A NON-HIERARCHICAL PROCEDURE FOR RE-SYNTHESIS OF COMPLEX TEXTURES

Paul Harrison

School of Computer Science and Software Engineering
Monash University
Wellington Rd.
Clayton, 3800
Melbourne, Australia
pfh@yoyo.cc.monash.edu.au

ABSTRACT

A procedure is described for synthesizing an image with the same texture as a given input image. To achieve this, the output image is built up by successively adding pixels selected from the input image. Pixels are chosen by searching the input image for patches that closely match pixels already present in the output image. It is shown that the accurate reproduction of features in the input texture depends on the order in which pixels are added to the output image. A procedure for selecting an ordering which transfers large complex features of the input to the output image is described. This procedure is capable of reproducing large features even if only the interactions of nearby pixels are considered. The procedure can be altered to allow specification of the placement of particular features in the output texture. Several applications of this are described.

Keywords: texture synthesis, image manipulation

1 INTRODUCTION

The texture of an image might be defined broadly as the interrelationships between pixels in that image. The ability to analyse and manipulate image texture has a number of interesting applications. The simplest application is to create a new image with the same texture but of different size and shape to a sample image. Seamless editing of images is also a possibility. For example an object could be removed from an image by synthesizing a new section of the background texture over the top of it. These applications all rely on the ability to re-synthesize a sample texture to fit a variety of constraints.

A number of texture re-synthesis methods are described in the literature. One approach is based on searching for specific features in textures [Heege95] [Bonet97] [Porti99]. In these methods, the input image is decomposed into a set of features. Statistics about these features are collected, and used to synthesize a new image. One problem with these methods is that they can only recognize a set of features which have been specified in advance. While the results can be impressive, it is difficult to devise a generic feature set that can be used to describe all textures.

Another approach to the texture re-synthesis problem is to analyse and reproduce interactions between individual pixels [Monne81] [Gagal90]. Methods based on this approach define a function describing a pixel in terms of its neighbours and a random element. This function is used to generate a new image, pixel by pixel.

Recently, a new pixel-based technique has been developed based on best-fit searching. Garber [Garbe81] and Efros and Leung [Efros99] independently developed this approach, in which each pixel is selected by searching the input image for a patch of pixels closely matching nearby pixels already present in the output image. This results in an output image pieced together from small parts of the input image. Using this technique, textures can be synthesized in which the relationships between neighbouring pixels are extremely

Ω_I :	The set of pixel locations in the input image.
$I(s),\ s \in \Omega_I$:	The pixel in the input image at location s.
Ω_O :	The set of locations containing pixels in the output image.
$O(s),\ s \in \Omega_O$:	The pixel in the output image at location s.
$L(s),\ s \in \Omega_O$:	The location of the pixel in the input image that is in the output image at location s.
Ω_K :	The set of offsets considered when calculating the similarity of two texture patches.
$K(u),\ u \in \Omega_K$:	Weighting given to a particular offset u.
$A(s),\ s \in \Omega_I$ $B(s),\ s \in \Omega_O$:	Two uniformly random functions.

Table 1: Table of symbols.

complex.

Efros and Leung report that for good results the size of the patch searched for in the input image should correspond to the size of the largest feature in the texture. To allow larger features, Wei and Levoy [Wei00] propose a hybrid of the feature-based and best-fit techniques that generates the output in a pyramid of successively finer resolutions.

This paper presents a refinement of the pixel-based best-fit re-synthesis procedures introduced by Garber [Garbe81] and Efros and Leung [Efros99]. The most important change is to the selection of the order in which pixels are added to the output image. This order is selected to allow reproduction of features larger than the size of the patches searched for in the input image. The procedure avoids decomposing the input image into a predefined feature set, and can therefore re-synthesize a wide range of textures. An extension to this procedure is presented that allows specification of the layout of different regions of texture in the output image.

2 RE-SYNTHESIS PROCEDURE

The procedure described in this paper takes as input an image containing a sample of a texture and produces another image with the same texture.

The procedure has two stages. In the first stage, pixel interrelationships in the input image are analysed. The extent to which the value of each pixel constrains the values that can be taken by neighbouring pixels is determined. The reasons for doing this are discussed in section 2.1 and the calculations necessary are described in section 2.2.

In the second stage, pixels are added to the initially blank output image until all locations have been filled. The order in which pixels are added is chosen to facilitate faithful reproduction of the texture of the input image, using the results from the first stage, as described in sections 2.1 and 2.2.1. Section 2.2.1 also describes the initialization of this stage in terms of choice of the locations and values of seed pixels.

To add a pixel to the output image at a particular location, the surrounding pixels which have already been added to the output image are examined. The closest match to these pixels is located in the input image, and the corresponding pixel from the input image is inserted in the output. This is described below and in section 2.2.2.

Symbols used in this paper are listed in Table 1.

Colours are stored as RGB values. To compare individual pixels, the sum of the absolute values of the differences in each colour component was used:

$$d((r_1, g_1, b_1), (r_2, g_2, b_2)) = \\ |r_2 - r_1| + |g_2 - g_1| + |b_2 - b_1| \qquad (1)$$

To measure how closely patches from the input image match a patch in the output image, a distance function can be used. The distance function used in this paper is a weighted Manhattan (city block) distance function. To allow synthesis of textures with a random element, the distance function contains a small random component with weight specified by the parameter ε:

$$D(s,t) = \varepsilon\,|A(s) - B(t)| \\ + \textstyle\sum_{u \in \Omega_K, t+u \in \Omega_O} K(u) d(I(s+u), O(t+u)) \qquad (2)$$

The Manhattan distance was chosen over the more commonly used Euclidean distance as it is more forgiving of outliers. I.e. a good match in most pixels in the patch will not be negated by a poor match in one or two pixels.

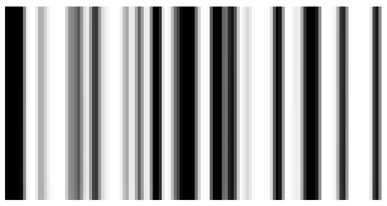


Figure 1: Barcode texture.

For each location t in the output image, a pixel $I(s)$ from the input image is chosen. The location s in the input image is selected to minimize the distance $D(s,t)$ as given in Eq. 2.

2.1 Ordering of pixel insertion

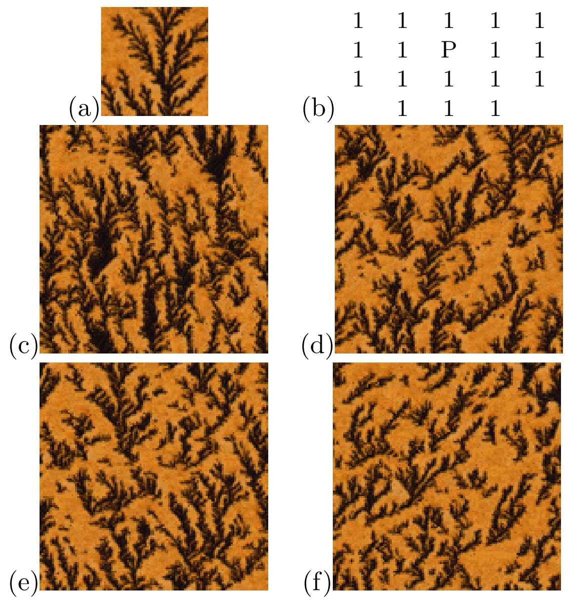
This section discusses the order in which pixels are added to the output image in the second stage of the procedure. As will be demonstrated, certain features such as branching patterns are only reproduced correctly in the output image if the placement of pixels proceeds in a particular order.

Consider the values a particular pixel in a textural image could take. The pixel value must be consistent with the values of all other pixels in the image. These other pixels may be said to constrain the values the selected pixel can take.

An important property of these constraints is that the constraint imposed by a distant pixel in some direction may also be provided by a closer pixel in the same direction. In general, most of the constraint will be imposed by nearby pixels. For example, in Fig. 1 the constraint imposed by pixels far above or below a given pixel will also be provided by pixels a small distance above or below that pixel. This kind of relationship may be directional. In Fig. 1 it applies vertically but not horizontally.

These constraints can be thought of in terms of information theory [Blahu87]. Without any prior knowledge, a pixel is equally likely to take any value, and this requires a certain number of bits to encode. If its neighbours impose constraints on the values it can take, it will require (on average) less bits to encode. The average number of bits required to encode some information is referred to as its *entropy* [Blahu87, pp. 55]. Entropy can therefore be used to measure the constraints imposed on a pixel.

The pixel selection procedure is based on a search for patches of pixels in the input image that match those already placed in the output. These patches must cover enough pixels to capture all



$$\begin{array}{ccccc} & 1 & 1 & 1 & \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & P & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & \end{array}$$

Figure 2: Textures generated using different orders of pixel insertion.
Input image (a), weighting of the distance function $K(u)$ used about the center pixel P (b), and the output from raster scans proceeding from top to bottom (c), bottom to top (d), left to right (e), and right to left (f).

the constraints on each location, or the texture will not be reproduced correctly. However, if the patch size is too large, there will be less chance of finding good matches in the input image. It is therefore desirable to choose an order to add pixels to the output image in which the constraints imposed on each location are captured by a small neighbourhood of pixels.

One order that can be used to add pixels is a raster scan. Fig. 2 shows images generated using raster scans in different directions. The lichen texture being synthesized has a complex branching pattern. As can be seen, the scan that proceeds from bottom to top (Fig. 2d) reproduces this pattern the most accurately.

The other raster scans in Fig. 2 have various problems. The top to bottom scan (Fig. 2c) produces anomalous dark blotches, and the lichen in the sideways scans (Fig. 2e and 2f) have branches on one side only.

These artifacts can be explained by considering how the pixels constrain one another. The constraint imposed on a given pixel by the pixels immediately below it is almost the same as the constraint imposed by all pixels below it. This is not true of pixels above or to either side of that pixel. The bottom-to-top raster scan therefore produces a better result than the others tested.

To utilize these types of constraint, the resynthesis procedure assigns a priority to each location in the output image. Highest priority is given to locations which are highly constrained by neighbouring pixels that have already been added. Then the following procedure is followed:

```
While there are still empty locations in
 the output image:
   • Find the empty location with the
     highest priority.
   • Choose a pixel from the input image
     to place in that location.
   • Update priorities of empty
     neighbouring locations based on the
     new pixel value.
```

A system for assigning priorities needs to be chosen. A simple prioritization scheme estimates the entropy of each unfilled location given its nearby neighbours, and gives higher priority to locations with lower entropy. The problem with this scheme is that there may be some areas of a texture in which nearby pixels tightly constrain each other, and others where there is less constraint and pixel values are more random. The areas with tight constraints will always be given higher priorities and therefore be disproportionately represented in the output image.

The approach taken in this paper is to define a normalized weighting $W(s,u)$. This weighting indicates the relative amount of information a pixel $I(s)$ gives about each of its neighbours $I(s+u)$. To ensure no part of the image is given an advantage over another part, this weighting is defined so that it sums to unity for each pixel $I(s)$. The priority of each empty location in the output can then be defined as a sum of the weightings from neighbouring pixels.

2.2 Analysing interactions between neighbouring pixels

Weightings are needed to prioritize the order in which pixels are inserted. They are also needed in the distance function used to select these pixels. To create these weightings, interactions between neighbouring pixels in the input image are analysed. The calculation of these weightings is the first stage of the synthesis procedure. For purposes of computational simplicity, the effect of each neighbour on a location is assumed to be independent of the effects of other neighbours. The interactions between constraints of different neighbours on a location are neglected.

To compute the weightings, the amount of information every pixel provides about each of its near neighbours first needs to be estimated. The neighbours being considered are those whose offset from the central pixel is a member of the set of offsets Ω_K used to calculate the distance function $D(s,t)$ (Eq. 2).

These amounts of information may be measured by considering each offset $u \in \Omega_K$ in turn. For each offset, the number of bits of information $G(s,u)$ provided by each pixel in the input image $I(s)$ about the pixel offset from it $I(s+u)$ is estimated, as explained below.

Offset pixels $I(s+u)$ are grouped into sets based on the value of the corresponding pixel $I(s)$. Within each set, the pixels will have a distribution of values. These values are modelled as being normally distributed in each colour component (red, green and blue), and the parameters of these distributions are estimated. This is used to calculate the number of bits required to encode pixels of each set.

To distribute the pixels $I(s+u)$ into sets, they are classified by the most significant m bits of the red, green and blue components of $I(s)$. The value of m should be chosen so that the sets are neither so small that an accurate estimate of the mean and standard deviation is impossible, nor so large that most of the information from $I(s)$ is discarded.

If it is assumed that all the pixels in a set come from the same distribution, standard deviations calculated from the sets can be used to describe individual pixels in those sets. This gives standard deviations for each pixel at each offset for each colour component: $\sigma_r(s,u)$, $\sigma_g(s,u)$, $\sigma_b(s,u)$.

These standard deviations can be used to calculate the entropy of each pixel $I(s+u)$ given $I(s)$. The average number of bits required to store a normally distributed variable with standard deviation σ, to an accuracy of $\pm\frac{1}{2}$, is:

$$\frac{1}{2}\log_2(2\pi) + \frac{1}{2\ln 2} + \log_2 \sigma \tag{3}$$

Summing this for each colour component gives the entropy of the pixel $I(s+u)$ given $I(s)$:

$$\begin{aligned} H(s,u) = \quad & \tfrac{3}{2}\log_2(2\pi) + \tfrac{3}{2\ln 2} \\ & + \log_2 \sigma_r(s,u)\sigma_g(s,u)\sigma_b(s,u) \end{aligned} \tag{4}$$

Performing the same calculation without splitting the image into sets gives the entropy of pixels in the image independent of any of their neighbours, H_{image}.

The number of bits given by a pixel $I(s)$ about another pixel offset from it $I(s+u)$ can then be found by subtracting the entropy of $I(s+u)$ given $I(s)$ from the entropy of pixels in the image if nothing is known about their neighbours. Call this value $G(s,u)$, where:

$$G(s,u) = H_{image} - H(s,u) \tag{5}$$

2.2.1 Prioritization weighting

A normalized weighting $W(s,u)$ suitable for prioritizing the order in which pixels are added to the output image can be defined from $G(s,u)$:

$$W(s,u) = \frac{G(s,u)}{\sum_{-v\in\Omega_K} G(s,v)} \tag{6}$$

The priorities of unfilled locations in the output image may be defined from $W(s,u)$:

$$P(s) = \sum_{u\in\Omega_K, s+u\in\Omega_O} W(L(s+u),-u) \tag{7}$$

Additionally, locations near the edge of the image have their priorities adjusted as if the positions beyond the edge of the image were already filled by pixels having average properties (i.e. with weights $W(s,u)$ averaged over all s). These locations then act as starting positions for filling the output image. The value of the first pixel added, having no neighbours, is chosen at random from the input image (on the basis of the random component of the distance function).

2.2.2 Distance function weighting

The distance function $D(s,t)$ (Eq. 2) used to select pixels also requires a set of weightings. To make best use of the input image, these weightings should reflect the degree to which each neighbour constrains the value of the pixel.

The entropy of a normally distributed random variable, such as one of the color components of a pixel, is the logarithm to base two of its standard deviation, to within a constant (see Eq. 3). This means that for each time a constraint on a particular location halves the standard deviation of possible values of one color component of that location, one more bit of information about the location becomes known. In light of this relationship, the weighting given to a neighbour might be doubled for every bit it gives about a location.

The weighting system used in the distance function was chosen to satisfy this constraint:

$$K(u) = 2^{G(-u)} \tag{8}$$

where $G(u)$ is the average value of the weightings $G(s,u)$ for a particular offset u.

2.3 Results of the procedure

Example results from the procedure are shown in Fig. 3. A seven by seven neighbourhood of pixels ($\Omega_K = [-3,3]\times[-3,3]$) was used to generate these images.

The images in Fig. 3 required an average of four and a half minutes and six and a half megabytes of memory to produce on a 300Mhz Pentium II. The time taken by the procedure increases approximately linearly with the size of the input image, and can be extremely slow for large input images.

Less successful results are shown in Fig. 4. In particular the procedure is very sensitive to features that only appear at the edge of the input image. In this case the output tends to contain artifacts such as repeating patterns. Also, as can be seen in Fig. 4b, the procedure does not recognize regularly spaced components of a texture such as tiles.

3 SPECIFICATION OF THE PLACEMENT OF TEXTURE REGIONS

Some textures contain several regions, with properties that differ from one region to another. A simple extension to the distance function used to select pixels gives a means of specifying the layout of these regions in the output image. This extension requires the definition of two new images, one which maps regions in the input, $J(s)$, and one which specifies corresponding regions in the output, $Q(t)$. It also requires weightings, $L(u)$,

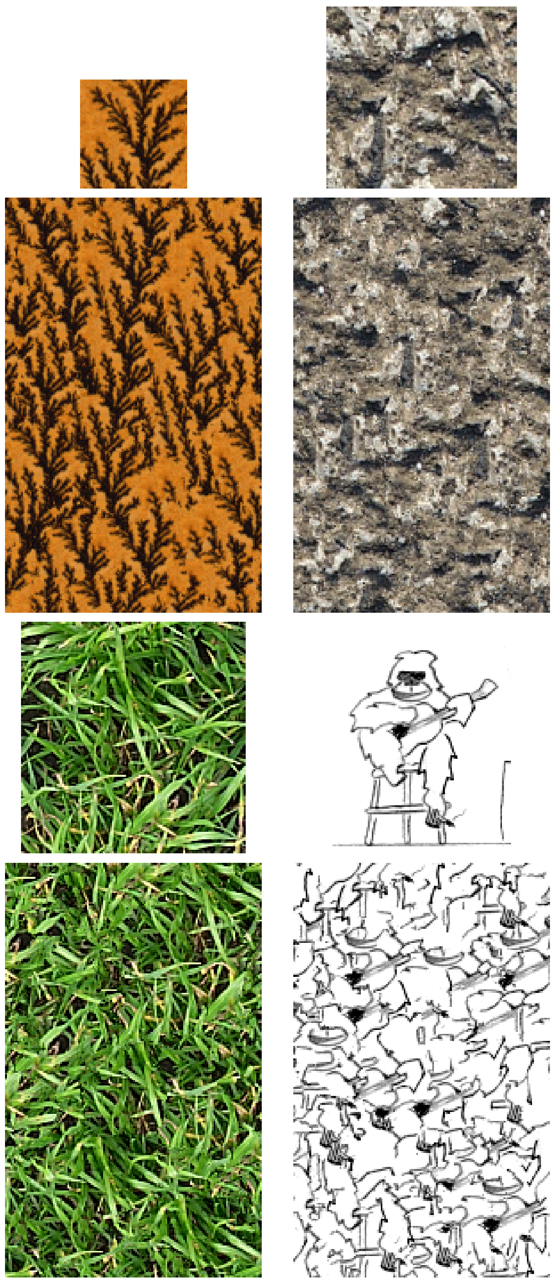
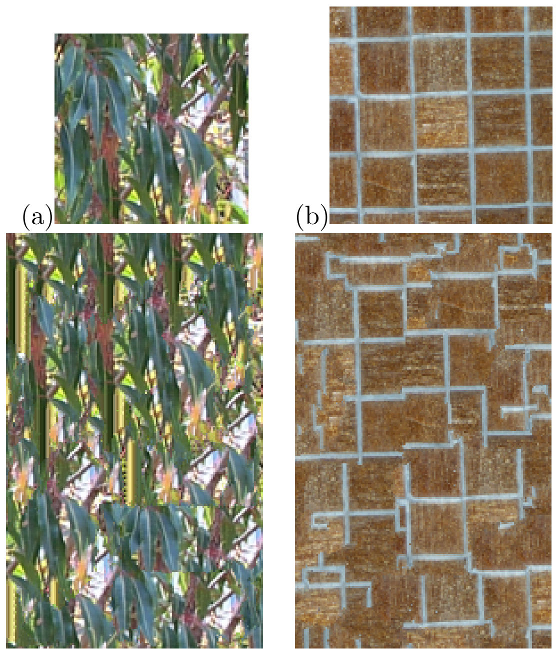


Figure 3: Sample results.
The output of the procedure is shown below each input image.

which represent the relative importance of pixels from the input map in determining the value of their surrounding pixels in the input image. These are chosen using the same method as that used to choose $K(u)$. Then a new distance function is used in place of Eq. 2:

$$\begin{aligned} D(s,t) = \quad & \varepsilon\,|A(s) - B(t)| + \\ \sum_{u \in \Omega_K, t+u \in \Omega_O} \quad & \{K(u) d(I(s+u), O(t+u)) + \\ & L(u) d(J(s+u), Q(t+u))\} \end{aligned} \tag{9}$$



An implementation of the procedure can be obtained from `http://www.csse.monash.edu.au/~pfh/resynthesizer/`

Figure 4: Some failures.

3.1 Results of the extended procedure

An example of the use of the extended synthesis procedure is shown in Fig. 5. The texture used was a photograph of clouds (Fig. 5b). A map of this texture was created showing regions of cloud and blue sky (Fig. 5a). Another map, of a checkerboard pattern, was used as the output map (Fig. 5c). The result of applying the procedure to these three images is shown in Fig. 5d.

Texture re-synthesis has been previously applied to the removal of objects on a homogeneous background by synthesizing a new section of that background [Igehy97] [Efros99]. The ability to constrain placement of texture regions allows synthesis of a new background to replace an object, even if the background is non-homogeneous. An example of this is shown in Fig. 6. Fig. 6a shows a picture of a donkey standing in a field. In this picture, the field is not homogeneous because of perspective expansion. A feature map of the image (Fig. 6b) was constructed with parts of the image equidistant from the camera having the same value. A new section of the background was then synthesized to replace the donkey, using the rest of the background as input texture. The new section was constrained to join with the existing background and to follow the feature map. The

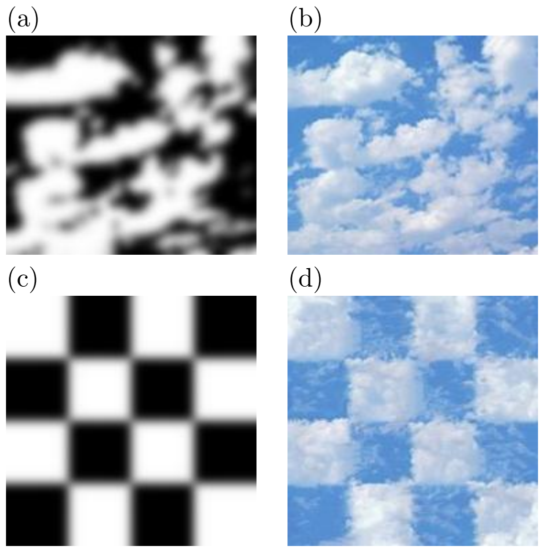


Figure 5: Constraining a cloud texture to a checkerboard pattern.
Input map (a), input texture (b), output map (c), and output of the extended synthesis procedure (d).

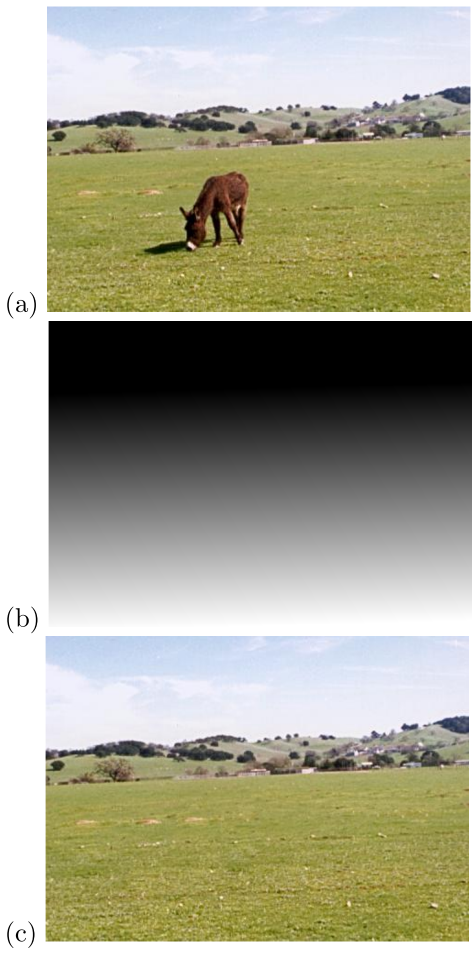


Figure 6: Example of object removal.
Original image (from [Igehy97]) (a), map (b), and image with object removed (c).

result is shown in Fig. 6c.

4 CONCLUSION AND FUTURE WORK

A procedure has been described for synthesizing an image of arbitrary size with the same texture as that of a sample input image. The procedure is capable of reproducing large features from this input image, even though it only examines interactions between pixels that are close neighbours. Significantly, the procedure does not have any preconceived notion of what features to expect in the input texture.

A simple extension to this procedure allows feature placement in the new image to be constrained. It is effectively a generic filter, in that when given an example of a change to one image it can reproduce the same change in another image.

Future work on the procedure described here could refine the distance function to consider matches that have similar structure even if they are slightly lighter or darker or have a different hue. It could also be extended to work on non-flat geometries, to allow synthesis of textures covering three-dimensional objects. It might also be applied to three dimensions to manipulate animations or solid textures, or to one dimension as a sound synthesis technique.

ACKNOWLEDGEMENTS

I thank my P.h.D. supervisor, Dr. Alan Dorin, for his patience and for many helpful suggestions. I also thank Homan Igehy and Lucas Pereira for the use of one of their images in this paper.

REFERENCES

[Blahu87] R. E. Blahut. *Principles and Practice of Information Theory.* Addison-Wesley, 1987.

[Bonet97] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of SIGGRAPH 1997*, pages 361–368, 1997.

[Efros99] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 1033–1038, 1999.

[Gagal90] A. Gagalowicz and S. De Ma. Sequential synthesis of natural textures. In *Selected Papers on Digital Image Processing*, pages 184–210. SPIE Optical Engineering Press, 1990.

[Garbe81] D. Garber. *Computational Models for Texture Analysis and Texture Synthesis*. PhD thesis, University of Southern California, 1981.

[Heege95] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH 1995*, pages 229–238, 1995.

[Igehy97] H. Igehy and L. Pereira. Image replacement through texture synthesis. In *Proceedings of the 1997 IEEE International Conference on Image Processing*, 1997. Available: `http://graphics.stanford.edu/papers/texture_replace/` (Accessed: 2000, September 7).

[Monne81] J. Monne, F. Schmitt, and D. Massaloux. Bidimensional texture synthesis by Markov chains. *Computer Graphics and Image Processing*, 17:1–23, 1981.

[Porti99] J. Portilla and E. P. Simoncelli. Texture modeling and synthesis using joing statistics of complex wavelet coefficients. In *Proceedings of the IEEE Workshop on Statistical and Computational Theories of Vision*, 1999. Available: `http://www.cis.ohio-state.edu/~szhu/SCTV99.html` (Accessed: 2000, September 7).

[Wei00] L. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH 2000*, 2000. Available: `http://graphics.stanford.edu/projects/texture/` (Accessed: 2000, September 7).