THE RECOVERY OF 3-D STRUCTURE USING VISUAL TEXTURE PATTERNS

This thesis is

presented to the

School of Computer Science and Software Engineering

for the degree of

Doctor of Philosophy

of

The University of Western Australia

By

Angeline M. Loh BSc BE(Hons)

February 2006

Declaration

This thesis is the author's own composition. All sources have been acknowledged and the author's contribution is clearly identified in this thesis. For any work in this thesis that has been co-published with other authors, the permission of all co-authors has been obtained to include the work in this thesis.

This thesis was completed during the course of enrollment in this degree at the University of Western Australia and has not previously been accepted for a degree at this or another institution.

Author	Date
Principal Supervisor	Date

Abstract

One common task in Computer Vision is the estimation of three-dimensional surface shape from two-dimensional images. This task is important as a precursor to higher level tasks such as object recognition — since the shape of an object gives clues to what the object is — and object modeling for graphics. Many visual cues have been suggested in the literature to provide shape information, including the shading of an object, its occluding contours (the outline of the object that slants away from the viewer) and its appearance from two or more views. If the image exhibits a significant amount of texture, then this too may be used as a shape cue. Here, 'texture' is taken to mean the pattern on the surface of the object, such as the dots on a pear, or the tartan pattern on a tablecloth. This problem of estimating the shape of an object based on its texture is referred to as shape-from-texture and it is the subject of this thesis.

One motivation for studying shape-from-texture is the fact that, according to psychophysical experiments, texture plays an important role in the human perception of shape. It would be useful if computers could mimic this behaviour. Some advantages of using texture as a cue are that it allows shape to be recovered from static monocular images (rather than multiple views) and the fact that textures are ubiquitous in the world around us. Another reason for studying texture as a shape cue is to use it in combination with other cues for a more robust solution.

During the past three decades, there has been much work in shape-from-texture. This thesis contributes to the existing body of work by providing three new algorithms: two are shape-from-texture algorithms that solve the problem under different sets of assumptions regarding the texture and viewing geometry; the other algorithm solves the low-level task of estimating the transformation between patches

of texture. These three algorithms are described in more detail below.

The first shape-from-texture method is fast and direct, and works with homogeneous and stationary textures viewed orthographically, with the frontal texture known. The method is based on the fact that as textures are foreshortened due to the relative orientation of the surface patch to the viewer, the second spectral moments do not change about the axis orthogonal to the tilt axis. From this, the tilt axis may be identified, which in turn leads to an estimation of the slant angle of the surface patch. In this way the orientation of each surface patch may be solved. A number of issues affecting the ideal behaviour of the system are explored, including the scaling property of the Fourier transform, windowing schemes, illumination and blur. A property of the method is that occluding boundaries are estimated to curve away from the viewer. The new method is compared to a recent and well-known method from the literature that uses the same assumptions regarding texture and viewing geometry. This thesis demonstrates that the new method has many advantages over the other existing method; among other things it is more robust, does not exhibit any ambiguities other than the unavoidable tilt ambiguity of $\pm\pi$, and never returns complex, and hence unusable, values.

The second shape-from-texture method aims to solve the problem in one of its most general forms; the texture is not assumed to be isotropic, homogeneous, stationary or viewed orthographically. In addition, the frontal texture is not assumed to be known a priori, or from a known set, or even present in the image. Instead it is assumed that the surface is smooth and covered in identical texture elements; this assumption allows the entire surface to be recovered via a consistency constraint. The key idea is that if the correct transformation from an arbitrary reference texel to a frontal texel can be estimated only then will a consistent, integrable surface be produced. It is shown that a Levenberg-Marquardt search can estimate the frontal texture efficiently. The methods described in this thesis have been quantitatively tested on the entire set of Brodatz textures and also on real images.

This thesis also investigates the relationship between shape-from-texture and structure-from-motion. If the camera is stationary and the moving object is planar, or nearly so, then the superposition of the images of the moving object produce

a texture, the structure of which can be solved. The second shape-from-texture algorithm was adapted to demonstrate an example of such a structure-from-motion reconstruction.

The other algorithm that is presented in this thesis estimates the transformation between patches of the same texture, viewed from different orientations. Previous methods for doing this are shown to be non-robust or have limitations if the change between the two texture patches is not incremental. The new method overcomes the drawbacks of these previous methods, as well as being robust to blurring and illumination variations.

The work in this thesis is likely to impact in a number of ways. The second shape-from-texture algorithm provides one of the most general solutions to the problem. On the other hand, if the assumptions of the first shape-from-texture algorithm are met, this algorithm provides an extremely usable method, in that users should be able to input images of textured objects and click on the frontal texture to quickly reconstruct a fairly good estimation of the surface. And lastly, the algorithm for estimating the transformation between textures can be used as a part of many shape-from-texture algorithms, as well as being useful in other areas of Computer Vision. This thesis gives two examples of other applications for the method: re-texturing an object and placing objects in a scene.

Preface

Some of the research presented in this thesis has been published previously.

An early version of the new method for the orthographic camera model (Chapter 3) has been published in the proceedings of *Digital Image Computing: Techniques and Applications* [56]. An extended technical report on a later version of the algorithm is available in the School of Computer Science Technical Report Series [54]. Peter Kovesi contributed some of the theory regarding factors such as blur and illumination.

The contents of Chapter 5 on the new method for the perspective camera model have been published in the proceedings of the *British Machine Vision Conference* [55]. Richard Hartley originally suggested that the assumption of surface smoothness, together with some consistency constraint could be used to solve shape-from-texture for perspective views of a non-homogeneous, anisotropic and non-stationary texture.

The work on the estimation of affine transformations between textures (Chapter 4) has been published in the proceedings of *Digital Image Computing: Techniques and Applications* [57]. This paper won best student paper at the conference. The original idea was jointly produced by Andrew Zisserman and myself.

Unless otherwise stated, all the work presented in this thesis is entirely my own.

Acknowledgments

I feel lucky to have had supervisors as wise and amiable as Peter Kovesi and Robyn Owens. Peter has been a dependable source of Computer Vision knowledge, and his dedication to my work over the years has been gratefully received. His technical help, approachable manner, and sense of direction have really been the driving force behind getting this thesis finished. In spite of her very busy schedule, Robyn always provided timely and thoughtful feedback for all my papers and my thesis. I am thankful for the improvements she has made to my work, and for her advice and encouragement.

Some of this work was completed at the Australian National University (ANU) in Canberra, after I was invited there by Richard Hartley in February 2004. Richard provided a welcoming and intellectually stimulating environment in which to produce ideas and work, as well as providing some of the ideas themselves. Fredrik Kahl, who was also at ANU at that time, supplied encouragement and knowledge of the finer workings of MATLAB. In February the following year I visited Andrew Zisserman at the University of Oxford, where most of the work for Chapter 4 was produced. The final part of that work was completed later that year in August at ANU when Andrew and I were both visiting Richard Hartley. I am grateful for the very useful discussions with Andrew during these visits.

My postgraduate years have been enjoyable and for a large part this has been due to the fun people with whom I have shared an office. I am grateful to Mukta Prasad and Oliver Woodford at Oxford and Nick Barnes at ANU for their good company. In my permanent office at UWA, lively and interesting people have come and gone over time; these include Jonathan Knispel, Sanandanan Somasekaran, Courtenay Greig and Felix Margadant. I am particularly obliged to Anthony (Tone)

Prior, one of the students who, along with me, moved into the Interactive Virtual Environments Centre (IVEC) when it was created. Over the years, Tone and I have seen people move on, until numbers eventually dwindled down to only the two of us. I am thankful for having Tone's good company and am sorry to leave him as the only remaining student.

There are many other people to whom I am grateful for helping me along the way. Karen Haines gave me a beautiful office environment in which to work, as well as advice and encouragement. Scott Martyn, and Michael and Alison Kruger assisted me with an easy transition to living in the UK. Helene Fung, Tim Hillman and Adam Dunn contributed their proof-reading talents.

My Mum and Dad continue to be as they have always been: loving, and supportive of my work.

This work was funded by an Australian Postgraduate Award (APA) and a Western Australian IVEC Doctoral Scholarships (WAIDS) top-up. The visits to ANU were financially supported by the National ICT Australia, which is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council. The visit to Oxford was made possible through a BankWest Travel Award, jointly awarded by the UWA Postgraduates Association.

Lastly, I am indebted to Christopher Kings-Lynne who has whole-heartedly supported me throughout my years as a postgraduate student. His enthusiasm to act as a sounding-board was gratefully called upon, as well as the (sometimes well-) out-of-the-box ideas he has thrown my way. I am also thankful to Christopher for the deliveries of ice cream and other unhealthy refreshments that have found their way to my laboratory when I have worked late into the night.

Contents

Declaration iii

Abstract v

Preface ix

Acknowledgments x

1 Introduction 1

2 Overview of Shape-from-Texture 5

2.1 Shape 5

2.1.1 Orientation 5

2.1.2 Producing surface shape from normals 6

2.1.3 Distance 7

2.2 Texture 8

2.3 Assumptions 9

2.4 Projection geometry 11

2.5 Previous work 13

2.6 Analogy with other studies 17

3 A Method for Orthographic Views 19

3.1 Description of new algorithm 20

3.1.1 Theoretical behavior 20

3.1.2 A simple planewave example 22

3.1.3 Accounting for real factors 26

3.1.4 Finding the frontal texture . . . 27
3.1.5 Choosing the window size . . . 28
3.1.6 Summary of algorithm . . . 29
3.1.7 Interpretation using representative ellipses . . . 30
3.2 Results . . . 31
3.2.1 Tests using the Brodatz texture database . . . 31
3.2.2 Tests using images of real objects . . . 36
3.3 Comparison with Super and Bovik's method . . . 41
3.3.1 Overview of Super and Bovik's method . . . 43
3.3.2 Super and Bovik's results . . . 47
3.3.3 Discussion . . . 54
3.4 Conclusions . . . 55

4 Finding The Affine Transformation **57**
4.1 Outline of problem . . . 58
4.2 Relevant work . . . 59
4.3 The method . . . 60
4.3.1 Finding the rotation . . . 62
4.3.2 Finding the scale . . . 63
4.3.3 Summary of algorithm . . . 64
4.3.4 Details of implementation . . . 64
4.4 Results . . . 67
4.4.1 Tests with Brodatz textures . . . 67
4.4.2 Tests with real images . . . 68
4.4.3 Simple applications of the method . . . 70
4.5 Discussion and conclusions . . . 74

5 A General Shape-from-Texture Method **75**
5.1 Introduction . . . 75
5.1.1 Basic idea . . . 75
5.2 Overview of the new method . . . 76
5.2.1 Searching for the frontal texel . . . 77

5.2.2 Surface consistency measure 78
5.2.3 Estimating shape using the frontal texel 80
5.2.4 Finding texels 81
5.3 Results 81
5.3.1 First synthetic example 82
5.3.2 Second synthetic example 86
5.3.3 First test on a real image 91
5.3.4 Another real image 93
5.3.5 A structure-from-motion experiment 100
5.3.6 Discussion and conclusions 103

6 Conclusions **107**
6.1 Contributions 107
6.2 Summary of work 108
6.3 Directions for future work 109

Bibliography **111**

A The Brodatz Textures **121**

Chapter 1

Introduction

Electronics give machines the ability to capture information from the visible part of the electromagnetic spectrum, an ability that humans already possess. However simply capturing this information is not the same as understanding it. A digital camera can no better differentiate between the various subjects it is aimed at, than a tape recorder can differentiate between the works of Sibelius and Black Sabbath. In comparison, for most humans the understanding of visual input is so fundamental as to seem instantaneous. We barely devote a conscious thought to the task of making sense of the constant stream of pictures that flash before our eyes. Our understanding of this data allows us to perform vastly complicated tasks, such as the recognition of objects and navigation in almost all aspects of our daily lives. Just like humans, the more computers are able to understand their visual input, the more complex and useful the tasks they can perform. This is the motivation behind Computer Vision, the study of making computers "understand" what they are seeing.

The existence of Computer Vision is due to the fact that the world is composed of three-dimensional objects, but the information captured by a camera is two-dimensional, so processing is required to understand these images in the context of the world. This task was, in the sixties, perceived by Artificial Intelligence experts as being no more difficult than a summer student's project [22]. Decades of Computer Vision research has proven that the problem is far more difficult than at first thought. This points toward the fact that the human brain does a significant

amount of processing, using many visual cues, to arrive at our interpretation of the world.

In Computer Vision language, the word 'understanding' is taken to mean the extraction of a certain piece of information. Some of the most important pieces of information that may be derived from visual data are the position, orientation, size or shape of the objects in an image. The study of extracting shape information has itself evolved into branches of Computer Vision known as shape-from-X, where 'X' is replaced with a visual cue that provides some shape information. These cues include shading, contours, multiple views, shadows, and texture. The latter cue, *texture*, refers in this context to the visual pattern that lies on the surface of the object, such as the dots on a pear or the tartan pattern on a tablecloth. An example of the application of shape-from-texture is given below.

Although the image in Figure 1 is nothing more than a two-dimensional image, it is perceived by most people as a three-dimensional cylinder. The aim of my research

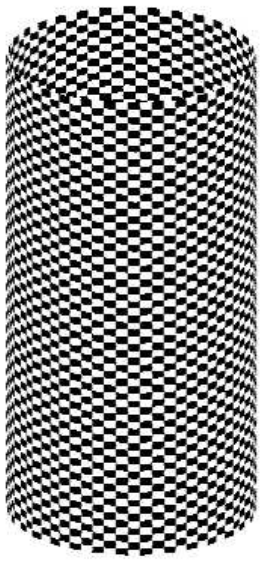


Figure 1: An image perceived by most people as a hollow cylinder covered in a checkered pattern.

has been to create algorithms that allow a computer to arrive at the same conclusion. This aim is fulfilled by interpreting the apparent visual distortion of the texture.

For example, the checkered pattern in Figure 1 appears to compress towards the left and right edges of the cylinder, giving the appearance that the surface is curving away from the viewer. By analysing the mathematical relationship between an object's shape and this form of distortion (as well as other forms of distortion), one may find shape-from-texture solutions.

A lot of work has been devoted to other forms of shape-from-X. This includes the studies in shape-from-shading [26, 28, 6, 27, 45], shape-from-multiple-views [65, 4, 76], shape-from-contours [40, 59, 82] and shape-from-shadows [13, 24, 38, 87]. Unlike these studies, there are surprisingly few shape-from-texture methods [15], and this is one motivation for its continuing study. There are also many other reasons for studying shape-from-texture. Textures provide shape information from static monocular images, rather than multiple views. Psychophysical experiments have demonstrated that texture plays an important role in the human perception of shape [8, 12, 81]. It is a reasonable approach to use the same cues as people use, to achieve automated shape estimation. Textures are ubiquitous in the world around us, providing a good source of shape information. Some common textures are shown in Figure 2.

Figure 2: Textures are ubiquitous in the world around us. They occur in nature, as shown by examples in the top row, or can be manufactured, as shown by examples in the bottom row.

For a machine to robustly estimate the shape of objects in the real world, it must, like the human visual system, utilise many different shape cues [11]. I have studied shape-from-texture in the hope that one day my work may become part of

a larger system that combines many shape cues. It is my hope that machines that can perceive the shape of objects will then use this ability to perform very useful tasks; I imagine that one day robots will be able to identify and retrieve objects, move about autonomously to transport goods, and accomplish other tasks that will help us with our daily lives.

Chapter 2

Overview of Shape-from-Texture

2.1 Shape

The aim of my research has been to automatically estimate the shape of objects. A coordinate frame is placed on an object and its shape is given by the three-dimensional coordinates of points on its surface. No universal coordinate frame is used and therefore the *pose* of the object is not calculated.

In the work described in this thesis, shape estimation generally occurs in two parts: firstly, the orientation is calculated at points on the object's surface, giving the surface normal directions. This has been the focus of my work. Secondly, the surface is reconstructed using the surface normals. For this task, a number of algorithms may be used; these are discussed in Section 2.1.2.

2.1.1 Orientation

The **orientation** at a point on a surface may be specified by the **slant** σ and **tilt** τ of the surface. Slant is defined as the angle between the surface normal and the optic axis (see Figure 3). Tilt is defined as the angle between the projection of the surface normal onto the image plane and the x-axis of the image. The more an image is slanted, the more it visually appears to be compressed. Figure 4(a) shows an image viewed frontally i.e. such that its surface normal points directly out of the page. If the image were slanted in three-dimensional space it might look like

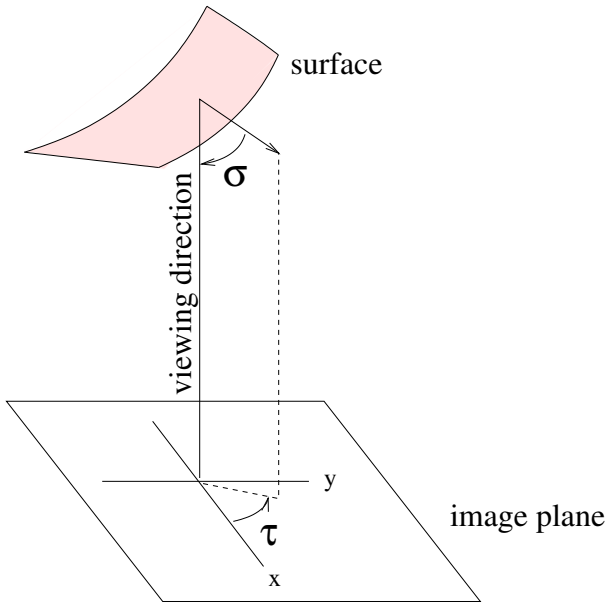


Figure 3: Depiction of slant and tilt angles (adapted from [19]).

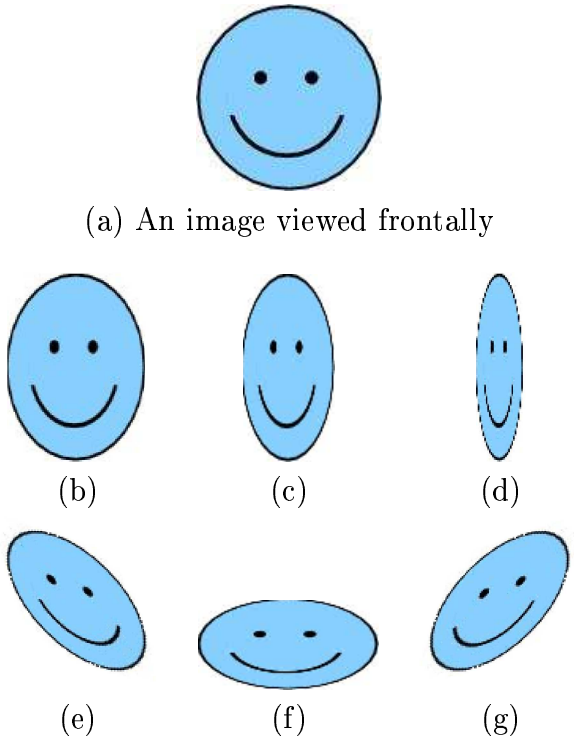
the images in Figures 4(b)–(d), where (b) has the slightest slant angle of the three images, and (d) has the greatest slant angle. The tilt angle is the angle in which an image visually appears to compress. Figures 4(e)–(g) show the image when the slant angle remains constant, but the tilt angle varies.

2.1.2 Producing surface shape from normals

Once the slant and tilt have been determined for points on the object's surface the shape of the surface may be reconstructed. The literature describes several ways to do this.

In the work described in this thesis, surface shapes have been reconstructed using the method by Kovesi [44]. This method correlates the surface gradients with the gradients of some basis functions. The correlations are summed to give the surface reconstruction. The method works because the correlation between the gradient of a basis function with the surface gradients gives the same information (up to scale) as the correlation between the surface and the basis functions themselves. The method is robust to noise and has the advantage of having the code readily available [42].

There are other techniques which may have been used instead of Kovesi's method.



(a) An image viewed frontally

(b) (c) (d)

(e) (f) (g)

Figure 4: Depiction of slant and tilt. Increasing slant is shown in (b)–(d) while tilt remains at 0. Various tilt angles are shown in (e)–(g) while slant remains constant.

Frankot and Chellappa [17] reconstruct a surface by projecting the surface gradients onto a set of integrable basis functions (Fourier basis functions). Alternatively, the traditional approach is quite different to the methods of Kovesi, and Frankot and Chellappa. It involves integrating the surface normals, and requires some regularisation technique to ensure that the surface gradient is integrable. For example, Terzopoulos [80] uses an energy function to minimise deviation from depth and orientation constraints, as well as surface discontinuity. However the energy function exhibits some gross dimensional inconsistencies. Many other regularisation techniques have been proposed [29, 85, 66, 39], however in general, the approach that integrates surface normals can be very sensitive to noise.

2.1.3 Distance

Instead of calculating surface normals then using them to reconstruct shape, one may use another approach to shape reconstruction. Distances may be calculated

between the camera's focal point, and points on the surface. Assuming that the camera calibration is known, surface point coordinates may be obtained by ray cutting along rays where the distance is known (see Figure 5). The entire surface

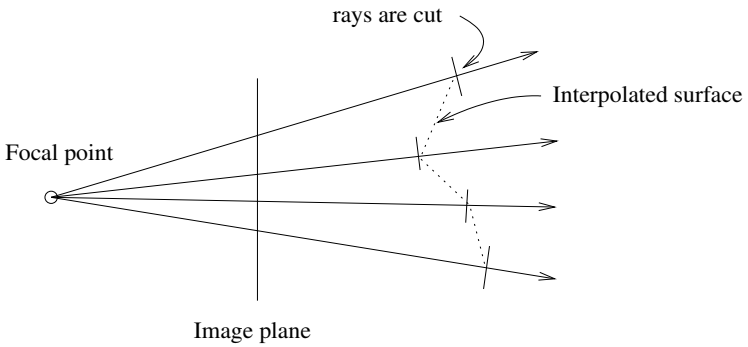


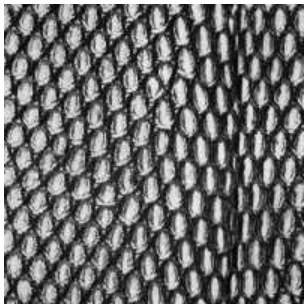
Figure 5: Ray cutting to obtain surface coordinates. The rays are cut at the estimated distance from the focal point to the surface. The estimated surface, shown as the dotted line, is then interpolated.

shape may be reconstructed by interpolating between the known 3D coordinates of surface points. Although this is not the approach generally used in my research this thesis does present an example where shape is constructed using distances in Section 5.3.1.

2.2 Texture

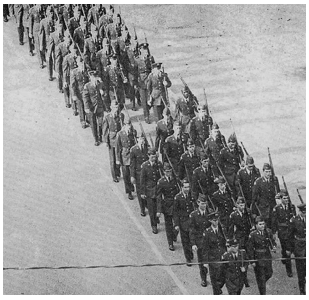
In this thesis, the word, 'texture' is taken to mean the visual surface pattern of an object. This pattern is assumed to have no surface height; in other words it is "painted onto" the surface. This is a common assumption in Computer Vision, although novel work exists that assumes texture to have in itself a three-dimensional structure; Leung and Malik [50] recover shape from textures that are self-occluding, such as the field of flowers shown in Figure 6(c) and the rows of soldiers shown in Figure 6(d). The occlusions themselves are used as a cue to infer shape. Koenderink and Pont [41], and Chantler [9] use the rugosity (or roughness) of textures to determine information about the illumination. The assumption that textures have no height does not drastically restrict the applicability of shape-from-texture algorithms because this assumption is representative of many real world textures.

(a) Spotted shoes



(b) Scales

(c) Field of flowers



(c) Soldiers marching

Figure 6: Textures assumed to have no height are shown in (a) and (b). The textures in (c) and (d) do not fit this assumption.

The literature often states that textures consist of texture elements, otherwise known as *texels*[1]. Texels are instances of a pattern that is repeated across a surface. For example in Figure 6(a), each spot is a texel. However many real textures, such as wood grain, are not composed of individual texels.

2.3 Assumptions

In all shape-from-texture work, it is necessary to make assumptions about the camera model, texture and surface. This section outlines the most common assumptions, and defines them as they are used in this work.

Surfaces are usually either assumed to be **planar** or **curved**. Earlier work tended to focus on estimating the orientation of planar surfaces and in later work, the focus shifted to curved surfaces as will be seen in the following section. In the work described in this thesis, shape is estimated for curved surfaces.

[1] Texels are sometimes referred to in the literature as textels, texons or textons.

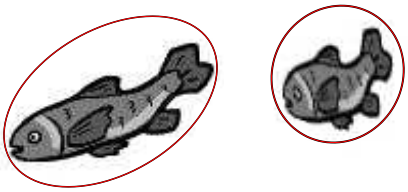
The most common camera models are the **orthographic** model and the **perspective** or **pin-hole** model. In the orthographic model, it is assumed that the viewing angle is small compared to the distance to the surface. As such, the light rays incident on the image plane are modeled as parallel rays. In practice this set up can be approximated by photographing the object from a large distance and using a zoom function on the camera. In the perspective model, the light rays converge to a focal point. When performing shape-from-texture using the orthographic camera model, the texture is assumed to undergo the foreshortening effect, that is, the texture visually appears to compress in some tilt direction but does not scale depending on its distance from the camera. On the other hand, when using the perspective model, it is assumed that the texture may scale depending on its distance to the camera, as well as foreshorten.

Common assumptions for texture are **homogeneity**, **stationarity** and **isotropy**. In this work, these terms are defined as such: Homogeneity means that the distance between texels and their pattern of placement is consistent across the surface. The term refers to the *location* of texels, without regard to the rotation of each texel. Stationarity means that the texels differ by a translation on the surface but not a rotation. Isotropy means that the texel has a constant second moment or inertia about every axis. The inertia of a texel is a function of angle, given by

$$I(\theta) = \sum_i \sum_j r^2 f(i,j) \tag{1}$$

where $f(i,j)$ is the value of the image of the texel at (i,j) and r is the perpendicular distance from (i,j) to the axis in the direction θ. The inertia of a texel can be depicted with an ellipse as shown in Figure 7. For each image in Figure 7 the distance from the centroid to the point on the ellipse in the direction θ is the value of the inertia about the axis orthogonal to θ. Roughly speaking, a texture is isotropic if the texels are as round as they can be. The fish in Figure 7(d), for example, is isotropic and therefore its inertia is represented by a circle.

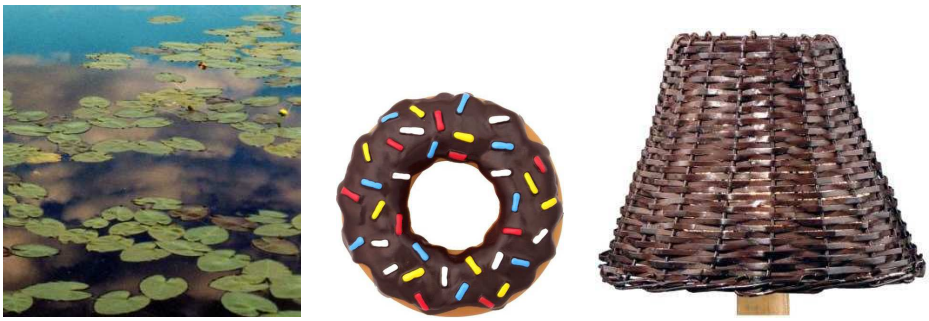
To further illustrate the definitions of homogeneity, stationarity and isotropy, it is easiest to consider the cases where textures do not exhibit these properties. See Figure 8. The waterlilies are non-homogeneous because they are not placed



(a) (b)

Figure 7: The inertia of a texel can be depicted with an ellipse.

on the water's surface in a regular pattern. The donut's texture is non-stationary because the sprinkles are rotated on the surface. The lamp has an anisotropic texture because the texels are long and thin, instead of round; in other words the inertia is not constant about all axes because higher frequencies are encountered in the vertical direction than in the horizontal direction. As another example, in



Waterlilies: non-homogeneous | Donut: non-stationary | Woven lamp: anisotropic

Figure 8: Examples of non-homogeneous, non-stationary and anisotropic textures.

Figure 6(a) the texture on the shoes is homogeneous, stationary and isotropic.

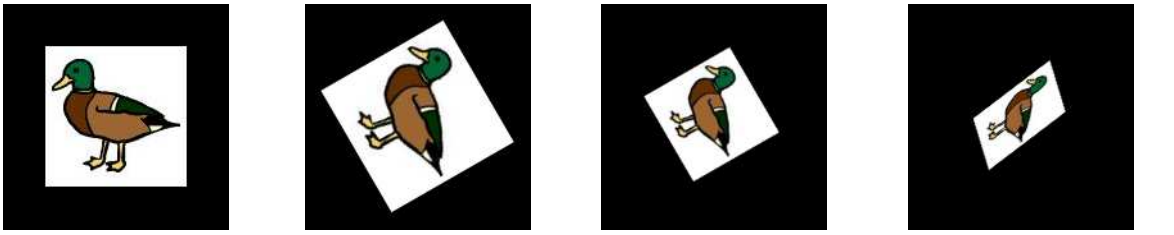
2.4 Projection geometry

In this work, the general transformation $\mathbf{T}$ of a surface point $\mathbf{x}_s$ expressed in two-dimensional surface coordinates to an image point $\mathbf{x}_i$ is given by

$$
\begin{aligned}
\mathbf{x}_i &= \mathbf{T}\mathbf{x}_s \\
&= (\mathbf{U}\mathbf{K}\mathbf{U}^{\mathrm{T}})(\mathbf{S})(\mathbf{U}\mathbf{V}^{\mathrm{T}})\mathbf{x}_s
\end{aligned}
$$

where

$$
\begin{aligned}
\mathbf{U} &= \begin{pmatrix} \cos(\tau) & \sin(\tau) \\ -\sin(\tau) & \cos(\tau) \end{pmatrix}, \\
\mathbf{K} &= \begin{pmatrix} \cos(\sigma) & 0 \\ 0 & 1 \end{pmatrix}, \\
\mathbf{S} &= \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix}, \text{ and} \\
\mathbf{V} &= \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}
\end{aligned}
\tag{2}
$$



(a) Original image (or texel in this context) (b) Rotated by $\mathbf{U}\mathbf{V}^{\mathrm{T}}$ (c) Rescaled by $\mathbf{S}$ (d) Slanted by $\mathbf{U}\mathbf{K}\mathbf{U}^{\mathrm{T}}$ to give reoriented image (or texel)

Figure 9: Components of a local texture imaging transformation, shown as a sequence of transformations.

The overall transformation $\mathbf{T}$ linearly approximates the transformation from the frontal view of a texture to its observed view. Its components are explained in the following list.

- The transformation $\mathbf{U}\mathbf{V}^{\mathrm{T}}$ is some initial rotation of the frontal texture by the angle $-\theta + \tau$. An example is shown in Figure 9(b). This transformation does not play a role in determining orientation since initial rotations of the frontal texture do not alter its slant or tilt. Also, $\mathbf{U}\mathbf{V}^{\mathrm{T}}$ is omitted when dealing with stationary textures since surface rotations are not considered.

- The transformation $\mathbf{S}$ scales the frontal texture isotropically by a factor of s. See Figure 9(c). This transformation is omitted when dealing with an orthographic camera model.

- The foreshortening effect is due to $\mathbf{UKU}^{\mathrm{T}}$. This transformation is interpreted as a rotation about the negative of the tilt angle τ, a foreshortening in the horizontal direction by a factor of $\cos(\sigma)$, followed by a rotation about the tilt angle. An example is shown in Figure 9(d).

The overall transformation $\mathbf{T}$ is in the set of *local texture imaging transformations* [16], which is a subset of affine transformations.

2.5 Previous work

Gibson first suggested in 1950 that the human perception of 3-dimensionality is influenced by texture gradients [18]. Gibson analysed the case of a perspective projection of a textured ground plane. The plane was assumed to be covered with texels at a uniform density (where density is given in number of texels per unit area in the image). Gibson observed that the orientation of the surface could be derived from the gradient of the texture density.

Since Gibson's seminal work, a plethora of computational methods have emerged which estimate shape using texture cues. Early work includes that of Witkin, who solved the orientation of surfaces using the assumption that the texture is isotropic [86]. This assumption is rather restrictive and does not apply to the majority of textures which are directional. For example, in Witkin's paper, he showed how a hair texture gave incorrect results because it was not isotropic.

Like Witkin's work, many early studies were *feature-based*. They involved detecting features such as texels, edges, or line segments and shape was then estimated by analysing properties of those features, such as their density gradient, area, and size. These early studies include the work of Stevens [77], Kender [37], Ohta et al. [67], Ikeuchi [30], Aloimonos and Swain [1], Blostein and Ahuja [3] and Kanatani and Chou [35]. One drawback of the feature-based approach is that it requires a feature-detection step. It has also been argued [79] that the feature-based approach does

not make full use of the information available.

An alternative approach utilises the spectral information. Methods of this type compare the spectral representation of windowed image patches to recover orientation. Commonly used spectral representations are the Fourier transform, wavelet decomposition and Gabor transform. An advantage of the spectral approach is that windowing issues are made simple; a slight shift in the windowing function affects the image domain, but only affects the phase part of the spectral representation. Often, the phase information is discarded and only the amplitude information used.

The first work using a spectral approach was by Bajcsy and Lieberman [2]; they used the Fourier power spectrum to estimate the transformation between a pair of spectra using the location of peaks in each representation. The transformations gave the relative depths of a ground plane.

Brown and Shvaytser [7] also worked on the ground plane problem. They assumed an isotropic texture and used the autocorrelation function to recover orientation. However, as implied earlier the assumption of isotropy is generally considered as being overly-restrictive.

Gårding [20] developed an elegant differential framework for shape-from-texture. In his work, Gårding derived the relationships between changes in texture gradient and the differential geometry of curved surfaces with stationary textures. However Gårding's contribution was mainly theoretical, as he assumed isotropy in empirical tests. His theoretical framework was then adopted by Malik and Rosenholtz [60]. They proposed a method to solve for the affine transformation directly and showed some good results for a variety of surfaces. Clerc and Mallat [10] proposed a method similar to that of Malik and Rosenholtz, using wavelets instead of the Fourier transform.

Forsyth [16] pointed out that in the work of Gårding, Malik and Rosenholtz, and Clerc, stationary textures are assumed. In his work Forsyth solves shape-from-texture without assuming homogeneity, isotropy or stationarity for orthographic views. Forsyth also points out that in order to solve the same problem for perspective views, something of the surface geometry must be known.

This thesis presents a new shape-from-texture method that uses the same assumptions as Forsyth's work [16], except that perspective views are accounted for — that is, scaling of texels are allowed. The texture is assumed to be smooth and a consistency constraint is used to determine surface shape. This algorithm is presented in Chapter 5 of this thesis.

The shape-from-texture method in Chapter 5 is designed to be a very general method that avoids many assumptions about the texture. In contrast, the other new shape-from-texture method presented in this thesis (Chapter 3) is designed to work for cases that are less general but that are still common for many everyday objects; the textured surface is assumed to be homogeneous, stationary and viewed orthographically. The method is fast and direct, and gives good results for real images. The new method uses the same assumptions as the well-known method by Super and Bovik [79], and a comparison of the two methods is given in Chapter 3.

One aspect of many shape-from-texture algorithms that deserves special attention is calculating the affine transformations between patches of texture. Not only does this task play a key role in many shape-from-texture methods (since the transformation can lead to surface orientation), but it is useful in a variety of applications such as object re-texturing. A usual way to find the affine transformation between two images is to locate feature points (such as Harris points) in the two images and to compute the correspondences [34]. Three points (that are not co-linear) are required to specify the affine transformation. This method will work if the two patches of texture are two views of the *same portion* of texture. However if the two patches are *different portions* of the texture, point correspondences cannot be made. This observation is described in detail, with examples, in Chapter 4.

Ways of solving the affine transformation without using point correspondences have been proposed. For example, Malik and Rosenholtz [60] solve the affine transformation using the Fourier spectrum, however their method can only be used in cases where there exists a small difference between the two textures. In Krumm and Shafer's work [46], the affine transformation between textures was solved by searching over every combination of slant and tilt in a discrete set, however this method is computationally very expensive. Ribeiro et al. [72] observed the shift in

peaks of the Fourier transform to estimate the affine transformation. Correspondences between peaks were made manually. In theory these correspondences can be made automatically, based on the rank of the peaks in terms of magnitude. In practice however, the rank of peaks is highly subject to noise. Kanatani [33] solved the slant and tilt components by relating these to the second Fourier components of the inverse Buffon transform. It was found [74] that the approximations that are used to solve the problem cause errors that increase with slant and tilt angles. Sato and Cipolla [74] developed a method for solving the affine transformation between textures using the changes in the first, second and third moments of the edge orientation, and the changes in density. Unlike the feature-point approach, their method did not require the correspondence of features, however extraction of features was required.

This thesis contributes a new method for calculating the affine transformation between two patches of texture that does not require features to be detected. The method simply transforms each patch of texture to its isotropic state and the calculates the similarity transform between the two isotropic textures. From this the transformation between the two original texture patches can be calculated. This method is presented in Chapter 4 and it is used as part of the algorithm in the following Chapter 5.

Another aspect of shape-from-texture that deserves special attention is the *scale* at which we are analysing the texture. Analysis with an inappropriate scale can lead to severe errors in shape estimation. The issue of scale is not mentioned much in the majority of shape-from-texture papers, nor in this thesis. There is a commentary in Chapter 3 about automatically choosing a window size with which to extract patches of texture. However, automatically setting the scale of analysis is not the focus of this work and it is assumed that any robust method that adaptively chooses a correct scale may be used. For shape-from-texture methods which use an adaptive scale the reader is referred to the work of Stone and Isard [78], Lee and Kuo [47], and Ribeiro and Hancock [71].

2.6 Analogy with other studies

To finish this chapter's general discussion on shape-from-texture, it worth noting that shape-from-texture is a similar problem to structure-from-motion and shape-from-multiple-views. This has been pointed out previously [61, 16]. A textured surface can be thought of as many views of the same texel; any two texels have different appearances due to their geometric relationship to the observer. If the true (or frontal) appearance of the texel can be estimated, the orientation of each observed texel can be estimated, and from this the surface can be reconstructed.

This is similar to the problem of structure-from-motion, in that each observed texel on a textured surface may represent different observations of a moving object, the texel. See Figure 10. The aim is to estimate the true shape of the texel.

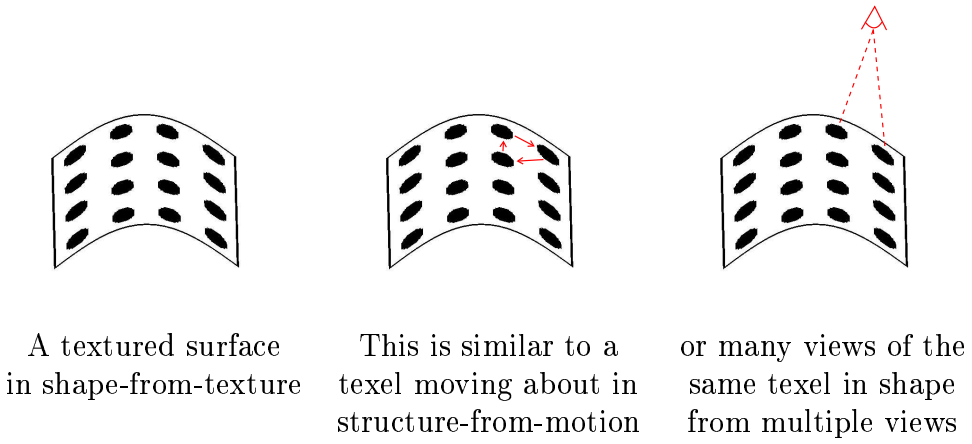


Figure 10: Shape-from-texture and analogous problems.

Structure-from-motion is also a similar problem. The relative motion of the observer and the object produces the different views of the object, from which the shape of the object is estimated.

As pointed out by Forsyth [16] much of the theory arising out of shape-from-texture is analogous to the theory in structure from motion. In Chapter 5.3.5 we see show how our second shape-from-texture algorithm can be altered to solve a structure from motion problem.

Chapter 3

A Method for Orthographic Views

This chapter presents a novel shape-from-texture method that assumes that the texture is homogeneous and stationary, and that the viewing geometry is orthographic. The method is based on the observation that, as an image is slanted, the spectral inertia does not vary about the tilt axis. Hence the tilt axis can be calculated without explicitly calculating the transformation undergone by the image. Once the tilt angle is known, slant is uniquely determined using the inertia at the angle orthogonal to tilt, as will be described in Section 3.1.1, thus completely solving for the surface normal. In this work, the algorithm was implemented so that it included a step for user input to identify the frontal texture. This step can be substituted by any method that automatically identifies the frontal texture, and approaches for this are discussed. The novelty of the method lies in the exploitation of properties of the orthographic viewing geometry and hence it is the focus of my work.

The previous algorithms by Gårding [20], Malik and Rosenholtz [61] and Clerc and Mallat [10] also assume texture homogeneity and stationarity, however they also allow for a perspective viewing geometry. The new method assumes orthographic views because it is intended to be used on smaller objects such as animals, clothing and fruit, rather than very large objects such as vast textured landscapes. If photographed properly (as described in Section 2.3), the perspective effects are negligible. In the methods by Gårding, Malik and Rosenholtz, and Clerc and Mallat, transformations between patches of texture are explicitly calculated, whereas this is not required in the new method due to the exploitation of properties of the

orthographic viewing geometry. Thus, the new method has an advantage of being more direct, with the cost of restricting the type of image one can use.

The shape-from-texture method by Super and Bovik [79] has the same assumptions regarding the texture and viewing geometry as the method presented in this chapter. However their method has an ambiguity in the solution of tilt direction (due to their mathematical formulation) that the new method does not have. In addition, their method is prone to returning complex values for tilt, especially near areas of frontal texture, as will be shown in Section 3.3.

3.1 Description of new algorithm

3.1.1 Theoretical behavior

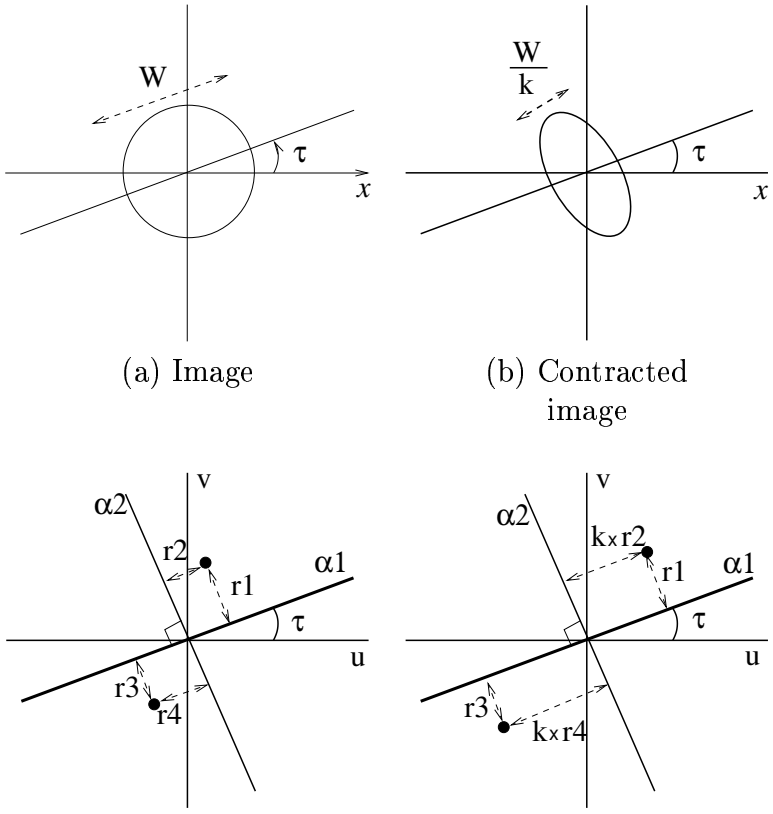
The method relies on the fact that when an image is foreshortened in a certain direction τ its Fourier transform[1] stretches in that same direction, so that there is theoretically no resulting change in the spectral inertia about the axis in the direction τ (see Figure 11). The inertia, I is given as follows.

$$I(\theta) = \sum_u \sum_v r^2 F(u,v) \tag{3}$$

where $F(u,v)$ is the amplitude of the Fourier transform at (u,v) and r is the perpendicular distance from (u,v) to the axis in the direction θ. Figure 11 (a) represents the frontal view of some image and Figure 11 (b) represents the image after it has been slanted in some tilt angle. In theory the tilt could be easily estimated as such: take the Fourier transforms of the frontal and slanted views of the image. Calculate the inertia about every axis for both transforms. The tilt angle τ is the angle at which the inertia are equal.

Slant can also be estimated. Comparing Figure 11 (c) and (d), we note that the points in the Fourier image increase their perpendicular distance to the axis α_2 by a factor of k. Thus the inertia about α_2 is increased by k^2 since inertia is dependent on perpendicular distance squared. Therefore k is given by $\sqrt{\frac{I_2}{I_1}}$ where I_1 and I_2 are

[1] For brevity, the term 'Fourier transform' is taken to mean the amplitude of the Fourier transform throughout this chapter.



(a) Image

(b) Contracted image

(c) Two arbitrary points in the Fourier transform of (a)

(d) The corresponding points in the Fourier transform of (b)

Figure 11: Principle for using spectral inertia. The image is contracted in the direction τ by a factor of k. In the Fourier image, each point increases its distance from the $\alpha 2$ axis by a factor of k. Since $r1$ and $r3$ are unaffected, this results in no change in the inertia about the $\alpha 1$ axis.

the inertia of the frontal and slanted images respectively, at the angle orthogonal to the tilt angle, $\tau + \frac{\pi}{2}$. The slant angle, σ is given by $\arccos(\frac{1}{k})$.

In practice, there are factors which affect the behaviour described above. These include windowing issues and scaling of the Fourier transform when it is stretched. In the following section it is shown that a simple windowing scheme may be used to cancel the scaling property of the Fourier transform.

The scaling property

If $\mathcal{F}(f(x)) = F(\omega)$ is the Fourier transform of a function $f(x)$, then

$$\mathcal{F}(f(kx)) = \frac{1}{k} F(\frac{\omega}{k}) \tag{4}$$

In plain English, if the Fourier transform is stretched out by a factor k, its magnitude decreases by k.

Windowing scheme

The windowing scheme refers to the method used to isolate patches of texture. Here it is shown that a simple Gaussian windowing scheme leads to cancelation of the scaling property.

Consider the image of a texture windowed with a Gaussian shown in Figure 12 (a). This windowed image was contracted by a factor of 2 in the direction 90° to give the image shown in Figure 12(b). The Fourier transforms of (a) and (b) are displayed directly below them in Figures 12(d) and (e). The transform in (e) differs from that in (d) in two ways: firstly, the peaks have been stretched outward by a factor of 2 in the direction 90°. Secondly, the peaks have decreased in magnitude by a factor of 2 due to the scaling property. In Figure 12 (c), we see the case where the *texture* has been contracted by a factor of 2 in the direction 90°, but the *window* used is the *same Gaussian* as was used in (a). The Fourier transform of (c) is shown below in Figure 12(f). The transforms (e) and (f) are the same, except that the magnitudes of the peaks are twice as large in (f). This is because twice as much of the contracted texture is visible in the windowed region. Thus, if we compare the Fourier transforms in (d) and (f), a shift in peaks will still occur, but the magnitudes of the peaks remain the same. In other words, the movement of peaks is apparent, whilst the scaling property has been canceled.

By the same reasoning, we can cancel the scaling property for *any* factor of contraction, not just 2, and in any direction of contraction. Therefore, as long as the same Gaussian window is used to extract patches of texture, the new method of comparing moments is valid.

3.1.2 A simple planewave example

Figure 13 demonstrates the method proposed so far. A planewave texture windowed with a Gaussian is shown in (a). This represents the frontal view of some texture. In Figure (b) we see the texture when it has been contracted by a factor of 1.6

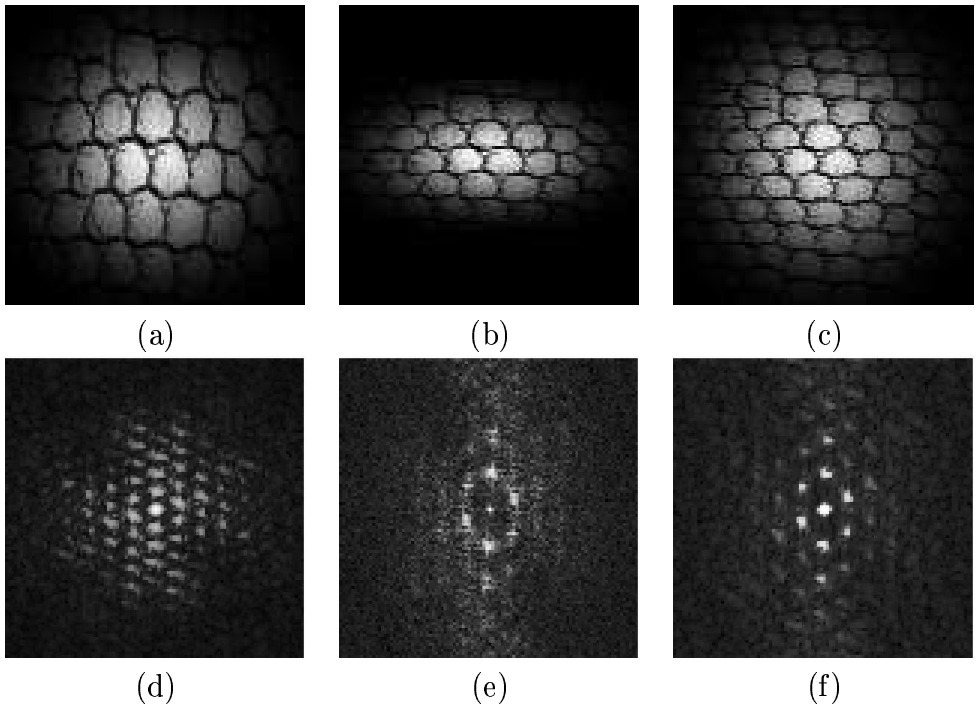
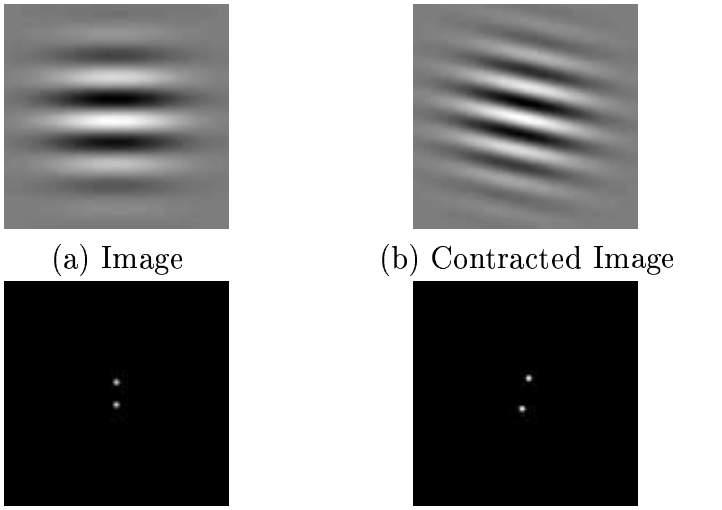


Figure 12: In (a) we see a texture windowed with a Gaussian. The image in (a) was contracted by a factor of 2 in the direction 90° to produce the image shown in (b). In (c) the *texture* has been contracted, but the window used is the *same Gaussian* as was used in (a). Their corresponding Fourier transforms are shown in the row below.

in the direction of 50°. This represents the texture seen from some oblique view. The aim is to find the orientation of the second planewave. The respective Fourier transforms are shown in (c) and (d) respectively. The inertia of the two Fourier transforms as a function of angle of axis are shown in Figure 14, denoted I_1 and I_2. The dashed line depicts the true tilt angle of 50°. The intersection of the two inertia lies on the required 50° line. Indeed, it was verified that the method of predicting the tilt angle was extremely accurate for this particular example, regardless of tilt angle. The parameter k (which leads to slant) is also accurately estimated for the planewave example. The dash-dot line in Figure 14 gives the angle that is orthogonal to the true tilt angle at -40°. At this angle, $\sqrt{I_2/I_1}$ is 1.55, which closely matches the true multiplying factor of 1.6 — this corresponds to an error in slant of 1.5°.

Note that another intersection is also present at about 80°, giving a false solution



(a) Image (b) Contracted Image

(c) Fourier transform of (a) (d) Fourier transform of (b)

Figure 13: Planewaves and their Fourier images. The planewave in (a) represents a frontal view of the texture and the planewave in (b) represents an oblique view of the texture.

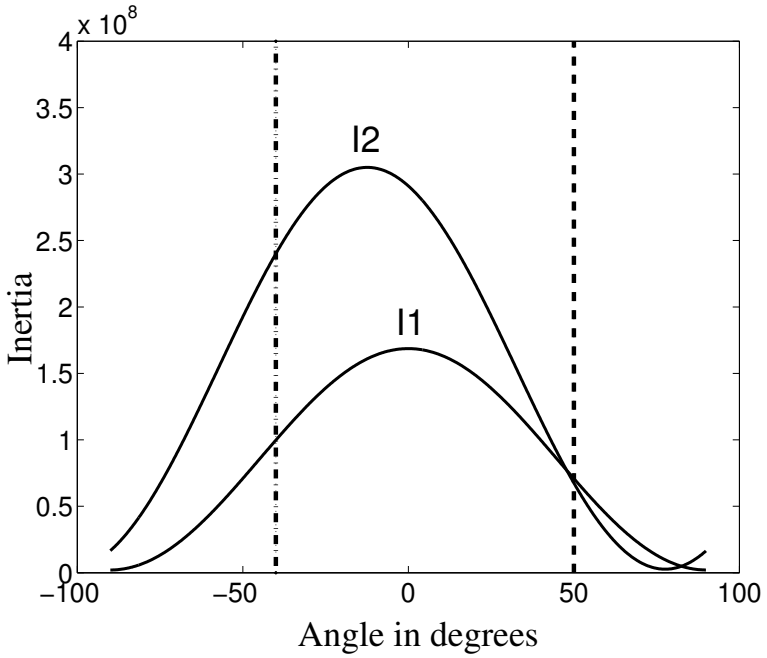


Figure 14: Spectral inertia of the two planewaves. I_1 corresponds to the original planewave and I_2 corresponds to the contracted planewave. The two curves intersect at the tilt angle of 50°, as required. The dashed line gives the true tilt angle and the dash-dot line gives the angle orthogonal to the tilt.

for tilt angle. The reason for this false solution is discussed in the following section, where it is also shown that false solutions for tilt can be easily avoided.

False tilt angles

In Figure 14 the curves display two points of intersection, and hence two candidates for tilt angle. Here it is shown why the false tilt angle occurs, and that it can be easily detected as false.

Figure 15 represents the peaks in the Fourier transforms of the two planewaves. The peaks corresponding to the frontal planewave are shown as $p1$ and $p2$, and the

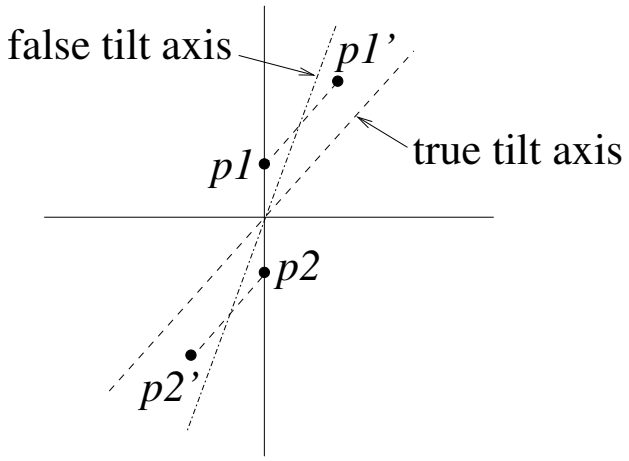


Figure 15: Reason for false tilt angle occurring. The points $p1$ and $p2$ represent the peaks in the Fourier transform of the frontal planewave. The points $p1'$ and $p2'$ are similar but for the transformed planewave.

peaks corresponding to the contracted planewave are shown as $p1'$ and $p2'$. The distance from $p1$ to the dashed axis is the same as the distance from $p1'$ to this axis. The same is true for $p2$ and $p2'$. Thus the dashed line is detected as the tilt axis. However, the same can be said for the dash-dot line, which may be falsely detected as the tilt axis.

For any real textures this is not a problem. Firstly, as there are only two candidates for tilt, both can be quickly tested by transforming the frontal planewave using both candidates. Only one version will match the planewave with the oblique view. Secondly, testing both candidates may not be necessary at all for realistic textures; the possibility of a false tilt angle only occurs when the number of peaks

in the Fourier spectrum is very small. Ideally the number of peaks is large as is the case with real spectra, and this means there will only be *one* tilt solution for which the inertia remains unchanged for *all* frequency components. At all angles other than the tilt angle, the spectral inertia of the original image will be less than the spectral inertia of the contracted image. This may be better understood when one considers what takes place in the image domain; the texture is contracted in some direction, leading to a stretch in the spectral domain, so the inertia is unaffected in only one direction.

3.1.3 Accounting for real factors

The planewave example above works well, but it differs to real world textures. For real images, two other factors to consider when comparing spectral inertia are illumination and blurring.

Illumination

Illumination results in a scaling of the Fourier transform by some unknown amount, which leads to a scaling of the inertia by that same amount. This is seen by referring again to Equation 3; if the values of $F(u, v)$ are scaled by some S, then the inertia I is scaled by S also.

Blur

Blurring draws energy in the Fourier transform closer to the origin, as higher frequencies are suppressed. This leads to a general decrease in the spectral inertia, however the inertia is not necessarily scaled by the same amount for all directions.

Normalisation

Illumination can be accounted for by inserting a normalisation step into the new method so far: after obtaining the two inertia I_1 and I_2 corresponding to the frontal patch and the local patch respectively, scale I_2 such that at all angles the scaled I_2

is greater than I_1, except for one angle, where the two are equal. The tilt angle is given by the angle where the inertia are equal. The slant angle is found as before.

The normalisation step also partially compensates for blur by expanding the inertia I_2 such that it can be compared to I_1. The algorithm is therefore robust to some amount of blurring for anisotropic textures.

3.1.4 Finding the frontal texture

The algorithm relies on obtaining a patch of the frontal texture so that all other patches may be compared to it. There are several approaches for doing this. If the image is known to display some areas of frontal texture these can in theory be automatically identified using a method such as the one suggested by Super and Bovik [79]. They claim that the frontal texture is where $\sqrt{Mm}$ is a minimum (M and m being the maximum and minimum values for the spectral inertia of a patch of texture). In theory this method should work, since $\sqrt{Mm}$ can only increase as a frontal segment of texture is slanted. However due to factors such as blurring this method is not robust. For example, if a patch of non-frontal texture is blurred, this decreases the magnitudes of the high frequency components which in turn decreases the spectral inertia. If the value of $\sqrt{Mm}$ becomes lower for the blurred patch than for the true frontal texture, an incorrect patch of texture will be identified as frontal.

After discovering the downfalls of using the above method for finding the frontal texture, I decided that my shape-from-texture algorithm could require the user to click on the frontal texture. This small manual input is a reasonable cost for a fast and direct three-dimensional reconstruction of a surface. Note that the frontal texture is often easy for a user to identify; for example, on a spotted surface, it is easy to find a patch of near circular spots if they are present in the image. However this method of finding the frontal texture limits the allowable input images to ones where some frontal texture is visible, although this disadvantage also applies to Super and Bovik's method.

Alternatively, if no frontal texture is visible in the image, shape estimation could still be achieved using a separate image of the frontal texture. For example, if a brick building were photographed such that no frontal point appeared in the image,

the algorithm could still be used, as long as an image of the frontal brick texture can be acquired.

Clearly the best solution to finding the frontal texture requires no knowledge of the true texture, is automatic, and works regardless of whether or not the frontal texture appears in the image. Forsyth pointed out [16] that, if the texture is composed of identical repeating texels, sets of three or more instances of viewed texels could be used to determine the appearance of the frontal texel. If Forsyth's method were adapted to working with *patches* of texture rather than individual texels, it could be used to replace the user interaction step of this new shape-from-texture algorithm. However, finding the frontal texture has not been the focus of the work in this chapter, and so this adaptation of Forsyth's work was not implemented although it remains an interesting and useful issue for future work.

3.1.5 Choosing the window size

Although the *shape* of the Gaussian window used to isolate patches remains the same for all patches, the *size* of the window needs to be chosen. In theory the window size needs to be large enough to encompass enough texture so as to accurately characterise the entire texture, but not so large as to lose resolution across a textured surface. For the experiments that follow, the window size is set manually. What is not covered here is an algorithm that automatically sets the window size depending on the texture, however there are several avenues one might try. Firstly, one might try to find the "characteristic scale" (the size of the main features) of the frontal texture using a method such as Lindeberg's [52]. The window would be set to a size that is larger than the characteristic scale in order to capture enough of the texture. There is also an iterative approach, where the window size is increased until the net change in the Fourier transform is below some threshold. The idea can be described with the example of a spotted texture; the window begins small and only encompasses part of a spot. As it increases in size, new frequencies are introduced and the Fourier transform changes quickly until and entire spot is viewed in the window. If the window is increased further in size, new spots enter but they only reinforce the frequencies that are already present so the Fourier transform does

not change a great deal.

3.1.6 Summary of algorithm

The algorithm is summarised below.

1. Begin with an image of a stationary, homogeneous texture viewed orthographically.

2. Obtain a patch of frontal texture using one of the methods in Section 3.1.4.

3. Calculate the spectral inertia I_1 as a function of angle for the frontal patch using the following formula for inertia [25]:

$$I(\theta) = \frac{1}{2}(c+a) - \frac{1}{2}(a-c)\cos(2\theta) - \frac{1}{2}b\sin(2\theta) \tag{5}$$

   where θ is the angle of the axis and a, b and c are the second moments, given by

$$\begin{aligned} a &= \textstyle\sum_u \sum_v u^2 F(u,v), \\ b &= 2\textstyle\sum_u \sum_v uv F(u,v), \\ c &= \textstyle\sum_u \sum_v v^2 F(u,v), \end{aligned} \tag{6}$$

   where u and v are the frequencies in the x and y directions respectively and F is the amplitude of points in the Fourier image. Note that Equations 3 and 5 are equivalent, but using Equation 5 is computationally much more efficient.

4. Window the image of the surface into local patches using a Gaussian window. The size of the Gaussian window has been discussed in Section 3.1.5.

5. For each windowed patch, take the Fourier transform.

6. Calculate the spectral inertia I_2 for the local patch using Equation 5.

7. Scale $I_2(\theta)$ such that at all angles it is greater than I_1, except at one angle where they are equal. Identify this as the tilt angle.

8. Locate the angle which is orthogonal to the estimated tilt angle. Using the values of I_1 and I_2 at this angle, calculate $k = \sqrt{\frac{I_2}{I_1}}$.

9. The estimated slant angle is given by $\sigma = \arccos(\frac{1}{k})$.

3.1.7 Interpretation using representative ellipses

It is helpful to describe the algorithm and the normalisation process using ellipses to represent the spectral inertia. Suppose we are estimating the tilt angle in a sample patch of texture such as the one in Figure 16(b), and we do this by comparing it to a patch containing the frontal texture, shown in Figure 16(a).

This comparison takes place in Fourier space. We take the Fourier transform of each patch of texture; these are shown in Figures 16(c) and (d). For simplicity, these figures only display the highest peaks. Then we calculate the inertia of each Fourier transform. In Figures 16(e) and (f) we see the resulting inertia curves I_1 and I_2 represented as ellipses. The distance from the origin to the ellipse[2] in some direction θ is equal to the spectral inertia about the axis in the direction θ. Figure 16(e) also depicts the concept of maximum and minimum inertia, M and m respectively. The minimum inertia m is the minor radius of the ellipse and the maximum inertia is the major radius. Although M and m are not used in our algorithm, they are an important part of Super and Bovik's algorithm discussed later in Section 3.3. The basic shape of the cloud of Fourier peaks is similar to the shape of the ellipse. However, one should not mistakenly assume that the ellipse roughly encompasses the cloud of Fourier peaks; rather, it lies at right angles to the ellipse that would. Thus the direction of the minimum inertia is the same as the direction where the cloud of Fourier peaks extends outward the most. The orientations of M and m are based entirely on the appearance of the texture, and not on the tilt direction τ.

[2]In reality, the shape of the inertia curve would not be elliptical, but peanut-shaped. Ellipses are used in this section to simply convey the idea of normalisation, although when the algorithm was implemented, this representation was not used; I used the inertia curves directly and used a binary search to find the scale factor for normalisation.

For isotropic textures $M = m$.

In order to solve the tilt angle, we need to find the axis about which the spectral inertia curves of the two patches are equal. This is depicted in Figure 16(g), where the ellipses corresponding to the frontal and sample patches are superimposed. The tilt angle is where the two ellipses just touch. This illustrates the ideal case.

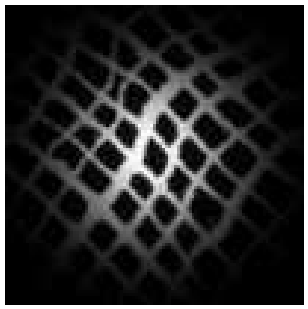
In practice, we see the case in Figure 16(h) where, due to illumination, the inertia curve has been scaled. Illumination affects only the *size* of the ellipse and not it its *shape*. This is also true to some extent for blur if it is not too severe. I_1 can be thought of as some unit inertia shape, multiplied by some factor c_1, where c_1 is unpredictably affected by blur and illumination in the image. Similarly, I_2 can be thought of as another unit inertia shape multiplied by a different factor, c_2. I_2 can be normalised to I_1 if it is scaled by a factor $\frac{c_1}{c_2}$. Normalisation requires that I_2 be resized such that it is larger than I_1 in all directions, except one. To understand why this is, consider what is happening in the image; as a frontally viewed image is slanted under orthographic viewing conditions, the frequency components increase in all directions except one, the direction orthogonal to tilt. Therefore we expect the spectral inertia of the slanted image to be greater than or equal to the spectral inertia of the frontal image in all directions.

In summary, the approach of estimating tilt angle using the intersection of the two inertia is only accurate if an ideal texture is used. The addition of illumination and blur require that we analyse instead the *change in the shape* of the inertia, and this can only be done in a normalised framework.

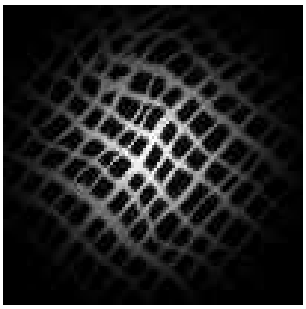
3.2 Results

3.2.1 Tests using the Brodatz texture database

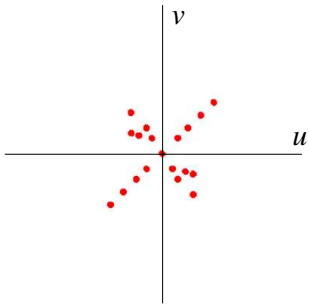
The algorithm was initially tested on texture patches created using the Brodatz texture collection [5]. Two images of the same texture were compared: the first image displaying the frontal texture, and the second image displaying the contracted version. Examples are given in Figure 17(a) and (b). Each pair of images was obtained using the following steps: window the centre portion of the Brodatz texture



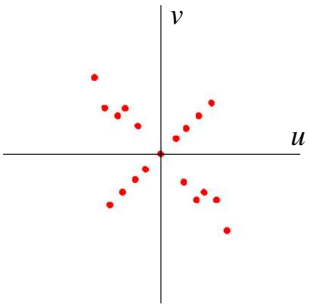
(a) Patch of frontal texture



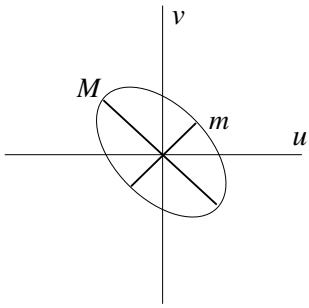
(b) Sample patch



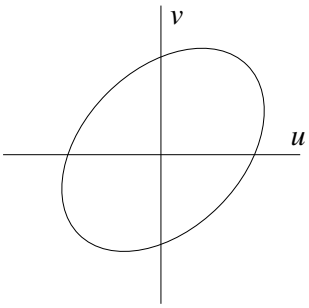
(c) Highest peaks in the Fourier transform of (a)



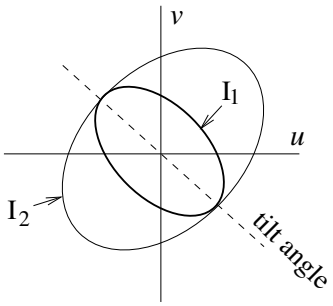
(d) Highest peaks in the Fourier transform of (b)



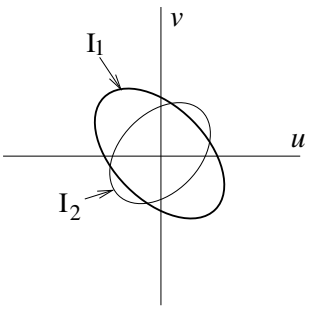
(e) Inertia curve I_1 depicted with an ellipse



(f) Inertia curve I_2

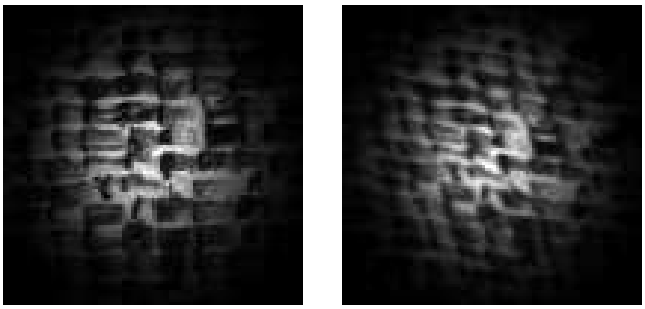


(g) The two inertia curves in the ideal case



(h) The two inertia curves before normalisation

Figure 16: Normalisation of inertia, illustrated using representative ellipses.



(a) Frontal texture (b) Contracted texture

Figure 17: Examples of input images

image, using a Gaussian window. This gives the first image and the frontal texture. Contract the original texture image by a factor k in direction τ, where k and τ are some initial test values. Window the center portion of this contracted image with the same Gaussian to obtain the second image. When the two images were input into the algorithm, the aim was to estimate τ and k. Then τ and k were varied to test the algorithm under different tilt and slant angles.

The frontal texture in Figure 17(a) was contracted in the tilt direction of 50° by a factor of 1.5 to give the patch of texture in Figure 17(b). Figure 18 shows the spectral inertia of the frontal texture, I_1, and the spectral inertia of the contracted texture, I_2. We also see the scaled I_2 which intersects I_1 at 50° giving the correct tilt angle. The dash-dot line shows the angle orthogonal to tilt. At this angle, $\sqrt{\frac{I_2}{I_1}}$ $= \sqrt{\frac{11.6}{6.0}} = 1.4$, which is close to 1.5.

The method was tested on 111 textures in the Brodatz database where τ was varied over the range -90°, -80°, ..., +90°, and k was varied over the range 1.5, 2, ..., 4. The mean absolute tilt error was 8.0°. Slant was calculated using the angle orthogonal to the *estimated* tilt angle instead of the real tilt angle to ensure fair testing. This gave an average absolute error of 13.6°.

Reasons for error

It was found that the error values given above were heavily influenced by the frequencies present in the textures. Figure 19 displays the mean error taken over all

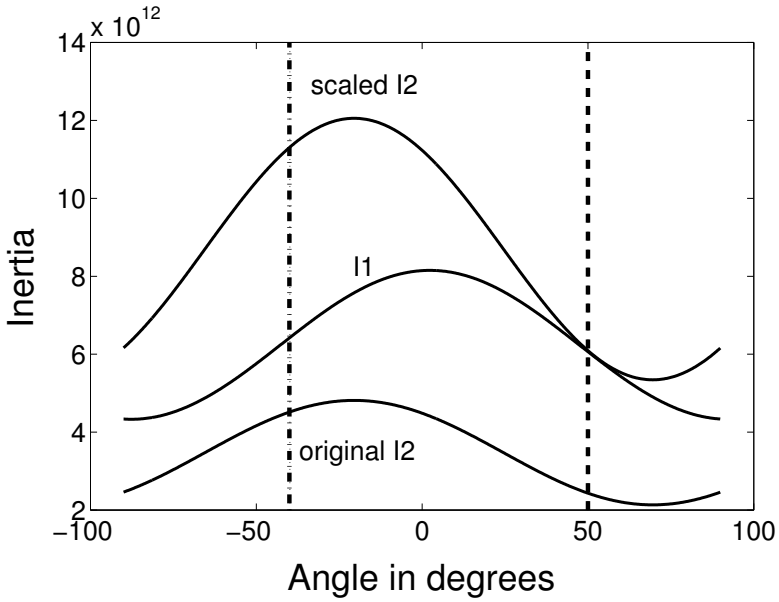


Figure 18: Spectral inertia of the two example textures. I_1 corresponds to the frontal texture and I_2 corresponds to the contracted texture. The scaled I_2 intersects I_1 close to the true tilt angle of 50° shown by the dashed line. The dash-dot line gives the angle orthogonal to tilt. The values of I_1 and I_2 at this angle lead to an accurate estimation of slant.

slant and tilt angles, versus the Brodatz number. The textures with the highest errors are labeled. These are the textures numbered D16, D21, D38, D49, D79, D105 and D106. Observation of the entire set of Brodatz textures (see Appendix A) reveals that these highlighted textures exhibit the highest frequencies, with little low frequency content. As a quick comparison, the highlighted textures are shown in the top row of Figure 20 and some textures that performed far better are shown in the bottom row.

It was found that textures that essentially only contained high frequencies performed badly because contracting them resulted in even higher frequencies, cutting out much of the texture's detail. From a spectral point of view, the high frequencies were more prone to the affects of aliasing, and so with little low frequency content, a comparison of the two Fourier transforms was not meaningful.

The way to deal with these textures is to increase the resolution. Using closer views of the texture and a smaller window size, the algorithm more easily tracks the changes in the texture. In our experiment, the resolutions of the textures were fixed according to the image database [70].

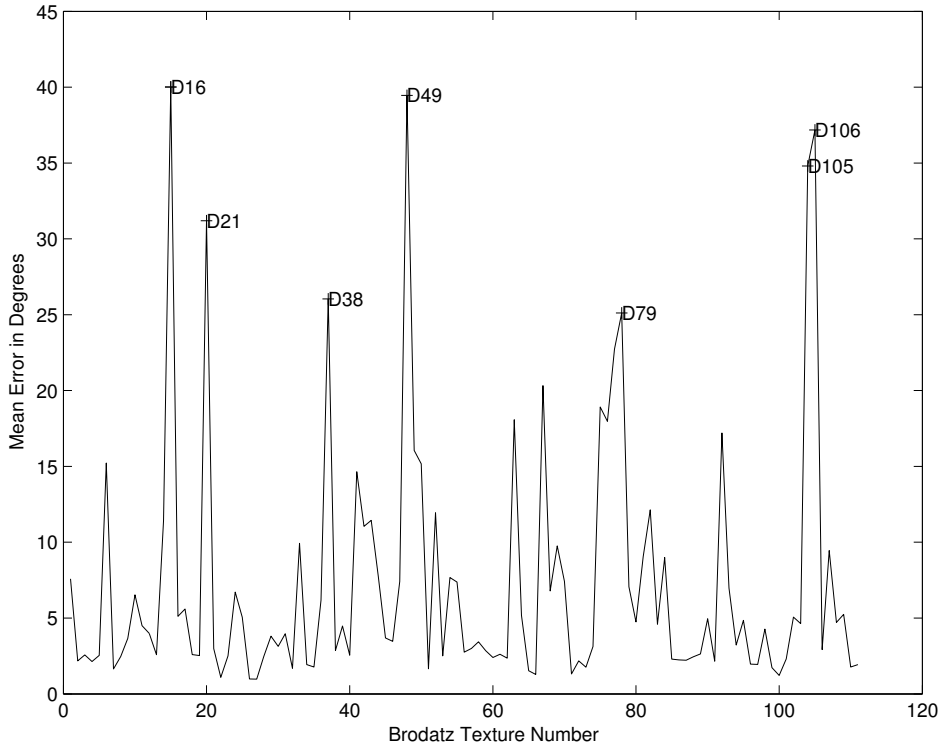
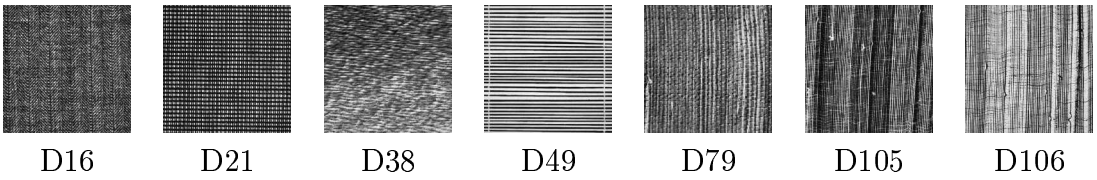
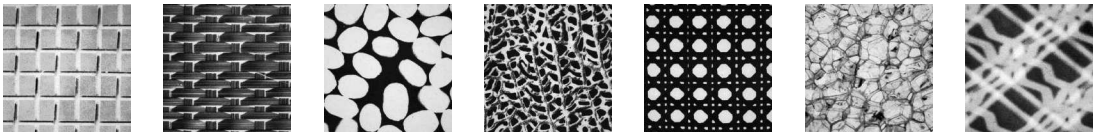


Figure 19: Mean error versus Brodatz number. The Brodatz textures with the highest errors have been labeled



D16 D21 D38 D49 D79 D105 D106

Worst performing textures. These have little low frequency content.



D1 D65 D75 D87 D102 D112 D45

A sample of textures that performed well. These have more low frequency content.

Figure 20: Frequencies affecting performance.

Another factor contributing to error is inhomogeneity. Several of the Brodatz textures were not very homogeneous, which meant that the patch of contracted texture contained elements not seen in the patch of frontal texture.

3.2.2 Tests using images of real objects

The robustness of the algorithm was demonstrated on images of real objects. To begin, the images were segmented to remove the effects of the background. This was done manually although a number of algorithms exist that do this [68, 36, 69, 84]. The background was set to be black. Patches were extracted from the textured surface such that the locations of all patches formed a regular grid. Some of these patches included part of the sharp border between the texture and black background, something so far unaccounted for in our description of the algorithm. In general this problem might be dealt with by either (i) having knowledge of the orientation along the border (via some other shape cue such as shading) so that this does not need to be determined using texture, or (ii) simply not calculating the orientation if part of the border appears in any particular windowed patch. In the second case, the surface cannot be reconstructed along the border, and the algorithm returns only the shape of the object inside the border.

Under certain conditions however, the algorithm may be used for all patches including those containing part of the border. In the experiments on real images that follow, the objects that were photographed have the property that its surface slants away from the camera at an angle of $\frac{\pi}{2}$ along its border. It turns out that the algorithm satisfactorily deals with such surfaces as shown in the following section.

Border of texture

Along the texture border, the algorithm returns a slant angle of close to $\frac{\pi}{2}$ and a tilt angle that is normal to the border; often this is exactly what we require, and some examples are given later. This result for slant and tilt at the texture border can be explained as follows. For patches containing part of the border, the sharp edge dominates the Fourier transform. An example is shown in Figure 21(b), where the highest values lie along a line that is orthogonal to the texture border shown in Figure 21(a). A plot of the inertia I_2 is shown in Figure 21(c) and it demonstrates that the largest moment is far greater than the smallest moment. The frontal inertia I_1 has a much smaller range than I_2. As before, normalisation involves scaling I_2 such that at all angles it is larger than I_1 except in one direction. Given the relative

flatness of the I_1 curve, the algorithm detects the tilt direction as the direction that corresponds to the lowest point of I_2. This is the direction that is along the bright line in the Fourier transform, in other words the direction orthogonal to the texture border. To calculate slant, the algorithm analyses the inertia at the angle orthogonal to tilt, depicted with a dashed line in Figure 21(c). The contraction factor k is given by

$$\begin{aligned} k &= \sqrt{\frac{\text{Scaled } I_2}{I_1}} \\ &= \sqrt{\frac{n_1}{n_2}} \qquad \text{where } n_1 >> n_2 \\ &= \mathrm{n}_3 \qquad \text{where } n_3 >> 1 \end{aligned}$$

and the slant σ is given by

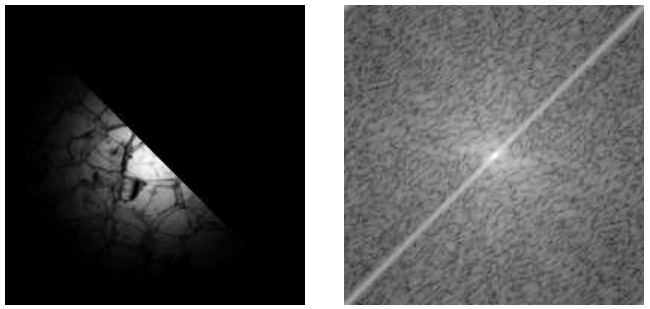
$$\begin{aligned} \sigma &= \arccos(\tfrac{1}{k}) \\ &= \arccos(\tfrac{1}{n_3}) \\ &\approx \arccos(0) \\ &= \tfrac{\pi}{2} \end{aligned}$$

In summary, at the border we often require a tilt angle orthogonal to the edge and a slant angle of $\frac{\pi}{2}$. Where this is the case it is reasonable to apply the algorithm to images where the texture has been segmented with a sharp border.

Golfball example

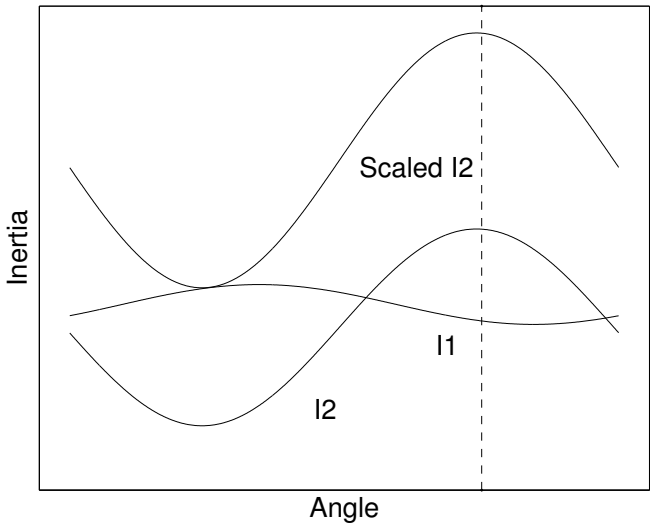
Figure 22 shows an example with a golfball. The original image is shown in (a). The golfball texture was segmented manually from the background to give the image in Figure 22(b). The image was windowed into local patches to calculate the slant and tilt in each patch. Also in (b) are the locations of patches showing some frontal texture and slanted texture. The circles depict where the standard deviation of the windowing Gaussian is unity. The corresponding patches used by the method are shown in (c) and (d).

The resulting slant and tilt are depicted as a needle diagram with tilt giving the angle of the needle and slant affecting the length of the needle. This is shown in Figure 23(a). Since it does not make sense to calculate the slant and tilt outside of the texture, the slant was set to zero for all background pixels.



(a) A patch containing some of the border

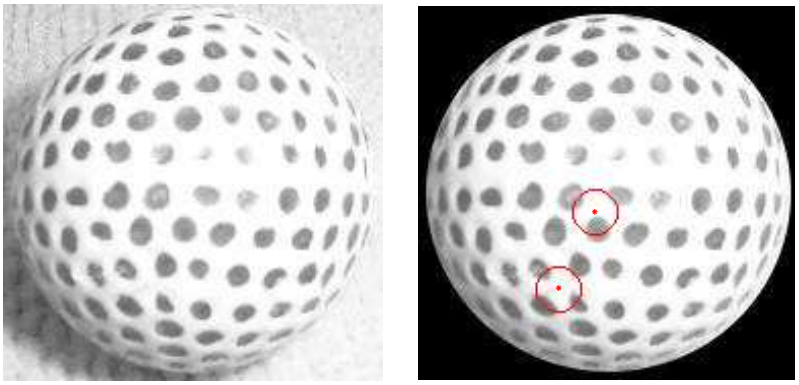
(b) log-Fourier transform of (a)



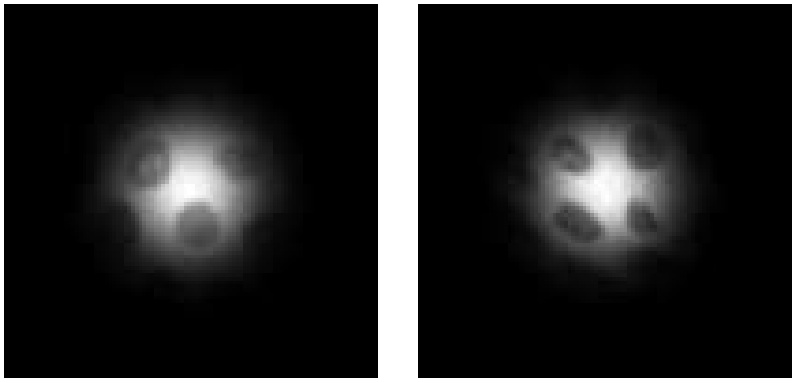
(c) Inertia I_2 of (b) and I_1 of some frontal patch (not shown). Since I_1 is rather flat compared to the scaled I_2, the tilt angle (shown as the dashed line) is estimated to coincide with the highest value of I_2. This angle is normal to the texture border, as required.

Figure 21: Effect of texture border

The algorithm so far determines the tilt angle only up to the well-known tilt ambiguity. This ambiguity is demonstrated by the fact that under orthographic projection, it is impossible to distinguish between a concave textured sphere and a convex one, based on texture alone. In terms of the needle diagram, we do not know which end the the needle is the point. For this sort of problem, a smoothness constraint is often used. For example one might try to avoid large second derivatives, although this approach was found to be unreliable. Lobay and Forsyth [53] have found that the ambiguity can be solved satisfactorily by choosing tilt values that do not allow the surface height to change too rapidly. In the golfball example and

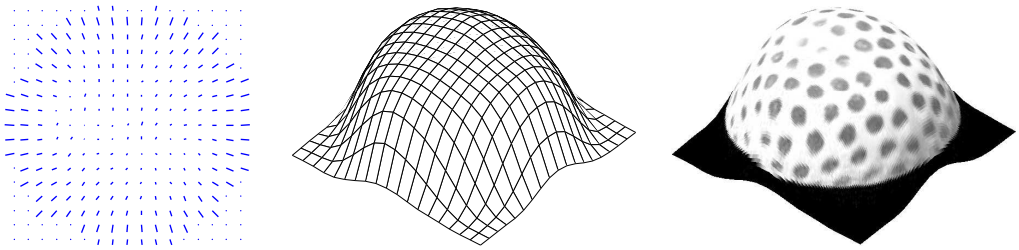


(a) Original image (b) Segmented image



(c) Patch showing frontal texture (d) Patch showing slanted texture

Figure 22: Test on an image of a golfball.



(a) Needle diagram (b) Mesh surface (c) Side view

Figure 23: Shape estimate of golfball

in the real examples that follow, I have simply used the fact that the needles point outward to solve the tilt ambiguity. Clearly this is not applicable to multimodal surfaces and in these situations one could employ Lobay and Forsyth's approach.

The map of slant and tilt angles was then used to build a surface using Kovesi's Shapelet method [43], seen in Figure 23(b) as a mesh surface. The original image

of the golfball was then texture mapped onto the surface to give a $2\frac{1}{2}$-D reconstruction of the golfball, which could be viewed from new angles. A side view is shown in Figure 23(c). This view has the same orientation as the mesh surface in Figure 23(b).

Accuracy

One expects a reasonable amount of error due to the fact that the texture is not perfectly homogeneous. The surface spots are hand-drawn and in some areas, not very circular. Since the golfball is spherical, is was possible to calculate ground truth values for slant and tilt. The mean slant error was 10.4° and the average tilt error was 16.6°. Analysis showed that the worst tilt errors occurred when the slant was low i.e. in areas closest to the frontal point. When the slant is zero, the tilt angle is undefined, but in areas nearby of low slant, the tilt can be highly inaccurate without changing the surface reconstruction dramatically; imagine a patch where the surface is near frontal, so that the needle in a needle diagram points almost directly out of the page. We can go so far as to reverse the direction of the tilt without substantially altering the surface. Figure 24 is a graph of the tilt versus true slant. It verifies that at low slant angles, the tilt error can be very inaccurate, showing an almost random spread.

Errors occur in the surface reconstruction too. For example, due to the nature of the Shapelet method, slant angles of exactly 90° are not achieved on the surface at the texture border.

Other examples

The algorithm was tested on other textured objects seen in Figures 25 and 27. With the cheetah example (Figure 25), shape is calculated only for areas with the spotty texture. In all other areas slant is set to zero. This is why in Figure 26 the cheetah's head is flat.

The estimated shape of the body of the cheetah is dependent on both the effect of the border and the texture foreshortening. Referring to Figure 26, we see that at points A and C the surface is relatively flat, but that at point B the surface

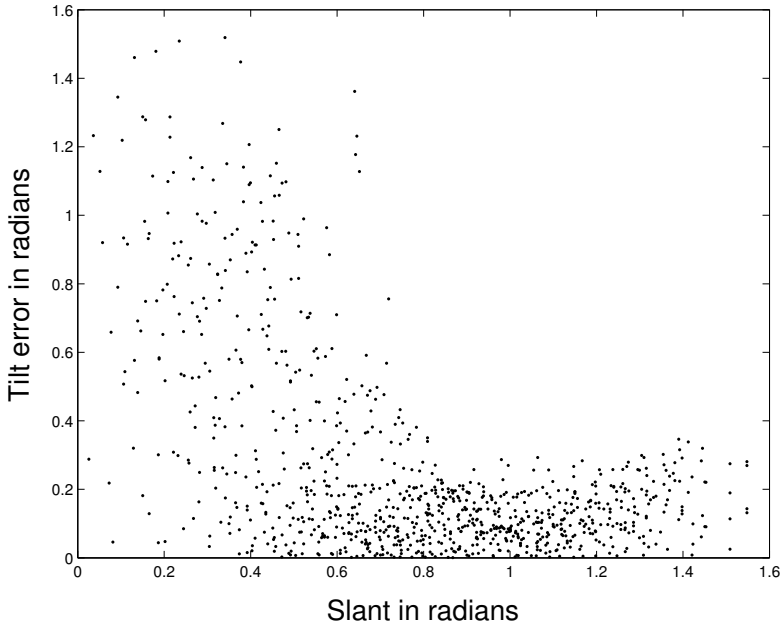


Figure 24: Large errors for tilt are encountered at areas of low slant.

is slanted due to the texture being foreshortened. At point D, and all along the cheetah's back, the surface slants away from the camera due to the effect of the border.

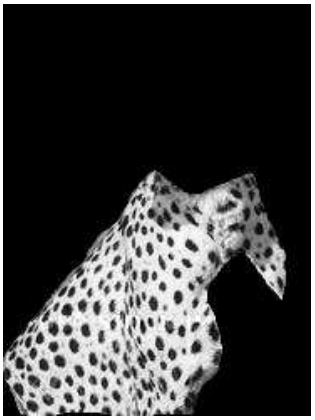
For the pear and strawberry examples (Figure 27) the effect of the border plays a more important role than the texture. Reasonable shape estimates are achieved even though the pear's texture contains mostly high frequencies (earlier in Section 3.2.1 we saw that these kinds of textures usually do not give good results) and the strawberry's texture contains specularities.

3.3 Comparison with Super and Bovik's method

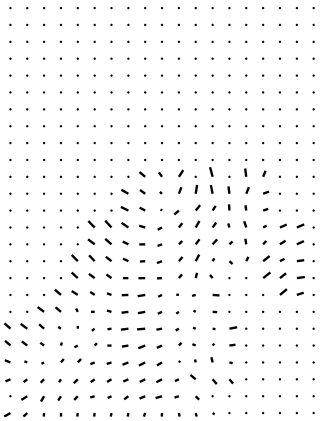
I chose to compare the new method to Super and Bovik's [79] because their method is relatively well-known and recent and uses the same assumptions as the new method regarding texture and viewing geometry.

Essentially, Super and Bovik's method evaluates slant and tilt by looking at changes in the spectral moments, as in the new method. However, there are many differences between their method and the new method. The change in moments is characterised entirely by the maximum and minimum moments and α, the angle

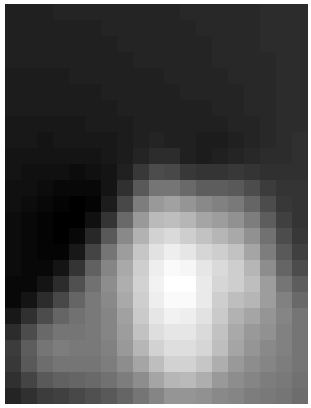
(a) Original image



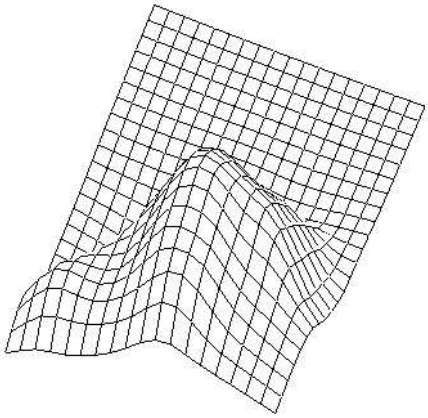
(b) Segmented image



(c) Needle diagram



(d) Surface height



(e) Mesh surface

(f) Projection of image onto surface

Figure 25: Results from performing shape-from-texture on the image of a cheetah.

of the axis about which the minimum moment is achieved. In contrast, the new method uses the entire curve of spectral moments versus angle instead of only two points on that curve. Another difference is the spectral representation used; Super and Bovik use a Gabor transform and the new method uses the Fourier Spectrum.

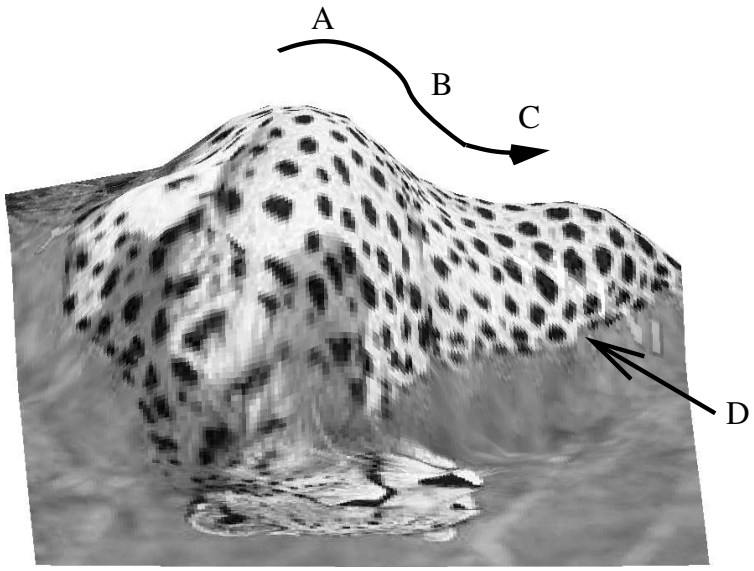


Figure 26: New view of cheetah.

Although the Gabor transform gives the most efficient image reconstruction in terms of balancing the resolution in the image and spectral domains, it is argued that the finer spectral resolution of the Fourier transformation is better at tracking the movement of spectral peaks, and hence the changes in spectral moments.

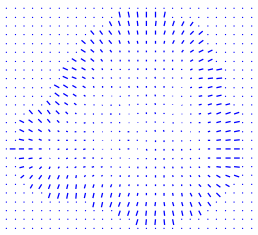
Another difference between the algorithms is the somewhat unnecessary ability of Super and Bovik's method to return the surface orientation at every pixel, although in theory this resolution can easily be reduced to improve processing speed. The new algorithm was not designed to be used on every pixel (although one might do this) but instead to extract and process patches at spatial intervals that are not unreasonable for the image size. For this reason, in Section 3.3.2 the results for the new method appear rather blocky compared to the results for Super and Bovik's method. A comparison of the speeds of the algorithms is presented in Section 3.3.2.

3.3.1 Overview of Super and Bovik's method

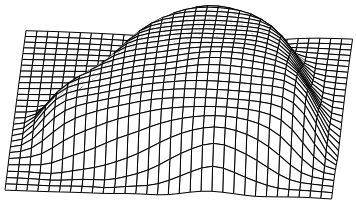
A viewer centered (x, y, z) co-ordinate system is adopted, where the x-y plane is the image plane, and the optical axis is the $-z$ axis. See Figure 28 (adapted from Super and Bovik's paper [79]) .

The local surface (x_s, y_s, z_s) coordinate system is defined as having z_s-axis as

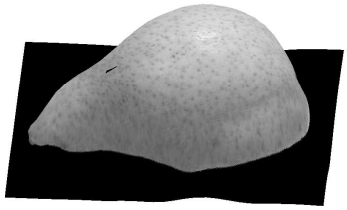
(a) Segmented image



(b) Needle diagram

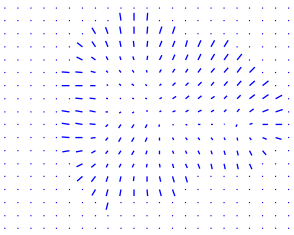


(c) Mesh surface

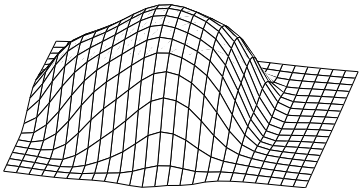


(d) New view of pear

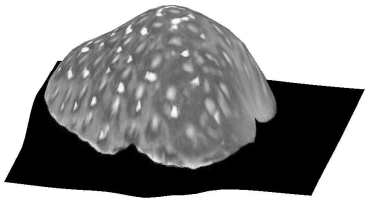
(e) Segmented image



(f) Needle diagram



(g) Mesh surface



(h) New view of strawberry

Figure 27: Results from performing shape-from-texture on a pear and strawberry.

the surface normal at the point on the surface and the x_s-axis so that its projection points in the direction of the apparent surface gradient. The y_s-axis is directed to give a right-handed orthogonal coordinate system. The tilt τ is the angle between

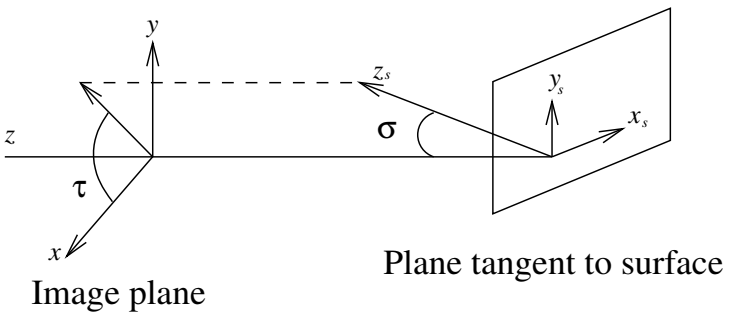


Figure 28: The coordinate systems used in Super and Bovik's algorithm.

the x-axis and the projection of the z_s-axis onto the image plane. The slant σ is the angle between the z_s- and z-axes.

Surface texture is characterised by the response of the image to filters of various frequencies. The local energy spectrum, A, is the result of convolving the image, $f(\mathbf{x})$ with the filter $h(\mathbf{x};\mathbf{u})$, and taking the absolute value:

$$A(\mathbf{u};\mathbf{x}) = |f(\mathbf{x}) * h(\mathbf{x};\mathbf{u})| \tag{7}$$

where $\mathbf{x} = (x, y)$ gives the image coordinates and $\mathbf{u} = (u, v)$ gives the spectral coordinates.

The filter $h(\mathbf{x};\mathbf{u})$ is a two-dimensional Gabor filter, centered about $\mathbf{u}$. A two-dimensional Gabor filter is a Gaussian-restricted planewave; for more information regarding Gabor filters, see the paper by Lee [48].

The second-order moments of the energy spectra are given by:

$$\begin{aligned} a(\mathbf{x}) &= \textstyle\sum_i u_i^2(\mathbf{x}) A_i^2(\mathbf{x}), \\ b(\mathbf{x}) &= 2 \textstyle\sum_i u_i(\mathbf{x}) v_i(\mathbf{x}) A_i^2(\mathbf{x}), \\ c(\mathbf{x}) &= \textstyle\sum_i v_i^2(\mathbf{x}) A_i^2(\mathbf{x}). \end{aligned} \tag{8}$$

where i indexes the samples in the spectral domain.

The canonical moments, M and m are the crux of Super and Bovik's algorithm. These, along with axis of least second moment, α, are used to directly calculate the tilt and slant. The canonical moments, M and m are the maximum and the minimum second moments respectively, and were depicted in Figure 16(e). M, m

and α are calculated from a, b and c according to the following equations.

$$\begin{aligned} M &= \tfrac{1}{2}(a + c + \sqrt{b^2 + (a-c)^2}), \\ m &= \tfrac{1}{2}(a + c - \sqrt{b^2 + (a-c)^2}), \\ \alpha &= \tfrac{1}{2}\arctan(\tfrac{b}{a-c}). \end{aligned} \tag{9}$$

Super and Bovik's algorithm is summarised by the following steps.

1. Convolve the image with Gabor filters to obtain the local amplitude spectra, A, according to Equation 7.

2. Calculate the image moments, a, b and c, according to Equation 8.

3. Compute M, m and α according to Equation 9.

4. Determine the frontal point, where $\sqrt{Mm}$ is minimised. Set $M_s = M$ and $m_s = m$. M_s and m_s are the canonical moments at the surface, or the frontal moments.

5. The slant σ and tilt τ are then given by[3]:

$$\begin{aligned} \cos\sigma &= \sqrt{\tfrac{M_s m_s}{Mm}}, \\ \tau &= \alpha \pm \tfrac{1}{2}\arccos\lambda,\ \alpha \pm \tfrac{1}{2}\arccos\lambda + \pi, \\ \text{where } \lambda &= \tfrac{(cos^2+1)(M+m)-2(M_s+m_s)}{sin^2\sigma(M-m)}. \end{aligned} \tag{10}$$

The slant angle is computed with no ambiguities, as it is required to be within the angles 0 and +90 degrees. However, the tilt angle has two ambiguities: one is the fundamental tilt ambiguity of $\pm\pi$, seen as the two alternatives for τ in Equation 10. The other ambiguity is a result of the mathematical derivation, manifesting as the $\pm$ sign in Equation 10.

[3]The full derivation for these equations is given by Super and Bovik [79]

3.3.2 Super and Bovik's results

A synthetic image

To compare Super and Bovik's algorithm to the new algorithm, a very simple test image was initially used to highlighted strengths and weaknesses in the two algorithms. The input image was a planewave with varying frequencies, shown in Figure 29. There are four regions in the input image where the wavelengths in

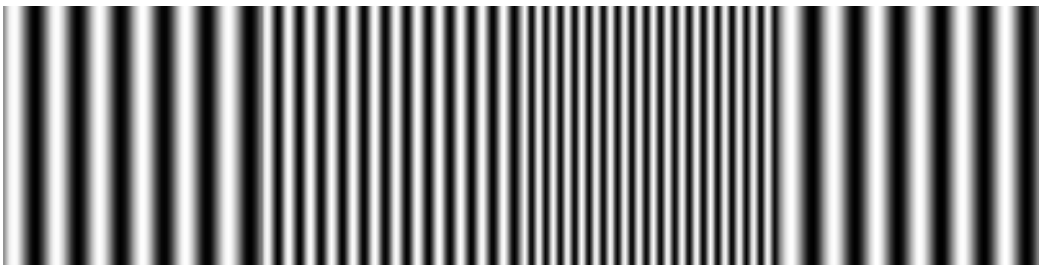


Figure 29: A simple test image

pixels are 16.6, 8.3, 5.6 and 16.6 from left to right. The image was used to test whether the algorithms could accurately estimate the amount of foreshortening in each region compared to the first region which was deemed the frontal region.

In order to compare the two algorithms, the filter bank for Super and Bovik's method had to be carefully chosen, as well as an appropriate window size in the new algorithm; a 'bad' choice for either of these could result in sub-optimal performances. Therefore these were chosen in a way that made sense. For the bank of Gabor filters, the largest and smallest filter had wavelengths of 20 pixels and 5.4 pixels respectively, which encompass all of the frequencies in the image. These filters are shown in Figure 30. There were six different wavelengths all together, giving good coverage of the spectral domain. Four evenly-spaced orientations for the filters were used. The size of window used in the new method was required to be large enough to view the lowest frequency in enough detail while being small enough to give a good resolution across the image. The Gaussian window chosen had a standard deviation of 2.67 pixels, shown in Figure 31.

Tilt was calculated correctly up to the tilt ambiguity by both algorithms. A comparison of the estimated slant is given in Figure 32. This figure shows how slant varies with increasing x-coordinate, when the y-coordinate is set to be halfway

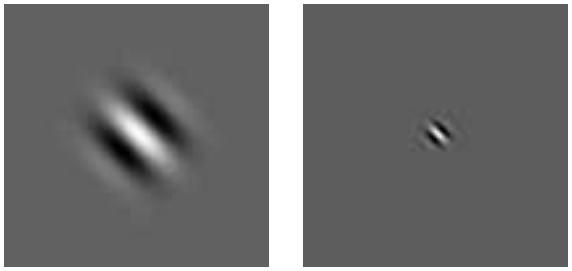


Figure 30: Gabor filters used with first test image. The real part is shown.

Figure 31: Gaussian window used for first test image

down the image (although any y-coordinate might have been chosen). The graph

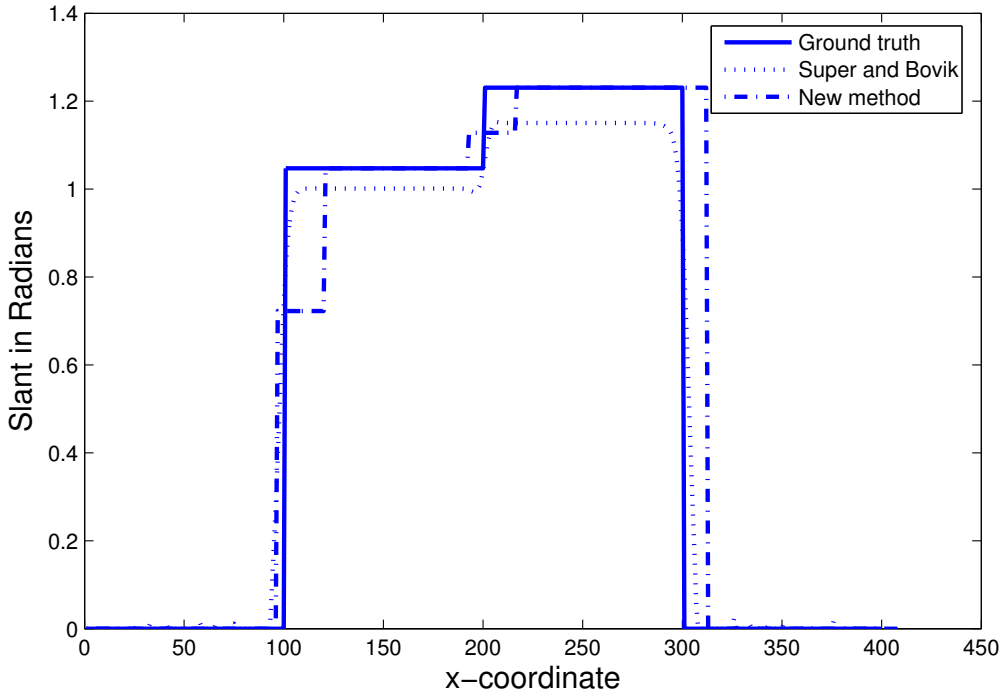


Figure 32: Comparison between the two algorithms. While Super and Bovik's method performs better at texture borders, the new method is better able to estimate slant where the texture is constant.

demonstrates that the new algorithm more accurately estimates slant in all areas except at the borders between the regions of different frequencies. These borders are

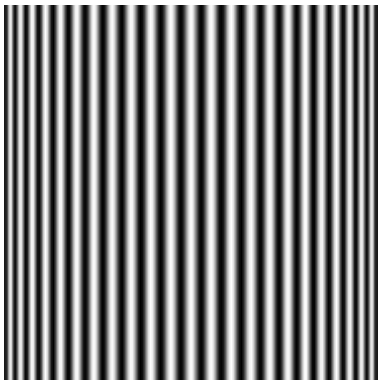
discussed later. One might expect the new algorithm to generally be more accurate for several reasons. Firstly, the windowed Fourier transform samples the spectral domain with constant weight. In contrast, when using the Gabor transform one attempts to choose filters that sum to give a constant sample of the spectral domain. In reality these filters do not sample with exactly constant power. Secondly, the windowed Fourier transform tracks the movement of spectral peaks in more detail than the Gabor transform, the cost of which is a blockier resolution in the spatial domain.

Several other filter banks for Super and Bovik's algorithm were tested to try to improve the accuracy of the slant, however none of these improved the accuracy shown in Figure 32. Indeed, a characteristic of Super and Bovik's algorithm is that it tended to underestimate slant, as was noted for several tests such as the one depicted in Figure 33. The input image of a cylinder shown in Figure 33(a) was used as an initial test of my implementation of Super and Bovik's algorithm. However no choice of filter bank was able to give a better result for slant than is shown in Figure 33(b), where the estimated slant is always lower than the real slant.

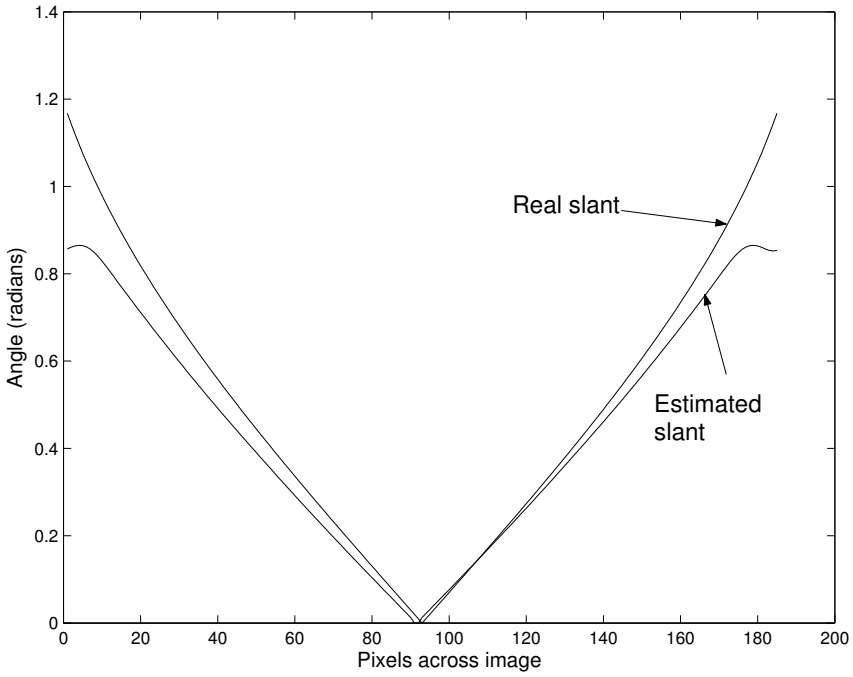
It was noted that Super and Bovik's algorithm performed better where the texture changed abruptly in frequency. This is because in the new algorithm, the window used to extract patches encompasses *part of both* frequencies on either side of the transition. This would not be a problem so long as abrupt transitions do not occur, and even if they do, the new algorithm simply estimates slant and tilt to be values that are between the true values either side, which makes good sense under the circumstances. Real life textures do not generally display stark transitions in slant, for example any smooth surface would not produce this problem.

A real image

Super and Bovik's algorithm was tested on the image of the golfball previously seen in Section 3.2.2. When encountered previously, the image of the golfball displayed a sharp border between the texture and background, a result of setting the background to black. Super and Bovik's algorithm did not work when used with this



(a) Input image



(b) Real and estimated slant

Figure 33: Super and Bovik's algorithm tested on a synthetic cylinder. The estimated slant tends to undershoot the real slant, as shown in (b)

image, because the algorithm is not designed to be used with such segmented images. Therefore, the input image is the original image before segmentation, shown in Figure 34. Values for slant were set to be zero in the background as a final step, since it does not make sense to calculate these values outside of the golfball texture. Again, the filter bank was carefully chosen to give a sensible, uniform coverage of the frequencies present.

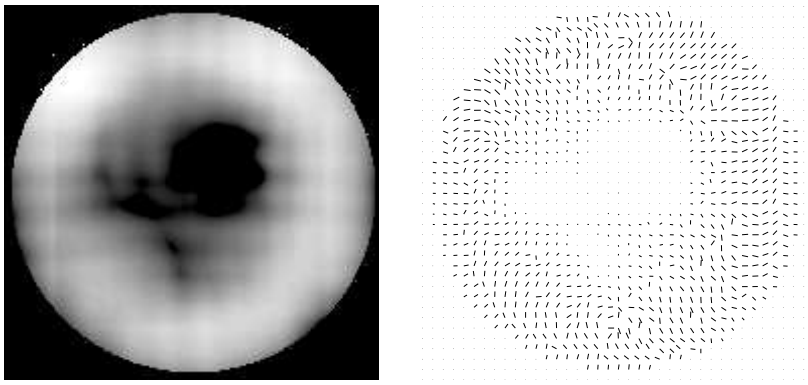
The calculated slant is shown in Figure 35(a). The black patch in the center is

Figure 34: Input image

an artifact of the algorithm. In areas where the texture is close to frontal $M_s \times m_s$ may be larger than $M \times m$, even if marginally. Remembering that M_s and m_s are the frontal moments, we expect them to be always smaller than all other M and m since the texture is contracted in all areas except where it is frontal, increasing the frequencies and increasing the moments. However due to noise and small amounts of inhomogeneity it turns out that $\frac{M_s m_s}{Mm}$ may be slightly larger than unity. Slant is calculated using $\sigma = \arccos(\sqrt{\frac{M_s m_s}{Mm}})$ which results in a complex value for slant. Super and Bovik do not specify what to do in these cases; the most sensible step is to set slant to zero since $\frac{M_s m_s}{Mm}$ is most likely to be larger than unity at frontal locations. This is the reason for the black patches in Figure 35(b) and is a disadvantage of the algorithm. In contrast, the new algorithm never returns complex values, but instead returns the best guess for slant from the set of real angles. The needle diagram in Figure 35(a) shows that tilt is calculated only to a fair accuracy; the needles generally point outward but errors are seen in patches.

A quantitative comparison of the two algorithms is given in Figure 36(a) where the slant along a profile is shown. The values from Super and Bovik's algorithm clearly show inaccuracies where the texture is almost frontal, and complex values for slant were calculated. Also the slant tends to drop off as the true slant increases. This phenomenon was seen in the earlier example of the synthetic cylinder, and is also in examples in Super and Bovik's paper [79].

In comparison, slant values from the new algorithm tend to be more accurate. The 'blockiness' is due to the spacing between patches of texture, and may be



(a) Slant estimated by Super and Bovik's algorithm

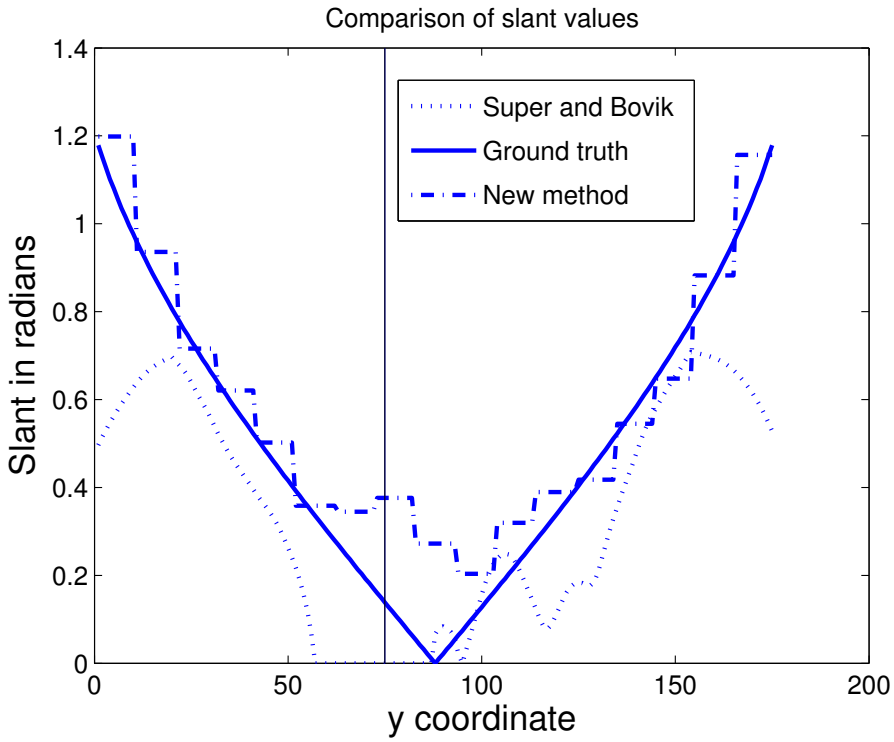
(b) Needle diagram

Figure 35: Results from Super and Bovik's algorithm tested on the golfball image.

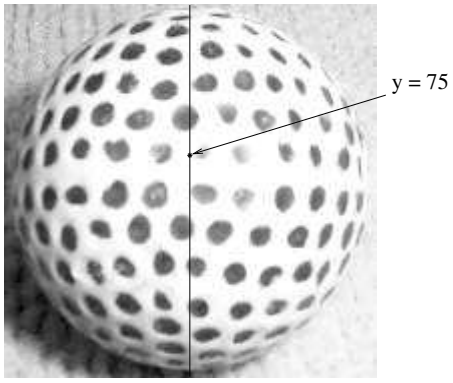
reduced by taking closer samples, the cost of which is an increased processing time. The greatest error is seen near the y coordinate equal to 75, depicted with a vertical line in Figure 36(a). This error is due to the specularity on the golfball where the texture is not visible. Figure 36(b) shows the profile on the golfball, and labels the point where $y = 75$.

Speed of algorithm

Both algorithms were implemented in MATLAB version 6.5 [31], running under Linux on an Intel Pentium 4 2400 MHz CPU. A general observation is that both algorithms were completed within a reasonable amount of time, although this was highly dependent on the choice of filters and window sizes and the efficiency of the algorithms. Super and Bovik's algorithm processed the image of the golfball in around 32 seconds, however this involved returning a value of slant and tilt for all pixels. In theory, this resolution could have been reduced for a faster processing time. Most of the algorithm's time was spent creating the Gabor filters and convolving using them. To increase speed, convolution was implemented as a multiplication process in the spectral domain. As mentioned earlier, the new algorithm was not run over every pixel since this would take too long and was not necessary for shape estimation as long as the number of pixels between estimated slant and tilts were not too large. It was possible to keep the running time to around 7 seconds when



(a) Slant along profile



(b) Profile across golfball

Figure 36: Comparison of calculated slant

the slant and tilt were calculated at intervals of ten pixels. This spacing was more than sufficient to recover the shape of the golfball.

3.3.3 Discussion

After testing both algorithms on several images, the following observations were made.

- Super and Bovik's algorithm is highly non-robust in areas where the texture appears close to frontal. As previously seen, small amounts of noise can cause the algorithm to return complex, and hence unusable, values for slant.

- While neither algorithm can avoid the π-shift tilt ambiguity, Super and Bovik's algorithm has an additional ambiguity due to the mathematical equations used to calculate tilt. While the π-shift ambiguity is solvable by methods such as suggested by Lobay and Forsyth [53], the additional ambiguity (seen as the $\pm$ sign in Equation 10) requires extra information, according to Super and Bovik's paper [79]. This information may come from knowledge of the true surface orientation at one point, another process such as shape-from-shading, or constraining the type of surface. In my particular experiments with Super and Bovik's algorithm, the additional ambiguity had little effect due to the textures used; in the case of the two synthetic examples and the one real example, the tilt angle τ ought to be equal to α, the angle about which the smallest moment is achieved — as is the case with a sinusoidal (stripy) texture and spotted texture. In experiments λ (from Equation 10) was found to be always close to one, and thus

$$\begin{aligned} \tau &= \alpha \pm \tfrac{1}{2} \arccos \lambda \\ &\approx \alpha \pm \tfrac{1}{2} \arccos(1) \\ &= \alpha \pm 0 \\ &= \alpha \end{aligned}$$

  as expected. However for general textures this ambiguity would need to be solved by some means.

- As pointed out earlier, Super and Bovik's algorithm tends to return slant values that are lower than real slant. Reasons for this have been discussed.

3.4 Conclusions

A new algorithm has been presented that solves shape-from-texture for homogeneous, stationary textures viewed orthographically. Tests using the Brodatz database of textures were performed to verify that for local patches of texture the algorithm behaved as theory suggests. The results were quantitatively good, with errors attributed to textures featuring only high frequencies, since these textures lost a lot of detail due to the slanting process, and textures that did not satisfy the assumption of homogeneity.

For the algorithm to be tested on real images border effects were taken into consideration. It was shown that the algorithm returns along the texture border a slant angle of $\frac{\pi}{2}$ and a tilt angle that is orthogonal to the border. This made the algorithm usable for surfaces whose borders slant away from the viewer, as is the case for many textured objects.

When the algorithm was tested on real images, it was found again to give good quantitative results for the image of a golfball. For the test image of a cheetah it was noted that the algorithm's good reconstruction was the result of two behaviours: (i) away from the texture border, the algorithm performed as expected in the same fashion as was verified by the earlier trials on Brodatz textures, and (ii) using the texture border to enforce a slant value of $\frac{\pi}{2}$ and tilt angle that is orthogonal to the border.

When the new algorithm was compared to Super and Bovik's algorithm, a test image revealed that while the latter performed better at borders between different slant values, the new algorithm was more accurate everywhere else. When Super and Bovik's algorithm was applied to the image of the golfball, it was found to give complex values for slant in areas near the frontal texture due to noise and a small amount of inhomogeneity. In comparison, the new algorithm gave real, hence usable, values for slant. The new algorithm also gave more accurate results for slant and tilt. The main disadvantage of Super and Bovik's algorithm is that it requires *another* source of shape information other than the texture to resolve an ambiguity in the tilt angle, whereas the new method does not.

Chapter 4

Finding The Affine Transformation

The method in the previous chapter implicitly solves for only two of the components of the affine transformation between textures (the slant and the tilt). It was assumed that no scaling was involved since there are no perspective effects for the orthographic view. The rotation component was also ignored since the texture was assumed to be stationary. Translation effects are not considered because only local patches of texture are compared. However, if the texture is not assumed to be stationary or viewed orthographically, the scale and rotation components must be solved along with the slant and tilt.

This chapter presents a method that solves the affine transformation (ignoring translation components) between two images of the same texture. This method was developed to be used as part of the shape-from-texture algorithm that will be described in Chapter 5. This chapter outlines the problem of calculating the affine transformation between textures and shows why previous techniques cannot be used. I also discuss the many uses of this technique, giving examples where it is used in object re-texturing and the placement of an object in a scene.

4.1 Outline of problem

This work addresses the following problem: we have two images of some texture, but they are not necessarily the same portion of the texture, and they are viewed from different angles. Either image may be affine transformed such that its texture has the same orientation as in the other image. The aim is to calculate this affine transformation. (See Figure 37).

(a) First input image (b) Second input image (c) First image transformed

Figure 37: Computing the affine transformation between two textures. The image in (a) has been transformed to give (c) such that the texture is viewed with the same orientation as in (b).

A different, but related problem exists: finding the affine transformation between two images of the same planar object or surface. A common approach to this problem is to identify "interesting" points (such as Harris points [21] or SIFT points [58]) in both images and to match corresponding points after assigning some feature vector to each point. The affine transformation is then found such that it minimises some criterion such as distance between points after one set has been transformed. This approach cannot be used in this current work with textures since, in general, we are dealing with two different portions of the texture, and hence there is no guarantee that robust point correspondences can be established. For example see Figures 38(a) and (b) which show two different portions of the same texture. In Figures 38(c) and (d), the images have been superimposed with crosses that show detected Harris feature points. If the points in the first image are matched with points in the second image, they will not be related by some affine transform and so the feature point method would not work in this example. Feature matching

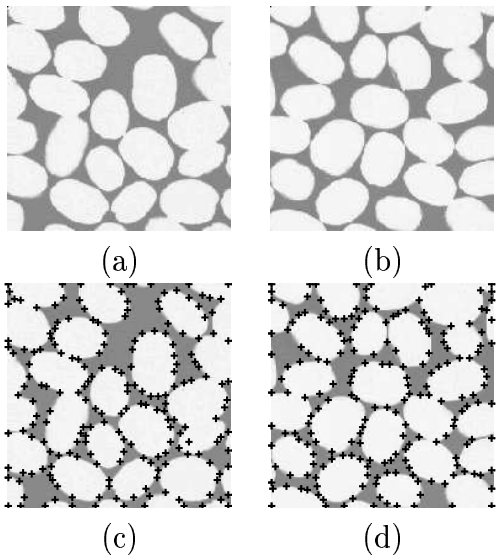


Figure 38: Two different portions of the same texture. Figures (c) and (d) show where features have been found in (a) and (b) respectively.

is also inherently difficult on textures because, by definition, they have repeating structures.

This chapter outlines a new method that is rather different to feature methods; firstly, both images of the texture are transformed into an isotropic state so that they differ only by a rotation and scale. The required rotation is calculated which then allows us to construct the affine transformation up to scale. Since it does not make sense to consider the translation part of the affine transformation, this component does not need to be calculated. The method does not require that we input corresponding regions of texture i.e. the pair of input texture regions may be of any shape and may contain a different number of texels.

4.2 Relevant work

A number of methods have been used to estimate the affine transformations between textures. Many of these are frequency domain techniques that essentially track the movement of spectral peaks to recover the transformation. Ribeiro, Worthington and Hancock [72] match corresponding peaks in two spectral images according to their ordering of energy amplitudes. In practice however, this process of matching

corresponding peaks is made difficult by noise and aliasing, which alter the magnitude, and hence the ordering, of peaks. Malik and Rosenholtz [61] developed a method for estimating the affine transformation between two nearby patches of texture on a surface. The method uses differential analysis to produce a system of constraints that is solved iteratively. Their method relies heavily on there being little change between texture patches and therefore cannot be used to solve for example the case shown in Figure 37. The new method presented in this chapter will work even if there is a large difference between the two textures. Irani et al. proposed a method for calculating the affine transformation between planes [32], and like this new method does not require that point correspondences are explicitly computed. However their method requires that the input images are of exactly the same plane, and hence cannot be used in this current work. Krumm and Shafer [46] find the transformation between textures by testing every combination of slant and tilt from a discrete set. Their method is computationally expensive, especially since a fine search in the slant-tilt space is required for an accurate estimated transformation. Sato and Cipolla [73] developed a method that uses the first, second and third moments of the edge orientation to estimate the affine transformation. In their method, features must be detected, but correspondences between the features need not be made. The new method presented in this chapter does not require features to be detected at all.

4.3 The method

Starting with two images of the same texture that has undergone two different affine transformations (for example see Figures 39(a) and 39(b)), the method aims to find the affine transformation between the two images. The main idea is to firstly transform each image such that the texture is isotropic. This idea was influenced by the work of Schaffalitzky and Zisserman [75], who transformed textured regions to an isotropic state in order to assign an affine invariant feature descriptor.

Textures are made isotropic by firstly calculating the second moment matrix of

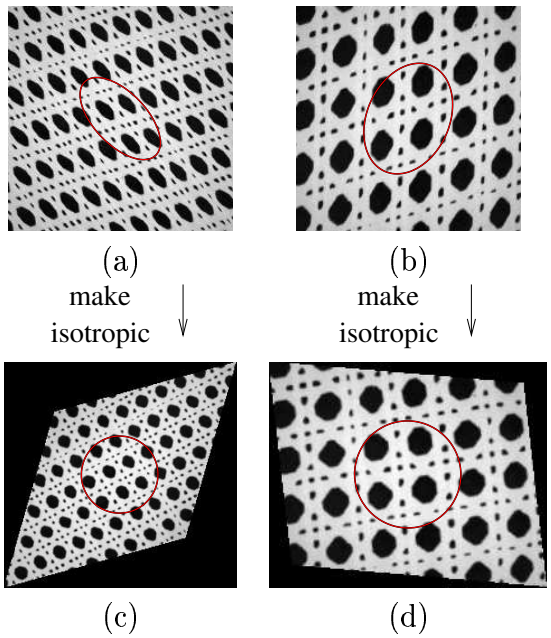


Figure 39: Basis of algorithm. The original images in (a) and (b) are made isotropic to give (c) and (d) respectively. The latter pair differ by a scale and rotation.

each texture. The second moment matrix of a region Ω with intensity I is given by

$$\mathbf{M} = \int_\Omega \nabla(I) \otimes \nabla(I) \frac{dx\,dy}{|\Omega|} = \int_\Omega \begin{pmatrix} I_x\,I_x & I_x\,I_y \\ I_x\,I_y & I_y\,I_y \end{pmatrix} \frac{dx\,dy}{|\Omega|}$$

where I_x and I_y denote derivations in x and y respectively. This matrix may be represented as an ellipse, as superimposed onto Figures 39(a) and 39(b). The ellipse is locus of distances $\mathbf{x}$ from the centroid that satisfy $\mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{x} = k$ where k is some constant.

The transformation $\mathbf{A}$ that makes the region isotropic is required to satisfy $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{M}$. Since $\mathbf{M}$ is always positive definite, one solution for $\mathbf{A}$ is the square root of $\mathbf{M}$ given by

$$\mathbf{A} = \mathbf{V} \begin{pmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \end{pmatrix} \mathbf{V}^{\mathrm{T}} \tag{11}$$

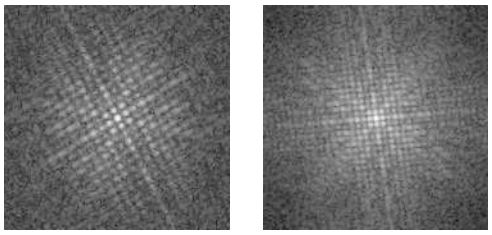
where λ_1 and λ_2 are the eigenvalues of $\mathbf{M}$, and $\mathbf{V}$ is the matrix whose columns are the corresponding eigenvectors of $\mathbf{M}$.

Figures 39(c) and 39(d) show the isotropic versions of the textures, where as expected the second moments are now represented as superimposed circles.

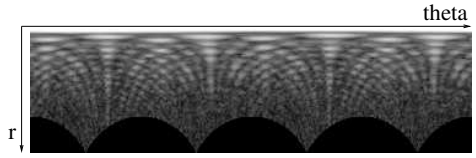
4.3.1 Finding the rotation

The textures in Figures 39(c) and 39(d) differ by a scale and rotation, and the next step of the algorithm is to calculate the rotation. This is done by computing the Fourier transform of each isotropic texture (to eliminate any translation effects), as shown in Figures 40(a) and (b) where the Fourier transforms of Figures 39(c) and 39(d) are displayed. Recall that rotating any image results in an equivalent rotation in its Fourier transform, but scaling an image results in the inverse scaling of its Fourier transform. Therefore Figures 40(a) and (b) differ by only a scale and rotation.

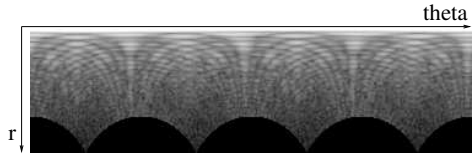
Each Fourier transform is then converted to polar coordinates, as shown in Figures 40(c) and (d), so that the rotation between the Fourier transforms is seen as a translational shift. The next step is to take the mean along each column of



(a) (b)



(c) Polar transformed version of (a)



(d) Polar transformed version of (b)

Figure 40: Finding the rotation angle. Figure 40(a) and (b) show the Fourier transforms of Figures 39(c) and (d) respectively. Their polar transforms, shown in Figure 40(c) and (d) exhibit a translational shift with respect to each other, as well as a scaling in the direction r.

Figures 40(c) and (d) to eliminate the effect of scale. This produces two 1-D signals that are similar, except that one signal is shifted in the θ-direction with respect to the other signal. A correlation is used to calculate the shift between the two signals which then gives the rotation angle between the textures.

4.3.2 Finding the scale

As mentioned, this work does not include calculating the scale since it is not required for the shape-from-texture algorithm that will be described in Chapter 5, however in this section we outline ways that the scale may be calculated, without having tested all of these methods.

One obvious approach might be to use the second moments of the two isotropic images. If the second moment matrices of the images are $\begin{pmatrix} S_1 & 0 \\ 0 & S_1 \end{pmatrix}$ and $\begin{pmatrix} S_2 & 0 \\ 0 & S_2 \end{pmatrix}$ one might imagine that the scale factor between the two images is $\frac{S_1}{S_2}$ or $\frac{S_2}{S_1}$ however in practice this behaviour is rather affected by illumination; increasing the brightness of an image increases the magnitudes of the peaks in its Fourier transform. This in turn increases the second moments. Blurring also affects the second moments since it decreases the magnitudes of the high frequency components, decreasing the second moments.

Another approach might be to use log-polar transforms for the estimation of scale. However, an experiment with this approach implied that it was not robust; the responses of log-Gabor filters [14] were used to try to estimate the scale via the frequency domain. The problem was that it was difficult to get sufficient resolution in the frequency domain from the filters. A very large number of narrow bandwidth filters would have to be applied, and even then if the actual scale change fell between filter centre frequencies an inaccurate estimate of scale would result.

One approach which might be applied is influenced by Lindeberg [51], who uses a bank of normalised Laplacian filters on an image and finds which gives the strongest response. The size of this filter corresponds to the size of the main underlying structure of the image. For example Lindeberg uses this method to detect the size of blobs in an image [52]. This is similar to Lowe's use of difference-of-Gaussian filters [58] to establish the scale associated with each feature. The shape of the difference-of-Gaussian is almost identical to the Laplacian although they differ by a scale factor — which actually corresponds to the scaling factor proposed by Lindeberg that 'normalises' his filters. While this approach has not been used in this work to estimate the scale factor, results in the literature suggest

that it would be successful. This is a natural area of future work.

4.3.3 Summary of algorithm

To summarise, the algorithm that estimates the affine transformation between textures up to scale is given by the following steps.

1. Begin with two images I and J displaying the same texture that differs by an affine transformation. Calculate the transformation $\mathbf{T_1}$ required to convert I into its isotropic version I_{iso}. (See Figure 41.)

2. Similarly, calculate $\mathbf{T_3}$ that converts J into its isotropic version J_{iso}.

3. Calculated the rotation angle between I_{iso} and J_{iso} using the method above. Set $\mathbf{T_2}$ to be the matrix that rotates by this angle.

4. Then, according to Figure 41, the transformation from I to J is $\mathbf{T_1 T_2 T_3}^{-1}$.

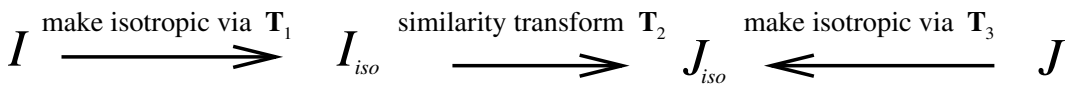


Figure 41: Finding the affine transformation between two images i and j of the same texture.

4.3.4 Details of implementation

The following steps were required to make the algorithm robust.

- Before calculating the Fourier transforms of I_{iso} and J_{iso} these images were windowed with a 2-D circular Gaussian to suppress the effect of the border of the texture. For example, differences in pixel values between the right and left edges manifest as high frequency components in the Fourier Transform, if the windowing process is not implemented.

- After transferring the Fourier transforms to polar coordinates, the top rows (corresponding to the low frequency data in the original Fourier transforms) were deleted to stop the low frequency data from dominating the calculation

of the mean along each column. Precisely what defines a frequency as 'low' has not been thoroughly investigated. For my experiments, the first 20% of rows were deleted since this gave a satisfactory performance.

- The scalloped bottom edge of the polar transformed Fourier Transforms seen in Figure 40(c) and (d) is accounted for simply by not including these values in the calculation of the mean in each column.

- Finally, after computing the mean in each column to produce a signal for each Fourier Transform, these signals are centered at zero and scaled to the same range before correlating them.

Due to the symmetric nature of the Fourier transform, the rotation can only be calculated up to a shift by π. This manifests as an initial rotation of π applied to the image i as part of the estimated affine transformation. This can in theory be avoided by producing two versions of the transformed i, one with the initial rotation by π and one without. These two versions may be compared to the destination texture j to see which is correct. For the work described in Chapter 5, this rotation by π was never an issue, since the method was altered to be used on individual texels instead of patches of texture, thereby avoiding the Fourier transform.

Ambiguities

In some cases, an ambiguity occurs when calculating the similarity transformation, however it turns out that this ambiguity is inherent in the task of calculating the transformation from one texel to another when scaling and slanting are involved. The ambiguity is not due to the method used to solve the task. The ambiguity occurs whenever the isotropic version of a texel exhibits rotational symmetry, as demonstrated by the example in Figure 42. Even if the texel in Figure 42(a) is known to be frontal, and the texel in Figure 42(b) is some affine transformed version, we cannot tell whether (i) the frontal texel has simply rotated about its normal axis and decreased in size due to increased distance from observer, or (ii) the frontal texel has slanted in the vertical direction, whilst keeping its distance from the observer constant, or (iii) the frontal texel has undergone some other combination of rotation

(a) Some frontal texel (b) An affine transformed version

Figure 42: Ambiguity in affine transformation between texels. Even if the texel in (a) is known to be frontal, and the texel in (b) is known to be some affine transformed version of (a), it is still impossible to calculate the affine transformation between the two.

and scaling. Therefore there are infinite possibilities for the affine transformation between these two texels, however this is only the case for ellipses.

Other examples give a finite number of solutions. The rectangle in Figure 43 gives four possibilities for the affine transformation from the texel in (a) to the one in (b). This is because without more distinguishing features it is impossible to tell which of the four sides of the rectangle in (a) correspond to each of the four sides in (b). In summary, the ambiguity discussed here occurs whenever the isotropic

(a) Some frontal texel (b) An affine transformed version

Figure 43: An ambiguity with finite possibilities

version of the texel exhibits rotational symmetry and the frontal texel may undergo any combination of rotation, scale and slant. Some other source of information or constraints may help.

4.4 Results

4.4.1 Tests with Brodatz textures

To test the algorithm, pairs of images were created using the Brodatz textures [5]. Random affine transformations were applied to the textures to create the pairs shown in the first two columns of Figure 44. Using these pairs of images, the

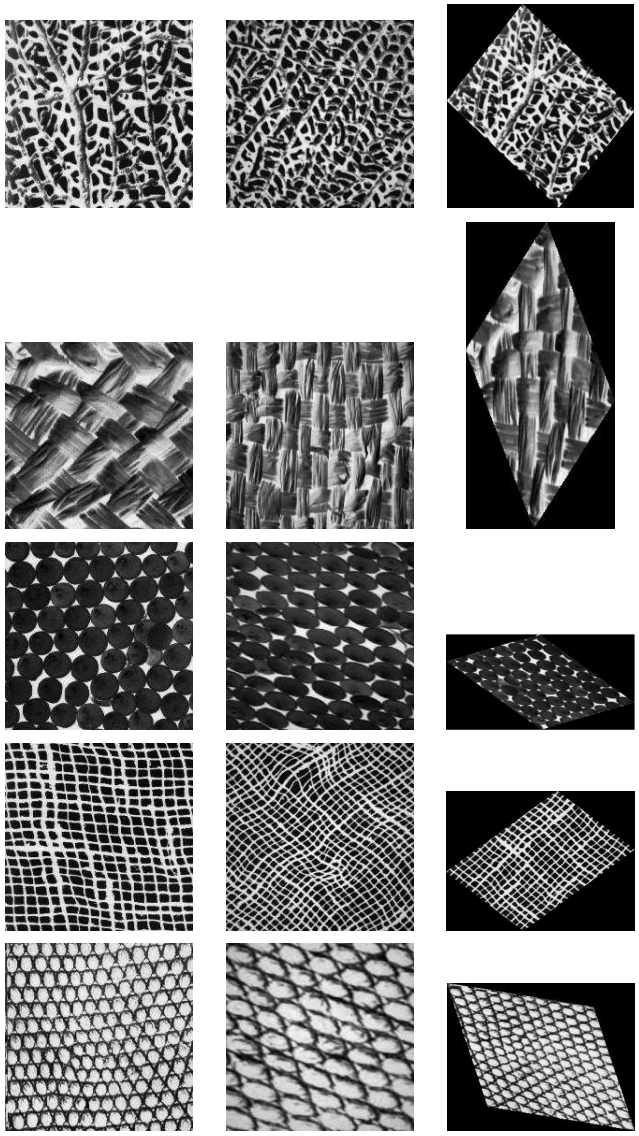


Figure 44: Results of tests on Brodatz textures. The first and second columns are input images. The third column shows the affine-transformed version of the first column such that the texture resembles the second column, up to scale.

algorithm estimated the transformation from the first image to the second. The

last column of Figure 44 shows the transformed versions of the first column, such that the textures approximate those in the second column, up to scale.

The estimated affine transformations were then compared to the true affine transformations. An affine transformation $\mathbf{T}$ may be decomposed via singular value decomposition [55] into

$$\begin{aligned} \mathbf{T} &= \mathbf{U}\,\mathbf{D}\,\mathbf{V}^{\mathrm{T}} \\ &= (\mathbf{U}\,\mathbf{D}\,\mathbf{U}^{\mathrm{T}})\,\mathbf{U}\,\mathbf{V}^{\mathrm{T}} \end{aligned} \tag{12}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and $\mathbf{D}$ is a diagonal matrix. The decomposition may be interpreted as such: the image is firstly rotated by $\mathbf{U}\,\mathbf{V}^{\mathrm{T}}$. The image is then rotated by $\mathbf{U}^{\mathrm{T}}$, stretched in the coordinate directions by $\mathbf{D}$ then rotated back by $\mathbf{U}$. The net effect is a slant operation in some tilt direction followed by an isotropic scale. If $\mathrm{rot}(x)$ is a matrix that rotates by x we can rewrite this as

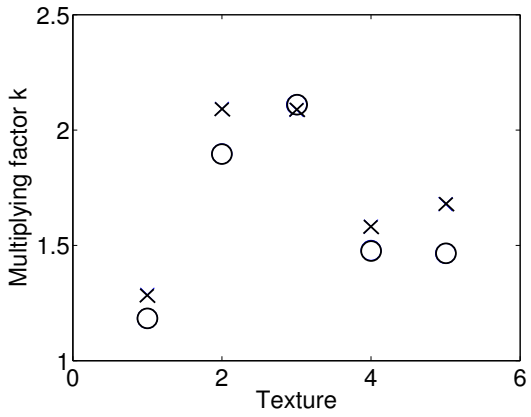
$$\mathbf{T} = rot(-\tau) \begin{pmatrix} k & 0 \\ 0 & 1 \end{pmatrix} rot(\tau)\, rot(\theta) \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \tag{13}$$

where s is a scaling factor, k is a multiplying (contraction) factor related to slant, τ is the tilt direction and θ is the initial rotation angle.

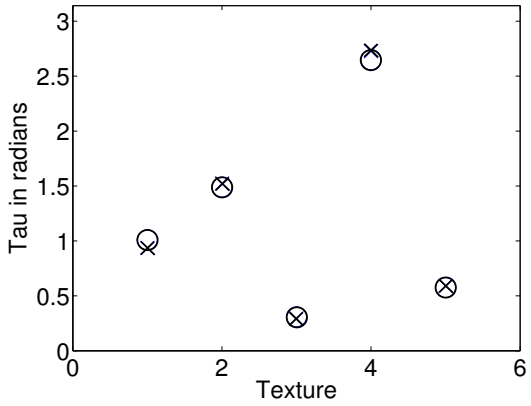
For this particular experiment, the estimated transformation is only computed up to scale, and so both it and the real transformation were scaled such that the first entry in their matrices was unity. The matrices were then decomposed via singular value decomposition to give values for k, τ and θ. The errors are shown in Figure 45. A couple of points are noted: firstly, one expects some error in the estimated affine transformation due to the slightly non-homogeneous nature of the textures. Secondly, the third example shows a large error in θ. This is due to the fact that the texels roughly form lines in two directions, but the algorithm chose the wrong direction to line up the texels. However this is an error that even a human might make, given the same task, as it is impossible to distinguish between the two directions of lines, without more detail in the texels.

4.4.2 Tests with real images

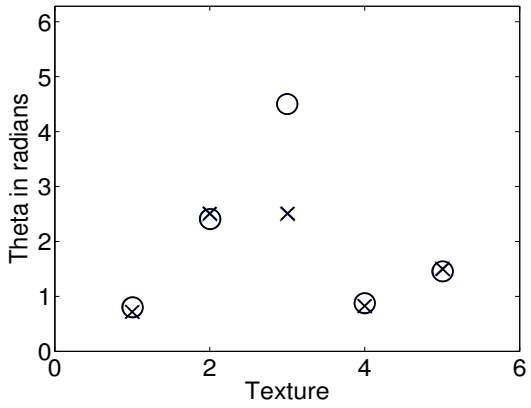
The algorithm was also tested on real images. Figure 46 shows the objects that were photographed. Close up pictures of textures were taken from different angles as



(a) Real and estimated k



(b) Real and estimated τ



(c) Real and estimated θ

Figure 45: Errors in affine parameters. The crosses denote estimated values and the circles denote real values.

seen in the first two columns of Figure 47. The third column shows the transformed version of images from the first column, such that they resemble the textures seen

in the second column. The images show good qualitative results.

(a) drain cover

(b) basket

(c) air-conditioning outlet

(d) necktie

(e) dish cloth

Figure 46: Objects that were photographed.

4.4.3 Simple applications of the method

As mentioned previously, the method may be used in a number of applications apart from shape-from-texture; these include object re-texturing and placing planar objects in a scene. In this section we show an example of each.

First we show the example of re-texturing. If the only input we have is the textured cube shown in Figure 48(a) and a sample of the frontal texture shown in Figure 48(b), then it is possible to re-texture the cube. Using the method, the transformation from the frontal texture to each of the three visible sides of the cube can be estimated. Then, given some other frontal texture (such as the samples shown in Figures 49(a) and (b)), we can use the same three transformations to place the new texture on the cube. The resulting cubes are shown in Figure 49(c) and (d).

The next application demonstrated here is placing a planar object in a scene. For this example, we wish to place an embroidered motif onto a dish cloth. Figure 50(a) shows the motif. The dish cloth was the last example in Figure 47. In Figure 47 we

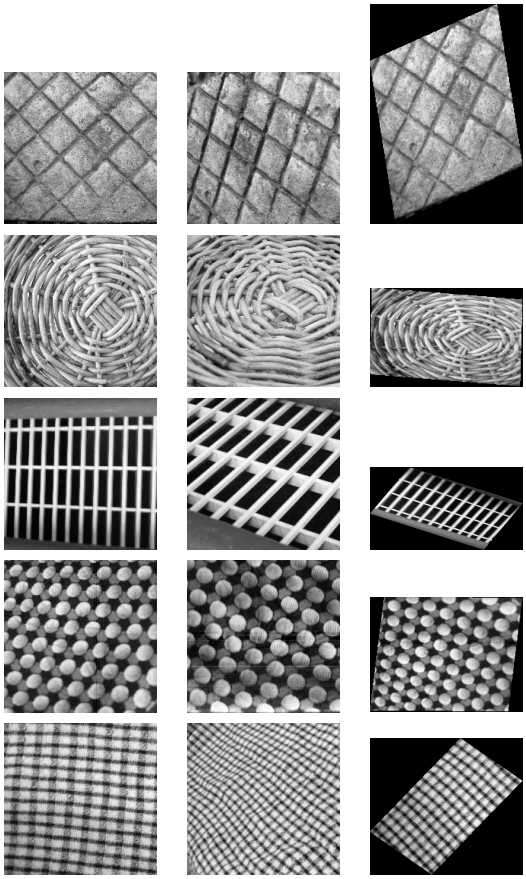
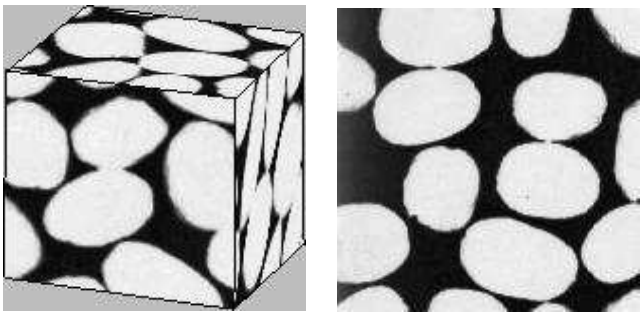


Figure 47: Results from real images. The first and second columns show the input images. The third column shows the transformed versions of the first column so that the texture matches that in the second column.

saw the dish cloth when viewed frontally (in the first column), as well as the dish cloth viewed with some slant angle and rotation (in the second column). Using the



(a) (b)

Figure 48: Input for re-texturing

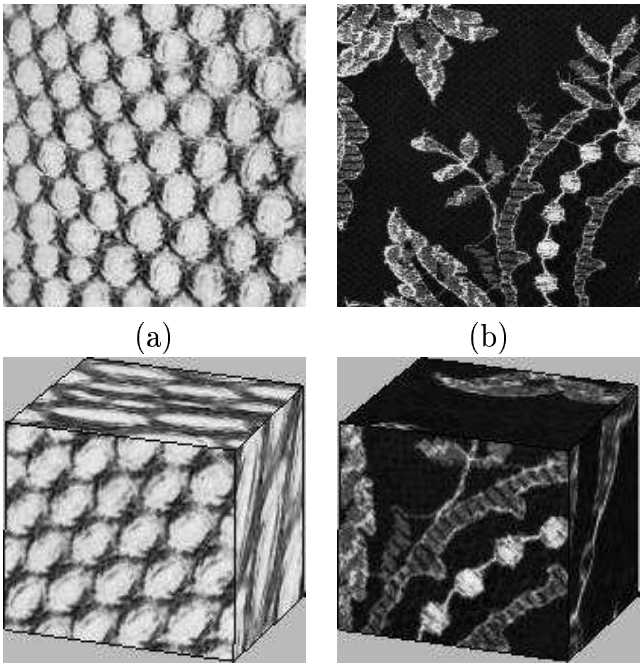


Figure 49: Re-texturing a cube. (a) and (b) show samples of textures used in the re-texturing. The results are given in (c) and (d).

algorithm, the affine transformation $\mathbf{T}$ from the frontal view to the slanted view was calculated. Since we have a frontal view of the embroidered motif, it may also be transformed by $\mathbf{T}$ to give the same orientation as the slanted dish cloth; the result is given in Figure 50(d).

(a) The original motif

(b) A slanted and rotated view of the dish cloth

(c) The transformed motif

(d) The motif placed on the dish cloth

Figure 50: Placing an embroidered motif on the dishcloth.

4.5 Discussion and conclusions

This chapter has presented a new method to estimate the affine transformation between pairs of textures. Previous methods are susceptible to noise, or only usable in a very restricted set of cases, or too computationally expensive. The new method has been shown to give good quantitative results when tested on a set of images created using the Brodatz textures. Real images gave good qualitative results.

This method should not be affected by contrast and illumination variations in the images since neither affect (i) the ability to transform a texture into its isotropic version, or (ii) the ability to calculate the rotation between the isotropic versions. This assertion could be verified more thoroughly in future work.

Chapter 5

A General Shape-from-Texture Method

5.1 Introduction

A shape-from-texture method in one of its most general forms is presented in this chapter. Previous shape-from-texture methods assume that the texture is constrained by one or more of the following properties: homogeneity, isotropy, stationarity, or orthographic view. In this work, none of these assumptions are made. The frontal texture is not presumed to be known a priori, or from a known set, or even present in the image. Instead, surface smoothness is assumed, and the surface is recovered via a consistency constraint. The key idea is that the frontal texture is estimated, and a correct estimation leads to the most consistent surface. In addition to surface shape, a frontal view of the texture is also recovered. Results are given for synthetic and real examples.

5.1.1 Basic idea

Suppose that a texture is composed of individual texels, and that one texel is known to be viewed frontally in an image. Other texels may be slanted away from the camera, and lie at a different distance from the camera than the frontal texel. Slanting the texel will cause foreshortening in the tilt direction and changing

distance from the camera will result in a change of scale. The appearance of any texel will be related to the appearance of the frontal texel by a geometric transform, which locally can be modeled as an affine transformation. By measuring at each texel the affine transformation that relates that local texel to the known frontal texel, we may determine the orientation of the surface at that point in the image. From this information it is possible to reconstruct the surface.

Sometimes however, no frontal texel is visible, or identifiable in the image. Our approach is to search over all possible texels to determine which is frontal. Each hypothesized frontal texel leads to an estimate of the orientations across the surface. However, the key observation of this work is that an incorrect hypothesis of the frontal texel leads to estimates of surface orientation that are inconsistent, and in fact cannot be realized by a reconstructed surface. Therefore, a search over all possible frontal texels leads to a unique consistent estimate of frontal texel, and hence surface shape. We demonstrate that this is feasible, even if only the orientation cue is used to calculate surface consistency. The search is carried out using the Levenberg-Marquardt method, and does not require that the frontal texel be visible in the image.

It has been shown by Forsyth [16] that under orthographic projection, the frontal texel can be determined regardless of the shape for the surface. However, his approach cannot be applied in the case where individual texels may be scaled due to perspective views or for any other reason. The contribution of this work is to provide a shape-from-texture method that does not assume that the texel is homogeneous, isotropic, or stationary, or that an orthographic view is employed. To my knowledge, no algorithm exists for this level of generality.

5.2 Overview of the new method

We begin with the image of a smooth textured surface. The shape of the surface is found by firstly searching for the frontal texel, as described in Section 5.2.1. The search aims to find the texel that leads to the best surface consistency measure, described in Section 5.2.2. Once the frontal texel has been found, surface shape is

estimated using the procedure in Section 5.2.3.

5.2.1 Searching for the frontal texel

Write $\mathbf{T}_{f\to i}$ for the affine transformation from the frontal texel f to the texel i. The aim is to determine $\mathbf{T}_{f\to i}$ for all i, as these matrices may be decomposed to give the orientation and distance of each i.

Since we do not know the appearance of the frontal texel, we cannot compute the matrices $\mathbf{T}_{f\to i}$ directly from the image. Instead, we can write

$$\mathbf{T}_{f\to i} = \mathbf{T}_{j\to i}\mathbf{T}_{f\to j} \tag{14}$$

where $\mathbf{T}_{j\to i}$ is the transformation from a reference texel j, arbitrarily chosen as a texel from the image (or some affine transform of it), to texel i, and $\mathbf{T}_{f\to j}$ gives the affine transformation from f to j. The matrices $\mathbf{T}_{j\to i}$ may be computed from the

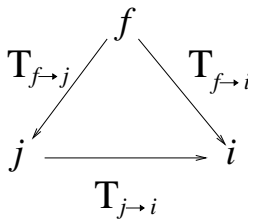


Figure 51: Transformations between texels

image, it is then a matter of searching for the affine transformation $\mathbf{T}_{f\to j}$ such that the most consistent surface is achieved from the matrices $\mathbf{T}_{j\to i}\mathbf{T}_{f\to j}$. The measure of surface consistency used in this work is given in the next section. The matrix $\mathbf{T}_{f\to j}$ may be decomposed via RQ decomposition [23] to give $\mathbf{T}_{f\to j} = \mathbf{RQ}$ where $\mathbf{R}$ is a lower triangular matrix and $\mathbf{Q}$ is orthogonal. We are free to replace $\mathbf{T}_{f\to j}$ with $\mathbf{T}_{f\to j}\mathbf{P}$ where $\mathbf{P}$ is a rotation matrix, as an initial rotation does not alter the interpretation of the surface orientation, and hence does not affect the measure of surface consistency. We can choose $\mathbf{P} = \mathbf{Q}^{\mathrm{T}}$ so that we need only search lower triangular matrices[1] as candidates for $\mathbf{T}_{f\to j}\mathbf{P}$.

[1] If $\mathbf{Q}$ is not orthonormal, then $\mathbf{Q}^{\mathrm{T}}$ will not be a pure rotation, however scaling factors are irrelevant as seen in the next paragraph.

We can further reduce the search space by the fact that the surface consistency measure does not depend on scale, so that we only need to search for matrices of the form $\begin{pmatrix} 1 & 0 \\ c & b \end{pmatrix}$. Using a Levenberg-Marquardt search [62], the values of c and b that give the most consistent surface are determined, say c_f and b_f. The transformations $\mathbf{T}_{f\to i}$ are then given by $\mathbf{T}_{j\to i}\begin{pmatrix} 1 & 0 \\ c_f & b_f \end{pmatrix}$.

5.2.2 Surface consistency measure

To test whether the transformations $\hat{\mathbf{T}}_{f\to i}$ (where the $\wedge$ denotes the fact that $\hat{\mathbf{T}}_{f\to i}$ is only a candidate) give a consistent surface, the following procedure is employed. First, calculate the surface normals that arise from the transformations $\hat{\mathbf{T}}_{f\to i}$. This is done by decomposing $\hat{\mathbf{T}}_{f\to i}$ via singular value decomposition such that

$$\begin{aligned} \hat{\mathbf{T}}_{f\to i} &= \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}} \\ &= \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}(\mathbf{U}\mathbf{V}^{\mathrm{T}}) \end{aligned} \tag{15}$$

where $\mathbf{D}$ is a diagonal matrix satisfying $d_{11} < d_{22}$, and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices. The decomposition is interpreted as follows.

1. The texel is rotated by the matrix $\mathbf{U}\mathbf{V}^{\mathrm{T}}$.
2. The image is now rotated by $\mathbf{U}^{\mathrm{T}}$, stretched in the coordinate directions by $\mathbf{D}$ and then rotated back by the rotation $\mathbf{U}$. The net effect of this is to scale it along two axis directions whose orientation is determined by $\mathbf{U}$.
3. Thus, $\mathbf{U}$ gives the tilt direction.
4. The amount of scaling (given by the entries of $\mathbf{D}$) indicates the degree of slant.

Note that the degree of tilt or slant of the image does not depend on $\mathbf{V}$. Hence $\mathbf{V}$ may be ignored. Alternatively, we can multiply on the right by any rotation, without changing the result of the slant or tilt calculation. If the slant and tilt of the surface patch are given by σ and τ respectively, then

$$\begin{aligned} \cos(\tau) &= u_{11}, \\ \sin(\tau) &= u_{12} \end{aligned} \tag{16}$$

and

$$\begin{aligned} \cos(\sigma) &= d_{11}/d_{22}, \\ \sin(\sigma) &= \pm\sqrt{1-\cos^2(\sigma)}. \end{aligned} \tag{17}$$

Equation 16 comes from the fact that

$$\mathbf{U} = \begin{pmatrix} \cos(\tau) & \sin(\tau) \\ -\sin(\tau) & \cos(\tau) \end{pmatrix} \tag{18}$$

and Equation 17 arises from

$$\mathbf{D} = \begin{pmatrix} \cos(\sigma)r & 0 \\ 0 & r \end{pmatrix} \tag{19}$$

where r is the scaling factor due to the distance from the camera. The surface normal is then given by

$$\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} \pm\cos(\tau)\sin(\sigma) \\ \pm\sin(\tau)\sin(\sigma) \\ \cos(\sigma) \end{pmatrix}. \tag{20}$$

From the surface normals, the gradients in the x- and y-directions are calculated for each patch:

$$\begin{aligned} f_x &= -n_x/n_z \\ f_y &= -n_y/n_z \end{aligned}$$

The Fundamental Theorem of Line Integrals states that

$$\| \int_C \bigtriangledown z.dr \| = 0 \tag{21}$$

for all closed curves C parameterized by r and a differentiable function z whose gradient $\bigtriangledown z$ is continuous on C. Using the calculated surface gradients as $\bigtriangledown z$ in Equation 21 and setting C to be any closed curve, we would expect that the left-hand-side of Equation 21 would be close to zero if we had correctly estimated the frontal texel, and much larger if we had not. Therefore, a measure of surface

inconsistency is obtained by summing the left-hand-side of Equation 21 around various loops. The loops used in the method are the triangles formed by choosing sets of three neighbouring texels and using their coordinates as the corners of the triangles. The cost term is then given by

$$\begin{aligned} cost = \sum_{i=1}^{N} & (f_{x,i,1}, f_{y,i,1}).(x_{i,2} - x_{i,1}, y_{i,2} - y_{i,1}) \\ + & (f_{x,i,2}, f_{y,i,2}).(x_{i,3} - x_{i,2}, y_{i,3} - y_{i,2}) \\ + & (f_{x,i,3}, f_{y,i,3}).(x_{i,1} - x_{i,3}, y_{i,1} - y_{i,3}) \end{aligned} \tag{22}$$

where N is the number of loops, $(x_{i,n}, y_{i,n})$ is the location of the nth texel in the ith loop, and $(f_{x,i,n}, f_{y,i,n})$ are the corresponding gradients in the $x-$ and $y-$ directions. The cost term may be interpreted as such: choose any three points on the surface, and as you move along the three sides of the triangle formed by these points, sum the change in surface height. Do this using the known gradients at each point and the distance between points. If the gradients were correctly estimated, the net change in surface height will be equal to zero. We use this cost term with the Levenberg-Marquardt method [62] to search for the values of c and b that result in the most consistent surface.

5.2.3 Estimating shape using the frontal texel

Once the frontal texel has been determined, it is used to estimate the surface shape. This is done by decomposing the transformations $\mathbf{T}_{f \to i} = \mathbf{T}_{j \to i} \begin{pmatrix} 1 & 0 \\ c_f & b_f \end{pmatrix}$ via equations 15, 16 and 17. Then

$$\tau = \arctan(u_{12}/u_{11}), \text{ and} \tag{23}$$

$$\sigma = \arccos(d_{11}/d_{22}) \tag{24}$$

specify the orientation at each texel.

5.2.4 Finding texels

The initial step of the shape-from-texture algorithm described in this chapter is to detect texels in the image. There are various ways to do this. Leung and Malik [49] detected texels by finding "interesting elements" within an image, and growing them to form a distinctive unit. The interest operator is based on the eigenvalues of the second moment matrix computed for a windowed segment of the image. Recently, many other interest point operators have emerged including the scale invariant feature transform (SIFT), which has been reported to perform well using various criteria [64]. In my experiments, interest points are found using a Maximally Stable Extremal Region (MSER) detector [63] as this was found to work well, with the additional advantage of having the code readily available [83].

5.3 Results

The new method for the perspective camera model was tested under various scenarios. The five main experiments are explained below.

1. Section 5.3.1 describes a test where a synthetic example was created with a smooth surface, and texels placed on the surface. The aim of this experiment was to test the novel part of the method — the minimisation of surface inconsistency, and as such it was necessary to avoid the texel-detection step and the automated estimation of affine transformations between texels.

2. Section 5.3.2 describes a synthetic example where texel detection was implemented, as was the automated estimation of the transformation between texels. The surface was more realistic as texels were placed at random locations on the surface with random initial rotation angles.

3. Section 5.3.3 describes the first test on a real image.

4. Section 5.3.4 describes more detailed testing of the method: the effect of the choice of reference texel, and whether a small number of texels is sufficient to reasonably estimate the surface shape.

5. Section 5.3.5 describes a version of the method that has been altered to be used in a structure-from-motion setting. A toy car was moved about on a smooth surface and the shape of the top view of the car were estimated by the algorithm, as well as the car's path.

5.3.1 First synthetic example

Firstly, the novel part of the method was tested (the minimisation of surface inconsistency) without either the texel-detection task or the estimation of affine transformation task playing a role. Both of these tasks have been previously tackled by others, but to my knowledge the minimisation using surface consistency has not. Therefore in the first synthetic example, the frontal texel (the image of the duck seen in Figure 52) was affine transformed and placed onto the surface in a regular grid to avoid having to search for texels. In order to place the texels, the frontal

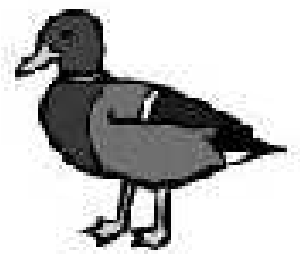


Figure 52: Texel used for first example

texel was randomly rotated about its normal axis, and placed on the surface such that its surface normal matched that of the surface patch under it. The texels were also scaled to account for the shrinking effect as texels move away from the camera. The untextured surface is shown in Figure 53 (from an oblique angle) and the textured surface (as viewed from on top) is shown in Figure 54.

The second texel in the top row of the textured surface was arbitrarily denoted the reference texel, j. The transformations from this texel to every other texel $\mathbf{T}_{j\to i}$ were estimated using a manual process: each of the transformed texels were presented to the user who clicked on three corresponding points (the tip of the

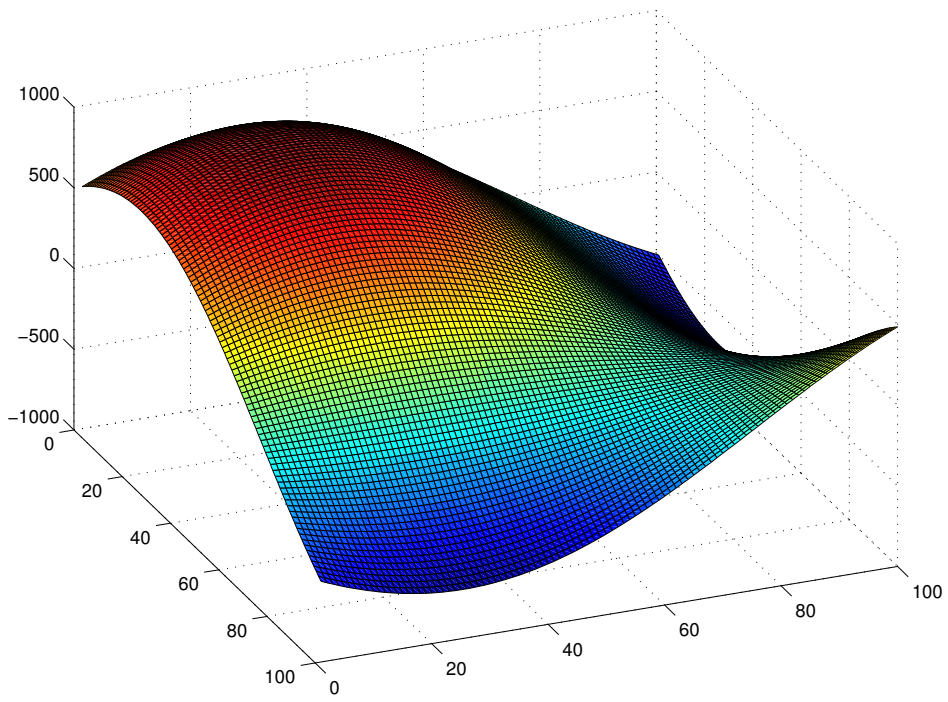


Figure 53: Synthetic surface given by $z = \sin(x + \frac{\pi}{4})\sin(y + \frac{\pi}{4})$ viewed from an oblique angle.

duck's beak, tail and front foot). Using these three points, the transformations between texels could be calculated. The transformations $\hat{\mathbf{T}}_{f\to i} = \mathbf{T}_{j\to i}\begin{pmatrix} 1 & 0 \\ c & b \end{pmatrix}$ were formed using some initial (c,b). The surface normals corresponding to each $\hat{\mathbf{T}}_{f\to i}$ were calculated along with the corresponding gradients in the x- and y-directions. Note that from each transformation $\hat{\mathbf{T}}_{f\to i}$, *two* surface normals (and hence two choices for gradients in the x- and y-directions) are obtainable, denoted by the positive and negative choices in Equation 20. This is the previously mentioned π-shift tilt ambiguity. For this example, a fast way of resolving the ambiguity for the special case of texels placed in a grid was used, however other methods for solving the ambiguity have already been mentioned in Section 3.2.2. The ambiguity was resolved up to gradient sign by regularizing gradient values one by one such that they were most similar to the gradients around them. It was found that the best way to do this was to begin with the texel with the highest slant and to arbitrarily choose either of the gradient options. Then its neighbour with the highest slant

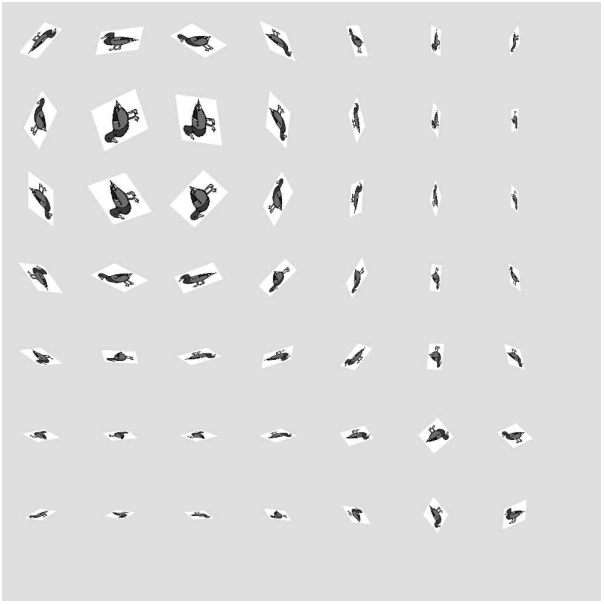


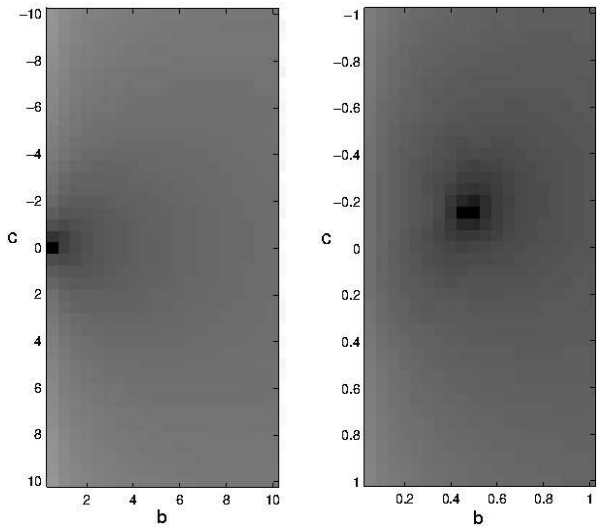
Figure 54: The textured surface used as input for the algorithm. The surface is viewed from directly above.

was regularized to match the gradient of the first patch. This pattern continued such that the gradients were resolved in the order of highest slant of any texel that neighboured an already resolved texel. In this way, the texels of low slant were resolved last.

Once all gradients had been decided on, the Levenberg-Marquardt method was used with a surface inconsistency measure to search for (c_f,b_f). Instead of using triangular loops for the cost function in Equation 22, it made more sense to use square loops since the texels were arranged in a grid. The transformations $\mathbf{T}_{f\to i} = \mathbf{T}_{j\to i}.\begin{pmatrix} 1 & 0 \\ c_f & b_f \end{pmatrix}$ were then decomposed to produce estimates for distance from camera (up to scale), slant and tilt via Equations 15 and 24.

The optimization was initialized from the point $(c,b) = (5,5)$ however the starting point might have been any reasonable value as shown by Figure 55. In (a) we see a plot of the log of the surface inconsistency measure as it varies across (c,b) space. The darkest point in the figure corresponds to a global minimum in surface inconsistency; in other words, the values of c and b at this point give rise to the most consistent surface. Only positive values of b are considered since negative values result in a flip of the texel (a transformation not allowed), and the case where $b = 0$

gives rise to a singular $\mathbf{T}_{f\to j}$. Figure 55 (b) shows a zoomed-in version of (a) where the global minimum is at the location (c_f,b_f) = (-0.16,0.52), which is close to the required value of (-0.19,0.52).



(a) (b)

Figure 55: Surface inconsistency as it varies with c,b. Brightness implies high surface inconsistency. The dark spots are the values of (c,b) that give rise to a consistent surface. (a) c,b with bounds [-10,10]. (b) zoomed in version with bounds [-1,1]. b =0 corresponds to a singular $\mathbf{T}_{f\to j}$.

The frontal texel was reconstructed from j using

$$\begin{aligned} f &= (\mathbf{T}_{f\to j})^{-1}\ j \\ &= \begin{pmatrix} 1 & 0 \\ c_f & b_f \end{pmatrix}^{-1} j \end{aligned} \tag{25}$$

and is shown in Figure 56 (b). Given that an initial rotation of the frontal texel makes no difference to our interpretation of surface shape, the texel in Figure 56 (b) is a good match of the true frontal texel. The texel defined as j is shown in (a). The surface shape was then estimated. The distance to camera was calculated up to scale using the values of r in Equation 19 for each texel, to give the estimation of surface shape shown in Figure 57. Qualitatively the shape estimation is good. Slant and tilt were calculated to produce the needle diagram shown in Figure 58, in which the length of each needle depends on the slant angle and the direction of

(a) (b)

Figure 56: (a) Texel defined as j. (b) Frontal texel as estimated by algorithm.

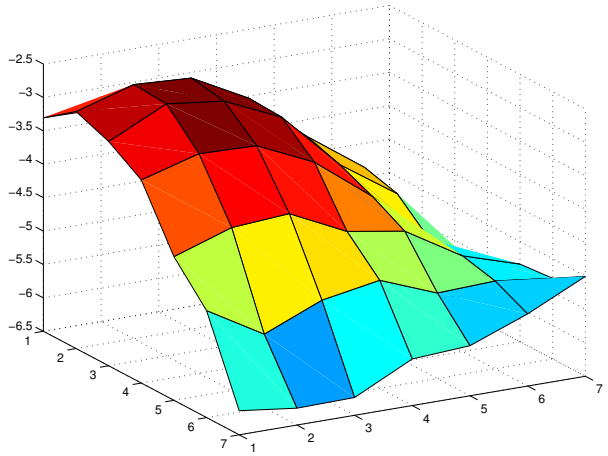


Figure 57: Estimation of surface shape using the scale of each texel. The surface is shown from an oblique view.

the needle is the tilt angle. The average tilt error was 1.47° and the average slant error was 1.51°. We would expect some error due to the manual input of the user to obtain feature point coordinates. The fact that the error still remains low implies a good level of robustness when estimating the frontal texel and the subsequent shape reconstruction.

The Levenberg-Marquardt optimization located the minima in a time of around one second using MATLAB version 6.5 [31], running under Linux on an Intel Pentium 4 2400 MHz CPU.

5.3.2 Second synthetic example

The algorithm was altered to include automated texel detection and estimation of affine transformations. It was tested on a synthetic image of a sphere covered in the texel shown in Figure 61(a). As with the first synthetic example, the texel was chosen to demonstrate that the algorithm works even if the texture is anisotropic.

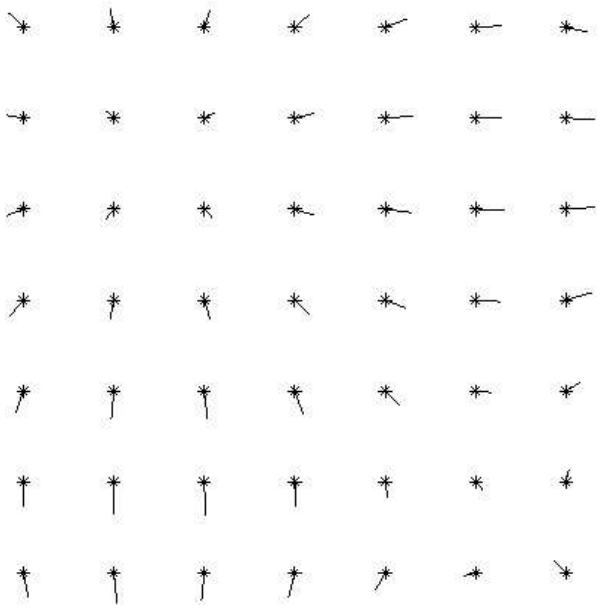


Figure 58: Calculated surface orientation shown from top view.

The texel also has many right angles which clearly expose any potential error in the estimate of the frontal texel.

As with the previous example, the texels were placed by randomly rotating the frontal texel about its normal axis such that its surface normal matched that of the surface under it. The texels were also scaled to account for the shrinking effect as texels move away from the camera. The difference with this example is that the texels are *not* placed in a regular grid but at random locations on the surface.

The textured surface shown in Figure 59 is a non-homogeneous, anisotropic and non-stationary texture.

Finding texels

Interesting regions were found using the Maximally Stable Extremal Regions (MSER) detector of Matas et al. [63]. A number of different techniques for distinguishing true texels from other detected regions were tried. Firstly SIFT features [58] were used for this task, where each texel was assigned a 128-vector. Ideally one would cluster the regions based on their SIFT feature vectors, however as a proof of concept, an experiment was conducted to test the classification ability of the SIFT features. The region with the median feature vector was assumed to be a true texel

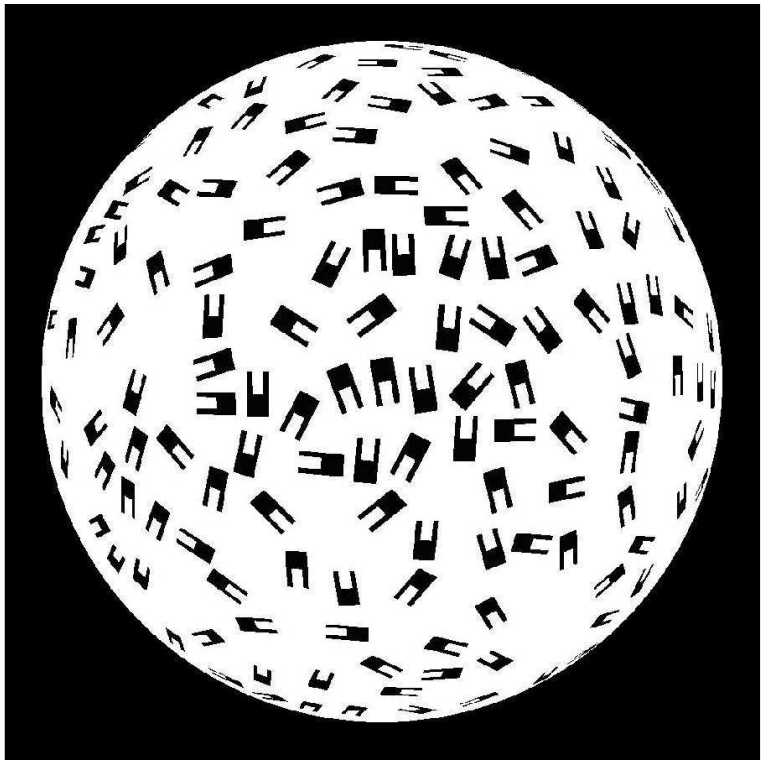


Figure 59: The textured surface

(the 'average' texel). This is reasonable since the detected regions comprise many more true texels than non-texels. Then, the vector distance from the average texel to every other region was calculated, and regions were classified as texels/non-texels based on whether that distance exceeded a threshold or not. This resulted in a fairly good classification of texel/non-texel regions, however a small number of non-texel regions always managed to be classed as texels.

In the end, a related but different method was used. A first pass removed regions that were large compared to the image size. In practice, regions were removed if their maximum radius was greater than ten percent of the shortest image axis because we could safely assume that true texels contributing shape information would never be this large. The average texel was again found using SIFT features, but this time the distance from this texel to any detected region was found using the following steps: transform the average texel to its isotropic version and do the same for the detected region. Rotate the isotropic version of the average texel in fine steps, calculating the correlation between it and the isotropic version of the detected region. The distance between the two is given by the maximum correlation. The

angle at which the maximum correlation is achieved is also stored for later use in the shape-from-texture algorithm. Again regions are sorted into texels and non-texels using a threshold on the distance.

Searching for the frontal texel

In order to search for the frontal texel, a reference texel j needed to be chosen. A good choice for j was an arbitrary texel that had been transformed to make it isotropic. This is a natural choice for j since the first step of calculating the transformations $\mathbf{T}_{j\to i}$ is to make j isotropic, and bypassing this step speeds up the algorithm. The texel j is shown in Figure 61(b).

To find the values of c_f and b_f an initial coarse search in (c, b) space was performed to find some near-optimal values. This was followed by a Levenberg-Marquardt optimization starting from these values to find the optimal solution. The (c, b) surface in Figure 60 shows that there is a definite global minimum in surface inconsistency, indicated by the localised dark spot.

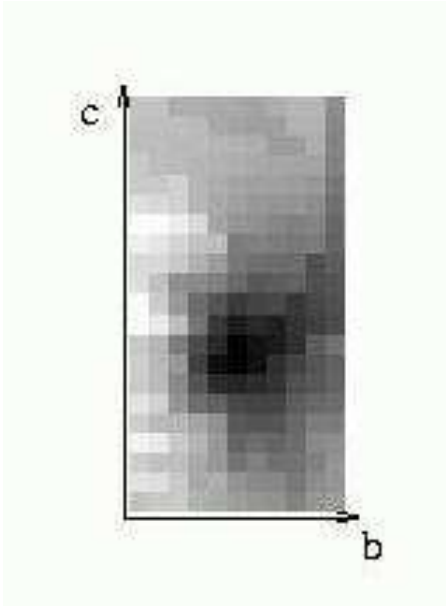
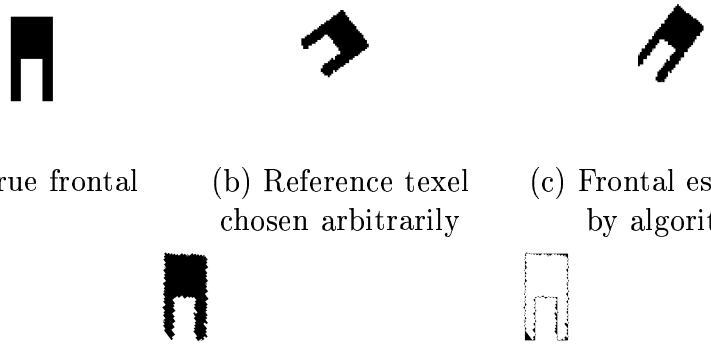


Figure 60: The (c,b) surface. The darker the pixel, the lower the surface inconsistency (and the higher the surface *consistency*). Therefore the darkest point corresponds to a global minimum where the values of c and b produce the true frontal texel.

The estimated frontal texel which resulted is shown in Figure 61(c). Note that its

rotation and scale compared to the true frontal in Figure 61(a) are of no consequence since these factors play no role in determining surface orientation. Therefore the estimated frontal texel can be scaled and rotated in order to compare it to the true frontal texel. This is depicted in Figure 61(e) where the pixels that differ between the two are in black. Most of these pixels that differ are due to the noisy border of the estimated frontal texel, and the fact that one of the reference texel's 'legs' was cut slightly short during the texel-finding step. However most of the pixels from the estimated and true frontal texels are in agreement, and therefore the estimation is accurate.



(a) True frontal

(b) Reference texel chosen arbitrarily

(c) Frontal estimated by algorithm

(d) the texel in (c) after rotation and scaling

(e) the pixels that differ between (d) and the true frontal texel

Figure 61: The true and estimated frontal texel. They are shown in (e) to be practically the same shape.

The surface orientation at each texel can then be calculated. This is demonstrated with the needle diagram shown in Figure 62. Note that the well-known tilt ambiguity can be resolved here since during the placement of texels, scaling occurred to account for the distance of each texel from the theoretical camera. The scale at each texel can be deduced from the transformation $\mathbf{T}_{f \to i}$. The tilt ambiguity can be resolved by choosing the candidate for tilt which is in the direction of decreasing scale.

The slant and tilt values were interpolated at values in a regular grid and the surface was reconstructed using the algorithm by Kovesi [43]. This surface is shown

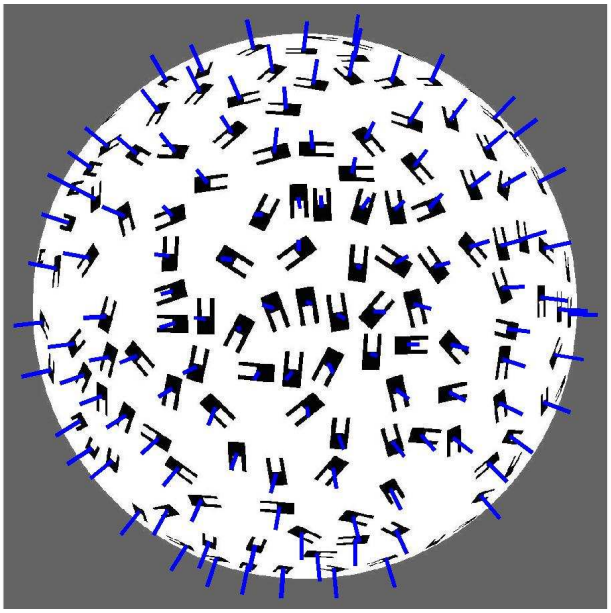


Figure 62: Needle diagram showing the calculated surface orientation at each texel.

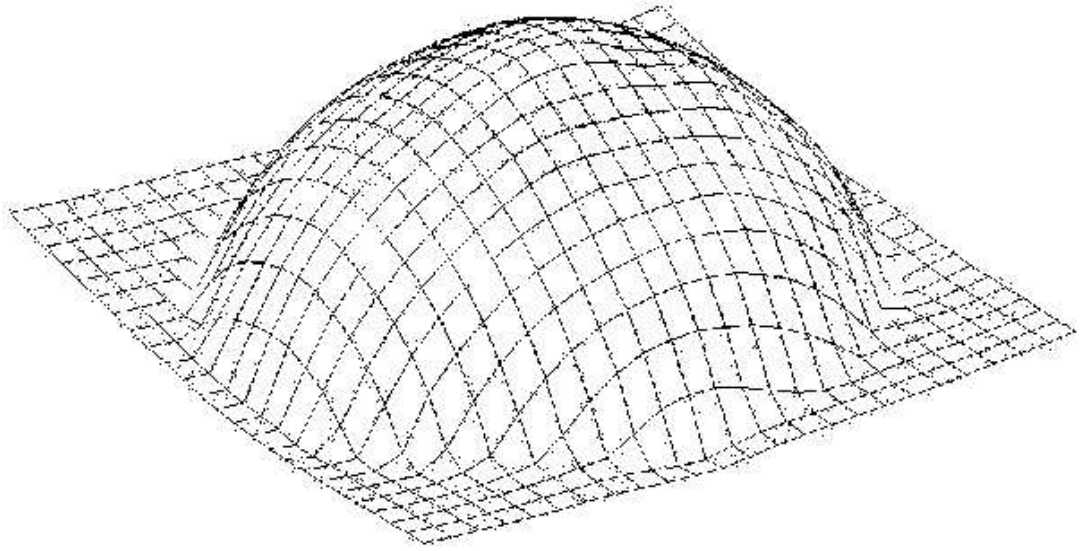


Figure 63: Calculated mesh surface.

in Figure 63.

5.3.3 First test on a real image

Next the algorithm was tested on a real image. Fabric was draped over the back of a chair to form the curved textured surface seen in Figure 64. This particular fabric was chosen because it demonstrates that the algorithm works on non-stationary

and anisotropic textures in a real image. The image has clear perspective effects, seen by the decreasing scale of the leaves as they move away from the camera. Again, potential texels are found using the MSER detector and true texels are

Figure 64: Real image: fabric with a non-stationary, anisotropic texture.

sorted from other regions. The reference texel j was chosen to be an arbitrary texel that has been transformed to make it isotropic, as shown in Figure 65(a). The transformations $\mathbf{T}_{j\to i}$ are found as before: each visible texel (such as the one shown in Figure 65(b)) is firstly made isotropic (Figure 65(c)), and this is rotated by the angle that makes it most similar to the reference texel (Figure 65(d)). Therefore the sequence (b)–(d) shows the transformation $\mathbf{T}_{i\to j}$; we take the inverse transformation to give $\mathbf{T}_{j\to i}$. As before, an initial coarse search in (c, b) space yields near optimal

(a) Reference texel (b) Some visible texel (c) Isotropic version of (b) (d) Rotated version of (c) to match (a)

Figure 65: Calculating the transformations $\mathbf{T}_{i\to j}$. The sequence (b)–(d) shows how some visible texel is transformed into the reference texel shown in (a).

values for surface consistency, which are then refined with a Levenberg-Marquardt method. Figure 66(a) displays the frontal texel as estimated by the algorithm. It is a transformed version of the reference texel shown in Figure 65(a). A real image of the fabric when viewed frontally is shown in Figure 66(b). For comparison, and considering that initial rotations play no role in shape information, the image in (b) has been rotated so that it matches the orientation in (a).

(a) Frontal estimated by algorithm

(b) Real image of fabric viewed frontally

Figure 66: Estimated and true frontal texels.

The estimated frontal texel is used to calculate the orientation at each texel. This is shown in a needle diagram in Figure 67. The surface is constructed again with Kovesi's method [43] to give the surface seen in Figure 68(a). The curved corner of the chair is clearly seen. In Figure 68(b) the original image of the texture has been orthogonally projected onto the surface. Although it does not technically make sense to do this, since there are clear perspective effects in the image and it is being projected orthographically, the purpose of Figure 68(b) is to allow the surface in (a) to be visualised. For comparison, Figure 69 shows a real image of the chair from a similar angle.

The aim of this test was to qualitatively show that the algorithm could be used on real photographs. As indicated by the estimated frontal texel and the resulting needle diagram, a qualitatively good estimation of the curved surface can be achieved. The next section describes some more detailed testing of the algorithm.

5.3.4 Another real image

In this experiment, the aim was to perform more detailed tests on the method. In particular, the effect of the choice of reference texel was investigated, and it was

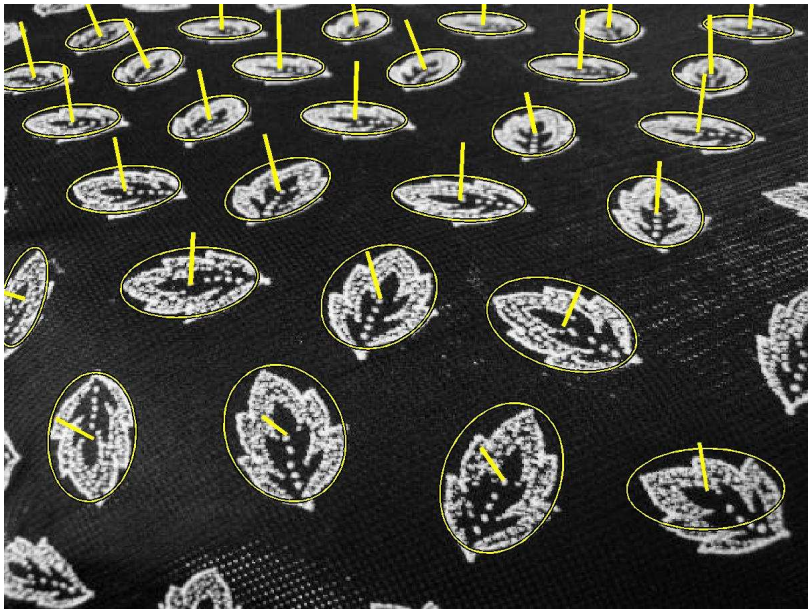
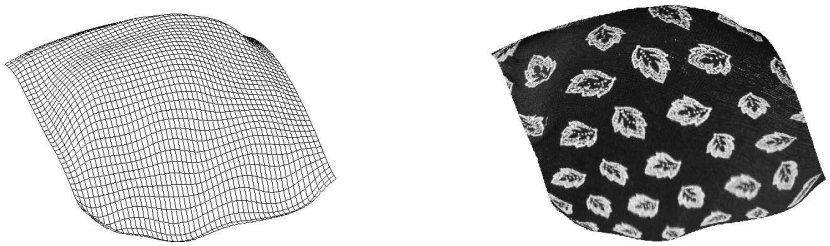


Figure 67: Needle diagram also showing detected texels



(a) Estimated surface shape (b) Texture projected onto surface

Figure 68: Reconstruction of chair

observed whether only a few texels could indicate the general shape of the curved surface.

Figure 69: Real image of the chair.

The set up of the experiment is shown in Figure 70. The camera was placed looking diagonally downward onto a sloped surface. Paper leaves were used as texels

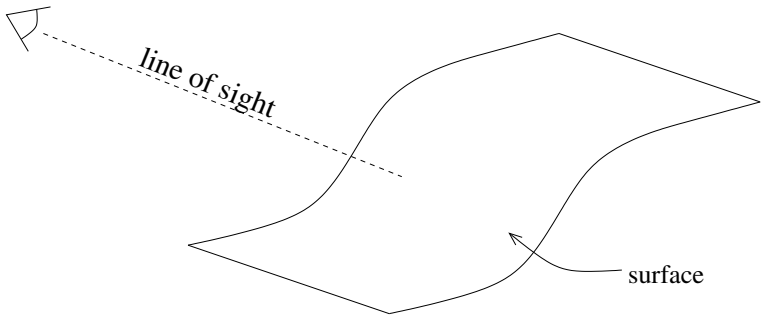


Figure 70: Set up for experiment.

so that so that all texels were identical. 24 leaves were placed on the surface with random rotations as shown in Figure 71. This image was used as the input to the algorithm.

Figure 71: Set up for experiment. The leaves were numbered 1–24 with each leaf's number appearing below it in the figure.

Since the aim here was to test the optimisation part of the method (the novel part), the estimation of affine transformations between leaves was achieved by clicking on three feature points of each leaf and calculating the transformations between these sets of points.

The shape-from-texture method was run 24 times, each time using a different

leaf as the reference texel. The estimated frontal texels that result are displayed in Figure 73. Below each estimated frontal texel is the number of the leaf that was used as the reference texel. For comparison, an image of the real frontal texel is shown in Figure 72. Most of the leaves gave a reasonable result, considering that rotations and scaling are not important because they do not alter the surface normal and hence the consistency measure. Most notably, good results are achieved when the leaves numbered 3, 4, 6, 7, 11, 1 ,13, 15, 18, and 19 are used as reference texels. The worst performing reference texels were 2, 5, 16, and 20. However, an inspection of Figure 71 shows that leaves 2, 5, and 16 are the only leaves that were substantially curled which does not match the assumption that all texels are 'painted' onto the surface, and that leaf 20 is heavily foreshortened, to the point where the three feature points were difficult to click on. This experiment suggests that when choosing a reference texel, most choices will give a reasonable result, however if a bad choice is made e.g. choosing a leaf that does not fit the assumption of having no height, the estimate of the frontal texel suffers. This suggests that some sort of 'average' frontal texel should be used to construct the surface shape, to help reduce the effect of an outlier. Future work in this area may be to devise a method of calculating the average frontal texel.

The surface was then reconstructed. In the results that follow, leaf number 3 has been used as the reference texel. The aim here is to show that even with as few as 24 texels a reasonable surface reconstruction can be achieved. The (c,b) surface shown in Figure 74 shows a definite global minimum in surface inconsistency, corresponding to the true frontal texel. Using this frontal texel, the resulting surface is shown in Figure 75, viewed from almost side-on so that it can be compared to Figure 70. The camera has been drawn onto the diagram to show that it looks directly down in this reconstruction. The reconstruction is satisfactory, considering the few texels present on the surface to indicate its shape. The surface does cup a little in its centre, however the overall shape is not too bad; we strongly see the surface slant away from the camera at point A, before becoming more orthogonal to the line of sight at B, and then slanting away again at point C. Future work would be to see how the number of texels, and their relative size compared to the 'wiggles'

on the surface, affect the reconstruction. An interesting experiment would be to reduce the number of texels to the point where the surface cannot be satisfactorily reconstructed. In practise however, one assumes that the method could only be reliably used where there are plenty of texels to give shape information. For all other cases, this method can still be used to supply additional information to other reconstruction methods — say, to sort out ambiguities in a shape-from-shading problem — even if a very accurate surface shape cannot be achieved from texture alone.

Figure 72: True frontal texel.

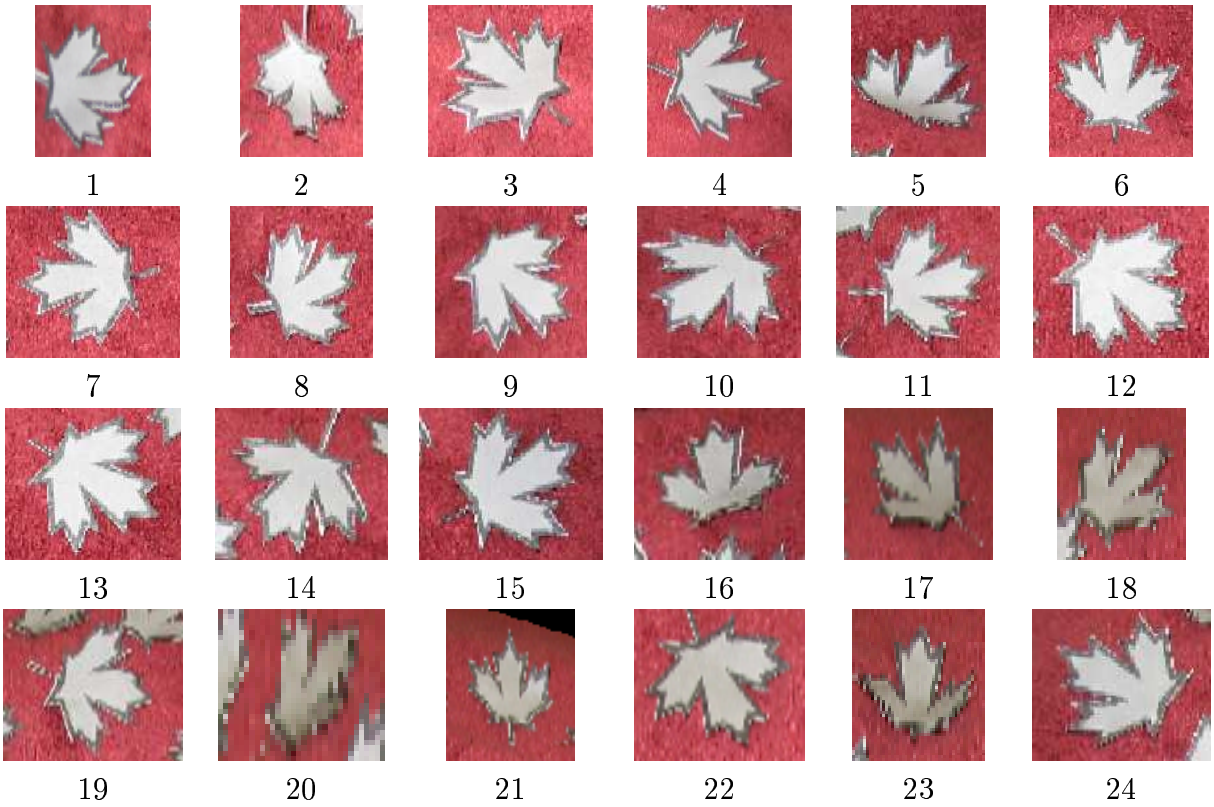


Figure 73: Estimated frontal texels when various leaves are used as the reference texel. Leaf numbers appear below the corresponding leaves.

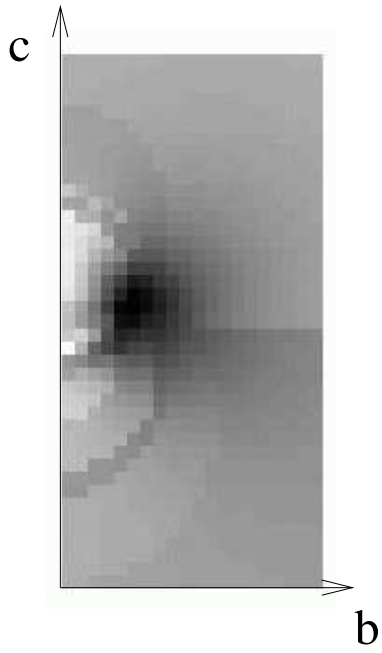


Figure 74: The (c,b) surface.

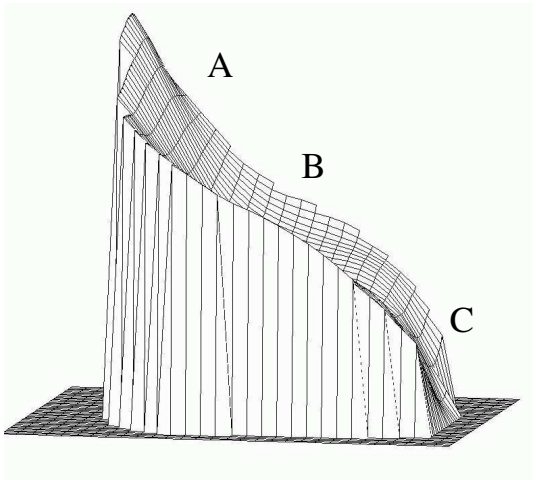


Figure 75: The reconstructed surface.

5.3.5 A structure-from-motion experiment

It has been previously noted that shape-from-texture shares many similarities with structure-from-motion; if the camera is stationary and the moving object is planar, or nearly so, then the superposition of the images of the moving object produce a texture, the structure of which can be solved.

In this experiment, the aim was to test whether a shape-from-texture algorithm could be altered for a structure-from-motion setting. The experiment served as a proof of concept, rather than a detailed investigation and so quantitative measures were not been applied.

A toy car was moved about on the surface of a chair with the camera pointing down on an angle as seen in Figure 76. This is effectively the same as a camera aimed horizontally at a sloped surface. The aim of the experiment was to estimate the shape of the top view of the car, as well as its motion in 3-D space.

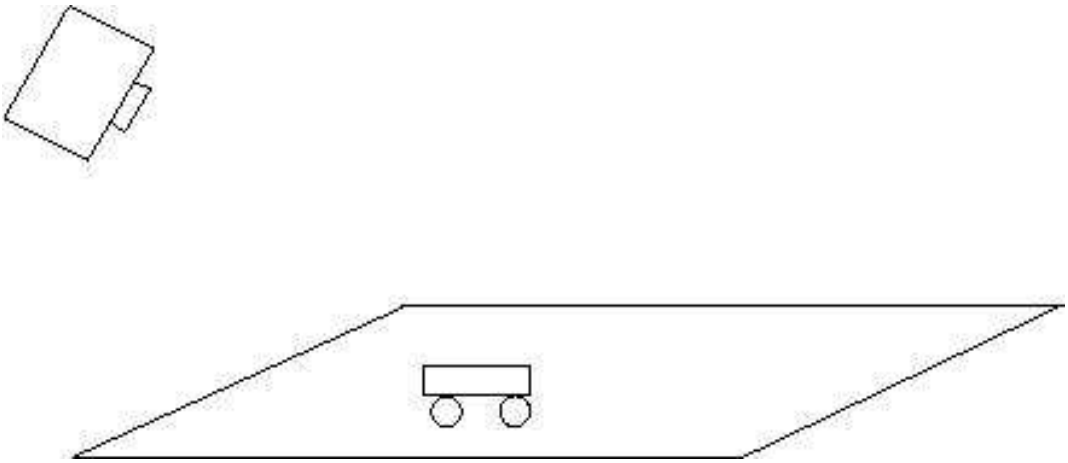


Figure 76: Set up for structure-from-motion experiment.

The sequence of the car moving about on the surface is shown in Figure 77.

The shape-from-texture algorithm was adapted to the structure-from-motion task by firstly assuming that the car can be approximated by the horizontal plane that goes through the car's hood and back lights. This plane has approximately the same orientation as the surface beneath the car. In order to find the affine transformations between this plane in the various frames of the sequence, each frame was presented to the user in turn, and three points on the plane were clicked on; these are shown in Figure 78.

Figure 77: Sequence of car moving about on surface

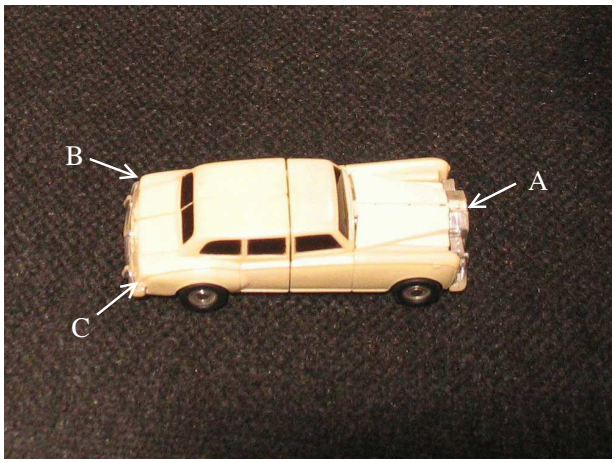


Figure 78: The car used in the experiment. The feature points labeled A,B, and C correspond to the front middle of the car's hood, and the two back lights respectively.

As with the shape-from-texture algorithm, a reference texel was chosen. In this case, the 'texel' was the car in one of the frames. The method was then used to find the transformation from the reference car to the 'frontal' car i.e. the top view as shown in Figure 79(a).

Figures 79(b–d) show the car as it appears in various frames. As mentioned, the 'texels' here are the horizontal planes that run through the hood of the car, which are depicted as the trapeziums superimposed in (b–d). When each of these cars is used as the reference car, the method estimates that the 'frontal' car is as shown in Figures 79(e–g). Note that regardless of the appearance of the cars in (e–g) we require only that the trapeziums from (b–d) transform into rectangles that have very similar dimensions to the rectangle in Figure 79(a). The rectangles in

Figures 79(e–g) have not been redrawn onto the image by hand; the images in (b–d) were simply transformed by the method to show the estimate of the frontal car. The rectangles that result in (e–g) have close to the correct dimensions as shown in Figure 80 where the rectangle representing the true frontal dimensions has been superimposed with scaled and rotated versions of the quadrilaterals representing the estimated frontal dimensions. They also have close to right angles; the average angle at the 12 corners is 90° with a standard deviation of 3.2°. Satisfactory results were obtained no matter which frame was chosen to provide the reference texel.

In order to show the estimated surface normals, all frames were binary thresholded and added to give the image in Figure 81. The superimposed arrows indicate the direction of the estimated surface normals. Since the cars are driving about on effectively an inclined surface we expect the surface normals to be pointing directly upward. This is the case for most arrows although two of the arrows have a small horizontal component which may be due to user error when clicking on the feature points, or the fact that the surface does indeed curve upward at the edges (the surface is an office chair). The estimated surface is shown in Figure 82 with the car's trajectory superimposed. At some points in the figure, the car's trajectory appears to extend beyond the surface beneath it; this is an artifact of the implementation rather than the estimated surface. The edges of the surface are slightly degraded during the process of interpolating surface normal information all over the surface, using the information from a few points on the surface. The main observation is that the surface shape slopes upward as required.

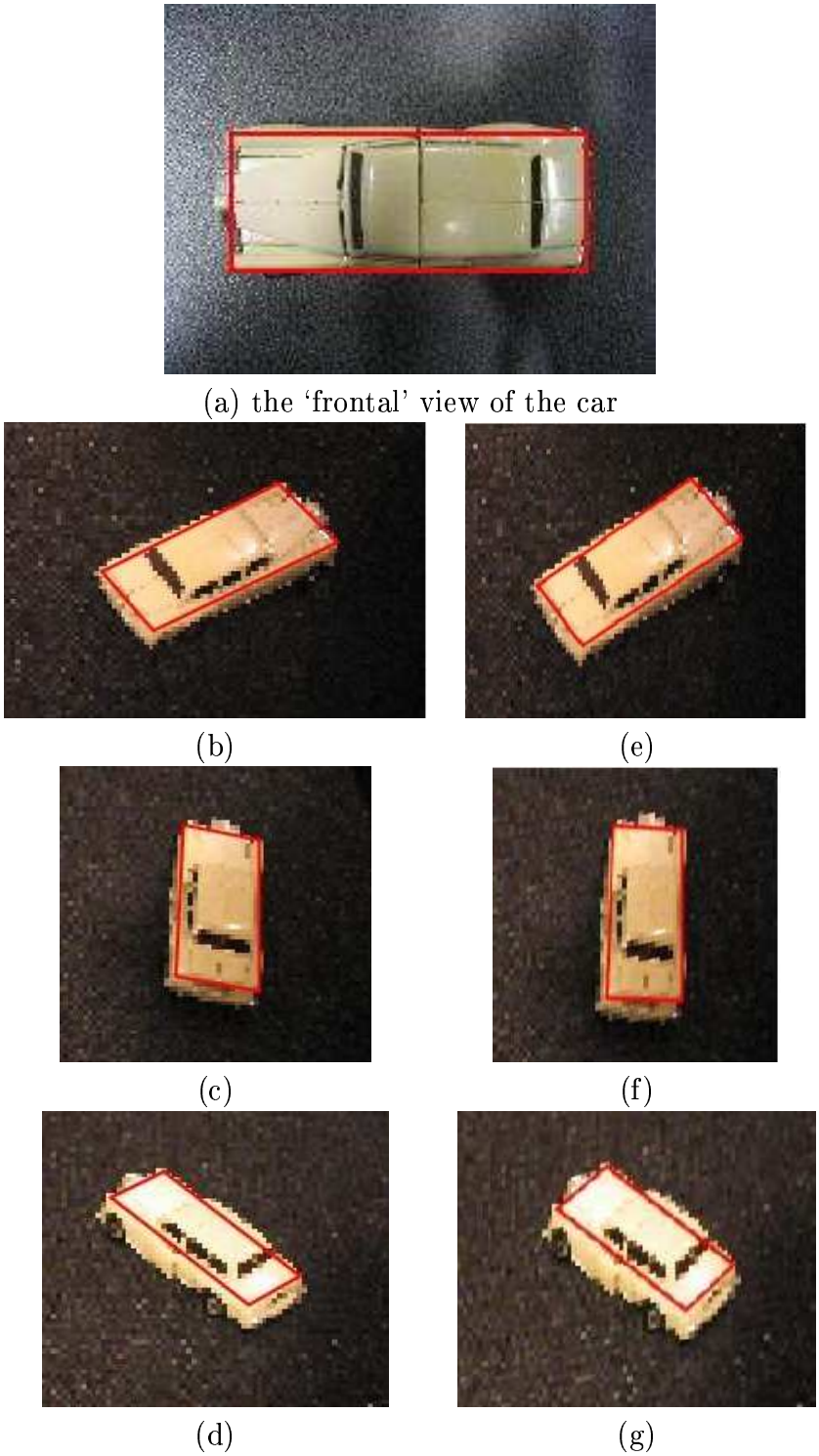
In this experiment, quantitative measures have not been applied. The aim has been to see whether the shape-from-texture algorithm could be applied in a structure-from-motion setting. This test shows that it can. Future work in this area would incorporate more robust testing with quantitative error measurements.

It is an interesting question as to whether many objects can be used for this kind of experiment. The toy car was fairly ideal in that a visible horizontal plane through the car could be used to approximate the car itself. Also this plane had an orientation similar to the surface beneath the car, since cars tend only to yaw, but not pitch or roll with respect to the surface.

5.3.6 Discussion and conclusions

So far in this work, textures have been described in terms of distinct texels, but in principle our algorithm will work for textures where one cannot distinguish individual texels; interest point locators such as Lowe's [58] might be used to detect corresponding points in the repeated pattern, and a second moment matrix assigned to each point. Transformations between textures at different points would be determined using the algorithm outlined in Chapter 4.

A method was demonstrated for estimating the frontal texture on a surface and using it to reconstruct the surface shape. The main contribution of this work is a means of solving the shape-from-texture problem using a perspective camera model without the assumptions that the texture is stationary, homogeneous or isotropic, or that the frontal texture can be located in the image, is known a priori or from a given set.



(a) the 'frontal' view of the car

(b) (e)

(c) (f)

(d) (g)

Figure 79: Cars as they appear in various frames are shown in (b–d). When each of these is used as the reference car, they are transformed into the 'frontal' car as shown in (e–g). Since the car is approximated with a horizontal plane we need only to check that the superimposed trapeziums in (e–g) have the same angles and dimensions as that in (a). The rotations and size of the rectangles do not matter.

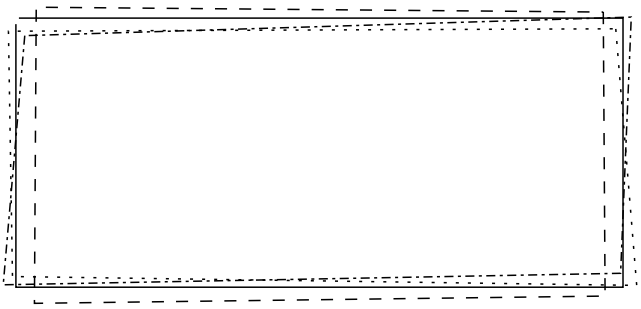


Figure 80: Rectangles that represent the dimensions of the frontal texel. The true dimensions are depicted with a solid line. The three estimates are superimposed with non-solid lines.

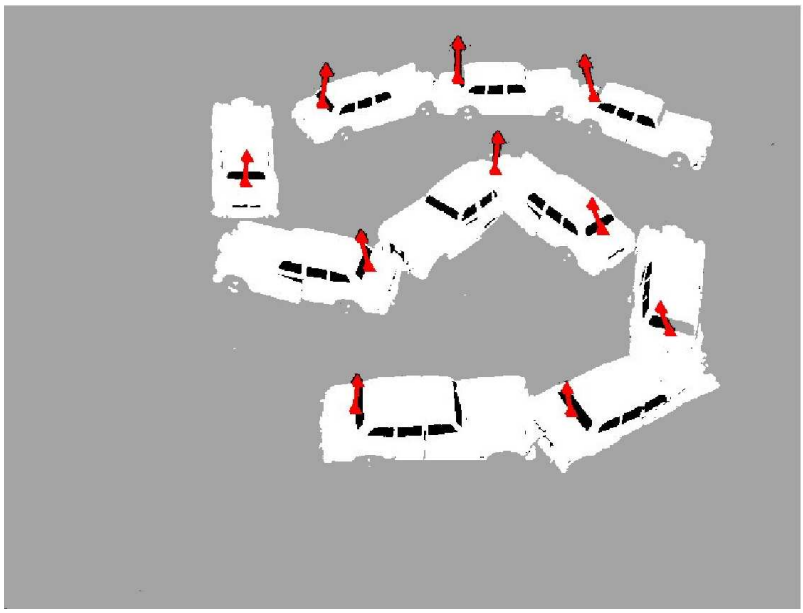


Figure 81: Estimated surface normals

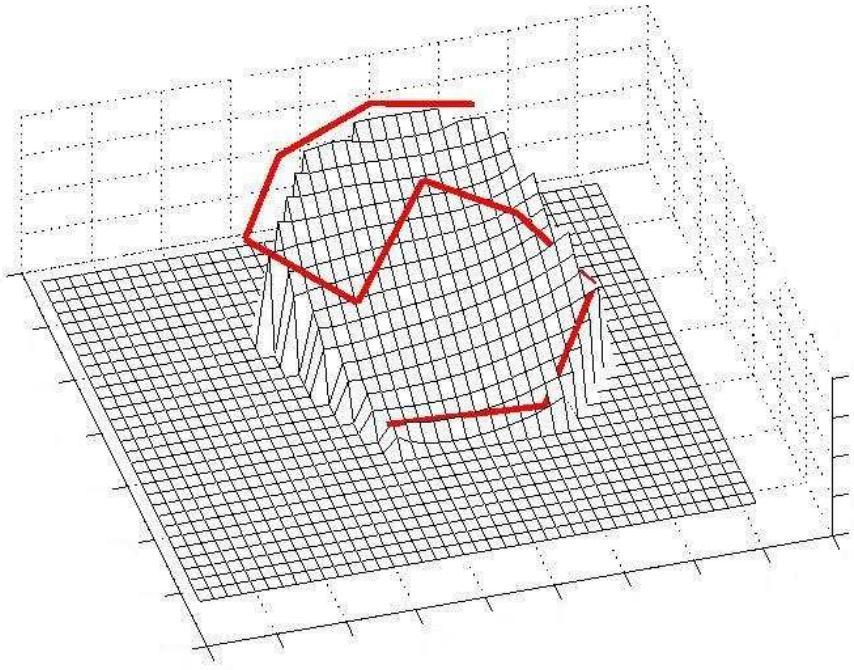


Figure 82: Estimated surface and trajectory of car

Chapter 6

Conclusions

This thesis has presented work in shape-from-texture, which is the problem of deducing the 3-D shape of an object using the patterns of markings on its surface.

6.1 Contributions

This work has produced the following original work.

- A novel shape-from-texture algorithm for use with stationary, homogeneous textures viewed orthographically. It was shown that the surface normal for a patch of texture can be solved without explicitly calculating the transformation between any patches of texture.
- An algorithm that solves shape-from-texture in one of its most general forms; it is designed to work regardless of whether or not the texture is homogeneous, stationary or isotropic, and whether or not a perspective view is used. A smooth surface is assumed.
- A method for estimating the affine transformation between patches of texture. This method was shown to be useful as part of a shape-from-texture algorithm, as well as many other applications.

6.2 Summary of work

The first algorithm (presented in Chapter 3) was designed for orthographic views of a texture. Although the previous methods by Gårding, Malik and Rosenholtz and Clerc and Mallat also assume that the texture is homogeneous and stationary, the new method does not require the estimation of the transformation between texture patches, as these other methods do. This more direct approach is possible only if the viewing geometry is orthographic; the other methods mentioned above also account for perspective views. The new method is based on the fact that as an image is slanted, the second moments do not change about the tilt axis. This simple property leads to an estimation of tilt, which in turn leads to an estimation of the slant angle.

The effects of blur, illumination variations and windowing schemes were investigated. Then the algorithm was tested on synthetic textures to verify that it behaved as expected. In tests on real images, the algorithm was found to respond to segmented texture borders in a way that can be useful: the surface normal is calculated by the algorithm to have a slant angle of $\frac{\pi}{2}$ and tilt angle orthogonal to the texture border. The algorithm was compared to Super and Bovik's algorithm which is the most similar previous algorithm. Tests showed that, as well as producing an additional ambiguity in the surface normal, their algorithm was less accurate at estimating slant in areas where the texture did not change abruptly. Super and Bovik's algorithm was also prone to producing complex, and hence unusable, values for slant near areas of frontal texture. Thus, while using similar assumptions as Super and Bovik's algorithm, the new method presented in this thesis has further advantages.

The second algorithm presented in Chapter 5 removed the assumptions that the texture was isotropic, homogeneous and/or stationary. Perspective views were also permitted. It is not assumed that the frontal texture is known, since the algorithm firstly estimates the frontal texture and uses this to deduce surface shape. The surface is assumed to be smooth, which allows estimation of the frontal texture via a consistency constraint. No other algorithm solves shape-from-texture under these

assumptions. Good results were achieved for real images. Experiments looked at the effect of choosing various reference texels. It was found that a reference texel chosen from the image generally led to a reasonable estimate for the frontal texel provided that the reference texel conformed to the assumptions of being flat ('painted onto the surface') and not too heavily foreshortened. For texels that did not fit these assumptions, a good estimate of the frontal texel was not obtained. This implies that some sort of 'average' frontal texel may be used to decrease the effect of a 'bad' reference texel. Another experiment showed that the algorithm may be adapted for a structure-from-motion setting; using several snap shots of a car moving about on a surface, the algorithm was able to estimate the dimensions of the car, the surface shape (a simple inclined plane was used) and the car's trajectory in space.

This thesis also presented a method for estimating the affine transformation between patches of textures (presented in Chapter 4). This method had been developed to be used as part of the shape-from-texture algorithm in Chapter 5, however other uses for the method were explored, namely re-texturing an object and placing an object in a scene.

6.3 Directions for future work

For the shape-from-texture method for orthographic views, future work could be undertaken in the following areas:

- removing the need for user interaction in identifying the frontal texel. A method for automatically estimating the frontal texture was developed by Forsyth [16] for textures composed of distinct texels. If a similar method could be developed for all textures (irrespective of whether or not distinct texels can be located) this would replace the user interaction part of the method.

- finding a method that automatically adjusts the size of the 2D Gaussian used to window patches of texture. Potential ways of doing this were discussed in Section 3.1.5. These suggestions are based on trying to find the characteristic

scale of the texture so that an appropriately sized window can be used to extract patches.

- determining what to do when the patch contains some of the texture border and a slant angle of $\frac{\pi}{2}$ is *not* required. One suggestion might be to remove all such patches from the shape-from-texture process; shape information could be inferred from the remaining patches. This should be a reasonable solution since these patches containing some of the texture border do not generally make up a large proportion of the set of all extracted patches; they only reside around the edge of an object.

For the algorithm that estimates the affine transformation between texture patches, future work could be undertaken in the following areas:

- finding a method to calculate the scale factor between textures. This does not seem too difficult; ways have been discussed in Section 4.3.2.

- finding other ways that the method may be useful. For example the method may be useful in rectifying text for optical character recognition, or deducing shape changes in non-rigid objects by observing how the patches of texture change.

For the shape-from-texture method for perspective views, future work could be undertaken in the following areas:

- varying the texel used as the reference texel, and somehow incorporating all of the estimates of the frontal texel into an 'average' frontal texel. This should decrease the effect of a bad choice of reference texel.

- more experimentation to find the limits of the algorithm in terms of the size and number of texels required to reasonably estimate the shape of a surface.

- demonstrating that the algorithm can work on images where the distinct texels are not located, but instead patches of texture are used. This would allow the algorithm to work on textures that do not exhibit distinct texels.

Bibliography

[1] Y. Aloimonos and M.J. Swain. Shape from texture. *Biological Cybernetics*, 58(5):345–360, 1988.

[2] R.K. Bajcsy and L.I. Lieberman. Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5(1):52–67, 1976.

[3] D. Blostein and N. Ahuja. Shape from texture: integrating texture-element extraction and surface estimation. *Pattern Analysis and Machine Intelligence*, 11(12):1233–1251, December 1989.

[4] B. Boufama, R. Mohr, and F. Veillon. Euclidian constraints for uncalibrated reconstruction. In *International Conference on Computer Vision*, pages 466–470, 1993.

[5] P. Brodatz. *Textures: A photographic album for artists and designers.* Dover Publications, New York, 1966.

[6] M.J. Brooks and W. Chojnacki. Direct computation of shape from shading. In *International Conference on Pattern Recognition*, pages A:114–119, 1994.

[7] L.G. Brown and H. Shvaytser. Surface orientation from projective foreshortening of isotropic texture autocorrelation. *Pattern Analysis and Machine Intelligence*, 12(6):584–588, June 1990.

[8] D. Buckley, J.P. Frisby, and E. Spivey. Stereo and texture cue combination in ground planes: an investigation using the table stereometer. *Perception*, 20:91, 1991.

[9] M.J. Chantler. Why illuminant direction is fundamental to texture analysis. In *Vision, Image and Signal Processing, IEE Proceedings*, pages 199–206, 1995.

[10] M. Clerc and S. Mallat. The texture gradient equation for recovering shape from texture. *Pattern Analysis and Machine Intelligence*, 24(4):536–549, April 2002.

[11] J.E. Cryer and M. Tsai, P.-S.and Shah. Integration of shape from x modules: combining stereo and shading. In *Computer Vision and Pattern Recognition*, pages 720–721, 1993.

[12] J.E. Cutting and R.T. Millard. Three gradients and the perception of flat and curved surfaces. *Journal of Experimental Psychology: General*, 113(2):198–216, 1984.

[13] M. Daum and G. Dudek. On 3-D surface reconstruction using shape from shadows. In *Computer Vision and Pattern Recognition*, pages 461–468, 1998.

[14] D. Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of The Optical Society of America*, 4(12):2379–2394, December 1987.

[15] D.A. Forsyth. Shape from texture and integrability. In *International Conference on Computer Vision*, pages II: 447–452, 2001.

[16] D.A. Forsyth. Shape from texture without boundaries. In *European Conference on Computer Vision*, pages III: 225–239, 2002.

[17] R.T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *Pattern Analysis and Machine Intelligence*, 10(4):439–451, July 1988.

[18] J.J. Gibson. *The perception of the visual world.* Houghton Mifflin, 1950.

[19] J. Gårding. Shape from texture and contour by weak isotropy. In *International Conference on Pattern Recognition*, pages I 324–330, 1990.

[20] J. Gårding. Shape from texture for smooth curved surfaces. In *European Conference on Computer Vision*, pages 630–638, 1992.

[21] C.G. Harris and M.J. Stephens. A combined corner and edge detector. In *Proceedings Fourth Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[22] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, pages xi–xii. Cambridge University Press, 2000.

[23] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, pages 551–553. Cambridge University Press, 2000.

[24] M. Hatzitheodorou and J.R. Kender. An optimal algorithm for the derivation of shape from shadows. In *Computer Vision and Pattern Recognition*, pages 486–491, 1988.

[25] B.K.P Horn. *Robot Vision.* McGraw-Hill, 1986. New York.

[26] B.K.P. Horn. Height and gradient from shading. In *MIT AI Memo*, pages 37–75, 1989.

[27] B.K.P. Horn and M.J. Brooks. Shape and source from shading. In *International Joint Conferences on Artificial Intelligence*, pages 932–936, 1985.

[28] B.K.P. Horn and M.J. Brooks. *Shape from Shading.* M.I.T. Press, 1989.

[29] J.W. Hsieh, H.Y.M. Liao, M.T. Ko, and K.C. Fan. Wavelet-based shape from shading. *Graphical Models and Image Processing*, 57(4):343–362, July 1995.

[30] K. Ikeuchi. Shape from regular patterns. *Artificial Intelligence*, 22(1):49–75, 1984.

[31] The MathWorks Inc. MATLAB version 6.

[32] M. Irani, P. Anandan, and M. Cohen. Direct recovery of planar-parallax from multiple frames. *Pattern Analysis and Machine Intelligence*, 24(11):1528–1534, November 2002.

[33] K.I. Kanatani. Detection of surface orientation and motion from texture by a stereological technique. *Artificial Intelligence*, 23(2):213–237, July 1984.

[34] K.I. Kanatani. Structure and motion from optical flow under orthographic projection. *Computer Vision, Graphics and Image Processing*, 35(2):181–199, August 1986.

[35] K.I. Kanatani and T.C. Chou. Shape from texture: general principle. *Artificial Intelligence*, 38(1):1–48, February 1989.

[36] L.M. Kaplan and C.C.J. Kuo. Texture segmentation via Haar fractal feature estimation. *Journal of Visual Communication and Image Representation*, 6(4):387–400, December 1995.

[37] J.R. Kender. Shape from texture: an aggregation transform that maps a class of textures into surface orientation. In *International Joint Conference on Artificial Intelligence*, pages 475–480, 1979.

[38] J.R. Kender and E.M. Smith. Shape from darkness. In *International Conference on Computer Vision*, pages 539–546, 1987.

[39] R. Klette and K. Schluns. Height data from gradient fields. *Proceedings of SPIE Machine Vision Applications, Architecture, and Systems Integration*, pages 204–215, 1996.

[40] J.J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.

[41] J.J. Koenderink and S.C. Pont. Irradiation direction from texture. *Journal of the Optical Society of America*, 20(10):1875–1882, October 2003.

[42] P. Kovesi. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/~pk/research/matlabfns/>.

[43] P. Kovesi. Surface normals to surfaces via shapelets. In *Proceedings Australia-Japan Advanced Workshop on Computer Vision*, pages 101–108, 2003.

[44] P. Kovesi. Shapelets correlated with surface normals produce surfaces. In *International Conference on Computer Vision*, pages 994–1001, 2005.

[45] R. Kozera. An overview of the shape-from-shading problem. *Machine Graphics and Vision*, 7:291–312, 1989.

[46] J. Krumm and S.A. Shafer. A characterizable shape-from-texture algorithm using the spectrogram. In *Proceedings of the IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, pages 322–325, 1994.

[47] K.M. Lee and C.C.J. Kuo. Direct shape from texture using a parametric surface model and an adaptive filtering technique. In *Computer Vision and Pattern Recognition*, pages 402–407, 1998.

[48] T.S. Lee. Image representation using 2D Gabor wavelets. *Pattern Analysis and Machine Intelligence*, 18(10):959–971, October 1996.

[49] T. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In *European Conference on Computer Vision*, pages I: 546–555, 1996.

[50] T. Leung and J. Malik. On perpendicular texture or: Why do we see more flowers in the distance? In *Computer Vision and Pattern Recognition*, pages 807–813, 1997.

[51] T. Lindeberg. On scale selection for differential operators. In *Scandinavian Conference on Image Analysis*, pages 857–866, 1993.

[52] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, November 1998.

[53] A. Lobay and D.A. Forsyth. Recovering shape and irradiance maps from rich dense texton fields. In *Computer Vision and Pattern Recognition*, pages I: 400–406, 2004.

[54] A.M. Loh. *Technical Report UWA-CSSE-05-001:Shape from texture without estimating transformations.* School of Computer Science and Software Engineering, The University of Western Australia, 2005.

[55] A.M. Loh and R. Hartley. Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. In *British Machine Vision Conference*, pages I: 69–78, 2005.

[56] A.M. Loh and P. Kovesi. Estimation of surface normal of a curved surface using texture. In *Proceedings of the Digital Image Computing: Techniques and Applications Conference*, pages 155–164, 2003.

[57] A.M. Loh and A. Zisserman. Estimating the affine transformation between textures. In *Digital Image Computing: Techniques and Applications*, pages 462–467, 2005.

[58] D. Lowe. Distinctive image features from scale invariant keypoints. In *International Journal of Computer Vision*, pages 91–110, 2004.

[59] D.G. Lowe and T.O. Binford. The recovery of three-dimensional structure from image curves. *Pattern Analysis and Machine Intelligence*, 7(3):320–326, May 1985.

[60] J. Malik and R. Rosenholtz. A differential method for computing local shape-from-texture for planar and curved surfaces. In *Computer Vision and Pattern Recognition*, pages 267–273, 1993.

[61] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 23(2):149–168, June 1997.

[62] D.W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Japan Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963.

[63] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[64] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *Computer Vision and Pattern Recognition*, pages II: 257–263, 2003.

[65] R. Mohr, L. Quan, and F. Veillon. Relative 3d reconstruction using multiple uncalibrated images. In *Computer Vision and Pattern Recognition*, pages 543–548, 1993.

[66] L. Noakes and R. Kozera. Nonlinearities and noise reduction in 3-source photometric stereo. *Journal of Mathematical Imaging and Vision*, 18(2):119–127, March 2003.

[67] Y. Ohta, K. Maenobu, and T. Sakai. Obtaining surface orientation from texels under perspective projection. In *International Joint Conferences on Artificial Intelligence*, pages 746–751, 1981.

[68] O. Pichler, A. Teuner, and B.J. Hosticka. An unsupervised texture segmentation algorithm with feature space reduction and knowledge feedback. *IEEE Transactions on Image Processing*, 7(1):53–61, January 1998.

[69] R. Porter and N. Canagarajah. A robust automatic clustering scheme for image segmentation using wavelets. *IEEE Transactions on Image Processing*, 5(4):662–665, April 1996.

[70] T. Randen. Brodatz textures. Available from: <http://www.ux.his.no/ tranden/brodatz.html>.

[71] E. Ribeiro and E.R. Hancock. Adapting scale by minimising spectral defocusing for shape from texture. In *International Conference on Image Processing*, pages III: 893–896, 2000.

[72] E. Ribeiro, P.L. Worthington, and E.R. Hancock. An eigendecomposition method for shape-from-texture. In *International Conference on Pattern Recognition*, pages I: 758–761, 2000.

[73] J. Sato and R. Cipolla. Extracting the affine transformation from texture moments. In *European Conference on Computer Vision*, pages 165–172, 1994.

[74] J. Sato and R. Cipolla. Extracting group transformations from image moments. *Computer Vision and Image Understanding*, 73(1):29–42, January 1999.

[75] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *International Conference on Computer Vision*, pages 636–643, 2001.

[76] W.B. Seales and O.D. Faugeras. Building three-dimensional object models from image sequences. *Computer Vision and Image Understanding*, 61(3):308–324, May 1995.

[77] K.A. Stevens. The information content of texture gradients. *Biological Cybernetics*, 42:95–105, 1981.

[78] J.V. Stone and S.D. Isard. Adaptive scale filtering: A general-method for obtaining shape from texture. *Pattern Analysis and Machine Intelligence*, 17(7):713–718, July 1995.

[79] B.J. Super and A.C. Bovik. Shape from texture using local spectral moments. *Pattern Analysis and Machine Intelligence*, 17(4):333–343, April 1995.

[80] D. Terzopoulos. The computation of visible-surface representations. *Pattern Analysis and Machine Intelligence*, 10(4):417–438, July 1988.

[81] J.T. Todd and R.A. Akerstrom. Perception of three-dimensional form from patterns of optical texture. *Journal of Experimental Psychology: Human Perception and Performance*, 13(2):242–255, 1987.

[82] J.K. Udupa. Determination of 3-d shape parameters from boundary information. *Computer Graphics and Image Processing*, 17(1):52–59, 1981.

[83] Univerity of Oxford. Visual Geometry Group software. Available from: <http://www.robots.ox.ac.uk/~vgg/software>.

[84] M. Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, November 1995.

[85] T. Wei and R. Klette. Depth recovery from noisy gradient vector fields using regularization. In *Computer Analysis of Images and Patterns*, pages 116–123, 2003.

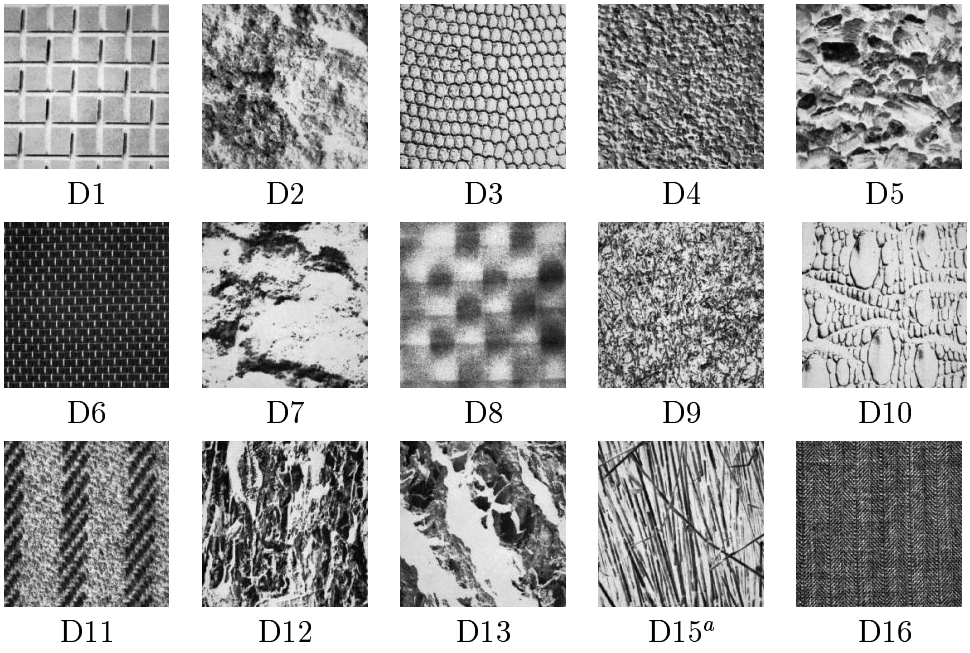
[86] A.P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1-3):17–45, August 1981.

[87] Y. Yu and J.T. Chang. Shadow graphs and surface reconstruction. In *European Conference on Computer Vision*, pages 31–45, 2002.

Appendix A

The Brodatz Textures

These textures were used to test the new method for the orthographic camera model in Chapter 3, and the method for estimating the affine transformation between textures in Chapter 4.

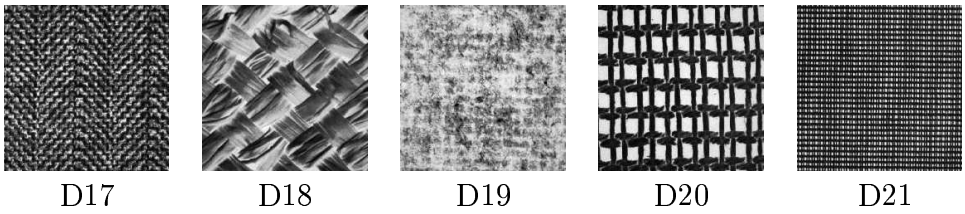


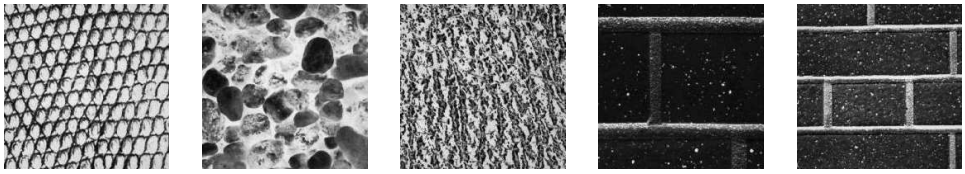
D1 D2 D3 D4 D5

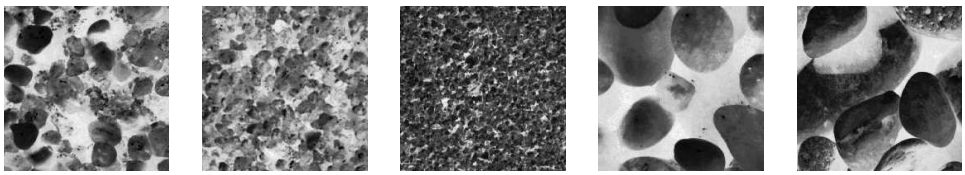
D6 D7 D8 D9 D10

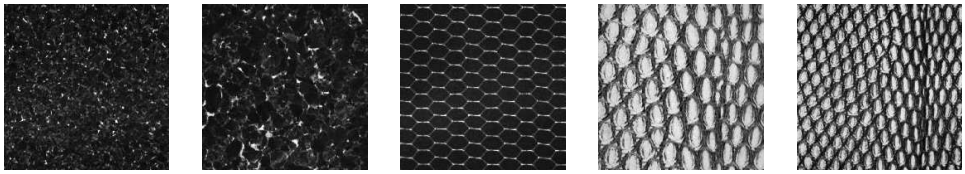
D11 D12 D13 D15[a] D16

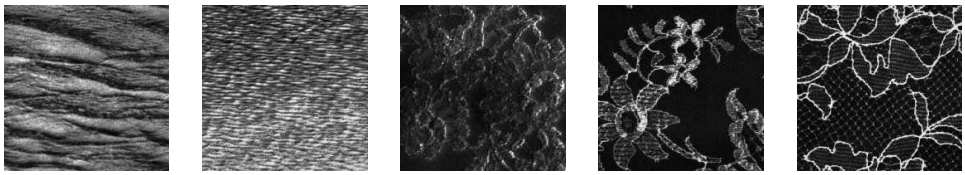
[a]The texture D14 was not available electronically at the time that these textures were collected from the web and so was not used for any experiments.


D17 D18 D19 D20 D21


D22 D23 D24 D25 D26

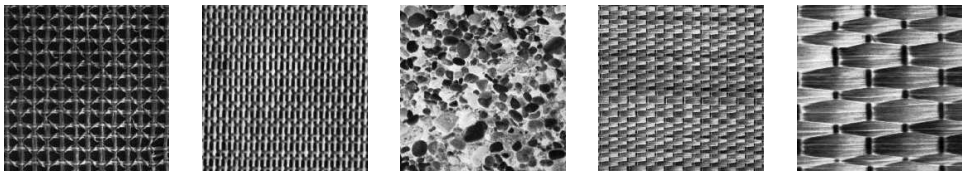

D27 D28 D29 D30 D31

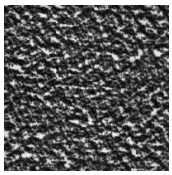

D32 D33 D34 D35 D36

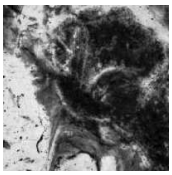

D37 D38 D39 D40 D41


D42 D43 D44 D45 D46

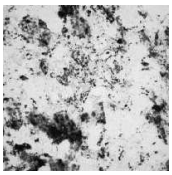

D47 D48 D49 D50 D51

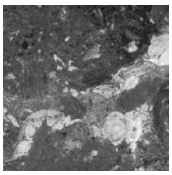

D52 D53 D54 D55 D56

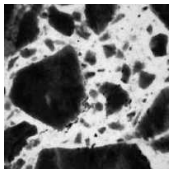

D57

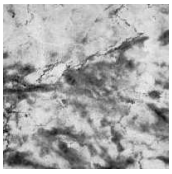

D58


D59


D60

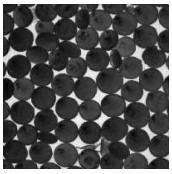

D61

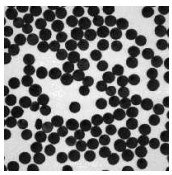

D62

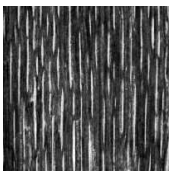

D63

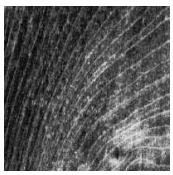

D64

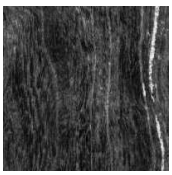

D65

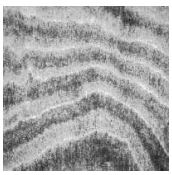

D66

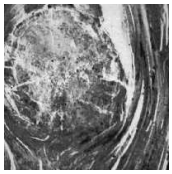

D67

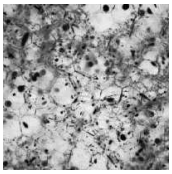

D68


D69

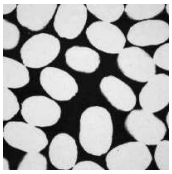

D70

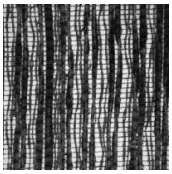

D71

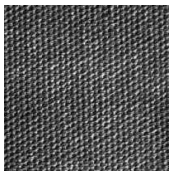

D72

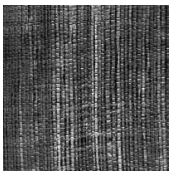

D73

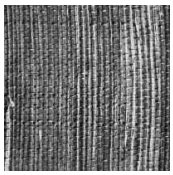

D74

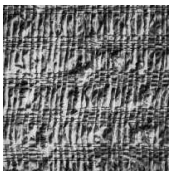

D75

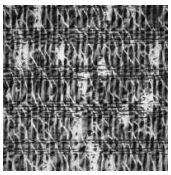

D76


D77

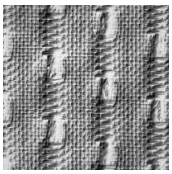

D78

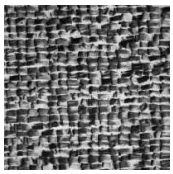

D79

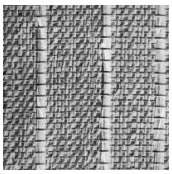

D80

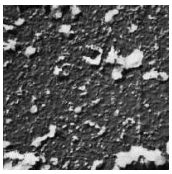

D81

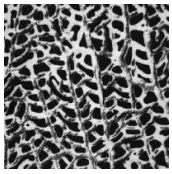

D82


D83


D84

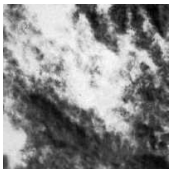

D85


D86

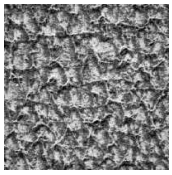

D87


D88


D89


D90


D91

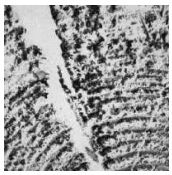

D92

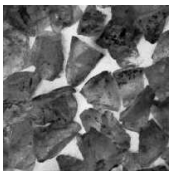

D93


D94

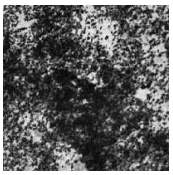

D95

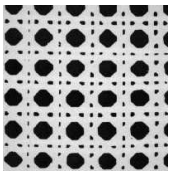

D96

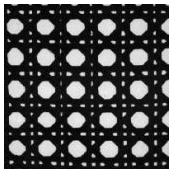

D97

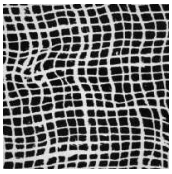

D98

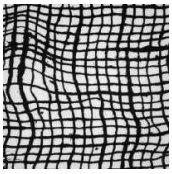

D99

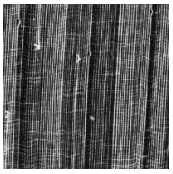

D100

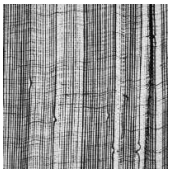

D101

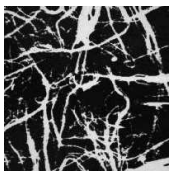

D102

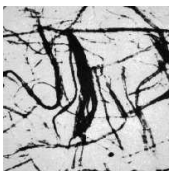

D103

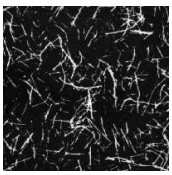

D104


D105

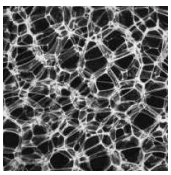

D106

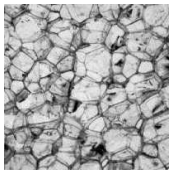

D107


D108


D109


D110


D111


D112