Trucco + Verri, "Introductory Techniques for 3-D Computer Vision," Prentice Hall 1998

8

Motion

Eppur si muove.[1]
Galileo

This chapter concerns the analysis of the *visual motion* observed in time-varying image sequences.

Chapter Overview

Section 8.1 presents the basic concepts, importance and problems of visual motion.

Section 8.2 introduces the notions of *motion field* and *motion parallax*, and their fundamental equations.

Section 8.3 discusses the *image brightness constancy equation* and the *optical flow*, the approximation of the motion field which can be computed from the changing image brightness pattern.

Section 8.4 presents methods for estimating the motion field, divided in *differential* and *feature-matching/tracking* methods.

Section 8.5 deals with the *reconstruction of 3-D motion and structure*.

Section 8.6 discusses *motion-based segmentation* based on change detection.

What You Need to Know to Understand this Chapter

- Working knowledge of Chapters 2 and 7.
- Eigenvalues and eigenvectors of a matrix.
- Least squares and SVD (Appendix, section A.6).
- The basics of Kalman filtering (Appendix, section A.8).

[1] And yet it is moving.

8.1 Introduction

Until now, we have studied visual computations on single images, or two images acquired simultaneously. In this chapter, we broaden our perspective and focus on the processing of images *over time*. More precisely, we are interested in the visual information that can be extracted from the spatial and temporal changes occurring in an *image sequence*.

Definition: Image Sequence

An image sequence is a series of N images, or *frames*, acquired at discrete time instants $t_k = t_0 + k\Delta t$, where Δt is a fixed time interval, and $k = 0, 1, \ldots, N-1$.

In order to acquire an image sequence, you need a frame grabber capable of storing frames at a fast rate. Typical rates are the so called *frame rate* and *field rate*, corresponding to a time interval Δt of 1/24sec and 1/30sec respectively. If you are allowed to choose a different time interval, or simply want to subsample an image sequence, make sure that Δt is small enough to guarantee that the discrete sequence is a representative sampling of the continuous image evolving over time; as a rule of thumb, this means that the apparent displacements over the image plane between frames should be at most a few pixels.

Assuming the illumination conditions do not vary, image changes are caused by a *relative motion between camera and scene*: the viewing camera could move in front of a static scene, or parts of the scene could move in front of a stationary camera, or, in general, both camera and objects could be moving with different motions.

8.1.1 The Importance of Visual Motion

The temporal dimension in visual processing is important primarily for two reasons. First, the apparent motion of objects onto the image plane is a strong visual cue for understanding structure and 3-D motion. Second, biological visual systems use visual motion to infer properties of the 3-D world with little *a priori* knowledge of it. Two simple examples may be useful to illustrate these points.

Example 1: Random Dot Sequences. Consider an image of *random dots*, generated by assigning to each pixel a random grey level. Consider a second image obtained by shifting a squared, central region of the first image by a few pixels, say, to the right, and filling the gap thus created with more random dots. Two such images are shown in Figure 8.1. If you display the two images in sequence on a computer screen, in the same window and one after the other at a sufficiently fast rate, you will unmistakably *see* a square moving sideways back and forth against a steady background. Notice that the visual system bases its judgement on the only information available in the sequence; that is, the displacement of the square in the two images.[2]

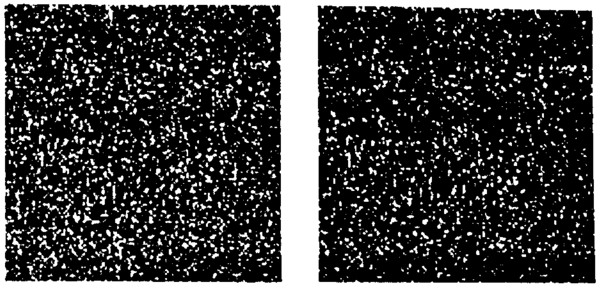


Figure 8.1 A sequence of two random dot images: a square has been displaced between the two frames.

Example 2: Computing Time-to-Impact. Visual motion allows us to compute useful properties of the observed 3-D world with very little knowledge about it. Consider a planar version of the usual pinhole camera model, and a vertical bar perpendicular to the optical axis, travelling towards the camera with constant velocity as shown in Figure 8.2. We want to prove a simple but very important fact: *It is possible to compute the time, τ, taken by the bar to reach the camera only from image information;* that is, without knowing either the real size of the bar or its velocity in 3-D space.[3]

As shown in Figure 8.2, we denote with L the real size of the bar, with V its constant velocity, and with f the focal length of the camera. The origin of the reference frame is the projection center. If the position of the bar on the optical axis is $D(0) = D_0$ at time $t = 0$, its position at a later time t will be $D = D_0 - Vt$. Note that L, V, f, D_0, and the choice of the time origin are all unknown, but that τ can be written as

$$\tau = \frac{V}{D}. \tag{8.1}$$

From Figure 8.2, we see that $l(t)$, the *apparent* size of the bar at time t on the image plane, is given by

$$l(t) = f\frac{L}{D}.$$

[2] Incidentally, you can look at the two images of Figure 8.1 as a *random-dot stereogram* to perceive a square floating in the background. Stand a diskette (or a sheet of paper of the same size) between the two images and touch your nose against the diskette, so that each eye can see only one image. Focus your eyes *behind* the page. After a while, the two images should fuse and produce the impression of a square floating against the background.

[3] In the biologically-oriented community of computer vision, τ is called, rather pessimistically, *time-to-collision* or even *time-to-crash*!

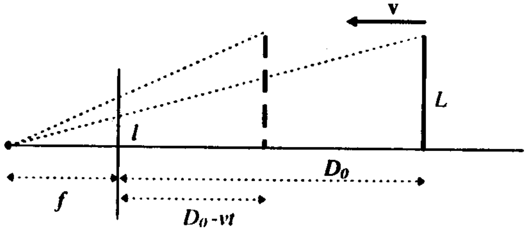


Figure 8.2 How long before the bar reaches the camera?

If we now compute the time derivative of $l(t)$,

$$l'(t) = \frac{dl(t)}{dt} = -f\frac{L}{D^2}\frac{dD}{dt} = f\frac{LV}{D^2},$$

take the ratio between $l(t)$ and $l'(t)$, and use (8.1), we obtain

$$\frac{l(t)}{l'(t)} = \tau. \qquad (8.2)$$

This is the equation we were after: since both the apparent size of the bar, $l(t)$, and its time derivative, $l'(t)$, are measured from the images, (8.2) allows us to compute τ in the absence of *any* 3-D information, like the size of the bar and its velocity.

8.1.2 The Problems of Motion Analysis

It is now time to state the main problems of motion analysis. The analogies with stereo suggest to begin by dividing the motion problem into two subproblems.

Two Subproblems of Motion

Correspondence: Which elements of a frame correspond to which elements of the next frame of the sequence?

Reconstruction: Given a number of corresponding elements, and possibly knowledge of the camera's intrinsic parameters, what can we say about the 3-D motion and structure of the observed world?

Main Differences between Motion and Stereo

Correspondence: As image sequences are sampled temporally at usually high rates, the spatial differences (disparities) between consecutive frames are, on average, much smaller than those of typical stereo pairs.

Reconstruction: Unlike stereo, in motion the relative 3-D displacement between the viewing camera and the scene is not necessarily caused by a single 3-D rigid transformation.

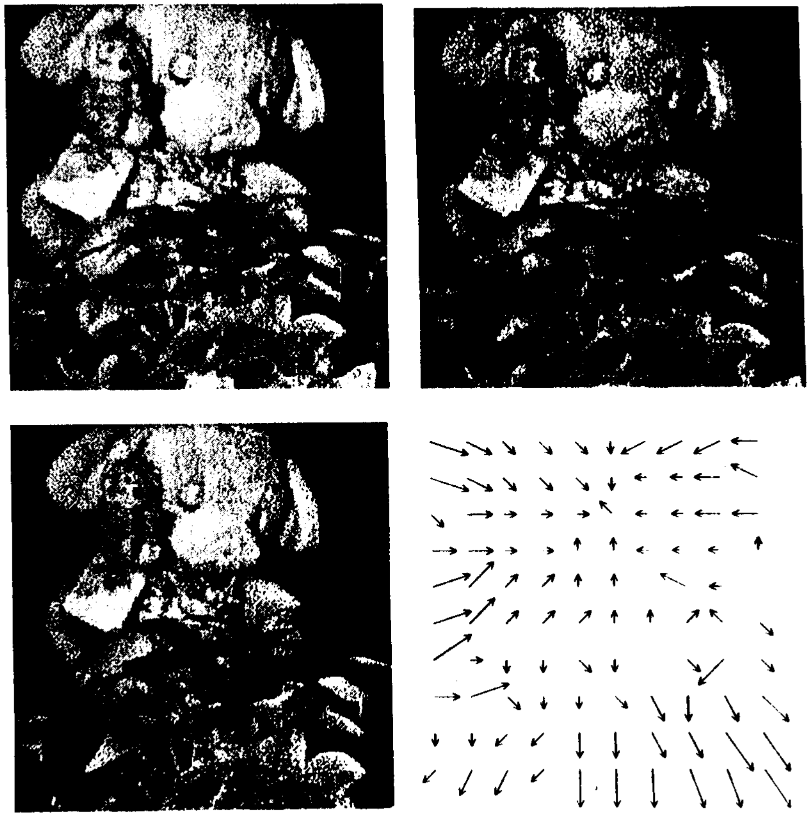


Figure 8.3 Three frames from a long image sequence (left to right and top to bottom) and the optical flow computed from the sequence, showing that the plant in the foreground is moving towards the camera, and the soft toys away from it.

Regarding correspondence, the fact that motion sequences make many, closely sampled frames available for analysis is an advantage over the stereo case for at least two reasons. First, feature-based approaches can be made more effective by *tracking* techniques, which exploit the past history of the features' motion to predict disparities in the next frame. Second, due to the generally small spatial and temporal differences between consecutive frames, the correspondence problem can also be cast as the problem of *estimating the apparent motion of the image brightness pattern*, usually called *optical flow* (see Figure 8.3).

We shall use two strategies for solving the correspondence problem.

Differential methods (section 8.4.1) lead to *dense* measures; that is, computed at each image pixel. They use estimates of time derivatives, and require therefore image sequences sampled closely.

Matching methods (section 8.4.2) lead to *sparse* measures; that is, computed only at a subset of image points. We shall place emphasis on *Kalman filtering* as a technique for matching and tracking efficiently sparse image features over time.

Unlike correspondence, and perhaps not surprisingly, reconstruction is more difficult in motion than in stereo. Even in the presence of only one 3-D motion between the viewing camera and the scene, frame-by-frame recovery of motion and structure turns out to be more sensitive to noise. The reason is that the baseline between consecutive frames, regarded as a stereo pair, is very small (see Chapter 7). 3-D motion and structure estimation from both sparse and dense estimates of the image motion is discussed in sections 8.5.1 and 8.5.2, respectively.

This chapter discusses and motivates methods for solving correspondence and reconstruction under the following simplifying assumption.

Assumption

There is only one, rigid, relative motion between the camera and the observed scene, and the illumination conditions do not change.

This assumption of single, rigid motion implies that *the 3-D objects observed cannot move of different motions*. This assumption is violated, for example, by sequences of football matches, motorway traffic or busy streets, but satisfied by, say, the sequence of a building viewed by a moving observer. The assumption also rules out flexible (nonrigid) objects: deformable objects like clothes or moving human bodies are excluded.

If the camera is looking at more than one moving object, or you simply cannot assume a moving camera in a static environment, a third subproblem must be added.

The Third Subproblem of Motion

The segmentation problem: What are the regions of the image plane which correspond to different moving objects?

The main difficulty here is a chicken and egg problem: should we first solve the matching problem and then determine the regions corresponding to the different moving objects, or find the regions first, and then look for correspondences? This question is addressed in section 8.6 in the hypothesis that the viewing camera is not moving. Pointers to solutions to this difficult problem in more general cases are given in the Further Readings.

We now begin by establishing some basic facts.

8.2 The Motion Field of Rigid Objects

Definition: Motion Field

The motion field is the 2-D vector field of velocities of the image points, induced by the relative motion between the viewing camera and the observed scene.

The motion field can be thought of as *the projection of the 3-D velocity field on the image plane* (to visualize this vector field, imagine to project the 3-D velocity vectors on the image). The purpose of this section is to get acquainted with the theory and geometrical properties of the motion field. *We shall work in the camera reference frame*, ignoring the image reference frame and the pixelization.[4] The issue of camera calibration will be raised in due time.

This section presents some essential facts of motion fields, compares disparity representations in motion and stereo, analyzes two special cases of rigid motion leading to generally useful facts, and introduces the concept of motion parallax.

8.2.1 Basics

Notation. We let $\mathbf{P} = [X, Y, Z]^\top$ be a 3-D point in the usual camera reference frame: The projection center is in the origin, the optical axis is the Z axis, and f denotes the focal length. The image of a scene point, $\mathbf{P}$, is the point $\mathbf{p}$ given by

$$\mathbf{p} = f\frac{\mathbf{P}}{Z}. \tag{8.3}$$

As usual (see Chapter 2), since the third coordinate of $\mathbf{p}$ is always equal to f, we write $\mathbf{p} = [x, y]^\top$ instead of $\mathbf{p} = [x, y, f]^\top$. The relative motion between $\mathbf{P}$ and the camera can be described as

$$\mathbf{V} = -\mathbf{T} - \boldsymbol{\omega} \times \mathbf{P}, \tag{8.4}$$

where $\mathbf{T}$ is the translational component of the motion,[5] and $\boldsymbol{\omega}$ the angular velocity. As the motion is rigid, $\mathbf{T}$ and $\boldsymbol{\omega}$ are the same for any $\mathbf{P}$. In components, (8.4) reads

$$\begin{aligned} V_x &= -T_x - \omega_y Z + \omega_z Y \\ V_y &= -T_y - \omega_z X + \omega_x Z \\ V_z &= -T_z - \omega_x Y + \omega_y X. \end{aligned} \tag{8.5}$$

[4] Remember, this means that we consider the intrinsic parameters known.

[5] Note that $\mathbf{T}$ denotes a velocity vector only in this chapter, not a displacement vector as in the rest of the book.

The Basic Equations of the Motion Field. To obtain the relation between the velocity of **P** in space and the corresponding velocity of **p** on the image plane, we take the time derivative of both sides of (8.3), which gives an important set of equations.

The Basic Equations of the Motion Field

'Image velocity' eqns.

The motion field, **v**, is given by

$$\mathbf{v} = f\frac{Z\mathbf{V} - V_z\mathbf{P}}{Z^2}. \tag{8.6}$$

In components, and using (8.5), (8.6) read

$$\begin{aligned} v_x &= \frac{T_z x - T_x f}{Z} - \omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f} \\ v_y &= \frac{T_z y - T_y f}{Z} + \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f}. \end{aligned} \tag{8.7}$$

Notice that *the motion field is the sum of two components, one of which depends on translation only, the other on rotation only*. In particular, the translational components of the motion field are

$$\begin{aligned} v_x^T &= \frac{T_z x - T_x f}{Z} \\ v_y^T &= \frac{T_z y - T_y f}{Z}, \end{aligned}$$

and the rotational components are

$$\begin{aligned} v_x^\omega &= -\omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f} \\ v_y^\omega &= \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f}. \end{aligned}$$

Since the component of the motion field along the optical axis is always equal to 0, we shall write $\mathbf{v} = [v_x, v_y]^\top$ instead of $\mathbf{v} = [v_x, v_y, 0]^\top$. Notice that, in the last two pairs of equations, the terms depending on the angular velocity, ω, and depth, Z, are decoupled. This discloses an important property of the motion field: *the part of the motion field that depends on angular velocity does not carry information on depth*.

Comparing Disparity Representations in Stereo and Motion. As we said before, stereo and motion pose similar computation problems, and one of these is correspondence. Point displacements are represented by disparity maps in stereo, and by motion fields in motion. An obvious question is, how similar are disparity maps and motion fields? The key difference is that *the motion field is a differential concept, stereo disparity is not*. The motion field is based on velocity, and therefore on time derivatives: Consecutive frames must be as close as possible to guarantee good discrete approximations of the continuous time derivatives. In stereo, there is no such constraint on the two images, and the disparities can take, in principle, any value.

Stereo Disparity Map and Motion Field

The spatial displacements of corresponding points between the images of a stereo pair (forming the stereo disparity map) are *finite*, and, in principle, unconstrained.

The spatial displacements of corresponding points between consecutive frames of a motion sequence (forming the motion field) are discrete approximations of time-varying derivatives, and must therefore be suitably small.

The motion field coincides with the stereo disparity map *only if* spatial and temporal differences between frames are sufficiently small.

8.2.2 Special Case 1: Pure Translation

We now analyze the case in which the relative motion between the viewing camera and the scene has no rotational component. The resulting motion field has a peculiar spatial structure, and its analysis leads to concepts very useful in general.

Since $\boldsymbol{\omega} = 0$, (8.7) read

$$\begin{aligned} v_x &= \frac{T_z x - T_x f}{Z} \\ v_y &= \frac{T_z y - T_y f}{Z} \end{aligned} \tag{8.8}$$

We first consider the general case in which $T_z \neq 0$. Introducing a point $\mathbf{p}_0 = [x_0, y_0]^\top$ such that

$$\begin{aligned} x_0 &= fT_x/T_z \\ y_0 &= fT_y/T_z, \end{aligned} \tag{8.9}$$

(8.8) become

$$\begin{aligned} v_x &= (x - x_0)\frac{T_z}{Z} \\ v_y &= (y - y_0)\frac{T_z}{Z}. \end{aligned} \tag{8.10}$$

Equation (8.10) say that *the motion field of a pure translation is radial*: It consists of vectors radiating from a common origin, the point $\mathbf{p}_0$, which is therefore the *vanishing point* of the translation direction. In particular, if $T_z < 0$, the vectors point away from $\mathbf{p}_0$, and $\mathbf{p}_0$ is called the *focus of expansion* (Figure 8.4 (a)); if $T_z > 0$, the motion field vectors point towards $\mathbf{p}_0$, and $\mathbf{p}_0$ is called the *focus of contraction* (Figure 8.4 (b)). In addition, the length of $\mathbf{v} = \mathbf{v}(\mathbf{p})$ is proportional to the distance between $\mathbf{p}$ and $\mathbf{p}_0$, and inversely proportional to the depth of the 3-D point $\mathbf{P}$.

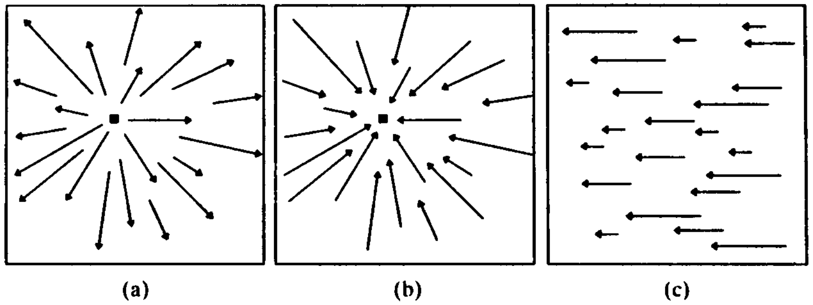


Figure 8.4 The three types of motion field generated by translational motion. The filled square marks the instantaneous epipole.

☞ Notice that the point $\mathbf{p}_0$ retains its significance and many of its properties even in the presence of a rotational component of 3-D motion (section 8.5.2).

If T_z vanishes (a rather special case), (8.8) become

$$v_x = -f\frac{T_x}{Z}$$
$$v_y = -f\frac{T_y}{Z}.$$

Therefore, *if $T_z = 0$, all the motion field vectors are parallel* (see Figure 8.4 (c)) and their lengths are inversely proportional to the depth of the corresponding 3-D points.

☞ In homogeneous coordinates, there would be no need to distinguish between the two cases $T_z \neq 0$ and $T_z = 0$: For *all* possible values of T_z, including $T_z = 0$, $\mathbf{p}_0$ is the vanishing point of the direction in 3-D space of the translation vector $\mathbf{T}$, and the 3-D line through the center of projection and $\mathbf{p}_0$ is parallel to $\mathbf{T}$.

Following is a summary of the main properties of the motion field of a purely translational motion.

Pure Translation: Properties of the Motion Field

1. If $T_z \neq 0$, the motion field is *radial* (see (8.10)), and all vectors point towards (or away from) a single point, $\mathbf{p}_0$, given by (8.8). If $T_z = 0$, the motion field is *parallel*.
2. The length of motion field vectors is inversely proportional to the depth Z; if $T_z \neq 0$, it is also inversely proportional to the distance from $\mathbf{p}$ to $\mathbf{p}_0$.
3. $\mathbf{p}_0$ is the vanishing point of the direction of translation (see (8.10)).
4. $\mathbf{p}_0$ is the intersection of the ray parallel to the translation vector with the image plane.

8.2.3 Special Case 2: Moving Plane

Planes are common surfaces in man-made objects and environments, so it is useful t investigate the properties of the motion field of a moving plane. Assume that the came is observing a planar surface, π, of equation

$$\mathbf{n}^\top \mathbf{P} = d \qquad (8.1$$

where $\mathbf{n} = [n_x, n_y, n_z]^\top$ is the unit vector normal to π, and d the distance between and the origin (the center of projection). Let π be moving in space with translation velocity $\mathbf{T}$ and angular velocity $\boldsymbol{\omega}$, so that both $\mathbf{n}$ and d in (8.11) are functions of tim By means of (8.3), (8.11) can be rewritten as

$$\frac{n_x x + n_y y + n_z f}{f} Z = d. \qquad (8.1$$

Solving for Z in (8.12), and plugging the resulting expression into (8.7), we have

$$\begin{aligned} v_x &= \frac{1}{fd}(a_1 x^2 + a_2 xy + a_3 fx + a_4 fy + a_5 f^2) \\ v_y &= \frac{1}{fd}(a_1 xy + a_2 y^2 + a_6 fy + a_7 fx + a_8 f^2) \end{aligned} \qquad (8.1$$

where

$$\begin{aligned} a_1 &= -d\omega_y + T_z n_x, & a_2 &= d\omega_x + T_z n_y, \\ a_3 &= T_z n_z - T_x n_x, & a_4 &= d\omega_z - T_x n_y, \\ a_5 &= -d\omega_y - T_x n_z, & a_6 &= T_z n_z - T_y n_y, \\ a_7 &= -d\omega_z - T_y n_x, & a_8 &= d\omega_x - T_y n_z. \end{aligned}$$

The (8.13) states, interestingly, that *the motion field of a moving planar surfac any instant t, is a quadratic polynomial in the coordinates (x, y, f) of the image poin*

The remarkable symmetry of the time-dependent coefficients $a_1 \ldots a_8$ is not incidental. You can easily verify that the a_i remain unchanged if d, $\mathbf{n}$, $\mathbf{T}$, and $\boldsymbol{\omega}$ replaced by

$$\begin{aligned} d' &= d \\ \mathbf{n}' &= \mathbf{T}/\|\mathbf{T}\| \\ \mathbf{T}' &= \|\mathbf{T}\|\mathbf{n} \\ \boldsymbol{\omega}' &= \boldsymbol{\omega} + \mathbf{n} \times \mathbf{T}/d. \end{aligned}$$

This means that, apart from the special case in which **n** and **T** are parallel, *the same motion field can be produced by two different planes undergoing two different 3-D motions.*[6]

☞ The practical consequence is that it is usually impossible to recover uniquely the 3-D structure parameters, **n** and d, and motion parameters, **T** and $\boldsymbol{\omega}$, of a planar set of points from the motion field *alone*.

You might be tempted to regard this discussion on the motion field of a planar surface as a mere mathematical curiosity. On the contrary, we can draw at least two important and general conclusions from it.

1. Since the motion field of a planar surface is described *exactly and globally* by a polynomial of second degree (see (8.13)), *the motion field of any smooth surface is likely to be approximated well by a low-order polynomial even over relatively large regions of the image plane* (Exercise 8.1). The useful consequence is that very simple parametric models enable a quite accurate estimation of the motion field in rather general circumstances (section 8.4.1).
2. As algorithms recoverying 3-D motion and structure cannot be based on motion estimates produced by coplanar points, measurements must be made at many different locations of the image plane in order to minimize the probability of looking at points that lie on planar or nearly planar surfaces.[7] We will return to this point in sections 8.5.1 and 8.5.2.

We conclude this section with a summary of the main properties of the motion field of a planar surface.

Moving Plane: Properties of Motion Field

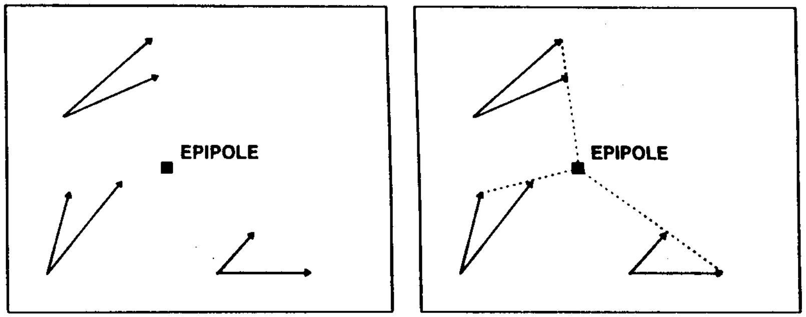
1. The motion field of a planar surface is, at any time instant, a quadratic polynomial in the image coordinates.
2. Due to the special symmetry of the polynomial coefficients, the same motion field can be produced by two different planar surfaces undergoing different 3-D motions.

8.2.4 Motion Parallax

The decoupling of rotational parameters and depth in the (8.7) is responsible for what is called *motion parallax*. Informally, motion parallax refers to the fact that *the relative motion field of two instantaneously coincident points does not depend on the rotational*

[6] This result should not surprise you. Planar surfaces lack generality: The eight-point algorithm (Chapter 7), for example, fails to yield a unique solution if the points are all coplanar in 3-D space.

[7] A "nearly planar" surface is a surface that can be approximated by a plane within a given tolerance, which is typically proportional to the distance of the surface from the image plane.



(a) (b)

Figure 8.5 Three couples of instantaneously coincident image points and their flow vectors (a); the difference vectors point towards the instantaneous epipole (b).

component of motion in 3-D space; this section makes this statement more precise. Motion parallax will be used in section 8.5.2 to compute structure and motion from optical flow.

Let two points $\mathbf{P} = [X, Y, Z]^\top$ and $\bar{\mathbf{P}} = [\bar{X}, \bar{Y}, \bar{Z}]^\top$ be projected into the image points **p** and $\bar{\mathbf{p}}$, respectively. We know that the corresponding motion field vectors can be written as

$$v_x = v_x^T + v_x^\omega$$
$$v_y = v_y^T + v_y^\omega$$

and

$$\bar{v}_x = \bar{v}_x^T + \bar{v}_x^\omega$$
$$\bar{v}_y = \bar{v}_y^T + \bar{v}_y^\omega.$$

If, at some instant t, the points **p** and $\bar{\mathbf{p}}$ happen to be coincident (Figure 8.5(a)), we have

$$\mathbf{p} = \bar{\mathbf{p}} = [x, y]^\top ,$$

and the rotational components of the observed motion, (v_x^ω, v_y^ω) and $(\bar{v}_x^\omega, \bar{v}_y^\omega)$, become

$$\begin{aligned} v_x^\omega = \bar{v}_x^\omega &= -\omega_y f + \omega_z y + \frac{\omega_x xy}{f} - \frac{\omega_y x^2}{f} \\ v_y^\omega = \bar{v}_y^\omega &= \omega_x f - \omega_z x - \frac{\omega_y xy}{f} + \frac{\omega_x y^2}{f}. \end{aligned} \tag{8.14}$$

Therefore, by taking the difference between **v** and $\bar{\mathbf{v}}$, the rotational components cancel out, and we obtain

$$\Delta v_x = v_x^T - \bar{v}_x^T = (T_z x - T_x f)(\frac{1}{Z} - \frac{1}{\bar{Z}})$$

$$\Delta v_y = v_y^T - \bar{v}_y^T = (T_z y - T_y f)(\frac{1}{Z} - \frac{1}{\bar{Z}}).$$

The vector $(\Delta v_x, \Delta v_y)$ can be thought of as the *relative motion field*. Other factors being equal, Δv_x and Δv_y increase with the separation in depth between **P** and $\bar{\mathbf{P}}$.

Notice that the ratio between Δv_y and Δv_x can be written as

$$\frac{\Delta v_y}{\Delta v_x} = \frac{y - y_0}{x - x_0}$$

with $[x_0, y_0]^T$ image coordinates of $\mathbf{p}_0$, the vanishing point of the translation direction (Figure 8.5(b)).[8] Hence, *for all possible rotational motions, the vector* $(\Delta v_x^T, \Delta v_y^T)$ *points in the direction of* $\mathbf{p}_0$. Consequently, the dot product between the motion field, **v**, and the vector $[y - y_0, -(x - x_0)]^T$, which is perpendicular to $\mathbf{p} - \mathbf{p}_0$, depends neither on the 3-D structure of the scene nor on the translational component of motion, and can be written as

$$v_\perp = (y - y_0)v_x^\omega - (x - x_0)v_y^\omega.$$

We will make use of this result in section 8.5.2, where we will learn how to compute motion and structure from dense estimates of the motion field.

☞ Be aware that the vanishing point of translation, $\mathbf{p}_0$, and the point at which **v** vanishes, call it **q**, are in general *different*; they coincide only if the motion is purely translational. Any rotational component about an axis not perpendicular to the image plane shifts the position of **q**, whereas the position of $\mathbf{p}_0$ remains unchanged, as it is determined by the translational component only. Somewhat deceptively, the flow field in the neighborhood of **q** might still look very much like a focus of expansion or contraction (see Figure 8.3).

And here is the customary summary of the main ideas.

Motion Parallax

The relative motion field of two instantaneously coincident points:

1. does not depend on the rotational component of motion
2. points towards (away from) the point $\mathbf{p}_0$, the vanishing point of the translation direction

[8] Section 8.2.5 makes it clear that this point can be regarded as an *instantaneous epipole*.

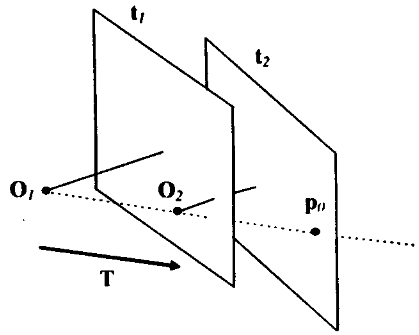


Figure 8.6 The point $\mathbf{p}_0$ as instantaneous epipole.

8.2.5 The Instantaneous Epipole

We close this introductory section with an important remark. The point $\mathbf{p}_0$, being the intersection of the image plane with the direction of translation of the center of projection, can be regarded as the *instantaneous epipole* between pairs of consecutive frames in the sequence (Figure 8.6). The main consequence of this property is that *it is possible to locate* $\mathbf{p}_0$ *without prior knowledge of the camera intrinsic parameters* (section 8.5.2).

☞ Notice that, as in the case of stereo, knowing the epipole's location in image coordinates is *not* equivalent to knowing the direction of translation (the baseline vector for stereo). The relation between epipole location and translation direction is specified by (8.9), which is written in the camera (not image) frame, and contains the focal length f. Therefore, *the epipole's location gives the direction of translation only if the intrinsic parameters of the viewing camera are known.*

8.3 The Notion of Optical Flow

We now move to the problem of *estimating the motion field from image sequences*, that is, from the spatial and temporal variations of the image brightness. To do this, we must model the link between brightness variations and motion field, and arrive at a fundamental equation of motion analysis, the *image brightness constancy equation*. We want also to analyze the power and validity of this equation, that is, understand how much and how well it can help us to estimate the motion field. For simplicity, we will assume that *the image brightness is continuous and differentiable as many times as needed in both the spatial and temporal domain.*

8.3.1 The Image Brightness Constancy Equation

It is common experience that, under most circumstances, the apparent brightness of moving objects remains constant. We have seen in Chapter 2 that the image irradiance is proportional to the scene radiance in the direction of the optical axis of the camera; if we assume that the proportionality factor is the same across the entire image plane, the constancy of the apparent brightness of the observed scene can be written as the stationarity of the image brightness E over time:

$$\frac{dE}{dt} = 0. \tag{8.15}$$

☞ In (8.15), the image brightness, E, should be regarded as a function of both the spatial coordinates of the image plane, x and y, and of time, that is, $E = E(x, y, t)$. Since x and y are in turn functions of t, the *total* derivative in (8.15) should not be confused with the *partial* derivative $\partial E/\partial t$.

Via the chain rule of differentiation, the total temporal derivative reads

$$\frac{dE(x(t), y(t), t)}{dt} = \frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0. \tag{8.16}$$

The partial spatial derivatives of the image brightness are simply the components of the spatial image gradient, ∇E, and the temporal derivatives, dx/dt and dy/dt, the components of the motion field, $\mathbf{v}$. Using these facts, we can rewrite (8.16) as the image brightness constancy equation.

The Image Brightness Constancy Equation

Given the image brightness, $E = E(x, y, t)$, and the motion field, $\mathbf{v}$,

$$(\nabla E)^{\top}\mathbf{v} + E_t = 0. \tag{8.17}$$

The subscript t denotes partial differentiation with respect to time.

We shall now discuss the relevance and applicability of this equation for the estimation of the motion field.

8.3.2 The Aperture Problem

How much of the motion field can be determined through (8.17)? *Only its component in the direction of the spatial image gradient,*[9] v_n. We can see this analytically by isolating the measurable quantities in (8.17):

$$-\frac{E_t}{\|\nabla E\|} = \frac{(\nabla E)^{\top}\mathbf{v}}{\|\nabla E\|} = v_n \tag{8.18}$$

[9] This component is sometimes called the *normal component*, because the spatial image gradient is normal to the spatial direction along which image intensity remains constant.

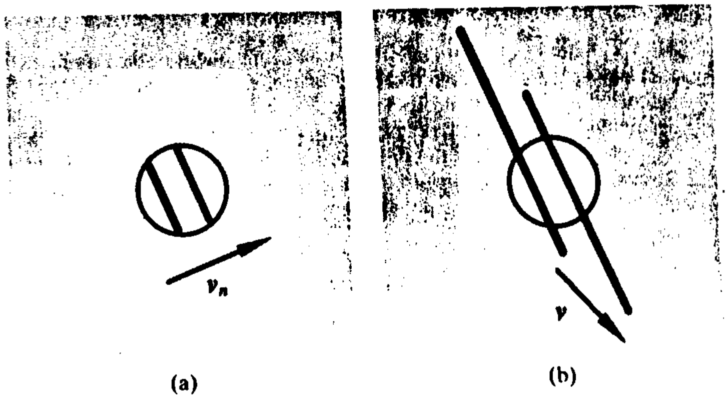


Figure 8.7 The aperture problem: the black and grey lines show two positions of the same image line in two consecutive frames. The image velocity perceived in (a) through the small aperture, $\mathbf{v}_n$, is only the component parallel to the image gradient of the true image velocity, $\mathbf{v}$, revealed in (b).

The Aperture Problem

The component of the motion field in the direction *orthogonal* to the spatial image gradient is not constrained by the image brightness constancy equation.

The aperture problem can be visualized as follows. Imagine to observe a thin, black rectangle moving against a white background through a small aperture. "Small" means that the corners of the rectangle are not visible through the aperture (Figure 8.7(*a*)); the small aperture simulates the narrow support of a differential method. Clearly, there are many, actually infinite, motions of the rectangle compatible with what you see through the aperture (Figure 8.7(*b*)); the visual information available is only sufficient to determine the velocity in the direction *orthogonal* to the visible side of the rectangle; the velocity in the *parallel* direction cannot be estimated.

☞ Notice that the parallel between (8.17) and Figure 8.7 is not perfect. Equation (8.17) relates the image gradient and the motion field at the *same* image point, thereby establishing a constraint on an *infinitely small* spatial support; instead, Figure 8.7 describes a state of affairs over a *small but finite* spatial region. This immediately suggests that a possible strategy for solving the aperture problem is to look at the spatial and temporal variations of the image brightness over a neighborhood of each point.[10]

[10] Incidentally, this strategy appears to be adopted by the visual system of primates.

8.3.3 The Validity of the Constancy Equation: Optical Flow

How well does (8.17) estimate the normal component of the motion field? To answer this question, we can look at the difference, Δv, between the true value and the one estimated by the equation. To do this, we must introduce a model of image formation, accounting for the reflectance of the surfaces and the illumination of the scene.

For the purposes of this discussion, we restrict ourselves to a Lambertian surface, S, illuminated by a pointwise light source infinitely far away from the camera (Chapter 2). Therefore, ignoring photometric distorsion, we can write the image brightness, E, as

$$E = \rho \mathbf{I}^{\top}\mathbf{n}, \tag{8.19}$$

where ρ is the surface albedo, $\mathbf{I}$ identifies the direction and intensity of illumination, and $\mathbf{n}$ is the unit normal to S at $\mathbf{P}$.

Let us now compute the total temporal derivative of both sides of (8.19). The only quantity that depends on time on the right hand side is the normal to the surface. If the surface is moving relative to the camera with translational velocity $\mathbf{T}$ and angular velocity $\boldsymbol{\omega}$, the orientation of the normal vector $\mathbf{n}$ will change according to

$$\frac{d\mathbf{n}}{dt} = \boldsymbol{\omega} \times \mathbf{n}, \tag{8.20}$$

where $\times$ indicates vector product. Therefore, taking the total temporal derivative of both sides of (8.19), and using (8.17) and (8.20), we have

$$\nabla E^{\top}\mathbf{v} + E_t = \rho \mathbf{I}^{\top}(\boldsymbol{\omega} \times \mathbf{n}). \tag{8.21}$$

We can obtain the desired expression for Δv from (8.18) and (8.21):

$$|\Delta v| = \rho \frac{|\mathbf{I}^{\top}\boldsymbol{\omega} \times \mathbf{n}|}{\|\nabla E\|}.$$

We conclude that, even under the simplifying assumption of Lambertian reflectance, the image brightness constancy equation yields the true normal component of the motion field (that is, $|\Delta v|$ is identically 0 for every possible surface) only for (a) purely translational motion, or (b) for any rigid motion such that the illumination direction is parallel to the angular velocity.

Other factors being equal, the difference Δv decreases as the magnitude of the spatial gradient increases; this suggests that *points with high spatial image gradient are the locations at which the motion field can be best estimated by the image brightness constancy equation.*

In general, $|\Delta v|$ is unlikely to be identically zero, and *the apparent motion of the image brightness is almost always different from the motion field.* For this reason, to avoid confusion, we call the apparent motion an *optical flow*, and refer to techniques estimating the motion field from the image brightness constancy equation as *optical flow techniques*. Here is a summary of similarities and differences between motion field and optical flow.

Definition: Optical Flow

The *optical flow* is a vector field subject to the constraint (8.17), and loosely defined as the *apparent motion* of the image brightness pattern.

Optical Flow and Motion Field

The optical flow is the *approximation of the motion field* which can be computed from time-varying image sequences. Under the simplifying assumptions of

- Lambertian surfaces
- pointwise light source at infinity
- no photometric distortion

the *error* of this approximation is

- *small* at points with high spatial gradient
- *exactly zero* only for translational motion or for any rigid motion such that the illumination direction is parallel to the angular velocity

We are now ready to learn algorithms estimating the motion field.

8.4 Estimating the Motion Field

The estimation of the motion field is a useful starting point for the solution of many motion problems. The many techniques devised by the computer vision community can be roughly divided into two major classes: *differential techniques* and *matching techniques*. Differential techniques are based on the spatial and temporal variations of the image brightness at all pixels, and can be regarded as methods for computing optical flow. Matching techniques, instead, estimate the disparity of special image points (features) between frames. We examine differential techniques in section 8.4.1; matching is the theme of section 8.4.2.

8.4.1 Differential Techniques

In recent (and not so recent) years a large number of differential techniques for computing optical flow have been proposed. Some of them require the solution of a system of partial differential equations, others the computation of second and higher-order derivatives of the image brightness, others again least-squares estimates of the parameters characterizing the optical flow. Methods in the latter class have at least two advantages over those in the first two:

- They are not iterative; therefore, they are genuinely local, and less biased than iterative methods by possible discontinuities of the motion field.
- They do not involve derivatives of order higher than the first; therefore, they are less sensitive to noise than methods requiring higher-order derivatives.

We describe a differential technique that gives good results. The basic assumption is that the motion field is well approximated by a *constant* vector field, $\mathbf{v}$, within any small region of the image plane.[11]

Assumptions

1. The image brightness constancy equation yields a good approximation of the normal component of the motion field.
2. The motion field is well approximated by a *constant* vector field within any small patch of the image plane.

An Optical Flow Algorithm. Given Assumption 1, for each point $\mathbf{p}_i$ within a small, $N \times N$ patch, Q, we can write

$$(\nabla E)^\top \mathbf{v} + E_t = 0$$

where the spatial and temporal derivatives of the image brightness are computed at $\mathbf{p}_1, \mathbf{p}_2 \ldots \mathbf{p}_{N^2}$.

☞ A typical size of the "small patch" is 5 × 5.

Therefore, the optical flow can be estimated within Q as the constant vector, $\bar{\mathbf{v}}$, that minimizes the functional

$$\Psi[\mathbf{v}] = \sum_{\mathbf{p}_i \in Q} \left[(\nabla E)^\top \mathbf{v} + E_t \right]^2 .$$

The solution to this least squares problem can be found by solving the linear system

$$A^\top A \mathbf{v} = A^\top \mathbf{b}. \tag{8.22}$$

The i-th row of the $N^2 \times 2$ matrix A is the spatial image gradient evaluated at point $\mathbf{p}_i$:

$$A = \begin{bmatrix} \nabla E(\mathbf{p}_1) \\ \nabla E(\mathbf{p}_2) \\ \cdot \\ \cdot \\ \cdot \\ \nabla E(\mathbf{p}_{N \times N}) \end{bmatrix}, \tag{8.23}$$

and $\mathbf{b}$ is the N^2-dimensional vector of the partial temporal derivatives of the image brightness, evaluated at $\mathbf{p}_1, \ldots \mathbf{p}_{N^2}$, after a sign change:

$$\mathbf{b} = -\left[E_t(\mathbf{p}_1), \ldots, E_t(\mathbf{p}_{N \times N})\right]^\top . \tag{8.24}$$

[11] Notice that this is in agreement with the first conclusion of section 8.2.3 (motion field of moving planes) regarding the approximation of smooth motion fields.

The least squares solution of the overconstrained system (8.22) can be obtained as[12]

$$\bar{\mathbf{v}} = (A^\top A)^{-1} A^\top \mathbf{b}. \tag{8.25}$$

$\bar{\mathbf{v}}$ is the optical flow (the estimate of the motion field) at the center of patch Q; repeating this procedure for all image points, we obtain a dense optical flow. We summarize the algorithm as follows:

Algorithm CONSTANT_FLOW

The input is a time-varying sequence of n images, $E_1, E_2, \ldots E_n$. Let Q be a square region of $N \times N$ pixels (typically, $N = 5$).

1. Filter each image of the sequence with a Gaussian filter of standard deviation equal to σ_s (typically $\sigma_s = 1.5$ pixels) along each spatial dimension.
2. Filter each image of the sequence along the temporal dimension with a Gaussian filter of standard deviation σ_t (typically $\sigma_t = 1.5$ frames). If $2k + 1$ is the size of the temporal filter, leave out the first and last k images.
3. For each pixel of each image of the sequence:
   (a) compute the matrix A and the vector $\mathbf{b}$ using (8.23) and (8.24)
   (b) compute the optical flow using (8.25)

The output is the optical flow computed in the last step.

☞ The purpose of spatial filtering is to attenuate noise in the estimation of the spatial image gradient; temporal filtering prevents aliasing in the time domain. For the implementation of the temporal filtering, imagine to stack the images one on top of the other, and filter sequences of pixels having the same coordinates. Note that the size of the temporal filter is linked to the maximum speed that can be "measured" by the algorithm.

An Improved Optical Flow Algorithm. We can improve CONSTANT_FLOW by observing that the error made by approximating the motion field at $\mathbf{p}$ with its estimate at the center of a patch increases with the distance of $\mathbf{p}$ from the center itself. This suggests a *weighted* least-square algorithm, in which the points close to the center of the patch are given more weight than those at the periphery. If W is the weight matrix, the solution, $\bar{\mathbf{v}}_w$, is given by

$$\bar{\mathbf{v}}_w = (A^\top W^2 A)^{-1} A^\top W^2 \mathbf{b}.$$

Concluding Remarks on Optical Flow Methods. It is instructive to examine the image locations at which CONSTANT_FLOW fails. As we have seen in Chapter 4, the 2×2 matrix

$$A^\top A = \begin{pmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{pmatrix}, \tag{8.26}$$

[12] See Appendix, section A.6 for alternative ways of solving overconstrained linear systems.

computed over an image region Q, is singular if and only if all the spatial gradients in Q are null or parallel. In this case the aperture problem cannot be solved, and the only possibility is to pick the solution of minimum norm, that is, the normal flow. The fact that we have already met the matrix $A^{\top}A$ in Chapter 4 is not a coincidence; the next section tells you why.

Notice that CONSTANT_FLOW gives good results because the spatial structure of the motion field of a rigid motion is well described by a low-degree polynomial in the image coordinates (as shown in section 8.2.3). For this reason, the assumption of local constancy of the motion field over small image patches is quite effective.

8.4.2 Feature-based Techniques

The second class of methods for estimating the motion field is formed by so-called *matching techniques*, which estimate the motion field at feature points only. The result is a sparse motion field. We start with a two-frame analysis (finding feature disparities between consecutive frames), then illustrate how *tracking* the motion of a feature across a long image sequence can improve the robustness of frame-to-frame matching.

Two-Frame Methods: Feature Matching. If motion
consecutive frames, the same matching methods can be us
This is true for both correlation-based and feature-based m
concentrate on *matching feature points*. You can easily ada
case too.

The point-matching method we describe is reminiscent of the CONSTANT_FLOW algorithm, and based on the features we met in Chapter 4. There, we looked at the matrix $A^{\top}A$ of (8.26), computed over small, square image regions: the features were the centers of those regions for which the smallest eigenvalue of $A^{\top}A$ was larger than a threshold. The idea of our matching method is simple: compute the displacement of such feature points by iterating algorithm CONSTANT_FLOW.

The procedure consists of three steps. First, the uniform displacement of the square region Q is estimated through CONSTANT_FLOW, and added to the current displacement estimate (initially set to 0). Second, the patch Q is *warped* according to the estimated flow. This means that Q is displaced according to the estimated flow, and the resulting patch, Q', is resampled in the pixel grid of frame I_2. If the estimated flow equals (v_x, v_y), the gray value at pixel (i, j) of Q' can be obtained from the gray values of the pixels of Q close to $(i - v_y, j - v_x)$. For our purpose, bilinear interpolation[14] is sufficient. Third, the first and second steps are iterated until a stopping criterion is met. Here is the usual algorithm box, containing an example of stopping criterion.

[13] But keep in mind the discussion of section 8.2.1 on the differences between stereo and motion disparities.

[14] Bilinear interpolation means that the interpolation is linear in each of the four pixels closest to $(i - v_y, j - v_x)$.

Algorithm FEATURE_POINT_MATCHING

The input is formed by I_1 and I_2, two frames of an image sequence, and a set of corresponding feature points in the two frames.

Let Q_1, Q_2, and Q' be three $N \times N$ image regions, and τ a fixed, positive real number. Let $\mathbf{d}$ be the unknown displacement between I_1 and I_2 of a feature point $\mathbf{p}$ on which Q_1 is centered.

For all feature points $\mathbf{p}$:

```
1. Set $\mathbf{d} = 0$ and center $Q_1$ on $\mathbf{p}$.
2. Estimate the displacement $\mathbf{d}_0$ of $\mathbf{p}$, center of $Q_1$, through (8.25) and let $\mathbf{d} = \mathbf{d} + \mathbf{d}_0$.
3. Let $Q'$ be the patch obtained by warping $Q_1$ according to $\mathbf{d}_0$. Compute $S$, the sum of the
   squared differences (the SSD of Chapter 7) between the new patch $Q'$ and the corresponding patch $Q_2$ in the frame $I_2$.
4. If $S > \tau$, set $Q_1 = Q'$ and go to step 1; otherwise exit.
```

The output is an estimate of $\mathbf{d}$ for all feature points.

☞ In both the smoothing stage, necessary to compute the derivatives in (8.25), and the warping stage of steps 2 and 3 respectively, you should consider a region actually larger than Q_1 (say by a factor 2). This enables you to iterate the procedure without introducing boundary effects.

☞ An alternative stopping criterion is to control the relative variation of the estimated flow at each iteration and exit the loop if the relative variation falls below a fixed threshold.

Multiple-Frame Methods: Feature Tracking. As we assume to analyze long image sequences, not just pairs of frames, we can improve on two-frame feature matching. We start with an intuitive fact: if the motion of the observed scene is continuous, as it nearly always is, we should be able to make *predictions* on the motion of the image points, at any instant, on the basis of their previous trajectories. In other words, we expect the motion of image points to be continuous, and therefore predictable, in most cases: we should be able to use the disparities computed between frames I_{i-1} and I_{i-2}, I_{i-2} and I_{i-3}, and so on, to make predictions on the disparities between I_{i-1} and I_i, *before* observing frame I_i.

Definition: Feature Tracking

Feature tracking is the problem of matching features from frame to frame in long sequences of images.

We approach tracking in the general framework of *optimal estimation theory*; our solution is the *Kalman filter*. For our purposes, a Kalman filter is a recursive algorithm which estimates the *position* and *uncertainty* of a moving feature point in the next frame, that is, where to look for the feature, and how large a region should be searched in the next frame, around the predicted position, to be sure to find the feature within a certain confidence. An introduction to the basic elements of Kalman filter theory is necessary to understand this section, and can be found in the Appendix, section A.8. Read it now if you are not familiar with the Kalman filter.

Let us formalize the tracking problem. A new frame of the image sequence is acquired and processed at each instant, $t_k = t_0 + k$, where k is a natural number. The sampling interval is assumed 1 for simplicity, and, more importantly, small enough to consider the motion of feature points from frame to frame linear.

We consider *only one* feature point, $\mathbf{p}_k = [x_k, y_k]^\top$, in the frame acquired at instant t_k, moving with velocity $\mathbf{v}_k = [v_{x,k}, v_{y,k}]^\top$. We describe the motion on the image plane with the state vector $\mathbf{x} = [x_k, y_k, v_{x,k}, v_{y,k}]^\top$. Assuming a sufficiently small sampling interval (and therefore constant feature velocity between frames), we write the system model of the linear Kalman filter as

$$\begin{aligned}\mathbf{p}_k &= \mathbf{p}_{k-1} + \mathbf{v}_{k-1} + \boldsymbol{\xi}_{k-1} \\ \mathbf{v}_k &= \mathbf{v}_{k-1} + \boldsymbol{\eta}_{k-1},\end{aligned} \tag{8.27}$$

where $\boldsymbol{\xi}_{k-1}$ and $\boldsymbol{\eta}_{k-1}$ are zero-mean, white, Gaussian random processes modelling the system noise. In terms of the state vector $\mathbf{x}_k$, (8.27) rewrites

$$\mathbf{x}_k = \Phi_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_{k-1},$$

with

$$\Phi_{k-1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{w}_{k-1} = \begin{bmatrix} \boldsymbol{\xi}_{k-1} \\ \boldsymbol{\eta}_{k-1} \end{bmatrix}.$$

As to measurements, we assume that a fast feature extractor estimates $\mathbf{z}_k$, the position of the feature point $\mathbf{p}_k$, at every frame of a sequence. Therefore, the measurement model of the Kalman filter becomes

$$\mathbf{z}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix} + \boldsymbol{\mu}_k,$$

with $\boldsymbol{\mu}_k$ a zero-mean, white, Gaussian random processes modelling the measurement noise.

Assumptions and Problem Statement

In the assumptions of the linear Kalman filter (Appendix, section A.8), and given the noisy observations $\mathbf{z}_k$, compute the best estimate of the feature's position and velocity at instant t_k and their uncertainties.

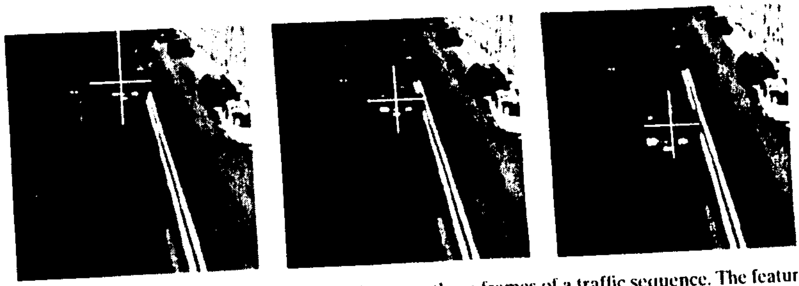


Figure 8.8 An example of feature tracking over three frames of a traffic sequence. The feature tracked is the centroid of the car, marked with a cross.

The Kalman filter algorithm is summarized in the following equations, repeated here from the Appendix, section A.8 for completeness.

Algorithm KALMAN_TRACKING

The input is formed, at instant t_k, by the covariance matrices of system and measurement nois at time t_{k-1}, Q_{k-1} and R_{k-1} respectively, the time-invariant state matrix, Φ, the time-invariant measurement matrix, H, and the position measurement at time t_k, $\mathbf{z}_k$. The entries of P_0 are set t high, arbitrary values.

$$\begin{aligned}P'_k &= \Phi_{k-1}P_{k-1}\Phi^\top_{k-1} + Q_{k-1} \\ K_k &= P'_kH^\top_k(H_kP'_kH^\top_k + R_k)^{-1} \\ \hat{\mathbf{x}}_k &= \Phi_{k-1}\hat{\mathbf{x}}_{k-1} + K_k(\mathbf{z}_k - H_k\Phi_{k-1}\hat{\mathbf{x}}_{k-1}) \\ P_k &= (I - K_k)P'_k(I - K_k)^\top + K_kR_kK^\top_k.\end{aligned}$$

The output is the optimal estimation of the position and velocity at time t_k, $\hat{\mathbf{x}}_k$, and their unc tainties, given by the diagonal elements of P_k.

Two things are worth noticing here. First, we do not just believe the noisy measu ments of $\mathbf{p}_k$ of the feature detector; the filter integrates them with model prediction obtain optimal estimates. Second, *the filter quantifies the uncertainty on the state estim* in the form of the diagonal elements of the state covariance matrix. This informat allows the feature detector to dimension automatically the image region to be searc to find the feature point in the next frame. The search region is centered on the position estimate, and is larger the larger the uncertainty. The elements of the covariance matrix are usually initialized to very large values; in a well-designed fil they decrease and reach a steady state rapidly, thereby restricting the search regio an image feature within a few frames. An example of tracking with Kalman filteri shown in Figure 8.8. The centroid of the car in the image (indicated by the white cro tracked over time. The size of the cross is proportional to the uncertainty in the syst state.

Two problems arise in the implementation of this algorithm.

Missing Information

Kalman filtering is based on the knowledge of the following:

1. the system model and the corresponding noise covariance matrix, Q_k
2. the measurement model and the corresponding noise covariance matrix, R_k
3. the initial (time t_0) system state, $\hat{\mathbf{x}}_0$, and state covariance matrix, P_0

However, several of these quantities are usually unknown.

Data Association

In the presence of several image features and multiple measurements, which observed measurements should be associated with which feature?

Missing Information. Fortunately, this problem is not as bad as you could expect. The system model is usually unknown, but is assumed linear if the time sampling is fast enough. The measurement model is available, as we assume that feature positions are computed at each frame. A really critical parameter, instead, is *the relative weight of model prediction and measurements* expressed by the filter's *gain*, K_k. From the equation of K_k, we see that the gain depends on the covariance matrices of system and measurement noise. In particular, if the entries of R_k are much smaller than those of Q_k (that is, the system model is much noisier, and therefore more uncertain, than the measurements), the Kalman filter ignores the prediction of the system model and relies almost entirely on measurements. Conversely, if the entries of Q_k are much smaller than the entries of R_k (that is, the measurements are much more uncertain than the prediction), the filter ignores the measurements and relies almost entirely on the prediction of the system model. Clearly, one aims at a balanced situation to achieve the greatest benefit from the integration of measurements and prediction. To achieve a balance, one can estimate R_k on the basis of the information available on the measuring process, then scale the entries of Q_k, making them comparable with those of R_k.

Finally, the state and its covariance can be initialized far off their asymptotic values with no risk of compromising the filter convergence.[15]

Data Association. This is a nontrivial problem in general, as there may be many features to be tracked. You should look into the Further Readings for a detailed analysis of techniques dealing with it. Here, we just consider briefly the case of *low clutter* and multiple but *noninterfering targets*. Low clutter means that the likelihood of noisy features at each frame (e.g., false features, features appearing for one frame only) is low. Noninterfering targets means that feature paths do not intersect. In this case, the technique known as *nearest neighbor data association* (NNDA) is the most effective. NNDA just selects the measurement associated with the updated state nearest to the predicted state (see Figure 8.9(a)). It is good practice to measure the distance between states by means of the inverse of the state covariance matrix (Exercise 8.5). NNDA is clearly suboptimal (Figure 8.9(b)) and more sophisticated methods are required to deal with high clutter and interfering targets.

[15] If the filter's assumptions are satisfied, of course.

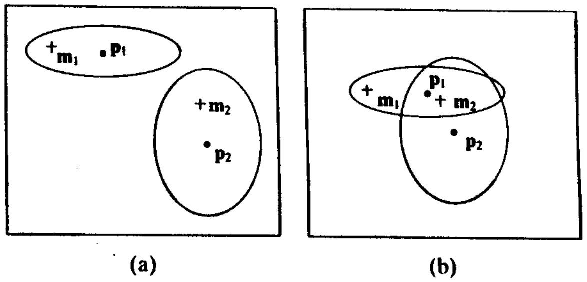


Figure 8.9 (a) Disjoint search regions of two features, centered around the best position estimates $\mathbf{p}_1, \mathbf{p}_2$; the measurements $\mathbf{m}_1, \mathbf{m}_2$ are associated to the closest estimates. (b) If the search regions intersect, the minimum-distance criterion fails.

8.5 Using the Motion Field

Now that we have various ways to estimate the motion field, what do we do with it? We target two tasks of practical importance, the reconstruction of 3-D motion and structure.

Problem Statement

Given the motion field estimated from an image sequẽnce, compute the shape, or *structure*, of the visible objects, and their *motion* with respect to the viewing camera.

Once again we distinguish between methods using dense and sparse estimates of the motion field.

8.5.1 3-D Motion and Structure from a Sparse Motion Field

In this section we estimate 3-D motion and structure from a sparse set of matched image features. If the average disparity between consecutive frames is small, the reconstruction can gain in stability and robustness from the time integration of long sequences of frames. If, on the contrary, the average disparity between frames is large, this problem can be dealt with in a stereo-like fashion, for example by means of the eight-point algorithm of Chapter 7 applied to a pair of frames. Of the many methods proposed in the literature for the former case, we have chosen the *factorization method*, which is simple to implement and gives very good (and numerically stable) results for objects viewed from rather large distances. The necessary assumptions are summarized below.

Assumptions: Factorization Method

1. The camera model is orthographic.
2. The position of n image points, corresponding to the scene points $\mathbf{P}_1, \mathbf{P}_2 \dots \mathbf{P}_n$, not all coplanar, have been tracked in N frames, with $N \geq 3$.

☞ Note that Assumption 2 is equivalent to acquiring the *entire* sequence before starting any processing. This may or may not be acceptable depending on the application. Notice also that, since the camera model is orthographic, camera calibration can be altogether ignored *if* we accept to reconstruct the 3-D points only up to a scale factor.

The remainder of this section introduces the necessary notations, discusses the *rank theorem*, on which the whole method is based, and states the complete factorization algorithm.

Notation. We let $\mathbf{p}_{ij} = [x_{ij}, y_{ij}]^\top$ denote the jth image point ($j = 1, \dots n$) at the i-th frame ($i = 1, \dots N$), and think of the x_{ij} and y_{ij} as entries of two $N \times n$ matrices, X and Y, respectively. We then form the $2N \times n$ *measurement matrix*

$$W = \begin{bmatrix} X \\ Y \end{bmatrix}.$$

For reasons that will be clear shortly, we subtract the mean of the entries on the same row from each x_{ij} and y_{ij}:

$$\begin{aligned} \tilde{x}_{ij} &= x_{ij} - \bar{x}_i \\ \tilde{y}_{ij} &= y_{ij} - \bar{y}_i, \end{aligned} \qquad (8.28)$$

where

$$\begin{aligned} \bar{x}_i &= \frac{1}{n}\sum_{j=1}^{n} x_{ij} \\ \bar{y}_i &= \frac{1}{n}\sum_{j=1}^{n} y_{ij} \end{aligned} \qquad (8.29)$$

are the coordinates of $\bar{\mathbf{p}}_i$, the centroid of the image points in the i-th frame. Again, we think of the $\tilde{x}_{ij}$ and $\tilde{y}_{ij}$ as entries of two $N \times n$ matrices, $\tilde{X}$ and $\tilde{Y}$, and form the $2N \times n$ matrix $\tilde{W}$, called the *registered measurement matrix*.

$$\tilde{W} = \begin{bmatrix} \tilde{X} \\ \tilde{Y} \end{bmatrix}. \qquad (8.30)$$

The Rank Theorem. The factorization method is based on the proof of a simple but fundamental result.

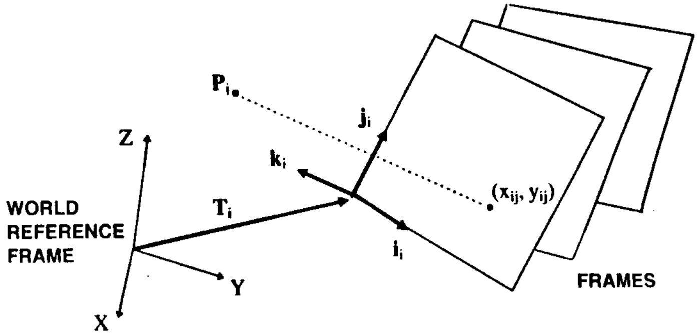


Figure 8.10 The geometry of the factorization method.

Rank Theorem

The registered measurement matrix (without noise) has at most rank 3.

The proof is based on the decomposition (*factorization*) of $\tilde{W}$ of (8.30) into the product of a $2N \times 3$ matrix, R, and a $3 \times n$ matrix, S. R describes the frame-to-frame rotation of the camera with respect to the points $\mathbf{P}_j$. S describes the points' structure (coordinates). The proof is essential for the actual algorithm, so we will go through it in detail.

We consider all quantities expressed in an object-centered reference frame with the origin in the centroid of $\mathbf{P}_1, \dots \mathbf{P}_n$ (Figure 8.10), and let $\mathbf{i}_i$ and $\mathbf{j}_i$ denote the unit vectors of the image reference frame, expressed in the world reference frame and at time instant i. Thus, the direction of the optical axis is given by the cross product of $\mathbf{i}_i$ and $\mathbf{j}_i$,

$$\mathbf{k}_i = \mathbf{i}_i \times \mathbf{j}_i.$$

It can be seen from Figure 8.10 that

$$x_{ij} = \mathbf{i}_i^\top (\mathbf{P}_j - \mathbf{T}_i) \qquad (8.31)$$

$$y_{ij} = \mathbf{j}_i^\top (\mathbf{P}_j - \mathbf{T}_i), \qquad (8.32)$$

where $\mathbf{T}_i$ is the vector from the world origin to the origin of the i-th image frame; moreover, as the origin is in the centroid of the points,

$$\frac{1}{n}\sum_{j=1}^{n} \mathbf{P}_j = 0. \qquad (8.33)$$

Now, plugging (8.31) and (8.32) into (8.28), and using (8.29), we obtain

$$\tilde{x}_{ij} = \mathbf{i}_i^\top(\mathbf{P}_j - \mathbf{T}_i) - \frac{1}{n}\sum_{m=1}^{n}\mathbf{i}_i^\top(\mathbf{P}_m - \mathbf{T}_i)$$

$$\tilde{y}_{ij} = \mathbf{j}_i^\top(\mathbf{P}_j - \mathbf{T}_i) - \frac{1}{n}\sum_{m=1}^{n}\mathbf{j}_i^\top(\mathbf{P}_m - \mathbf{T}_i). \tag{8.34}$$

But due to (8.33), and to the fact that the index i is not summed, (8.34) become

$$\tilde{x}_{ij} = \mathbf{i}_i^\top \mathbf{P}_j$$
$$\tilde{y}_{ij} = \mathbf{j}_i^\top \mathbf{P}_j$$

Therefore, if we define the $2N \times 3$ *rotation matrix* R as

$$R = \begin{bmatrix} \mathbf{i}_1^\top \\ \mathbf{i}_2^\top \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{i}_N^\top \\ \mathbf{j}_1^\top \\ \mathbf{j}_2^\top \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{j}_N^\top \end{bmatrix}, \tag{8.35}$$

and a $3 \times n$ *shape matrix* S as

$$S = [\,\mathbf{P}_1 \;\; \mathbf{P}_2 \;\; \ldots \;\; \mathbf{P}_n\,]. \tag{8.36}$$

we can write

$$\tilde{W} = RS.$$

Since the rank of R is 3 because $N \geq 3$, and the rank of S is also 3 because the N points in 3-D space are not all coplanar, the theorem is proved.

☞ Notice the importance of the assumption of noncoplanar points.

The importance of the rank theorem is twofold. First, it tells you that there is a great deal of redundancy in the image data: no matter how many points and views you are considering, the rank of the registered measurement matrix does not exceed three. Second, and most importantly, the factorization of the registered measurement matrix, $\tilde{W}$, as the product of **R** and **S** suggests a method for reconstructing structure and motion from a sequence of tracked image points.

The Factorization Algorithm. The factorization of $\tilde{W}$ is relatively straightforward. First of all, note that this factorization is not unique: if R and S factorize $\tilde{W}$, and Q is any invertible 3×3 matrix, then RQ and $Q^{-1}S$ also factorize $\tilde{W}$. The proof is simple:

$$(RQ)(Q^{-1}S) = R(QQ^{-1})S = RS = W.$$

Fortunately, we can add two constraints:

1. the rows of R, thought of as 3-D vectors, must have unit norm;
2. the first n rows of R (the $\mathbf{i}_i^\top$) must be orthogonal to the corresponding last n rows (the $\mathbf{j}_i^\top$).

Our last effort before reaching an algorithm box is to show that these constraints allow us to compute a factorization of $\tilde{W}$ which is unique up to an unknown initial orientation of the world reference frame with respect to the camera frame (Figure 8.10). At the same time, we also show how to extend the method to the case in which, due to noise or imperfect matching, the rank of the matrix $\tilde{W}$ is greater than 3. Here is the proof.

First, consider the singular value decomposition (Appendix, section A.6) of $\tilde{W}$,

$$\tilde{W} = UDV^\top. \tag{8.37}$$

The fact that the rank of $\tilde{W}$ is greater than 3 means that more than 3 singular values along the diagonal of D will not be zero. The rank theorem can be enforced simply by setting all but the three largest singular values in D to zero, and recomputing the corrected matrix $\tilde{W}$ from (8.37).

☞ By now, this should not surprise you. We used the same method elsewhere: e.g., to compute the closest rotation matrix to a numerical estimate in Chapter 6. Notice that, if the ratio between the third and fourth singular value is not large, as expected, the SVD warns you about the consistency of the data.

Then let D' be the 3×3 top left submatrix of D corresponding to the three largest singular values, σ_1, σ_2, and σ_3, and U' and V' the $2N \times 3$ and $n \times 3$ submatrices of U and V formed by the columns corresponding to σ_1, σ_2, and σ_3.

$$\hat{R} = U'D'^{1/2}$$
$$\hat{S} = D'^{1/2}V'^\top. \tag{8.38}$$

In general, the rows $\hat{\mathbf{i}}_i^\top$ and $\hat{\mathbf{j}}_i^\top$ of the matrix $\hat{R}$ will not satisfy the constraints mentioned above; however, if we look for a matrix Q such that

$$\hat{\mathbf{i}}_i^\top QQ^\top \hat{\mathbf{i}}_i = 1$$
$$\hat{\mathbf{j}}_i^\top QQ^\top \hat{\mathbf{j}}_i = 1 \tag{8.39}$$
$$\hat{\mathbf{i}}_i^\top QQ^\top \hat{\mathbf{j}}_i = 0.$$

then the new matrices $R = \hat{R}Q$ and $S = Q^{-1}R$ still factorize $\tilde{W}$, and the rows of R satisfy the constraints. The obtained factorization is now clearly unique up to an arbitrary rotation. One possible choice is to assume that at time $t = 0$ the world and camera reference frame coincide.

Here is a concise description of the entire method. A method for determining Q from (8.39) is discussed in Exercise 8.8.

Algorithm MOTSTRUCT_FROM_FEATS

The input is the registered measurement matrix $\tilde{W}$, computed from n features tracked over N consecutive frames.

1. Compute the SVD of $\tilde{W}$,

$$\tilde{W} = UDV^{\top},$$

where U is a $2N \times 2N$ matrix, V $n \times n$, and D $2N \times n$; $U^{\top}U = I$, $V^{\top}V = I$; and D is the diagonal matrix of the singular values.

2. Set to zero all but the three largest singular values in D.
3. Define $\hat{R}$ and $\hat{S}$ as in (8.5.1).
4. Solve (8.39) for Q.for example by means of Newton's method (Exercise 8.8).

The output are the rotation and shape matrices, given by

$$R = \hat{R}Q \quad \text{and} \quad S = Q^{-1}\hat{S}.$$

The algorithm determines the rotation of a set of 3-D points with respect to the camera, but how about their translation? The component of the translation parallel to the image plane is simply proportional to the frame-by-frame motion of the centroid of the data points on the image plane. However, because of the orthographic assumption, *the component of the translation along the optical axis cannot be determined.*

8.5.2 3-D Motion and Structure from a Dense Motion Field

We now discuss the reconstruction of 3-D motion and structure from optical flow. The two major differences with the previous section are that

- optical flow provides dense but often inaccurate estimates of the motion field;
- the analysis is instantaneous, not integrated over many frames.

Problem Statement

Given an optical flow and the intrinsic parameters of the viewing camera, recover the 3-D motion and structure of the observed scene with respect to the camera reference frame.

We have chosen a method that represents a good compromise between ease of implementation and quality of results. The method consists of two stages:

1. determine the direction of translation through approximate motion parallax;
2. determine a least-squares approximation of the rotational component of the optical flow, and use it in the motion field equations to compute depth.

Stage 1: Translation Direction. The first stage is rather complex. We start by explaining the solution in the ideal case of *exact* motion parallax, then move to the case of approximate parallax. We learned in section 8.2 that the relative motion field of two *instantaneously coincident* image points, $[\Delta v_x, \Delta v_y]^{\top}$, is directed towards (or away from) the vanishing point of the translation direction, $\mathbf{p}_0$ (the instantaneous epipole), according to

$$\Delta v_x = (T_z x - T_x f)\left(\frac{1}{Z} - \frac{1}{\bar{Z}}\right)$$
$$\Delta v_y = (T_z y - T_y f)\left(\frac{1}{Z} - \frac{1}{\bar{Z}}\right) \qquad (8.40)$$

where Z and $\bar{Z}$ are the depths of the 3-D points $P = (X, Y, Z)$ and $\bar{P} = (\bar{X}, \bar{Y}, \bar{Z})$, which project onto the same image point, $\mathbf{p} = [x, y]^{\top}$, in the frame considered. If (8.40) can be written for two different image points, we can locate the epipole, $\mathbf{p}_0$, as the intersection of the estimated, relative motion fields. Once the epipole is known, it is straightforward to get the direction of translation from (8.9).

☞ If (8.40) can be written for more than two points, we can resort to least squares to obtain a better estimate of the epipole's location.

This solution can be extended to the more realistic case of *approximate* motion parallax, in which the estimates of the relative motion field are available only for *almost coincident* image points. The key observation is that *the differences between the optical flow vectors at an image point* **p** *and at any point close to* **p** *can be regarded as noisy estimates of the motion parallax at* **p** (section 8.2.4).

We must now rewrite the (8.40) for the case of approximate parallax. We begin by writing the translational and rotational components of the relative motion field, $\left[\Delta v_x^T, \Delta v_y^T\right]^{\top}$ and $\left[\Delta_x^{\omega}, \Delta_y^{\omega}\right]^{\top}$ respectively, for two almost coincident image points, $\mathbf{p}$ and $\bar{\mathbf{p}}$. These are

$$\Delta v_x^T = \frac{T_z x - T_x f}{Z} - \frac{T_z \bar{x} - T_x f}{\bar{Z}}$$
$$\Delta v_y^T = \frac{T_z y - T_y f}{Z} - \frac{T_z \bar{y} - T_y f}{\bar{Z}} \qquad (8.41)$$

and

$$\Delta v_x^{\omega} = \omega_z(y - \bar{y}) + \frac{\omega_x}{f}(xy - \bar{x}\bar{y}) - \frac{\omega_y}{f}(x^2 - \bar{x}^2)$$
$$\Delta v_y^{\omega} = -\omega_z(x - \bar{x}) - \frac{\omega_y}{f}(xy - \bar{x}\bar{y}) + \frac{\omega_x}{f}(y^2 - \bar{y}^2). \qquad (8.42)$$

From the rotation equations we notice that $\Delta v_x^{\omega} \rightarrow 0$ and $\Delta v_y^{\omega} \rightarrow 0$ for $\bar{p} \rightarrow p$. As to the translation equations, we can rewrite them as

$$\begin{aligned} \Delta v_x^T &= (T_z x - T_x f)\left(\frac{1}{Z} - \frac{1}{\bar{Z}}\right) + \frac{T_z}{\bar{Z}}(x - \bar{x}) \\ \Delta v_y^T &= (T_z y - T_y f)\left(\frac{1}{Z} - \frac{1}{\bar{Z}}\right) + \frac{T_z}{\bar{Z}}(y - \bar{y}). \end{aligned} \tag{8.43}$$

The second terms of the right-hand side of the (8.43) tend to zero for $\bar{p} \rightarrow p$, while the first terms tend to the expression obtained for the exact motion parallax, (8.40). We can therefore write the relative motion field of two almost coincident points concisely as

$$\begin{aligned} \Delta v_x &= (T_z x - T_x f)(\frac{1}{Z} - \frac{1}{\bar{Z}}) + e_x(\mathbf{p} - \bar{\mathbf{p}}) \\ \Delta v_y &= (T_z y - T_y f)(\frac{1}{Z} - \frac{1}{\bar{Z}}) + e_y(\mathbf{p} - \bar{\mathbf{p}}), \end{aligned} \tag{8.44}$$

with e_x and e_y smooth functions of the difference between $\mathbf{p}$ and $\bar{\mathbf{p}}$, and $e_x(0) = e_y(0) = 0$.

Equations (8.44) show that, if $\mathbf{p}$ and $\bar{\mathbf{p}}$ are close enough, a *large* relative motion field can only be due to a *large* difference in depth between the 3-D points $\mathbf{P}$ and $\bar{\mathbf{P}}$. This observation suggests a relatively simple algorithm for locating the instantaneous epipole (and therefore the direction of translation) from a number of approximate motion parallax estimates. We compute the flow differences $(\Delta v_x, \Delta v_y)$ between a point $\mathbf{p}_i$ and all its neighbors within a small patch Q_i, then determine the eigenvalues and eigenvectors of the matrix

$$A_i = \begin{bmatrix} \sum \Delta^2 v_x & \sum \Delta v_x \Delta v_y \\ \sum \Delta v_x \Delta v_y & \sum \Delta^2 v_y \end{bmatrix}, \tag{8.45}$$

where the sums are taken over the Q_i; the eigenvector corresponding to λ_i, the greater eigenvalue, identifies the direction of the line $\hat{\mathbf{l}}_i$ through $\mathbf{p}_i$ which minimizes the sum of the squared distances to the set of difference vectors (Appendix, section A.6). This direction is taken to be the optimal estimate of motion parallax within the patch Q_i.

Moreover, λ_i itself can be regarded as a measure of the estimate's *reliability*. If λ_i is large,[16] the underlying distribution of the flow differences has a peak in the direction of $\hat{\mathbf{l}}_i$. This is likely to be due to the presence of considerable differences in depth within Q_i. Instead, if λ_i is small, the underlying distribution of the flow differences is flatter, and almost certainly created by the flow field of a surface that does not vary much in depth within Q_i.

[16] One might argue that what really counts should be the ratio between the smaller and greater eigenvalues. However, since the range of Δv_x and Δv_y is finite, the greater eigenvalue is *large* in absolute terms.

We can now formulate a weighted least squares scheme to compute the intersection of the several lines $\hat{\mathbf{l}}_i$, that is, the epipole $\mathbf{p}_0$. Since $\hat{\mathbf{l}}_i$, $\mathbf{p}_0$, and $\mathbf{p}_i$ are coplanar, for each patch Q_i we can write

$$(\hat{\mathbf{l}}_i \times \mathbf{p}_i)^{\top} \mathbf{p}_0 = 0. \tag{8.46}$$

If there are N patches, we can write N simultaneous instances of (8.46), that is, in matrix notation,

$$B\mathbf{p}_0 = 0 \tag{8.47}$$

with

$$B = \begin{bmatrix} \hat{\mathbf{l}}_1 \times \mathbf{p}_1^{\top} \\ \hat{\mathbf{l}}_2 \times \mathbf{p}_2^{\top} \\ \cdot \\ \cdot \\ \cdot \\ \hat{\mathbf{l}}_N \times \mathbf{p}_N^{\top} \end{bmatrix} \tag{8.48}$$

The problem of determining a least-squares estimate of $\mathbf{p}_0$ is thus reduced to the problem of solving the overconstrained homogeneous system (8.47). As customary, the solution can be found from the SVD of B, $B = UDV^{\top}$ (Appendix, section A.6), as the column of V corresponding to the null (in practice, the smallest) singular value of B.

☞ In order to give appropriate weights to the different estimates, it is better to use a weighted least-square scheme and consider the matrix WB, where the entries of the diagonal matrix W are the larger eigenvalues of A_i.

Stage Two: Rotational Flow and Depth. The rest of the algorithm is straightforward. We simply form the pointwise dot product, $v_{\perp}$, between the optical flow at point $\mathbf{p}_i = [x_i, y_i]^{\top}$ and the vector $[y_i - y_0, -(x_i - x_0)]^{\top}$. As we know from section 8.2, $v_{\perp}$ depends only on the rotational component of motion; therefore, at each point $\mathbf{p}_i$ of the image plane we have

$$v_{\perp} = v_x^{\omega}(y_i - y_0) - v_y^{\omega}(x_i - x_0) \tag{8.49}$$

with v_x^{ω} and v_y^{ω} as in (8.42). If the intrinsic parameters of the camera are known, we can write a linear system of N simultaneous instances of (8.49) in the image reference frame by using (8.14), and solve for the three components of the angular velocity using least squares. Finally, we recover the translational direction from the epipole coordinates by means of (8.9), and solve (8.7) for the depth Z of each image point.

It is now time to summarize the method.

Algorithm MOTSTRUCT_FROM_FLOW

The input quantities are the intrinsic parameters of the viewing camera, and a dense optical flow field, $\mathbf{v}$, produced by a single rigid motion.

1. Write (8.7) in the image reference frame, using the knowledge of the intrinsic parameters.
2. For each image point $\mathbf{p}_i, i = 1, \ldots N$:
   (a) compute the flow differences Δv_x and Δv_y between the optical flow at $\mathbf{p}_i$ and at all the points $\mathbf{p}$ in a neighborhood of $\mathbf{p}_i$, Q_i;
   (b) compute the eigenvalues and eigenvectors of the matrix A_i of (8.45); let λ_i be the greater eigenvalue, and $\hat{\mathbf{l}}_i$ the unit eigenvector corresponding to λ_i.
3. Compute the SVD of WB, $WB = UDV^\top$, with B as in (8.48) and W a diagonal matrix such that $W_{ii} = \lambda_i$. Estimate the epipole $\mathbf{p}_0$ as the column of V corresponding to the smallest singular value.
4. Form the dot product of (8.49) for $i = 1, \ldots, N$ and rewrite the equations obtained in the image reference frame.
5. Determine the angular velocity components as the least-squares solution of a system of N simultaneous instances of (8.49).
6. Determine the translational direction from the epipole coordinates and the knowledge of the intrinsic parameters (see (8.9)).
7. Solve (8.7) for the depth Z of each image point.

The output quantities are the direction of translation, the angular velocity, and the 3-D coordinates of the scene points.

☞ Notice that, as discussed in section 8.2.5, the epipole can be estimated *without prior knowledge of the camera parameters*, that is, with an uncalibrated camera. The direction of translation, instead, can be obtained from the epipole *only if the intrinsic parameters of the camera are known*.

MOTSTRUCT_FROM_FLOW is not as accurate as MOTSTRUCT_FROM_FEATS. This is not surprising, as MOTSRUCT_FROM_FLOW is an instantaneous method, which relies on local approximations of the observed motion, on the assumption of large variation in depth in the observed scene, and on the accuracy of camera calibration.

8.6 Motion-based Segmentation

In this final section, we relax the assumption that the motion between the camera and the scene is described by a single 3-D motion to deal with the problem of *multiple motions*. For the sake of simplicity, we restrict the analysis to the case in which the camera is fixed. If you are interested in motion segmentation in the presence of camera motion, a problem which is still waiting for a general and satisfactory solution, see the Further Readings.

If the camera is fixed, identifying moving objects can be seen as a problem of *detecting changes against a fixed background*.

Problem Statement

Given a sequence of images taken by a fixed camera, find the regions of the image, if any, corresponding to the different moving objects.

This problem can be thought of as a classification problem. One has to classify the pixels of each frame of a sequence as either *moving* or *fixed* with respect to the camera. On the basis of what we have seen so far, a possible procedure seems to be the computation of optical flow followed by a thresholding procedure. The pixels for which the norm of the optical flow is large enough are labelled as "moving", the others as "fixed". Two criticisms can be raised against this approach. First, the estimation of optical flow inside patches that contain pixels from independently moving objects is usually rather poor. Second, in many applications (surveillance tasks for example) the detection of motion or changes in the scene is all that is needed.

Presumably the simplest strategy for detecting changes in an image sequence is *image differencing*: *(i)* take the pointwise difference between consecutive frames, and *(ii)* label as "moving" the pixels for which this difference exceeds a predetermined threshold, t.

Algorithm CHANGE_DETECTION

The input is an image sequence, $I_1, I_2, \ldots, I_n$, and a positive real number, τ; for each image pair (I_k, I_{k+1})

1. Compute the pointwise image difference $\Delta_k(i,j) = I_{k+1}(i,j) - I_k(i,j)$;
2. if $|\Delta_k(i,j)| > \tau$, label pixel (i,j) of the frame k as moving.

The output is a map of the moving image regions.

The threshold must be chosen so that the probability of mistaking differences created by image noise for real motion is very small. A simple way to do this is to acquire two images of a static scene, with no illumination changes, and look at the histogram of the difference image. In agreement with the assumption that this difference is mainly due to the camera noise (see Chapter 2), the histogram should look like a zero-mean Gaussian function (see Figure 8.11). The threshold can thus be estimated as a multiple of the standard deviation of the computed distribution.

☞ Of course, the more different the images used to estimate the standard deviation, the more accurately the histogram reflects the distribution of the noise.

Better results can be obtained in a number of different ways. One possibility is to resort to statistical tests (see Further Readings). A second possibility is to take *motion measures* more sophisticated than image differencing. For instance, an alternative to image differencing is the weighted average of the normal flow magnitude, $|E_t|/\|\nabla E\|$.

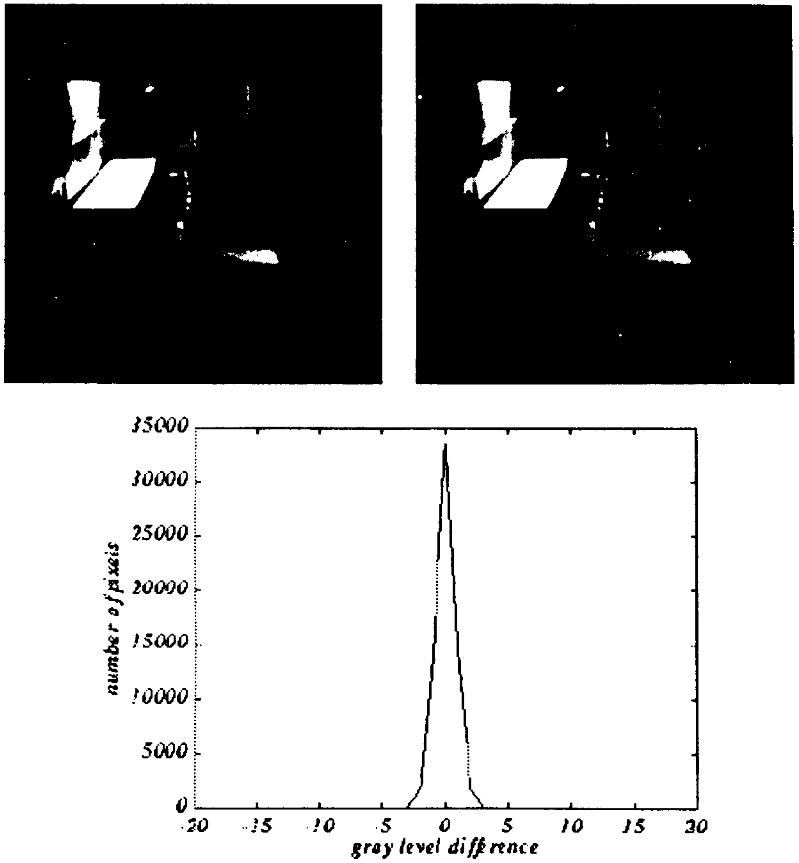


Figure 8.11 Two sample snapshots, taken at close instants, from an overnight, indoors surveillance sequence, and the histogram of the differences of the gray values between the two images.

over a small patch Q_i centered at point $\mathbf{p}_i$. The weights are taken to be the square of the norm of the spatial image gradient, so that this motion measure can be written as

$$M = \frac{|E_t|\|\nabla E\|}{C + \sum \|\nabla E\|^2},$$

where the sum are taken over Q_i, and the temporal and spatial derivatives are meant to be computed at each point of Q_i. The purpose of the constant C is to remove the instability that may be caused by uniform patches. The choice of the weights agrees with the results of section 8.3, which proved that the difference between the true motion field and the apparent motion of the image brightness is smaller at the locations where the norm of the spatial image gradient is larger. The conversion of this idea into an algorithm for change detection is straightforward, and left as an exercise.

8.7 Summary

After working through this chapter you should be able to:

- □ explain the fundamental problems of motion analysis in computer vision
- □ estimate the optical flow from a sequence of images
- □ match and tracking image features over time
- □ estimate 3-D motion and structure from both dense and sparse estimates of the motion field
- □ detect changes in image sequences taken from a fixed camera

8.8 Further Readings

The discussion on the difference between motion field and optical flow is taken from Verri and Poggio [19]. The method described here for estimating optical flow is due to Lucas and Kanade [12]. A variational approach to the computation of optical flow was first proposed by Horn and Schunck [8]. Of the many parametric methods proposed since see, for example, Campani and Verri [3] (in which you can also find a discussion on the spatial properties of the motion field). For a strictly local computation of optical flow you should start with Nagel's work (see [14], for example). A correlation-based method is described by Poggio, Little, and Gamble [15]. Finally, the paper by Barron, Fleet, and Beauchemin [2] is an excellent review of many methods for estimating optical flow, and includes Internet sites with public-domain code.

The literature on feature matching for motion is also vast. Among the many techniques available, you may want to look at the classic books by Ullman [18] and Hildreth [7]. A nice and simple account of Kalman filtering is the paper by Cooper [4]. Alternatively, you can look at the very clear and complete book by Maybeck [13]. Data association is discussed, for example, in the book edited by Bar-Shalom [1].

The factorization method for 3-D motion and structure from sparse features is due to Tomasi and Kanade [17]. Among the other feature-based method we suggest the technique proposed by Faugeras, Lustman, and Toscani [5]. The epipole's method for reconstruction from optical flow is based on the paper by Rieger and Lawton [16] and on an idea originally suggested by Longuet-Higgins and Prazdny [11]. Part of the algorithm MOTSTRUCT_FROM_FLOW is based on the implementation proposed in the appendix of the paper by Heeger and Jepson [6], the main topic of which is an alternative algorithm to MOTSTRUCT_FROM_FLOW.

For a thorough analysis of change detection see Hsu, Nagel and Rekers [9]. As to motion-based segmentation in the general case, you may want to look at the work of Irani, Rousso, and Peleg [10], for example, and references therein.

8.9 Review

Questions

□ *8.1* What are the properties of the motion field generated by a planar surface? Would an arbitrarily oriented planar surface generate a motion field across the entire image plane?

□ *8.2* What is the relation between the instantaneous epipole and the focus of expansion (or contraction)?

□ *8.3* What is the difference between motion field and optical flow?

□ *8.4* The derivation of (8.17) assumes that the image brightness is continuous and can be differentiated. How plausible is this assumption in general? Why?

□ *8.5* What are the assumptions behind the algorithm CONSTANT_FLOW?

□ *8.6* Given our discussion of the Kalman filter for feature tracking, how would you decide the shape and size of the search regions given the uncertainties produced by the filter?

□ *8.7* How would you decide whether the effective rank of $\tilde{W}$ in algorithm MOTSTRUCT_FROM_FEATS is 3?

□ *8.8* What happens if you apply MOTSTRUCT_FROM_FLOW to the optical flow of a planar surface?

□ *8.9* Why change detection methods are not useful (in themselves) for motion-based segmentation in the general case?

Exercises

○ *8.1* Estimate the ratio between the quadratic and linear terms, the quadratic and constant terms, and the linear and constant terms in (8.13). Set the motion and structure parameters of the planar surface to some arbitrary but reasonable values.

○ *8.2* Show that the aperture problem can be solved if a corner is visible through the aperture.

○ *8.3* Extend algorithm CONSTANT_FLOW by assuming that the motion field is locally approximated by a linear vector field.

○ *8.4* Create a simple, synthetic image sequence by shifting a given image by $[v_x, v_y]^\top$ pixel per frame (with v_x and v_y not integer), and using bilinear interpolation in the resampling stage. Then, apply the *inverse* transformation; that is, create an image sequence starting from the last frame and applying a motion of $[-v_x, -v_y]^\top$ pixel per frame. Compare the last frame of this sequence with the original image. Do you expect them to be *exactly* equal? If not, why?

○ *8.5* Show that the distance between system states computed through the covariance matrix, named *Mahalanobis distance*, is not isotropic. What is the advantage of using Mahalanobis distance instead of the usual Euclidean distance?

○ *8.6* Write the Kalman filter equations for a one-dimensional state vector.

○ *8.7* The celebrated structure-from-motion theorem, due to Shimon Ullman, states that, under orthographic projection, at least N views of at least n noncoplanar points are needed to uniquely recover structure. Can you guess the values of N and n from the factorization method?

○ *8.8* The nonlinear system (8.39) can be solved by means of *Newton's method*, an iterative procedure, the main idea of which is rather simple. Write down the full system of $3 \times N$ quadratic equations in the nine entries of the matrix Q, say $q_1, \ldots, q_9$. Starting from an initial guess for the values of $q_1, \ldots, q_9$ (like the identity matrix for example), take the partial derivatives of each equation with respect to each unknown, and evaluate the expressions obtained at the current value of the entries. If M_{ij} denotes the partial derivatives of the i-th equation with respect to q_j, the matrix M_{ij} can be viewed as the system matrix of the linear system

$$M \Delta \mathbf{q} = \boldsymbol{\epsilon},$$

where $\Delta \mathbf{q} = [\Delta q_1, \ldots, \Delta q_9]^\top$, the components of which should be used to update the current estimate of $q_1, \ldots, q_9$. The components of the $3N$-D vector $\boldsymbol{\epsilon}$ are the residuals of all equations, computed by means of the previous estimates of $q_1, \ldots, q_9$. The procedure is iterated until the components of $\Delta \mathbf{q}$ are sufficiently small. Implement and solve this method for estimating Q.

○ *8.9* Show that Step 3 of MOTSTRUCT_FROM_FLOW works equally well even if the epipole is at infinity.

Projects

● *8.1* Implement a coarse-to-fine version of CONSTANT_FLOW. Build the coarser levels by iteratively averaging over the four neighboring pixels at the immediately finer levels. Start CONSTANT_FLOW at the coarsest level and propagate the estimates to each corresponding group of four pixels at the finer level. Compare the results with a standard implementation of CONSTANT_FLOW. Why should the two methods differ? Which method is expected to perform better in the presence of large image displacement?

● *8.2* Implement FEATURE_POINT_MATCHING and MOTSTRUCT_FROM_FEATS. Use the set of matched features, output of the former, as input of the latter.

● *8.3* Implement CONSTANT_FLOW and MOTSTRUCT_FROM_FLOW. Use the optical flow estimates, output of the former, as input of the latter.

● *8.4* Implement a tracking system based on the Kalman filter, and use it to track a moving object viewed from a fixed camera. Use the centroid of the largest set of connected pixels in which a change is detected, and measure the object's position through CHANGE_DETECTION. Let the centroid position and velocity be the system state. Devise a simple data association algorithm to deal with the possibility of detecting changes in more than one large set of connected pixels.

References

[1] Y. Bar-Shalom (ed.), *Multi-Target Multi-Sensor Tracking*, Artec House (1990).

[2] J.L. Barron, D.J. Fleet, and S. Beauchemin, Performance of Optical Flow Techniques, *International Journal of Computer Vision*, Vol. 12, pp. 43–77 (1994).

[3] M. Campani and A. Verri, Motion Analysis from First Order Properties of Optical Flow. *CVGIP: Image Understanding*, Vol. 56, pp. 90–107 (1992).

[4] W.S. Cooper, Use of Optimal Estimation Theory, in Particular the Kalman Filter, in Data Analysis and Signal Processing, *Review of Scientific Instrumentation*, Vol. 57, pp. 2862–2869 (1986).

[5] O.D. Faugeras, F. Lustman, and G. Toscani, 3-D Structure from Point and Line Matches, *Proc. 1st International Conference on Computer Vision*, London (UK), pp. 25–34 (1987).

[6] D.J. Heeger and A.D. Jepson, Subspace Methods for Recovering Rigid Motion I: Algorithm and Implementation, *International Journal of Computer Vision*, Vol. 7, pp. 95–117 (1992).

[7] E.C. Hildreth, *The Measurement of Visual Motion*, MIT Press, Cambridge (MA), (1984).

[8] B.K.P. Horn and B.G. Schunck, Determining Optical Flow, *Artificial Intelligence*, Vol. 17, pp. 185–203 (1981).

[9] Y.Z. Hsu, H.-H. Nagel, and G. Rekers, New Likelihood Test Methods for Change Detection in Image Sequences, *Computer Vision, Graphics, and Image Processing*, Vol. 26, pp. 73–106 (1984).

[10] M. Irani, B. Rousso, and S. Peleg, Computing Occluding and Transparent Motions, *International Journal of Computer Vision*, Vol. 12, pp. 5–16 (1994).

[11] H.C. Longuet-Higgins and K. Prazdny, The Interpretation of a Moving Retinal Image, *Proc. Royal Soc. London B*, Vol. 208, pp. 385–397 (1980).

[12] B.D. Lucas and T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, *Proc. 7th International Joint Conference on Artificial Intelligence*, Vancouver (CA), pp. 674–679 (1981).

[13] P.S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. I, Academic Press, New York (1979).

[14] H.-H. Nagel, Displacement Vectors Derived from 2nd Order Intensity Variations in Image Sequences, *Computer Vision, Graphics, and Image Processing*, Vol. 21, pp. 85–117 (1983).

[15] T. Poggio, J.J. Little, E. Gamble, Parallel Optical Flow, *Nature*, Vol. 301, pp. 375–378 (1986).

[16] J.H. Rieger and D.T. Lawton, Processing Differential Image Motion, *Journal of the Optical Society of America A*, Vol. 2, pp. 354–359 (1985).

[17] C. Tomasi and T. Kanade, Shape and Motion from Image Streams under Orthography: a Factorization Method, *International Journal of Computer Vision*, Vol. 9, pp. 137–154 (1992).

[18] S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge (MA) (1979).

[19] A. Verri and T. Poggio, Motion Field and Optical Flow: Qualitative Properties, *IEEE Transactions on Pattern Analysis Machine Intelligence*, Vol. 11, pp. 490-498 (1989).

[20] T. Viéville and O.D. Faugeras, Feed-Forward Recovery of Motion and Structure from a Sequence of 2D-lines Matches, *Proc. 3rd International Conference on Computer Vision*, Osaka (Japan), pp. 517–520 (1990).

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.