Vision, Video, and Graphics (2003)
P. Hall, P. Willis (Editors)

On the editing of images: selecting, cutting and filling-in

Frédéric Labrosse

Computer Science Department, University of Wales, Aberystwyth, Wales, UK, SY23 3DB
ffl@aber.ac.uk

Abstract

This paper is two-fold. On the one hand, we present a system using qualitative spatial reasoning to help a user choose parts of an image that form interesting objects. The system then allows the user to cut these parts as if unwanted in the original image. On the other hand, we propose a modification of existing texture synthesis methods to create texture in the holes left by the cutting of previously selected areas of the image. This texture synthesis is constrained so that it creates boundaries between regions that are similar to existing ones and fills-in the created regions, yet producing random enough textures so that the new image looks realistic.

The context of this work is post-production special effects in cinema or image manipulation where one often wants to remove parts of an image, e.g. when they correspond to props needed for the filming or objects inadvertently included in the shot.

A note to the reader: Most images are better viewed in colour and are available on the electronic version of this paper or at `http://users.aber.ac.uk/ffl/` .

1. Introduction

Special effects for cinema at the post-production stage very often consist in cut, copy and paste operations. For example, one may want to cut undesirable parts of the individual frames making up a movie. One may also copy parts of a frame, possibly from another movie, and paste them onto another frame. Pasting usually involves re-scaling and/or re-aligning the pixels because the pasted pixels are usually at a different resolution and/or at a different sub-pixel position in the new image. It has thus been suggested that these operations should be performed using a vectorial representation of the images[3]. A system producing such representations has been described by Labrosse and Willis[7]. This system produces, apart from the vectorial description of the image, a data structure containing a segmentation of the image and the parameterisation of the regions that make up the image (the statistical properties used to segment the image).

In this paper, we describe a system that uses such a segmentation to help a user to efficiently perform the selection, copying and cutting of parts of the image. Moreover, we propose a way of filling-in holes left out by the cutting operation. A simple qualitative spatial reasoning engine is used to guide the user in the selection of interesting parts. The filling-in is performed by synthesising texture using the remainder of the image as model.

As an example, consider the scene depicted in Figure 1 representing a nice patch of grass. The photographer did not notice the dustbin in the middle of the grass that spoils the picture. The photographer needs to cut the area of the image that corresponds to the dustbin and to fill-in the "hole" left by the cutting using grass.

The system we propose will help a human operator to solve that kind of problem by determining what "objects are in front of what background." A segmentation of the image, for example as it is performed by Labrosse *et al.*[7] using relaxation labelling, is shown on Figure 2. In this example, it is easy to determine that the dustbin is indeed "an object in front of a background," the grass. That information can be extracted from the segmentation because the region corresponding to the dustbin is completely contained in

Figure 1: A spoilt landscape

Figure 2: A segmentation of the image shown in Figure 1

the region corresponding to the grass. More formally, the part of the image corresponding to the dustbin is only *connected* to the part corresponding to the grass.

Section 2 describes how our system helps a user to select interesting parts of an image. Previous related work is described and discussed in Section 2.2 while our system is described in Sections 2.1 to 2.5. Section 3 explains how holes left by the cutting operation are filled-in. Previous work on texture synthesis is presented in Section 3.1 while our constrained texture synthesis is explained in Section 3.2. Finally, Sections 4 and 5 present and discuss some results and propose future directions for the work.

2. Selecting interesting parts of an image

2.1. Some definitions

Let us first introduce some definitions.

Definition 1 (Regions) A region of an image is either a set of contiguous pixels that share some properties or the outside of the image.

Regions can be extracted by a segmentation algorithm, for example as is done by Labrosse *et al.*[7] Several regions can be characterised by the same properties, but they will be considered as being different regions (as they form several non-contiguous sets of pixels). Regions can be *connected*.

Definition 2 (Connection) Two regions are connected if at least one pixel of one of the regions is adjacent to one pixel of the other region or, if one of the regions is the outside region, at least one pixel of the region that is not the outside region is on the border of the image.

The connections, represented using a graph, are extracted from the segmentation image. We define a *part* of an image as follows.

Definition 3 (Parts) A part of an image is a group of one or more regions creating a contiguous area of the image. A part is said to be connected to a region or a part if and only if at least one of the regions making up the part is connected to the region or part.

We also need to define what we mean by an *object*, a *background* and their relationship.

Definition 4 (Objects and background) An object is any area of the image that, if removed, leaves a *single contiguous* hole in its background.

Objects and backgrounds are parts.

2.2. Previous work on spatial reasoning

Systems allowing the selection of parts of images have been described in the past. A notable example is the one by Mortensen and Barrett[8]. It allows a user to manually specify a rough contour of the interesting image part. Its boundary is then extracted allowing the cutting or copying of the object. However, such manual selection can be tedious, especially when objects are small, have many holes, or have a complex shape. The proposed system does not need the user to specify particular areas of the image but will provide the user with possible choices that are globally optimum according to criteria chosen by the user.

Qualitative spatial (topological) reasoning is an area that has been studied in the past. In particular, the Region-Connection Calculus (RCC) theory[5] uses the C relationship to express connections between regions[9]. However, our situation is less general than the one targeted by the RCC theory. Most possible configurations of regions presented by Randell *et al.*[9] do not apply in our case. Indeed, regions are the result of a segmentation which associates with each pixel of the image the region label that best describes it[7]. As such, pixels can only belong to one region and thus regions cannot be part of others. The two remaining configurations from the RCC theory that thus interest us are *not connected* and *connected*. These respectively correspond to $\mathsf{DC}(a, b)$ and $\mathsf{EC}(a, b)$ in the RCC theory, where a and b are two regions.

In the following sections, we propose axioms that use the connection relationship obtained from a segmentation of the image to construct interesting objects in the image.

2.3. So, what can connections tell us?

Paraphrasing Gotts, Gooday and Cohn[5], let us first examine what information can be extracted using only the connection relationship.

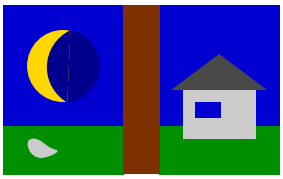


Figure 3: A possible landscape

Figure 3 shows a possible landscape. The picture should be seen as the segmentation of an image, each region being represented using its average colour. This figure will help us identify the different types of object one may want to select.

2.3.1. Simple background

The case described in Figures 1 and 2 is simplified in Figure 3. The grey area at the bottom-left of the picture could be seen as a stone on the grass. One may want to select the "stone" region to either copy it (and paste it somewhere else, possibly in another image) or to cut it as an undesirable element of the image, like the dustbin in Figure 1. It can clearly be seen that the "stone" region is completely surrounded by the "grass" region.

Another interpretation of the same phenomenon can be made. The (blue) rectangle in the house of Figure 3 can be interpreted as a window (actually two windows facing each other in the house!) through which the background (the sky) can be seen. Again, the "window" region is completely surrounded by the "wall" region. In this interpretation, however, the "window" region is a hole in the surrounding region. In that case, one may more probably just want to cut the region possibly to remove the window.

It can thus be seen that whatever the interpretation made and the operation one wants to perform, it can always be said that the central region corresponds to an object and the surrounding region corresponds to its background.

In terms of topological relationships, what characterises this case is that the central region is connected to only one other region, the background region. Remembering that the outside of the image is considered as a region, a first axiom characterising an object is the following.

Axiom 1 A region corresponds to an object having a single region background if and only if the object region is only connected to the background region.

The next case is exemplified by the two regions making the moon on the top-left of Figure 3. The two corresponding regions are completely surrounded by a "sky" region. As in the previous case, the two regions can be interpreted as an object in front of a background made of a single region. Examples where the two regions can be interpreted as a hole can also

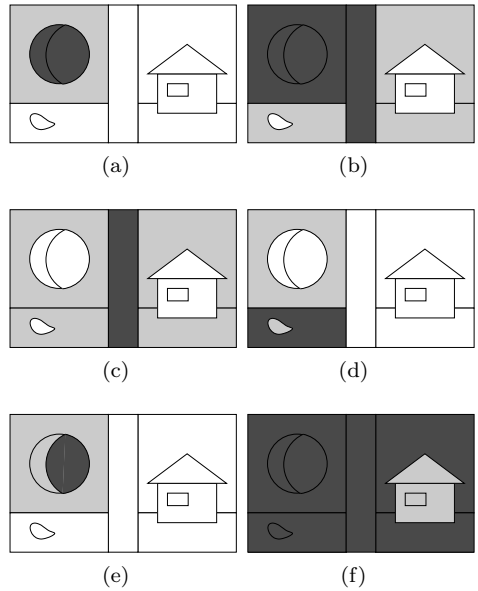


Figure 4: Some interpretations of Figure 3 given by Axiom 3. Dark grey regions make up the object part while light grey regions make up the background part.

be constructed. This can readily be extended to more than two connected regions, *i.e.* a part. Axiom 1 can be replaced by Axiom 2.

Axiom 2 A part corresponds to an object in front of a background made of a single region if and only if the part corresponding to the object is only connected to the background region.

2.3.2. Complex background

The background can also be made of several regions. This is for example the case of the (brown) vertical region at the centre of Figure 3 which can be interpreted as a tree trunk, its background being the part made of two (blue) regions corresponding to the sky, two (green) regions corresponding to grass and the outside region.

Obviously, the case of an object made of multiple regions over a background also made of multiple regions is also possible. The house in Figure 3 is an example of this case. This leads to a more general axiom.

Axiom 3 A part corresponds to an object over a background made of a part if and only if the object part is only connected to a single part, the background part.

2.3.3. Simple image interpretation

The interpretations we made of Figure 3 were all driven by our understanding of the picture but other interpretations could have been made, all satisfying Axiom 3, some being shown on Figure 4.

It is clear that some of these interpretations make

no sense at all (Figure 4(b)), that some do make sense (Figures 4(a) and 4(c)), while some can possibly make sense, depending on the kind of special effect one wants to perform. For example, Figure 4(d) shows an interpretation where the selected region corresponds to grass that needs to be analysed as a texture in view of re-synthesising grass in a new image[4, 11]. Figure 4(e) shows an interpretation that could be useful to remove the shadow of the Earth on the Moon. Figure 4(f) would be useful to copy the whole image but the house.

Axiom 3 is recursive since adding a region to a part creates a new part. Applying Axiom 3 on the image will thus produce a directed graph of possible interpretations, which we call the *groupings graph*. Actually, Axiom 3 produces all possible parts made from connected regions! This leads to a large graph, the exact size of which cannot be predicted because it depends not only on the number of regions but also on their topological relationships. An arc of the graph leads from an interpretation to another by adding a region to a part making up an object and updating the background part. The "leaf" of the graph is the interpretation in which the union of all the regions is an object while the "roots" represent the interpretations where objects are made of one region. The graph thus creates a hierarchy of interpretations. Paths from "roots" to the "leaf" can be followed. This will be used to rapidly select objects in images (see Section 2.5).

2.4. Using more information

Additional information exists in the image that could be used to reduce the number of interpretations and/or produce an ordering of the interpretations.

Relationships between regions constituting the background of the parts defined by Axiom 3 allow humans to determine that some interpretations do not make sense while some do. We present some now.

Some criteria can be used to filter interpretations according to their plausibility. For example, objects can either be made of a single region (*simple objects*) or have a single region as their background (*simple background*). *Centre objects* are objects that do not touch the border of the image while *border objects* are objects that do touch the border of the image. We define a *hole in an object* as any region belonging to the part having the same properties (given by the segmentation) as any region not belonging to the part but connected to it. In Figure 3, the (blue) rectangle in the house is such a hole. This criterion allows the user to select the house without the region corresponding to its background.

These criteria allow us to filter out parts that do or do not satisfy them, reducing the number of parts the user can choose from.

We can also sort the interpretations by quantifying then according to (a combination of) criteria. For example, the more consistent the part surrounding another part is, the more likely these will respectively make a background and an object. Indeed, objects are often made of very different textures and colours while backgrounds are usually made of homogeneous, but possibly highly textured, areas. Moreover, the background is generally wider than the object and thus will be present all around the image part making up the object. The number of regions making up the part as well as the area (in pixels) of the region added to a part can also be sorting criteria.

2.5. Selecting an object

Having a mechanism to generate parts, filter them and sort them, we need an application using it to allow the selection of objects by a user. The main mechanism used in the application we developed uses paths in the groupings graph (Section 2.3.3) to start from simple objects and add regions to it to finally arrive at the desired object. This process is done interactively. Each part is represented by an icon on a canvas. The icon is a small version of the image where all pixels not belonging to the part are made transparent, displaying a configurable background instead. The groupings graph is displayed on the canvas by showing all the parts. Parts having the same number of regions are all displayed on the same row, starting with the "root" parts at the top of the canvas. The user can select a part by clicking on its icon around which a red rectangle is then drawn. Arcs linking that part to immediate predecessors and successors on all the paths going through it are displayed using the average colour of the removed or added region as their colour. A full size version of the icon of the selected part is also displayed. A final aid to the navigation in the graph is provided: the previously selected part is identified by a blue rectangle drawn around its icon. This helps the user to "undo" a selection.

Because the groupings graph can be very large, generating it in its entirety can take a long time and use a lot of memory for all the icons. Moreover, our experiments showed that only a small proportion of the graph is need to arrive at the interesting object. The application thus starts by displaying the "root" parts and new parts are generated on demand: when the user clicks on a part, all parts that can be created from it by adding a region are created (if they have not been before). Figure 5 shows a snapshot of the application. Only some of the parts created to reach the currently selected one are displayed. The user needs only to click as many times as there are regions in the desired part, a number that is generally low, depending on the segmentation.

Filters (Section 2.4) are used to grey-out the parts that are not kept (but the user can still select them). It is not yet clear how the sorting can be used. Icons

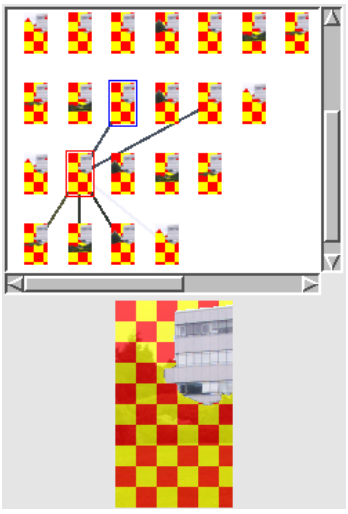


Figure 5: A snapshot of the application showing a portion of the groupings graph and a selected part.

could be sorted on their row, but because regions are added to the connections graph by scanning the image from one corner to the opposite, the icons of the parts are naturally presented in the same spatial order on the canvas, which helps a lot the selection process. More experiments need to be performed to assess the effect of filtering.

3. Cutting the selection

Once a selection has been made, the user may either want to copy it (typically to paste it somewhere else) or to cut it. Copying the selection uses the same process as done by Froumentin *et al.*[3]: a vectorial description of the selected part is saved on disk for later re-use (for example to paste it in another image).

Cutting a part creates a hole in the image that needs to be filled-in. We address here the problem where the background of the part is a stochastic texture and propose a way of "propagating" existing textures into the hole. The method is based on the work described by Efros and Leung[2], but could extend other methods of synthesising texture[10, 4, 11] as well.

3.1. Texture synthesis

Methods to synthesise texture create a model of it and use the model to synthesise a new texture having the same properties. In some cases, the model does not contain the original texture but only its sufficient statistics[4, 11] and in other cases it contains an example of the texture to be synthesised[1, 6, 2, 10]. However, all the methods assume that the given example contains one texture only and that this texture is spatially homogeneous.

For example, Efros *et al.*[2] use an example of the texture to create more of it. The method assumes that a part of the texture has already been synthesised (a small area of the original texture is copied to initiate the process). Finding the value for the next pixel is done by looking for a window in the example that is similar to the neighbourhood of the pixel. Some of the best matches are kept according to a fixed threshold. One is chosen at random and its centre pixel is used as the new pixel. Similarity is measured using a simple pixel-wise normalised Euclidean distance. Our experiments with that method showed that the randomness of the produced texture is dependent on the threshold and that smooth textures need a lower threshold than rough textures. Moreover, if the threshold is too low, entire patches of the original texture are replicated.

In our case, the background of the object is often made of very different textures. Moreover, we found that directly applying the method to our case was leading to an un-natural filling-in of the hole with the smoothest texture because of the unique threshold, also creating un-realistic boundaries between regions as well as an un-plausible topology (see Figure 10). We thus developed a *constrained texture synthesis*.

Note that Igehy and Pereira[6] propose a method to perform such a replacement. However, their method uses a mask to gradually merge texture that has been synthesised *off-situ* with the existing texture.

3.2. Constrained texture synthesis

We solve the problem of the different textures in neighbouring regions with two modifications of the method proposed by Efros *et al.*[2] We create boundaries between regions that can then be filled-in as homogeneous regions. We also do not use a threshold but instead select at random in the sorted (according to the similarity measure) list of matches using a Gaussian random generator.

3.2.1. Creating boundaries

When the part is cut from the image (pixels made transparent), corresponding pixels are also cut from the segmentation image. This produces boundaries between regions that end on the boundary of the hole. Figure 6 shows an image and its segmentation. We want to remove the building, which corresponds to the part selected in Figure 5. Figure 7 clearly shows such boundaries: between the sky and the trees, between the trees and the bushes and between two different kinds of bushes.

To solve the problem of un-realistic boundaries created by the method of Efros *et al.*[2] (Figure 10), we first propagate these boundaries by applying the same texture synthesis method but constraining it to pixels at the boundary of the two regions and only using existing neighbourhoods at the boundary of the two regions. The constraint is done by making sure that

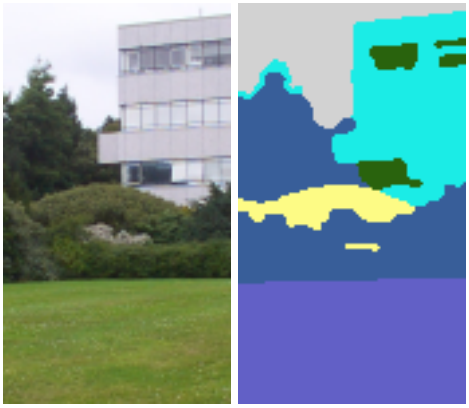


Figure 6: An image and its segmentation

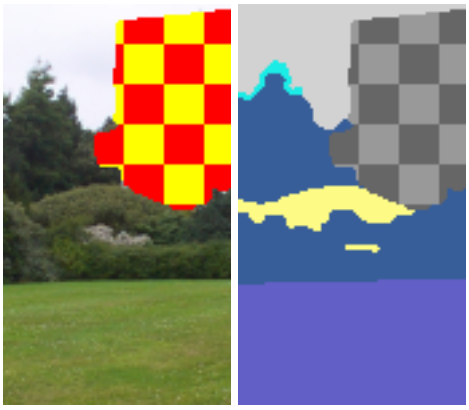


Figure 7: The image on Figure 6 with a hole in place of the building

the neighbourhood around the pixel to create and the neighbourhood looked for along the existing boundary are compatible, *i.e.* have the same pixels in the segmentation image at the same place. The constraint is progressively relaxed should no match be found. This propagates realistic boundaries into the hole because the new boundary pixels are taken from existing boundary examples at places that are similar in terms of the colours in the neighbourhood of the pixels.

Once the boundaries have been created, we are left with closed homogeneous areas. These areas can now be filled by using the similar texture samples that are present in the image. This is done by verifying that the possible matches indeed belong to the same region, which is determined using the segmentation image.

3.2.2. Selecting a match

Our experiments with the original method[2] showed that using a threshold to limit the number of possible matches was not adequate. Indeed, using a threshold relative to the minimum distance (all matches for which the distance d is in $[d_{min}, (1+\epsilon)d_{min}]$ can be chosen[2]) leads to having to choose between the same

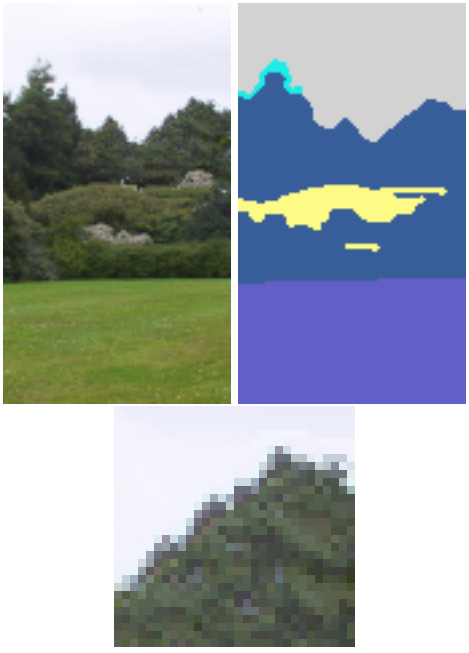


Figure 8: The hole on Figure 7 filled-in

number of matches whether the texture is smooth or not, the number depending on the threshold. However, smooth textures require a low threshold while rough textures require a high threshold. Should too low a threshold be chosen, we only replicate parts of the sample texture. On the contrary, if the threshold is too high, the produced boundaries and/or textures are too irregular. The optimum threshold value not only depends on the textures but also on how many samples we have of it, number typically very low for the boundaries.

Instead, we keep all the matches, sort them from best to worst, and use a Gaussian random generator to select a match. The distribution of the generator is centred on 0 (corresponding to a very low threshold in the original method[2]) with a standard deviation that can be specified to provide more or less randomness (we used a value of 1 in the results we present). This ensures the selection of the best match most of the times, yet keeps some randomness. Possible matches are only considered among pre-existing pixels, never using newly generated ones. This ensures the propagation of samples from the original texture only.

4. Results

Figure 8 shows the hole of Figure 7 filled-in with our method as well as a close-up of the generated trees area. The image in Figure 1 with the dustbin and building removed can be seen in Figure 9. Both boundaries and inside of regions look realistic. To compare, Figure 10 shows the result produced by the method

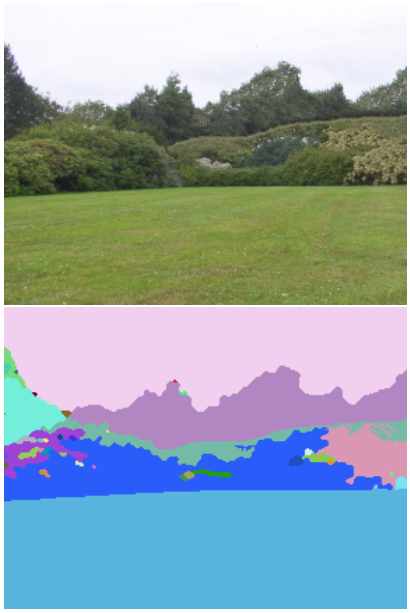


Figure 9: The landscape is now nicer!

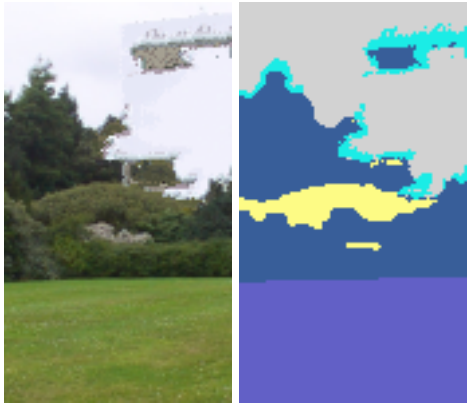


Figure 10: A hole filled-in with the method of Efros *et al.*[2] and the corresponding regions

of Efros *et al.*[2] This clearly shows that boundaries do indeed need to be propagated first!

Figure 11 shows that sometimes filled-in pixels do not match their immediate neighbourhood. This is the case of the sky pixels replacing the wing because pixels "above" the wing are darker than any visible sky pixels around the transition between clouds and sky (see Section 5).

5. Discussion, future work and conclusion

The propagation of existing boundaries into the holes sometimes fails to preserve the original topology of the regions, either because of the process of propagating the boundary, or because it cannot be preserved because of the original topology of the image. The way

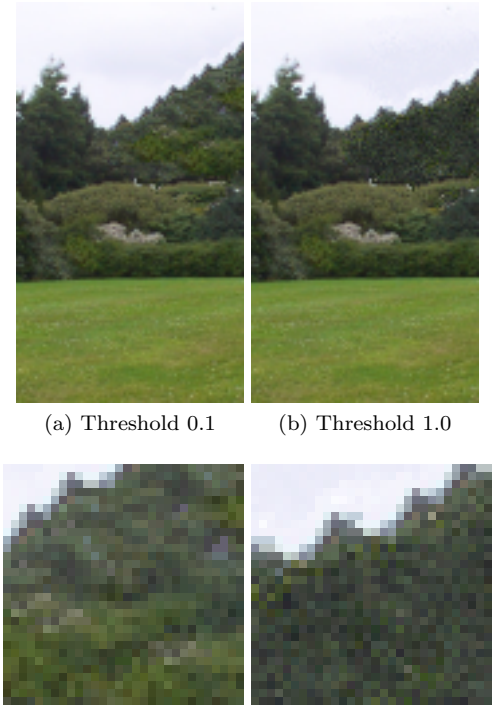
Figure 11: Nuages

we select the next point in the hole to be part of the boundary tends to push the boundary away from regions that do not belong to either of the regions delimited by the boundary. This is done by not using points for which the neighbourhood contains such a region. This works well when the boundary is tangential to the region. However, the constraint sometimes needs to be relaxed so that the boundary can continue, failing to push the boundary away from the region. This results in having to create a boundary for which we have no example because it was not present in the original image. When this happens, we allow matches which have neighbouring pixels in either region without constraining their spacial configuration. This tends to create a boundary that will stay close to one of the two regions, thus limiting the "problem" to as small an area as possible.

In some cases (*e.g.* Figure 11 around the wing), the propagation of boundaries locally creates incompatibilities between the colour of the pixels on the boundary and previously existing pixels. This happens when the texture of one of the regions delimited by the boundary is not spatially homogeneous. This is a limitation of the method.

Our way of selecting a match (Section 3.2.2) works and does not tend to copy entire patches as the original method does. Figure 12 shows Figure 7 filled-in using the selection process as proposed in the original method[2] where the boundary between trees and sky is either very regular and the trees correct or the boundary less regular but the trees very random (shown by the close-up views). Moreover, a close look at the images show that large parts of the sample textures are copied verbatim in the hole either on the boundary or the inside of the regions.

There are some decisive factors that influence the quality of the synthesised texture. An obvious one is the size of the neighbourhood used to find matches. The size specifies the largest texture element that can be copied/synthesised. In our experiments, we used a size of 5×5 pixels which proved sufficient for all tree/grass textures.



(a) Threshold 0.1 (b) Threshold 1.0

(c) Close-up of (a) (d) Close-up of (b)

Figure 12: A constrained fill-in with the selection from Efros *et al.*[2] with two different thresholds

The segmentation is also important. We used a segmentation that only considers the colour average and variance, which is not a sufficient statistics to segment texture. In particular, the trees/bushes areas of Figure 1 are not very well segmented. Moreover, because of natural blur present in images, the segmentation fails to cleanly separate pixels from two neighbouring regions (Labrosse *et al.*[7]). To make sure that no pixel from a removed region remains in the image (which would tend to propagate a "wrong" texture), we do a small number (usually one or two are enough) of dilation steps of the hole. A more appropriate segmentation should use a sufficient statistics to represent the textures, as is done by Gimel'farb[4] or Zalesny and Van Gool[11]. We are currently working on finding a similar method to represent and synthesise the boundaries.

Some problems still remain. For example, a less short-sighted method of constructing boundaries needs to be developed. Experiments with users also need to be done to improve further the selection process. Finally, the filling-in of the homogeneous regions is slow because the neighbourhood of every new pixel needs to be compared to all similar pixel configurations in the image. However, this algorithm is inherently parallel, thus providing an obvious solution to this problem. However, the results we have presented show that the method is promising.

References

1. J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proc. of SIGGRAPH 97*, *Comp. Graph.* Proc., Annual Conf. Series, pages 361–368, Los Angeles, CA, USA, 1997.
2. A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. of the Int. Conf. on Comp. Vision*, Corfu, Greece, 1999.
3. M. Froumentin, F. Labrosse, and P. Willis. Vector-based representation for image warping. *Comp. Graphics Forum*, 19(3):C419–C425 and C543, 2000.
4. G. L. Gimel'farb. Texture modeling by multiple pairwise pixel interactions. *IEEE Trans. on Pat. Anal. and Machine Intel.*, 18(11):1110–1114, 1996.
5. N. Gotts, J. Gooday, and A. G. Cohn. A connection based approach to commonsense topological description and reasoning. *The Monist: An Int. J. of General Philosophical Inquiry*, 79(1):51–75, 1996.
6. H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proc. of the Int. Conf. on Image Processing*, Santa Barbara, CA, USA, 1997.
7. F. Labrosse and P. Willis. Towards continuous image representations. In *Proc. Int. Conf. in Central Europe on Comp. Graphics, Visualization and Comp. Vision (WSCG)*, volume 1, pages 206–213, Plzen, Czech Republic, 2001.
8. E. N. Mortensen and W. A. Barrett. Intelligent scissors for image composition. In *Proc. of SIGGRAPH 95*, *Comp. Graph.* Proc., Annual Conf. Series, pages 191–198, Los Angeles, CA, USA, 1995.
9. D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. of the Int. Conf. on Knowledge Rep. and Reasoning*, pages 165–176, 1992.
10. L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. of SIGGRAPH 00*, *Comp. Graph.* Proc., Annual Conf. Series, New Orleans, LO, USA, 2000.
11. A. Zalesny and L. Van Gool. A compact model for viewpoint dependent texture synthesis. In *SMILE 2000*, volume 2018 of *Lecture Notes in Computer Science*, pages 124–143, 2001.