Shape from Periodic Texture Using the Spectrogram

John Krumm and Steven A. Shafer

CMU-RI-TR-91-29 2

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

November 1991

is research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aero-utical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract 3615-90-C-1465, Arpa Order No. 7597. This first author is supported by NASA under the Graduate Stu-ıt Researchers Program, Grant No. NGT-50423.

e views and conclusions contained in this document are those of the authors and should not be interpreted representing the official policies, either expressed or implied, of the U.S. Government.

Table of Contents

Abstract

Texture has long been recognized in computer vision as an important monocular shape cue, with texture gradients yielding information on surface orientation. A more recent trend is the analysis of images in terms of local spatial frequencies, where each pixel has associated with it its own spatial frequency distribution. This has proven to be a successful method of reasoning about and exploiting many imaging phenomena. Thinking about both shape-from-texture and local spatial frequency, it seems that texture gradients would cause systematic changes in local frequency, and that these changes could be analyzed to extract shape information. However, there does not yet exist a theory that connects texture, shape, and the detailed behavior of local spatial frequency. We show in this paper how local spatial frequency is related to the surface normal of a textured surface. We find that the Fourier power spectra of any two similarly textured patches on a plane are approximately related to each other by an affine transformation. The transformation parameters are a function of the plane's surface normal. We use this relationship as the basis of a new algorithm for finding surface normals of textured shapes using the spectrogram, which is one type of local spatial frequency representation. We validate the relationship by testing the algorithm on real textures. By analyzing shape and texture in terms of the local spatial frequency representation, we can exploit the advantages of the representation for the shape-from-texture problem. Specifically, our algorithm requires no feature detection and can give correct results even when the texture is aliased.

1. Introduction

Texture has long been considered an important shape cue in monocular images, starting with observations in biological vision by Gibson[14] in 1950. The corresponding algorithms developed in computational vision exploit the systematic changes in a projected texture's appearance to find the surface normal of the underlying shape. This effect is illustrated in Figure 1, which shows a Brodatz[7] cotton canvas texture synthetically mapped onto a plate. The angle and changing depth of the plate combine to make the texture appear "smaller" as the plate recedes. A more recent trend in image understanding, also with roots in biological vision, is local spatial frequency analysis. Here, the image is represented in terms of the local spatial frequencies at every pixel -- the "space/frequency representation". Coherence and changes in local spatial frequency from point to point can be used to understand a rich set of image phenomena that cannot be analyzed easily in the space or frequency domain alone[20]. Since texture is fundamentally a frequency phenomenon, and since shape is fundamentally a spatial phenomenon, it is natural to approach the shape-from-texture problem in terms of this representation. In Figure 1, for example, we show the local Fourier power spectrum (spectrogram) in two places on the image. The frequencies on the right are higher than those on the left, due to perspective and foreshortening. However, there does not exist a theory that relates texture, shape, and the detailed behavior of local spatial frequency. In this paper, we develop a theory that predicts the systematic frequency shifts due to shape and use the theory in a new shape-from-texture algorithm based on the spectrogram. This has proven to be a simple and intuitive approach to the problem. The method is attractive because it exploits a representation that is useful for understanding other important image phenomena as well.

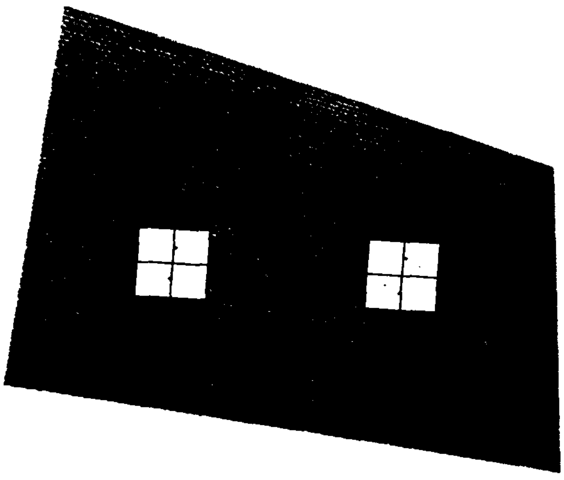

Figure 1: A textured plate with part of its spectrogram superimposed

1.1. The Space/Frequency Representation

The space/frequency representation shows the frequencies of a signal at every point in the signal. Figure 2 shows an example. The one-dimensional function of x consists of a low-frequency sinusoid with a higher-frequency sinusoid replacing the middle. The space/frequency representation, shown on the right, is necessarily a two-dimensional function of x and u, since it must show a one-dimensional frequency distribution for every point in the signal. The frequencies u are shown along the vertical axis. It is like having a little Fourier transform plotted vertically at every point along the x axis. If the original signal were a two-dimensional function of x and y (an image), then the space/frequency representation would be a four-dimensional function of x and y and the two frequencies, u and v.

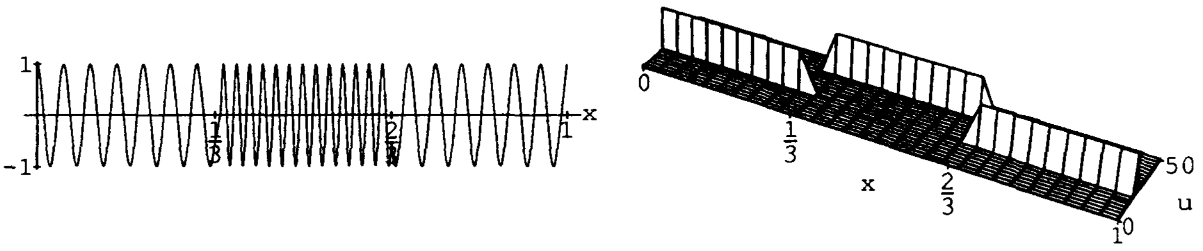

Figure 2: A signal and its space/frequency representation

The space/frequency representation shown in Figure 2 is ideal, and cannot be computed by any commonly used techniques. We use the image spectrogram as our instantiation of the representation. For each point in the image, we extract a square neighborhood of surrounding pixels and multiply this block of intensities by a window function that falls off at block's edges. We compute the two-dimensional Fourier transform of this product and take the squared magnitude as the local frequency representation, giving the local power spectrum. This is the image spectrogram $S(x, y, u, v)$, defined as

$$S(x, y, u, v) = \left| \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(x', y') f(x' - x, y' - y) e^{-j2\pi(ux' + vy')} dx' dy' \right|^2 \tag{1}$$

where $f(x, y)$ is the image and $w(x, y)$ is the window function. This is what we used to compute the two light-colored blocks in Figure 1.

There are several other methods of computing the space/frequency representation. The well-known ones are Gabor functions[12], the Wigner distribution[9], and wavelets[21]. We chose the spectrogram because it gives an intuitive-looking picture, provides a dense sampling in space and frequency, and comes with the well-developed theory of Fourier transforms. The method of computing the representation is really only important at the algorithmic level of our development. The basic theory of projecting frequencies applies regardless of the particular representation.

1.2. Shape from Texture

Notable work in shape-from-texture includes that done by Witkin[27], Blostein and Ahuja[5], Aloimonos[1], Bajcsy and Lieberman[3], Kender[19], Stevens[25], Kanatani and Chou[18], and Blake and Marinos[4]. When Gibson first speculated that humans could infer surface normals based on texture gradients, he assumed that the frontally viewed version of the texture had constant texture density. Most computational shape-from-texture work follows the same paradigm: assume that a certain parameter is uniform when the texture is viewed frontally, model the deformation of this parameter due to the shape of the textured object and camera projection, and then compute the surface normal of the shape by measuring the change of the parameter in the image. In our development, we assume the frequencies of the frontally viewed texture remain the same from point to point -- i.e. that the frontally viewed texture is stationary. The changes in local spatial frequencies on the projected image then give information about the shape of the surface. The resulting algorithm works directly on the spectrogram of the image, requiring no feature detection. This is an important advantage over many other shape-from-texture algorithms, as it is very difficult to reliably find texels in an image. Blake and Marinos said in 1990:

> Our greatest practical problems arise from isolating independent oriented elements from an [texture] image.[4]

And Aloimonos said in 1988:

> There is no known algorithm that can successfully detect texels from a natural image.[1]

Thus it makes sense to develop an algorithm that requires no feature detection. Furthermore, our algorithm does not even depend on weak texture features such as edges. Instead we work with a dense representation of local spatial frequency, allowing us to exploit all the useful data in an image patch. And by keeping a dense representation of the data, we can apply basic theory all through the algorithm, allowing us to easily account for complicated phenomena like aliasing.

Local spatial frequency analysis of texture started with descriptions and segmentation of frontally viewed textures. Such work includes the use of the Fourier transform by Bajcsy[2], Gramenopoulos[15] and Matsuyama *et al.*[22], Gabor filters by Turner[26], Fogel and Sagi[11], and Bovik *et al.*[6] and the Wigner distribution by Reed and Wechsler[24].

Starting with Bajcsy and Lieberman[3], one branch of shape-from-texture research has focused on using local spatial frequencies for the problem. They studied qualitative and quantitative aspects of windowed power spectra of images with receding ground planes. They tracked the peak frequencies from window to window, showing how the gradient of these frequencies qualitatively matched the texture gradient. They stopped short of actually computing surface orientation. Brown and Shvaytser[8] use the autocorrelation of an entire texture image to determine the slant and tilt of the textured surface. Although this is not explicitly a spatial frequency technique, it is close, because the autocorrelation is the Fourier transform of the power spectrum. Jau and Chin[17] use the Wigner distribution and report good results by examining only a scalar measure of the high spatial frequencies. These last two efforts both report good results. Instead of examining aggregate frequency characteristics, our formulation allows us to exploit the shift of *each* frequency component from point to point in the projected texture. This means we can take full advantage of the space/frequency representation and account for other effects like aliasing.

2. Math

This section contains a derivation of the connection between the surface normal of a textured surface and the local Fourier transform of the projected texture in an image. This is important because it relates a physical characteristic of a 3D scene to the measurable behavior of the projected frequencies in an image. We show how the local spatial frequencies in the image are approximately related by an affine transformation to the frontal texture's frequencies. The affine parameters are functions of known camera parameters and the unknown depth and surface normal of the texture. From this we show that the frequencies of two image patches are also related by an affine transform. If we assume the two patches come from the same plane, then the depth variable drops out, leaving the surface normal as the only unknown. We exploit this fact in our shape-from-texture algorithm in Section 3.

2.1. Coordinate Systems

Figure 3 shows the coordinate systems used in the derivation. The camera's pinhole is at the origin of the (X, Y, Z) frame. This serves as the world coordinate system, and points defined in it will be referred to with upper-case (X, Y, Z). The $-Z$ axis is coincident with the camera's optical axis and points into the scene being imaged. The image plane is the (x, y) frame with its origin on the optical axis at a distance d behind the pinhole.

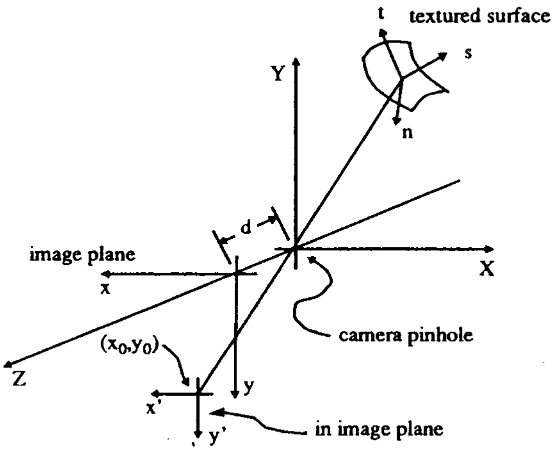

Figure 3: Coordinate systems used in derivation

We imagine that each point on the locally planar textured surface has its own coordinate frame (s, t, n), with the n axis coincident with the surface normal. The surface normal is defined with the gradient space variables (p, q), thus the unit vector along the n axis is $\hat{n} = \frac{1}{r}(p, q, 1)$, with $r = \sqrt{p^2 + q^2 + 1}$, in the world frame. The origin of this surface frame is $(\Delta X, \Delta Y, \Delta Z)$ with respect to the world frame.

The 4x4 homogeneous transformation matrix that locates and orients the surface frame with respect to the world frame is

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} \frac{p^2 + rq^2}{p^2 + q^2} & \frac{pq(1-r)}{p^2 + q^2} & p & r\Delta X \\ \frac{pq(1-r)}{p^2 + q^2} & \frac{rp^2 + q^2}{p^2 + q^2} & q & r\Delta Y \\ -p & -q & 1 & r\Delta Z \\ 0 & 0 & 0 & r \end{bmatrix}. \qquad (2)$$

This was derived by making a single rotation of the (s, t, n) frame around the unit vector $(-q, p, 0) / (\sqrt{p^2 + q^2})$ by an angle ϕ with $\cos\phi = \frac{1}{r}$ and $\sin\phi = \frac{\sqrt{p^2 + q^2}}{r}$.

2.2. Projected Texture

This subsection concludes with an expression for a perspectively projected texture. We begin by assuming the texture on the surface is "painted" on and not a relief pattern. It is locally characterized in the (s, t, n) surface frame as a pattern of surface markings given by $f(s, t)$. Points on this locally planar surface are given by coordinates $(s, t, 0)$. Applying the transformation matrix, the corresponding world coordinates are

$$\begin{aligned} X &= t_{11}s + t_{12}t + \Delta X \\ Y &= t_{21}s + t_{22}t + \Delta Y \\ Z &= t_{31}s + t_{32}t + \Delta Z \end{aligned} \tag{3}$$

Under perspective, these points project to the image plane at

$$\begin{aligned} x &= -d\frac{X}{Z} = -d\frac{t_{11}s + t_{12}t + \Delta X}{t_{31}s + t_{32}t + \Delta Z} \\ y &= -d\frac{Y}{Z} = -d\frac{t_{21}s + t_{22}t + \Delta Y}{t_{31}s + t_{32}t + \Delta Z} \end{aligned} \tag{4}$$

The origin of the (s, t, n) frame thus projects to $(x_0, y_0) = (-d\frac{\Delta X}{\Delta Z}, -d\frac{\Delta Y}{\Delta Z})$ on the image plane. In order to avoid carrying a coordinate offset through the calculations, we define another coordinate system, (x', y'), on the image plane that is centered at (x_0, y_0) with its axes parallel to those of the image plane. Given an (x, y) on the surface,

$$\begin{aligned} x' &= x - x_0 = -d\frac{t_{11}s + t_{12}t + \Delta X}{t_{31}s + t_{32}t + \Delta Z} - x_0 \\ y' &= y - y_0 = -d\frac{t_{21}s + t_{22}t + \Delta Y}{t_{31}s + t_{32}t + \Delta Z} - y_0 \end{aligned} \tag{5}$$

Solving these two equations for (s, t) will give equations that give a point in the surface frame for any corresponding point in the (x', y') frame. Doing so, using $(\Delta X, \Delta Y) = (-\frac{x_0 \Delta Z}{d}, -\frac{y_0 \Delta Z}{d})$ and the orthonormality relationships among the vectors in the transformation matrix, we have

$$
\begin{aligned}
s(x',y') &= -\frac{\Delta Z\,[d\,(y't_{12}-x't_{22})+t_{32}\,(y'x_0-x'y_0)\,]}{d\,[t_{13}\,(x'+x_0)+t_{32}\,(y'+y_0)-d\Delta Z]} \\
t(x',y') &= \frac{\Delta Z\,[d\,(y't_{11}-x't_{21})+t_{31}\,(y'x_0-x'y_0)\,]}{d\,[t_{13}\,(x'+x_0)+t_{32}\,(y'+y_0)-d\Delta Z]}
\end{aligned}
\qquad (6)
$$

Thus, if the brightness pattern on a locally planar patch on a textured surface is $f(s, t)$, then the projected pattern on the image plane is a nonlinear warping of the pattern given by $f(s(x', y'), t(x', y'))$.

2.3. Approximating the Fourier Transform

In order to work with frequencies, we would like to find an expression for the Fourier transform of the projected texture, $f(s(x', y'), t(x', y'))$. But the warpings represented by Equation (6) are too complex to allow us to say anything general. We can make progress by linearizing $s(x', y')$ and $t(x', y')$ using a truncated Taylor series around $(x', y') = (0, 0)$. The approximation is justified since we are only examining a relatively small window of intensities around the point of interest. We have

$$
\begin{aligned}
s(x',y') &\approx s_x x' + s_y y' \\
t(x',y') &\approx t_x x' + t_y y'
\end{aligned}
\qquad (7)
$$

with

$$
\begin{aligned}
s_x &= \frac{\partial}{\partial x'}s(x',y')\Big|_{(x',y')\,=\,(0,0)} = \frac{\Delta Z\,[d\,(rp^2+q^2)-qy_0\,(p^2+q^2)\,]}{d\,(p^2+q^2)\,(px_0+qy_0-d)} \\
s_y &= \frac{\partial}{\partial y'}s(x',y')\Big|_{(x',y')\,=\,(0,0)} = \frac{\Delta Z\,[dpq\,(r-1)+qy_0\,(p^2+q^2)\,]}{d\,(p^2+q^2)\,(px_0+qy_0-d)} \\
t_x &= \frac{\partial}{\partial x'}t(x',y')\Big|_{(x',y')\,=\,(0,0)} = \frac{\Delta Z\,[dpq\,(r-1)+py_0\,(p^2+q^2)\,]}{d\,(p^2+q^2)\,(px_0+qy_0-d)} \\
t_y &= \frac{\partial}{\partial y'}t(x',y')\Big|_{(x',y')\,=\,(0,0)} = \frac{\Delta Z\,[d\,(p^2+rq^2)-py_0\,(p^2+q^2)\,]}{d\,(p^2+q^2)\,(px_0+qy_0-d)}
\end{aligned}
\qquad (8)
$$

where we have substituted the values of t_{ij} from Equation (2).

The projected version of $f(s, t)$ is then approximately $f(s_x x' + s_y y', t_x x' + t_y y')$, which is just an affine transformation (without translation) of the coordinates. A similar relationship holds in the Fourier domain given by the following Fourier transform pairs[13]:

$$\begin{aligned} f(x', y') &\Rightarrow F(u, v) \\ f(s_x x' + s_y y', t_x x' + t_y y') &\Rightarrow \frac{1}{|D|} F\left(\frac{t_y}{D} u - \frac{t_x}{D} v, -\frac{s_y}{D} u + \frac{s_x}{D} v\right) \end{aligned} \tag{9}$$

where $D = s_x t_y - s_y t_x$. Here (u, v) are spatial frequency coordinates in cycles/unit distance, an upper-case function refers to the Fourier transform of the corresponding lower-case function, and the Fourier transform is defined as

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x', y') e^{-j2\pi (ux' + vy')} dx' dy'. \tag{10}$$

The significant conclusion is that the Fourier transform of a perspectively projected texture patch is approximately an affine transformation of the Fourier transform of the frontally viewed texture. The affine transformation parameters are given by the camera focal length, the pixel coordinates of the point of interest, and the depth and orientation of the patch.

2.4. Relation Between Fourier Transforms of Two Patches

Since there is usually no way to determine what the frontally viewed texture looks like, we resort to comparing patches of the same texture at different locations in the image. We showed above that the Fourier transform of each patch is related to the Fourier transform of the frontally viewed texture by an affine transformation. This means that the Fourier transforms of patches themselves are related by affine transformations. We will show that if we assume two patches come from the same plane, then the affine parameters connecting them are functions of known parameters and the plane's surface normal.

Suppose the two patches $f_1(s, t)$ and $f_2(s, t)$ are related to the frontally viewed texture by the affine parameters $(s_{x1}, s_{y1}, t_{x1}, t_{y1})$ and $(s_{x2}, s_{y2}, t_{x2}, t_{y2})$. In Fourier space, an affine transformation of the first into the second means that

$$\frac{1}{|D_1|} F_1\left(a_1\left(\frac{t_{y1}}{D_1} u - \frac{t_{x1}}{D_1} v\right) + b_1\left(-\frac{s_{y1}}{D_1} u + \frac{s_{x1}}{D_1} v\right), a_2\left(\frac{t_{y1}}{D_1} u - \frac{t_{x1}}{D_1} v\right) + b_2\left(-\frac{s_{y1}}{D_1} u + \frac{s_{x1}}{D_1} v\right)\right) = \frac{1}{|D_2|} F_2\left(\frac{t_{y2}}{D_2} u - \frac{t_{x2}}{D_2} v, -\frac{s_{y2}}{D_2} u + \frac{s_{x2}}{D_2} v\right) \tag{11}$$

where $F_1(u, v)$ and $F_2(u, v)$ are the Fourier transforms of the two patches, $D_1 = s_{x1}t_{y1} - s_{y1}t_{x1}$, $D_2 = s_{x2}t_{y2} - s_{y2}t_{x2}$, and (a_1, b_1, a_2, b_2) are the affine transformation parameters connecting the two Fourier transforms. Note that we have ignored phase differences here. In reality, the Fourier phases of the two patches will be different. This difference is masked because each patch is defined with respect to its own local coordinate system. In our formulation, phase would only complicate the derivation, since we discard it by computing the Fourier transform's magnitude in our algorithm.

Equating coefficients on (u, v) in Equation (11) leads to the following linear equation.

$$\frac{1}{D_1}\begin{bmatrix} t_{y1} & -s_{y1} & 0 & 0 \\ -t_{x1} & s_{x1} & 0 & 0 \\ 0 & 0 & t_{y1} & -s_{y1} \\ 0 & 0 & -t_{x1} & s_{x1} \end{bmatrix}\begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix} = \frac{1}{D_2}\begin{bmatrix} t_{y2} \\ -t_{x2} \\ -s_{y2} \\ s_{x2} \end{bmatrix}, \tag{12}$$

whose solution is

$$\begin{bmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{bmatrix} = \frac{1}{D_2}\begin{bmatrix} s_{x1}t_{y2} - s_{y1}t_{x2} \\ t_{x1}t_{y2} - t_{x2}t_{y1} \\ s_{x1}t_{y2} - s_{y1}t_{x2} \\ s_{x2}t_{y1} - s_{y2}t_{x1} \end{bmatrix}. \tag{13}$$

Thus, the affine parameters connecting the two Fourier transforms are functions of the affine parameters connecting the two patches to the frontally viewed texture. In order to relate this equation to the physical parameters of the camera and the textured surface, we take the values of $(s_{x1}, s_{y1}, t_{x1}, t_{y1})$ and $(s_{x2}, s_{y2}, t_{x2}, t_{y2})$ from Equation (8). Before doing this, however, we will make the assumption that the two texture patches have the same surface normal, i.e. $(p_1, q_1) = (p_2, q_2) = (p, q)$, and that both patches are on the same plane, i.e.

$$\frac{\Delta Z_2}{\Delta Z_1} = \frac{d - px_{0_1} - qy_{0_1}}{d - px_{0_2} - qy_{0_2}} \tag{14}$$

Substituting values from Equation (8), the affine parameters connecting the Fourier transforms of the two patches are then

$$
\begin{aligned}
a_1 &= A\left[(-d^2r)\,(p^2+q^2)+dr\,(p^3x_{0_1}+p^2qy_{0_1}+pq^2x_{0_2}+q^3y_{0_2})+dpq\,(q\Delta x_0-p\Delta y_0)-pq\,(p^2+q^2)\,(x_{0_1}y_{0_2}-x_{0_2}y_{0_1})\right]\\
b_1 &= qA\left[(-drp)\,(p\Delta x_0+q\Delta y_0)-dq\,(q\Delta x_0-p\Delta y_0)+q\,(p^2+q^2)\,(x_{0_1}y_{0_2}-x_{0_2}y_{0_1})\right]\\
a_2 &= pA\left[(drq)\,(p\Delta x_0+q\Delta y_0)-dp\,(q\Delta x_0-p\Delta y_0)+p\,(p^2+q^2)\,(x_{0_1}y_{0_2}-x_{0_2}y_{0_1})\right]\\
b_2 &= A\left[(-d^2r)\,(p^2+q^2)+dr\,(p^3x_{0_2}+p^2qy_{0_2}+pq^2x_{0_1}+q^3y_{0_1})-dpq\,(q\Delta x_0-p\Delta y_0)+pq\,(p^2+q^2)\,(x_{0_1}y_{0_2}-x_{0_2}y_{0_1})\right]
\end{aligned}
\tag{15}
$$

where

$$
\begin{aligned}
A &= \frac{px_{0_1}+qy_{0_1}-d}{dr\,(p^2+q^2)\,(px_{0_2}+qy_{0_2}-d)^2}\\
r &= \sqrt{p^2+q^2+1}\\
\Delta x_0 &= x_{0_1}-x_{0_2}\\
\Delta y_0 &= y_{0_1}-y_{0_2}
\end{aligned}
\tag{16}
$$

These equations are not easy to interpret intuitively. The notable feature is that the only unknowns are (p, q). This allows us to use a simple algorithm that determines the correct surface normal by finding which (p, q) generates the affine parameters that best transform one patch into another. In our algorithm we actually use the squared magnitude of the Fourier transform, but the same affine parameters apply.

To summarize this section, we first showed how a locally planar surface patch projects by perspective into the image. Since this projection is complicated, we approximated it with a truncated Taylor series. This gave an affine relationship between the frontally viewed texture and the projected texture. A property of the Fourier transform says that an affine transformation in space is an affine transformation in frequency. Since the Fourier transform of each image patch is related by an affine transformation to the Fourier transform of the frontally viewed texture, the Fourier transforms of the image patches are also related by an affine transformation. If we assume the two patches are on the same plane, the affine parameters that connect their Fourier transforms are functions of known camera parameters and the unknown surface normal.

3. Algorithm

Here we discuss our core shape-from-texture algorithm using the plate in Figure 1 as an example. The five major steps involved in computing a surface normal from an image of a textured surface are

1. Pick two test points on the surface that have the same texture when viewed

frontally.

2. Multiply the neighborhood of each point by a window function.
3. Compute the 2D Fourier transform of each windowed patch.
4. Compute the squared magnitude of each Fourier transform, giving the local power spectrum at each point (part of the spectrogram).
5. Search for the (p,q) that gives the best affine warping from one local power spectrum to the other.

We will consider each of these general steps in this section, and then show results in the next section.

Step 1 requires that we find pairs of test points on the same textured surface. In the future we hope to integrate our algorithm with a segmentation scheme. For now, however, the choice of points must be done manually. Even if the test points are known to be on the same textured surface, their relative location is important. In some situations, the frequency differences on a slanted plate will be too small to accurately determine the surface orientation. For instance, consider a plate rotated slightly around a vertical axis. Any two points in the same column will show hardly any frequency shift, and the algorithm will not give the correct solution. There remains work to be done on assessing the sensitivity of this method to the relative location of test points.

In choosing a window for step 2, one must choose a shape and size. There are many different shapes of windows, and *Numerical Recipes*[23] puts the choice into perspective:

> There is a lot of perhaps unnecessary lore about the choice of a window function, and practically every function which rises from zero to a peak and then falls again has been named after someone...However, at the level of this book, there is effectively *no difference* between any of these (or similar) window functions.

The window function we use happens to be named after two people: the "Blackman-Harris minimum 4-sample" window[16][10]. In two dimensions, its equation is

$$w(l) = w_0 - w_1 \cos\left(\frac{2\pi}{L} l\right) + w_2 \cos\left(\frac{4\pi}{L} l\right) - w_3 \cos\left(\frac{6\pi}{L} l\right) \qquad (17)$$

where L is the radius of the window, $0 \le l \le L$, and $l = \sqrt{x^2 + y^2}$. The coefficients are $(w_0, w_1, w_2, w_3) = (0.35875, 0.48829, 0.14128, 0.01168)$. This function is plotted in Figure 4.

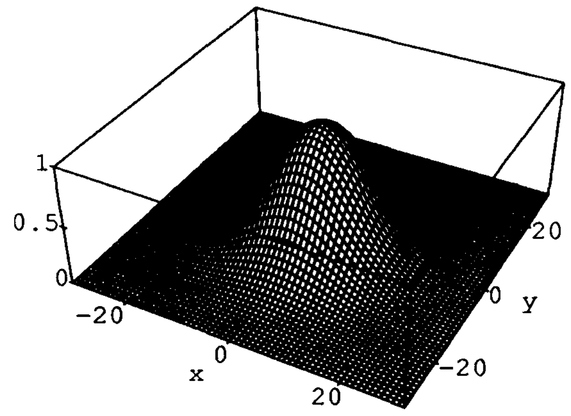

Figure 4: Blackman-Harris minimum 4-sample window function

The choice of the window size is much more important than its exact shape. A smaller window has a smaller chance of overlapping two different texture regions in the image, which would violate one of our assumptions. The frequency of the underlying texture would also change less over the extent of a smaller window. If the frequencies change a lot, the resulting Fourier transform tends to be smeared. On the other hand, smaller windows tend to produce more smearing than larger windows even when the underlying function is stationary or close to stationary. This makes a larger window attractive. In our experiments, we have settled on a window size of 63x63 pixels in images that are typically 512x512 pixels. In Figure 1, the size of the two light-colored squares is equal to the window size.

One alternative is to use the "variable window spectrogram" which we investigated in[20]. In this scheme, the window size varies with the spatial frequency. One reasonable choice is to have the window size be some factor (*e.g.* 5) times the corresponding wavelength of the frequency, which means that we examine the same number of wavelengths at every frequency. This is closer to the idea of using wavelets and Gabor functions for computing the space/frequency representation. The Wigner distribution has the same window dilemma as the spectrogram. In this work, we use a constant sized window to make the Fourier transform computation more efficient. We can justify it physically by noting that the high frequencies we see in textures are usually the higher harmonics of the fundamental texture frequency, meaning that their extent is the same as that of the lower frequencies.

The application of a window is also affected by the randomness of the texture. Theoretically, our method should work for both periodic and random textures. However, when we applied it to a simulated slanted plate with a random fractal texture on it, we found the spectrogram was too noisy for our algorithm. This could be solved by averaging the power spectra from a neighborhood before doing any further computation. However, this involves using more data, which has the same disadvantages as using a large window. We plan to investigate this further.

For computing the Fourier transforms in step 3, we use a 2D FFT routine from the IMSL math library. It can handle arrays whose size is not necessarily an integer power of 2. Before we window the image intensities, we subtract the mean intensity value in the neighborhood to eliminate the d.c. peak in the Fourier transform.

We next compute the squared magnitude (power spectrum) of the Fourier transform. This is shown in the two lighter-colored squares in Figure 1. In using only the squared magnitude, we are ignoring phase information. Phase could be useful for periodic textures in a light-stripping-like algorithm. However, the phase information in a random texture would be useless. In addition, if part of a texture is occluded as in Figure 5, the phase information would be misleading, because the number of wavelength traversed by the texture in the occluded region is unknown.

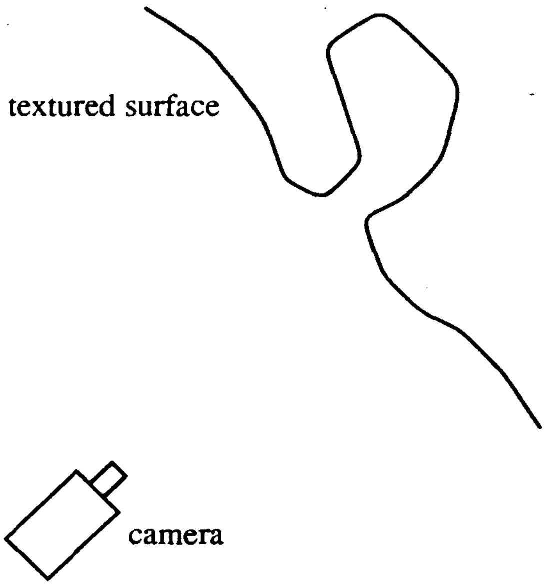

Figure 5: Phase information would be misleading in this case

Because of varying phase, the Fourier transforms at any two general points even on a frontally viewed texture would be different. For the same reason, strictly speaking, Equation (11) would not hold. In order to match the phases of two patches, we would have to use a six-parameter affine transformation (including translation) rather than the four-parameter version (no translation) that we use now. By ignoring phase, we can reduce the complexity of the affine transformation and speed up the program.

The last step of our core algorithm is an exhaustive search for the (p, q) that best transforms the power spectrum of one patch into another. Our current implementation searches over a 61x61 grid, with $(-2, -2) \le (p, q) \le (2, 2)$. This corresponds to a maximum slant of about 63^o. Given a (p, q) to try, we compute the corresponding affine parameters from Equation (15), use these to transform the power spectrum of the first patch using bilinear interpolation, and compute the sum of squared differences (ssd) between the two power spectra. We take the (p, q) that generates the minimum ssd as the solution. The ssd surface from the data in Figure 1 is shown in Figure 6, where we have scaled so the minimum ssd is one.

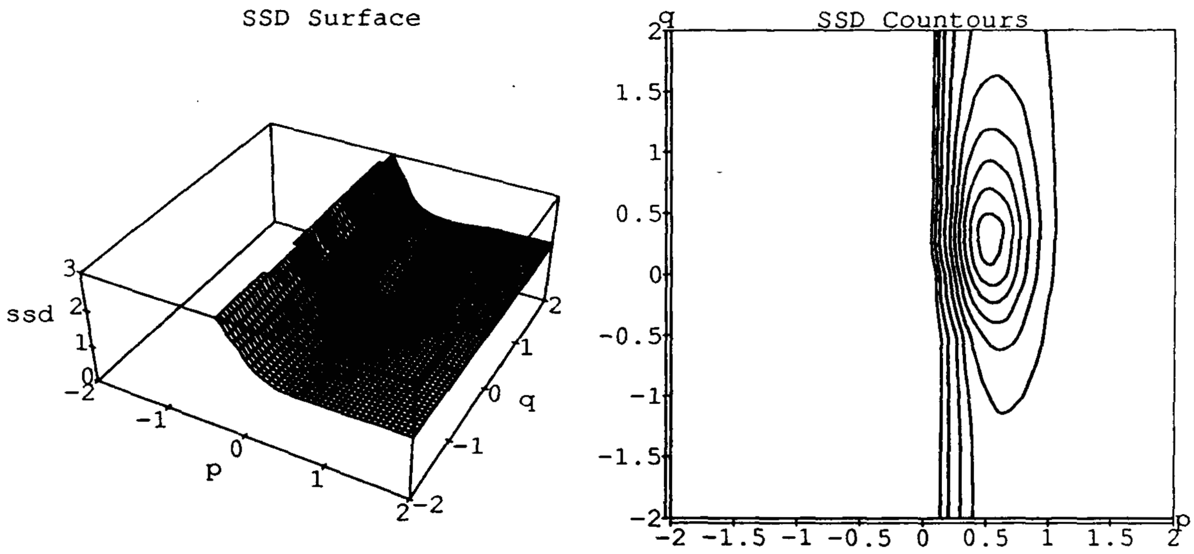

Figure 6: SSD surface and contour plots from comparing patches in Figure 1

This algorithm is better than other shape-from-texture algorithms in several ways. It requires no feature-finding, which is normally an unreliable step. We make no strong assumptions about the frontally-viewed texture, only that it is stationary. Specifically, we do not require that the texture be isotropic. Theoretically, the method should work for both periodic and random textures. We will have to find a better spectral power estimator before we can make it work on random textures, however. Finally, by formulating and solving the problem with the space/frequency representation, we can easily account for other frequency phenomena such as focus and aliasing in the same framework. We show how the method successfully deals with aliasing in the next section.

4. Results

4.1. Flat Plate

In Figure 7 we show four geometrically identical plates with different textures mapped on by a computer graphics program. The surface normal of the plates is $(p, q) = (0.614, 0.364)$. In terms of slant and tilt, $(\sigma, \tau) = (\operatorname{atan}(\sqrt{p^2 + q^2}), \operatorname{atan}(\frac{q}{p})) = (35.5^o, 30.7^o)$ [27]. The first plate has a simple intensity function of two crossed cosines. The rest are Brodatz[7] textures: wire screen (D14), cotton canvas (D77) (same as Figure 1), and straw cloth (D52). Each of the images is shown with two patches removed and replaced by their Fourier power spectra, which are part of the total image spectrogram. These patches are size 63x63.

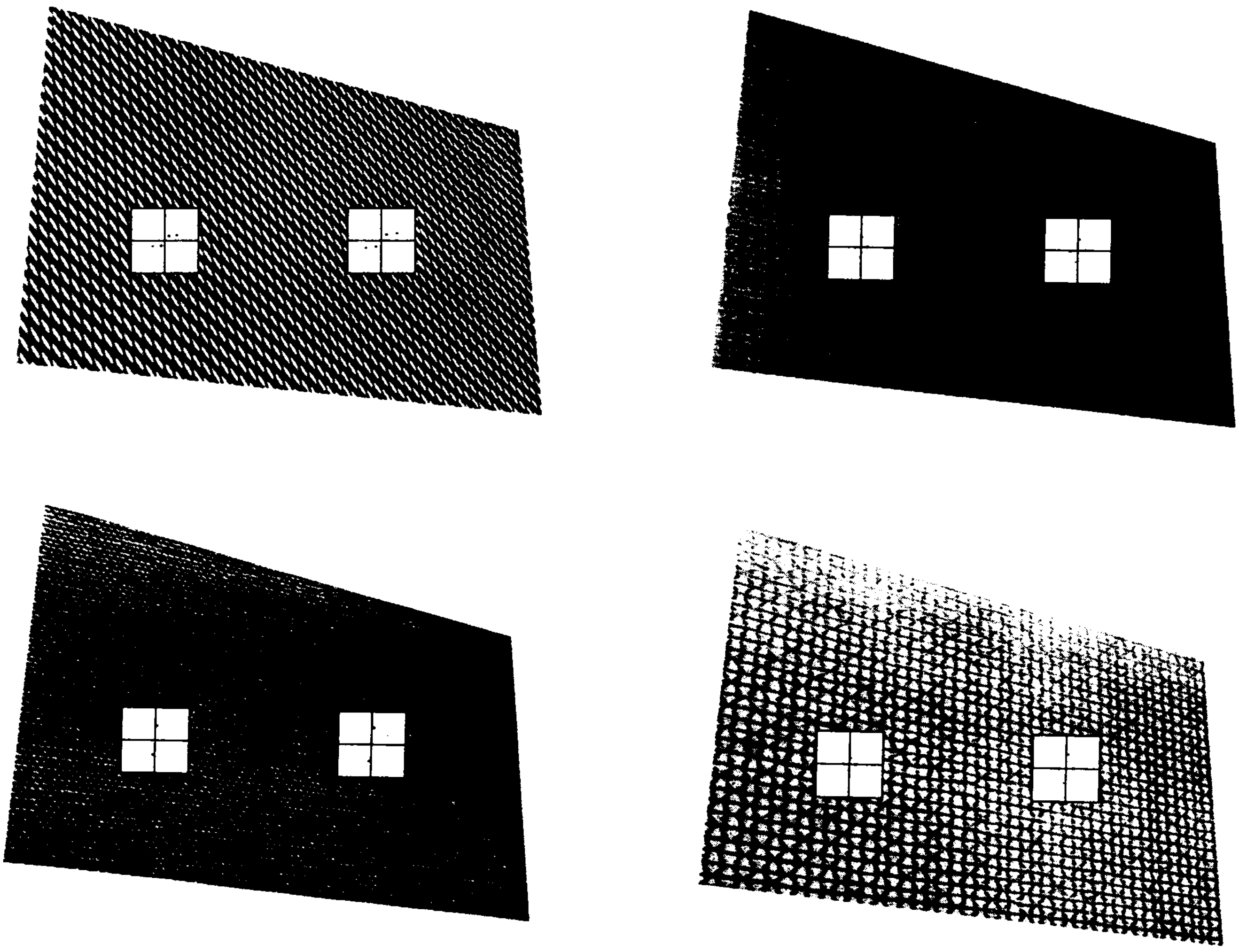

Figure 7: Plates with cosines, screen, canvas and straw cloth textures mapped on

We ran our algorithm on each of these pairs of power spectra, and the results are shown in Table 1. The method works best on textures that are closest to being purely periodic, like the cosine and canvas textures. It loses some accuracy for textures with slightly more random spacing like the wire screen and straw cloth. Considering that the algorithm is only examining data from about 5.5% of the pixels on each textured region, these results are good. Most algorithms for shape-from-texture examine an entire image of a plane covering the whole field of view.

texture	window size	computed (p, q)	equivalent (σ, τ)	error
cosines	63x63	$(0.533, 0.400)$	$(33.7^o, 36.9^o)$	4.0^o
wire screen	63x63	$(0.400, 0.200)$	$(24.1^o, 26.6^o)$	11.6^o
cotton canvas	63x63	$(0.600, 0.333)$	$(34.5^o, 29.1^o)$	1.4^o
straw cloth	63x63	$(0.400, 0.400)$	$(29.5^o, 45.0^o)$	9.7^o

Table 1: Results of algorithm on textured plates

texture	window size	computed (p, q)	equivalent (σ, τ)	error
cosines	81x81	$(0.600, 0.333)$	$(34.5^o, 29.1^o)$	1.4^o
wire screen	121x121	$(0.577, 0.295)$	$(32.9^o, 27.1^o)$	3.3^o
cotton canvas	101x101	$(0.600, 0.333)$	$(34.5^o, 29.1^o)$	1.4^o
straw cloth	121x121	$(0.600, 0.333)$	$(34.5^o, 29.1^o)$	1.4^o

Table 2: Results of algorithm on textured plates with best window size

We investigated the effect of window size by running our program on the same four textures with different window dimensions. The results are show in Figure 8. The abscissa is the length of a side of the square window in pixels. The ordinate shows the surface normal error in degrees. For all four textures, an undersized window causes inaccuracy. This is probably because the window does not contain enough wavelengths of the texture to allow a Fourier transform of adequate resolution. For these textures, window sizes between 50 and 100 seem best. Beyond 100, the error for the canvas texture increases sharply. Although we expect an oversize window to degrade performance because of increasing non-stationarity, the other textures exhibit this tendency only slightly, if at all. Using the data in these plots, if we manually tailor the window size to the particular image, we get the smaller errors shown in Table 1. There remains work to be done on window size considerations. The choice of window size is fairly arbitrary for almost all shape-from-texture algorithms that require it.

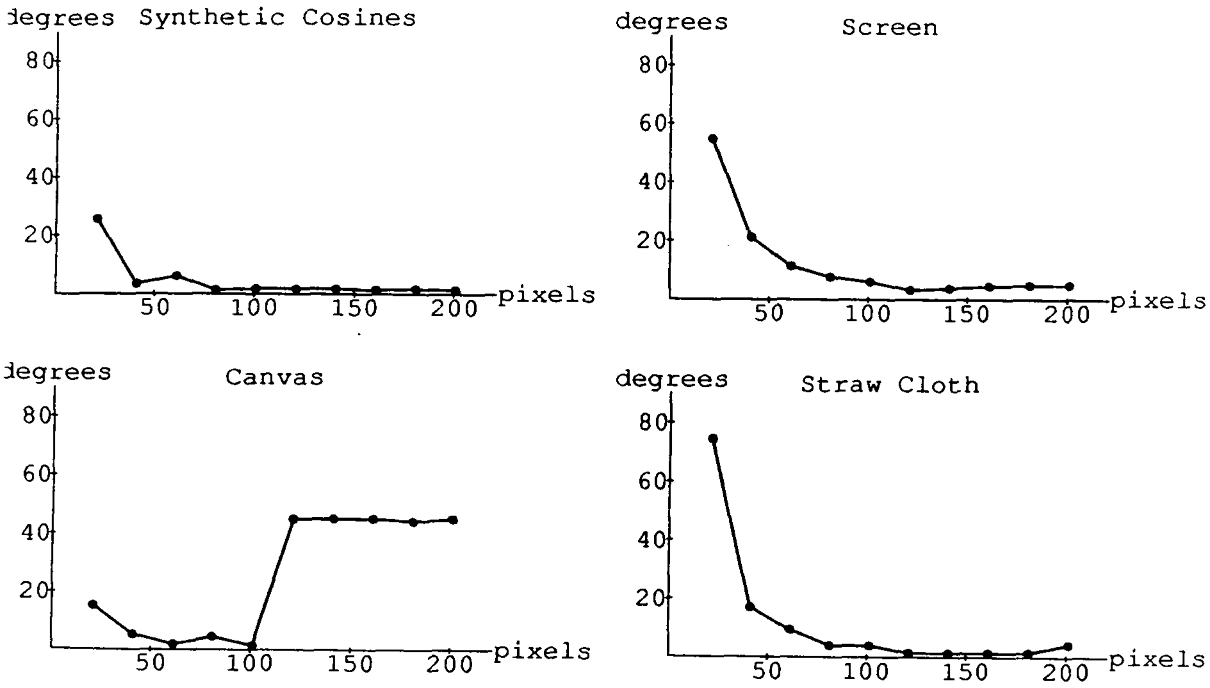

Figure 8: Angle error vs window size for four textures in Figure 7

4.2. Aliasing

Although aliasing can cause real problems in image understanding, it is rarely dealt with explicitly in machine vision algorithms. Aliasing occurs when the image projected on the sampling grid has spatial frequencies that are higher than half the spatial sampling rate. If the aliased pattern is periodic, moire patterns appear. This is shown in Figure 9, which is geometrically the same plate as before, this time with two different, higher frequency cosines painted on. The cosines run at $\pm 45^o$ from the horizontal. The left side of the plate is not aliased, while the right side is, because the projected frequencies have grown beyond half the sampling rate. The series of local power spectra across the center of the image show what happens to the frequencies. As the peaks move out from the center, they approach the edges of the squares. The squares' edges are at half the sampling rate, and thus represent the highest frequencies that can be successfully sampled. The peaks in the first and third quadrants hit the edges in the fourth square from the left. In the next square to the right, they reappear in the second and fourth quadrants along with the peaks that were already there. This is the onset of aliasing. In the last square the aliased peaks have moved a little more back into the square.

If the sampling rates in the x and y directions are u_s and v_s respectively, then any (u, v) outside the boundaries $(\frac{\pm u_s}{2}, \frac{\pm v_s}{2})$ will be aliased. It can be shown that the aliased frequency will be given by

$$(u_{aliased}, v_{aliased}) = \left(\frac{u_s}{2} saw_{u_s}(u), \frac{v_s}{2} saw_{v_s}(v)\right) \tag{18}$$

where

$$saw_T(x) = \frac{2}{T}\left(x - T\left\lfloor \frac{x + \frac{T}{2}}{T} \right\rfloor\right) \tag{19}$$

with $\lfloor x \rfloor$ being the "floor" function, returning the largest integer not exceeding x. The function $saw_T(x)$ has a period of T. We show a plot of $u_{aliased}$ as a function of u in Figure 10. It shows how the unalised frequency rises and then reappears at a different frequency when aliasing occurs.

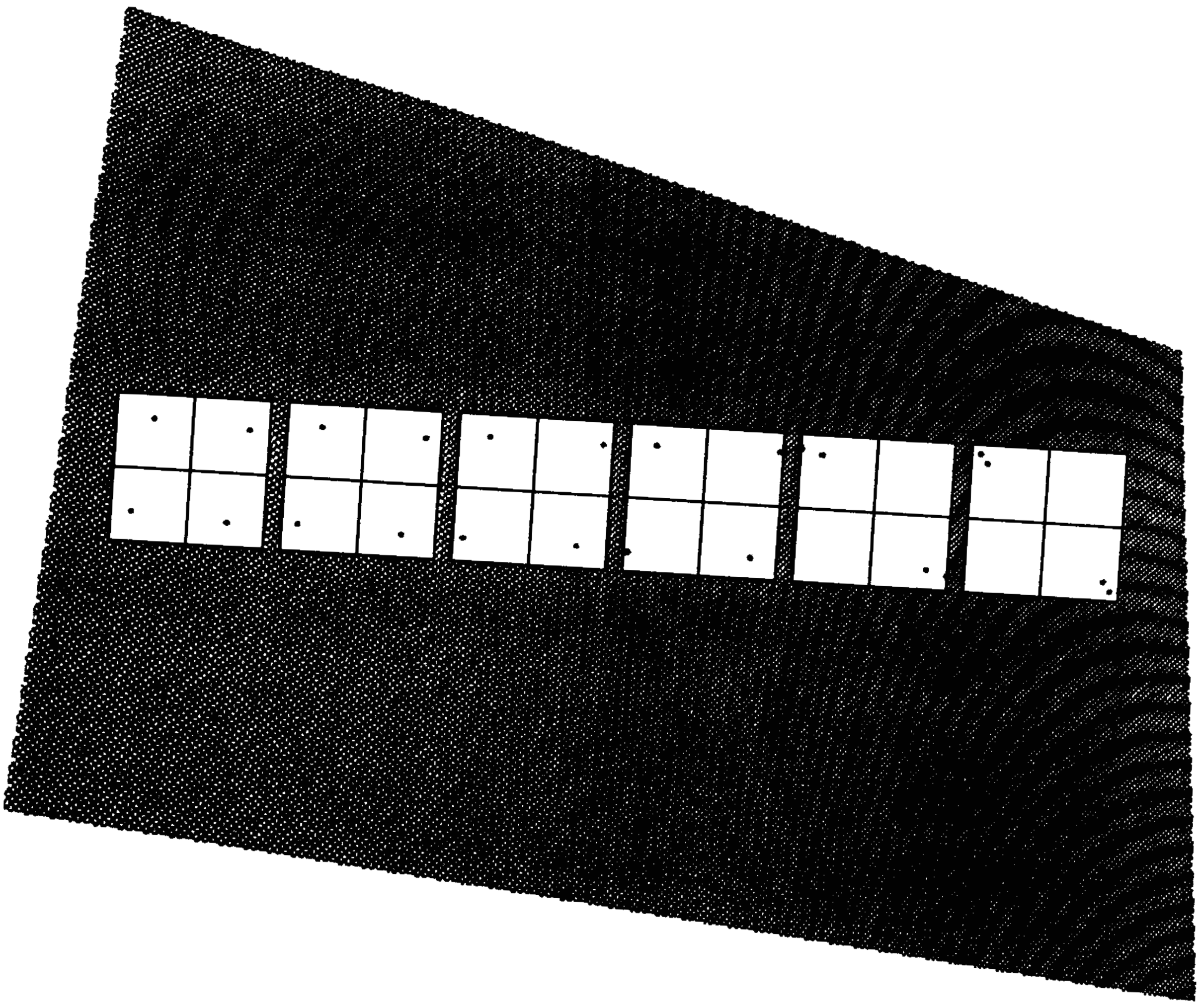

Figure 9: Plate showing aliasing on the right

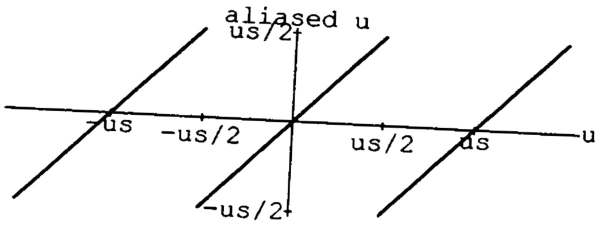

Figure 10: $u_{aliased} = \frac{u_s}{2} saw_{u_s}(u)$

Our algorithm allows us to account for aliasing very easily. When we test a given (p, q), we warp the frequency coordinates in one power spectrum by an affine transformation. We simply put all the transformed (u, v) 's through Equation (18) to adjust them for aliasing. This way, if a given (p, q) causes frequencies to be transformed outside the half-sampling-frequency limits, they will be aliased back in at the proper coordinates. This is also a convenient way of making sure both frequency patches overlap exactly, instead of having one skewed off the other with no corresponding frequencies in the other patch after the affine transformation.

We ran our algorithm on the left and right patches in Figure 9 and got $(p, q) = (0.667, 0.467)$ with a window size of 63x63. This is an error of about 4.5^o, so the method successfully accounts for aliasing. There are two restrictions. First, it is assumed that the first patch is not aliased. Second, we cannot yet account for the fact that aliased frequencies actually sum with nonaliased frequencies. We hope to remove this second restriction in the future.

We know of no other shape-from-texture algorithm that can account for aliasing even in this simple case. We attribute the ability to the fact that the space/frequency representation preserves essentially all the data in the original signal and that frequency is the natural domain for the analysis of aliasing.

5. Conclusion

We have advocated the use of the space/frequency representation, which shows an image's spatial and local spatial frequency characteristics simultaneously. One natural application for such a representation is the shape-from-texture problem. If we assume that the frontally viewed texture is stationary, we can expect to see systematic changes in frequency from point to point due to shape and perspective projection. We developed a new theory that predicts the detailed behavior of spatial frequencies in the image of a projected surface. Because it makes predictions at a low level, this theory can be applied to any space/frequency representation of an image. Using this math, we developed an algorithm based on the spectrogram that successfully finds surface normals of textured surfaces by searching through gradient space. The algorithm requires no feature-finding, working instead on a low-level representation that is still convenient for analysis. Because the representation is low-level, it should support other kinds of image analysis as well. For instance, the algorithm can easily handle simple cases of aliasing.

References

[1] Aloimonos, J. "Shape from Texture." *Biological Cybernetics* 58 (1988): 345-360.

[2] Bajcsy, Ruzena. "Computer Description of Textured Surfaces." In *Third International Joint Conference on Artificial Intelligence*, 572-577, 1973.

[3] Bajcsy, Ruzena and Lawrence Lieberman. "Texture Gradient as a Depth Cue." *Computer Graphics and Image Processing* 5 (1976): 52-67.

[4] Blake, Andrew and Constantinos Marinos. "Shape from Texture: Estimation, Isotropy and Moments." *Artificial Intelligence* 45 (1990): 323-380.

[5] Blostein, Dorothea and Narendra Ahuja. "Shape from Texture: Integrating Texture-Element Extraction and Surface Estimation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (December 1989): 1233-1251.

[6] Bovik, Alan Conrad, Marianna Clark, and Wilson S. Geisler. "Multichannel Texture Analysis Using Localized Spatial Filters." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12, (January 1990): 55-73.

[7] Brodatz, Phil. *Textures: A Photographic Album for Artists and Designers*, New York: Dover, 1966.

[8] Brown, Lisa Gottesfeld and Haim Shvaytser. "Surface Orientation from Projective Foreshortening of Isotropic Texture Autocorrelation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (June 1990): 584-588.

[9] Claasen, T. A. C. M. and W. F. G. Mecklenbrauker. "The Wigner Distribution -- A Tool for Time-Frequency Signal Analysis, Part I: Continuous-Time Signals." *Philips Journal of Research* 35 (1980): 217-250.

[10] DeFatta, David J., Joseph G. Lucas, and William S. Hodgkiss. *Digital Signal Processing: A System Design Approach.* New York: John Wiley & Sons, 1988, p. 270.

[11] Fogel, I. and D. Sagi. "Gabor Filters as Texture Discriminator." *Biological Cybernetics* 61 (June 1989): 103-113.

[12] Gabor, D. "Theory of Communication." *The Journal of the Institute of Electrical Engineers*, Part III, 93 (January 1946): 429-457.

[13] Gaskill, Jack D. *Linear Systems, Fourier Transforms, and Optics*. New York: John Wiley & Sons, 1978.

[14] Gibson, James T. "The Perception of Visual Surfaces." *The American Journal of Psychology* 63 (July 1950): 367-384.

[15] Gramenopoulos, Nicholas. "Terrain Type Recognition Using ERTS-1 MSS Images." In *Symposium on Significant Results Obtained from the Earth Resources Technology Satellite-1*, Vol. 1, *Technical Presentations*, Section B, 1229-1241, 1973.

[16] Harris, Fredric, J. "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform." *Proceedings of the IEEE* 66 (January 1978): 51-83.

[17] Jau, Jack Y. and Roland T. Chin. "Shape from Texture Using the Wigner Distribution." *Computer Vision, Graphics, and Image Processing* 52 (1990): 248-263.

[18] Kanatani, Ken-ichi and Tsai-Chia Chou. "Shape from Texture: General Principle." *Artificial Intelligence* 38 (1989): 1-48.

[19] Kender, John R. "Shape from Texture." Carnegie Mellon University Computer Science Technical Report No. CMU-CS-81-102, November 1980.

[20] Krumm, John and Steven A. Shafer. "Local Spatial Frequency Analysis for Computer Vision." Carnegie Mellon Robotics Institute Technical Report CMU-RI-TR-90-11, May 1990.

[21] Mallat, Stephane G. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (July 1989): 674-693.

[22] Matsuyama, Takashi, Shu-Ichi Miura, and Makoto Nagao. "Structural Analysis of Natural Textures by Fourier Transformation." *Computer Vision, Graphics and Image Processing* 24 (1983): 347-362.

[23] Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1986.

[24] Reed, Todd R. and Harry Wechsler. "Segmentation of Textured Images and Gestalt Organization Using Spatial/Spatial-Frequency Representations." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (January 1990): 1-12.

[25] Stevens, Kent A. "The Information Content of Texture Gradients." *Biological Cybernetics* 42 (1981): 95-105.

[26] Turner, M.R. "Texture Discrimination by Gabor Functions." *Biological Cybernetics* 55, (October 1986): 71-82.

[27] Witkin, Andrew P. "Recovering Surface Shape and Orientation from Texture." *Artificial Intelligence* 17 (1981): 17-45.