TEXTURE TRANSFER USING AFFINE TRANSFORMATION GROUPS AND PATCH BOUNDARY MATCHING

Timothy Smith and Abhir Bhalerao

Department of Computer Science, Warwick University, UK
{tsmith|abhir}@dcs.warwick.ac.uk

Keywords: texture, transfer, modelling, synthesis, patch based

Abstract

An algorithm for performing texture transfer based on an affine transformation model is presented. Image textures are modelled by placing a single patch of texture, a prototype patch, from which the texture is derived using a set of affine transformations. Overlapping target patches covering the entire image are used and a corresponding set of transformations are found to map the prototype patch to these target patches. Texture transfer can be achieved by using different prototype patches. An iterative patch boundary matching algorithm is then applied to minimise the discontinuities between adjacent patches. Illustrative results of the affine texture model and its use in texture synthesis and texture transfer are given.

1 Introduction

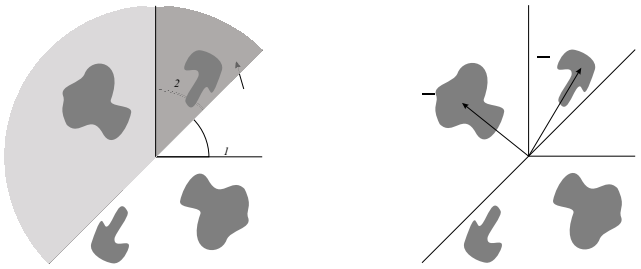
Being able to take a texture from one image and apply, or *transfer*, it to regions in another image has a number of applications in non-photorealistic rendering, computer graphics and animation. The problem is also of more general interest as it requires a model of image texture and its variation across an image. One approach is to use a syntactical patch model of texture where a texture image is represented by an example patch (or texton) and a set of transformations of the patch. For example, Efros and Freeman [2] performed texture transfer as an extension of a patch based texture synthesis algorithm. The destination image was split into overlapping square blocks and patches selected from the source texture to be placed in these blocks. Source patches were selected so as to agree with the current block's already synthesized neighbours in the region of overlap. A minimum error boundary cut was used between two overlapping blocks to reduce blockiness at the boundary.

Several texture transfer methods have been reported in the literature [3, 4, 10, 11]. Hertzmann et al. describe an analogy operator [4] so that two pairs of images can be said to be analogous. By deriving this operator for a pair of source images, a target image can then have the same operator applied to it to produce an image similar in appearance to the source image. Allowing the two source images to be identical produces a texture transfer effect. The algorithm is based around the texture synthesis algorithm from [1] where a feature vector is used to describe the similarity of pixels. Liu et al. [11] looked specifically at near regular textures. A deformed texture was separated into a geometric deformation field and a lighting deformation field with some user input to correct misplaced lattice points in the deformation lattices. Once these deformation fields have been found they can be applied to another texture and this newly deformed texture used to replace the original. Rather than extracting a deformation field, Fang and Hart [3] perform a shape from shading estimation and cluster similarly orientated regions into patches. Texture is then synthesized onto these patches using the Graph-Cut algorithm (see [10]) then combined with displacement mapping to improve silhouette appearance. Finally the patches are combined using feature matching and used to replace the original texture. Here, we aim to first explicitly model the texture variation of the target image based on a representative image patch from that texture and then replace the prototype with an example of the texture to be transferred.

Many textures have a regular or semi-regular appearance. For example, a brick wall is very regular with subtle changes in brick size and mortar thickness while a spotted leopard skin has a greater element of randomness. These textures also have common texture elements - a brick or a single leopard spot - often referred to as "textons" [8] which repeat across the image with some variation. By taking a portion of an image containing one of these elements as a prototype patch and repeating it, a synthesized image can be formed. This does not capture any of the variations that may occur across the texture so the prototype patch must be distorted as well as being repeated to produce a convincing synthesis. This was the basis of the work by Hsu et al. [6]. In fact, *any* prototype patch can be used in the texture synthesis process and *any* image can be synthesized, not restricted to the texture image used to learn the placement rules. This allows a prototype texture patch to come from one image and be mapped onto another, unrelated, image making it possible to transfer texture from one texture image to another, possibly non-textured, image. The patch distortions are represented as a set of linear transformations estimated from the spectral energy centroids of each patch and subsequently patch translations are calculated to minimize boundary errors during synthesis.

2 Affine transform model

In this section a method of obtaining an estimate of the optimal affine transformation to take a prototype patch to a target patch is outlined (further details are in [12]). Given a prototype patch, $s^{(p)}(\mathbf{x})$, and a target patch, $s^{(t)}(\mathbf{x})$, an approximation of the target patch, $s^{(p)'}(\mathbf{x})$, is to be found as an affine transformation



θ_1 defines a half-plane which is further split by θ_2 into two segments Λ_1 and Λ_2

Figure 1: Representative centroid location of Fourier energy maxima found by minimizing segment variances in Fourier half-plane.

of the prototype patch

$$s^{(p)'}(\mathbf{x}) = s^{(p)}(\mathbf{T}\mathbf{x} + \mathbf{d}) \tag{1}$$

where $\mathbf{T}$ is a linear transformation and $\mathbf{d}$ is a coordinate offset. The actual transformation is carried out using an inverse transformation to map coordinates in the target image to real number coordinates in the prototype image. Bilinear interpolation is then used to find pixel values from the discrete samples in the prototype image. A set of possible linear transformations is found by estimating representative centroids for both prototype and target patches from their respective Fourier energy spectra. Transforming the prototype patch and then matching it to the target patch allows an offset to be found as well as the best matching transformed prototype patch.

2.1 Representative centroid estimation

Most textures have one or more principal directions and vary in their local frequency. In [5], Hsu demonstrated that by estimating the location of the two energy centroids in the Fourier half-plane of a texture patch, its *intrinsic* affine frame could be estimated. Furthermore, the symmetry properties of the Fourier transform and its closure under affine transformation mean that it is possible to estimate linear transformations of the patch by the tracking the location of these energy centroids. Representative centroid vectors can be located by minimizing the sum of variances in two quadrants of the magnitude spectrum, $\hat{s}(\omega)$, of the patch being analysed:

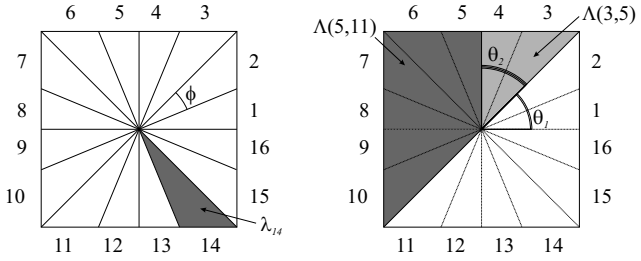
$$\hat{s}(\omega) = |\mathcal{F}(s(\mathbf{x}))| \tag{2}$$

where $\mathcal{F}$ denotes the symmetrical Fourier transform. The sectors are defined by the angles θ_1 and θ_2 so that

$$\sigma^2_{total} = \sigma^2_1(\theta_1, \theta_2) + \sigma^2_2(\theta_1, \theta_2) \tag{3}$$

Two angles, rather than one, are necessary to resolve any ambiguities that may arise if the energy magnitude spectra consist of more than a single energy cluster [6]. The segment variances can be found using

$$\sigma^2_i(\theta_1, \theta_2) = \frac{1}{M(\theta_1, \theta_2)} \sum_{\omega \in \Lambda_i(\theta_1, \theta_2)} |\hat{s}(\omega)| \, \|\omega - \mu(\theta_1, \theta_2)\|^2 \tag{4}$$



Left to right: (a) energy spectrum split into N segments of $\phi = \frac{2\pi}{N}$ with the set of coordinates in segment n denoted by λ_n; (b) $\Lambda(k,l) = \Lambda(\theta_1, \theta_2)$ with $\theta_1 = (k-1)\phi$ and $\theta_2 = (l-k)\phi$.

Figure 2: A partial summation technique is used to calculate energy variances.

where the total segment energy mass is

$$M_i(\theta_1, \theta_2) = \sum_{\omega \in \Lambda_i(\theta_1, \theta_2)} |\hat{s}(\omega)| \tag{5}$$

the segment centroid is

$$\mu_i(\theta_1, \theta_2) = \frac{1}{M_i(\theta_1, \theta_2)} \sum_{\omega \in \Lambda_i(\theta_1, \theta_2)} |\hat{s}(\omega)| \, \omega \tag{6}$$

and $\Lambda_i(\theta_1, \theta_2)$ is the set of coordinates in segment i as shown in Figure 1.

A brute force calculation of σ_{total} for each distinct combination of $0 < \theta_1 < \pi$ and $0 < \theta_2 < \pi$, where $\theta_i = \pi n / N$ and $0 \leq n < N$, is possible but very slow. In [9], Kruger greatly improved the speed by using a partial summation technique. The absolute energy spectrum is split into N segments of $\phi = \frac{2\pi}{N}$ (Figure 2a) and for each segment f_1, $\mathbf{f}_2$ and M are calculated

$$f_1(\lambda_n) = \sum_{\omega \in \lambda_n} |\hat{x}(\omega)| \, \|\omega\|^2 \tag{7}$$

$$\mathbf{f}_2(\lambda_n) = \sum_{\omega \in \lambda_n} |\hat{x}(\omega)| \, \omega \tag{8}$$

$$M(\lambda_n) = \sum_{\omega \in \lambda_n} |\hat{s}(\omega)| \tag{9}$$

where λ_n is the set of coordinates in segment n. This allows Equation (4) to be rewritten to give the variance in the region from segment k up to, but not including, segment l as

$$\sigma^2(k,l) = \frac{\sum_{n=0}^{l-1} f_1(\lambda_n) - \sum_{n=0}^{k-1} f_1(\lambda_n)}{\sum_{n=0}^{l-1} M(\lambda_n) - \sum_{n=0}^{k-1} M(\lambda_n)} - \left\| \frac{\sum_{n=0}^{l-1} \mathbf{f}_2(\lambda_n) - \sum_{n=0}^{k-1} \mathbf{f}_2(\lambda_n)}{\sum_{n=0}^{l-1} M(\lambda_n) - \sum_{n=0}^{k-1} M(\lambda_n)} \right\|^2 \tag{10}$$

(Figure 2b) with $f_1(\lambda_0) = \mathbf{f}_2(\lambda_0) = M(\lambda_0) = 0$. The summations can then be precomputed for each segment as

$$f_1^{\Sigma}(l) = \sum_{n=0}^{l-1} f_1(\lambda_n) \tag{11}$$

$$\mathbf{f}_2^{\Sigma}(l) = \sum_{n=0}^{l-1} \mathbf{f}_2(\lambda_n) \quad (12)$$

$$M^{\Sigma}(l) = \sum_{n=0}^{l-1} M(\lambda_n) \quad (13)$$

and used in Equation (10). Performance can be further increased by observing that due to the symmetry of the Fourier transform, only half the segments need to be calculated. If f_1, $\mathbf{f}_2$, M, f_1^{Σ}, $\mathbf{f}_2^{\Sigma}$ and M^{Σ} are calculated for $1 \leq l \leq \frac{N}{2}$ then f_1^{Σ}, $\mathbf{f}_2^{\Sigma}$ and M^{Σ} can be calculated for $\frac{N}{2} < l \leq N$ using

$$f_1^{\Sigma}(l) = f_1^{\Sigma}(N/2) + f_1^{\Sigma}(l - N/2) \quad (14)$$

$$\mathbf{f}_2^{\Sigma}(l) = \mathbf{f}_2^{\Sigma}(N/2) - \mathbf{f}_2^{\Sigma}(l - N/2) \quad (15)$$

$$M^{\Sigma}(l) = M^{\Sigma}(N/2) + M^{\Sigma}(l - N/2) \quad (16)$$

This requires that N is even. σ^2_{total} can be calculated from

$$\sigma^2_{total} = \sigma^2(k, l) + \sigma^2(l, k + N/2) \quad (17)$$

for all combinations of $1 \leq k \leq N$ and $k < l < k + \frac{N}{2}$ to find the minimum total variance. Once the segments with the minimum total variance have been identified the representative centroid pair can be found as

$$\mu_1(k, l) = \frac{\mathbf{f}_2^{\Sigma}(l) - \mathbf{f}_2^{\Sigma}(k)}{M_1^{\Sigma}(l) - M_1^{\Sigma}(k)} \quad (18)$$

$$\mu_2(l, k + N/2) = \frac{\mathbf{f}_2^{\Sigma}(k + N/2) - \mathbf{f}_2^{\Sigma}(l)}{M_1^{\Sigma}(k + N/2) - M_1^{\Sigma}(l)} \quad (19)$$

Note that the image patches used for centroid estimation must be *windowed* first to remove edge effects from the Fourier transforms.

2.2 Linear transformation estimation

Given a prototype block with centroids $\mu_1^{(p)}$ and $\mu_2^{(p)}$ and a target block with centroids $\mu_1^{(t)}$ and $\mu_2^{(t)}$ as column vectors $\mu_i^{(\cdot)} = \begin{pmatrix} \vdots \end{pmatrix}$ then the transformation $\mathbf{A}$ must satisfy $\mu_i^{(t)} = \mathbf{A}\mu_i^{(p)}$, $i = 1, 2$. This can be expressed as the matrix equation

$$\begin{pmatrix} \mu_1^{(t)} & \mu_2^{(t)} \end{pmatrix} = \mathbf{A} \begin{pmatrix} \mu_1^{(p)} & \mu_2^{(p)} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \mathbf{C}^{(p)} \quad (20)$$

In fact, there are eight possible transformations, $A_1 ... A_8$, resulting from the symmetry of the prototype and target spectra [5]. These can be found using the four centroid pairs

$$\mathbf{C}^{(p)} = \left\{ \begin{pmatrix} \mu_1^{(p)} & \mu_2^{(p)} \\ \mu_2^{(p)} & \mu_1^{(p)} \end{pmatrix} \begin{pmatrix} \mu_1^{(p)} & -\mu_2^{(p)} \\ \mu_2^{(p)} & -\mu_1^{(p)} \end{pmatrix} \right. \quad (21)$$

and a further four pairs $-\mathbf{C}^{(p)}$. The transformations $A_1 ... A_8$ are in the frequency domain of the patches and the corresponding spatial transformations can be found using

$$\mathbf{T}_j = \left(\mathbf{A}_j^T\right)^{-1} \quad (22)$$

2.3 Translation estimation

Once a set of linear transformation has been found, the translational part of the overall affine transformation must be calculated. The first step is to transform the *unwindowed* prototype patch using $T_1 ... T_8$ to form a set of possible synthesized target patches $s_j^{(p)'}(\mathbf{x})$ and to then take the Fourier transform of these to give $\hat{s}_j^{(p)'}(\omega)$. A spatial correlation function, ρ_j, can then be calculated for each possible transformation using

$$\rho_j = \mathcal{F}^{-1}\left(\hat{s}_j^{(p)'}(\omega)\hat{s}_j^{(t)*}(\omega)\right) \quad (23)$$

where * denotes the complex conjugate. The positions of the maxima in ρ_i directly show the translations giving the best fit to the target patch for each possible linear transformation. So that every synthesized patch completely covers the target patch, only translations within a quarter block shift are used. With no restraint on translations, synthesized patches tend to gravitate toward already synthesized areas of the image leaving areas of missing texture in the final synthesized image.

2.4 Minimum error transformation

To select the best affine transformation from the set of eight previously calculated transformations, the transformed prototypes are directly compared to the target patch. The *unwindowed* prototype patch is transformed using T_j then *windowed* (Section 3.1) and cropped to be the same dimensions as the target patch. An intensity normalization is then applied (Section 3.2). For each synthesized target path, $s_j^{(p)'}$, the root-mean-squared (RMS) error between it and the actual target patch, $s^{(t)}$, is calculated. The affine transformation with the lowest RMS error is assumed to be the best match to the target patch.

3 Synthesis procedure

The source texture image is split into square patches overlapping by half their size. These patches extend by half their size outside the defined image region which is handled by zero padding the edges of the image. A set of optimal affine transformations between a manually selected prototype patch and all of the target patches is found using the procedure described in Section 2. The transformed prototypes are windowed using a cosine squared window function, an energy normalization applied and then added to the synthesized

output image. Finally, the synthesized image is cropped to match the size of the original, unpadded, texture image. Both monochrome and colour images can be synthesized using slightly modified techniques. The same synthesis method can also be modified to perform texture transfer.

3.1 Cosine squared windowing

It is well known that two cosine squared functions overlapping by half their width will sum to unity in the overlap region. By windowing each synthesized patch using a cosine squared function $\cos^2(\pi x/B)\cos^2(\pi y/B)$, as in [5], the patches can be directly summed into the synthesized output image without blocking artifacts. Windowing also removes high frequency edge effects from the Fourier transforms of the patches allowing more accurate estimation of representative centroid pairs.

3.2 Intensity normalization

Intensity normalization is needed to produce a synthesized patch with the same intensity as the target patch resulting in a synthesized image with the same overall intensity variation as the original texture image. An intensity normalization factor α must be calculated and the synthesized patch multiplied by this factor to perform the normalization. This normalization factor is found by matching the standard deviation of intensities in the spatial domain of the synthesized patch, $\sigma_{p'}$, to that of the target patch, σ_t so $\alpha_s = \frac{\sigma_t}{\sigma_{p'}}$.

3.3 Lowpass frequency component

As observed by [5], greater accuracy can be achieved by the centroid estimation algorithm if the lowpass frequency component of the patches being processed is first removed. This highpass filter can be integrated into the main synthesis algorithm in one of two ways.

The first way is to filter the texture image to produce a lowpass version and subtract this from the original. A subsampled lowpass image is retained and added into the synthesized image as a final step. Otherwise the synthesis algorithm remains unchanged only with the prototype patch being taken from and matched to the highpass image.

The second way also performs the majority of synthesis using the highpass image. Rather than retaining the lowpass image, it is discarded after being subtracted from the original. To allow this, two versions of the prototype patch are used: one taken from the original image and one from the highpass version. Affine transformations are found between the highpass prototype and target patches taken from the highpass image. To calculate the intensity normalizations and to choose the best transformation, the affine transformations found using the highpass patches are applied to the unfiltered prototype patch then compared to appropriate target patches taken from the unfiltered image. A final synthesized image can then be reconstructed using only the unfiltered prototype patch as low frequency intensity variations across the image are now subsumed into the intensity normalization factor.

3.4 Handling colour

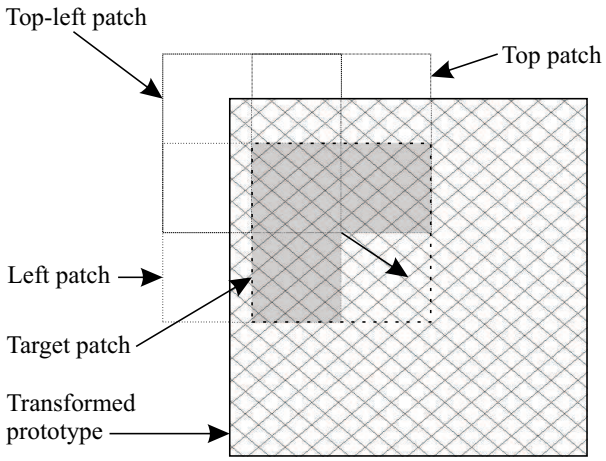
Colour can be handled in much the same way as the lowpass component of the texture image with colour information being removed leaving a greyscale image to be processed. Analogous to the first greyscale technique, the image can be converted to YUV space and subsampled versions of the colour (UV) channels can be retained while synthesis proceeds on the intensity (Y) channel. At the end of synthesis the colour channels are upsampled and added back to the synthesized image.

Alternatively, a colour prototype and greyscale prototype can be used allowing the colour channels to be discarded. A colour prototype is taken from the original colour texture and a second prototype is taken from the intensity channel once the image has been transformed to YUV colour space. Affine transformations and intensity normalizations are found using the greyscale prototype mapped to greyscale target patches. To produce the colour synthesized image, the set of transformations found are applied to the colour prototype and integrated into the final image. All windowing and blending operations are performed as in greyscale synthesis only now all operations are carried out simultaneously on three colour channels.

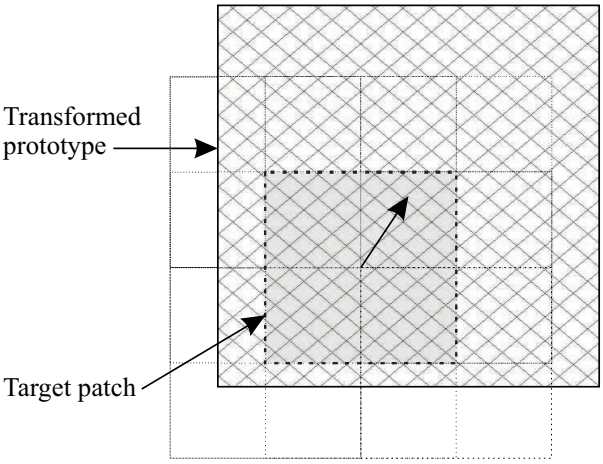
An important aspect of handling colour images in the ways described is encapsulating as much information about the image in the greyscale version since the transformations are estimated solely from this. Using a standard YUV colour space does not produce optimal results. To alleviate this problem as much as possible, the greyscale image must represent the principal component with greatest variance in the image's colour space [7]. Using an optimal colour space also reduces the visible colour fading when subsampling the chrominance channels.

4 Texture transfer

Depending on how the low-pass component (Section 3.3) and the colour channels (Section 3.4) of the image are handled, the synthesized image can retain more or less of the original's colouration. When subsampled lowpass and colour components are used with a highpass greyscale prototype the synthesized image will retain much of the colour and shading of the original. This may be desirable when transferring a texture onto a non-texture image. If an unfiltered colour prototype is used the synthesized image will take its colour and shading from the prototype losing the original's colour



(a) partial correlation



(b) full correlation

Hatched area shows synthesized patch and grey area shows correlation mask

Figure 3: Correlation masks used for patch boundary matching

and shading. A more complete texture transfer is achieved in this way, which is generally the preferred case.

Texture transfer can be achieved by either replacing the prototype patch before calculating the set of affine transformations or replacing it after calculation and re-synthesizing based on the existing transformations. For highest synthesis quality the transformations must be calculated with respect to the desired prototype patch. After synthesis the prototype patch cannot be altered and the transformations stay valid except when the prototype retains its structure, such as if it is recoloured or a filter is applied to it.

Once a set of linear transformations has been found to map a prototype patch to the target patches in the source image, a set of translations must also be found. In Section 2.3 the translation is chosen to maximize the correlation between the transformed prototype and the target patch so as to produce the best synthesis of the original texture. When performing texture transfer, however, the goal is not to produce an accurate synthesis of the original image but to create a new textured image with a resemblance to the original. For the new image to appear textured each patch must merge seamlessly into the next. The existing cosine squared windowing provides one element of this transition while matching the overlapping areas at the patch boundaries provides the other.

Estimating translations based on the overlapping areas at the boundary of patches to aid texture continuity can be achieved with a second synthesis pass. The first pass simply involves using the existing texture synthesis algorithm to find a set of linear transformations and translations. In the second pass the translations can either be slightly modified to improve the correlation between overlapping patches or completely recalculated. To recalculate the translations from scratch, each patch is synthesized using its corresponding affine transformation applied to the prototype patch and as each synthesized patch is integrated into the destination image, its translational offset is calculated to provide a best match with those patches already synthesized.

To calculate the best match a partial correlation is performed using

$$\rho_{\Delta\mathbf{x}} = \sum_{\mathbf{x}\in\Omega} (m(\mathbf{x}) - \overline{m})(s(\mathbf{x}+\Delta\mathbf{x}) - \overline{s}) \qquad (24)$$

where Ω is the set of coordinates in the mask data set m, $\overline{m}$ is the mean of the pixels being considered in the mask, s is the image data, $\overline{s}$ is the mean of the pixels under the mask and $\Delta\mathbf{x}$ is the offset of the mask in the image. Altering $\Delta\mathbf{x}$ changes the position of the mask in the image and produces a correlation function. A non-rectangular mask is used (Figure 3a) containing the overlap areas of neighbouring patches synthesized thus far while the image, s, is the transformed prototype. As the mask must lie completely within the image, the translational offset selected is implicitly restricted to a quarter block shift in any direction ($\pm\frac{\text{prototype patch size}}{4}$).

Alternatively, an iterative "shaker box" algorithm can be applied to modify the translation of each patch attempting to find the best overall synthesized image with the highest total inter-patch correlation. The starting translations can either be those produced by the first synthesis pass when mapping the prototype patch onto the target patches or completely new translations as described above. At each iteration, all the synthesized target patches are visited in a random order and a new translation calculated based on maximizing the correlation with all the surrounding patches. Rather than performing a partial correlation, so that only the patches to the left and above the target patch influence its positioning, a full correlation is performed. This can be done using Equation (24) with a rectangular mask (Figure 3b) or using Equation (23) for improved performance. To ensure consistency with the translations selected using the partial correlation method, Equation (24) is used despite being the slower of the two methods. The translations are again implicitly limited to $\pm\frac{\text{prototype patch size}}{4}$ so that the transformed prototype completely covers the target patch. By repeatedly "shaking" the synthesized patches they should fall into place with each other and the synthesized image converge.

Unfortunately, this is not always the case (see [12]) and the solution either does not converge or converges very slowly. The shaker box algorithm can be modified to force convergence by successively restricting the change in translation from one iteration to the next until the solution has converged or no change in translation is allowed. The initial change in translation can also be limited to provide a greater bias toward previously chosen translations.

5 Experiments and Discussion

The reptile skin and burlap texture images (Figure 4a) represent perfect examples of the class of textures best suited to this patch based synthesis method. There is clear repetition and a subtle change in texton orientation across the images. Applying the synthesis algorithm with a 32x32 prototype patch allows the change in orientation about two-thirds of the way across the reptile image to be captured and the structural detail in both textures to be accurately synthesized (Figure 4b). The patch size is approximately the same as the feature size in the reptile texture allowing for a better synthesis than for the burlap texture. In the burlap synthesis there is some blurring visible at the patch boundaries which can be removed by using a smaller patch size better matched to the burlap feature size. The signal-to-noise ratios (SNR) of the original image to the synthesized image for the reptile and burlap textures reveal a significant increase in synthesis quality compared to Hsu's original algorithm [5]. The reptile synthesis has an SNR of 17.5dB compared to Hsu's value of 4.7dB while the burlap synthesis has an SNR of 15.2dB compared to 1.8dB.

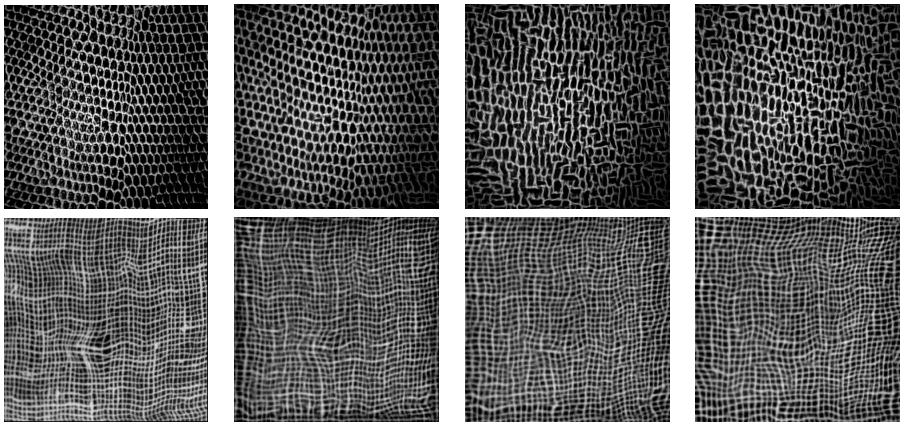
To examine how well the shaker box algorithm (Section 4) matches patch boundaries, the reptile and burlap textures were used as they both have clear structures making discontinuities clearly visible. The affine transformations were found as above with a 32x32 target patch but new patch translations were calculated rather than using the existing translations. Figure 4c shows the reptile and burlap textures synthesized with new translations calculated using the non-iterative partial correlation method. There is a high level of patch to patch continuity with no obvious patch boundaries although much of the regular structure of the original reptile texture has been lost. While the target patch size (32x32) is large enough to contain a complete reptile "cell", the fractional patch overlap used in the correlation calculations is only large enough to contain part of a cell leading to this loss of structure. A simple way to alleviate this problem would be to use a larger target patch size but this also reduces the overall synthesis quality so is not an entirely satisfactory solution.

The results of applying the forced shaker box algorithm to both sets of translations are shown in Figure 4d. The search region was decreased by one in size every five iterations starting with a search region of $\pm\frac{\text{target patch size}}{4}$ after new translations were initially calculated for all patches as above. The number of iterations to spend at each search size can be decreased to force faster convergence but this sacrifices the overall quality of the boundary matching. A larger number of iterations per search size could potentially produce better boundary matches but there is no guarantee of convergence at a particular search size. Consequently, five iterations per search size were chosen to allow some room for the algorithm to initially make large changes to a patches translation within the current search space while forcing fairly rapid convergence, typically after around 20 iterations. Highly satisfactory results are obtained with virtually no visible boundary discontinuities (Figure 4d) and some of the structure of the reptile skin texture previously lost when boundary matching is regained. Also, there is arguably less blurring at the patch boundaries with the new translations than with the original translations (Figure 4b and d respectively) for the burlap texture.

The top two rows of Figure 5 show the extreme cases of texture transfer. In the top row the colour snakeskin prototype is sufficiently similar to the reptile prototype that it can be directly substituted into the existing set of transformations (those used to generate Figure 4) and produce a convincing transfer. All colour and lowpass components are contained in the 64x64 prototype patch. Not only does this texture transfer add colour to the original but it subtly modifies texture detail. Recalculating the transformations for the modified prototype patch would improve synthesis quality, albeit in this case only slightly, and allow the prototype used to be highly dissimilar from the original.

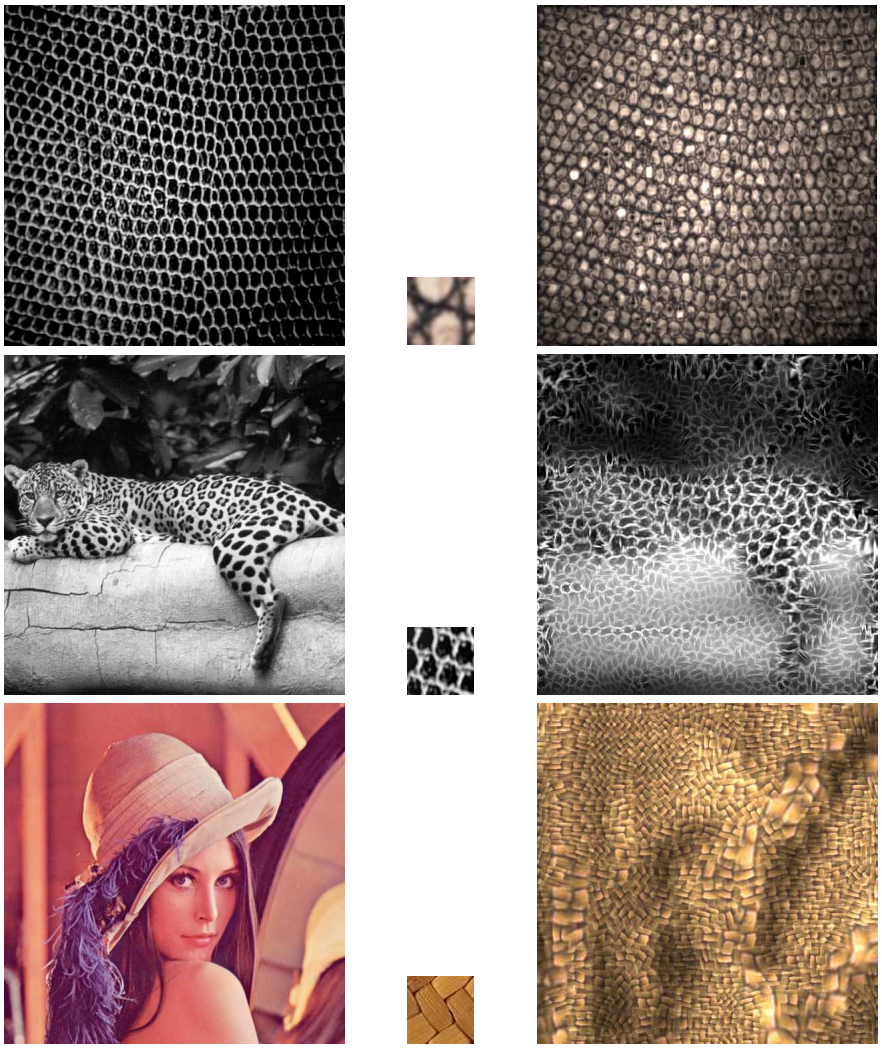
The middle row of Figure 5 shows the opposite case where a texture has been transferred onto a radically dissimilar non-texture image. All the transformations are calculated from scratch mapping the 64x64 reptile prototype onto the jaguar image while retaining the lowpass component of the jaguar image. This process overlays the highpass detail of the reptile texture onto the jaguar with the lowpass component clearly visible through this detail. The success of the boundary matching is most apparent in both the light area of wood which the jaguar is resting on and along the jaguar's back where it merges into the background. With boundary matching, the texture on the light wood is structured more like in the original reptile texture and the edge between the jaguar and the dark background is quite indistinct. By using patch boundary matching the textured jaguar appears to be well integrated into the rest of the image.

Patch boundary matching can also be applied to colour images as shown in the bottom row of Figure 5. Using a target patch size of 64x64 allows much of the weave texture to be transferred producing the feeling of an image made of basket weave. Calculating new translations using the shaker box algorithm improves the appearance of the texture transfer producing large regions of consecutive texture. There is a slight pink colouration of some patches, probably due to an incorrect intensity normalization being applied to the patch. Although the shaker box algorithm changes the translations of the patches, it does not renormalize their intensities relative to the original image which may be required. Usually, in the case of the prototype patch being a texture, the intensity is constant across the prototype so renormalization is not required.



(a) original (b) synthesized (c) no iterations (d) iterated

Figure 4: Synthesis results for reptile and burlap textures



(a) original (b) prototype (c) texture transfer

Figure 5: Texture transfer of snakeskin, reptile and weave texture onto reptile, jaguar and Lena images respectively (top to bottom)

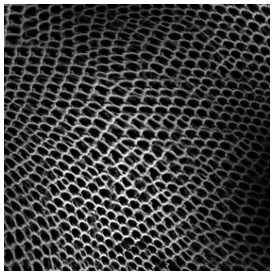


Figure 6: Reptile texture synthesized with smooth transformations

6 Conclusion and suggestions for future work

It has been shown that a set of local affine transformations can be effectively used to model texture and to perform texture transfer. By matching patch overlap regions, the transferred texture can be made continuous across the final synthesized image giving the illusion of the image being made out of the texture. Such an algorithm may be used to place new textures on existing objects in an image as in [3] and [11]. While the results are not as visually appealing as those achieved by others, such as [2], they do serve to illustrate one of the wider applications of the affine transformation based texture representation.

The proposed "shaker box" algorithm also has applications in texture synthesis. If a texture is to be synthesized where only a set of linear transformations is known with no original image to derive the patch translations from, the shaker box algorithm can be successfully used to construct a visually continuous texture. Given a smooth set of transformations generated from an innovative process or a parametric function, as in Figure 6, a continuous texture may be generated by using the shaker box boundary matching algorithm.

Despite matching the boundary overlap regions of the target patches, the synthesized images lack some of the feel of the applied texture. Some of the loss in structure of the texture is due to the overlap regions being too small. Rather than correlating target patches to the partially synthesized image, each synthesized patch could be stored unwindowed and then retrieved when needed. By removing the cosine squared windowing more of the texture's structure would play a part in the correlation and allow the synthesized texture to preserve the original's overall structure. Separate correlations would need to be performed with each of the target patch's neighbours and then combined to form an overall correlation function. As the neighbouring patches would be *whole* synthesized patches rather than just their central parts, the effective overlap region would also be extended. Another way to improve the visual quality could be to incorporate the Graph-Cut seaming idea [10] into the patch overlaps after the boundary matching.

Another cause of poor texture transfers is the affine transformations changing too quickly from patch to patch. If there is a large change between adjacent patches there is a visible discontinuity that cannot be removed by minimizing the overlap error. To reduce this problem the synthesis algorithm must be modified to take into account the surrounding patches' transformations when selecting an affine transformation to map the prototype patch to the target patch, perhaps by smoothing or an explicit autoregressive type model.

References

[1] Michael Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 217–226. ACM Press, 2001.

[2] Alexei A. Efros and William T. Freeman. Image Quilting for Texture Synthesis and Transfer. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 341–346. ACM Press / ACM SIGGRAPH, 2001.

[3] Hui Fang and John C. Hart. Textureshop: texture synthesis as a photograph editing tool. *ACM Transactions on Graphics*, 23(3):354–359, 2004.

[4] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image Analogies. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 327–340. ACM Press / ACM SIGGRAPH, 2001.

[5] Tao-I. Hsu. *Texture Analysis and Synthesis using the Multiresolution Fourier Transform*. PhD thesis, University of Warwick, UK, 1994.

[6] Tao-I. Hsu, A. D. Calway, and R. Wilson. Texture Analysis using the Multiresolution Fourier Transform. In *Proc 8th Scandinavian Conference on Image Analysis*, pages 823–830. IAPR, 1993.

[7] I. T Jolliffe. *Principal component analysis*. Springer, New York, second edition, 2002.

[8] B. Julesz and J. R. Bergen. Textons, the Fundamental Elements in Preattentive Vision and Perception of Textures. *Bell Systems Technical Journal*, 62(6), July-August 1983.

[9] Stefan Kruger and Andrew Calway. Multiresolution Motion Estimation using an Affine Model. Technical Report CSTR-96-002, University of Bristol, June 1997.

[10] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut Textures: Image and Video Synthesis Using Graph Cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 22(3):277–286, July 2003.

[11] Yanxi Liu, Wen-Chieh Lin, and James Hays. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics*, 23(3):368–376, 2004.

[12] Timothy Smith. Texture modelling and synthesis in two and three dimensions. Master's thesis, University of Warwick, UK, 2004.