9 Shape from Single-image Cues

Je voudrais pas crever
Sans savoir si la lune
Sous son faux air de thune
A un côté pointu[1]

Boris Vian, *Je Voudrais pas Crever*

The subject of this chapter, inferring the shape of objects from a single intensity image, is a classic problem of computer vision. We do not consider range images, in which shape is already explicit.

Chapter Overview

Section 9.1 lists the main methods for inferring shape from intensity images.

Section 9.2 introduces the concept of *reflectance map* and the problem of *shape from shading* from a physical and mathematical viewpoint.

Section 9.3 discusses a method for estimating *albedo* and *illuminant direction*.

Section 9.4 describes a method for extracting the shape of an object from a shading pattern.

Section 9.5 shows how shape can be computed from the distortion of 3-D textures caused by the imaging projection, considering deterministic and statistical textures.

What You Need to Know to Understand this Chapter

- Working knowledge of Chapters 2 and 4.
- Working knowledge of Fast Fourier Transform (FFT).

[1] I would not want to die without knowing whether the moon, behind her false look of coin, has a pointed side.

9.1 Introduction

A common experience of our everyday life is the perception of solid shape, indeed so common that it is hard to appreciate its full complexity. We perceive without effort the shape of complex objects like faces and cars, irregular surfaces like mountains, and even of changing surfaces like a tree in the wind. In spite of this apparent simplicity, shape reconstruction has proven a very hard problem for computer vision; indeed, one which is solved only partially. Many methods, collectively known as *shape from X*, have been proposed for reconstructing 3-D shape from intensity images, and the algorithms in the previous two chapters can be regarded as shape-from-X methods (i.e., shape from stereo, shape from motion). Shape-from-X methods exploit a large variety of image cues; Table 9.1 mentions the best-known ones, and classifies them according to two important characteristics: the number of images needed, and whether or not the method requires purposive modification of the vision system's parameters; that is, whether the method is *active* or *passive*.

The active-passive distinction is worth commenting. In *active* methods, the vision system's parameters are modified *purposively*; for instance, a shape-from-defocus system controls the focus of the lens to acquire two or more out-of-focus images, and use estimates of the focus level at each pixel to compute local shape. Shape-from-motion systems can be either active (if the sensors are moved purposively to generate a relative motion between sensor and scene) or passive (if relative motion occurs without purposive sensor motion).[2]

This chapter concentrates on *shape reconstruction from a single intensity image*. Notice that we make no assumptions on the shape of the objects in the scene. This situation is similar to trying to make out the shapes of unknown, solid objects from a single photograph. Here is a concise statement of the problem.

Problem Statement

Given a single image of unknown objects, reconstruct the shape of the visible surfaces.

As suggested by the table above, various cues can be exploited to tackle the problem. This chapter discusses two of them in detail, *shading* and *texture*, and gives algorithms for computing *shape from shading* and *shape from texture*. The reasons for choosing these two methods are that they do not require special hardware (unlike for instance shape from zoom or focus/defocus, which require a motorized lens), and do not depend on image preprocessing (unlike for instance shape from contours, which assumes that the contours of objects have been identified).

---

[2] *Active vision* denotes a whole area of computer vision, the scope of which goes far beyond reconstruction: in general, the term refers to *strategies for observation*. In the active vision paradigm, images are analyzed and acquired purposefully, in order to accomplish specific tasks. Thus, sensor and observer interact continuously, and vision is not just about interpreting isolated snapshots or sequences. For more on active vision, see the Further Readings.

Shape from	How many images	Method type	Find more in
Stereo	2 or more	passive	Chapter 7
Motion	a sequence	active/passive	Chapter 8
Focus/defocus	2 or more	active	Further Readings
Zoom	2 or more	active	Further Readings
Contours	single	passive	Further Readings
Texture	single	passive	this chapter
Shading	single	passive	this chapter

Table 9.1 Shape-from-X methods and their classification.

☞ Notice that our problem statement implies the use of intensity images, as range images are themselves a representation of 3-D shape. In fact, the problem of this chapter can be regarded as one of *producing a range image from an intensity one.*

9.2 Shape from Shading

Shape from shading uses the pattern of lights and shades in an image to infer the shape of the surfaces in view. This sounds very useful, also considering that single intensity images are easier to acquire than stereo pairs or temporal sequences. Unfortunately, the problems we face in shape from shading are considerably more complicated that any others encountered so far. There are at least two main reasons for this fact: the first has to do with the *physics* of the problem, the second with the *mathematics*. We will have to look into both in order to arrive at a method for computing shape from shading.

In spite of its difficulties, shape-from-shading has been used successfully in various applications. A typical example is astronomy, where shape from shading is used to reconstruct the surface of a planet from photographs acquired by a spacecraft. An example is given in Figure 9.1, which takes forward the Magellan example of Chapter 7. Here, a shape-from-shading algorithms refines an initial range image obtained by stereopsis.

9.2.1 The Reflectance Map

The fact that shape *can* be reconstructed from shading is due to the existence of a fundamental equation that links image intensity and surface slope. In order to establish this equation we need to introduce the important concept of *reflectance map*. We first discuss the simple case of Lambertian surfaces.

In Chapter 2 we have seen that a uniformly illuminated Lambertian surface appears equally bright from all viewpoints. The radiance L at a 3-D point, $\mathbf{P}$, is proportional to the cosine of the angle between the surface normal and the direction of the illuminant:

$$L(\mathbf{P}) = \rho \mathbf{i}^\top \mathbf{n}, \tag{9.1}$$

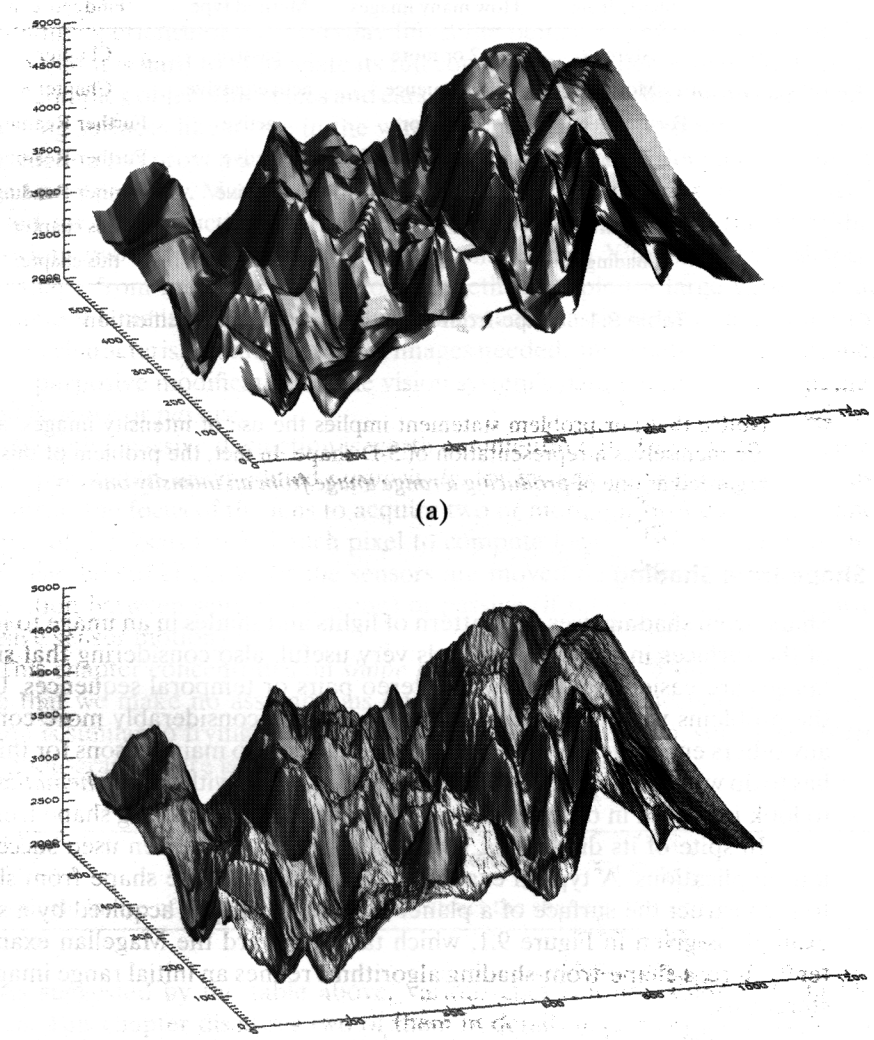


Figure 9.1 (a): 3-D rendering of the surface reconstructed by stereopsis from the stereo pair on page 142. (b): the refined surface obtained by the shape-from-shading algorithm; notice the increased level of detail. Courtesy of Alois Goller, Institute for Computer Graphics and Vision, Technical University of Graz.

with $\mathbf{i}$ the vector which gives the illuminant direction (pointing towards the light source), $\mathbf{n}$ the surface normal at $\mathbf{P} = [X, Y, Z]^\top$, and ρ the *effective albedo*; that is, the real albedo times the intensity of illuminant. To stress the dependence of the radiance on the surface normal, we rewrite (9.1) as

$$R_{\rho,\mathbf{i}} = \rho \mathbf{i}^\top \mathbf{n}. \tag{9.2}$$

We use R instead of L because (9.2) tells you how light is reflected by a Lambertian surface in terms of the surface normal for *any* given albedo and illuminant. This is a particular example of *reflectance map*. In general, the function $R_{\rho,\mathbf{i}}$ is more complicated or known only numerically through experiments.

9.2.2 The Fundamental Equation

In Chapter 2, we also met the *image irradiance equation*, which, denoting with $\mathbf{p} = [x, y]^\top$ the image of $\mathbf{P}$, we wrote as

$$E(\mathbf{p}) = L(\mathbf{P})\frac{\pi}{4}\left(\frac{d}{f}\right)^2 \cos^4 \alpha, \tag{9.3}$$

with $E(\mathbf{p})$ the brightness measured on the image plane at $\mathbf{p}$. We now make two important assumptions:

1. We neglect the constant terms in (9.3) and assume that the optical system has been calibrated to remediate the $\cos^4 \alpha$ effect.
2. We assume that all the visible points of the surface receive direct illumination.

Assumption 1 simplifies the mathematics of the problem, and Assumption 2 avoids dealing with secondary reflections. In these assumptions, combining (9.3) and (9.2) gives

$$E(\mathbf{p}) = R_{\rho,\mathbf{i}}(\mathbf{n}). \tag{9.4}$$

Equation (9.4) constrains the direction of the surface normal at $\mathbf{P}$, and is the *fundamental equation of shape from shading*.

In order to make profitable use of (9.4), we need to express the normal in terms of the surface slopes. To this purpose, we further assume that the visible surface

3. is far away from the viewer
4. can be described as $Z = Z(X, Y)$

Assumption 3 enables us to adopt the *weak-perspective camera model*, and write

$$x = f\frac{X}{Z_0}$$

and

$$y = f\frac{Y}{Z_0},$$

with Z_0 the average distance of the surface from the image plane. Thus, by a suitable rescaling of the horizontal and vertical axis, the surface, Z, can be thought of as a function of the (x, y) coordinates on the image plane,

$$Z = Z(x, y).$$

The surface slopes can now be computed by simply taking the x and y partial derivatives of the vector $[x, y, Z(x, y)]^\top$, which gives

$$[1, 0, \partial Z/\partial x]^\top \quad \text{and} \quad [0, 1, \partial Z/\partial y]^\top.$$

Both vectors lie on the tangent plane to the surface, and hence their normalized vector product is the unit normal $\mathbf{n}$ to the surface at (x, y) (Appendix, section A.5); that is,

$$\mathbf{n} = \frac{1}{\sqrt{1 + p^2 + q^2}}[-p, -q, 1]^\top, \tag{9.5}$$

where we have denoted with p and q the partial derivatives $\partial Z/\partial x$ and $\partial Z/\partial y$ respectively. By means of (9.2) and (9.5), we can rewrite (9.4) as

$$E(x, y) = \frac{\rho}{\sqrt{1 + p^2 + q^2}}\mathbf{i}^\top[-p, -q, 1]. \tag{9.6}$$

Equation (9.6) is the typical starting point of many shape from shading techniques. Unfortunately, it is of a great mathematical complexity: It is a nonlinear partial differential equation through $p = p(x, y)$ and $q = q(x, y)$, the slopes of the unknown surface $Z = Z(x, y)$, and depends on quantities not necessarily known (albedo, illuminant, and boundary conditions).[3]

The best way to grasp the meaning of (9.6) is to produce a synthetic shaded image yourself. To do this, you adopt the Lambertian assumption, write the equation of your favorite surface in the form $Z = Z(x, y)$, choose values for ρ and $\mathbf{i}$, and compute the numerical partial derivatives of the surface Z over a pixel grid. You can then compute the corresponding image brightness according to (9.6). Two examples of this simple procedure to produce a shaded image are shown in Figure 9.2.

☞ Note that, for certain illuminants, some image locations might violate Assumption 2 (Figure 9.2(*b*)), and the image brightness computed through (9.6) become negative; in this case, the brightness should be set to 0. Therefore, the right hand side of (9.6) should always be regarded as

$$\max\{0, \frac{\rho}{\sqrt{1 + p^2 + q^2}}\mathbf{i}^\top[-p, -q, 1]\}. \tag{9.7}$$

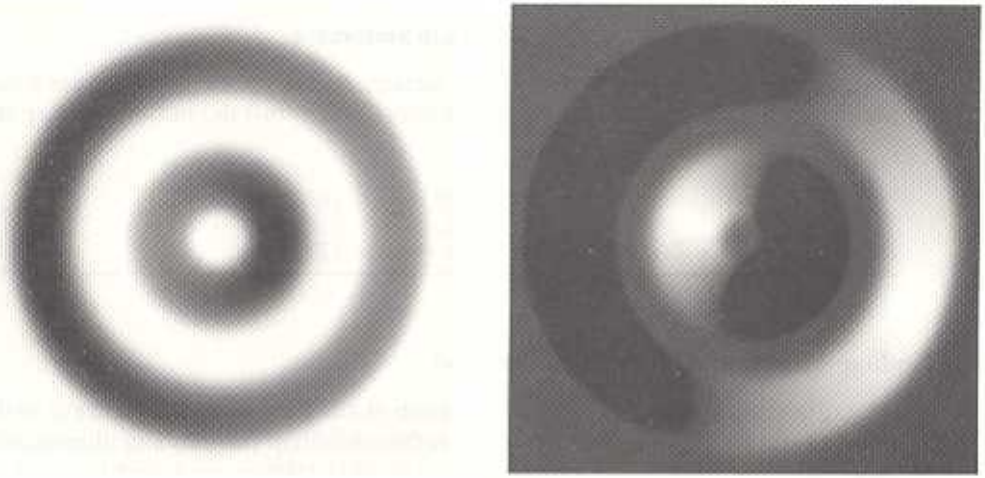
The number of unknowns in (9.6) seems to suggest that this equation does not provide enough constraints to reconstruct p and q at all pixels: in the discrete setting ($N \times N$ pixels), it seems that we end up with N^2 equations, one for each pixel, in $2N^2$ unknowns, the p and the q. But this is not true, because the p and q are not independent: since $Z_{xy} = Z_{yx}$, we have N further equations of the kind

$$p_y = q_x.$$

We will come back to this point in the discussion of a method for solving (9.6).

---

[3] Just like an ordinary differential equation depends on initial conditions, a partial differential equation depends on boundary conditions. In essence, the solution is determined uniquely only if certain *a priori* information on the solution is available. This information is typically given at the boundary of the domain of interest (in our case, the image boundaries) or on some particular curves.



(b)

(a)

Figure 9.2 Two images of the same Lambertian surface seen from above but illuminated from different directions and 3-D rendering of the surface. Practically all the points in the top left image receive direct illumination ($\mathbf{i} = [0.20, 0, 0.98]^\top$); some regions of the top right image are in the dark due to self-shadowing effects ($\mathbf{i} = [0.94, 0.31, 0.16]^\top$).

We can now summarize the various assumptions made, and state the problem of shape from shading in more technical terms.

Assumptions

1. The acquisition system is calibrated so that the image irradiance, $E(\mathbf{p})$, equals the scene radiance, $L(\mathbf{P})$, with $\mathbf{p} = [x, y]^\top$ image of the 3-D point $\mathbf{P} = [X, Y, Z]^\top$.
2. All the visible surface points receive direct illumination.
3. The surface is imaged under weak perspective.
4. The optical axis is the Z axis of the camera, and the surface can be parameterized as

$$Z = Z(x, y).$$

Problem Statement

Given the reflectance map of the viewed surface, $R = R_{\rho,\mathbf{i}}(p, q)$, and full knowledge of the parameters ρ and $\mathbf{i}$ relative to the available image, reconstruct the surface slopes, p and q, for which

$$E(x, y) = R_{\rho,\mathbf{i}}(p, q),$$

and the surface $Z = Z(x, y)$ such that $\partial Z/\partial x = p$ and $\partial Z/\partial y = q$.

9.3 Finding Albedo and Illuminant Direction

We are one last step away from a shape-from-shading method: We still have to deal with the problem that the parameters of the reflectance map (albedo and illuminant) can be unknown. This problem is difficult *per se*; in what follows, we restrict our attention to the relatively simpler Lambertian case.

9.3.1 Some Necessary Assumptions

Similarly to shape from shading itself, the problem of estimating albedo and illuminant direction is only apparently simple; in fact, an effective way to provide good estimates under general circumstances, or even in the simpler Lambertian case, must still be found. The major difficulty is the fact that *the problem is heavily underconstrained*. To overcome this difficulty, we must make assumptions on either the shape of the viewed surface or the distribution of the surface normals; here, we favor the latter view.

Our task is therefore to derive a method which, under appropriate assumptions, estimates albedo and illuminant direction. We begin by stating the problem in more formal terms.

Assumptions and Problem Statement

Under the same assumptions of shape from shading, and with the further assumptions that

1. the surface imaged is Lambertian, and
2. the directions of the surface normals are distributed uniformly in 3-D space,

determine albedo and illuminant direction from a single image of the surface.

Let us fix the notation and select a reasonable distribution of the surface normals in a generic scene. It is customary to describe surface normals by means of the *tilt* and *slant* angles, α and β (see Figure 9.3 with $\alpha = \tau$ and $\beta = \sigma$ respectively). By definition, we have that $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi/2]$ and

$$\mathbf{n} = [\cos\alpha \sin\beta, \sin\alpha \sin\beta, \cos\beta]^\top. \tag{9.8}$$

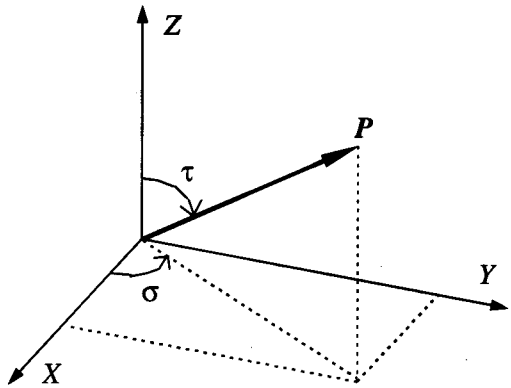


Figure 9.3 Geometric interpretation of tilt and slant.

In our assumptions, the distribution of the normal vectors, $\mathcal{P}$, *as seen from the image plane*, is

$$\mathcal{P}(\alpha, \beta) = \frac{\cos \beta}{2\pi}. \tag{9.9}$$

In agreement with intuition, $\mathcal{P}$ is uniform in α but, due to foreshortening, the number of normal vectors with slant equal to β is proportional to $\cos \beta$.

☞ Like all approximations, especially coarse ones, (9.9) does not describe accurately the distribution of the normals of *every possible* surface as seen from the image plane. However it does a good job on average, and makes it possible to obtain estimates of albedo and illuminant direction in the absence of any guess.

9.3.2 A Simple Method for Lambertian Surfaces

This section describes method for recovering albedo and illuminant direction composed by three steps:

1. precompute the averages of the image brightness and its derivatives, using the hypothesized distribution
2. evaluate the same averages from the image brightness
3. enter the results in equations that can be solved for albedo and illuminant direction

We start by denoting with σ and τ, respectively, the slant and tilt angle of the illuminant, so that we can write

$$\mathbf{i} = [\cos \tau \sin \sigma, \sin \tau \sin \sigma, \cos \sigma]^\top. \tag{9.10}$$

We now compute the average of the image brightness, as given by (9.4), using the distribution in (9.9). By means of (9.8) and (9.10), the image brightness can be written as a function of α and β as

$$E(\alpha, \beta) = \rho(\cos\alpha \sin\beta \cos\tau \sin\sigma + \sin\alpha \sin\beta \sin\tau \sin\sigma + \cos\beta \cos\sigma). \quad (9.11)$$

The average, $< E >$, becomes therefore

$$< E >= \int_0^{2\pi} d\alpha \int_0^{\pi/2} d\beta \mathcal{P}(\alpha, \beta) E(\alpha, \beta),$$

This integral breaks down into three additive terms, of which the first two vanish (because the tilt, α, is integrated over a full period), and the third yields

$$< E >= \frac{\pi}{4} \rho \cos\sigma. \quad (9.12)$$

The interesting fact about this result is that, as we assumed that the surface normals are distributed according to (9.9), we can compute $< E >$ as the average of the brightness values of the given image, and hence look at (9.12) as an equation for albedo and slant.

A similar derivation for the average of the square of the image brightness, $< E^2 >$, (see Exercise 9.1 for some hints) give

$$< E^2 >= \frac{1}{6} \rho^2 \left(1 + 3\cos^2\sigma\right). \quad (9.13)$$

From (9.12) and (9.13), it is immediate to recover albedo and slant as

$$\rho = \frac{\gamma}{\pi} \quad (9.14)$$

and

$$\cos\sigma = \frac{4 < E >}{\gamma}, \quad (9.15)$$

where $\gamma = \sqrt{6\pi^2 < E^2 > -48 < E >^2}$.

We are still left with the problem of estimating τ, the tilt of the illuminant. This can be obtained from the spatial derivatives of the image brightness, E_x and E_y. Through some simple but lengthy algebra that we omit (if you are curious, see the Further Readings), one finds

$$\tan\tau = \frac{< \hat{E}_y >}{< \hat{E}_x >} \quad (9.16)$$

with $< \hat{E}_x >$ and $< \hat{E}_y >$ the averages of the horizontal and vertical components of the direction of the image spatial gradient, $\left[\hat{E}_x, \hat{E}_y\right]^\top = (E_x^2 + E_y^2)^{\frac{1}{2}} \left[E_x, E_y\right]^\top$. We now summarize this method:

Algorithm APPRX_ALBEDO_ILLUM_FINDER

The input is an intensity image of a Lambertian surface.

1. Compute the average of the image brightness, $< E >$, and of its square, $< E^2 >$.
2. Compute the spatial image gradient, $[E_x, E_y]^\top$, and let $[\hat{E}_x, \hat{E}_y]^\top$ be the unit vector giving the direction of $[E_x, E_y]^\top$. Compute the average of both components, $< \hat{E}_x >$ and $< \hat{E}_y >$.
3. Estimate ρ, $\cos\sigma$, and $\tan\tau$ through (9.14), (9.15), and (9.16).

The output are estimates of ρ, $\cos\sigma$, and $\tan\tau$.

☞ This method gives reasonably good results, though it fails for very small and very large slants. However, one of the hypothesis underlying the derivation of this method is *inconsistent*. It might have occurred to you that a surface whose normal vectors are uniformly distributed in 3-D space is likely to give rise to self-shadowing, especially for large slant of the illuminant direction. This means that in some of our precomputed integrals the image brightness was negative. To avoid this inconsistency, the integrals should be evaluated numerically using (9.7) as a definition of image brightness. For details on the consistent (but considerably more complicated) version of APPRX_ALBEDO_ILLUM_FINDER refer to the Further Readings. Curiously enough, the "consistent" method does not improve much the final result.

We are now ready to search for a solution to the shape from shading problem.

9.4 A Variational Method for Shape from Shading

We picked one of the many existing methods based on a variational framework, the solution of which gives the slopes of the unknown surface Z.

9.4.1 The Functional to be Minimized

Even under the simplifying Lambertian assumption, the direct inversion of (9.6) is a very difficult task. In essence, one has to solve a nonlinear partial differential equation in the presence of uncertain boundary conditions. For many such equations, even a slight amount of noise can means that the solution (*a*) does not exist, (*b*) is not unique, or (*c*) does not depend continuously on the data.[4] Our equation is no exception. If you are interested to the mathematical aspects of this problem you will find pointers in the Further Readings.

A typical trick to circumvent at least existence and continuity problems (conditions (*a*) and (*c*)) is to recast the problem in the *variational framework*. Instead of looking for an exact solution to (9.6), we allow for some small deviations between the

[4] In the mathematical literature, a problem for which at least one of the conditions (*a*), (*b*), or (*c*) holds is said to be *ill posed*.

image brightness and the reflectance map, and enforce a *smoothness constraint* which controls the smoothness of the solution. One possible way to implement this idea is to look for the minimum of a functional $\mathcal{E}$ of the form

$$\mathcal{E} = \int dxdy \left((E(x,y) - R(p,q))^2 + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2) \right), \tag{9.17}$$

in which the smoothness constraint is given by the sum of the spatial derivatives of p and q. The parameter λ is always positive and controls the relative influence of the two terms in the minimization process. Clearly, a large λ encourages a very smooth solution not necessarily close to the data, while a small λ promotes a more irregular solution closer to the data.

Unlike the case of deformable contours, the minimization of this functional cannot be performed effectively by means of a greedy algorithm and we have to make use of the full machinery of the calculus of variations.

9.4.2 The Euler-Lagrange Equations

The calculus of variations gives you a straightforward procedure to derive equations minimizing a generic functional,[5] the *Euler-Lagrange equations*. This section simply tells you how to set up these equations; refers to the Further Readings for more information.[6]

For a functional $\mathcal{E}$ which, like (9.17), depends on two functions p and q of two real variables x and y, and on their first order spatial derivatives, the Euler-Lagrange equations read

$$\frac{\partial \mathcal{E}}{\partial p} - \frac{\partial}{\partial x}\frac{\partial \mathcal{E}}{\partial p_x} - \frac{\partial}{\partial y}\frac{\partial \mathcal{E}}{\partial p_y} = 0,$$

and

$$\frac{\partial \mathcal{E}}{\partial q} - \frac{\partial}{\partial x}\frac{\partial \mathcal{E}}{\partial q_x} - \frac{\partial}{\partial y}\frac{\partial \mathcal{E}}{\partial q_y} = 0.$$

Since R is the only function of p and q in (9.17), and neither E or R depend on p_x, p_y, q_x, and q_y, the Euler-Lagrange equations associated with (9.17) become

$$-2(I - R)\frac{\partial R}{\partial p} - 2\lambda p_{xx} - 2\lambda p_{yy} = 0$$

and

$$-2(I - R)\frac{\partial R}{\partial q} - 2\lambda q_{xx} - 2\lambda q_{yy} = 0,$$

[5] It is far easier to write the equations than to solve them, however!

[6] For our purposes, the derivation of the Euler-Lagrange equations associated to a variational method is a rather boring and not much informative exercise of calculus. The real problem is not the derivation of the equations, but finding a good numerical algorithm for solving them.

which can be simplified to give

$$\Delta p = -\frac{1}{\lambda}(I - R)\frac{\partial R}{\partial p} \tag{9.18}$$

and

$$\Delta q = -\frac{1}{\lambda}(I - R)\frac{\partial R}{\partial q}, \tag{9.19}$$

with Δp and Δq denoting the *Laplacian* of p and q (that is, $\Delta p = p_{xx} + p_{yy}$ and $\Delta q = q_{xx} + q_{yy}$). Our next task is to solve (9.18) and (9.19) for p and q.

9.4.3 From the Continuous to the Discrete Case

It turns out that solving (9.18) and (9.19) is easier in the discrete than in the continuous case. Thus we immediately proceed to find the discrete counterpart of (9.18) and (9.19).

We start by denoting with $p_{i,j}$ and $q_{i,j}$ the samples of p and q over the pixel grid at the location (i, j). Through the usual formula for the numerical approximation of the second derivative (Appendix, section A.2), (9.18) and (9.19) become

$$-4p_{i,j} + p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1} = -\frac{1}{\lambda}(E(i, j) - R(p_{i,j}, q_{i,j})\frac{\partial R}{\partial p} \tag{9.20}$$

and

$$-4q_{i,j} + q_{i+1,j} + q_{i-1,j} + q_{i,j+1} + q_{i,j-1} = -\frac{1}{\lambda}(E(i, j) - R(p_{i,j}, q_{i,j})\frac{\partial R}{\partial q}. \tag{9.21}$$

☞ The partial derivatives of the reflectance map in (9.20) and (9.21) are either computed analytically (in the Lambertian case, for example) and then evaluated at $p_{i,j}$ and $q_{i,j}$, or evaluated numerically from the reflectance map itself.

The problem is now reduced to finding the slopes $p_{i,j}$ and $q_{i,j}$, solutions of (9.20) and (9.21), and determining the unknown surface $Z = Z(x, y)$ from them.

9.4.4 The Algorithm

We observe that (9.20) and (9.21) can be rewritten as

$$p_{i,j} = \bar{p}_{i,j} + \frac{1}{4\lambda}(E - R)\frac{\partial R}{\partial p} \tag{9.22}$$

and

$$q_{i,j} = \bar{q}_{i,j} + \frac{1}{4\lambda}(E - R)\frac{\partial R}{\partial q}, \tag{9.23}$$

with (Appendix, section A.2)

$$\bar{p}_{i,j} = \frac{p_{i+1,j} + p_{i-1,j} + p_{i,j+1} + p_{i,j-1}}{4}$$

and

$$\bar{q}_{i,j} = \frac{q_{i+1,j} + q_{i-1,j} + q_{i,j+1} + q_{i,j-1}}{4}.$$

As $\bar{p}_{i,j}$ and $\bar{q}_{i,j}$ are the averages of $p_{i,j}$ and $q_{i,j}$ over the four nearest neighbors, (9.22) and (9.23) can be turned into an *iterative scheme* that, starting from some initial configurations for the $p_{i,j}$ and $q_{i,j}$, advances from the step k to the step $k+1$ according to the updating rule

$$p_{i,j}^{k+1} = \bar{p}_{i,j}^{k} + \frac{1}{4\lambda}(E - R)\frac{\partial R}{\partial p}\Big|^{k} \tag{9.24}$$

and

$$q_{i,j}^{k+1} = \bar{q}_{i,j}^{k} + \frac{1}{4\lambda}(E - R)\frac{\partial R}{\partial q}\Big|^{k}. \tag{9.25}$$

However, where have all those boundary conditions gone? The answer is not easy, because in many cases of interest the boundary conditions are practically unknown. In this case, one attempts to impose "neutral" boundary conditions, like the so-called *natural* boundary conditions (p and q constant over the image boundary), or the *cyclic* boundary conditions (p and q wrapped around at the image boundary). Since we are going to compute the Fourier transform of p and q, in what follows we adopt the *cyclic* boundary conditions. If the boundary conditions are known, they should be enforced in the iterative scheme at every step.

9.4.5 Enforcing Integrability

As most pioneers of shape from shading, we too have left out what turns out to be a very important detail. If you actually attempt to reconstruct the viewed surface from the normals obtained by iterating (9.24) and (9.25), you will soon find out that the solution is *inconsistent*! This is hardly surprising: since the functional was not told that p and q were the partial derivatives of the same function Z, there is no Z such that $Z_x = p$ and $Z_y = q$. To circumvent this inconvenience, a good idea is to insert a step enforcing integrability after each iteration. It is more complicated to explain the reason why it works than doing it, so we first tell you how it works and then discuss why.

At each iteration, we compute the Fast Fourier Transform (FFT) of p and q. In complex notation,[7] with i as the imaginary unit, we can write

[7] If you are not familiar with the complex notation for the FFT, you may just skip the derivation and go all the way to the description of the algorithm. Of course, you need at least to be able to use an FFT routine!

$$p = \sum c_p(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)}$$

and

$$q = \sum c_q(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)},$$

where the sums range over all possible values of ω_x and ω_y (multiple of a fundamental frequency) and the c_p and c_q are the Fourier coefficients. Then let

$$Z = \sum c(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)}, \tag{9.26}$$

with

$$c(\omega_x, \omega_y) = \frac{-i\omega_x c_p(\omega_x, \omega_y) - i\omega_y c_q(\omega_x, \omega_y)}{\omega_x^2 + \omega_y^2}. \tag{9.27}$$

The function Z in (9.26) has three important properties.

- It provides a solution to the problem of reconstructing a surface from a set of nonintegrable p and q.
- Since the coefficients $c(\omega_x, \omega_y)$ do not depend on x and y, (9.26) can be easily differentiated with respect to x and y to give a new set of *integrable* p and q, say p' and q', such that

$$p' = \frac{\partial Z}{\partial x} = \sum (i\omega_x c(\omega_x, \omega_y)) e^{i(\omega_x x + \omega_y y)} = \sum c'_p(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)} \tag{9.28}$$

and

$$q' = \frac{\partial Z}{\partial y} = \sum (i\omega_y c(\omega_x, \omega_y)) e^{i(\omega_x x + \omega_y y)} = \sum c'_q(\omega_x, \omega_y) e^{i(\omega_x x + \omega_y y)}. \tag{9.29}$$

- Most importantly, p' and q' are the integrable pair *closest* to the old pair of p and q.

In more technical terms, we have projected the old p and q onto a set which contains only integrable pairs. It is a projection in a mathematical as well as intuitive sense, because if you do it twice (or more times) you invariably get the result of the first time. This can easily be seen by plugging the coefficients c'_p and c'_q of (9.28) and (9.29) in (9.27).

We now summarize the procedure we have discussed:

Algorithm SHAPE_FROM_SHADING

The input is formed by an image of an unknown surface, Z, the reflectance map of the surface, the surface's albedo, and the direction and intensity of the illuminant. The same assumptions of section 9.2 hold. Moreover, the surface slopes, p and q, (initialized to 0) are assumed to wrap around the image boundaries (cyclic boundary conditions).

Let ρ be the effective albedo and $\mathbf{i}$ the illuminant direction.

Until a suitable stopping criterion is met, iterate the following two steps.

1. Update p and q through (9.22) and (9.23).
2. Compute the FFT of the updated p and q, estimate Z according to (9.26), and p' and q' according to (9.28) and (9.29). Set $p = p'$ and $q = q'$.

The output is formed by the estimate of Z, p and q.

☞ At each iteration, make sure that the reflectance map is positive at each pixel; set negative values to 0. The parameter λ is usually set to a rather large value, for instance 1000.

☞ Notice that the step enforcing integrability can also be employed as a stand-alone procedure for reconstructing a surface Z from nonintegrable pairs of slopes p and q over a pixel grid.

9.4.6 Some Necessary Details

What can be said about the convergence properties of SHAPE_FROM_SHADING, the optimal λ and the stopping conditions?

Convergence. Unfortunately, not much can be said in general about convergence, except that it depends on how far the initial p and q are from the true values, and on the precision with which the reflectance map and the illuminant are known.

The optimal λ. Something more can be said on λ. The algorithm tends to converge faster for smaller λ (as it can be realized by looking at the right-hand-side of (9.24) and (9.25)). Not surprisingly, though, if λ becomes too small (typically below a few hundreds), the algorithm becomes unstable. For better (and more sophisticated) ways to speed up the iterative step of SHAPE_FROM_SHADING, look into the Further Readings. Instead, if λ is too large, the algorithm tends to promote very regular solutions and can even walk away from the "correct" solution . . . having started from it!

Stopping Condition. It is not easy to give a stopping condition valid in all cases. Looking at the residual,

$$\int dxdy(E - R)^2,$$

often helps but is not always appropriate, and the same can be said for the functional $\mathcal{E}$ itself. The problem is, the history of values of a residual function is not always a faithful indication of what is going on in the minimization process. It may happen that the residual appears stuck while the solution is actually getting closer to the desired minimum. It is also not unusual that the residual does not change appreciably for many iterations and then starts a relatively rapid descent toward the minimum value.

We leave you with the somewhat uncomfortable feeling that implementing a shape-from-shading algorithm means dealing with a number of open issues. Ultimately, you must get acquainted with the particular problem at hand, and develop your own ideas about it.

We conclude this part of the chapter with an example of SHAPE_FROM_SHADING when run on the data of Figure 9.2 (a). Figure 9.4 displays (from left to right) the surface reconstructed after 100 iterations, 1000 iterations, and 2000 iterations.

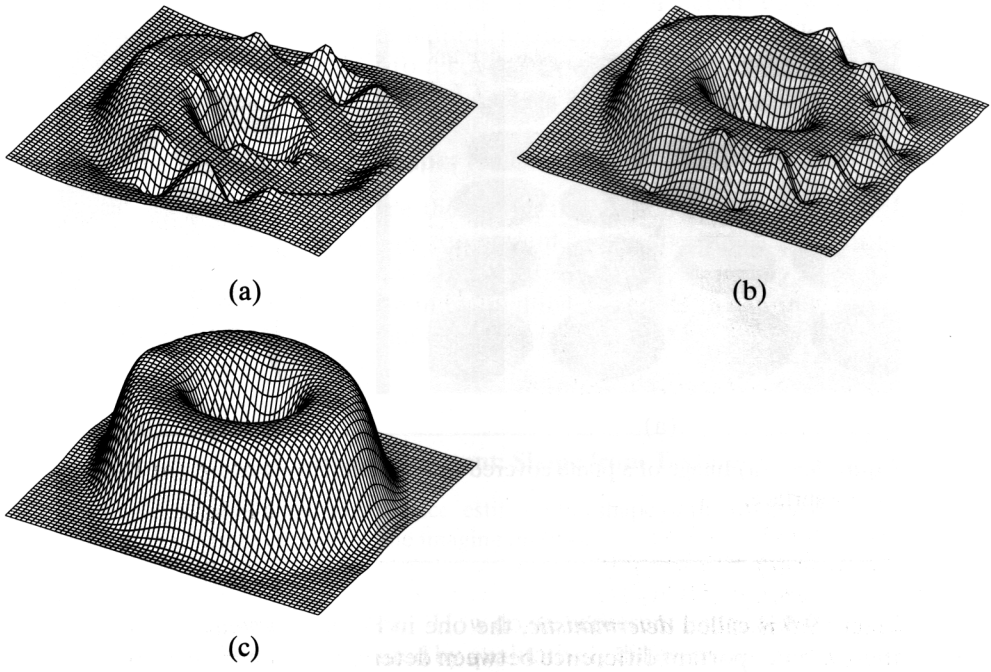


Figure 9.4 Reconstructions of the surface in Figure 9.2 after 100 (a), 1000 (b) and 2000 (c) iterations. The initial surface was a plane of constant height. The asymmetry of the first two reconstruction is due to the illuminant direction.

9.5 Shape from Texture

We now move on to the second theme of this chapter, *shape from texture*. First of all, we must specify what we mean by texture.

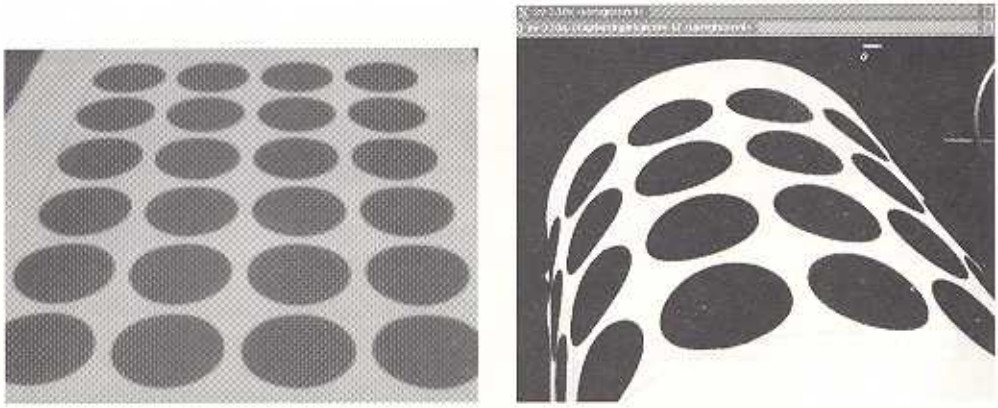
9.5.1 What is Texture?

Definition: Texture, Texels

A *surface texture* is created by the regular repetition of an element or pattern, called *surface texel*, on a surface.

An *image texture* is the image of a surface texture, itself a repetition of *image texels*, the shape of which is distorted by the projection across the image.

Figures 9.5 and 9.6 illustrate our definition. Figure 9.5 shows images of a regular grid of circles covering a plane (a) and a cylinder (b). Notice the distortion of the ellipses (image texels), projections of the circles in the scene (surface texels), across the image; we shall comment on this important feature soon. Figure 9.6 shows textures of natural surfaces: from left to right, wood sticks and small leaves, rock, sand. The texture in



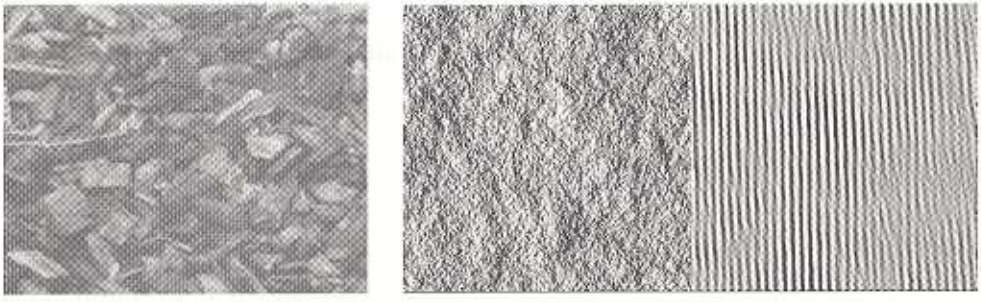
(a) (b)

Figure 9.5 (a) Image of a plane covered by a deterministic texture. (b) The same texture on a curved surface.

Figure 9.5 is called *deterministic*, the one in Figure 9.6 *statistic*. As we shall see soon, there is an important difference between deterministic and statistic textures in practice.

Definition: Deterministic and Statistic Textures

Deterministic textures are created by the repetition of a fixed geometric shape such as a circle, a square, a decorative motif.
Statistic textures are created by changing patterns with fixed statistical properties.



(a) (b) (c)

Figure 9.6 Three natural textures. (a) Wood sticks and small leaves. (b) The surface of a rock. (c) Linear patterns on sand.

Examples of deterministic textures are images of patterned wallpaper, bricks walls, and decorative tiles. Most natural textures are statistic: think for example of pebbles, gravel, wood, or lawns. To recover shape from both types of texture we need to learn a few, basic facts and make some choices.

9.5.2 Using Texture to Infer Shape: Fundamentals

Why Does it Work? Notice the strong impression of shape you get from Figures 9.5, in which texture is the only cue present.[8] How does this happen? In the image of a textured 3-D surface, *the texels appear distorted*. This is the key fact behind shape from texture: the distortion of the individual texels and its variation across the image create the 3-D impression.[9] We can therefore formulate the computational problem of shape-from-texture as follows.

Problem Statement: Shape from Texture

Given a single image of a textured surface, estimate the shape of the observed surface from the distortion of the texture created by the imaging process.

Representing Image Texture. How do we represent image texels? Deterministic and statistic textures are represented by qualitatively different techniques, and methods vary accordingly.

Deterministic texels are represented naturally by the *shape parameters* of the specific shape at hand. For instance, the ellipses in Figure 9.5 can be represented by the parameters of the ellipse equation (Chapter 5). Statistic textures are represented typically in terms of *spatial frequency properties*; for instance, by the power spectrum computed over image regions.

Representing Texture Distortion. What kind of texture distortion does the imaging process introduce? Considering Figure 9.5, we notice two distortions:

perspective distortion: due to the perspective projection, which makes circles increasingly far from the camera project to smaller and smaller ellipses;

foreshortening: which makes circles not parallel to the image plane appear as ellipses.

Under suitable assumptions, the amount of both distortions can be measured from an image. For example, in Figure 9.5 (a) perspective distortion can be quantified by the area variation across the ellipses, and foreshortening by the ratio of the ellipse's semiaxes. In fact, we can use two different classes of texture-based measures to estimate shape:

---

[8] Texture can be also a powerful mean to segment images: In Figure 9.6, although intensity varies with no apparent regularity, we have a strong impression of a uniform texture in each image, and it is perfectly obvious that the three images represent different surface types.

[9] Indeed, both are important cues for shape perception in human vision.

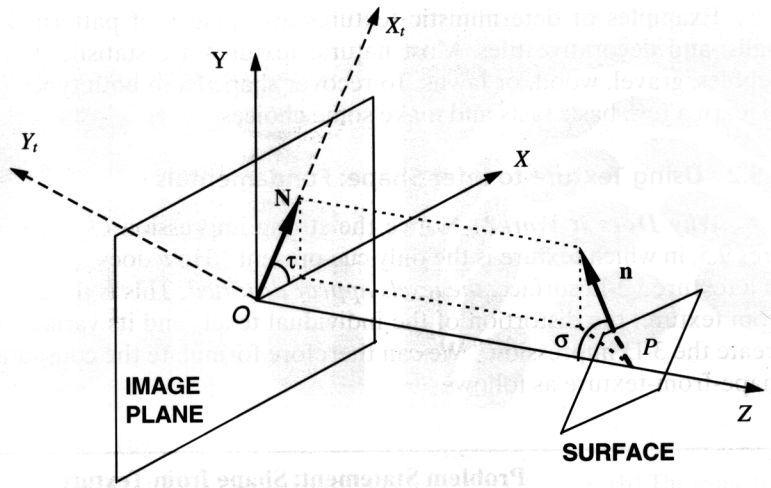


Figure 9.7 The angles tilt, τ, and slant, σ, defining a unit normal $\mathbf{n}$. Notice that this figure does *not* illustrate a projection, which is why the origin of the reference frames is on the image plane, and that these angles are *different* from those used in shape from shading (Figure 9.3).

1. a measure of shape distortion (applicable to individual image texels)
2. the rate of change of a measure of shape distortion, called *texture gradient* or *distortion gradient* for that measure (applicable to regions containing several image texels)

Representing Surface Shape. In general, the shape of a surface at any point is completely identified by the surface's orientation (i.e., the normal) and curvatures (Appendix, section A.5). However, it turns out that estimating curvatures from texture is far from trivial. We shall therefore concentrate on recovering just surface normals, as we did in section 9.4. As we know from our discussion of shape from shading, a map of normals specifies the surface's orientation only at the points at which the normals are computed (e.g., the center of deterministic texels), but, assuming that the normals are dense enough and the surface is smooth, the map can be integrated to recover surface shape (for example, as explained in section 9.4.5).

Representing Normals. In the literature of shape from texture, it is common to represent unit normals by the tilt and slant angles illustrated in Figure 9.7.

☞ Notice that these angles are defined in a different way from their shape-from-shading namesakes (Figure 9.3).

Let $\mathbf{n}$ be the normal to the surface at P. The surface is locally approximated by its tangent plane, perpendicular to $\mathbf{n}$. The tilt, τ, is the angle taking the X axis of the camera frame, XYZ, on the projection of the normal on the image plane, $\mathbf{N}$. The slant, σ, is the

angle taking $\mathbf{n}$ onto the $-Z$ axis of the camera frame. Rotating the frame $OXYZ$ around Z by τ generates the new reference frame OX_tY_tZ; in this frame, the normal $\mathbf{n}$ lies in the X_tZ plane, so that the tangent plane is parallel to Y_t.

The General Structure of Shape-from-Texture Algorithms. We have now all the elements necessary to state the general structure of shape-from-texture methods.

1. Select a representation adequate for the image texture at hand.
2. Compute the chosen distortion measures (if required, their gradients) from the image, in terms of the representation selected.
3. Use local distortion (if required, texture gradients) to estimate the local orientation of the surface.

The next section gives a simple shape-from-texture algorithm which estimates the orientation of a plane from a statistic texture.[10]

9.5.3 Surface Orientation from Statistic Texture

As usual, we begin by stating a set of assumptions.

Assumptions

1. The 3-D texels are small line segments, called *needles*.
2. The needles are distributed uniformly on the 3-D surface, and their directions are all independent.
3. The surface is approximately planar.
4. The image projection is orthographic.

We characterize the needles by their orientation only: their positions and lengths are irrelevant. The reason for choosing such apparently odd texels is that they allow us to establish a deterministic, geometric relation between the orientation of a 3-D texel and the one of its corresponding image texel. The idea is to use this relation to write the probability, say p, of the observed image given an orientation (σ, τ) of the 3-D plane, which allows us to estimate the orientation of the plane as the pair $(\hat{\sigma}, \hat{\tau})$ maximizing p (a *maximum likelihood* approach, see Appendix, section A.7). In actual fact, an approximate solution $(\hat{\sigma}, \hat{\tau})$ can be derived in closed form, and this is what we shall use in STAT_SHAPE_FROM_TEXT.

☞ Image needles can be extracted from images of 3-D textures not necessarily made up by small line segments. To do this, you run an edge detector (Chapter 4) with an adequately small kernel to extract short contours, followed by a module detecting and describing

[10] An algorithm recovering the orientation of circles in space is given in Chapter 10.

small, rectilinear segments, e.g., a variation of the Hough line detector of Chapter 5. Remember though, if the 3-D texture is not composed of small line segments, the conditions of assumptions 1 and 2 are only approximated, and this is likely to worsen results.

We now sketch the derivation behind STAT_SHAPE_FROM_TEXT, omitting much detail. Assume there are N needles in the image, and let α_i be the angle formed by the i-th image needle with the image's x axis. In our assumptions, α_i is a uniformly distributed random variable in $[0, \pi]$. We now introduce an auxiliary vector, $[\cos 2\alpha_i, \sin 2\alpha_i]^\top$, itself a random quantity. This vector is characterized by a probability distribution called *distribution on the unit circle*, which is a function of the distribution of the α_i. Its *center of mass* is defined by

$$C = \frac{1}{N}\sum_{i=1}^{N} \cos 2\alpha_i$$

$$S = \frac{1}{N}\sum_{i=1}^{N} \sin 2\alpha_i. \tag{9.30}$$

It can be proven that, in orthographic projections (Assumption 4), the center of mass is

$$C = \cos 2\tau \frac{1-\cos\sigma}{1+\cos\sigma}$$

$$S = \sin 2\tau \frac{1-\cos\sigma}{1+\cos\sigma}. \tag{9.31}$$

Solving for σ and τ, we find

$$\sigma = \arccos\frac{1-Q}{1+Q}$$

$$\tau = \psi \pm \frac{\pi}{2} \pmod{2\pi}, \tag{9.32}$$

where Q and ψ are the polar coordinates of the center of mass,

$$Q = \sqrt{C^2+S^2}, \qquad \psi = \frac{1}{2}\arctan\frac{S}{C}. \tag{9.33}$$

☞ Notice the ambiguity in the estimate of tilt.

The complete algorithm is stated below:

Algorithm STAT_SHAPE_FROM_TEXTURE

The input is an image containing N needles, each forming an angle α_i with the x axis of the image. The assumptions stated above hold.

1. Compute C, S using (9.30).

2. Compute the polar coordinates Q, ψ of the center of mass of the distribution on the unit circle using (9.33).
3. Estimate the orientation of the 3-D plane, $(\hat{\sigma}, \hat{\tau})$, using (9.32).

The output is $(\hat{\sigma}, \hat{\tau})$, the estimate of the orientation of the 3-D plane.

9.5.4 Concluding Remarks

Shape from Texture and Texture Segmentation. In our discussion of shape from texture, we assumed a uniform texture throughout the image. In reality, this is something of a special case: Images are likely to contain different textures, or textured areas surrounded by non-textured ones. In general, differently textured regions need separating before shape-from-texture algorithms can be applied. This problem is called *texture segmentation*, and it is a classic problem of image processing. As texture is an ubiquitous feature of surfaces, texture segmentation is frequently used to identify objects of interest in the image; for instance, it proves very useful in many defect detection systems.

In texture segmentation, image pixels are classified on the basis of several textural measures, called *texture features*, computed in local neighborhoods. These measures are usually statistical properties of the intensity values, or spatial frequency measures like the power spectrum. Unfortunately, segmentation relies on the assumption that texture features are constant within regions of uniform texture, whereas texture (and feature) distortions are exactly what shape-from-texture methods rely on! For this reason, performing texture segmentation and shape from texture at the same time is not trivial at all, which is why we have treated shape from texture as independent of texture segmentation. The Further Readings point you to an introduction to the literature of texture segmentation.[11]

Texture Depends on Spatial Scale. Textures appear and disappears at different spatial scales. For example, imagine to zoom in on a floor made of wooden planks: When the image contains many planks, the main image texture is given by the planks' contours; as you close in on one plank, the main texture is given by the wood's fibers. These textures look different: The wooden planks create a deterministic pattern, the fibers a statistical one. Therefore, "the texture of a surface" actually refers to the texture of a surface *at a given spatial scale.*

9.6 Summary

After working through this chapter you should be able to:

- ❑ explain the nature and objectives of shape-from-X methods
- ❑ explain the purpose and nature of shape from shading

[11] The reason why texture segmentation has not been included in our discussion of feature detection (Chapters 4 and 5) is exactly that it was not needed to support shape from texture.

- ❑ design an algorithm for shape from shading, and recover a surface from a map of normals
- ❑ explain the purpose and nature of shape from texture
- ❑ recover the orientation of a plane covered by a statistical texture

9.7 Further Readings

Table 9.1 mentions several shape-from-X methods; here are some introductory references, chosen from a vast literature. You can begin an investigation into *shape from focus and defocus* from [17, 21]. Ma and Olsen's work [19] is a good starting point for *shape from zoom*. A good illustration of *shape from contours* is given by [16], and a modern shape-from-contour approach is discussed by Cipolla and Blake [5]. *Active vision* is a recent, influential paradigm of computer vision; for an introduction, see [3], or the report of the recent panel discussion in [24], or again the seminal paper by Aloimonos *et al.* [1].

The approximate method for determining albedo and illuminant has been adapted from [26], which also explains in detail how to obtain (9.16). The field of shape from shading has been strongly influenced by a number of pioneering works due to Horn and coworkers [11, 14, 12]. More on the reflectance map can be found in [13]. The original method proposed by Horn more than twenty years ago in [11] is still a classic, though not easy to explain and implement. The algorithm described in this chapter has been proposed by Frankot and Chellappa [6] as an improvement of the method described in [12]. A rigorous account on ill-posed problem of computer vision and methods for their solution can be found in [2].

Much of our discussion of shape from texture is based on Gårding's work [7, 8] and references therein, in which you can also find a detailed treatment of texture-based curvature estimation. Methods for recovering shape from deterministic textures under perspective projections are discussed in [4, 15, 9]. The closed-form solution in STAT_SHAPE_FROM_TEXTURE is due to Gårding [8], and is a variation of Witkin's maximum-likelihood estimator [25] which involves a nontrivial minimization. Recent examples of shape-from-texture algorithms reconstructing curved surfaces covered by statistical textures, using spatial-frequency descriptors, are given in [23] and [20]. Approaches to the problem of segmenting texture and computing shape from texture simultaneously exists, but the methods are not trivial; an example is reported in [18]. Texture segmentation in itself is a classic topic of image processing, and its literature is vast. To begin its exploration, see [10] for a collection of texture-based segmentation algorithms, and [22] for a recent survey of the field.

9.8 Review

Questions

- ❒ *9.1* Why the methods in this chapter recover shape but not distance?
- ❒ *9.2* Why the methods in this chapter are applied to intensity images, and not to range images?

- ❒ *9.3* Explain the difference between the tilt and slant used in shape from shading and shape from texture.
- ❒ *9.4* Are the tilt and slant angles used in shape from shading and shape from texture the same as spherical coordinates? If not, what are the differences?
- ❒ *9.5* Explain why, in algorithm APPRX_ALBEDO_ILLUM_FINDER, it could happen that $\cos\sigma > 1$. What would you do to avoid this inconsistency?
- ❒ *9.6* Identify the ambiguity intrinsic to the reflectance map of a Lambertian surface. Discuss the consequences on shape from shading.
- ❒ *9.7* Why are the boundary conditions necessary in SHAPE_FROM_SHADING? Could you run the algorithm ignoring them?
- ❒ *9.8* How would you set a value for λ in SHAPE_FROM_SHADING? (*Hint:* Look at the right-hand side of (9.24) and (9.25).)
- ❒ *9.9* Explain the difference between foreshortening and perspective distortion.
- ❒ *9.10* Can you assume orthographic projections when using the area gradient? Why?
- ❒ *9.11* Identify the elements given in the general structure of shape-from-texture algorithms in STAT_SHAPE_FROM_TEXT.
- ❒ *9.12* How would you choose the parameters of an edge detector used to extract needles for STAT_SHAPE_FROM_TEXT?

Exercises

- ○ *9.1* Compute the integral

$$< E^2 >= \int_0^{2\pi} d\alpha \int_0^{\pi/2} d\beta \mathcal{P}(\alpha, \beta) E^2(\alpha, \beta)$$

with $E(\alpha, \beta)$ as in (9.11). *Hint:* Recall that

$$\int_{-\pi}^{\pi} \sin^2 x dx = \int_{-\pi}^{\pi} \cos^2 x dx = \pi.$$

- ○ *9.2* Show that the Fourier coefficients c'_p and c'_q of (9.28) and (9.29) leave the Fourier coefficients c of (9.26) unchanged.
- ○ *9.3* Explain why replacing the term multiplying λ in (9.17) with $(p_y - q_x)^2$ is an alternative way of enforcing integrability in shape from shading. Do you think this way of enforcing integrability is as effective as Step 2 in SHAPE_FROM_SHADING? Why does the new functional also enforce smoothness?
- ○ *9.4* Derive and discretize the Euler-Lagrange equations for the new functional of Exercise 9.3. Find an iterative scheme similar to the one in SHAPE_FROM_SHADING.
- ○ *9.5* Consider the textured plane shown in Figure 9.5 (a). Write an expression linking slant, the (known) size of an image texel, and the distance of the corresponding 3-D texel from the camera. Assume there is no tilt.

○ *9.6* Extend your solution to Exercise 9.5 by removing the assumption of zero tilt. You have now to write an expression for the tilt and one for the slant.

Projects

- *9.1* Implement SHAPE_FROM_SHADING. Test the algorithm on synthetically generated images of a Lambertian surface following the suggestions given in section 9.2. Study the sensitivity of the algorithm to noise, direction of illuminant and uncertainty in the knolwdge of albedo and illuminant.
- *9.2* Implement STAT_SHAPE_FROM_TEXT and test your implementation with synthetic images of planar textures corrupted by additive noise. Study the variation of the error as a function of the amount of additive noise and of the orientation of the plane. For which orientations do you expect higher errors? Why? Is your expectation confirmed by experiments?

References

[1] Y. Aloimonos, I. Weiss and A. Bandopadhay, Active Vision, *International Journal of Computer Vision*, Vol. 7, pp. 333 – 356 (1988).

[2] M. Bertero, T. Poggio and V. Torre, Ill-posed Problems in Early Vision, *Proc. IEEE*, Vol. 76, pp. 869–889 (1988).

[3] A. Blake and A. Yuill, *Active Vision*, MIT Press, Cambridge (MA) (1992).

[4] D. Blostein and N. Ahuja, Shape from Texture: Integrating Texture-Element Extraction and Surface Estimation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-11, no. 12, pp. 1233–1251 (1989).

[5] R. Cipolla and A. Blake, Surface Shape from the Deformation of Apparent Contours, *International Journal of Computer Vision*, Vol. 9, pp. 83 – 112 (1992).

[6] R.T. Frankot and R. Chellappa, A Method for Enforcing Integrability in Shape from Shading Algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, no. 4, pp. 439 – 451 (1988).

[7] J. Gårding, Shape from Texture for Smooth Curved Surfaces, *Proc. European Conf. on Computer Vision*, S. Margherita (Italy), pp. 630–638 (1992).

[8] J. Gårding, Direct Estimation of Shape from Texture, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-15, no. 11, pp. 1202–1207 (1993).

[9] J. Gårding, Shape from Texture for Smooth Curved Surfaces in Perspective Projection, *Int. Journ. of Mathematical Imaging*, Vol. 2, no. 4, pp. 329 – 352 (1992).

[10] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Vol. I, Addison-Wesley (1992).

[11] B.K.P. Horn, Obtaining Shape from Shading Information, in P.H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 115–155 (1975).

[12] B.K.P. Horn and M.J. Brooks, The Variational Approach to Shape from Shading, *Computer Vision Graphics and Image Processing*, Vol. 33, no. 2, pp. 174 – 208 (1986).

[13] B.K.P. Horn and B.G. Sjoberg, Calculating the Reflectance Map, *Applied Optics*, Vol. 18, no. 11, pp. 1770 – 1779 (1979).

[14] K. Ikeuchi and B.K.P. Horn, Numerical Shape form Shading and Occluding Boundaries, *Artificial Intelligence*, Vol. 17, no. 3, pp. 141 – 184 (1981).

[15] K. Kanatani and T.-C. Chou, Shape from Texture: General Principle, *Artificial Intelligence*, Vol. 38, pp. 1 – 48 (1989).

[16] J.J. Koenderink, What Does the Occluding Contour Tell Us About Solid Shape? *Perception*, Vol. 13 (1984).

[17] E. Krotkow, Focusing, *International Journal of Computer Vision*, Vol. 1, pp. 223 – 237 (1987).

[18] J. Krumm and S. A. Schafer, Texture Segmentation and Shape in the Same Image, Proc. IEEE Int. Conf. on Comp. Vision, Cambridge (MA), pp. 121–127 (1995).

[19] J. Ma and S. I. Olsen, Depth from Zooming, *Journ. of the Optical Society of America A*, Vol. 7, no. 10, pp. 1883 – 1890 (1990).

[20] J. Malik and R. Rosenholtz, Recovering Surface Texture and Orientation from Texture Distortion, *Proc. European Conf. on Computer Vision*, Stockholm, pp. 353–364 (1994).

[21] A. P. Pentland, A New Sense for Depth of Field, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, no. 4, pp. 523–531 (1987).

[22] T.R. Reed and J.M. Hand du Buf, A Review of Recent Texture Segmentation and Feature Extraction Techniques, *Computer Vision Graphics and Image Processing: Image Understanding*, Vol. 57, no. 3, pp. 359–372 (1993).

[23] B.J. Super and A.C. Bovik, Shape from Texture Using Local Spectral Moments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-17, no. 4, pp. 333–343 (1995).

[24] M.J. Swain and M.A. Stricker, Promising Directions in Active Vision, *International Journal of Computer Vision*, Vol. 11, no. 2, pp. 109 – 126 (1993).

[25] A.P. Witkin, Recovering Shape and Orientation from Texture, *Artificial Intelligence*, Vol. 17, pp. 17–45 (1981).

[26] Q. Zheng and R. Chellappa, Estimation of Illuminant Direction, Albedo, and Shape from Shading, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-13, no. 7, pp. 680 – 702 (1991).