

Aberystwyth University

Image completion by structure reconstruction and texture synthesis

Alotaibi, Najm; Labrosse, Frédéric

Published in:

Pattern Analysis and Applications

DOI:

[10.1007/s10044-013-0348-4](https://doi.org/10.1007/s10044-013-0348-4)

Publication date:

2015

Citation for published version (APA):

Alotaibi, N., & Labrosse, F. (2015). Image completion by structure reconstruction and texture synthesis. *Pattern Analysis and Applications*, 18(2), 333-350. Article n/a. <https://doi.org/10.1007/s10044-013-0348-4>

General rights

Copyright and moral rights for the publications made accessible in the Aberystwyth Research Portal (the Institutional Repository) are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the Aberystwyth Research Portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the Aberystwyth Research Portal

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

tel: +44 1970 62 2400
email: is@aber.ac.uk

Image completion by structure reconstruction and texture synthesis

Najm Alotaibi & Frédéric Labrosse
Department of Computer Science
Aberystwyth University
United Kingdom
ffl@aber.ac.uk (F.L.)

Abstract

We present a new image completion method that can deal with large holes surrounded by different types of structure and texture. Our approach is based upon creating image structure in the hole while preserving global image structure, and then creating texture in the hole constrained by this structure. The images are segmented into homogeneous regions. Similar regions touching the hole are linked, resulting in new areas in the hole that are flood-filled and made to match the geometry of the surrounding structure to provide a globally spatially coherent and plausible topology. This reconstructed structure is then used as a constraint for texture synthesis. The contribution of the paper is two-fold. Firstly, we propose an algorithm to link regions around the hole to create topologically consistent structure in the hole, the structure being then made to match that of the rest of the image, using a texture synthesis method. Secondly, we propose a synthesis method akin to simulated annealing that allows global randomness and fine detail that match given examples. This method was developed particularly to create structure (texture in label images) but can also be used for continuous valued images (texture).

1 Introduction

Images sometimes contain regions that are flawed in some respect. Fixing these flaws correctly is important in many applications such as photo editing, wireless transmission of images (recovering lost blocks), and film post-production. The problem of image completion is, given an original image and a hole mask, to automatically fill in the area of the image corresponding to the hole such that the synthesised part seamlessly fuses with the rest of the image. Another constraint is that the new content must be plausible, i.e. is similar to, but not a copy of, the rest of the image.

Typically, such completion propagates new texture from the surrounding areas into the hole by “copying” similar examples from such areas. This prop-

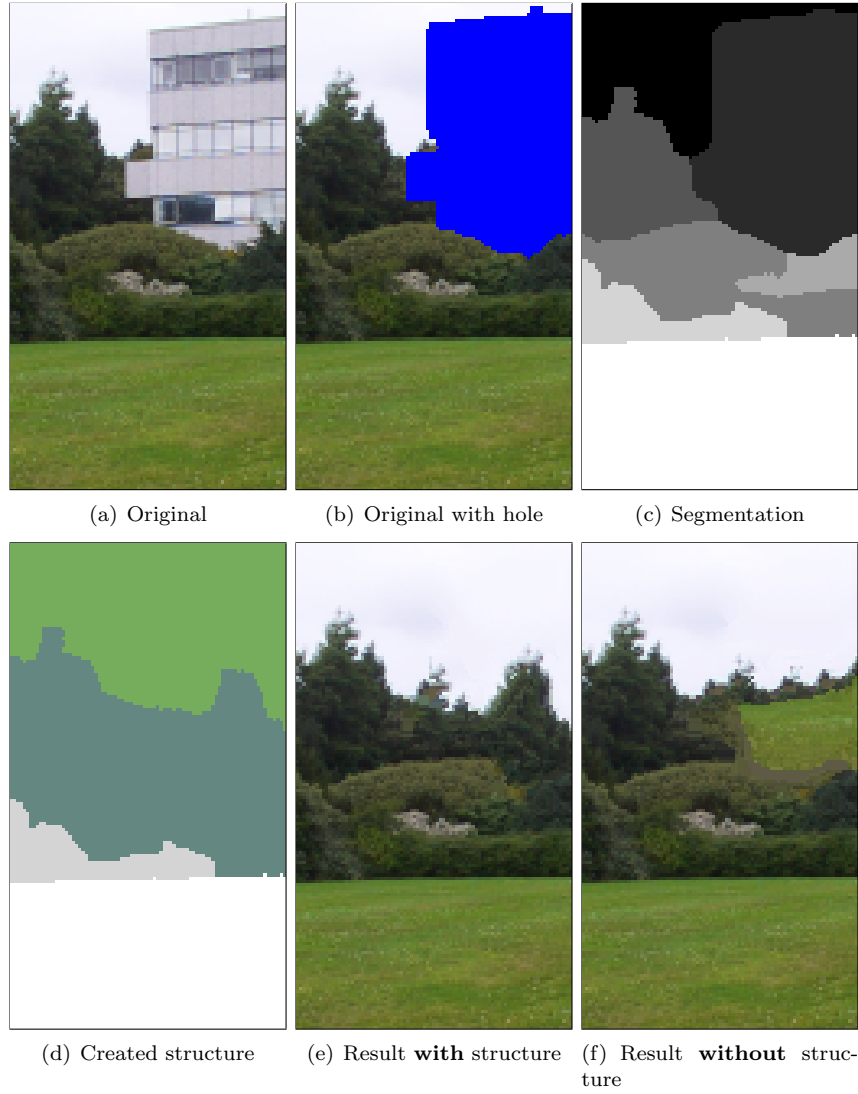


Figure 1: Result on the CLEDWYN image (image: 85×150 , neighbourhood: 11×11)

agation should produce realistic structure and texture inside the hole so that it matches the surroundings with no visual artefacts. The hole can be of any size and shape, and may be surrounded by many different areas. Figure 1 shows an example of applying our image completion method. In contrast to image completion where holes are typically large, image inpainting is usually the filling-in of small defects, perhaps created by scratches in analogue images, sensor noise,

etc., typically a few pixels wide, but possibly elongated¹.

During the last decade, there has been substantial work dedicated to dealing with the problem of image completion and current results are very convincing for many types of images. However, existing methods may produce visual artefacts such as hole boundaries, blurring, or the replication of large blocks of texture. Some simply cannot preserve the structure in the image, especially when the hole is large and surrounded by different types of texture. Methods to remove image flaws have used different approaches and strategies including completion using structure, completion using texture, and completion using both structure and texture. The methods of the first category fill in holes based on propagating linear structures often using mathematical models such as partial differential equations (PDEs), e.g. image diffusion [4, 8, 9]. They are more suited to image inpainting as the process is local and can cause blurring artefacts. The second group of methods completes holes by creating texture that is copied or modelled from the surrounding examples of texture [21, 15, 39, 7, 13, 42, 20]. This type of approach produces good results for a large variety of textures. However, it does not ensure the structure of the image remains correct, as many of these methods are based on local sampling, which ignores global image structure. The third category of methods combines the two approaches by creating texture that is constrained by structure. These methods have been successful in completing holes in images, by propagating structure into the hole and then creating texture based on that structural constraint [17, 25, 5, 22, 10, 33, 38, 36, 3, 34, 6].

1.1 Texture synthesis methods

Many texture synthesis methods have been proposed in the literature, all falling roughly in three categories: statistical, PDE-based and exemplar-based [23]. Many authors conclude that only exemplar-based methods can succeed in filling in large holes, and we therefore use such a method in this work. However, these methods are not without problems, especially when one wants to deal with label images [6]. We briefly describe the generic method on which many current, popular texture synthesis methods are based, and its limitations along with some variations. Section 2 will describe our modifications to this generic method so that it can be used in the context of image completion and, in particular, for label images.

The generic method is based on the pioneering work in [15], in which pixels being synthesised are given a value by matching their neighbourhood to similar pixel configurations elsewhere in the image. In the context of image completion, possible matches are only considered outside the hole. The neighbourhood of the current pixel is compared with the neighbourhood of possible matches to build a list of good matches. From that list, one match is selected and its centre value is used as the new value for the current pixel. A histogram of possible values from “good” neighbourhood matches is built for each pixel to

¹We understand that some authors have used the term ‘inpainting’ to mean general image completion. We limit the meaning to completion of small areas, as described in the text.

be synthesised. “Good” is any match for which the distance (often SSD or Euclidean, of the colours) is lower than a fixed threshold, relative to the best match. This histogram is then used to select the new pixel value using the frequency of the value as the selection criterion. Most publications using that same method, however, use the value of the best match (e.g. [40, 26, 17]), or one chosen at random from the selected matches (e.g. [5]), possibly with a bias towards the best match [25]. Using the best match, or a low threshold, coupled with a sequential order of the filling in, and in some case an *approximate search* and/or *coherence search* [17, 2, 3], can lead to replicating large areas of the sample texture, or at least bias the selection of matches towards sub-regions of the sample texture [26]. However, using too high a value for the threshold can lead to too much randomness in the result. In [6] the problem of replicating large areas is solved by greedily rejecting potential candidates if they have been used for nearby pixels. Introducing some randomisation in the selection process also helps. This can be done with an initial seed [39, 40] and/or at the match selection process [25, 3, 16].

Some methods also introduce geometrical and lighting transformations during the process of matching patches, allowing for such transformations to be introduced in the texture (and possibly better matches in some specific images). This however increases the complexity of the search, which is a problem tackled in [29] where a combination of approximate search (as in [3]) and continuous optimisation is performed.

Randomness is important for natural images and avoiding repeated areas is paramount. However, many man made scenes do have repeating patterns and recreating these is also paramount. The problem with the use of fixed thresholds as in [15] is that the randomness of the created texture is dependant on the value of the threshold, yet the appropriate threshold depends on the texture. Finding the right threshold value is therefore not trivial. Linked to this is the scale at which the texture is considered. In the original method, the size of the neighbourhood defined the scale. A large neighbourhood can capture larger features. However, a large size can break the MRF assumption, that is that a pixel’s value only depends on the values present in its neighbourhood, which is at the basis of these methods. A large neighbourhood size also makes the synthesis slower.

To solve this problem multi-scale methods have been proposed, creating a pyramid often using Gaussian smoothing [11, 39] or edge preserving smoothing [19]. These will typically synthesise each scale using the previous one as a seed, with the first being random. In [26] a Gaussian stack (Gaussian pyramid where all levels keep the same resolution) is used and texture is synthesised from coarse to fine levels by introducing randomness from one level to the next (jittering of match positions) and only running a “correction step” (essentially using [15]) at the finest of levels.

Others use iterative schemes that perform a (near) global optimisation. In [41] global spatial consistency is ensured by enforcing (via an optimisation) that all the neighbourhoods containing a given pixel match with the synthesised image, not only the one neighbourhood having the pixel at the centre. The

colour for a given pixel is then computed optimally from all neighbourhoods that contain that pixel. This can result in an exact copy of areas of the image into the hole. To solve this, hole pixels are down weighted compared to non-hole pixels in the similarity function, using yet another parameter [41]. This approach and that in [29] are more appropriate to copying coherent small details that exist elsewhere in the image to complete the hole. In [23] a belief propagation method is used that explicitly represents the interactions between neighbouring pixels and the outside of the hole (the absolute belief), in a message passing, iterative optimisation. The pixels are considered in order of belief which, as computed, roughly corresponds to the boundaries of homogeneous regions in the image.

Usually, pixels are considered in some order such as raster scan [39], or spiral from a central seed (in the case of unconstrained texture synthesis [15]). Such systematic order will tend to create visible boundary artefacts. Parallel synthesis has been used in [40] to remove this effect and in [26] to allow GPU computation.

Note that some methods copy entire patches rather than individual pixels, e.g. [14, 24, 27, 28, 32]. This has the advantage of reducing synthesis time but can lead to large patches being copied and require the additional step of seamlessly merging neighbouring patches.

1.2 Hole filling in methods

Most texture synthesis methods show limitations when filling in holes surrounded by different textures. One is that these methods need a set of specially tuned parameters that will work with a set of textures but not with others. Moreover, for images presenting different types of texture, it seems that the use of a structural constraint to make sure only the right texture is synthesised in the right place is primordial (Figure 1(f) and [30, 25, 22, 33]). Another problem is that boundaries between the various synthesised textures propagated from the surroundings of the hole must look realistic in the hole.

The problem of structure has been addressed in [10] by propagating image structure into the hole by first synthesising the pixels with a high gradient, along the isophote direction. In [30], level lines in grey-level images ending at the hole boundary are linked across the hole using curves that are as short and straight as possible, therefore being more appropriate for inpainting. In [35], a curvature prior is used to connect user defined boundaries across a hole. To achieve this, an MRF model selects the best matching pattern from a curvature prior that is learnt from a set of training shapes, and could perhaps be constructed from exemplars in the reconstructed image (as suggested by the authors of [35]). In [5] a decomposition of the image into two components is proposed. The first is the structure, the second being the texture (and noise). The two components are then independently filled in and then recombined to create the final image. The structure filling in is done using inpainting [4], and therefore only works for small holes, while the texture is synthesised using the method proposed in [15]. Because the two steps are done independently, it is possible that the final structure and texture might not match spatially, particularly in the case

of large holes. Similarly, a two-step hole completion is used in [36]. The first step creates a gradient image using a method that is essentially the same as in [10] but with a different similarity measure, taking into account not only colour but also gradient, and a different metric. The gradient image is then used to compute a colour image by solving the Poisson equation. The gradient image constitutes a structure image in that it will guide the colours in the final image. However, the method shares the same limitations as [10], being local and greedy.

In [19] an implicit region based constraint is used. Indeed the smoothing method used to create the multi-scale pyramid preserves high gradients, i.e. boundaries between regions. This is what makes that method work well (results obtained with Gaussian smoothing alone were worse). However, boundaries only appear at the coarsest of smoothing levels. Smoother transitions between textures might not be preserved by the smoothing procedure, and would therefore be lost by that method.

In [25] the image is first segmented into homogeneous regions and boundaries between these regions terminating on the edge of the hole are propagated into the hole using existing boundaries as exemplars for their shape. The process creates regions containing homogeneous textures, which are then filled in using a method based upon that of [15] but using the segmentation image as a constraint on where to consider possible matches.

These methods cannot control the correctness of the created topology as they use local processes. To create plausible topology, curves that will link structures outside the hole can be created in the hole. For example, in [38] users draw lines that are used to first propagate texture along them, creating the structure. In [3], a more elaborate interaction is possible whereby users can specify specific areas of the image to use as source pixels in specific areas of the hole. Methods to automatically create such curves in the hole have been proposed. In [22], tensor voting is used to create smooth curves that link regions that are similar (differences in intensity and intensity gradient below a set threshold) but separated by the hole. The smoothness of the curves is controlled by a parameter that needs to be specified so that the smoothness matches the curves outside the hole (with potentially multiple parameters per image, although this is not mentioned). Only simple topologies can be tackled (the provided results are all topologically simple and the algorithm cannot cope with intersecting curves). In [33], arcs of circles are used to link region boundaries and straight lines for boundaries that cannot be paired. Boundaries are paired based on circle fitting, corresponding region matching and sequentiality (to reduce the number of boundary crossings in the hole). An almost global optimisation is performed to find the best set of pairs of boundaries. Boundary and region matching is local, only performed in a belt around the hole, and regions that are too different are removed from the optimisation. Although in theory complex topologies can be tackled, no such examples are given.

Constraint images have been used to create a number of effects. For example, in [34] a layered (weathered) surface is simulated by creating a label image that has the right properties for the surface being simulated and that is then used to constrain texture synthesis to produce the right texture at the right place.

Famously, “texture-by-numbers” is proposed in [17] whereby a new image is user created by painting a label (structure) image with labels coming from an annotated known image, and then used as a constraint for texture synthesis.

Similarly, a method is proposed in [6] to create new images (not fill in holes) from a label image (the proxy), each label corresponding to known labels of a source image (typically manually specified). What is new in this method compared to [17] is that the proxy (initially hand drawn) is improved to match the example using an exemplar-based method that uses a Chamfer distance (instead of L_2) and a label-based voting pyramid. The Chamfer distance measures spatial proximity of labels, rather than overlaps, which is good for small features in label images (e.g. lines) but loses spatial coherence of the pixels (only the spatial distance to the nearest pixel of the same label is kept). Also, the labels correspond to semantically distinct regions of the source image (such as sky, mountain, lake) rather than corresponding to distinct textures. This means that the proxy does not necessarily help as a constraint for the texture synthesis.

To our knowledge, no published work attempts to create structure that matches the image to be completed, the nearest work being [6] where manually created boundaries are made to match similar ones in an example image. All other works on image completion create boundaries in the hole that are as straight as possible, rather than trying to match existing boundaries. Moreover, these works do not allow created regions to overlap or cover each other in the hole (manually specified depth information is used to allow this in [6]).

1.3 Method overview

We propose a method that attempts to automatically create structure, preserving global topological consistency when possible (i.e. examples exist outside the hole) and plausible topological consistency when no examples exist. The method also creates texture over the constructed structure such that no visible visual artefacts are present and the created structure and texture seamlessly fuse with the surrounding areas, not showing any visible boundary. Finally, the new content is meant to be similar to, but not replicate, the rest of the image.

The *structure* of an image is its decomposition into regions that correspond to identifiable parts. In this work, the parts will correspond to homogeneously coloured regions. The *topology* of the structure will describe how the regions that constitute the structure are connected to each other.

We can tackle complex topologies in a recursive manner. Because we do not use any high level information about the content of the images (in particular occlusion information) the automatically produced topology can be wrong and some manual intervention may be necessary.

Finally, the actual shape of the curves is matched to exemplars from the rest of the image rather than being arbitrarily shaped. As others have done, this structure is then used to constrain texture synthesis. We use the same method to create the boundaries of the structure and the texture of the final image. We propose a modification of the method in [15] that allows the convergence of the synthesis, akin to simulated annealing, in a coarse to fine scheme, that can

Algorithm 1 Overview of our method

- 1: Segment image (with hole, Figure 2(a)) into homogeneous regions (Figure 2(b))
 - 2: Create topology in the hole by linking similar regions across the hole (Sections 3.1 – 3.5)
 - 3: Match new boundaries to existing ones using *structure synthesis* (Figure 2(c), Sections 2 & 3.6)
 - 4: Use the new segmentation as a constraint in *texture synthesis* to produce the result (Figure 2(d), Sections 2 & 4)
-

tackle varied textures (and indeed labels, for which it was primarily created) providing both randomness and fine details.

An overview of our method, along with a graphical example and the corresponding sections where each step is described, is given in Algorithm 1. We start by describing the structure/texture synthesis method in Section 2, despite being used after the creation of the structure in the hole in Sections 3.1 to 3.5. This is because choices made for the construction of the structure in the hole depend on the structure/texture synthesis method. Sections 3.6 and 2 describe how the structure and texture are made to match that of the rest of the image. Section 4 presents results. We conclude with a discussion of the method and explore future avenues in Section 5.

2 Texture synthesis method for image completion

In this section we present our synthesis method. The first step of our method is to synthesise an image structure that matches the surroundings of the hole (Step 3 in Algorithm 1). The structure is encoded in a label image, each label corresponding to a region of the image. Therefore, our synthesis method was developed specifically with label images in mind. It can however also synthesise continuous valued images, such as texture images. Although both types of images are synthesised in the same way, there are subtle differences, which are highlighted.

Our method follows the general idea of the generic method presented in Section 1.1. We now present what makes our method different.

2.1 Parallel synthesis

Because the structure synthesis we present later in Section 3 uses a segmentation of the image, we cannot rescale images as small regions could disappear at lower resolutions. Instead, we use a parallel synthesis scheme similar to the one in [40]. At each iteration, the new value of any given pixel is independent of the new value of the other pixels and is therefore not affected by the order in which

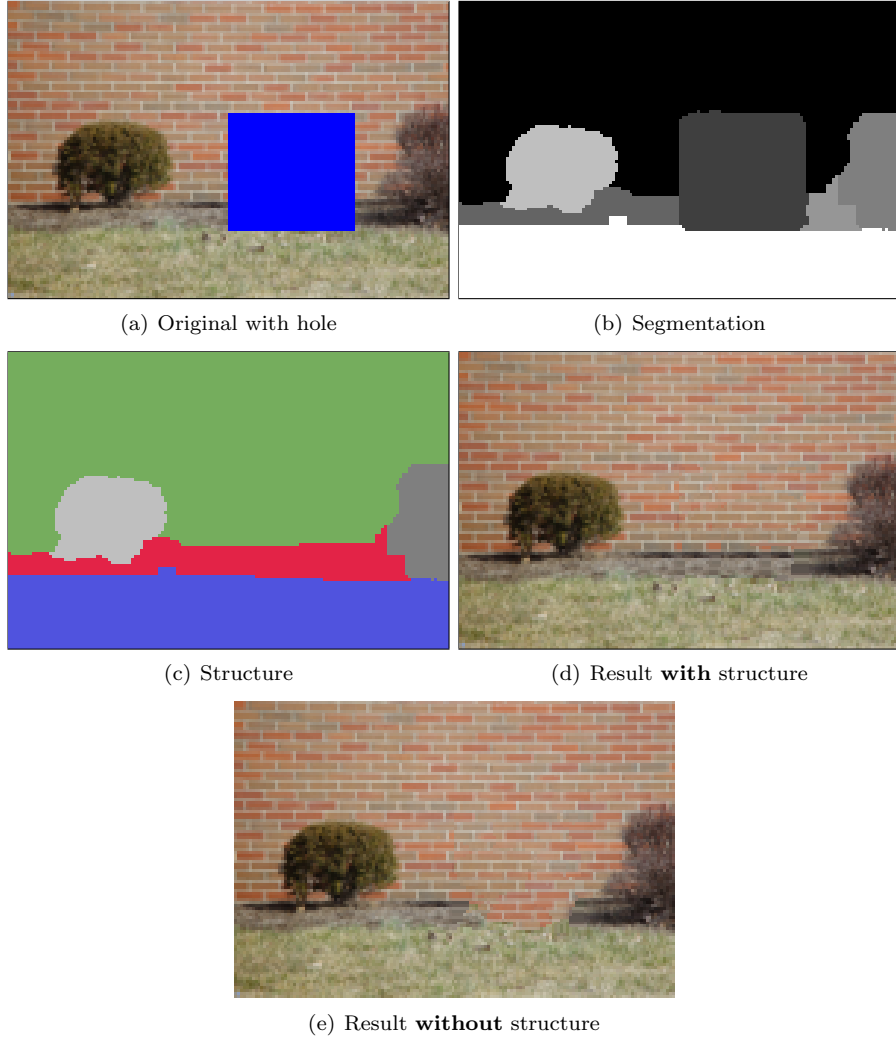


Figure 2: The BUSH image (image: 150×101 , neighbourhood: 15×15)

the pixels are considered. In practice, a new image (temporary buffer) holds the values of the synthesised pixels. This temporary buffer is then copied back to the image being synthesised at the end of each iteration, when all the pixels have been processed.

As with the generic method, a pixel is given the value of the centre of a good match selected at random from a list determined using a fixed threshold on the distance between the target neighbourhood and source neighbourhoods from the rest of the image. However, the threshold is updated as the iterations progress (see Section 2.3).

A square, symmetric neighbourhood is used (the figure captions include the image and neighbourhood sizes). In the case of texture images, neighbourhoods are compared using the Euclidean distance between the RGB pixel values of all pixels, where only neighbourhoods belonging to the correct region (centre pixel in the same region as the synthesised pixel) are considered.

In the case of structure (label) images, a different metric is used:

$$d(N_s, N_t) = \sum_i \bar{\delta}(N_s(i), N_t(i)), \quad (1)$$

where N_s and N_t are the source and target neighbourhoods and $N(i)$ is the i th pixel of the neighbourhood. $\bar{\delta}$ is the complement of the Kronecker delta function:

$$\bar{\delta}(i, j) = \begin{cases} 0, & i = j, \\ 1, & i \neq j. \end{cases} \quad (2)$$

This counts how many pixels have a different label, rather than how different the pixels are. No constraint is used in this case, and neighbourhood matching is left to wander everywhere in the image. However, it is unlikely that any neighbourhood not sharing any label will be selected.

To be able to use a complete square neighbourhood, initial pixel values must be provided for all the pixels in the hole. This is now described.

2.2 Initial hole filling

The aim is to provide initial values for the unknown pixels of the hole that are plausible yet have a certain amount of randomness. In [40], values are randomly copied from the texture example. Following this idea, we copy from nearby pixels of the image that do not belong to the hole, and belong to the same region for texture synthesis only. This is done using a Gaussian distribution centred on the current pixel. The distribution allows us to obtain random positions close to the current pixel. The standard deviation of the distribution is initially set to 1 pixel, implying that 99.7% of the positions will be within 3 pixels of the current pixel. If after a number of randomly selected positions (1000), none fall outside the hole and in the correct region (texture synthesis only), the standard deviation is increased by one pixel and the process is repeated. Figure 3 shows the process for a large value of the standard deviation.

For structure synthesis, we use information about the regions surrounding the hole to first explicitly create regions in the hole, Section 3. Random filling is only used for these areas that are not covered by a region.

2.3 Iterative synthesis scheme

To avoid having to choose the threshold used for neighbourhood selection and to provide an optimisation to the synthesis method, we use an iterative scheme akin to simulated annealing. This allows a coarse synthesis at the initial stages, using a large value for the threshold, creating random features. This randomness

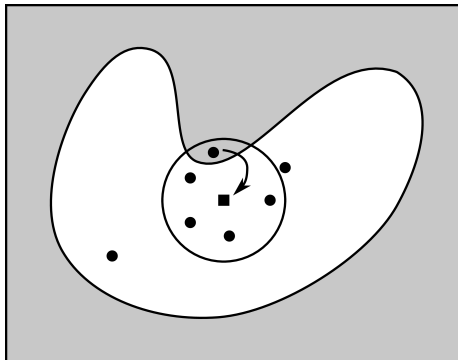


Figure 3: Random initial filling in

is then forced to converge towards detailed features that match the exemplars more precisely, using a lower threshold. A complete step of the parallel synthesis is performed for each threshold value.

More precisely, the threshold $T = d \times r$ is used in the match selection process, where d is the best match distance and r is a randomness factor. We set $r = 1.2$ at the first iteration. The threshold is then reduced by decreasing the value of r by 0.01 at each iteration. The initial value for r and its decreasing rate were determined experimentally. Their exact value is not critical because of the iterative nature of the process. However, a sharp decrease in r could lead to the convergence of the process to a local minimum, which typically would create repetitive texture, often at low scale (i.e. small details). This behaviour is similar to simulated annealing, which can converge to a bad local minimum if annealing is too fast.

The automatic termination of the synthesis can be based on the Euclidean distance between consecutive synthesised images, as reported in [2, 37]. However, the Euclidean distance is not a good measure of visual quality as the behaviour of visual similarity between two synthesised textures can vary depending on the particular texture [2]. Instead, the termination is based on the percentage $\mathcal{C}(t)$ of changed pixels in the hole compared with the previous iteration, as a function of time. The process is stopped when $\mathcal{C}(t)$ falls below a specific threshold (0.1 for the texture synthesis, 0.002 for the structure synthesis, lower because of the discrete nature of the labels). Experiments reported in [Ref. anonymised] show that $\mathcal{C}(t)$ decreases as the process progresses and that the synthesised textures or structures do not change significantly after a while, and that the given thresholds constitute a good compromise between quality of the result and number of iterations.

This simulated annealing process coupled with parallel synthesis not only removes the order dependency of the synthesis but also offers a more spatially global solution to the synthesis, which is what provides, in particular, the realistic (and topologically correct) boundaries between regions of the hole, Section 3.6.

Algorithm 2 Overview of structure creation (step 2 in Algorithm 1)

- 1: Create pairs of hole bounded regions (HBRs) based on their similarity (Section 3.1). Save the colour statistics.
 - 2: Create a list of pairs of HBRs that can be linked (the Region Connections Priority List, RCPL) and re-label HBRs belonging to the same pair (Section 3.2).
 - 3: **while** Not empty RCPL **do**
 - 4: Sort the RCPL according to HBR distance within pairs (Section 3.3).
 - 5: For each pair in the RCPL, draw the linking lines, flood-fill the region, remove the pair from the RCPL (Section 3.4).
 - 6: First iteration only: draw self-connections and flood-fill the corresponding regions (Section 3.4)
 - 7: Possibly add new regions that match regions from step 1 to RCPL (Section 3.5).
 - 8: **end while**
-

After describing the synthesis method that will be used to make the structure and texture in the hole match that of the rest of the image, we now describe how structure is created in the hole.

3 Reconstruction of image structure

The reconstruction of the hole structure is based on the assumption that distinct image areas (particularly in natural images) tend to be spatially continuous and are only separated by the hole: these must therefore be linked across the hole. Images are first segmented into homogeneous regions and those around the hole that are similar, based on colour statistics, are identified. Indeed, two regions with similar properties touching the hole should correspond to the same real area of the scene and should therefore be linked. However, spatial proximity between regions is also to be considered: regions that are close-by are more likely to correspond to the same area of the scene than regions that are far apart. This spatial information is used to order pairs of similar regions. This order is used to recursively link selected pairs of similar regions, using straight lines, and the newly created regions are flood-filled to create the corresponding regions. Areas of the hole might be left unfilled (e.g. due to non-matching regions). These are randomly filled-in. The created structure in the hole is topologically plausible but geometrically does not yet match the regions surrounding the hole. This is adjusted using the synthesis method described in Section 2.

The segmentation of the image is an important part in identifying the different regions. In this work, we have used the segmentation method in [12] called JSEG, which we found to work well for a wide variety of colour-textured images; this method has been reported to be one of the best available segmentation methods [22, 33].

Algorithm 2 describes the various steps of the structure synthesis method.

We start with matching regions that are touching the hole.

3.1 Hole Bounded Region (HBR) matching

A Hole Bounded Region (HBR) is a region, created by the segmentation, that intersects the hole. Based on the assumption that regions tend to be spatially continuous and are only broken up by the presence of the hole, the regions created by the segmentation may have to be relabelled to ensure that similar regions separated by the hole are indeed considered to be the same region. However, without high-level knowledge about the image, the relabelling can create false or too numerous possible matches that would be impossible to reconcile. Therefore, because it is more likely for nearby regions to match, connections between nearby regions will be preserved over connections between distant regions.

Similarity is based on colour histograms. The normalised colour histogram of each HBR is collected and the Euclidean distance between histograms is computed. In all our experiments we used the RGB colour space and 256 bins per colour component.

Pairs of matching regions are selected in a two-step filtering process. The first step ensures that only good matches are selected. This is done using an absolute threshold on the distance between histograms (0.2 in all our experiments); pairs for which the distance is below that threshold are selected as possible matches. The second step selects the best matches. It uses a threshold that is a function of the lowest distance d_m between pairs of histograms selected at the previous step (the threshold was set to $1.2 \times d_m$ in all our experiments). This two-step procedure ensures that only good matches are selectable (through the absolute threshold) and that the best matches are selected. If only the second threshold was used, then it would be possible for bad matches to be selected, depending on the quality of the best match. Note that at this stage, some regions can be selected as part of several pairs.

The value of the two constants was determined experimentally and represent a good compromise between selecting only regions that really match and creating too many pairs of regions. A high value of the absolute threshold will lead to a large list of potential matches that might not match well and will need to be weeded out at the next step. A too low value might lead to not linking regions that should have been matched. Two such cases are shown in Figures 7 and 9. A high value of the multiplicative constant in the second threshold can lead to a large list of retained matches. However, these should all be good matches (because of the absolute threshold). This will lead to more iterations when the regions will be linked (see Section 3.5). A low value of this same constant will lead to too few matches being kept and therefore the need for more random initial filling in (Section 2.2) and potentially erroneous structure being created, as in Figure 9.

Once pairs of similar regions have been established, the regions must be linked in the hole. A description of this process follows.

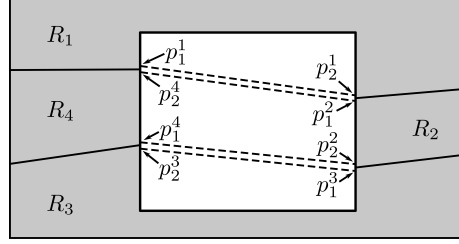


Figure 4: HBRs and their HBR-BEs. The hole is in white. p_1^i and p_2^i are the two HBR-BEs of HBR R_i . Potential self- (R_1 and R_3) and pair-connections (R_2 and R_4) are shown with dashed lines.

3.2 HBR connections

A HBR has, as part of its boundary, a section of the boundary of the hole. The two HBR boundary extremities (HBR-BEs) of this section are labelled in a consistent (clockwise) order and are noted p_1^i and p_2^i as the first and second extremities of the boundary between the hole and the HBR i . The clockwise ordering is there to prevent the creation of region pair connections that cross each other. Pairs of HBRs can be linked, using the HBR-BEs in a sequential order, to create a closed region in the hole. HBRs that cannot be paired can be self-connected. For the purpose of determining HBR-BEs, if the hole touches the edge of the image, the outside of the image is considered as a region. Therefore, HBRs touching the edge of the image will be self-connected, such as the sky region in Figure 1.

Figure 4 shows boundary extremities, and their connections, for a toy example. If (R_2, R_4) is a pair of matching HBRs, then their HBR-BEs can be linked using straight lines (lines (p_2^4, p_1^2) and (p_2^2, p_1^4)). Similarly, the HBRs that cannot be grouped in pairs (R_1 and R_3) can be self-connected (lines (p_1^1, p_2^1) and (p_1^3, p_2^3)).

One could argue that only creating pairs of regions is too constraining and that tripartite, or even larger, groups should be considered. For example, a cross of which only the four arms are visible, the centre being in the hole, should be dealt with using a group of four regions. However, this could lead to intractable situations that may be difficult, if not impossible, to solve. Instead, we iteratively create connections between pairs of HBRs, which overall provides a tractable way of creating larger groups of HBRs (Section 3.5).

Some connections cannot be kept as they would modify the structure of the image outside the hole. These connections, either pair- or self-connections, are detected by checking for intersections between the corresponding lines and any other region *outside* the hole, other than the connected ones. If such an intersection happens, the connection is discarded. However, a line-crossing that is completely inside the hole is allowed in order to preserve parts of the structure of the pair of regions.

HBRs, particularly natural ones, often tend to have two points which connect

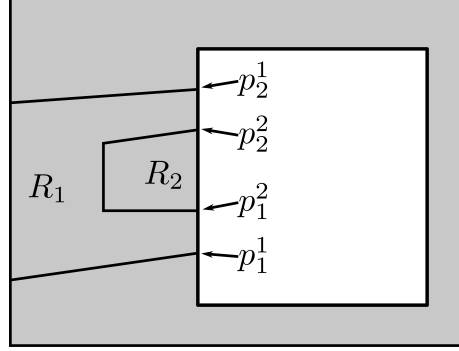


Figure 5: Enclosed HBR and HBR-BE numbering

them to different neighbouring HBRs. However, there can exist cases where regions can spread around the hole and enclose other regions that are also HBRs. In such cases, the HBR-BEs of such regions are the *first* and *last* points of the region, discarding intermediate boundary extremities. This is shown in Figure 5.

At the end of this process, pairs of HBRs that can be linked across the hole using straight lines are stored in a list. Regions belonging to a pair are relabelled with a single label for the pair. This list must now be prioritised, using proximity of the regions, before the hole can be filled in. This list is the Region Connections Priority List (RCPL). Regions that can only be self-connected are kept apart for now.

3.3 HBR spatial proximity

Pairs in the RCPL are sorted in descending order of the distance between their HBRs. Connections at the head of the list therefore correspond to regions that are far apart across the hole while regions at the tail of the list are close to each other. The reason for this sorting is that regions far apart are less likely to be relevant to the content of the hole than regions that are close by.

The distance between two HBRs i and j is expressed as:

$$d_{ij} = ||p_2^j - p_1^i|| + ||p_1^j - p_2^i||, \quad (3)$$

where $||q - p||$ is the Euclidean distance between points p and q . Note that one or two of the terms in (3) can vanish if the two HBRs are adjacent. In practice, this does not happen because adjacent regions that are similar enough to be considered as one should have been segmented as one by the segmentation procedure. This is nevertheless a possibility and just means that the two regions are touching on one or two sides.

Once the list of connections and their order has been created, the structure that matches regions outside the hole can be created in the hole.

3.4 HBR connection priorities and flood-filling

Straight lines are drawn in the structure image following the order of the RCPL, from head (lower priority, larger spatial distance) to tail. Self-connections are drawn after the pair connections. This is because they are more likely to correspond to local events in the structure of the image and must therefore be preserved as much as possible. The lines are drawn following the same order as for the creation of the HBR-BEs.

The lines are drawn with the colour of the corresponding region (self or pair), which creates a new enclosed area for each connection. This area is flood-filled with the same colour, therefore creating a new region in the hole.

As the HBRs are connected, each newly created region can override a previously created region. This however is done from less likely to more likely connections, so that the more likely new regions remain intact in the hole.

3.5 Iterative connections and flood-filling of hole regions

It was mentioned earlier that some regions could be involved in more than one pair of HBRs, based on the histogram matching procedure (Section 3.1). Moreover, new regions are created in Section 3.4 that could potentially be linked to HBRs. Therefore, the regions are connected and flood-filled in an iterative connection and flood-filling process.

The process is similar to the main connections and flood-filling stage described in the previous sections, which constitutes the first iteration of the iterative process. However there are small differences. First, for each new hole region, similar regions are selected from the already saved good match regions list, which is therefore not recomputed. This prevents any drift of the statistics of the colour of the regions due to merging. The new region pair list is checked for line-crossings as done previously. Secondly, pairs of regions that are not touching the same hole are excluded. This is because the initial hole might have been split into multiple holes from the previous iteration (this also allows us to tackle the case of multiple holes that might have been present initially). If the new pair passes the test, it is added to the RCPL at its correct position. This process is continued while there remains regions to be connected.

Figure 6 shows an example of iterative hole region connections and flood-filling. Figure 6(a) shows a hole (in white) that is surrounded by eight regions. We assume that regions R_1 and R_7 as well as regions R_2 , R_4 , R_5 and R_6 are similar according to their colour properties. Figure 6(b) shows the HBR-BEs that will be connected. Figure 6(c) is the result of the first iteration. The regions R_1 and R_7 are relabelled with the same label, connected and flood-filled. The same procedure applies to pairs (R_2, R_4) and (R_5, R_6) . Note the shape of the area created by pairs (R_2, R_4) and (R_5, R_6) ; it is due to the positions of the HBR-BEs. R_8 is self-connected and then flood-filled. Figure 6(d) shows the result of the first iteration with the new regions (R_1 replacing (R_1, R_7) , R_2 replacing (R_2, R_4) and R_5 replacing (R_5, R_6)) and a new hole. Figures 6(e) and 6(f) show the second iteration of connection, flood-filling and result where

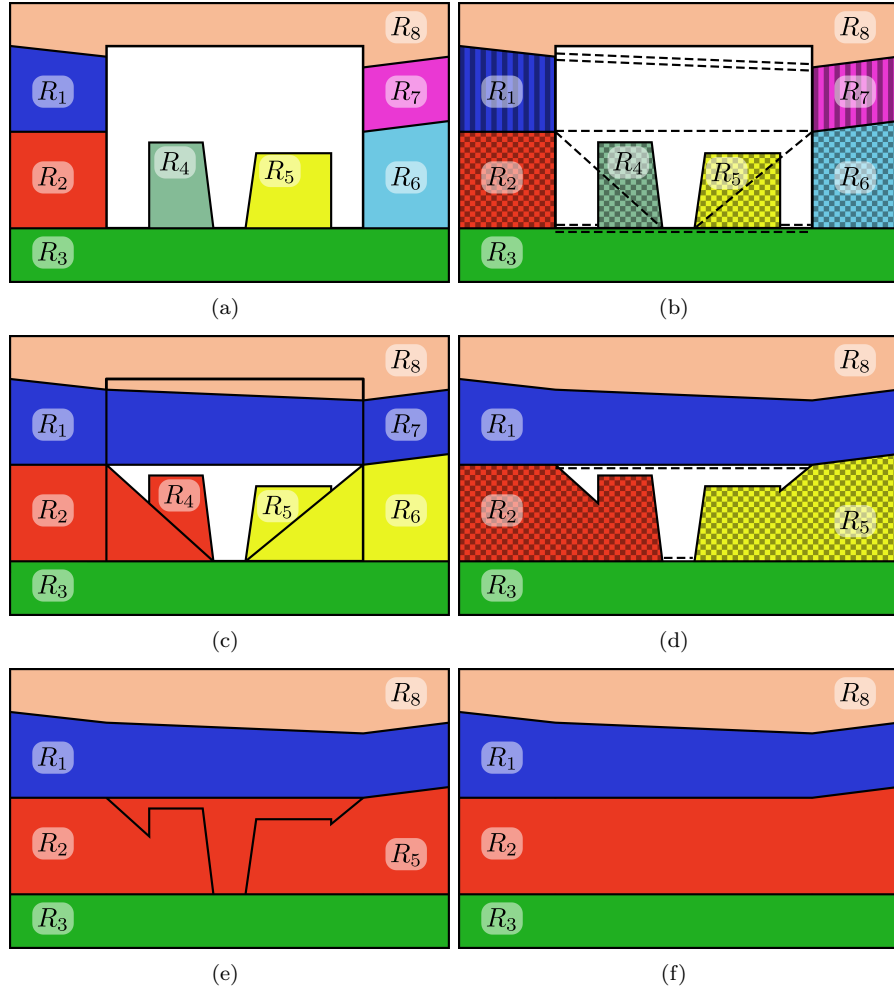


Figure 6: Example of iterative hole regions connection and flood-filling. The hole is in white. Regions with the same pattern match from the point of view of their colour statistics. Dashed lines indicate which HBR-BEs will be connected.

the regions R_2 and R_5 of the pairs (R_2, R_4) and (R_5, R_6) in Figure 6(d) were assumed similar according to the relabelling process, and therefore iteratively connected and flood-filled. Note that during the first iteration, HBRs R_4 and R_5 could have been linked first. This newly created region would then have been linked to either R_2 or R_6 at the second iteration and the remaining similar HBR (R_6 or R_2) during the third iteration, producing the same result.

We cannot prove that the established connections are independent of the order in which the regions are considered. However, the fact that the statistics of the regions and the list of good matches are not recalculated during the process

helps that order independence. There might be structures for which this is not the case, but we have not been able to create such an example. However, we are not after creating the only possible solution, but rather a solution that is possible and plausible. This lack of proof of independence is therefore not an issue.

It is important to point out that the connection process in some cases cannot be fully automated (because this is often not possible without high level knowledge about the scene). There could therefore be problems related to region mismatch in the relabelling process, creating wrong connections, or connections wrongly not created. This is particularly the case of occluding boundaries. As a result, some user interaction may be needed to modify these connections. Simple user interaction can happen at this stage by modifying the list of possible connections. The user can also modify the actual region connections by adding or deleting straight lines in the structure image.

Finally, it is possible that some areas of the hole have been left unfilled. This can happen when connections could not be made and the ones that could did not cover the hole. These areas will be filled in at the structure synthesis stage, in the next section.

3.6 Structure synthesis

In previous sections we proposed a method to fill in the hole with a structure that matches, as best as possible, the topology of the surroundings of the hole. However, the structure was created using straight boundaries between regions. This is unlikely to match the geometry of existing boundaries outside the hole. We therefore need to modify the shape of these boundaries to create a more plausible structure with boundaries that match the shape of, and seamlessly propagate, the boundaries outside the hole into the hole. This is performed using the synthesis method described in Section 2 but applied to label images, instead of colour images.

The synthesis process will first introduce randomness in the straight lines that will match the coarse shapes in the rest of the structure image. Because only good matches are used, the random shapes will remain plausible at all stages and will progressively converge towards shapes that match the fine details of the rest of the structure. The random fill-in will be smoothed out by the process, with unlikely region labels removed due to lack of support for their presence.

It is possible that a boundary created by linking regions across the hole does not have a counterpart outside the hole (boundary between regions that are not touching). In this case, pixels on either side of the boundary will match equally well (or badly) region pixels outside the hole. At the beginning, when the boundaries are straight, the probability of choosing one of the two regions will only depend on the relative sizes of the two regions, larger regions being picked more often. Similarly, when random initial filling-in is used, the selection will depend on the size of the regions around that part of the hole and the distance to the non-hole parts. Overall, this will tend to create boundaries that favour larger and closer regions, which is not unreasonable.

4 Results

We now present and discuss some results on images chosen from [1], selected because they illustrate different aspects of the performance and failures of our method, the latter to highlight the failure modes.

We evaluate our method in a qualitative manner, unlike others who proposed quantitative evaluations (e.g. [29]). There are two reasons for this. One is that it is not clear that the total cost of the reconstruction (how well each synthesised pixel’s neighbourhood matches the image) reflects the visual quality of the result. The second is that, even if we had ground truth, there is no reason that a synthesised image that is different from ground truth should not be visually as acceptable. This is the difference between image completion and inpainting, where one would expect the reconstruction to be close to the ground truth (original, perfect) image.

An example of reconstruction of structure is demonstrated in Figure 1 where the structure of regions inside the hole match the surrounding regions, particularly for the tree area. Compared with the same image, but without structure (only texture synthesis, Figure 1(f)), the structure in the hole was not properly preserved, particularly in the bushes. The tree line is however plausible, albeit somewhat repetitive. It is more plausible than the bush area because the contrast between trees and sky is greater than that between the various shades of green. The repetitiveness comes from the rather small number of tree line samples. This example is topologically simple, especially because most of the trees and bushes are seen as one region at the region connection stage. This implies that only self-connections were created (the sky area from the right of the trees to the top right of the image and the bush-trees area from the right of the trees to the right of the image), leaving a large unfilled area in between. This area was randomly filled in from the sky and bush-trees areas and improved to the given result.

Another example is shown in Figure 2(e) where, without structure, the region of the wall expanded into the grass area resulting in unrealistic completion. However, with structure (Figure 2(d)), the boundaries between the grass and wall regions is realistic. Note that the brickwork is well reconstructed with regular spacing of the bricks as in the original texture. This is partly due to the fact that one brick in the texture is 12 pixels wide, just smaller than the neighbourhood size. This is probably also due to the fact that at the end of the synthesis, the best match is chosen, therefore exactly replicating the pattern, although this would need to be verified.

Figure 7 shows a case where the structure is not properly reconstructed. Because of the change in colour of the edge of the pavement on either side of the hole, the two corresponding areas are not recognised as one and are therefore not connected. This left only self-connections of the regions above and below, the rest being randomly filled-in from the surrounding regions. Despite this, the two regions corresponding to the pavement edge are correctly created and the resulting structure looks plausible. As a result, the synthesised texture also looks plausible with the minor colour changes within each region being correctly

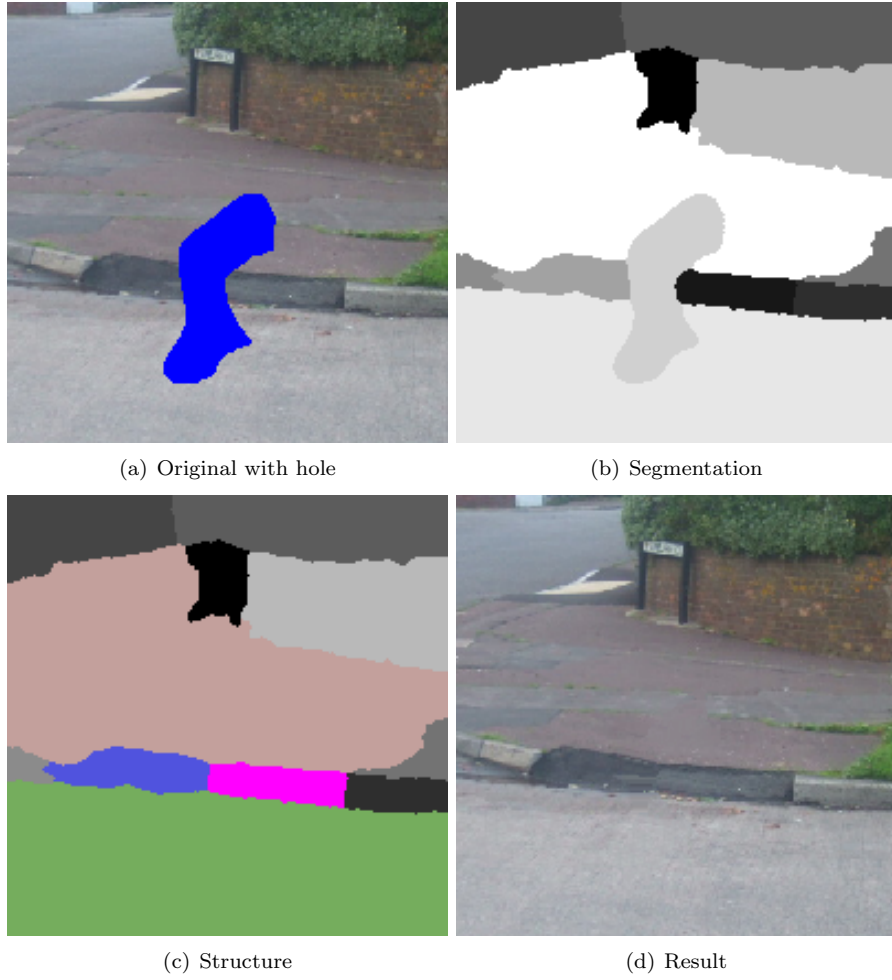


Figure 7: The PAVEMENT image (image: 200×200 , neighbourhood: 15×15)

reproduced.

Figure 8 shows a failure of topology creation. Because we only create boundaries in the hole that link HBR-BEs, the vertical tube boundaries have not been propagated upwards in the hole. If left like that, the wrong structure is created. However, simply drawing two vertical lines to close the tube upwards is enough to fix this problem.

Figure 9 shows an example of a failure due to the segmentation. The regions on either side of the hole were too different to create HBR connections across the hole, as should have been created. The result is due to a combined effect of more samples being taken from the grass areas than the sky areas during random filling-in, resulting in more of the grass areas being kept during synthesis, and the

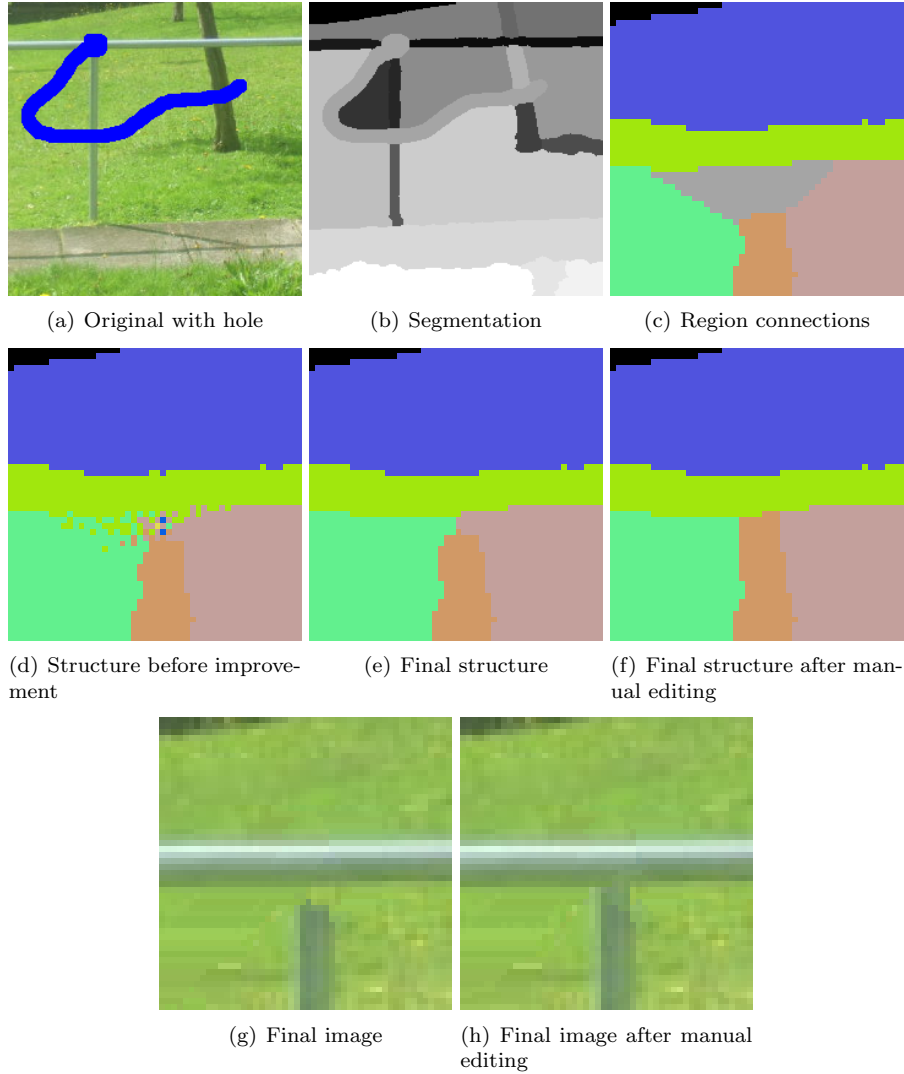


Figure 8: The TUBE image, and close ups of the intersection of the railing (image: 240×240 , neighbourhood: 15×15)

grass-sky boundary just left of the hole going up, therefore providing examples for the boundary going up in the hole. The hard vertical boundary created across the sky in the hole means that the texture did not link up correctly. If the sky and grass areas had been segmented as single areas, or recognised as being similar areas, this would not have happened. A simple manual intervention to force the desired HBR connections would have solved the problem.

In order to compare our method to other methods, we used images from

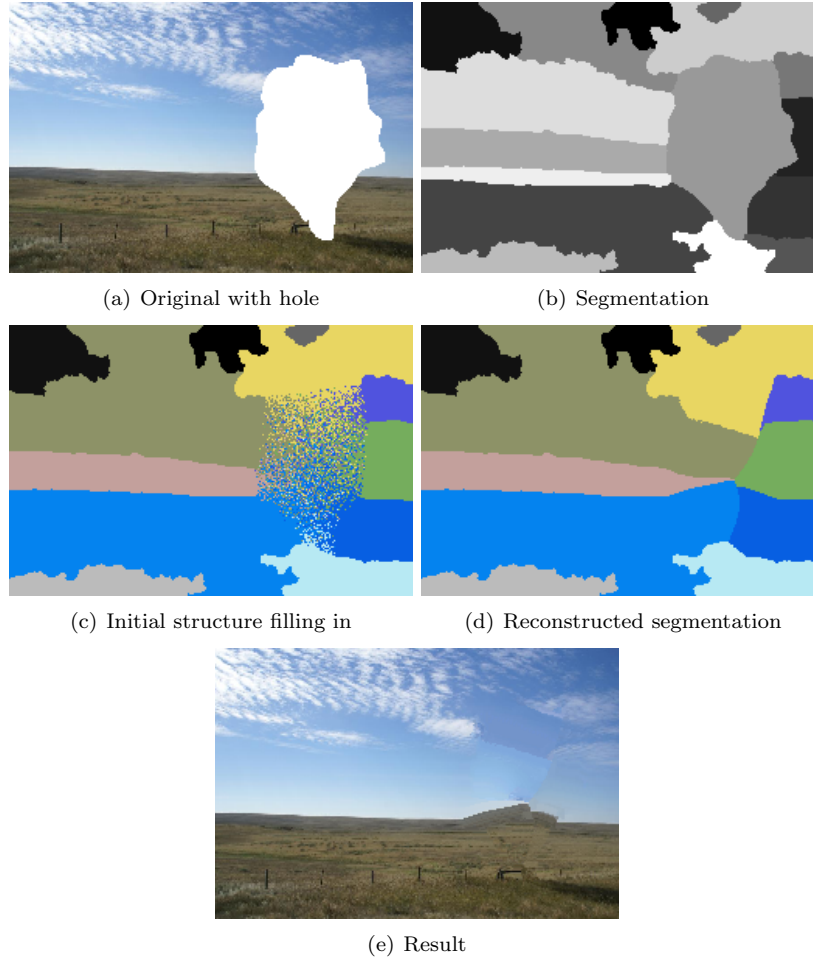


Figure 9: The TREE image (from <http://www.freefoto.com> (reference 15-15-44), image: 260×174 , neighbourhood: 11×11)

previous publications. Figure 10 shows results on two of these images. They show more complicated structures than the previous examples, the ELEPHANT being diagrammatically shown in Figure 6. In the case of the ELEPHANT image, a plausible structure and texture were created. The result is similar to that in [13, 23]. The BUNGEE JUMPER image on the other hand shows a limitation of the structure reconstruction. The two water areas on either side of the hole are not detected as similar because of their difference in colour. However, this was dealt with properly (this is the same situation as for Figure 7). The same does not apply to the two areas below the house on either side of the hole. These two areas were detected by the segmentation as part of the water. Moreover, because the bush and water areas were considered to be the same regions and because

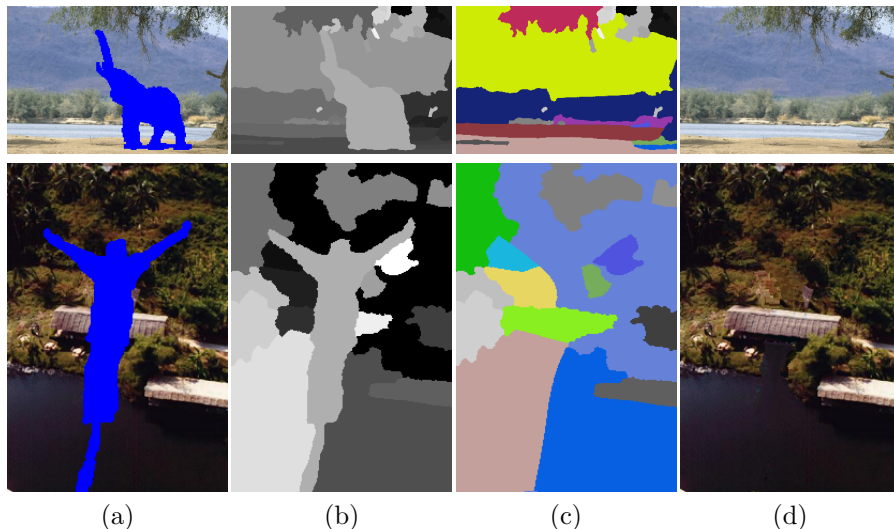


Figure 10: The ELEPHANT (top, [13], Copyright © 2003, Association for Computing Machinery, Inc., image: 384×256 , neighbourhood: 15×15) and BUNGEE JUMPER (bottom, [10], image: 206×308 , neighbourhood: 21×21) images: (a) original with hole, (b) segmentation, (c) structure and (d) result

there is more water than bush in the regions, the created texture corresponds to water rather than bush, even where clearly bushes should have been created. This is a case where the segmentation is not good enough. However, this could have been rectified using simple structure image editing. The roof area is well reconstructed and favourably compares to the results in [10, 41].

Finally, Figure 11 shows three additional results on images obtained from <http://www.freefoto.com>. These show that our method can cope with a variety of structures and textures.

5 Discussion, conclusions and future work

We have presented a method that fills in a hole left by the removal of an undesired area in an image. Our method first creates a structure in the hole that is then used as a constraint to guide the texture synthesis. The structure and texture synthesis is done using an exemplar-based method akin to simulated annealing. The method can create structure that is realistic at all scales and texture that shares the visual qualities of the exemplar, for both stochastic and regular textures. This is due to the annealing provided by our coarse-to-fine synthesis that combines the advantages, without their disadvantages, of spatially variable randomness (to tackle different textures) and random selection of matches (to prevent the replication of large areas). Moreover, this provides an effect that is similar to that provided by multi-scale methods. Indeed, large

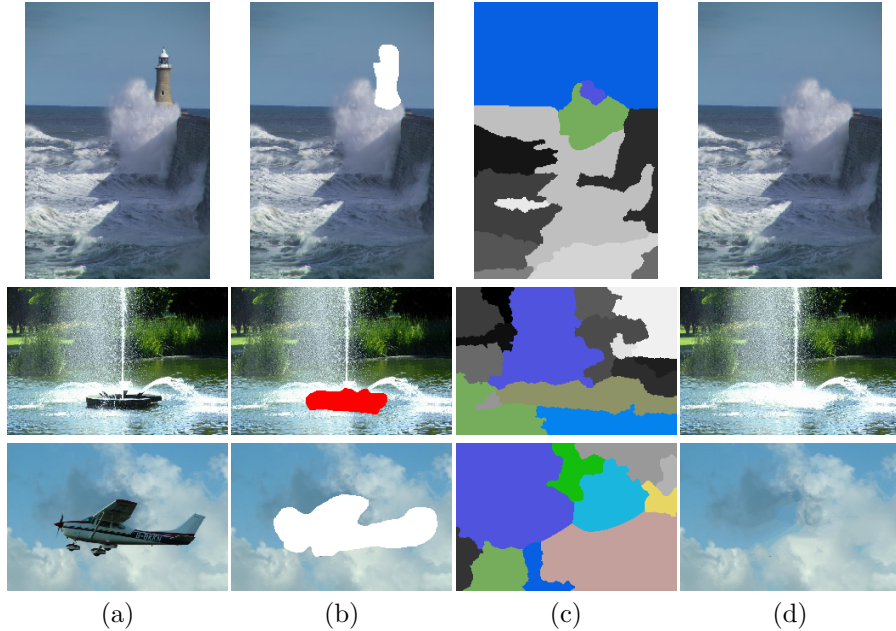


Figure 11: More results on images from <http://www.freefoto.com>. From top to bottom, references 16-09-74, 15-07-5 and 2052-02-27. Images are all 260×173 pixels while neighbourhoods are 11×11 pixels. (a) original, (b) original with hole, (c) synthesised structure, and (d) result.

changes of structure/texture happen at high levels of randomness, which are then refined at lower levels of randomness. From that point of view, this is similar to the effect changing one pixel at coarse scales has on many pixels at finer scales. This contributes to creating boundaries that match the examples and are not only constrained by their extremities, as in [22, 33].

Many publications mathematically express image completion as an optimisation problem and try to find the global optimum in an efficient manner. Our simulated annealing inspired method allows convergence to a minimum that is not necessarily global (from the point of view of the optimised objective function) but rather explores the solution space. We claim that the non-optimal solution is not necessarily worse from a visual, qualitative view point and in fact is possibly beneficial to offer randomness at large scales and detailed texture at smaller scales.

The aim was to automatically, without high-level knowledge about the content of the image, create a completion that is plausible. Figure 8 has shown that not extending unmatched boundaries into the hole can have a detrimental effect. Both [22, 33] do that and maybe this is something that could be incorporated in our method as a special case or by creating ‘ghost’ HBR-BEs. However, it is not clear that the simple straight continuation of such boundaries is always

a good solution. It is possible that occlusion analysis, such as in [18], could be used at the stage of initialising the new boundaries in the hole. However, the boundaries we use and create do not necessarily correspond to object boundaries and vice-versa. Other cues than colour and distance could also be used to match HBRs across the hole, such as angle of their boundaries with that of the hole or continuity of the boundaries of the candidate matches, to possibly improve smoothness of the created boundaries. This remains an open question and manual intervention might well be the best solution as others have done [3].

The quality of the segmentation has an influence on the result. The segmentation may fail by including irrelevant regions or excluding relevant ones, affecting the initialisation of the hole by copying pixels from irrelevant areas. Also, segmenting by colour is not always equivalent to segmenting by texture. If a segment contains several textures, degenerate texture could be synthesised that contains pixels from the multiple textures without their spatial consistency.

Other segmentation methods could have been used. The segmentation needs to produce regions of sufficient size that enough examples are available to copy from. Some methods will create non-connected regions. This implies that the matching of regions across the hole and their relabelling (Sections 3.1 and 3.2) might not be necessary with such methods.

If the hole cuts identifiable objects in two, it is very unlikely that our method (like most methods) will be able to reconstruct the missing part, unless there are other examples of the same object. Similarly, if the image contains repeating structures **and** colours (cityscape seen from the sky for example) and the two opposite sides of distinct elements are separated by the hole, then it is likely that a single element will be created because of the lines created across the hole.

By limiting ourselves to straight lines at the initial stage of the structure creation, it is possible that some regions that could have been linked with curves have been eliminated from the process because the line goes outside the hole. In [22] this situation is dealt with by forcing curves to be in the hole. It is not clear what happens in such a situation in [33]. It is possible that using smooth curves to create the initial topology could extend the range of images that can be dealt with, possibly with Euler’s elastica [31].

Our method implicitly performs a form of belief propagation, similar to the method described in [23], but without the explicit message passing. This is because updating one pixel at one iteration will influence its neighbouring pixels at the following iterations (therefore implementing the implicit message passing). The outside of the hole being static, it corresponds to absolute belief that is propagated inwards as the iterations progress. Although [23] do not first segment the image as we do, the results of message passing priority that they give (e.g. [23, Fig.10]) at the beginning of the process roughly correspond to one of the areas (and specifically its boundary) that our structure reconstruction could have produced. The belief is being propagated first to a specific reconstructed region and its boundary, which has the effect of facilitating the creation of structure. This is done implicitly, while we do this explicitly, which gives us the advantage of allowing user interaction to fix any errors that may arise.

Results show that the proposed synthesis method performs well for label

images, creating structure that can be mostly straight and/or horizontal (e.g. Figure 2) but also geometrically more complex in larger holes (e.g. Figure 1), matching exemplars. It would be interesting to assess how well the synthesis performs for texture compared to recent methods, and in particular assess the importance of the neighbourhood size. Our guess is that the proposed coarse-to-fine approach alleviates the need for large neighbourhoods, but extensive experimentation would be needed to verify this.

Our current implementation is far from being optimised. Its complexity is of the same order as that of [15], assuming the number of iterations is small. More specifically, the complexity of synthesising one structure or texture pixel in one iteration is $O(nN)$, where n is the number of pixels in the neighbourhood and N the number of possible matches outside the hole. The current implementation considers all pixels outside the hole. It would therefore be more efficient to index pixels based on the region they belong to and only consider relevant pixels, rather than going through all pixels and discarding the ones that do not belong to the correct region. Since only the regions surrounding the hole can exist in the hole, a significant reduction of the search space could also be attained by only using these regions as the source for potential matches. Similarly, for the structure image, when neighbourhoods are homogeneous (single label) it is unlikely that they will change. These could be ignored and not synthesised.

Many of the methods to accelerate exemplar-based methods could be used to accelerate our method. In particular, the Approximate Nearest-Neighbour Field method proposed in [3] could be adapted by retaining the multiple solutions explored in our list of possible matches. This would provide an incomplete, yet possibly sufficient, search to replace our exhaustive search. Early termination of distance computation for new patches could also be used as in [3]. Finally, a GPU implementation could also be done given that the algorithm is inherently parallel (e.g. [26]).

Neighbourhood matching could be done using more information than just labels at the structure synthesis stage or colour at the texture synthesis stage. Other authors have used a variety of filters [11, 17, 6] either to further constrain the match, to take into account geometrical information, or to force some effect. Although this adds complexity, it could help further in creating structure and features in the texture.

Acknowledgements

The authors wish to thank A. Criminisi, P. Pérez, K. Toyama, I. Drori, D. Cohen-Or and H. Yeshurun for allowing us to use their images, and the anonymous reviewers for their constructive and useful comments.

References

- [1] Njam Alotaibi. *Image Region Completion by Structure Reconstruction and Texture Synthesis*. PhD thesis, Department of Computer Science, Aberystwyth University, United Kingdom, 2009.
- [2] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the symposium on Interactive 3D graphics*, pages 217–226, 2001.
- [3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. In *Proceedings of SIGGRAPH*, 2009.
- [4] Marcelo Bertalmío, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of SIGGRAPH*, pages 417–424, 2000.
- [5] Marcelo Bertalmío, Luminita A. Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003.
- [6] Nicolas Bonneel, Michiel van de Panne, Sylvain Lefebvre, and George Drettakis. Proxy-guided texture synthesis for rendering natural scenes. In *Proceedings of Vision Modeling and Visualization*, pages 87–95, 2010.
- [7] Raphaël Bornard, Emmanuelle Lecan, Louis Laborelli, and Jean-Hugues Chenot. Missing data correction in still images and image sequences. In *Proceedings of the ACM International Conference on Multimedia*, pages 355–361, 2002.
- [8] Tony Chan and Jianhong Shen. Non-texture inpainting by curvature-driven diffusions (CDD). Technical report, Department of Mathematics, University of California, Los Angeles, September 2000.
- [9] Tony Chan and Jianhong Shen. Mathematical models for local nontexture inpaintings. *Journal of Applied Mathematics (SIAM 02)*, 62(3):1019–1043, 2002.
- [10] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004.
- [11] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH*, pages 361–368, 1997.
- [12] Yining Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):800–810, 2001.

- [13] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. Fragment-based image completion. *ACM Transactions on Graphics*, 22(3):303–312, 2003.
- [14] Alexei Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH*, pages 341–346, 2001.
- [15] Alexei Efros and Thomas Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the International Conference on Computer Vision*, pages 1033–1038, 1999.
- [16] Christian Eisenacher, Chuck Tappan, Brent Burley, Daniel Teece, and Arthur Shek. Example-based texture synthesis on Disney’s tangled. In *Proceedings of ACM SIGGRAPH, Production Talks*, 2010.
- [17] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of SIGGRAPH*, pages 327–340, 2001.
- [18] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering occlusion boundaries from an image. *International Journal of Computer Vision*, 91(3):328–346, 2011.
- [19] Michal Holtzman-Gazit and Irad Yavneh. A scale consistent approach to image completion. *International Journal of Multiscale Computer Engineering*, 6(6):617–628, 2008.
- [20] Zhang Hongying, Peng Qicong, and Wu Yadong. Image completion algorithm based on texture synthesis. *Journal of Systems Engineering and Electronics*, 18(2):385–391, 2007.
- [21] H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proceedings of the International Conference on Image Processing*, page 186, 1997.
- [22] Jiaya Jia and Chi-Keung Tang. Inference of segmented color and texture description by tensor voting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):771–786, 2004.
- [23] Nikos Komodakis and Georgios Tziritas. Image completion using efficient belief propagation via priority scheduling and dynamic pruning. *IEEE Transactions on Image Processing*, 16(11):2649–2661, 2007.
- [24] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.
- [25] Frédéric Labrosse. On the editing of images: selecting, cutting and filling-in. In *Proceedings of the International Conference on Vision, Video, and Graphics*, pages 71–78, 2003.

- [26] Sylvain Lefebvre and Hugues Hoppe. Parallel controllable texture synthesis. *ACM Transactions on Graphics*, 24(3):777–786, 2005.
- [27] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, 2001.
- [28] Jeremy Long and David Mould. Improved image quilting. In *Proceedings of Graphics Interface*, pages 257–264, 2007.
- [29] Alex Mansfield, Mukta Prasad, Carsten Rother, Toby Sharp, Pushmeet Kohli, and Luc Van Gool. Transforming image completion. In *Proceedings of the British Machine Vision Conference*, 2011.
- [30] Simon Masnou. Disocclusion: a variational approach using level lines. *IEEE Transactions on Image Processing*, 11(2):68–76, 2002.
- [31] David Mumford. Elastica and computer vision. In Chandrajit L. Bajaj, editor, *Algebraic Geometry and its Applications*, pages 491–506. Springer New York, 1994.
- [32] Andrew Nealen and Marc Alexa. Fast and high quality overlap repair for patch-based texture synthesis. In *Proceedings of Computer Graphics International*, pages 582–585, 2004.
- [33] Andrei Rares, Marcel J. T. Reinders, and Jan Biemond. Constrained texture restoration. *EURASIP Journal on Applied Signal Processing*, 2005(17):2758–2771, 2005.
- [34] Amir Rosenberger, Daniel Cohen-Or, and Dani Lischinski. Layered shape synthesis: Automatic generation of control maps for non-stationary textures. *ACM Transactions on Graphics*, 28(5):9 pages, 2009.
- [35] Alexander Shekhovtsov, Pushmeet Kohli, and Carsten Rother. Curvature prior for MRF-based segmentation and shape inpainting. *Pattern Recognition*, pages 41–51, 2012.
- [36] Jianbing Shen, Xiaogang Jin, Chuan Zhou, and Charlie C. L. Wang. Gradient based image completion by solving the poisson equation. *Computers & Graphics*, 31(1):119–126, 2007.
- [37] Seunghyup Shin, Tomoyuki Nishita, and Sung Yong Shin. On pixel-based texture synthesis by non-parametric sampling. *Computers & Graphics*, 30(5):767–778, 2006.
- [38] Jian Sun, Lu Yuan, Jiaya Jia, and Heung-Yeung Shum. Image completion with structure propagation. *ACM Transactions on Graphics*, 24(3):861–868, 2005.
- [39] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH*, 2000.

- [40] Li-Yi Wei and Marc Levoy. Order-independent texture synthesis. Technical Report TR-2002-01, Computer Science Department, Stanford University, 2002.
- [41] Yonatan Wexler, Eli Shechtman, and Michal Irani. Space-time completion of video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):463–476, 2007.
- [42] Yunjun Zhang, Jiangjian Xiao, and Mubarak Shah. Region completion in a single image. In *Proceedings of the Eurographics Conference*, volume 23, 2004.