INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Content-Aware Texture Synthesis

Pierre-Edouard Landes
Cyril Soler

Grenoble Universities, INRIA, LJK

N° 6959

Juin 2009

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6959--FR+ENG

Content-Aware Texture Synthesis

Pierre-Edouard Landes*
Cyril Soler†
Grenoble Universities, INRIA, LJK

Thème COG — Systèmes cognitifs
Projet ARTIS

Rapport de recherche n° 6959 — Juin 2009 — 20 pages

Abstract: Existing example-based texture synthesis techniques are inherently unadapted to textures consisting of a set of randomly disposed, individually discernible shapes. Local methods striving at pixel-based discontinuity reduction hardly preserve input's long-range structures. Alternatively, research built upon the supposed respect by the input's features of given placement rules are too restrictive to be straightly extended to stochastic arrangements. In this paper we present a new method for analyzing and resynthesizing such arrangements. Our objective is to acquire their constitutive shapes to enable structure-aware resynthesis. What characterizes such shapes is their repetition throughout the input. We exploit this trait by recording recurrences of visually similar neighborhoods which are later extended to regions. We bring those together to compute the input's coverage map and extract final repetitive shapes. By directly manipulating shapes, resynthesis can be enriched with high-level information unavailable in pixel-based approaches. We gather statistics on their placement and appearance variations and use those to produce new images. To achieve this, we draw inspiration and improve techniques for capturing element arrangements, techniques once limited to vectorized NPR primitives.

Key-words: texture analysis, texture synthesis, local descriptors

* e-mail: pierre.landes@inrialpes.fr
† e-mail: cyril.soler@inrialpes.fr

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Analyse et synthese de distributions d'objets

Résumé : Les techniques de synthèse de textures par l'exemple ne se prêtent guère à la génération de textures définies comme arrangements quelconques de formes individuellement distinguables. Par exemple, les méthodes non-paramétriques s'efforçant à localement prévenir les discontinuités entre pixels voisins ne parviennent pas à préserver de telles formes si leur taille en pixels est trop importante. Également, les techniques supposant le respect de règles de placement prédéfinies par les structures de la texture d'entrée s'avèrent trop restrictives pour être directement applicables aux arrangements stochastiques de formes.

Cet article expose une nouvelle méthode d'analyse et de re-synthèse de telles textures. Nous visons ici à l'extraction explicite des formes constitutives de l'échantillon d'entrée et à leur utilisation pour assurer la génération de nouvelles textures préservant au mieux les structures de celui-ci.

Ce qui caractérise ces formes est leur répétition non-triviale au sein de l'image d'entrée. Nous exploitons cette observation et consignons l'ensemble des co-occurrences de voisinages visuellement proches que nous agglomérons ensuite en régions continues de l'image. Nous partitionnons ensuite ces régions en classes de motifs et calculons la segmentation de l'exemple selon ces classes pour permettre l'extraction finale des formes répétitives. Via la manipulation directe des motifs ainsi calculés, la re-synthèse est alors enrichie d'informations de haut niveau, impossible à extraire par une analyse pixellique immédiate. Nous établissons alors des statistiques quant au placement relatif des motifs ainsi que leurs légères variations d'apparence afin de produire de nouvelles images. Nous pouvons ainsi étendre les techniques récentes de synthèse d'arrangements d'éléments vectoriels à des entrées rasterisées.

Mots-clés : Analyse d'image, synthese de texture, descripteurs locaux

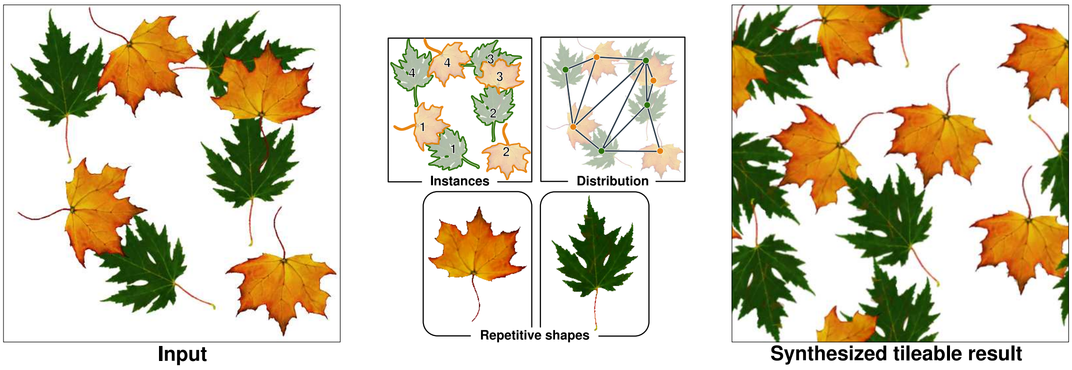


Figure 1: *We propose a method for example-based synthesis of images consisting of shape arrangements such as the one on the left. We first analyze the input to get a higher-level representation than its raster counterpart. Once repetitive basic shapes have been detected and their relative placement captured, we can synthesize visually similar shape arrangements. Since we infer relevant shapes thanks to their multiple duplicates throughout the image, we do not require them to be entirely visible at once. This is the case of the green leaf for instance.*

1 Introduction

In example-based texture synthesis, the main challenge is, given an input sample, to capture its visual characteristics while adding enough randomness to maintain its natural feel. Most state of the art methods directly use their input as what it is, a 2d array of pixel colors. As a result, recognizable shapes get inconsistently mixed together and with the background. In addition, shapes covering one another are neither understood as such, nor treated adequately. Our claim is that, in the case of *random arrangements of shapes*, pixel-based representations are unsuited and thus the main cause of synthesis failure. Being able to properly capture the arrangement's stochasticity is crucial, and all the more challenging than it resides not in color variations between direct neighboring pixels, but in the relative placements, orientations and overlaps between the shapes.

In this paper we propose an automatic method for detecting and extracting the input's constitutive shapes without any *a priori* knowledge. We factorize the different exemplars of a same shape as a unique *pattern*, along with the set of mappings describing their placement. We also compute a coverage map to handle shape overlap. We synthesize new images with an improved technique for reproducing arrangements of elements. This enables us to produce new textures while consistently preserving the nature of the input image.

Our algorithm finds shapes thanks to their repetition throughout the image. To achieve this, we detect occurrences of visually close neighborhoods which we further use to determine similarity relationships between whole regions. The obtained regions are then sorted and consistently assembled together in order to get a concise description of the image content. This description is composed of a small number of different patterns and, for each, the set of transformations mapping them to their copies in the image. Our pattern recovery does not need all instances of a shape to be entirely represented in the input. Instead, the only requirement is that all parts of a given shape appear at least twice in the image. Image regions where multiple shapes overlap and partially mask each other are therefore correctly detected. Finally, in most images, shapes are related to their duplicates by non-linear transformations. To ensure robustness, we approximate these mappings during shape extraction, provided that they remain reasonably close to rigid transformations. We then demonstrate the efficiency of our texture analysis method for re-synthesizing textures. It enables us to perform meaningful statistics over high-level varying parameters of the recovered shapes, such as their color and spatial distribution. We can then produce new textures which are consistent with the input by introducing randomness while preserving recognizable shapes (*cf.* figure 1).

Our method fills in the gap between local pixel/patch-based texture representations, unable to grasp high-level structural texture elements, and lattice-based methods, limited to near-regular textures and requiring user assistance to handle deformations. It also tends to complete recent research in expressive rendering that aims at capturing arrangements of non-photorealistic rendering primitives. These methods directly work with vectorized, tagged inputs. Our goal is to propose similar techniques for raster images.

Fully automatic and unsupervised, our texture analysis/synthesis method does not rely on semantic information. While producing sensible results, it may make unintuitive –yet valid– choices when collecting shapes. This might not always correspond to what a human being would naturally have done. It is mainly due to the fact that one recognizes shapes rather through comparison with past experience knowledge than actual explicit redundancy. Lastly, the requirement that transformations stay close to 2d similarities prevents us from extracting same objects related by more sophisticated deformations.

In summary, our contributions are:

- the analysis of the input which finds similarity between regions, sorts and assembles them in order to express its content as a collection of possibly overlapping shapes;
- the synthesis of new arrangements which exploits the randomness of high-level parameters that characterize the input's distribution and appearance.

We now review existing texture synthesis techniques and explain why they are hardly adapted to reproduce stochastic arrangements of shapes. Section 3 gives a technical overview of our method which is further described in Sections 4 and 5.

2 Related Work

2.1 Texture Synthesis

Texture synthesis has received much attention in Computer Graphics research, spanning over decades now. Despite the huge number of techniques available, no method is designed for analyzing and synthesizing stochastic arrangements of shapes.

Parametric Texture Modeling: Very first attempts at texture synthesis aim at defining good texture models, e.g., reaction-diffusion, fractal, frequency domain or Markov Random Field (MRF) models. Parametric texture representations have also been proposed and draw inspiration from the primary visual cortex processing [9, 20]. In their pioneering work on example-based synthesis, Heeger and Bergen use first-order statistics as a texture representation and match the histograms of the steerable pyramids of both input and output. Portilla and Simoncelli propose a more elaborate texture model, based on the joint statistics of wavelet coefficients. Although these methods yield satisfactory results on purely stochastic textures and are particularly useful for texture classification and perception, they cannot handle shape distributions correctly.

Non-Parametric Sampling Techniques: Other example-based techniques synthesize textures by directly picking colors from the input sample. Such methods assume textures are realizations of a MRF, and are guided by local neighborhood matching [4, 8, 24]. Simple and efficient, they yield surprisingly good results on a wide range of textures. However, because of the fixed size and shape of the matched neighborhoods, they cannot adapt to large structures and often inconsistently mix together regions seemingly similar at this chosen scale. This hurdle has been specifically tackled, either by reducing the number of candidate neighborhoods [2], or by copying whole patches from the input instead of isolated pixel colors [7, 13, 25]. The difficulty is then to avoid visible artifacts, such as blur or seams, at overlapping regions between patches. However, patches' shape and size are chosen more for seam reduction than actual shape detection. This may produce artifacts in textures containing discernible shapes.

Synthesis by Optimization: Another pitfall of local methods is the lack of global control over the synthesized output. Errors at each addition of new pixels/patches may accumulate and compromise the perceptual similarity of the result with the input. Specific global techniques prevent this by using a non-causal optimization framework [19, 12]. Kwatra's algorithm employs a EM approach to minimize a function quantifying the visual similarity between the input and the texture being created. This global measurement is obtained by summing distances between pairs of visually close neighborhoods from input and output. This approach is particularly successful: it not only yields compelling results on input samples ranging from stochastic to structured textures, but also enables constrained and flow-guided synthesis. Its strength lies in its combination of the controllability of global methods and the visual quality of local ones. Nevertheless, since the cost function is computed between neighborhoods, this approach still evolves at the scale of pixels and may thus not preserve long-range, discernible input's structures.

Near-Regular Texture Synthesis: This original research by Liu and colleagues relies on the spatial arrangement of features throughout the sample [15]. Such geometry-based approaches assume the input's characteristic structures follow specific placement rules (one of the seventeen wallpaper groups). They aim at identifying the underlying lattice structure and then extracting minimal tiles from the input. When deformed, automatically computed lattices can be easily corrected by the user [16]. While those methods are perfectly suited to near-regular textures and achieve impressive results, they cannot be directly used for random arrangements of shapes disregarding those placement rules.

To the best of our knowledge, no texture synthesis method attempts to explicitly take advantage of the repetition of characteristic structures throughout the input, and use it to extract basic shapes for further synthesis. Lacking those shapes, previous work cannot address the issue of their possible partial overlap appropriately. This shortcoming may stay unnoticed if one instance appears unoccluded once in the input. But most of the time, overlap regions are directly pasted onto the output. This is especially true for non-parametric sampling techniques and re-synthesis of random arrangements of shapes often ends in hazardous results. Intuitively, those methods suffer from the unsuited pixel-based representation of the input and attempting to fill this gap is the primary goal of our paper.

Inspiration from Expressive Rendering: NPR methods extend non-parametric sampling texture synthesis to 2d arrangements of elements [3, 10]. Their work capture the relative positions between those elements and reproduce them in order to synthesize visually similar arrangements. However, in order to lift the constraints due to pixel-based analysis, they directly deal with groups of already vectorized, tagged primitives. Our goal is to enable such high-level synthesis techniques for raster input samples, and to propose a suitable representation for that aim. We thus present a method for analyzing raster textures composed of distinguishable shapes, repeating themselves through 2d similarities, and possibly partially overlapping.

2.2 Image Analysis in Computer Vision

Although not directly exploited by classical texture synthesis techniques, local repetition throughout images has successfully been used in Computer Vision, for it can yield useful high-level information about the depicted scenes.
Jojic and co-workers base their appearance and shape model for reduced image encoding on the repetition of similar rectangular patches [11]. The input image is summarized by its *epitome*, condensed version retaining all textural and shape information, and a smooth mapping for its reconstruction. Both are acquired by Bayesian learning. First designed as generative models for images, epitomes have recently been used for texture compaction [23], image compression [22], and "content-aware" cropping [21]. Wang and colleagues slightly alter the epitome definition and compute it after explicit patch matching and greedy packing. Nevertheless, in both cases, epitomes' primary goal is image reduction with minimal loss and, since they do not embed structure information, epitomes are hardly suited to synthesis of new content.
A recent work by Ahuja and Todorovic aims at the automatic extraction of texels from textures composed of thin objects in frontal view [1]. The image is first hierarchically segmented and represented by a tree, where similarity between sub-trees guides texel detection. Then, given a parametric texel model specification, they infer both its structure and parameters by machine learning. Their method is particularly efficient for blob-like texels and can deal with occlusions to some extent. However, it is limited to a single type of texel and does not attempt to capture texel placement.

3 Overview

Given an input image, we express its content in terms of replicated copies of shapes by relying on the multiple occurrences of similar regions only (*cf.* figure 2). To get this information, we proceed as follows.

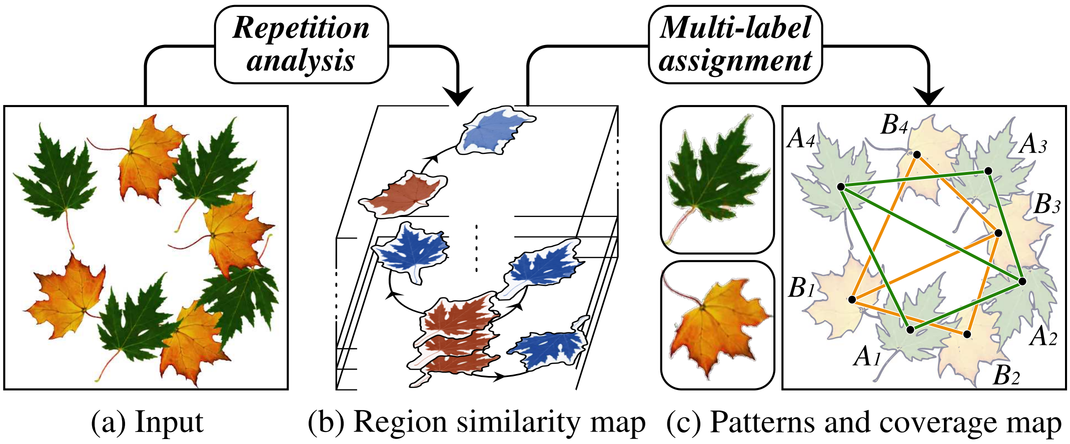


Figure 2: *Overview of our method. Given an input raster image* (a)*, we build its region similarity map* (b) *containing all pairs of similar regions across the image, along with the non-linear transformations that tie them together.* (c) *Lastly, we compute the repetitive shapes' instances along with the input's visibility map.*

First, statistics are performed over the input image in order to detect all pairs of maximal regions which are similar up to a non-rigid transformation. Although regions of a given pair are disjoint, regions of different pairs can overlap or even be completely included in one another. We call this information the *region similarity map* (*cf.* figure 2*b*). It contains all the similarity information in the image at the level of entire regions. This step involves the study of the matches between similar pixels, in a way to retrieve connected regions from the input and is detailed in Section 4.

In a second step, the region similarity map is exploited in order to turn independent pairs of similar regions into meaningful classes of repetitive objects in the image (*cf.* figure 2*c*). For each class, an *original object* is constructed by collecting information from its various instances. For that, all input's pixels must be tagged with the unique object/instance pair it belongs to. We will show in Section 5 that this step is an optimization problem we find an approximate solution of using an appropriate heuristic.

Finally, having obtained a discrete representation of the image content, we perform statistics over the placement, orientation and attributes of the recovered patterns. We then use those measurements and adapt existing techniques for copying element arrangements to produce new images, as explained in Section 6.

4 Constructing the Region Similarity Map

The main insight is to use image regions occurring multiple times throughout the input as handles to recover the structure of the image.

4.1 Visual Similarity Detection

Most successful Computer Vision techniques for image classification and object recognition involve local descriptors. Many variants exist, eventually encoding different visual aspects of images (e.g., luminance, gradients, colors), and they have been widely exploited since they offer improved robustness to noise and partial occlusion. The quality of a local descriptor is evaluated according to its repetitiveness and discriminability. From the literature in that field, studies conclude that Lowe's SIFT descriptor [17] yields the best results [18]. This descriptor is computed in gradient domain, accounts for both textural and shape local information, and is invariant by 2d similarities.

Optimal Scale Detection: To achieve invariance by uniform scaling, we evaluate SIFT descriptors at a specific scale, on an adequately Gaussian-blurred version of the input. We compute all pixels' scale of interest using Lindeberg's scale selection method in accordance to linear scale space theory [14]: the pixels' retained scale coincides with the scale at which the normalized Laplacian response reaches a local maximum along the scale dimension.

Canonical Orientation: Once again, in order to constitute a rotation-invariant description, SIFT descriptors are computed with respect to a local direction. We determine every pixel's associated canonical directions by computing a 36-binned gradient orientation histogram. Neighboring gradients within a 19×19 window are evaluated at the level of the scale space adapted to the pixel's scale. These gradients contribute to the bin associated to their direction proportionally to their magnitude. The pixels' canonical directions then correspond to the bins with locally maximal amplitude.

Local Neighborhood Description: Though many variants appeared since its definition, we implemented Lowe's original, 128-dimensional SIFT on the input's lightness channel (*cf.* figure 3). Taking advantage of more elaborate, chromatic descriptors does not change the proposed method and is currently left to future work.

Descriptor Matching: Our motivation for using SIFT descriptors is their ability to encode visual appearance in a concise, yet meaningful way. Thanks to them, evaluating visual similarity between two neighborhoods up to any rigid transformation becomes straightforward. The smaller the Euclidean distance between their respective descriptors is, the more visually close the pixels' neighborhoods are. This observation allows us to easily find local repetitions by finding for each pixel its nearest neighbors in SIFT feature space. We embed all the computed SIFT descriptors within a Kd-tree and perform for every pixel a fixed-radius search. In our examples, the search window radius ranges between 0.1 and 0.15 in normalized SIFT space.
The highly-dimensional feature space is beforehand reduced by Principal Component Analysis (PCA). It is performed on the whole distribution of computed descriptors and greatly alleviates the pairing computation costs. Neighboring pixels tend to be associated similar descriptors and blind matching often involves small clusters of redundant pixels. We limit this phenomenon by locally pruning matched pixels in 16×16 windows in image space and keeping only the match with minimal pairing error per window.

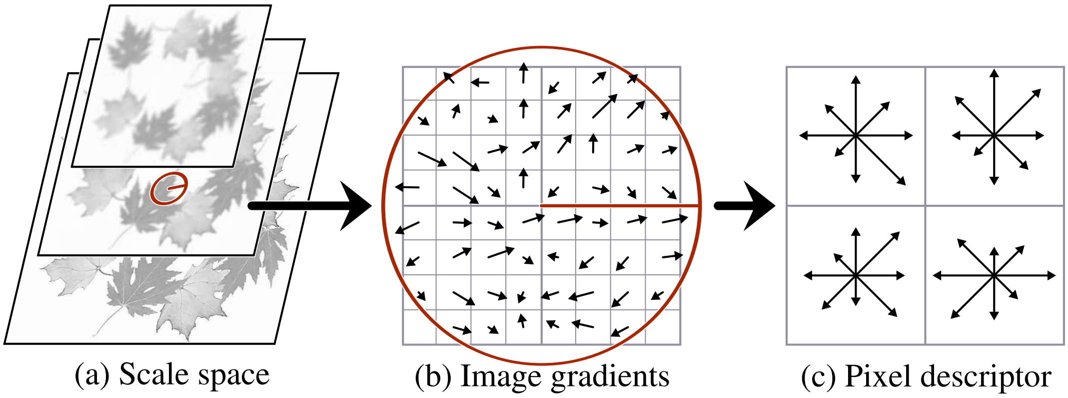
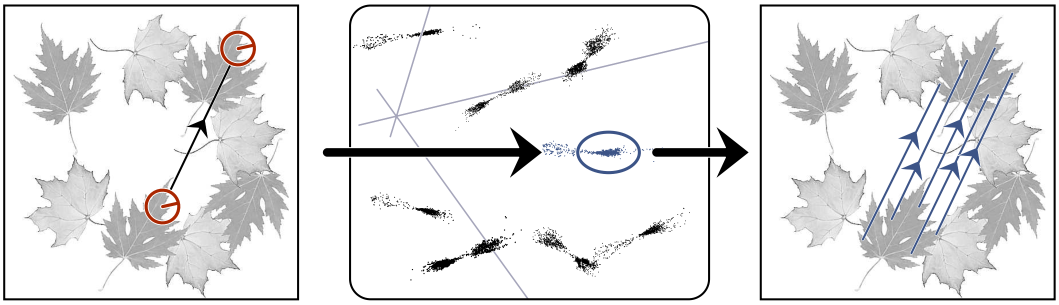


Figure 3: *Lowe's SIFT descriptor.* (a) *Once we get a scale and direction for a pixel,* (b) *we express its neighboring gradients in its rotated local frame.* (c) *Finally, the SIFT descriptor is the concatenation of several orientation histograms disposed around the pixel. In our implementation, we use 16 8-binned histograms of gradients sampled on a* 19×19 *window at the selected scale.*

4.2 Repetitive Shape Extraction

Shape Repetition as Match Aggregations: Each neighborhood match defines a unique 2d similarity thanks to the neighborhoods' respective position, and associated scale and rotation. All the established matches then act in transformation space as samples of some unknown density. Its local maxima correspond to accumulations of transformations. But more importantly, they also attest to the existence of sets of pixels repeating themselves under a roughly common transformation (*cf.* figure 4). We aim at finding those aggregations and use them as hints of presence of repetitive shapes.
We detect such gatherings by *mean shift clustering*, which enables data partitioning with no prior on the final number of classes [5]. Clustering is performed on the similarities' 4 degrees of freedom (translation, rotation, and effective scaling). We also enforce spatial locality by taking into account starting positions of the matches during clustering. While all clusters hint the presence of repetitive image regions, their cardinality quantifies their respective relevancy.
Before clustering, we keep the matches corresponding to one half of the most selective pixels only, assuming they yield the most significant information. We also only consider significant clusters, the ones whose cardinality is at least one fourth of the largest one.
Both operations discard "background" pixels, which found either no suitable match (in the case of stochastic backgrounds), either too many of them (near-uniform backgrounds), or did not give rise to consistent transformation clusters.

Filling up the Blanks: At this point, we have local, but only sparse pairs of pixels following a common transformation. We still need to extract a connected shape, which we obtain by region growing from the starting points of the clustered matches (*cf.* figure 5*a*).
New pixels at the boundary of the ongoing shape are merged to it if the distance between the SIFT descriptors, computed before and after transformation, remains less than a given threshold. By default,



(a) Descriptor matching (b) Transformation space (c) Local cluster of matches

Figure 4: *Each pair of matched descriptors fully determines a 2d similarity* (a)*, which represents a single sample point in transformation space* (b)*. Finding accumulations of such points is the first step towards repetitive shape recovery* (c)*. For illustration purposes, the 4d transformation space is here projected to 3d space.*

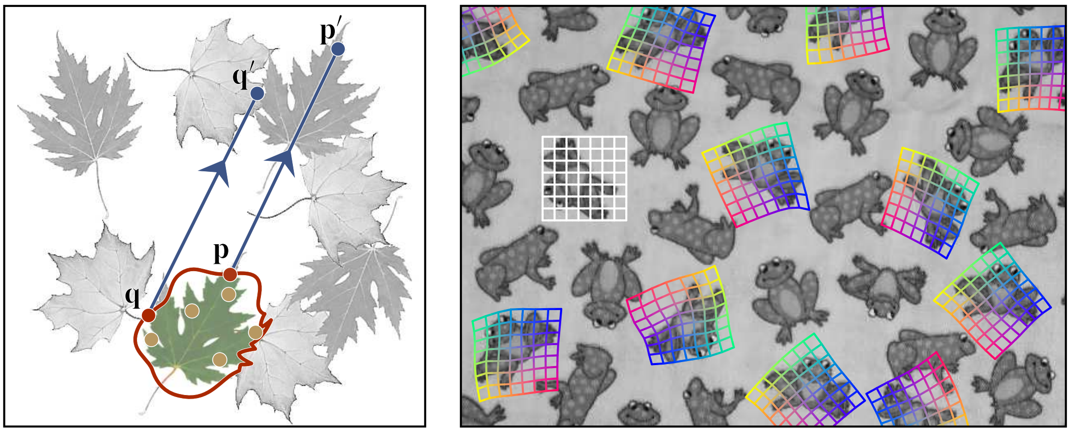
it equals 1.5 times the radius used for matching. Special care must be taken at shape boundaries. Indeed, since SIFT descriptors are evaluated over neighborhoods, they are likely to encode part of the background at shapes' actual boundaries. To cope with this, more local versions of SIFT are considered before halting region expansion. But once we resort to those, expansion must be limited. During this process, self-overlap is forbidden: extracted shapes must not fold onto their transformed counterpart. The order of pixel traversal then becomes important since as the shape grows, some pixel locations get unavailable. We favor regular shapes by prioritizing boundary pixels such as to minimize a constantly updated quality measure. This measure is defined as the ratio of the shape's squared perimeter on its area.

Transformations must also offer more flexibility than plain similarities (*cf.* figure 5*b*). The shapes' mappings are modeled with approximate thin-plate splines, used to describe the behavior across the image of their translation, rotation, and scaling. Those splines are constrained by the matches brought together by the clustering.

After region growing, we obtain the input's *region similarity map*, containing a list of – possibly overlapping – regions along with the non-linear mapping toward their duplicates. These regions are still independent and the region similarity map can be thought of as a multi-layer representation of the input. However, not only this representation is over-complete but a lot of information redundancy remains. Intersecting regions give rise to ambiguities which need to be taken care of in order to extract the final image shapes.

5 Recovering Pattern Classes

The region similarity map enables us to know all repetitive shapes contained in the input image, as well as the transformations they undergo. All those shapes are still independent though. We must



(a) SIFT-driven region growing (b) Spline fitting

Figure 5: (a) *For shape extraction, we perform region growing from the starting points of the clustered matches (yellow dots). Boundary pixels* **p**, **q** *are included if their SIFT remains unchanged after projection* **p'**, **q'** *respectively. We use a coarser SIFT descriptor when it may encode part of the background (here at* **q'***).* (b) *We also fit thin-plate splines in order to better capture subtle deformations. The reference grid is here displayed in white.*

hence gather the ones corresponding to the instances of one same pattern and mold its representative out of them.

5.1 Theoretical Analysis

We need to explicitly group the shapes related to the same patterns and compute the input image's *pattern visibility map*. This mapping indicates to which *unique* shape every input pixel belongs. This information is mandatory to finally extract the patterns and eventually deal with occlusions between their instances.

This problem is highly similar to image segmentation, each segment being one of the identified shapes. Piecewise continuous pixel labeling can be achieved via the optimization of a cost function evaluated over a graph. The set of its vertices is composed of the image's pixels –neighboring pixels yielding linked vertices– and as many additional terminal vertices as there are possible labels. First, each pixel node shares a common edge with every label vertex. Then, those edges, except for a single one, must be severed for the pixel to be labeled. All the edges in the graph are weighted, so that any path can be associated an energy value. This energy can then account for both the contextual information guiding the segmentation and the desired smoothness between neighboring labeled pixels. The segmentation then comes to find a multi-way minimal cut through a flow network.

However, for multi-labeled segmentation, finding this cut is NP-hard and only approximate iterative solutions exist [6]. Evaluating the cost function is also problematic in our case as it depends on the ongoing segmentation. Indeed, it must favor a label assignment such that a minimal number of maximally-instanced patterns appear on the visibility map. This dependence may cause convergence problems and makes the energy minimization framework unsuited to our problem.

5.2 Proposed Heuristic

We decide to use a two-fold approach to finish the analysis of the input. First, we build the *regions' overlap graph* and analyze it to group them together to form classes. Then, given those classes, we compute the pattern visibility map by assigning a unique class and member to each input pixel in an iterative way. Those two separate steps are presented in more details below.

Building and Traversing the Overlap Graph

From the similarity map, we know the locations in the image of the regions before and after their respective transformation. This information and overlaps between these regions allow their partitioning into classes.

We encode those overlaps in a graph: its nodes correspond to the regions and edges are created between them whenever their associated regions, either transformed or not, overlap. Those edges comprise two oriented half-edges –each of them storing the transformation mapping their starting shape to their ending shape– and the normalized strength of the ensued overlap. The overlap strength is equal to the area in pixels of the regions' intersection. Before its assignment to a half-edge, it is normalized by the area of the half-edge's ending shape. Each half-edge can now be interpreted as "to what extend its starting shape, after its associated transformation, contains its ending shape".

This encoding enables us to easily bring together regions sharing *significant* mutual overlap. For an edge to be deemed significant, its half-edges' lowest and highest strengths must exceed specified thresholds. We respectively use 0.25 and 0.75.

At this point, we need to organize the shape nodes and study their transformations, for that step is crucial to get consistent pattern classes. We achieve this grouping via connected component analysis along significant edges, while ensuring the respect of additional constraints. Indeed, each edge describing a transformation, we must guarantee the absence of inconsistent cycles within the ongoing class when adding new nodes to it. Nodes of a same class must not be linked by several paths producing different transformations. Some edges may be associated the identity transform and nodes adjacent to such edges may naturally overlap. Apart from that case though, we must also prevent overlapping regions separated by other transformation, from ending up in the same class, as it would lead to classes with overlapping members.

Once the graph analysis has been achieved, region nodes sharing the same transformation within the class are merged together in order to form its final members. Now, each class deals with a specific pattern and consists of a set of shapes, repeating themselves throughout the input according to a known set of transformations.

Establishing the Pattern Visibility Map

With the sets of pattern classes in hand, we now study back the input. We assign a class member to each of its pixels in order to get the *pattern visibility map*. This step is mandatory for satisfactory occlusion management between patterns. It also has to robustly identify and lift any ambiguities that

would have arisen from spurious or partially-repetitive shapes.
As stated in 5.1, using an energy minimization framework would be NP-hard in our case and close to computational intractability. Instead, we propose an iterative method which labels every pixel, first with a unique class, then with one of its members. This two-step approach is necessary to effectively deal with members of a same class sharing a common boundary.
Pixels get ambiguous when they can be assigned more than one class member. Our goal is to pick one unique *visible* candidate. Our iterative approach strives to determine the visibility of the class members in a way that ensures "well-behaved" members. It means a minimal number of classes with a maximal number of visible members.
We start by restricting the candidate members to a single class. First, we flag as *reliable* pixels whose set of candidates are from the same class, and then propagate this initial information. By assuming its membership to different candidates, a pixel can be applied the transformation network of its associated class. Studying the locations where the pixel gets projected by those transformations is paramount to choose among its candidates. If, while studying the transformations of every candidate, the pixel gets projected onto a position where a reliable class has been determined, without conflict (the reliable class must match the candidate's) nor ambiguity (only one reliable class must be encountered), then we can assign this particular class to the pixel and flag it as reliable. This process is repeated while pixels keep on being labeled. Finally, the class of pixels left unlabeled after this step, is determined by picking the candidate class with maximal score. Those scores quantify "how well" the pixel, if assumed to belong to a class, behaves. Each candidate contributes to its class: we apply its transformations to the pixel and count the number of times it encounters shapes from the same class once projected.
Some pixels may still be ambiguous though, potentially belonging to different members of the same class. This occurs at boundary areas between different instances of the same pattern. We lift this last incertitude by applying a decision scheme similar to the one proposed for the classes: we use the transformations from pixels whose *visible* member is final to iteratively propagate constraints onto other unlabeled pixels. Next, scores are attributed to the candidate members of the pixel's selected class, and the member of maximal score obtains the pixel.

5.3 Pattern Extraction

Once the pattern visibility map has been established and all ambiguities lifted, extracting patterns becomes straightforward even if they partially overlap. Indeed, not only we know the positions of the patterns' different instances (corresponding to the classes' visible members), but also the transformations mapping one onto another. Given a pixel on one instance, we can easily compute its corresponding positions on the other instances and then obtain a consistent traversal across them. The final pattern representative is then computed by gathering its instances' pixel colors. To deal with possible occlusions, only pixels appearing at least on two visible instances are taken into account.

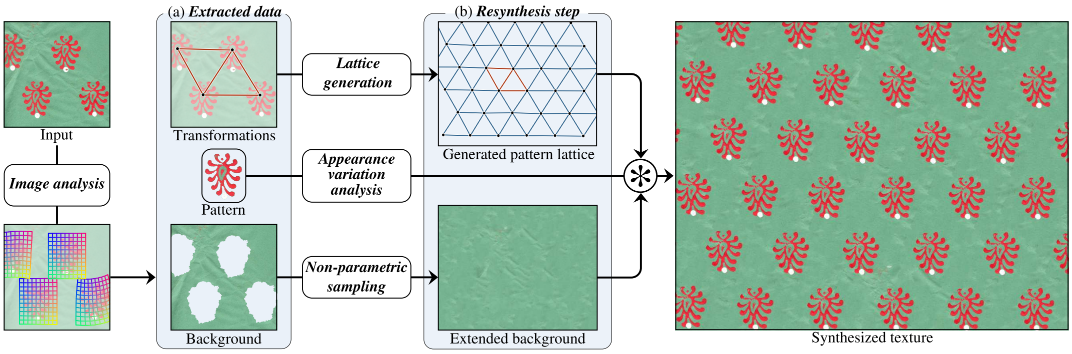


Figure 6: *Overview of our lattice-based texture synthesis. After the input's analysis, we dispose of meaningful extracted data* (a)*: its repetitive patterns are expressed as an instantiable object and the set of transformations towards its duplicates in the image.* (b) *From this, we generate a lattice of positions we can paste the original shape onto. The background is extended using non-parametric sampling texture synthesis [24]. As a result, we obtain a coherent distribution of patterns which closely mimics the given input.*

6 Results

Now having those patterns along with their relative positions at our disposal, we can re-synthesize new distributions of shapes and add high-level randomness to parameters actually defining the input's visual appearance. We now present several application scenarios and show how to apply existing techniques or possibly extend current work to take advantage of the texture representation we propose.

Generating Tileable Textures: The simplest application for our method is to convert non-tileable shape arrangement textures into tileable ones. This is made easy once one acquires the image's constitutive shapes. For stochastic placements, we just need to generate some random shape distribution with the only care to respect the output's toroidality when rendering shapes crossing the image's borders (*cf.* figure 1). For lattice-based distributions however, achieving tileability is slightly more involved as the transformation group defining the lattice must be compatible with the image's geometry.

Synthesizing Input-Consistent Distributions: If we think about the input shape distribution as a parameter of the output, one may want to generate a new distribution remaining close to the input's while being altered by an appropriate degree of randomness. Possibilities range from complete random distributions to faithful replications of the input. To handle the latter case, we build upon recent research in Expressive Rendering which broadened Markov random field-based texture synthesis to element arrangements. To achieve this, they extended the neighboring system, from the image pixel grid to the Delaunay triangulation of their input elements [3, 10]. We also build a Delaunay triangulation, over the extracted input instances' centers of mass and take a similar approach to generate new distributions out of it. Depending on the desired result, neighborhood matching is here controlled either by the shapes' positions, orientations or *ids* (*cf.* figures 8, 7 and 6).

Reintroducing Chromatic Variations: Since our analysis is performed on the input's lightness channel only, objects differing only in color are brought together since considered as visually equivalent. We can take advantage of this to study the chromatic appearance variations between instances of a same pattern. As those shapes are related to each other by non-rigid transformations, we place them in a common coordinate system and reduce the dimensionality of their RGB color distribution by PCA [16]. We then generate shapes of slightly varying appearance by modifying the coefficients of the original instance's decomposition onto the obtained eigenvectors (*cf.* figure 8). This treatment is however limited to the case of fully-visible pattern instances.

Handling the "Background": Figure-ground separation is an intricate problem which far exceeds the scope of our paper. In our context, the most straightforward definition for background pixels is pixels not instanced after analysis. And, even though counter-example images are easily produced, there are situations where such automatic background extraction becomes satisfactory (*cf.* figure 6). This is notably true when background pixels exhibit sufficient isotropy or cannot clearly be assigned a canonical orientation. In this case, we separately extend a wider background texture by using well-adapted non-parametric texture sampling techniques [24].

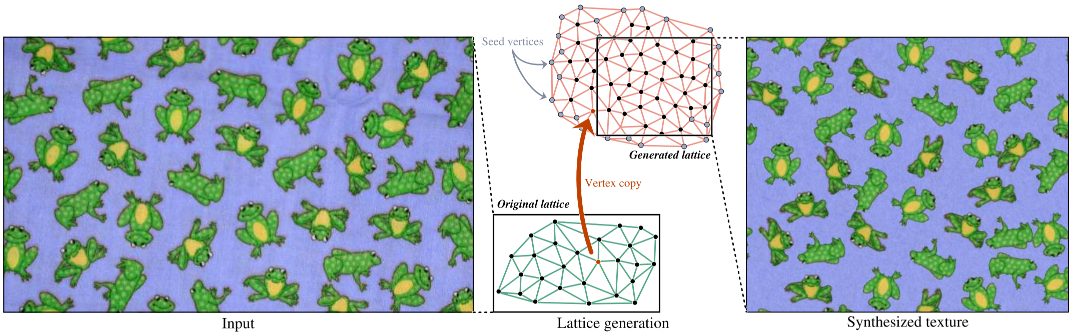
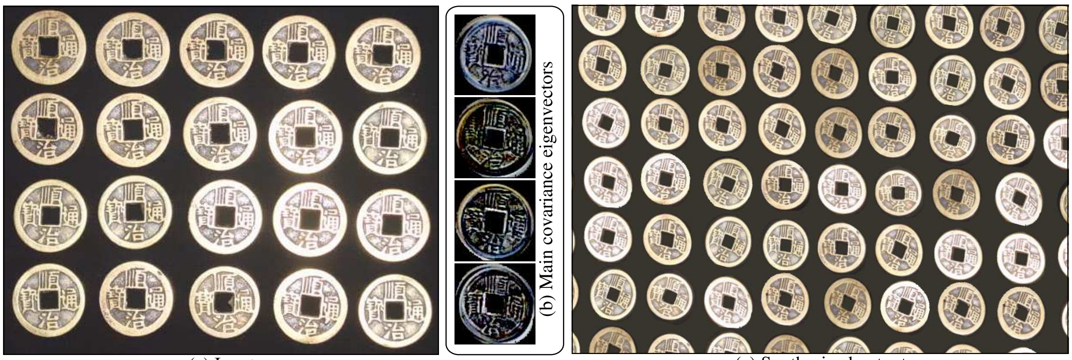


Figure 7: *Example where the input image contains different shapes. After extraction, the input lattice is copied using the technique of [3] and used to make a new image. The background is itself generated using non parametric image sampling [24].*

7 Discussion

7.1 Comparison to Related Work

Figure 10 shows comparative results between pixel-based existing techniques, namely non-parametric sampling synthesis [24] and graph cut textures [13], and our content-driven method. While they succeed in ensuring local continuity, they are bounded by the *a priori* fixed scale of the neighborhood



(a) Input (c) Synthesized output

Figure 8: *To account for local chromatic variations within the original distribution in the input image* (a)*, a Principal Component Analysis is performed over the different instances and the set of significant eigenvectors are extracted* (b)*, by using a technique similar to Liu and co-workers' [15]. This vector basis is further used to generate new instances throughout the new texture* (c).

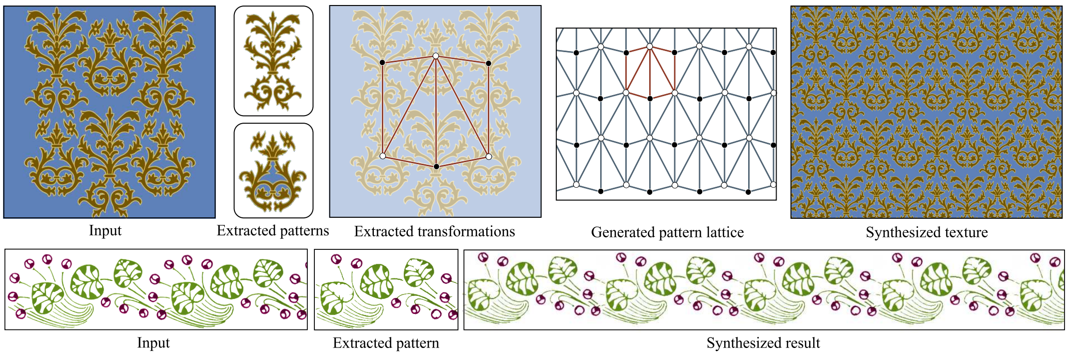


Figure 9: *Two examples of applying our texture analysis technique to redundant, yet not entirely tileable, images. The hand-drawn case is particularly challenging since SIFT descriptors get very discriminative in presence of high frequencies at the scale of pixels.*

used to evaluate visual similarity. Besides, unable to manipulate primitives other than pixels/patches, they fail to preserve actual structures and shapes end being mixed together.

7.2 Shortcomings

Occlusion Management: Though we do handle partially occluded shapes (*cf.* figures 1 and 11), some requirements exist. First, in order to be stitched together, occluded parts must share, up to a transformation, a significant overlap to end up in the same pattern class. Second, and more importantly, since our method is devoid of *a priori* knowledge, parts must be detected twice to be deemed of interest. Thus the integrity of the pattern must appear –even separately– throughout the image to be extracted.

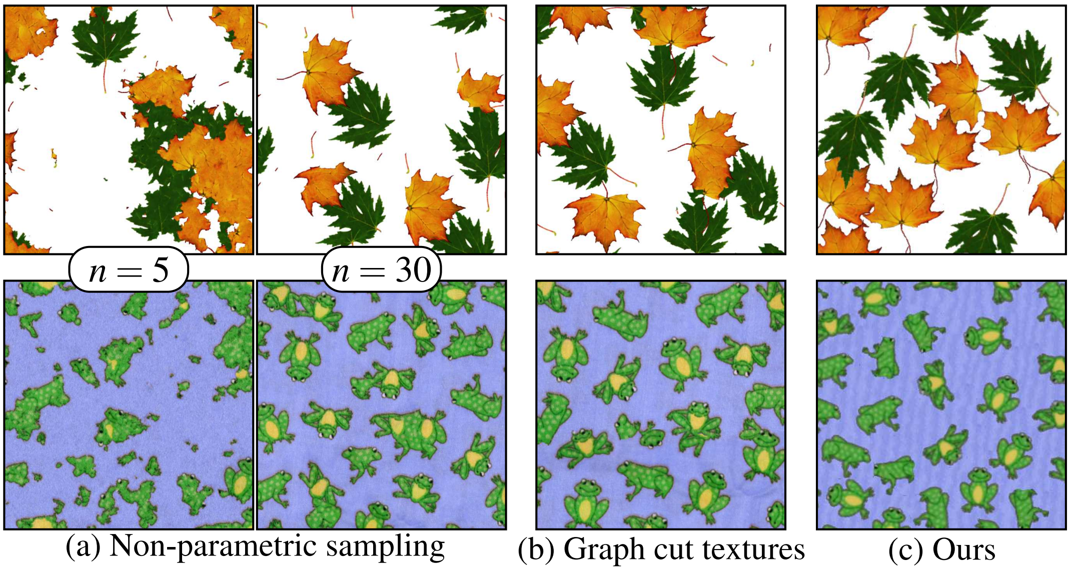


Figure 10: *Comparison with main existing techniques. By focusing on pixel-based artifacts, both non-parametric sampling* (a) *and graph-cut* (b) *techniques fail to preserve the input's recognizable shapes. Our method on the other hand manages to handle those limitations* (c). *Increasing the neighborhood size* n *does help capturing wider structures, but still without handling actual properties of the shape distribution.*

Invariance to 2d Similarities Only: Despite the use of thin-plate splines to confer more flexibility during shape extraction, our method relies on the SIFT descriptor and thus only detects shapes up to transformations close to 2d similarities. Reflexions for instance are not currently supported, for they would need to add to each pixel a "reflected" version of its descriptor in feature space and weight down computation costs.

Greedy Creation of Pattern Classes: Some unintuitive results find their explanation during the pattern class computation step (*cf.* Section 5.2). Once a connected component analysis onto a graph, this step is made dependent on the order of the visited shape nodes by the propagation of transformation constraints along the graph's edges. Several strategies for traversal have been tested but current implementation still needs improvements in that sense.

7.3 Computation Costs

The complexity of our algorithm is intrinsically $O(N^4)$ with N being possibly the total number of pixels since we look for clusters of pixel matches. Using discriminative descriptors, accelerated search structures and voxelling the transformation space is the key of its tractability. It also more directly depends on the input's gradient activity, intuitively speaking "textureness" than its resolu-

(a) Input (b) Extracted shape (c) Detected instances

Figure 11: *Illustration of the handling of partial occlusions.* (a) *If the input exhibits multiple instances of a single, but always partly occluded shape,* (b) *we can still build its representative pattern thanks to the consistent traversal of its instances allowed by the detected transformations.* (c) *However, its parts must appear at least twice along its recovered instances. This intrinsic limitation of our method is due to its unsupervised nature.*

tion since pixels whose SIFT descriptor shows weak magnitude do not take part in the following computations. Keypoint matching and shape extraction by region growing are the bottlenecks of our approach. Fortunately, the different parts of our algorithm comply well to parallelism and, in our implementation, all of them, except for the visibility map computation, benefit from multi-threading. The running times indicated below have been obtained on a 64 bit quad-core Intel(R) machine for the different results presented in our paper. References to the figures are given in parenthesis.

	Leaves (1)	*Coins* (8)	*Frogs* (7)	*Wrap* (6)	*Dollars* (11)	*Frieze 1* (9)	*Frieze 2* (9)
Description	4 s	5 s	12 s	3 s	3 s	18 s	2 s
Nb features	58790	77804	90251	36945	56272	147764	41507
Feature dim	22	22	22	21	22	23	22
Matching	**24 s**	**153 s**	**126 s**	**9 s**	**26 s**	**316 s**	**32 s**
Clustering	2 s	3 s	4 s	1 s	2 s	548 s	4 s
Growing	**123 s**	**172 s**	**190 s**	**73 s**	**55 s**	**546 s**	**112 s**
Recovery	5 s	7 s	8 s	1 s	3 s	30 s	8 s

8 Conclusion and Future Work

In this work we have presented a new analysis method for shape arrangement textures which, contrary to existing techniques, aims at the explicit, unsupervised detection and extraction of the input's constitutive *patterns*. Once they have been recovered, re-synthesis can base itself on higher-level, more meaningful building blocks than individual pixel colors. It can also concentrate on capturing and reproducing relationships between instanced patterns instead of neighboring local pixel variations. We detailed several application scenarios taking advantage of the better insight on the input our content-driven representation can yield. By directly dealing with raster input samples, our gait

also enables to use advanced, recent regeneration techniques once confined to vectorized elements. Originally designed to tackle non-parametric sampling techniques' weaknesses, our work targets at the handling of cases particularly challenging for traditional example-based texture synthesis. As a counterpart, we can only apply our method to a restricted range of textures.
In the future, we would like to strengthen our technique's robustness to natural images where slight gradient distortions due to perspective or object deformation can endanger the success of our results. To that aim, we would further investigate other local descriptors, either encoding different or complementary visual features of images (intensity- or color-based description) or invariant to more challenging transformations than 2d similarities. A first step toward this latter goal would be to apply Lindeberg's affine adaptation prior the computation of the SIFT descriptors, thus granting affine invariance to Lowe's descriptor. More careful examination of the transformation space could also prove highly beneficial: detecting grid-like placement of match aggregations would lead to the early discovery of transformation groups (such as the ones generating congruent periodic textures) which could be used for optimal shape extraction instead of pixel-based region growing.

References

[1] N. Ahuja and S. Todorovic. Extracting texels in 2.1d natural textures. In *ICCV '07*, pages 1–8, 2007.

[2] Michael Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D graphics (I3D)*, pages 217–226, 2001.

[3] Pascal Barla, Simon Breslav, Joëlle Thollot, François Sillion, and Lee Markosian. Stroke pattern analysis and synthesis. In *Computer Graphics Forum (Eurographics '06 Proceedings)*, volume 25, pages 663–671, 2006.

[4] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *SIGGRAPH '97*, pages 361–368, 1997.

[5] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24, pages 603–619, 2002.

[6] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 241–251, 1992.

[7] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01*, pages 341–346, 2001.

[8] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV '99*, pages 1033–1038, 1999.

[9] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH '95*, pages 229–238, 1995.

[10] Takashi Ijiri, Radomír Mech, Takeo Igarashi, and Gavin Miller. An example-based procedural system for element arrangement. In *Computer Graphics Forum (Eurographics '08 Proceedings)*, volume 27, pages 429–436, 2008.

[11] N. Jojic, B.J. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *ICCV '03*, pages 34–41, 2003.

[12] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. Texture optimization for example-based synthesis. In *ACM Transactions on Graphics (SIGGRAPH '05 Proceedings)*, volume 24, pages 795–802, 2005.

[13] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (SIGGRAPH '03 Proceedings)*, volume 22, pages 277–286, 2003.

[14] Tony Lindeberg. Feature detection with automatic scale selection. In *International Journal of Computer Vision*, volume 30, pages 77–116, 1998.

[15] Yanxi Liu, Robert T. Collins, and Yanghai Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 26, 2004.

[16] Yanxi Liu, Wen-Chieh Lin, and James Hays. Near-regular texture analysis and manipulation. In *ACM Transactions on Graphics (SIGGRAPH '04 Proceedings)*, volume 23, pages 368–376, 2004.

[17] D.G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99*, volume 2, pages 1150–1157, 1999.

[18] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 27, pages 1615–1630, 2005.

[19] R. Paget and I.D. Longstaff. Texture synthesis via a noncausal nonparametric multiscale markov random field. In *IEEE Transactions on Image Processing*, volume 7, pages 925–931, 1998.

[20] Javier Portilla and Eero P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. In *International Journal of Computer Vision*, volume 40, pages 49–70, 2000.

[21] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *CVPR '08*, 2008.

[22] Huamin Wang, Yonatan Wexler, Eyal Ofek, and Hugues Hoppe. Factoring repeated content within and among images. In *ACM Transactions on Graphics (SIGGRAPH '08 Proceedings)*, pages 1–10, 2008.

[23] Li-Yi Wei, Jianwei Han, Kun Zhou, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Inverse texture synthesis. In *ACM Transactions on Graphics (SIGGRAPH '08 Proceedings)*, pages 1–9, 2008.

[24] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH '00*, pages 479–488, 2000.

[25] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. In *ACM Transactions on Graphics (SIGGRAPH '04 Proceedings)*, volume 23, pages 364–367, 2004.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399