Novel Path Search Algorithm for Image Stitching and Advanced Texture Tiling

Petr Somol
Dept. of Pattern Recognition
Inst. of Information Theory and Automation
Pod vodárenskou věží 4
182 08, Prague 8, Czech Republic
somol@utia.cas.cz

Michal Haindl
Dept. of Pattern Recognition
Inst. of Information Theory and Automation
Pod vodárenskou věží 4
182 08, Prague 8, Czech Republic
haindl@utia.cas.cz

ABSTRACT

We propose a fast and adjustable sub-optimal path search algorithm for finding minimum error boundaries between overlapping images. The algorithm may serve as an efficient alternative to traditional slow path search algorithms like the dynamical programming. We use the algorithm in combination with novel adaptive blending to stitch image regions. The technique is then exploited in a framework for sampling-based texture synthesis where the learning phase is clearly separated and the synthesis phase is very simple and fast. The approach exploits the potential of tile-based texturing and produces good and realistic results for a wide range of textures.

Keywords

Path Search, Image Stitching, Image Transfer, Adaptive Blending, Texture Tiling, Texture Synthesis.

1. INTRODUCTION

Physically correct virtual model visualization can not be accomplished without naturally looking color textures covering virtual or augmented reality scene objects. These textures can be either smooth or rough (also referred to as BTF, see e.g. [MMu03]). The rough textures do not obey the Lambert law and their reflectance is illumination- and view-angle-dependent. Both types of textures occurring in virtual scene models can be rendered either through digitalization of natural samples or by synthesis from appropriate mathematical models. Exact sample digitalization may become prohibitive due to considerable memory requirements, particularly in case of BTFs where each texture is represented by a possibly high number of illumination and view-angle-dependent images. Therefore several texture synthesis methods have been defined to reduce the memory complexity. The related methods may be divided primarily to either intelligent sampling or model-based-

Conference proceedings ISBN 80-903100-7-9
WSCG'2005, January 31-February 4, 2005
Plzeň, Czech Republic.

Figure 1. The picture is made of rectangular tiles. Can you guess, what is the tiling grid size and how many different tiles have been used ? (see Fig.10)

analysis and synthesis. The model-based techniques (see, e.g., [Bes74], [Kas81], [BK98], [Hai91], [PJ00], [GH03], [HH00], [HH02]) describe texture data by means of multidimensional mathematical models and later use an extremely compact representation for seamless synthesis of arbitrarily sized texture images. Intelligent sampling approaches (see, e.g., [DB97], [EL99], [Efr01], [Hee95], [XGS00], [CS03], [KS03]) rely on sophisticated sampling from real texture measurements. Sampling based methods currently achieve better visual quality at a cost of less effective compression. Particularly the simpler intelligent-sampling methods have been receiving constant attention for their applicability in graphic hard-

ware. DeBonet's method [DB97] constructs the texture in coarse-to-fine fashion, preserving conditional distribution of filter outputs over multiple scales, while another multi-scale method [Hee95] uses histograms of filter responses. The "image quilting" method [Efr01] by Efros et al. connects rectangular pieces of the texture sample together while minimizing the boundary cut error. Similarly the algorithm by Xu et al.[XGS00] uses regular tiling combined with a deterministic chaos transformation. Very good results can be achieved by employing Wang tiles [CS03] or the so-called graphcut textures [KS03]. All of these methods implement some sort of source texture sampling and the best of them often produce very realistic synthetic textures.

However, no texture synthesis method can be considered ideal for all potential applications. Either the performance, universality, visual quality of results or applicability in current hardware may become the prohibiting factor.

Our Motivation

Many of the current sampling methods involve image operations that may result in visible seams, typically when combining incompatible pieces of texture. A good way to improve the visual quality in such cases is to find (possibly irregular) boundaries between the image pieces to minimize the visual error. In the following we propose a sub-optimal yet highly effective alternative to traditional slow path-search algorithms. Taking use of the algorithm we show a method of developing the texture by visually unrecognizable image transfers (to be referred to as patching). We also show how to utilize this technique in a simple way to obtain groups of mutually connectable tiles representing the given texture. However, the main part of the paper concentrates on the path search and seamless boundary creation problem as we believe the solution presented here is generally usable in many different contexts and applications.

The paper is structured as follows: Section 2 discusses in detail how a virtually invisible transition between two texture image regions can be created. Section 2.1 shows a novel sub-optimal algorithm for path search that can be used instead of slow exponential algorithms like the dynamical programming. Section 2.2 shows how to improve the visual transition quality in cases when minimum error path does not suffice to prevent discontinuities. Section 2.3 extends the stitching technique to enable seamless transfer of whole image regions (patching). In Section 3 we show a trivial yet well-performing way of seamless tile creation. Assuming one tile has been created, we show in Section 3.2 how new, visually different derivatives can be created based on it while all of the tiles remain mutually connectable. Such tile sets can then be used to synthesize texture images of significantly higher quality than it is possible with simple tiling approaches, as shown in the Experiments Section 4. Section 5 summarizes the advantages and discusses the drawbacks and perspectives of the proposed methods.

2. IMAGE STITCHING

Consider *image stitching* a process of creating natural transitions between two image regions. This task is simpler for naturally self-similar (e.g. homogenous) textured images. The transition is to be made as unnoticeable and indistinguishable from the neighboring image areas as possible. We define the technique based on the minimum error boundary cut idea, as used in the "image quilting" algorithm [Efr01]. Let us assume that each stitch between two equally sized overlapped image regions R_1 and R_2 is oriented. A right-oriented stitch image will consist mostly of pixels from R_1 along its left side and mostly of pixels from R_2 along its right side. Creating such stitch can be imagined as attaching a cropped part of R_1 (source) to R_2 (target) as demonstrated in Figure 2. The following sections show in detail how to crop and how to reduce unwanted visual errors for cases when cropping itself is not sufficient.

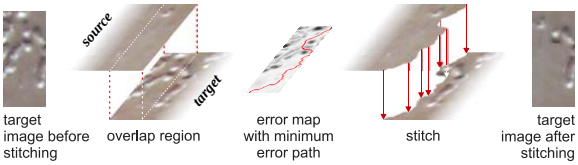


Figure 2. Image stitching (right-oriented case). The source image is cropped from the right along the minimum error path and placed over the target background image.

Minimum Error Path Search

Let us consider a right-oriented stitch creation problem, as demonstrated in Figure 2. Suppose the source region R_1 is to be placed over target region R_2 where the overlap size is $w \times h$ pixels. Width w is considered a user parameter that determines how relaxedly the transition between R_1 and R_2 should be constructed and thus trades the achievable visible quality for algorithm efficiency. To make the transition as invisible as possible, R_1 is cropped from the right side along a minimum error path before attaching to R_2. The minimum error path is constructed to lead vertically from the top row to the bottom row of error map E, which represents the visual difference between R_1 and R_2 for each pixel of the overlap region:

$$E[i,j] = d(R_1[i,j], R_2[i,j]), \quad i = 1,\ldots,w \quad j = 1,\ldots,h$$

where $d(.,.)$ is, e.g., the Euclidean distance of two RGB pixel color values. Note: Error maps in Figures 2, 4, 5 and 6 depict higher error by darker grey lev-

els. We adopt a simplified path representation model, as shown in Figure 3. Only the pixels lying to the left of (and on) the path are to be copied from R_1 to the underlying R_2.

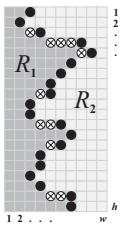


Figure 3. Simplified path representation model for the right-oriented stitch. Each row contains one control point (black dot). Complementary points (crossed dots) must be added to make the path continuous.

Each path is represented unambiguously by a sequence of *control points* **c**, one for each row:

$$\mathbf{c} =< c_1, c_2, \ldots, c_h >, \quad c_j \in \{1, \ldots, w\}$$

However, the complete path as a vertically oriented, continuous sequence of pixels in E must include not only the control points, but also complementary points (marked by crossed dots in Figure 3). From several possible complete path definitions we have adopted the one that suits our oriented-stitch approach, i.e., where each control point becomes visibly the rightmost point in its row:

$$Path^{\mathbf{c}} = \{(i,j) : j = 1, \ldots, h\,; \;\; i = \mu_j, \ldots, c_j\}$$

where

$$\mu_j = min(c_{j-1} + 1, c_{j+1} + 1, c_j).$$

For each path a criterion can be evaluated to asses the expected visible transition inconsistency:

$$\varepsilon(Path^{\mathbf{c}}) = \sum_{(i,j) \in Path^{\mathbf{c}}} E[i,j]$$

Now we can define a sub-optimal minimum path search algorithm on error map E of $w \times h$ pixels. The basic idea is to develop some initial $Path^{\mathbf{c}}$ in stepwise manner by conditional shifting of the control points. New control point position(s) get fixed only if $\varepsilon(Path^{\mathbf{c}})$ would decrease. This ensures the algorithm to converge. The algorithm first evaluates single control point shifts in each step as long as the criterion value can be decreased. Next it attempts to bypass larger error areas by shifting small groups of consecutive control points forming a vertical line at once. Whenever such step improves the criterion value, fine tuning in form of single control point shifts follows. The only user parameter $o_{\max}$ (where $1 \le o_{\max} \le h$) depicts the maximum number of control points processed in one step. Higher $o_{\max}$ values lead to better or equal solutions at a cost of longer computation.

Oscillating search *of minimum error path:*
Initialization: for $j = 1, \ldots, h$ let

$$c_j = \arg \min_{1 \le i \le w} E[i,j];$$

Let $\bar{\varepsilon} = +\infty$; *"initial error value boundary"*
1: Let $o = 1$; *"initial step size"*
2: For $j = 1, \ldots, h$
{
  Store $c_j \ldots c_{j+o-1}$ temporarily to $c_j^{temp} \ldots c_{j+o-1}^{temp}$;
  For $i = 1, \ldots, w$
  {
    For $k = j, \ldots, j + o - 1$ let $c_k = i$; *"vertical line"*
    If $\varepsilon(Path^{\mathbf{c}}) < \bar{\varepsilon}$ update $\bar{\varepsilon}$ and go to **1**;
  }
  Restore $c_j \ldots c_{j+o-1}$ from $c_j^{temp} \ldots c_{j+o-1}^{temp}$;
}
Let $o = o + 1$; If $o \le o_{max}$ go to **2**;

We use this algorithm as a fast alternative to slow optimal path search procedures like the dynamical programming. The main reason is computational speed. The *oscillating search* has polynomial complexity while optimal search is always exponential. The oscillating search is a step-wise procedure that sequentially improves some actual solution and thus can be stopped at any moment to yield a usable result. The visible differences between optimal and suboptimal search results can be considered marginal, as demonstrated in Figure 4.

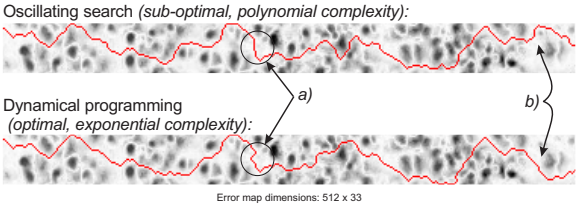


Figure 4. Sub-optimality vs. optimality of path search.

The suboptimal search as defined here prohibits returning path sections (Figure 4a). Differences, if any, occur in areas of evenly distributed error (Figure 4b) and thus remain visually unimportant. In the case depicted in Figure 4 the sub-optimal search was faster than dynamical programming by a factor of 5000 (depending on error map dimensions) while the numerical difference between the sub-optimal and optimal criterion values was about 10%. Moreover, experiments show that numerical optimality of found paths is not crucial for the visual appearance of transitions. It is more important to ensure that the overlap image region itself is positioned and sized not to rule out the existence of low error paths (for example of such a difficult error map see Figure 5).

Remark: In the context of off-line texture analysis to be discussed in the following (Section 4) the time factor is not crucial. However, it is of key importance in many other applications (see, e.g., [Efr01]).

Adaptive Boundary Blending

The minimum path based stitching often produces good natural appearance of image transition areas. However, if no good path exists in the error map, visible artifacts can not be avoided (as demonstrated in Figure 5–simple stitch).

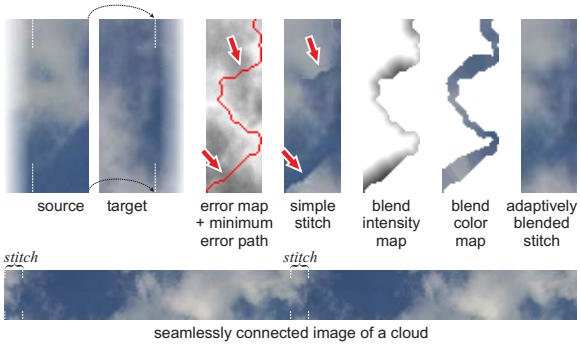


Figure 5. Adaptive blending to improve visual consistency of stitched image areas.

Therefore we have defined the *adaptive boundary blending* as an attempt to reduce the visibility of such unwanted and striking high-error artifacts, should they emerge during the stitching process. The idea is to interpolate between overlapped source region R_1 and target R_2 with a locally adjusted intensity while utilizing the *minimum error path* both as a boundary and as a coloring guideline. Our experiments show that to prevent unnaturally smoothed appearance it is better to keep the affected area minimal, just enough to mask the high error artifacts. In the following we consider a right-oriented stitch again. This is of importance now, because the blending process we adopt is targeted to one side (left in this case) of the path only what helps to better preserve the original image appearance.

Let us denote S the adaptively blended stitch region of $w \times h$ pixels to be created from R_1 overlapping R_2. We assume the minimum error path $Path^{\mathbf{c}}$ and error map E are known (see the previous Section). The *blending range* (maximum distance from the path where pixels get affected) is to be set as parameter ρ. The ρ value should be specified with respect to the properties of the processed source image. Higher image resolution should be reflected on higher ρ. However, with ρ being too high the blending effect can become visually too apparent. On the contrary, too small ρ may not be sufficient to suppress the worst visual stitching errors, should they appear. The stitch is created row-wise, i.e., for each $j = 1, \ldots, h$:

$$S[i,j] = \begin{cases} R_2[i,j] & \text{for } c_j < i \le w \\ R_1[i,j] * \alpha + R_2[m,n] * (1-\alpha) & \text{for } 1 < i \le c_j \end{cases}$$

where

$$(m,n) = \arg \min_{(a,b) \in Path^{\mathbf{c}}} \sqrt{(a-i)^2 + (b-j)^2}$$

$$\alpha = \max\left(\frac{\sqrt{(m-i)^2 + (n-j)^2}}{\rho * E[m,n]}, 1\right)$$

The adaptive blending process can be visualized using the *blend intensity* and *blend color* maps (see Figure 5). The blend intensity map represents the weighing information on how R_1 contributes to the blended result in the area left from the path. In Figure 5 the darker pixels depict lower contribution. R_2 contributes to the same area indirectly using the blend color map that shows how color information is extracted from R_2 path pixels. As seen on examples (Figures 5, 8 and 11), the boundary can be made almost unnoticeable in this way, except of cases when the transition is made between principally incompatible texture image areas.

Remark 1: We should note that simpler interpolation algorithms have been tested as well but led to worse results, usually emphasizing incompatibilities of R_1 and R_2 image contents alongside the minimum error path. Remark 2: A little better visual quality can be achieved if S[i, j] for $1 < i \le c_j$ would depend not only on the single closest on-path pixel (m,n) in $Path^{\mathbf{c}}$, but on several close neighboring pixels in $Path^{\mathbf{c}}$.

Image Patching

The image stitching method described in the previous Sections can be extended to transfer general continuous image regions while keeping the transition between the old and new unnoticeable.

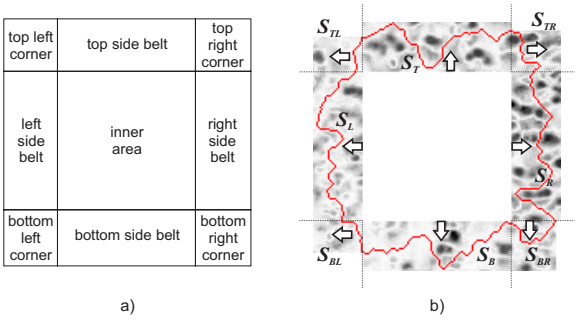


Figure 6. Patch creation. The stitching technique is used to create sides and corners of the patch; the inner area is simply copied. White arrows show stitch orientations (requires optimization).

For the sake of simplicity we define a patch as a part of image surrounded by a continuous minimum error path that does not extend out of a given surrounding rectangle. Our *image patching* algorithm is illustrated in Figure 6. The inner area is simply copied from source position to the target. Side stitches are then created inside the four side belts of a user-specified width with obvious orientation (see the white arrows in Figure 6). Finally the corner stitches are added,

with one additional restriction: the path initial and final *control points* must be fixed to remain connected to those of the side stitches to ensure the patch is surrounded by one continuous path.

3. TEXTURE TILING

Texture tiling is extensively used for various purposes ranging from simple web page design to realistic 3D display of natural surfaces. Creating a single seamless tile out of some source image is thus a traditional problem for which numerous algorithms exist. One of the simplest is probably the "Photoshop clone tool" approach. The idea is to half-shift the image and then to use the manual clone tool to blot out the now apparent horizontal and vertical seams that have emerged inside the image. From the automated methods many take use of extensive blending in not very sophisticated manner, what often results in too striking visual change of the texture appearance along the tile borders. The image stitching technique described in previous sections is well suited for the purpose of seamless tile creation and can be expected to give considerably better results than simple blending methods.

Tile Overlap Optimization and Stitching

First, we search for such rectangular region in the source texture image, where the opposite border areas are the most visually consistent in both the horizontal and vertical direction.

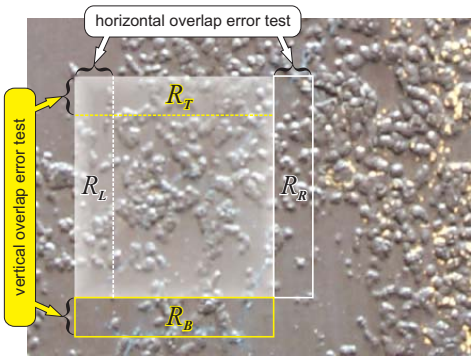


Figure 7. Tile template positioning on a source image.

As depicted in Figure 7, the search procedure minimizes the visual difference among image regions R_L and R_R, and among R_T and R_B, respectively. The overlap width is to be decided by the user, the width and height of the candidate region is optimized by the procedure. The tile (brightened region) is then cut out and made seamless by overlapping and stitching image region R_R over R_L, and R_B over R_T, respectively. The tile sizing and positioning phase is trivial; its purpose is mainly to provide for better stitching results. We use the average RGB Euclidean distance as a criterion of visual consistency. Remark: From performance reasons we split the tile positioning and sizing process to two sub-optimal independent phases, vertical and horizontal.

Deriving Mutually Connectable Tile Sets

For many textures a single tile is not sufficient to synthesize naturally looking images. Simple tiling usually leads to unwanted emphasis of the rectangular grid, despite the seamlessness of tiles (see the left image in Figure 8). Moreover, the character of many textures makes it impossible to create a single tile that would sufficiently represent all the texture variability. Our idea is to make tiling more realistic by employing more than one tile per texture. The positive effect of increasing the number of tiles is demonstrated in Figure 8.

New tiles can be obtained using the *patching* technique described in Section 2.3. New tiles can be created by making a copy of the template tile and subsequently covering its inner area by patches taken from different positions in the source texture image. Different algorithms can be defined to accomplish this task, both deterministic and non-deterministic, with different properties. It is possible to define sophisticated algorithms aiming to build tile sets of specified properties, e.g. representing well the variability of original texture image contents. This algorithm definition problem can be considered outside the scope of this paper. Therefore, we suggest here only the simplest possibility. Each new tile is can be obtained by applying one patch only, sized little less or equal to the tile size and taken from a random source position. Even if the patch and tile sizes are equal, the original tile contents remain usually unaffected alongside its borders (and thus the tile remains connectable), because of the expected irregularity of minimum error paths. This effect can be seen in Figure 6b. Remark: The described tile set derivation procedure is obviously independent on the particular technique used to obtain the initial template tile.

4. EXPERIMENTS

We have tested the presented techniques on a set of textures, mostly from VisTex and UTIA databases. The benefit of using tile sets instead of single tiles is demonstrated in Figure 8. The picture of pink bloom over green grass can be considered a difficult texture. A single tile is clearly insufficient to obtain a naturally looking synthetic image. Adding two tiles improves the result. However, no less than 5 tiles seem sufficient to suppress the striking visibility of the regular tiling pattern.

The examples in Figures 1, 8 and 11 have been obtained as follows: the first tile for each texture has been created automatically (see Section 3) with the only restriction of some reasonable minimum and

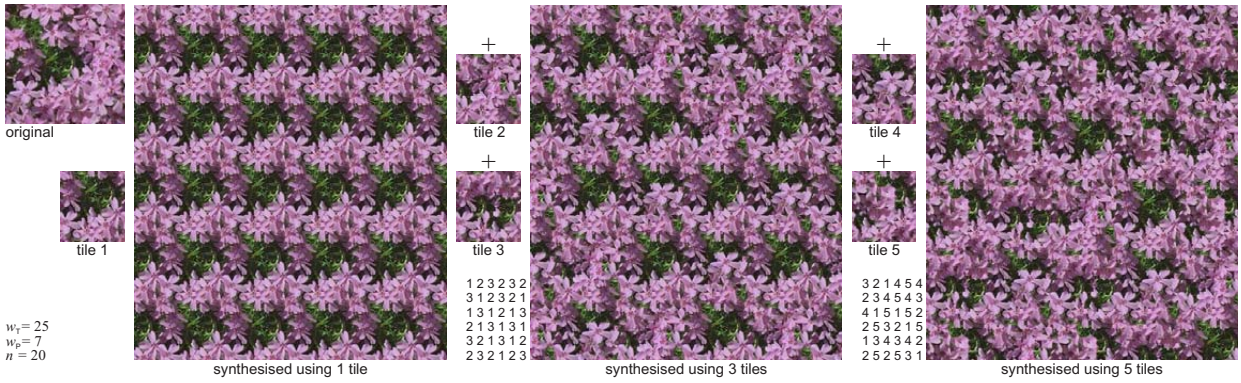


Figure 8. Visual improvement of synthesis results by combining an increasing number of tiles.

maximum tile size. The overlap width was set to ca. 1/10 of the expected tile size. The first tile was then used as a template for tile set generation (see Section 3.2). 30 patched tiles had been generated per texture from which the final tile subsets were selected manually. In cases where the results had been found unsatisfactory, the experiment was repeated with modified parameters. The stitching region width, both in the case of the first tile creation (see Section 3.1) and subsequent tile patching (see Sections 2.3 and 3.2) has proved to be important and is therefore marked in Figures 8 and 11 as w_T and w_P, respectively. Patch source positions were obtained randomly, but optimized subsequently on a small neighborhood of size n to avoid worst possible visual problems. For the adaptive blending the parameter ρ had been set to 3 in all cases, which has shown to be satisfactory in most cases. Note: The synthesized images in Figs. 8 and 11 are cropped to fit in the page.

Time complexity of all computations is low. Each tile can be generated typically in seconds on a 2GHz PC. Once a good set of tiles has been found, texture synthesis is reduced to assembling different tiles to the grid according to some index matrix. This can be done directly in the GPU at no additional time cost.

Expectably good results have been obtained with most of the regular textures (Fig. 11c). Very good results have been obtained even for some more difficult textures, where the irregular texture nature (chocolate, Fig. 11e) could have caused problems. Some of the most difficult textures displaying natural objects like tomatoes can be synthesized surprisingly well (Figs. 11b, 11d). However, with some textures it showed impossible to prevent unnatural artifacts (see in detail Fig. 11f, which looks fine at first sight). We have also experienced problems with textures where good overlapping regions could not be identified to create the initial tile, or with textures containing very distinct particles or structure that has a negative impact on the tile grid visibility (Figure 9). Nevertheless, the technique proved to be well capable of synthesizing broad range of natural textures. Some of the possible visual problems follow from principal reasons and cannot be avoided; others can be removed or suppressed by repeating the experiments with modified parameters, in particular with different stitch widths and patching sources. The quality of output also depends on the size of the original texture sample. Too small a source image would result in too homogenous and regular results. Small source images usually imply the necessity to create more tiles to compensate the effect of denser and thus more apparent tile grid. To capture low frequency information, larger tiles are necessary. To prevent the visibility of the tile grid the tiles in a set must have sufficiently varied content, i.e., the patching process must affect most of the tile image surface. No tiling can overcome certain principal problems like, e.g., the problem of source texture images containing slightly rotated linear structures (Figure 9) from which no smoothly connectable tile can be derived. In general, if the texture exhibits some apparent linear structure then it should be either vertically or horizontally aligned. Skewed and rotated linear structures require sufficiently large samples or are not manageable at all.

Figure 9. Example of a problem structure in source texture images.

Figure 10. The front page image is composed of 8 different tiles in a 4×4 grid according to the index matrix: ((7,6,1,5), (3,7,5,8), (5,1,6,2), (4,3,2,3)).

Finally, the answer to the front page Figure 1 question is in Figure 10.

5. CONCLUSION

We have presented a novel fast path search algorithm and adaptive blending technique that are suitable for seamless image transfer, in particular in the context of texture synthesis. Using these tools we have demonstrated a relatively simple technique that enables synthesis of naturally looking textures by means of advanced image tiling. We show how a set of mutually connectable yet differently looking rectangular tiles can be obtained for a broad range of source texture measurements. We show that even very irregular textures can by represented well using such tile sets. The main advantage of the presented technique is the clear separation of the off-line texture analysis, while the synthesis is reduced to trivial combination of precomputed tiles. The visual quality of output is close or comparable to the best of current techniques as shown in Figures 8 and 11.

The tiling technique is scalable. The trade-off between the visual quality and computational complexity can be controlled by changing the number of tiles in the tile set. For each texture some minimum number of tiles is usually necessary to ensure sufficient quality of results. Regular (possibly rectangular) textures without much detail can be represented by fewer tiles than highly irregular stochastic textures. Most of the algorithms presented here are extendable or modifiable. We have found the technique to be extendable for BTF modeling (bidirectional texture fields, see, e.g. [MMu03], [MMe03]) to enable particularly accurate display of natural surfaces with respect to view- and illumination- angles.

6. ACKNOWLEDGMENTS

This work has been supported by the EC project IST-2001-34744 RealReflect, FP6-507752 MUSCLE, grant No. A2075302 and 1ET400750407 of the Grant Agency of the Czech Academy of Sciences. We thank Alexei Efros for permission to reproduce illustrations from [Efr01] (Figure 12).

7. REFERENCES

[Bes74] Besag J.: Spatial interaction and the statistical analysis of lattice systems. Journ. of the Royal Statistical Society, B-36, 2 (1974), 192–236.

[BK98] Bennett J., Khotanzad A.: Multispectral random field models for synthesis and analysis of color images. IEEE Trans. on PAMI 20, 3 (Mar. 1998), 327–332.

[CS03] Cohen M.F., Shade J., Hiller S. and Deussen O.: Wang Tiles for image and texture generation. In ACM Transactions on Graphics 22, 3, SIGGRAPH 2003, 287-294.

[DB97] De Bonet J.: Multiresolution sampling procedure for analysis and synthesis of textured images. In Proc. SIGGRAPH 97 (1997), ACM Press, 361–368.

[Efr01] Efros A.A., Freeman W.: Image quilting for texture synthesis and transfer. In SIGGRAPH 01 (2001), Fiume E., (Ed.), ACM Press, 341–346.

[EL99] Efros A. A., Leung T. K.: Texture synthesis by non-parametric sampling. In Proc. Int. Conf. on Computer Vision (2) (1999), 1033–1038.

[GH03] Grim J., Haindl M.: Texture modelling by discrete distribution mixtures. Computational Statistics & Data Analysis 41, 3-4 (2003), 603– 615.

[Hai01] Haindl M.: Texture synthesis. CWI Quarterly 4, 4 (Dec. 1991), 305–331.

[HH00] Haindl M., Havlíček V.: A multiresolution causal color texture model. In Advances in Pattern Recognition, LNCS 1876. Springer-Verlag, Berlin, (Aug. 2000), ch. 1, 114–122.

[HH02] Haindl M., Havlíček V.: A multiscale color texture model. In Proc. 16th Int. Conf. on Pattern Recognition (2002), Kasturi R., Laurendeau D., (Eds.), IEEE Computer Society, 255–258.

[Hee95] Heeger D.J. Bergen J.: Pyramid based texture analysis/synthesis. In Proc. SIGGRAPH 95 (1995), ACM Press, 229–238.

[Kas81] Kashyap R.: Analysis and synthesis of image patterns by spatial interaction models. In Progress in Pattern Recognition 1 (North- Holland, 1981), Kanal L., A.Rosenfeld, (Eds.), Elsevier.

[KS03] Kwatra V., Schödl A., Essa I., Turk G., Bobick A.:Graphcut Textures:Image and Video Synthesis Using Graph Cuts. In ACM Transactions on Graphics 22, 3, SIGGRAPH 2003, 277-286.

[LLX*01] Liang L., Liu C., Xu Y.-Q., Guo B., Shum H.-Y.: Real-time texture synthesis by patchbased sampling. ACM Transactions on Graphics (TOG) 20, 3 (2001), 127–150.

[MMu03] Meseth J., Müller G., Sattler M., Klein R.: BTF Rendering for Virtual Environments. Virtual Concepts 2003, (2003), 356–363.

[MMe03] Müller G., Meseth J., Klein R.: Compression and real-time Rendering of measured BTFs using local PCA. Vision, Modeling, and Visualization, (2003), 271–280.

[PJ00] Portilla J. S. E.: A parametric texture model based on joint statistics of complex wavelet coefficients. Int. Journal of Computer Vision 40, 1 (2000), 49–71.

[SP00] Somol P., Pudil P.: Oscillating search algorithms for feature selection. In Proc. 15th Int. Conf. on Pattern Recognition (2000), vol. 2, IEEE Computer Society, 406–409.

[Wei01] Wei L. Levoy M.: Texture synthesis over arbitrary manifold surfaces. In Proc. SIGGRAPH 01 (2001), ACM Press / Addison Wesley.

[XGS00] Xu Y., Guo B., Shum H.: Chaos Mosaic: Fast and Memory Efficient Texture Synthesis. Tech. Rep. MSR-TR-2000-32, Redmont, 2000.

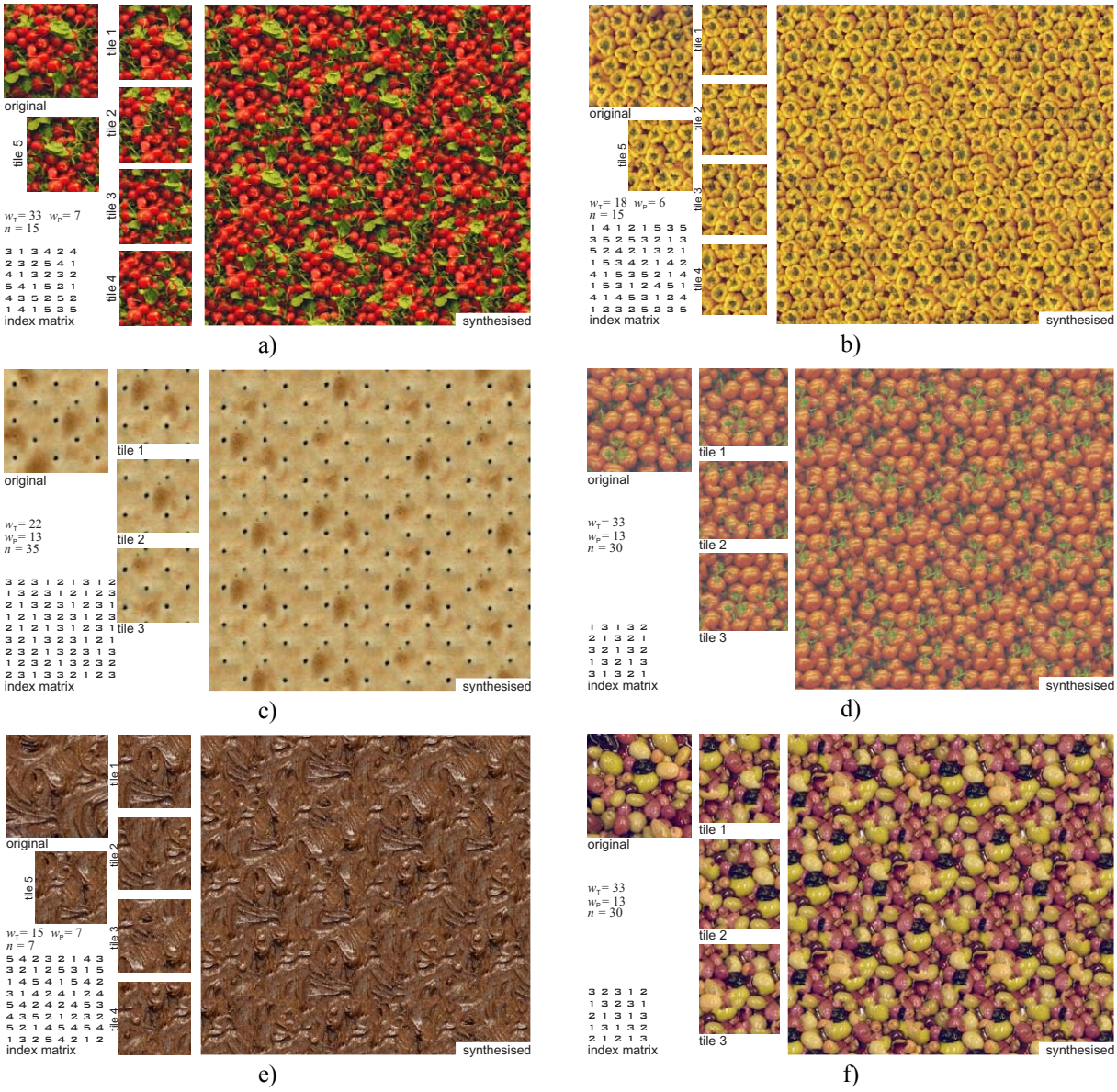
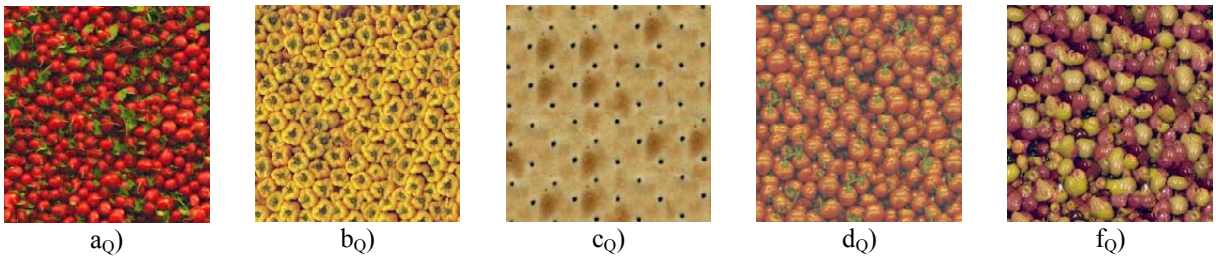


Figure 11. Examples of tilings obtained with the proposed method.



a_Q) b_Q) c_Q) d_Q) f_Q)

Figure 12. Image Quilting [Efr01] results for comparison.