Texture Synthesis from Non-Fronto-Parallel Textures and Related Applications
CS 790 Project Report

Saurabh Goyal
saurabh@cs.wisc.edu
Advisor: Prof. Chuck Dyer

1 Introduction

Previous work on texture synthesis has only addressed fronto-parallel textures, i.e., when a camera is parallel to a planar texture. In this report we describe how perspective deformations in the texture image can be dealt with for texture synthesis. The synthesized image retains the perspective effects visible in the original texture image. The first few sections contain preliminary discussion on texture synthesis and rectification.

Texture, in images, is a term used to refer to a pattern of intensities. The cause of this pattern to appear in the image is an interaction between lighting, surface normals and surface reflectance. This pattern could be stochastic or regular. Most natural textures are stochastic, for example an image of wood, grass, foliage, water, hair, etc. Examples of regular textures include images of a brick wall, patterned garments, etc. Regular textures are mostly images of man-made objects which are the result of a repetitive process.

Figure 1: The left image is a regular texture whereas the one on the right is stochastic.

The most common way of representing a texture is to apply various filters over the image and then gather statistics from the responses. Choosing the scale and orientation of the filters is not trivial and most vision algorithms for textures require the user to identify the texel size.

Two standard problems explored for texture are texture synthesis and shape from texture. The work presented in this report uses and develops techniques in these problem domains. The next two sections describe these two problems in detail.

2 Texture Synthesis

The goal here is to construct large regions of texture from a small finite sample. The assumption is that the finite sample is large enough to capture the properties of the texture. The crude approach to this would be to tile the small sample. However, this produces visual artifacts at the edges where the images are joined. There is considerable literature on better techniques for texture synthesis. Almost all of them essentially build probability models for the texture to be synthesized based on the example image, and then draw on the models to obtain the texture. Randomness is introduced by allowing tolerance to a certain level of error or by forcing random noise into the image.

A variety of techniques using Gibbs sampling [9], multi-resolution filtering [3], and properties of joint wavelet coefficients [7] have been proposed for texture synthesis. Efros and Leung [5] presented a simple technique that does well on both regular and stochastic textures. Their idea is to model the texture as a Markov Random Field. It is assumed that the intensity at a pixel given its neighbors is independent of the rest of the image. In order to synthesize a pixel, a square window around it is compared to other windows in the image and center pixels from matching windows are used to form the probability model for the pixel to be synthesized. A center pixel is then randomly chosen from the matching windows. To allow for randomness and because matches will not be perfect, some amount of error is allowed in the matching process.

Efros and Freeman noticed that a lot of extra work is done by only synthesizing one pixel at a time and proposed a faster algorithm that they called image quilting [4]. The image quilting algorithm synthesizes an entire patch at one time. The algorithm requires the user to choose a patch size and an overlap size. To synthesize a patch, all patches in the example image are checked. A random patch, which satisfies the overlap constraints, is chosen

and placed at the location of the patch to be synthesized. In order to match the boundaries of the overlapping patches, the minimum error boundary cut is found. Finding the minimum error boundary cut is done using dynamic programming. The complete quilting algorithm, as given in [4], is

1. Go through the image to be synthesized in raster-scan order in steps of one block (minus the overlap).

2. For every location, search the input texture for a set of blocks that satisfy the overlap constraints (above and left) within some error tolerance. Randomly pick one such block.

3. Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block onto the texture. Repeat.

We'll modify the image quilting algorithm for our texture synthesis technique so it is helpful to know the details of the image quilting algorithm [4].

3 Shape from Texture

The other problem that researchers have looked at is to recover the shape of a textured object from the projected texture. The foreshortening effects are reflected in the distribution of texture elements. There has been work on recovering a general surface but in this report we will only examine the case of planar textures. Many non-planar textures that have relatively small deviation from planarity compared to the viewing distance can also be assumed to be planar. The problem is to find the slant and tilt of the textured plane with respect to the camera. It can also be thought of as finding the homography between a fronto parallel view of the textured plane and the actual image. Various assumptions are required to solve this problem; typical camera assumptions include orthographic projection or camera calibration. Some assumption of uniformity is also required on the texture. An assumption of isotropy is used in [8]. An *isotropic* texture is one in which the distribution of texture elements is independent of the orientation. Such an assumption allows metric rectification of the texture image. However, it is impossible to recover the scale and rotation without forcing additional constraints. The major problem with this assumption is that isotropic textures are rare and therefore methods based on the isotropy assumption do not handle a general enough class.

The more general and useful assumption is that of *homogeneity*. A homogeneous texture is one that has repetitive texture elements, i.e., local windows drawn from the image look the same [6]. A property of homogeneous textures is that under an affine transformation the texture remains homogeneous. So, by only forcing homogeneity, we cannot recover the affine distortion in the texture image. It is, however, possible to recover the perspective distortion and the most common way of doing it is to use optimization methods to find a transformation that makes the texture most homogeneous. The methods are very slow as a large space of transforms has to be searched. Criminisi and Zisserman [2] presented a two step process for recovering the perspective distortion. First they computed the vanishing line by estimating the direction of the vanishing line and then its distance from the center of the image. Once the vanishing line is known it is easy to form a transformation matrix that can correct the perspective distortion. The important result used for finding the direction of the vanishing line is that points on a line parallel to the vanishing line have the same perspective distortion. So, they look for the direction in which points undergo the same distortion. The measure used for this is the variance of normalized auto correlation (NAC) values along a direction, which they seek to maximize. The algorithm, as stated in [2], is

1. Repeat
   (a) Select a random image patch (i.e., a random center location, $\mathbf{x}$) and compute the related NAC surface.
   (b) 1D search for the direction θ corresponding to the maximum value of variance of NAC values inside a strip of fixed width, centered on $\mathbf{x}$ and with direction θ.
2. The required direction corresponds to the median of the set of computed angles θ.

A crude implementation of the above algorithm was done. However, since no refinement techniques were used in our implementation, the algorithm does not work well for most images. Figure 2 shows the result of applying our implementation to a perspective image of a checkerboard pattern. The top image is the input texture image and the graph at the bottom shows variance of NAC values for each orientation with respect to a chosen patch. The maximum value of 0.1926 was obtained at approximately 46 degrees. The true vanishing line is at an angle of 45 degrees (the positive y axis faces downwards).

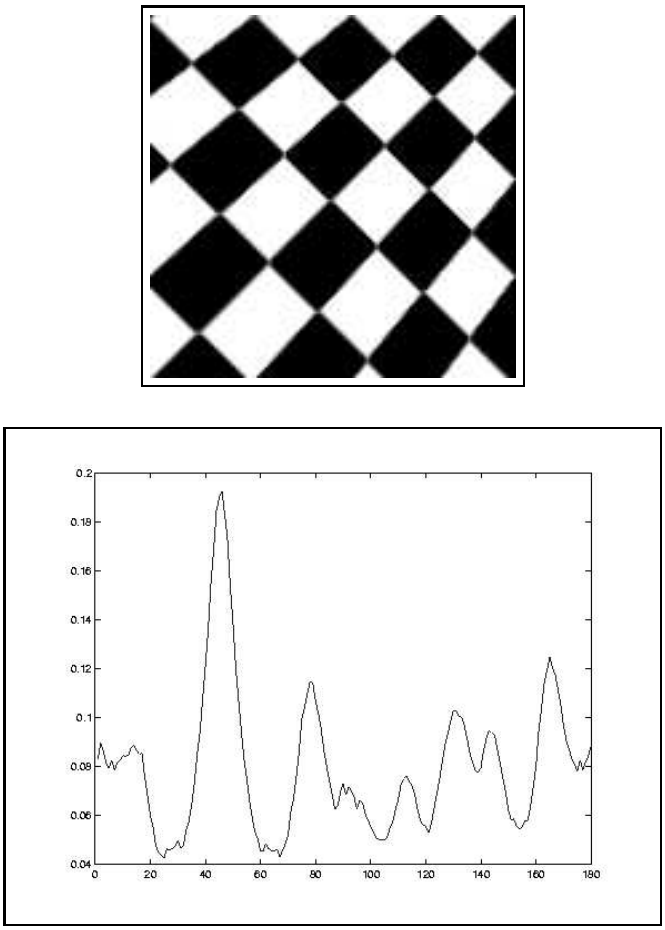


Figure 2: Results of computing the direction of the vanishing line.

One advantage of the algorithm is that it is quick because fast techniques can be used for computing auto-correlation and then only one scan over the image is required to add the contribution of every pixel to the variance of NAC values. Estimating the second degree of freedom of the vanishing line, i.e., its distance from the image center is done using a similar optimization process. Criminisi and Zisserman [2] defined a new similarity function they called Projective Normalized Auto Correlation (PNAC). They searched for the distance at which the variance of the entire PNAC surface is maximized. This second step is much more time consuming because of the following reasons:

1. The range of distances to be searched is unlimited whereas the search

space for an angle is between 0-180 degrees.

2. For each distance, the PNAC surface has to be computed over the entire image. Whereas, to find the direction, for each orientation the NAC surface has to be computed only for the pixels along that orientation.

3. To compute the PNAC, the square window has to warped for every pixel and therefore no fast techniques can be used.

The last problem can be resolved by rectifying the image before computing the PNAC, but this introduces sampling problems. We will see below how computing only the direction of the vanishing line can be helpful in non-fronto-parallel texture synthesis.

4 Texture Synthesis of Non-Fronto-Parallel Textures

In this section we describe how a homogeneous texture viewed with a general perspective camera can be used to synthesize larger regions of the same texture. The goal is to extend the texture in a way that the perspective nature of the texture remains unchanged. There are two ways to attack this problem:

1. Two step approach – First apply a texture rectification algorithm to generate a fronto-parallel view of the texture and then apply any of the previous texture synthesis algorithms. The synthesized texture can then be warped back using the inverse of the rectifying transformation. As pointed out in the previous section, it is not possible to obtain a metric rectification of a homogeneous texture. However, it turns out that this is not necessary. We can perform an affine rectification to yield a homogeneous texture. The homogeneous texture can then be extended using algorithms such as Image Quilting [4].

2. Projective invariant texture synthesis algorithm – Use a new algorithm for texture synthesis that is invariant to projective transformations. The advantage of such an algorithm is that it avoids sampling errors during rectification.

A problem with the first approach is that rectification of a texture is time consuming. If a projective invariant synthesis algorithm, which does

not require estimation of the transformation, can be used, the performance of texture synthesis will be much better. However, finding such an algorithm seems unlikely. If we could find such an algorithm, then we could probably keep extending the texture until the vanishing line of the texture is reached and beyond which the texture cannot be extended. We could, therefore, estimate the vanishing line of a texture image using this texture synthesis algorithm. Therefore, a projective invariant algorithm would be at least as time consuming as estimating the transformation.

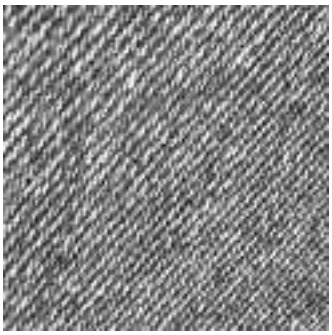
So, whichever approach we use, an algorithm for texture synthesis of non-fronto-parallel textures will be time consuming. However, the direction of the vanishing line can be estimated quickly and can be used for limited texture synthesis as described below.

4.1 Directional Quilting

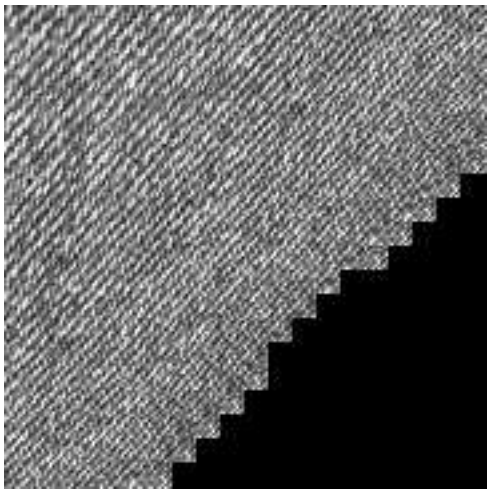
We now describe an image-based texture synthesis algorithm that only requires the estimation of the direction of the vanishing line. The key observation that leads to the algorithm is that *in a perspective image of a plane, points on a line parallel to the vanishing line have the same perspective distortion*. Criminisi and Zisserman presented a proof for this observation in [2]. In image quilting, a new patch is synthesized by searching through all patches of that size in the example image. However, under a perspective transformation, only the patches that lie on a line parallel to the vanishing line and containing the location of the new patch, should be used to form a probability model for the new patch. This is because only these patches have undergone the same perspective distortion as the new patch.

The following steps describe the complete directional quilting algorithm:

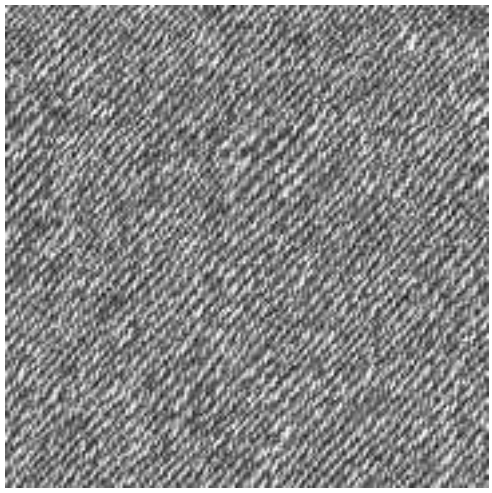
1. Estimate the direction of the vanishing line as described in [2].
2. Go through the image to be synthesized in raster-scan order in steps of one block (minus the overlap).
   (a) For every location, search through blocks in the input image that lie on a line passing through the location and are parallel to the vanishing line.
   (b) Select a random block from the blocks that satisfy the overlap constraints.



(a) Input texture image.



(b) Output of Directional Quilting.



(c) Output of Image Quilting.

Figure 3: Texture synthesis results.

(c) Compute the minimum error boundary cut between the chosen block and the overlapping blocks in the synthesized image and make that the boundary of the chosen block. Paste the chosen block onto the texture.

This algorithm is exactly the same as the image quilting algorithm except for the fact that we only look through blocks lying on a line. We use the same overlap constraints and techniques to find the minimum error boundary cut as in image quilting [4]. The error in overlap is computed using the SSD between the block and the synthesized image for the overlapping region. All blocks that fall within $(1 + threshold) * E_{min}$ are assumed to satisfy the overlap constraints. To find the minimum error boundary cut, we would have to search through all possible paths. However, by assuming that the boundary is continuous, this can be formulated as a dynamic programming problem.

$$E_{ij} = e_{ij} + min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1})$$

The above expression can be used to find the minimum error vertical cut and a similar expression can be used to find the horizontal cut.

Figure 3(b) shows the results obtained by applying our directional quilting algorithm to the image in (a). The vanishing line of the example image makes an angle of 135 degrees from the x axis. The direction was not computed but manually entered. It can be seen that the perspective effects are preserved in this image. It can also be seen that the texture has not been extended in the direction perpendicular to the vanishing line. Image (c) is obtained using the original image quilting algorithm and does not preserve the perspective effects in the original image.

One advantage of this algorithm is that it is quite fast because (1) only the direction of the vanishing line is estimated and (2) only blocks along a line are searched for synthesizing a new block. Since the algorithm is image based, no sampling errors are introduced due to rectification.

The main limitation of the algorithm is that it only allows synthesizing texture along the direction of the vanishing line. We cannot extend texture in the direction perpendicular to the vanishing line without computing the distance of the vanishing line as well.

5 Related Applications

5.1 Real Scene Synthesis

The first application explored was the synthesis of real scenes that contain perspective textures.

Figure 4(a) shows an example of an image of a sunrise. Since both the water and sky planes have the same vanishing line, no segmentation is required before synthesizing a larger image. Image (b) shows the result of applying directional quilting on the example image. The perspective effects in the water are clearly visible. Applying the standard texture synthesis algorithms on this image do not produce comparable results. For images containing multiple planes with different vanishing lines and non-planar objects, segmentation of each plane would be needed and each plane would have to be extended separately. Combining the extended planes may be complicated.

5.2 Foreground Removal

As with other texture synthesis algorithms, foreground removal is also an area where directional quilting can be applied. Directional quilting can possibly handle a larger range of textures from which a foreground object has to be removed. Foreground removal does not typically require extending the texture beyond the current image boundary and therefore directional quilting can be used. In addition, foreground removal for more realistic scenes is also possible. Figure 5 shows an example of foreground removal from a real scene. A rectangular area around the car was selected. Directional quilting was performed only in the rectangular area using blocks that lie outside the rectangular area. There are some artifacts visible on the right and bottom edges of the removed area because the original algorithm was not modified to take the overlap on the right and bottom edges into account. This extension should be easy and will produce better results. It may be useful to explore better filling orders to take linear structures into account. Criminisi et al. [1] provided a reference on foreground removal using a more sophisticated fill order.

6 Conclusions and Future Work

This report presented a brief summary of previous work on texture synthesis and planar texture rectification and also described a novel approach for

(a) Example image of a sunrise.

(b) Synthesized image.

Figure 4: Results of real scene synthesis.

(a) Input image with a foreground object.

(b) Image with the car removed.

Figure 5: Results of foreground removal.

texture synthesis of non-fronto-parallel textures. The algorithm is image based and therefore does not introduce sampling errors, but it does require the estimation of the direction of the vanishing line as a first step. Two applications of the algorithm were also described.

The major limitation of the directional quilting algorithm is that it only allows extension of a texture in one direction. And unless faster methods to compute the perspective transformation can be found, it seems unlikely that a fast algorithm that is not restricted to one direction is possible. Assuming less general transforms locally may allow extrapolation in the direction perpendicular to the vanishing line. This problem is similar to generating the next number in a series given the previous numbers. Depending on how many previous values are required to generate the new value, the new value would be more or less accurate. In the simplest case, we could assume the previous two values are related by a scale factor and use this scale factor to generate the new value. However, doing this repeatedly would introduce artifacts and the original perspective effects would not be retained. Assuming locally-affine models is also something that could be helpful. Looking at perspective textures in transform domains may give useful insights into developing faster algorithms for computing the second degree of freedom of the vanishing line.

There are various other applications of texture rectification and texture synthesis. Texture rectification algorithms will be useful for single view metrology in which a texture could be used for estimating vanishing points. Wide baseline stereo matching is another application. In wide baseline stereo, correlation-based approaches do not work well because of significant perspective distortion. In such cases, textures patches could be matched against each other but, in order to match them, the transformation between them may have to be computed. Texture synthesis also has other applications such as texture transfer, which can be extended to non-fronto-parallel textures.

References

[1] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *CVPR03*, pages II: 721–728, 2003.

[2] A. Criminisi and A. Zisserman. Shape from texture: Homogeneity revisited. In *Proceedings of the 11th British Machine Vision Conference, Bristol*, pages 82–91, UK, September 2000.

[3] J. S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. In T. Whitted, editor, *Proc. SIGGRAPH 97*, pages 361–368. Addison Wesley, 1997.

[4] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *Proc. SIGGRAPH 2001*, pages 341–346, 2001.

[5] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. 7th Int. Conf. Computer Vision*, pages 1033–1038, 1999.

[6] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*, chapter 9: Texture. Prentice Hall, Upper Saddle River, N.J., 2003.

[7] J. Portilla and E. Simoncelli. Texture modeling and synthesis using joint statistics of complex wavelet coefficients. In *Proc. Workshop on Statistical and Computational Theories of Vision*, 1999.

[8] A. P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1-3):17–45, 1981.

[9] S-C. Zhu, Y. Wu, and D. Mumford. FRAME: Filters, random field and maximum entropy: – Towards a unified theory for texture modeling. *Int. J. Computer Vision*, 27(2):1–20, 1998.