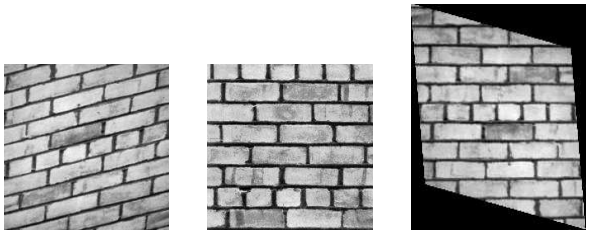
Estimating the affine transformation between textures

Angeline M. Loh
School of Computer Science and Software Engineering
The University of Western Australia

Andrew Zisserman
Department of Engineering Science
University of Oxford

Abstract

A method is presented that solves the affine transformation between two images of the same texture. This problem is encountered in many areas of Computer Vision such as object retexturing and it is an essential part of many Shape-from-Texture algorithms. The new method is based on transforming each image such that the texture is isotropic, then solving the similarity transformation between the isotropic versions. We point out the difficulties in using other methods such as feature point methods. Quantitative results are given for images created with the Brodatz textures, followed by results for real images. This is followed by an application of the method: placing an object on the textured surface.



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

Figure 1. Computing the affine transformation between two textures. The image in (a) has been transformed to give (c) such that the texture is the same as in (b).

1 Introduction

Estimating the affine transformation between two images is a common problem in Computer Vision. The affine transformation comprises a slant, rotation and scale operation that approximately transforms one image into the other. A common approach to estimating the affine transformation is to identify "interesting" points (such as Harris points [2] or SIFT points [4]) in both images and to match together corresponding points after assigning some feature 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. A slightly different problem involves calculating the affine transformation that relates two views of different portions of the same texture (see Figure 1).

In this problem individual texture elements (texels) may be distributed differently in the two images, and in this case the feature point approach will not work; for example see Figures 2(a) and (b) which show two different portions of the same texture. In Figures 2(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. In a similar fashion, many other textures will cause difficulties when a feature point matching algorithm is applied; examples include leaves scattered on the ground, pebbles and spots on a cheetah. Feature matching is also inherently difficult on textures because, by definition, they have repeating structures.

This paper proposes a method for estimating the affine transformation between two patches of texture. The method firstly transforms both patches of texture to an isotropic state so that they differ only by a rotation and scale. The required rotation and scale are then calculated from which the entire affine transformation can be constructed.

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 [6] match together 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

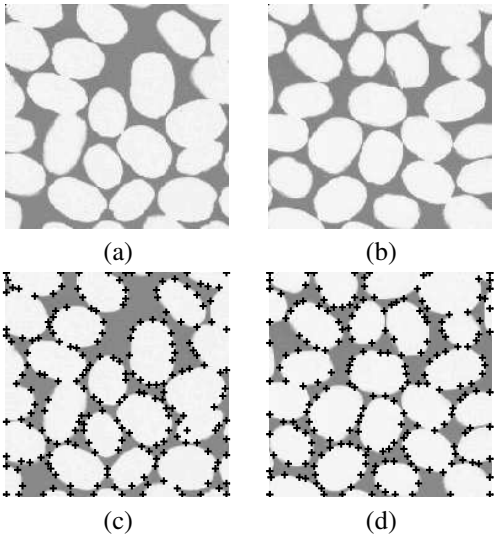


Figure 2. Two different portions of the same texture. Figures (c) and (d) show where features have been found in (a) and (b) respectively.

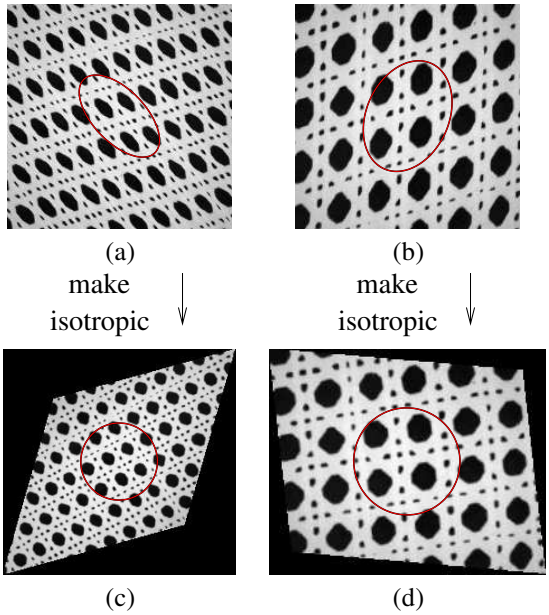


Figure 3. Basis of algorithm. The original images in (a) and (b) are made isotropic to give (c) and (d) respectively. The latter pair differ by a scale and rotation.

the ordering, of peaks. Malik and Rosenholtz [5] 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 1. The new method presented in this paper will work even if there is large difference between the two textures.

3 The method

Starting with two images of the same texture that has undergone two different affine transformations (for example see Figures 3(a) and 3(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 [7], who transformed textured regions to an isotropic state in order to assign an affine invariant texture descriptor.

Textures are made isotropic by firstly calculating the second moment matrix of each texture. The second moment matrix of a region Ω with intensity I is given by

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

This matrix may be represented as an ellipse, as superimposed onto Figures 3(a) and 3(b). The ellipse is locus of distances $\mathbf{x}$ from the centroid that satisfy $\mathbf{x^T M x} = \text{constant}$.

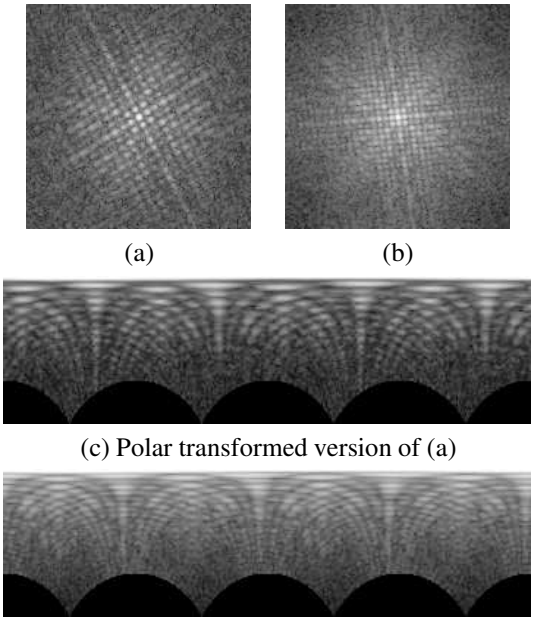
The transformation $\mathbf{A}$ that makes the region isotropic is required to satisfy $\mathbf{A^T A} = \mathbf{M}$. One solution for $\mathbf{A}$ is given by replacing the eigenvalues of $\mathbf{M}$ by their square roots. Figures 3(c) and 3(d) show the isotropic versions and hence the second moments are now represented as superimposed circles.

The textures in Figures 3(c) and 3(d) differ by a scale and rotation, and the next step of the algorithm is to calculate both of these. In theory one may adjust for scale using information from the second moment matrices; if the second moment matrices for the first and second isotropic textures are given by $\begin{pmatrix} S_1\, 0 \\ 0\, S_1 \end{pmatrix}$ and $\begin{pmatrix} S_2\, 0 \\ 0\, S_2 \end{pmatrix}$, then the first texture is made the same scale as the second by transforming the image by spatially rescaling it by $\sqrt{\frac{S_1}{S_2}}$. In practice however, the second moment matrices are heavily affected by phenomena such as blur, and so scale cannot be accounted for robustly in this way. In this current work, the affine transformation between two textures is calculated up to scale.

On the other hand, the *rotation* between the isotropic textures may be calculated easily and efficiently. This is done by computing the Fourier transform of each isotropic tex-

ture, as shown in Figures 4(a) and (b) where the Fourier transforms of Figures 3(c) and 3(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.)

Next we convert each Fourier transform to polar coordinates, as shown in Figures 4(c) and (d), where angle is represented by the horizontal axis and distance from the center is represented by the vertical axis. Then we take the mean



(a) (b)

(c) Polar transformed version of (a)

(d) Polar transformed version of (b)

Figure 4. Finding the rotation angle. (a) and (b) show the Fourier transforms of Figures 3(c) and (d) respectively.

along each column of Figures 4(c) and (d), to produce two signals which are phase-shifted versions of each other. A correlation is used to calculate the necessary shift which then corresponds to the rotation angle between the textures.

3.1 Summary of algorithm

To summarise, the steps of the algorithm are

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}.

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 5, the transformation from i to j is $\mathbf{T_1 T_2 T_3}^{-1}$.

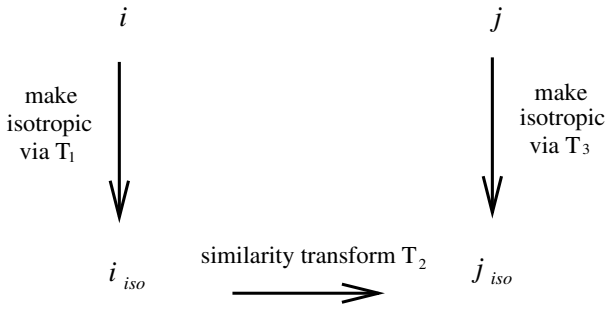


Figure 5. Finding the affine transformation between two images i and j of the same texture.

3.2 Details of Implementation

The following 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. To illustrate, 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 our 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 4(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 π-shift. This may be reflected as an initial rotation of π applied to the image i as part of the estimated affine transformation. This will be investigated in future work.

4 Results

4.1 Tests with Brodatz textures

To test the algorithm, pairs of images were created using the Brodatz textures [1]. Random affine transformations were applied to the textures to create the pairs shown in the first two columns of Figure 6. Using these pairs of images, the algorithm estimated the transformation from the first image to the second. The last column of Figure 6 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 [3] into

$$\begin{aligned} \mathbf{T} &= \mathbf{U}\,\mathbf{D}\,\mathbf{V}^{\mathbf{T}} \\ &= (\mathbf{U}\,\mathbf{D}\,\mathbf{U}^{\mathbf{T}})\,\mathbf{U}\,\mathbf{V}^{\mathbf{T}} \end{aligned} \tag{1}$$

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 $\mathbf{U}\,\mathbf{V}^{\mathbf{T}}$. The image is then rotated by $\mathbf{U}^{\mathbf{T}}$, 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 rot(x) is a matrix that rotates by x we can rewrite this as

$$\mathbf{T} = rot(-\tau) \begin{pmatrix} k & 0 \\ 0 & 1 \end{pmatrix} rot(\tau)\; rot(\theta)\; S \tag{2}$$

where S is a scaling matrix, 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 accurate 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 τ. The errors are shown in Figure 7. 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 an isotropic texture, that is one with no rotational preference. For such textures *any* initial rotation may be reasonably included in the estimated affine transformation. This accounts for the large error in θ seen for this example.

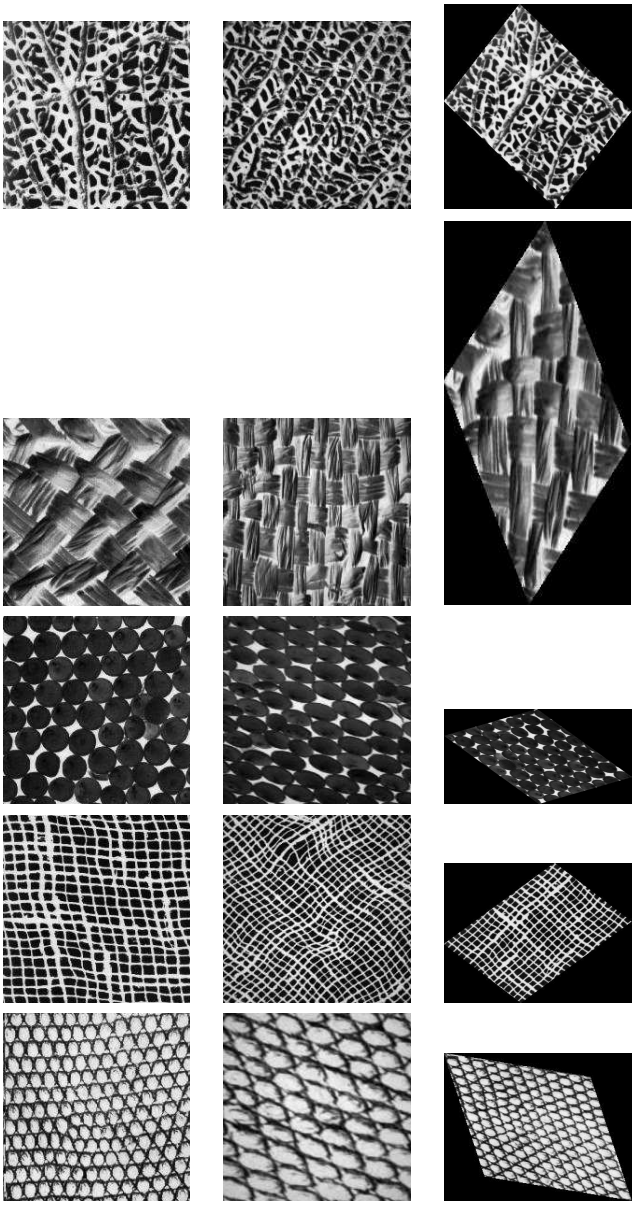
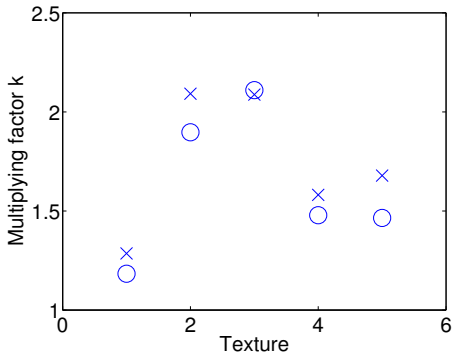


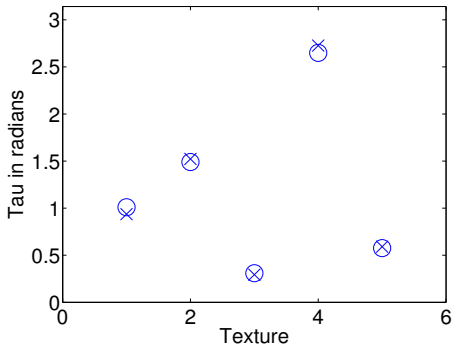
Figure 6. Results of tests on Brodatz textures. The first and second columns are input images. The third column shows the affine-transformed version of the first column such that the texture resembles the second column, up to scale.

4.2 Tests with real images

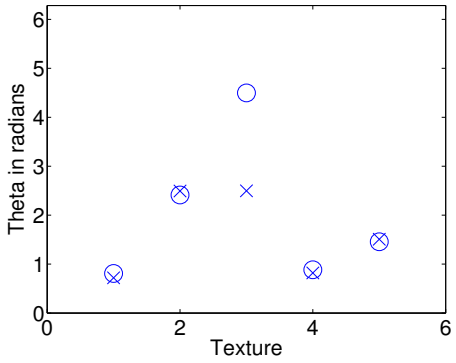
The algorithm was also tested on real images. Figure 8 shows the objects that were photographed. Close up pictures of textures were taken from different angles as seen in



(a) Real and estimated k



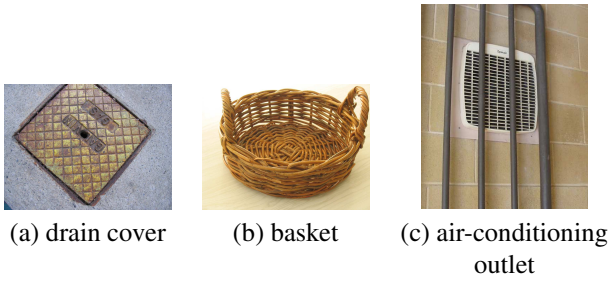
(b) Real and estimated τ



(c) Real and estimated θ

Figure 7. Errors in affine parameters. The crosses denote estimated values and the circles denote real values.

the first two columns of Figure 9. The third column shows the transformed version of images from the first column, such that they resemble the textures seen in the second column.



(a) drain cover (b) basket (c) air-conditioning outlet

(d) necktie (e) tea towel

Figure 8. Objects that were photographed.

4.3 A simple application of the method

As mentioned previously, the method may be used in a number of applications including Shape-from-Texture and object retexturing. Here we show another application: placing objects in a scene. Figure 10(a) shows an embroidered motif that we wish to place on the teatowel from the last example in Figure 9. In Figure 9 we saw the teatowel when viewed frontally i.e. viewed with no slant (in the first column), as well as the teatowel viewed with some slant angle and rotation (in the second column). Using the 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 teatowel; the result is given in Figure 10(d).

5 Conclusions

A new method has been shown 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. The new method has been shown to give good quantitative results when tested on a set of image created using the Brodatz textures. Real images gave good qualitative results. Future work includes improving the robustness of the estimation of scale.

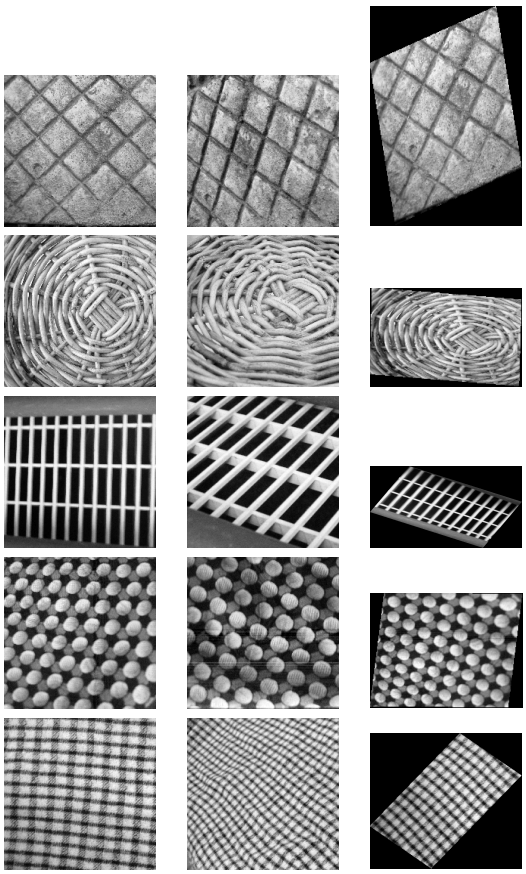
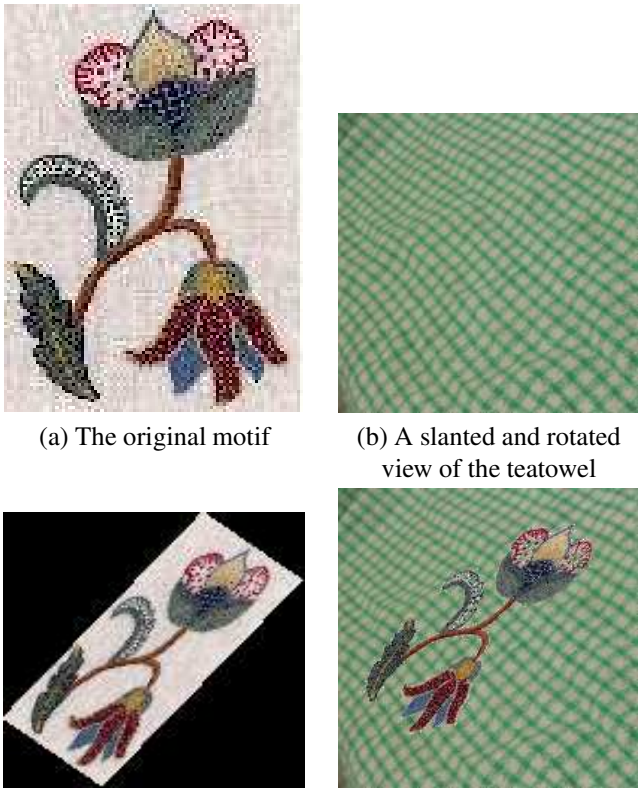


Figure 9. Results from real images. The first and second columns show the input images. The third column shows the transformed versions of the first column so that the texture matches that in the second column.



(a) The original motif

(b) A slanted and rotated view of the teatowel

(c) The transformed motif

(d) The motif placed on the tablecloth

Figure 10. Placing an embroidered motif on the teatowel.

6 Acknowledgments

This work was funded by an Australian Postgraduate Award and a Western Australian IVEC Doctoral Scholarships top-up. The authors acknowledge the assistance of the University of Western Australia Graduates Association for partially funding research undertaken in the UK through the Bankwest Travel Award.

References

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

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

[3] A.M. Loh and R. Hartley. Shape from non-homogeneous, non-stationary, anisotropic, perspective texture. In *BMVC05*, 2005.

[4] D. Lowe. Distinctive image features from scale invariant keypoints. In *IJCV04*, pages 91–110, 2004.

[5] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *IJCV*, 23(2):149–168, June 1997.

[6] E. Ribeiro, P.L. Worthington, and E.R. Hancock. An eigendecomposition method for shape-from-texture. In *ICPR00*, pages Vol I: 758–761, 2000.

[7] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *ICCV01*, pages 636–643, 2001.