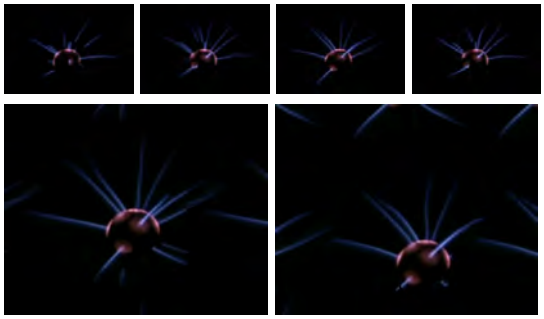
Table 4: CLOUDS: First row Bar-Joseph (temporal only), second row Kwatra (spatiotemporal), third our result (spatiotempotal); GRASS: first row Kwatra (spatiotemporal), second row Schödl (temporal only), third our result (spatiotemporal); SPARKLE: first row Kwatra (temporal), second row our result (spatiotemporal); WATER: first row Kwatra (spatiotemporal), second row out result (spatiotemporal); Last row : one example frame from original textures

CLOUDS

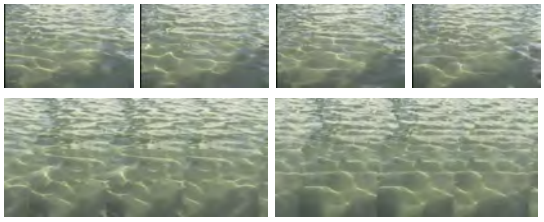


GRASS

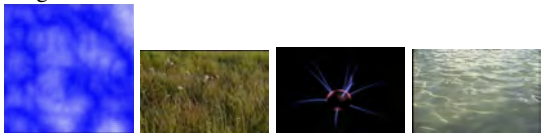
SPARKLE



WATER



Original tex.




FILIP, J., HAINDL, M., AND CHETVERIKOV, D. 2006. Fast synthesis of dynamic colour textures. In *Proceedings of the 18th International Conference on Pattern Recognition, ICPR 2006*, IEEE Computer Society, Los Alamitos, Y. Tang, S. Wang, D. Yeung, H. Yan, and G. Lorette, Eds., vol. IV, 25–28.

GREIG, D., PORTEOUS, B., AND SEHEULT, A. H. 1989. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 271–279.

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

HAINDL, M., AND HATKA, M. 2005. A roller - fast sampling-based texture synthesis algorithm. In *Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, UNION Agency - Science Press, Plzen, V. Skala, Ed., 93–96.

HAINDL, M., AND HATKA, M. 2009. Generalized roller. In *Proceedings of the 17th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2009*, University of West Bohemia, Plzeň, M. Chen and V. Skala, Eds., 73–80.

KWATRA, V., SCHODL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphicut textures: Image and video synthesis using graph cuts. *ACM T Graphic 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.

LIU, C.-B., LIN, R.-S., AND AHUJA, N. 2005. Modeling dynamic textures using subspace mixtures. In *ICME*, IEEE, 1378–1381.

PÉTERI, R., FAZEKAS, S., AND HUISKES, M. J. DynTex : a Comprehensive Database of Dynamic Textures. *Pattern Recognition Letters doi: 10.1016/j.patrec.2010.05.009*. http://projects.cwi.nl/dyntex/.

PHILLIPS, P. M., AND WATSON, G. 2003. Generalising video textures. In *TPCG*, IEEE Computer Society, 8–15.

SAISAN, P., DORETTO, G., WU, Y. N., AND SOATTO, S. 2001. Dynamic texture recognition. In *CVPR*, vol. 2, 58–63.

SCHODL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *ACM SIGGRAPH 2000*, ACM, New Orleans, ACM, 489–498.

SOATTO, S., DORETTO, G., AND WU, Y. N. 2001. Dynamic textures. In *IEEE International Conference on Computer Vision*, IEEE, Vancouver, BC , Canada, vol. 2, IEEE, 439446.

SZUMMER, M., AND PICKARD, R. W. 1996. Temporal texture modeling. In *Proc. IEEE Int. Con. Image Processing (ICIP)*, IEEE, vol. 3, IEEE, 823–826. MIT TR 381 report.

SZUMMER, M. 1995. *Temporal Texture Modeling*. PhD thesis, MIT, Cambridge. TR 346.

VIDAL, R., AND RAVICH, A. 2005. Optical flow estimation and segmentation of multiple moving dynamic textures. In *In CVPR*, 516–521.

ZHU, S. C., WU, Y., AND MUMFORD, D. 1998. Filters, random fields and maximum entropy (frame): Towards a unified theory for texture modeling. *Int. J. Comput. Vision 27*, 2, 107–126.