Chapter 8

Stereo Vision

LYNNE L. GREWE and AVINASH C. KAK

Robot Vision Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, Indiana

HANDBOOK OF PATTERN RECOGNITION
AND IMAGE PROCESSING: COMPUTER VISION

1. INTRODUCTION

One way in which humans perceive depth is through a process called *binocular stereopsis* or *stereo vision*. Stereo vision uses the images viewed by each eye to recover the depth information in the scene. A point in the scene is projected into different locations in each eye, and this difference is called the *disparity*. Using geometric relationships between the eyes and the computed disparity value, the depth of the scene point can be calculated. Stereo vision, as used in computer vision systems, is similar.

While there are other techniques known for recovering depth, stereo vision has the advantage of being a passive technique meaning that an active sensor is not required. Historically, stereo vision has been used in many areas including cartography, industrial object recognition, and mobile robot navigation.

The stereo vision process can be summarized by the following steps: (1) detection of features in each image, (2) matching of features between the images under certain geometric and other constraints, and (3) calculation of depth using the disparity values and the geometric parameters of the imaging configuration. While each of these steps is important in the stereo vision process, the matching of features is generally thought to be the most difficult step and can easily become the most time-consuming.

In the sections to come, we hope to inform the reader of the vast work that has been accomplished in the area of stereo vision. We begin in Section 2 by discussing the geometry of a camera and how a scene point is projected into a camera. In Section 3, we describe the imaging configuration used in a stereo vision system and how the difference in the projected locations of a scene point yields a measurement called *disparity*. Also, we discuss how the depth of a point can be calculated from its disparity measurement. Section 4 contains the main contribution of this chapter. Here we describe the different stereo vision systems that have been developed paying particular attention to the issues of feature detection and matching. To help the reader interpret the differences in many of the existing stereo vision algorithms, we have classified them into the following three categories: low-level-feature-based, high-level-feature-based, and object-level-feature-based. In Section 5, we discuss resolution issues and the factors that influence the selection of the various parameters that characterize a stereo vision system. The topic of Section 6 is trinocular stereo vision, where instead of two cameras, three are used in an attempt to improve the results obtained by a binocular system. The effects of scene illumination are discussed in Section 7. Finally, in the last section, Section 8, the topic of error analysis in stereo vision is discussed.

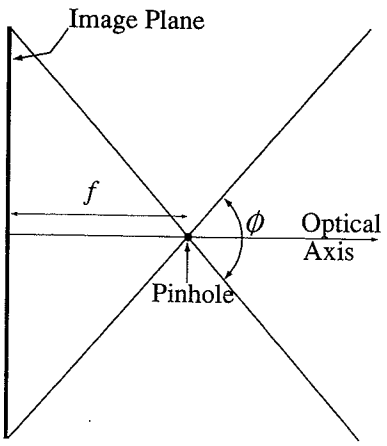

Figure 8.1. Illustration of the dependence of the view angle ϕ on the focal length f.

2. OPTICAL CONSIDERATIONS

Before discussing stereo vision itself, we first describe how the imaging process works in a camera. The process involved in capturing a scene point in the image is referred to as *perspective projection* and is mathematically described by what is called a *perspective transform*. In this section, we explain perspective projection and we also discuss a method of camera calibration that yields the parameters involved in this projection.

2.1. About the Camera Lens

A camera unit consists of two major components: the sensor plane and the lens. A typical sensor plane is 5.0×3.75 cm in size. The lens used is characterized by its focal length or a range of focal lengths if it is a zoom lens. As shown in Fig. 8.1, for a given sensor (image) plane size, the focal length f will determine the view angle ϕ. This angle determines the region of the scene in the field of view.

Another characteristic of a camera is its depth of field. The depth of field is defined as the range over which objects are in focus, meaning that their details will be sharply captured in the image plane. The depth of field is usually shallower in extent in the foreground and deeper in the background. The depth of field depends on the distance o from the lens center to the point at which the camera is focused and the size of the aperture as follows:

$$\text{Depth of field} = \frac{2HN(m+1)}{m^2},$$

where

H = permissible diameter of the circle of confusion
N = f-stop (aperture) number of the lens = f/A, where A is the diameter of the aperture
m = magnification = i/o, where i is the distance between the lens plane and the image plane

A typical value for H is $f/1000$. Note that o, i and f are related by the following equation:

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f}.$$

2.2. Perspective Projection

We model the imaging process through the use of the pinhole model of a camera, shown in Fig. 8.2. This model treats the camera lens as a pinhole at a distance of f, the focal length, along the optical axis from the center of the image plane. The line that is drawn through the image plane's center (u_o, v_o) and is perpendicular to the image plane is referred

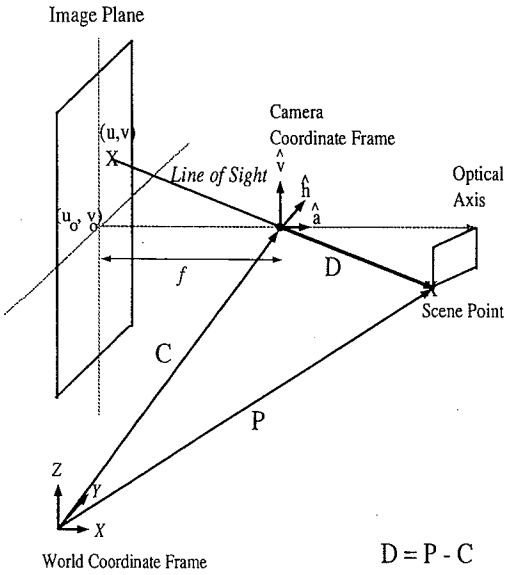

Figure 8.2. Perspective projection using the pinhole model of a camera.

to as the *optical axis*. The camera's 3D location is denoted by the vector C, which represents the world coordinates of the camera's focal point. The camera's orientation is described by the three vectors of the camera coordinate frame denoted by $\hat{h}$, $\hat{v}$, and $\hat{a}$. The image plane has two axes denoted by u and v that are parallel to the $\hat{h}$ and $\hat{v}$ vectors of the camera coordinate frame, respectively. Notice that the $\hat{a}$ vector is coincident with the optical axis.

In terms of $\hat{h}$, $\hat{v}$, and $\hat{a}$, any point on the image plane, (u, v), can be described in world coordinates as follows:

$$C - f\hat{a} + u\hat{h} + v\hat{v}.$$

We derive the perspective transformation equations for a given scene point P and its corresponding image point (u, v) by comparing the appropriate similar triangles in Fig. 8.2. The following is obtained:

$$\frac{u}{f} = \frac{D \cdot \hat{h}}{D \cdot \hat{a}}, \qquad \frac{v}{f} = \frac{D \cdot \hat{v}}{D \cdot \hat{a}},$$

where $D = P - C$ is the vector from the focal point to the scene point. An image is represented in a discrete domain in terms of pixels. Thus we now rewrite the above equation in terms of pixel coordinates, (i, j), instead of the continuously valued points (u, v). We know that the following is true where δu is the horizontal sampling interval and δv is the vertical sampling interval:

$$u = (i - i_o)\delta u, \qquad v = (j - j_o)\delta v.$$

Note that i_o, j_o is the center position of the image in pixel coordinates. We can rewrite the perspective transformation equations as follows:

$$\frac{(i - i_o)\delta u}{f} = \frac{D \cdot \hat{h}}{D \cdot \hat{a}}, \qquad \frac{(j - j_o)\delta v}{f} = \frac{D \cdot \hat{h}}{D \cdot \hat{a}},$$

therefore

$$\frac{i}{f} = \frac{D \cdot H}{D \cdot \hat{a}}, \qquad \frac{j}{f} = \frac{D \cdot V}{D \cdot \hat{a}}, \tag{1}$$

where

$$H = \frac{f}{\delta u}\hat{h} + i_o\hat{a}, \qquad V = \frac{f}{\delta v}\hat{v} + j_o\hat{a}.$$

We can rewrite Eq. (1) in matrix form as follows:

$$\begin{bmatrix} i' \\ j' \\ w \end{bmatrix} = \begin{bmatrix} H_x & H_y & H_z & 0 \\ V_x & V_y & V_z & 0 \\ a_x & a_y & a_z & 0 \end{bmatrix} \begin{bmatrix} D_x \\ D_y \\ D_z \\ 1 \end{bmatrix} \quad \text{and} \quad i = \frac{i'}{w}, \quad j = \frac{j'}{w}. \tag{2}$$

Given $D = P - C$, Eq. (2) becomes

$$\begin{bmatrix} i' \\ j' \\ w \end{bmatrix} = \begin{bmatrix} H_x & H_y & H_z & -C'_x \\ V_x & V_y & V_z & -C'_y \\ a_x & a_y & a_z & -C'_z \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = T \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}, \tag{3}$$

where

$$C' = RC \qquad \text{and} \qquad R = \begin{bmatrix} H_x & H_y & H_z \\ V_x & V_y & V_z \\ a_x & a_y & a_z \end{bmatrix}$$

The pinhole model is accurate for fixed-focal-length cameras with high-quality lenses. However, this model breaks down when a thick or poor-quality lens is used. There are other methods of modeling a camera and, if appropriate, they should be used (25).

2.3. Camera Calibration

As we will see later, the parameters of a camera, C, $\hat{h}$, $\hat{v}$, $\hat{a}$, f, δu, δv, i_o, and j_o can be calculated if the matrix T of Eq. (3) is known. In this section, we will discuss a camera calibration procedure that calculates the

matrix T. The matrix T, commonly referred to as the *camera calibration matrix* or *perspective transformation matrix*, transforms a point in world coordinates (x_m, y_m, z_m) to the corresponding point (i_m, j_m) in the image plane:

$$\begin{bmatrix} i'_m \\ j'_m \\ w_m \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix}, \quad \text{where} \quad i_m = \frac{i'_m}{w_m}, \quad j_m = \frac{j'_m}{w_m}. \tag{4}$$

Therefore, we have 12 unknowns (T_{ij}'s), which can be calculated if the camera is shown at least six scene points whose 3D world coordinates are known and whose corresponding image coordinates can be found. Each of these correspondences will generate two constraints on T as described in Eq. (1). Thus, a total of 12 equations will be generated. The form of these equations can be expressed as follows:

$$i_m = \frac{T_{11}x_m + T_{12}y_m + T_{13}z_m + T_{14}}{T_{31}x_m + T_{32}y_m + T_{33}z_m + T_{34}}, \tag{5}$$

$$j_m = \frac{T_{21}x_m + T_{22}y_m + T_{23}z_m + T_{24}}{T_{31}x_m + T_{32}y_m + T_{33}z_m + T_{34}}. \tag{6}$$

To create a solution from this set of equations we arbitrarily set T_{34} equal to one (this can be thought of as merely a scaling factor). Therefore, Eqs. (5) and (6) can be rewritten as follows:

$$T_{11}x_m + T_{12}y_m + T_{13}z_m + T_{14} + T_{31}(-i_m x_m) + T_{32}(-i_m y_m) + T_{33}(-i_m z_m) = i_m, \tag{7}$$

$$T_{21}x_m + T_{22}y_m + T_{23}z_m + T_{24} + T_{31}(-j_m x_m) + T_{32}(-j_m y_m) + T_{33}(-j_m z_m) = j_m. \tag{8}$$

Thus, given N correspondences we can write the constraints produced

from Eqs. (7) and (8) in matrix form:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -i_1x_1 & -i_1y_1 & -i_1z_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -j_1x_1 & -j_1y_1 & -i_1z_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N & y_N & z_N & 1 & 0 & 0 & 0 & 0 & -i_Nx_N & -i_Ny_N & -i_Nz_N \\ 0 & 0 & 0 & 0 & x_N & y_N & z_N & 1 & -j_Nx_N & -j_Ny_N & -j_Nz_N \end{bmatrix}$$

$$\times \begin{bmatrix} T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{21} \\ T_{22} \\ T_{23} \\ T_{24} \\ T_{31} \\ T_{32} \\ T_{33} \end{bmatrix} = \begin{bmatrix} i_1 \\ j_1 \\ i_2 \\ j_2 \\ \vdots \\ i_N \\ j_N \end{bmatrix}. \tag{9}$$

or, equivalently, by the shorter form

$$\mathbf{AU} = \mathbf{B},$$

where **A** is the $2N \times 11$ matrix shown at the left above, **U** the vector of unknowns and **B** the vector of known pixel coordinates. If N is greater than 6, we have an overdetermined system of equations, and thus an optimal solution can be determined. A common optimization goal with such equations is to find a solution that minimizes the squared error. The squared error involved is defined as $E = [AU - B]^T[AU - B]$. Minimization of this error is equivalent to solving the *normal equations*, $A^TAU = A^TB$. The resulting solution is

$$U = (A^TA)^{-1}A^TB.$$

This solution exists if A^TA is invertible, which is the case when A has linearly independent columns. Consequently, this stipulates that the world points (x_i, y_i, z_i) do not lie on a single plane.

Other techniques for solving for U, including using the QR decompositions of A and the pseudoinverse method with subsequent nonlinear optimization, are discussed in Lopez-Abadia and Kak (25).

2.3.1. Calculation of Camera Parameters from T

Using the matrix T, the camera parameters can be calculated. The relationships between the elements of T and the parameters can be shown to be

$$T_{11} = \frac{(f/\delta u)h_x + i_o a_x}{-C \cdot \hat{a}}, \tag{10a}$$

$$T_{12} = \frac{(f/\delta u)h_y + i_o a_y}{-C \cdot \hat{a}}, \tag{10b}$$

$$T_{13} = \frac{(f/\delta u)h_z + i_o a_z}{-C \cdot \hat{a}}, \tag{10c}$$

$$T_{14} = -\frac{(f/\delta u)C \cdot \hat{h} + i_o C \cdot \hat{a}}{-C \cdot \hat{a}}, \tag{10d}$$

$$T_{21} = \frac{(f/\delta v)v_x + j_o a_x}{-C \cdot \hat{a}}, \tag{10e}$$

$$T_{22} = \frac{(f/\delta v)v_y + j_o a_y}{-C \cdot \hat{a}}, \tag{10f}$$

$$T_{23} = \frac{(f/\delta v)v_z + j_o a_z}{-C \cdot \hat{a}}, \tag{10g}$$

$$T_{24} = -\frac{(f/\delta u)C \cdot \hat{v} + j_o C \cdot \hat{a}}{-C \cdot \hat{a}}, \tag{10h}$$

$$T_{31} = \frac{a_x}{-C \cdot \hat{a}}, \tag{10i}$$

$$T_{32} = \frac{a_y}{-C \cdot \hat{a}}, \tag{10j}$$

$$T_{33} = \frac{a_z}{-C \cdot \hat{a}}, \tag{10k}$$

$$T_{34} = 1. \tag{10l}$$

These equations have all been scaled by T_{34} to account for the assumption that $T_{34} = 1$ (Eq. 10l). A procedure to recover the parameters from Eqs.

(10a–l) was developed by Ganapathy (10). This method actually solves for C, $\hat{h}$, $\hat{v}$, $\hat{a}$, i_o, j_o, $f/\delta u$, and $f/\delta v$. If the sampling intervals (δu and δv) are known then f can be recovered. These sampling intervals are typically given by the camera's manufacturer. In the following derivations, T_i, denotes $[T_{i1}T_{i2}T_{i3}]$. Now, using Eqs. (10a–c) the following is obtained:

$$T_1 \cdot T_1 = \left(\frac{f/\delta u}{-C \cdot \hat{a}}\right)^2 + \left(\frac{i_o}{-C \cdot \hat{a}}\right)^2. \tag{11}$$

Using the fact that $\hat{h}$, $\hat{v}$ and $\hat{a}$ are mutually orthogonal we can similarly obtain

$$T_2 \cdot T_2 = \left(\frac{f/\delta v}{-C \cdot \hat{a}}\right)^2 + \left(\frac{j_o}{-C \cdot \hat{a}}\right)^2, \tag{12}$$

$$T_3 \cdot T_3 = \left(\frac{1}{-C \cdot \hat{a}}\right)^2, \tag{13}$$

$$T_1 \cdot T_3 = \left(\frac{i_o}{(-C \cdot \hat{a})^2}\right), \tag{14}$$

$$T_2 \cdot T_3 = \left(\frac{j_o}{(-C \cdot \hat{a})^2}\right). \tag{15}$$

From Eq. (13) we can compute $C \cdot \hat{a}$, and using this and Eqs. (14) and (15), we can compute i_o and j_o. In addition, using Eqs. (10i–k), the values for a_x, a_y, a_z, and $\hat{a}$ can be determined.

Now, Eqs. (11) and (12) are used to find the magnitudes of $f/\delta u$ and $f/\delta v$. We will now show how the signs of $-C \cdot \hat{a}$, $f/\delta u$ and $f/\delta v$ can be determined. We can set $-C \cdot \hat{a}$ and $f/\delta u$ to have positive signs, and then determine the appropriate sign of $f/\delta v$. Notice that the signs of $f/\delta u$ and $f/\delta v$ depend on the polarity of the axes of the camera coordinate frame; that is, the directions in which the row and column numbers of the pixels (i, j) increase. Given that we have assumed a positive sign for both $-C \cdot \hat{a}$ and $f/\delta u$, we will assume for a moment that $f/\delta v$ is positive. Given that we know $-C \cdot \hat{a}$, $f/\delta u$, i_o, j_o, a_x, a_y, a_z, and $\hat{a}$, we can determine h_x, h_y, and h_z from Eqs. (10a), (10b), and (10c), respectively. Similarly, v_x, v_y, and v_z can be determined from Eqs. (10e), (10f), and (10g), respectively. Also, $\hat{v}$ can be obtained by using the relationship $\hat{v} = \hat{a} \times \hat{h}$. If the two vectors $\hat{v}$ obtained by the different methods are the same then the choice of a positive sign for $f/\delta v$ was correct otherwise the sign must be reversed.

The components of the vector C can be recovered with the use of Eqs. (10d) and (10h). As mentioned above, the value of f can be calculated if δu or δv is known.

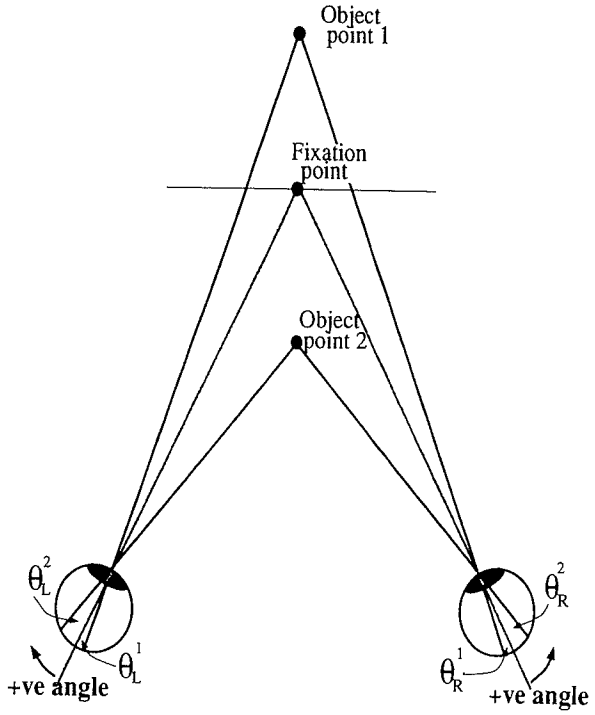

Figure 8.3. Object point 1 posses divergent disparity $d = -\theta_L^1 + -\theta_R^1$ since it lies behind the fixation point whereas object point 2 has convergent disparity, $d = \theta_L^2 + \theta_R^2$.

3. OVERVIEW OF THE STEREO VISION PROCESS

Binocular stereo vision or stereopsis is the process of matching features in one image with the corresponding features in the other image and then using these matches to derive the depth information in the scene being viewed. Figure 8.3 depicts the human stereo vision system. At any one moment a person's perception is centered on one point in space, called the *fixation point*. More specifically, the fixation point is the intersection of the two optical axes for the two eyes. Projection of a scene point into each eye's retina takes place as described in Section 2. As shown in Fig. 8.3, an object point will be projected into each eye's retina in a different position with respect to the directions of positive and negative movement from the center point. This leads to the definition of disparity, which is the difference between these positions. From the disparity the depth of the object point can be calculated by effectively inverting the projection process for each eye. When an object point is nearer than the fixation point, the disparity is called *convergent* or *crossed* and will be of a sign opposite to the disparity value produced by an object point behind the fixation point; the latter disparity is referred to as the *divergent* or *uncrossed disparity*. The term *crossed* refers to the fact that the optical axes of the two eyes or cameras will have to cross or converge further to fixate on the object point.

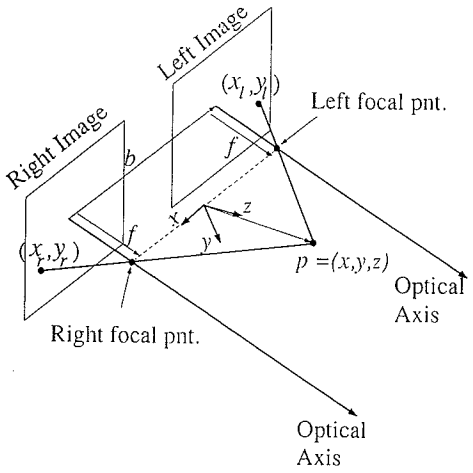

Figure 8.4. The canonical stereo camera configuration where the point p is projected to (x_l, y_l) in the left image plane and (x_r, y_r) in the right-image plane.

In Fig. 8.4, we show a point being projected into the image planes of a pair of stereo cameras. In this configuration, the optical axes are parallel and are also perpendicular to the line (baseline b) connecting the center of the image planes. In addition, the baseline is coincident with the horizontal (x axis) axis of each image plane, meaning that the two camera images are row registered; i.e., the nth row of the left image is collinear with the nth row of the right image. We will henceforth refer to this configuration as the canonical configuration. While other configurations can also be used, we will initially assume this one for convenience in our discussion of depth calculation. In any camera configuration, the disparity of a projected scene point is defined as the difference between the projected locations in the two image planes.

It is fairly trivial to compute depth from a disparity measurement given that a few parameters of the stereo camera setup are known. In our canonical camera configuration, shown in Fig. 8.4, notice that the origin of the world coordinate frame is located on the baseline half-way between the image centers. From the geometry of the projections, it follows trivially

$$\frac{x_l}{f} = \frac{x + b/2}{z}, \qquad \frac{x_r}{f} = \frac{x - b/2}{z}, \qquad \frac{y_l}{f} = \frac{y_r}{f} = \frac{y}{z}, \tag{16}$$

where b is the length of the baseline connecting the two image centers and f is the focal length of each camera. These equations lead to the following

expressions for the recovery of the coordinates of the scene point:

$$x = \frac{b(x_l + x_r)/2}{(x_l - x_r)}, \qquad y = \frac{b(y_l + y_r)/2}{(x_l - x_r)}, \qquad z = \frac{bf}{(x_l - x_r)} \tag{17}$$

where the disparity is equal to $(x_l - x_r)$. Because we are using the canonical camera configuration, a scene point will project into each image so that the only difference in location is along the x axis. It is interesting to note that depth (the z component) is inversely proportional to the disparity and is proportional to both b, the baseline length, and f, the focal length. In Section 5, we will discuss these relationships and how they affect the design of an appropriate stereo camera configuration.

The geometry of the camera configuration influences not only how the depth is calculated but also the efficiency of search for establishing correspondences between the image features from the two cameras. Many systems use what is called the *epipolar constraint*, which states that given an object point p and its projection in the left image p_l, then the corresponding right image point p_r must be located on the corresponding epipolar line. As shown in Fig. 8.5, the epipolar line is formed by the intersection of the epipolar plane with the right image plane. The epipolar plane is defined as the plane that passes through the points p_l, C_l, and C_r. C_l, and C_r are the focal points of the left and right images, respectively. This constraint is a direct result of the geometry of perspective projection.

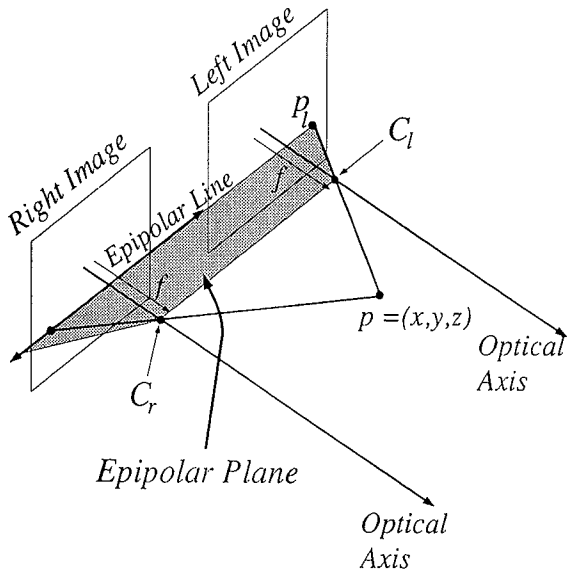

Figure 8.5. Illustration of epipolar plane and line. The corresponding right-image point can lie anywhere along the epipolar line.

It is interesting to note that the epipolar lines for a canonical camera configuration fall on horizontal scan lines of the images.

While a canonical camera configuration is nice for describing how depth values are calculated, its use may not be feasible for a particular set of system requirements. For example, the depth of field required in a scene at a certain depth z can result in the necessary use of a noncanonical camera configuration. An example of a noncanonical, or convergent, system is shown in Fig. 8.6. Generally, in a convergent system the cameras are set up so that the cameras are symmetrically oriented with respect to the y axis in Fig. 8.6. Here, for any scene point, the epipolar line will not fall on a horizontal scan line of the image. Therefore, the search space becomes complicated. In such cases, it is best to reproject the camera images, or the feature points extracted therefrom, into planes that would correspond to a canonical configuration, as illustrated in Fig. 8.7. This reprojection process, referred to as rectification in the literature, described below for each image, uses the position of the focal points of the cameras in 3D space (C), and the perspective transformation matrices T. Both the matrices and the focal point positions are results of the camera calibration procedure of Section 2. The goal is to reproject each image plane so that it is coplanar with the other one and also is parallel to the line that connects the two camera centers (focal points) in order to obtain epipolar lines that are parallel to the horizontal axis of each image. In addition, the focal points of the cameras should not move. The derivation of this method can be found in Ayache and Hansen (1). We will denote the transformation matrices of the two cameras as T_1 and T_2 and similarly the focal points as C_1 and C_2.

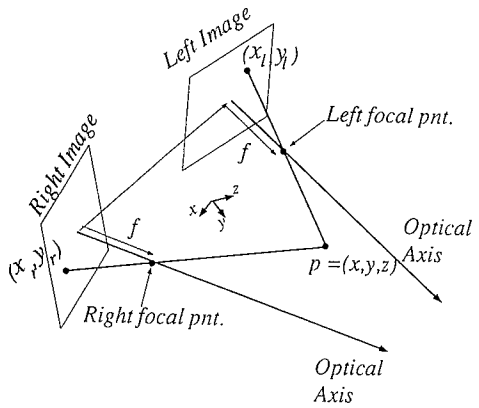

Figure 8.6. A stereo camera configuration with nonparallel optical axes.

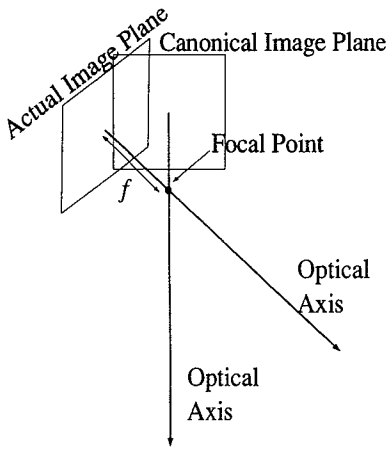

Figure 8.7. Illustration of the actual camera's rotation from a canonical camera.

For each point $p = (u, v)$ in image i, $p' = (u', v')$ is computed as follows:

$$\begin{bmatrix} a \\ b \\ w \end{bmatrix} = R_i \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad \text{and} \quad u' = \frac{a}{w}, \quad v' = \frac{b}{w},$$

where the following are 3×3 matrices

$$R_1 = \begin{bmatrix} [(C_1 \times C_2) \times C_1]^t \\ (C_1 \times C_2)^t \\ [(C_1 - C_2) \times (C_1 \times C_2)]^t \end{bmatrix} \cdot \begin{bmatrix} t_2 \times t_3 \\ t_3 \times t_1 \\ t_1 \times t_2 \end{bmatrix}^t,$$

where t_i denotes the ith row of the matrix T_1

$$R_2 = \begin{bmatrix} [(C_1 \times C_2) \times C_2]^t \\ (C_1 \times C_2)^t \\ [(C_1 - C_2) \times (C_1 \times C_2)]^t \end{bmatrix} \cdot \begin{bmatrix} t_2 \times t_3 \\ t_3 \times t_1 \\ t_1 \times t_2 \end{bmatrix}^t,$$

where t_i denotes the ith row of the matrix T_2

As a result of this procedure on each image, the newly created images have epipolar lines that lie on the horizontal scan lines of each image. In addition, the reprojected images are row registered, meaning that the nth row of one image corresponds to the nth row of the other image.

In the next section, we will discuss in detail various types of stereo vision algorithms.

4. PARADIGMS FOR STEREOPSIS

Historically, research in stereo vision can be described by two paradigms, which we will refer to as *low-level-feature-based stereo vision* and *high-level-feature-based stereo vision*. Initially, it was believed that recognition of high-level features such as straight-line edges in each of the two images must first be accomplished and then binocular matching of these features would take place. This belief is the foundation of what we call *high-level-feature-based stereo*. Then in 1960, Julez (18) performed a series of random-dot stereogram experiments that revolutionized the field and led to a new paradigm for modeling stereo vision, which we refer to as *low-level-feature-based stereo*. These psychophysical experiments indicated that humans in fact did not have to produce monocular cues before binocular fusion could take place. Subsequently, Marr and Poggio (26) advanced a computational theory, later implemented by Grimson (11, 12), that explained the observations made by Julez.

Various researchers and different psychophysical experiments have separately supported each of the paradigms, and as such each paradigm can be thought of as explaining different aspects of vision. The low-level-feature-based paradigm leads to a bottom–up procedure where the system starts with low-level features. On the other hand, the high-level paradigm results in a top–down or expectation-driven process because high-level features are extracted and this implies the expectation that these high-level features exist in the two images of the scene. Most recently, investigators have been implementing hybrid systems that use both high- and low-level features (23, 44). This we believe is currently the most robust method of achieving good stereo vision performance. Following is a detailed discussion of each paradigm in which the issues of feature extraction and methods of establishing correspondences are emphasized.

4.1. Low-Level-Feature-Based Stereo Vision

"Low" and "high" are subjective descriptors in the English language, and thus there are no set criteria for a feature to be classified as a low- or high-level feature. Tradition dictates that image features that are semantically significant, such as long straight lines or curved lines possessing particular attributes, be called "high-level" and the other semantically

nonsignificant, such as zero-crossings of the derivatives of the image gray levels, be called "low-level." In this section, we will deal with stereo vision using low-level features. We will start out with a discussion of what is certainly the most famous algorithm founded on the low-level paradigm, the Marr–Poggio–Grimson (MPG) algorithm.

4.1.1. The MPG Algorithm

4.1.1.1. Feature Extraction

The Marr–Poggio theory (26) proposes extracting point features by filtering the images with a set of 12 orientation-specific filters where each is represented by the difference of two Gaussian functions and then extracting zero-crossing points. Marr and Hildreth (27) showed that intensity changes occurring at a particular resolution may be detected by locating such zero-crossing points. In Grimson's implementation (11, 12) called the MPG algorithm, a single circularly symmetric Laplacian-of-a-Gaussian (LOG) filter is used. The use of a single filter is not only more computationally efficient, but, as discussed in Mayhew and Frisby (30), there is psychophysical evidence that humans may utilize a single circularly symmetric filter.

The LOG operator is often referred to as a primal sketch operator where a primal sketch can be defined as the representation of an image that makes explicit the information about gray-level variations. As will be illustrated below, the LOG is a smoothed second derivative of the image signal. The LOG operator assumes the following form:

$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2}{\sigma^2} - 2 \right) \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2} \right), \tag{18}$$

where ∇^2 is the Laplacian $\nabla^2 = (\delta^2/\delta x^2) + (\delta^2/\delta y^2)$ and $G(x, y)$ is the Gaussian function, which acts to low-pass-filter the image:

$$G(x, y) = \sigma^2 \exp\left(\frac{-(x^2 + y^2)}{2\sigma^2} \right). \tag{19}$$

Figure 8.8(A) shows the Gaussian function and Fig. 8.8(B), the LOG function. The "width" of the LOG function is the diameter of the circle formed by the ring of zeros of the function; this width is related to σ as follows: $w_{2D} = \sqrt{2}\,\sigma$.

Before continuing, let's observe how a zero-crossing of the second derivative of a signal can indicate a point of gray-level variation. For ease of illustration suppose we have the 1D (one-dimensional) gray-level signal of Fig. 8.9(A). Shown in Figs. 8.9(B) and (C) are the results of taking the

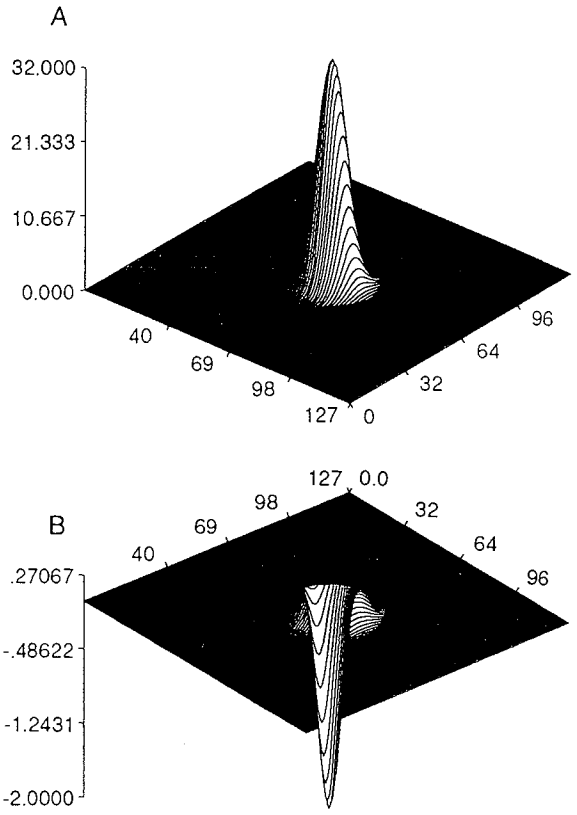

Figure 8.8. (A) 2D Gaussian function at $w_{2D} = 8$; (B) corresponding Laplacian-of-a-Gaussian function (viewed from below the x–y plane).

first and second derivatives. [Actually, the results shown in Figs. 8.9(B) and (C) are obtained by applying discrete approximation to the derivative operators to the discrete signal in Fig. 8.9(A)]. Observe that the zero-crossing of Fig. 8.9(C) is located at the center of the gray-level variation of Fig. 8.9(A). A left-to-right transition from a positive to a negative value at the zero-crossing is referred to as a *positive zero-crossing* and indicates that the corresponding gray-level variation is from low to high. Likewise a negative zero crossing, which is a negative-to-positive transition in values at the zero-crossing, indicates a gray-level variation from high to low.

The LOG function takes the second derivative after low-pass filtering with the Gaussian function, which is characterized by the width w_{2D}. Changing the width of the LOG function in effect captures gray-level variations at different scales or resolutions. Figure 8.10 shows a gray-scale

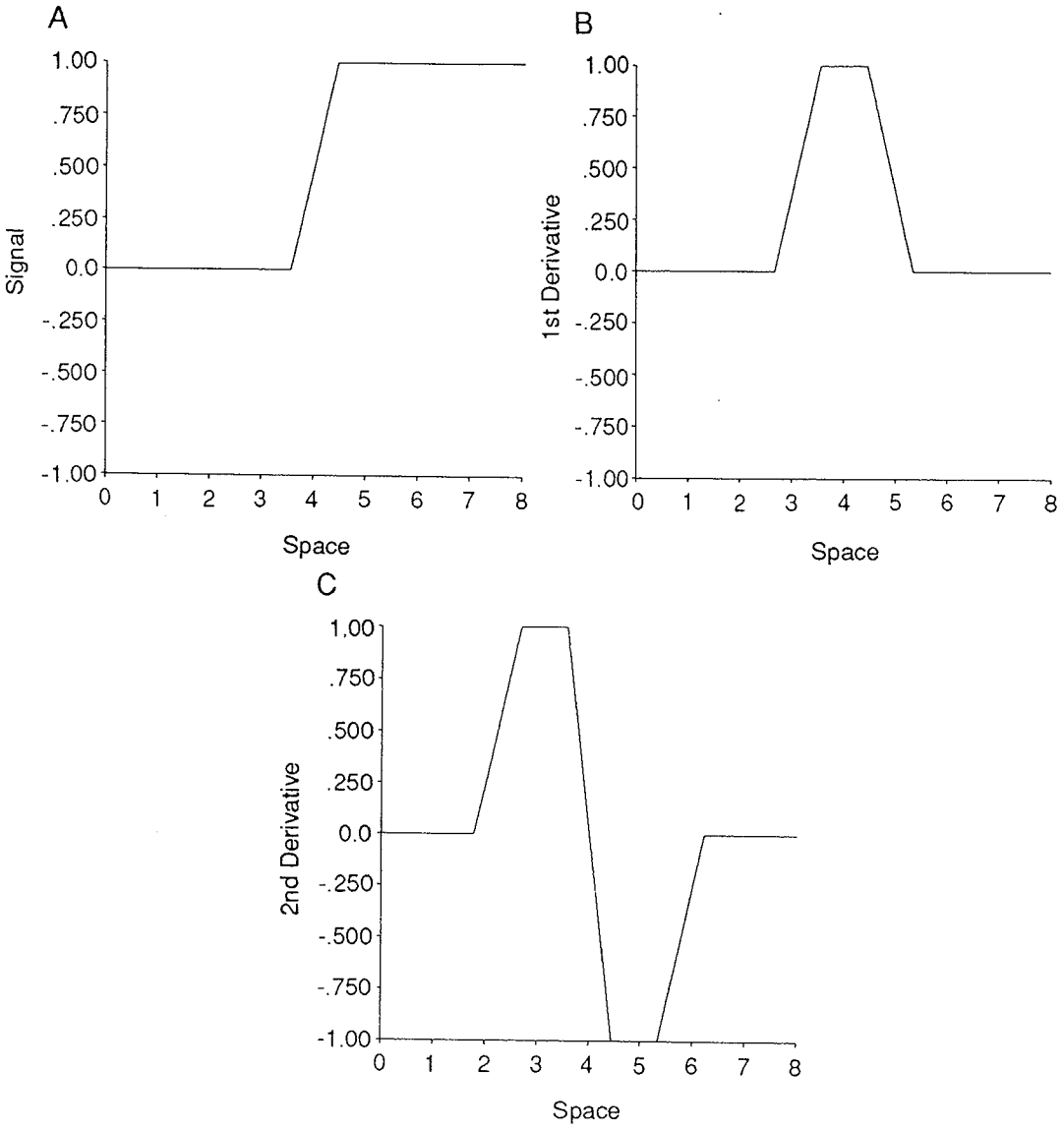

Figure 8.9. (A) Original 1D discrete signal; (B) first derivative of the signal; (C) second derivative of the signal.

image and the results of LOG filtering at different widths. As w_{2D} becomes smaller, finer gray-level variations are retained. The LOG operator is defined in the continuous domain and, theoretically speaking, it extends over the entire xy plane. The domain of the operator must evidently be truncated for computer implementation. It is usual to use only that piece of the LOG operator that is defined over the $2w_{2D} \times 2w_{2D}$ patch, centered at the origin, of the xy plane. This portion of the LOG function, appropriately sampled, will be referred to as the *LOG kernel.*

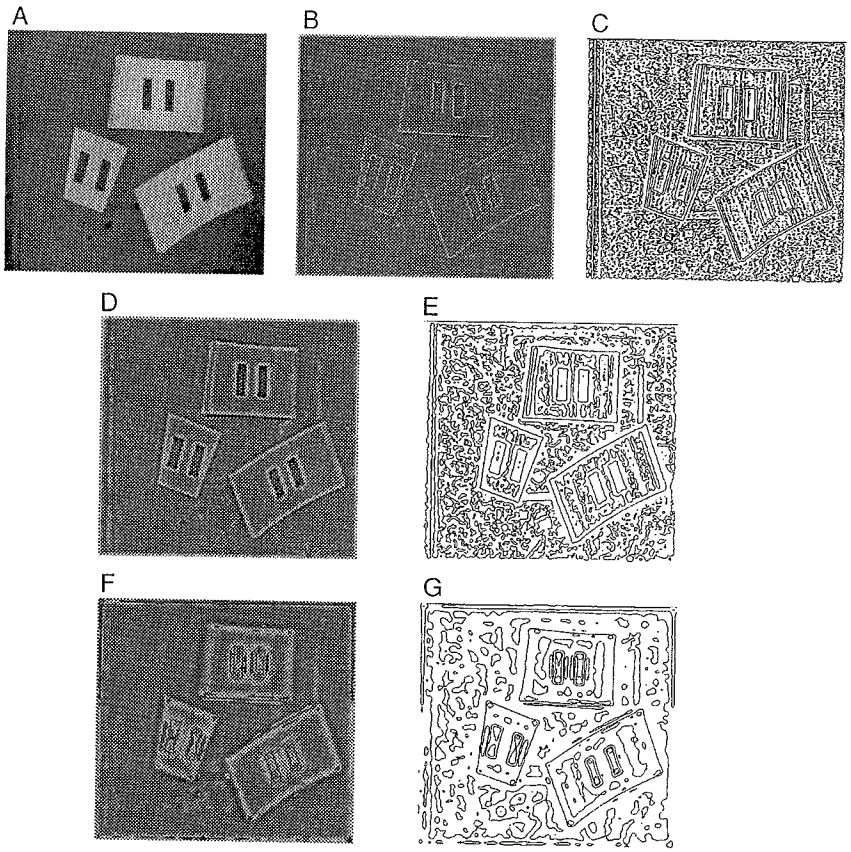

Figure 8.10. (A) Original gray-scale image; (B) image convolved with LOG filter at $w_{2D} = 4$; (C) extracted zero-crossings for $w_{2D} = 4$; (D) image convolved with LOG filter at $w_{2D} = 8$; (E) extracted zero-crossings for $w_{2D} = 8$; (F) images convolved with LOG filter at $w_{2D} = 16$; (G) extracted zero-crossings for $w_{2D} = 16$. Note that all zero-crossing images are displayed by gray-scale values that reflect the strength of the zero-crossing, i.e., indicate the difference in the gray values across the zero-crossing point.

After convolution with the LOG kernel, a simple algorithm such as the following one from Tanaka and Kak (44) may be used to extract the zero-crossings from the LOG filtered data.

```
1) Label all of the positive pixels in the LOG filtered
   image with +1's
2) Label all of the negative pixels in the LOG filtered
   image with 0's
3) The zero-crossing contours are extracted by following
   the boundaries of the positive regions, where the
```

```
    boundaries are defined as the 4-connectedness neighbor
    of negative regions.

During the contour extraction process, label each zero-
crossing as either 'p' or 'n', depending upon whether or
not its immediate-left neighbor is lesser or greater than
its immediate-right neighbor. If one of the neighbors in
this left-right comparison is a boundary pixel than the
zero-crossing is classified as 'o' which stands for
'other'. Notice that 'p' denotes a positive zero-crossing
and 'n' a negative zero-crossing.
```

If subpixel accuracy is desired, the exact location of a zero-crossing can be estimated by interpolating between the positive and negative values on either side of the zero-crossing found above. Subpixel calculation is feasible only in cases where the zero-crossings are fairly isolated.

As discussed by Grimson (13) there is support that the human visual system uses data from five different-sized LOG filters where each application of a LOG filter is commonly referred to as "passing the image through a channel of width w_{2D}." Figure 8.11 illustrates the flow of control in the MPG algorithm and shows the use of five channels with differing widths. The larger-width (coarser) channels capture the larger gray-level variations in the image, whereas the smaller width (finer) channels capture fine variations. The use of multiple channels will be explained in the next subsection on matching.

Besides the classification of a zero-crossing as either positive or negative, a zero-crossing point also has the additional attribute of orientation. It is theoretically known that the output of a LOG filtered image will have zero-crossing points that form continuous contours. Thus, the orientation of a zero-crossing can be defined as the orientation of the locally connected contour of zero-crossings passing through the point in question. Examination of the zero-crossings that are neighbors of the point in question can yield an estimate of the orientation using bit patterns as discussed in Tanaka and Kak (44). Also, the ratio of local Sobel operators applied to the original gray-scale image can be used to find the orientation as described in Kak (20).

4.1.1.2. Matching

Before discussing the multichannel matching algorithm of the MPG process, we will examine how matching is done for one channel independently. As discussed before, the right-image correspondent of a left-image zero-crossing point must lie on the epipolar line of the left image zero-crossing point in question. If we assume a canonical camera configuration, the epipolar line will lie on a horizontal scan line (row) of the right image. Now, assuming that we know an estimate of the average disparity, d_{av}, in

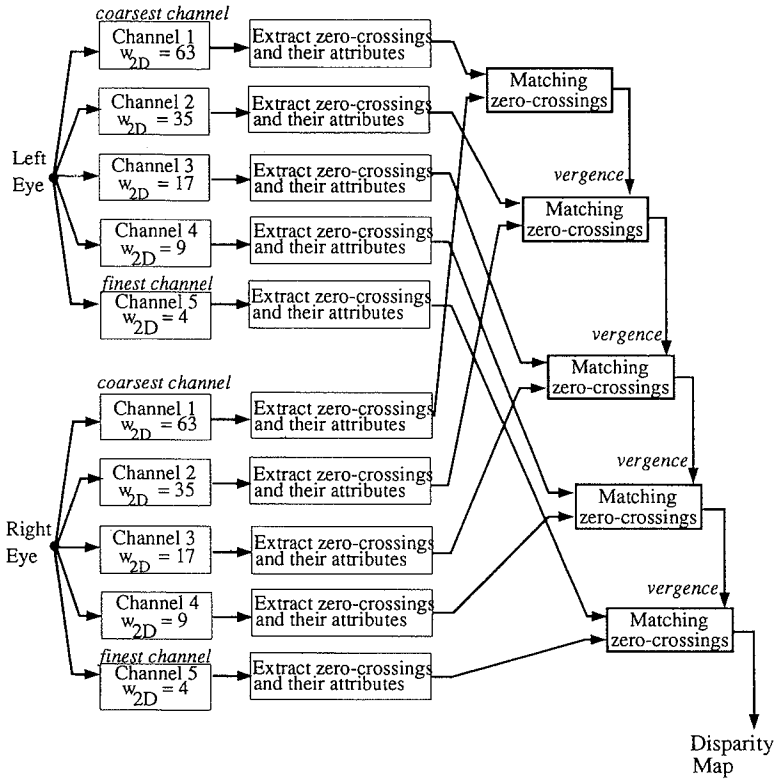

Figure 8.11. Block diagram of Marr–Poggio–Grimson algorithm. In each channel box the image is convolved with a Laplacian-of-a-Gaussian operator of width w_{2D}.

the image then the following steps are performed to find matches for a left-image zero-crossing point, $p = (x_l, y_l)$:

1) Search on the y_l scan line of the right image in a 1D window $\pm w_{2D}$ centered at the point $(x_l + d_{av}, y_l)$ for the possible candidate zero-crossing matches.
2) If a zero-crossing on the window is of the same sign type and is of approximately the same orientation as the left image zero-crossing in question then this zero-crossing produces a match.

See Fig. 8.12 for step 1 above.

It should be clear to the reader at this point that the choice of w_{2D} determines the range of disparities that can be calculated in the scene as

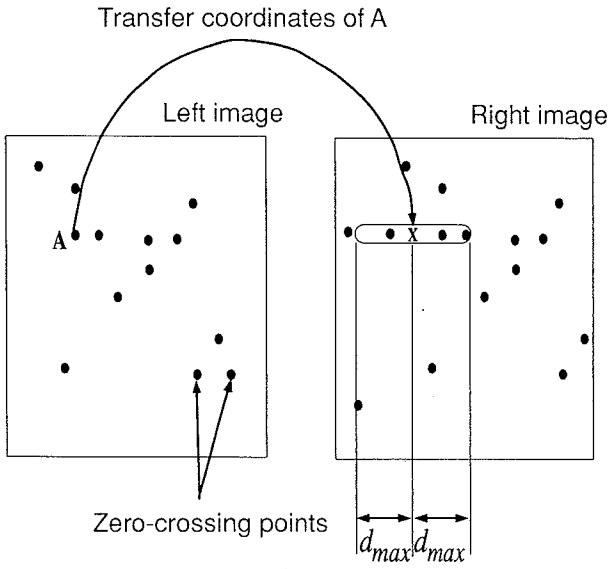

Figure 8.12. A search must take place in the right image to find all of the potential matching points for a left image point. The search space is shown in the figure to span the left-image point's position in the right image by $d_{\max}$ in both directions.

given by the following equation:

$$d_{\min} = d_{\text{av}} - w_{2D} \leq \text{disparity} \leq d_{\text{av}} + w_{2D} = d_{\max}. \qquad (20)$$

Given a camera configuration, if the maximum disparity range in the scene is known, then the w_{2D} that should be used can be calculated. If d_{av} is not known, then w_{2D} must be set to equal the maximum absolute disparity expected in the scene. Note that d_{av} could be an average over all of the image or, if available, a function of the location in the image. An important feature of using the $\pm w_{2D}$ search window is that the number of possible candidates is considerably reduced from the case of searching the entire scan line. The reader should not be misled into thinking that this choice of window size is arbitrary. Marr and Poggio (26) have shown that there is a 95% probability that there will be only one right-image zero-crossing in a window of size $\pm w_{2D}/2$. However, the choice of $\pm w_{2D}/2$ is too restrictive for practical use since, especially for large window sizes, adjacent gray-level variations can cause a shift in the location of a zero-crossing. Figures 8.13(A) and (B) show a pair of stereo images taken from Tanaka and Kak (44), and Fig. 8.14 shows the graphs that illustrate the shifts in the locations of zero-crossings along the PQ scan line for each image in the stereo pair as a function of applying LOG operators with different widths. Observe that there is more variation in the position of the left-image zero-crossing, which is due to the presence of a shadow illumi-

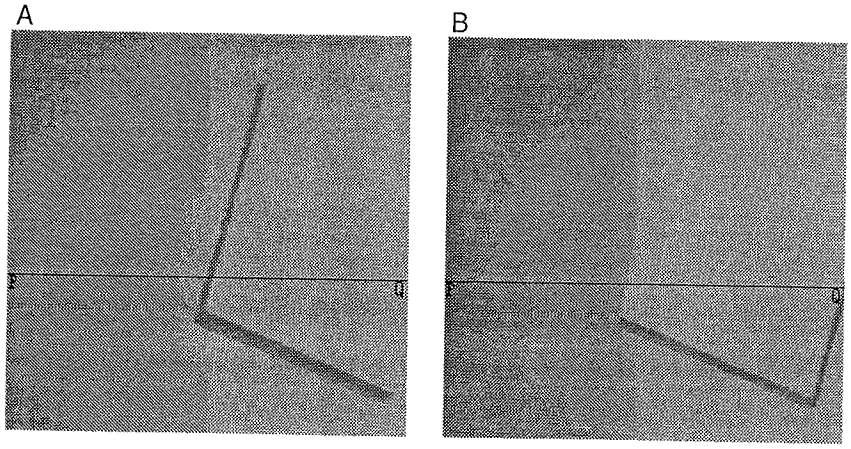

Figure 8.13. Stereo pair of images of a block on a flat background. A PQ scan line is shown across each image: (A) left image; (B) right image.

nation pattern in the left image that does not exist in the right image. This difference in illumination patterns between the images is due to the fact that each camera is at a different viewing position. The size of the search window is usually increased to $\pm w_{2D}$ in order to cope with the difficulties caused by the shifts in the locations of zero-crossings. The disadvantage of expanding the search window is that the probability of finding only a single zero-crossing in this window decreases to 50%, and thus the number of left-image zero-crossings with more than one possible match in the right-image increases.

When the scene objects are opaque, there can be only one right-image correspondent for each left-image zero-crossing. The MPG algorithm assumes this uniqueness constraint. If the reader is interested in techniques that work with transparent objects she is referred to Prazdny (38).

The processing of multiple channels in a coarse to fine sequence using what is called *vergence control* allows for a more accurate determination of the disparity values in the scene. In such multichannel systems, more low-pass filtering is performed in the coarser channels than in the finer channels. As a result, small variations in intensity disappear in the coarser channels so that very accurate disparity calculations are not possible. In addition, a coarse channel will produce a sparse disparity map. In a finer channel, the positions of the zero-crossings will be more accurate since not as much smearing of the intensity variations occurs, and thus a more accurate disparity calculation can be achieved. However, a correct value of d_{av} for the point in question in Eq. (20) is more critical for finer channels because the search window is much smaller than the one used by a coarser channel. This leads to the concept of vergence control, which is an

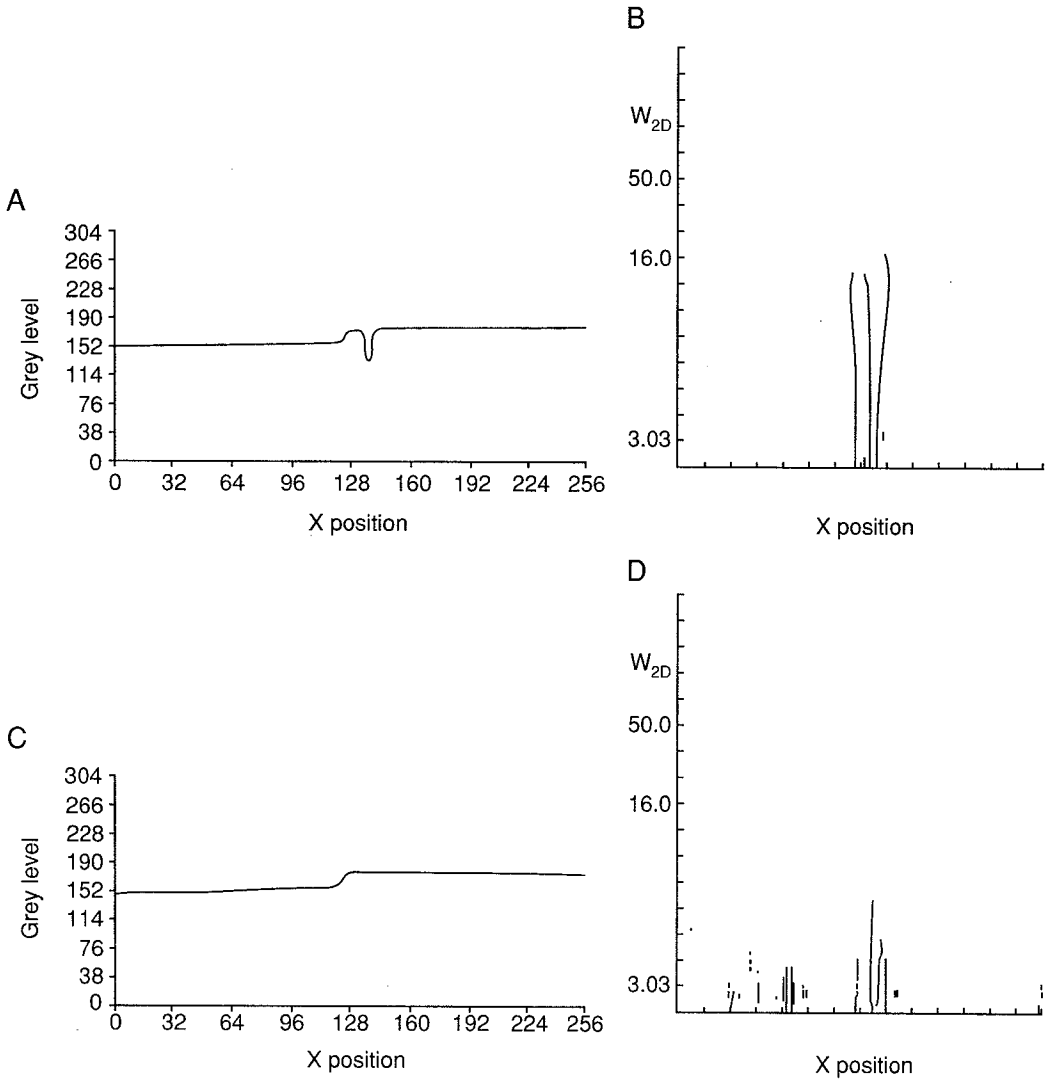

Figure 8.14. (A) Gray-level variations across line PQ in the left image of Fig. 8.13; (B) locations of left-image zero-crossings along PQ for different w_{2D}; (C) gray-level variations across line PQ in the right image of Fig. 8.13; (D) locations of right-image zero-crossings along PQ for different w_{2D}.

implementation of the idea that the disparities calculated from the coarser channels can be used to locally bring regions of the right image into range for appropriate matching to take place with the left image at the next finer channel. More specifically, the disparities from the coarser channels will be used to calculate the average disparity in a neighborhood of the left-image point in question, and then this is used to shift the search window [via d_{av} value of Eq. (20)] to the appropriate position in the right image for matching to occur.

In Grimson's implementation, for the $w_{2D} = 9$ channel a neighborhood of size 25×25 is examined, and if less than 70% of the points in this

region have matches, the region is considered to be out of the range for fusion in this channel. For each region that does not pass this test, a vergence shift must take place to bring the right-image region into the range of fusion for this channel. In another implementation, vergence is used everywhere the coarse channel produces disparity values without first applying a threshold to the number of matches. The disparity calculated for each matched pair of zero-crossings is stored in a buffer commonly called a *disparity map*, or the $2\frac{1}{2}$D sketch. If there exists more than one match at a zero-crossing point, the average of the corresponding disparities is stored. Notice that there is no theoretically correct size for the region that must be examined to determine whether its points are within the range of fusion for the current channel. However, this size should be a function of the density of the zero-crossings in the images, which is dependent on the scene being viewed. For example, in Kak (20) a neighborhood of 10×10 and a threshold of 50% yielded superior results. It is worth noting that vergence control is akin to humans observing a scene in an unfocused manner, forming a crude idea of the locations of the objects in the scene, and then fixating on each object of interest.

Instead of averaging the disparities when multiple matches occur for a particular left-image zero-crossing, a unique match may still be established by what Marr and Poggio (26) have termed as the "pulling effect." The pulling effect consists of enforcing some form of a continuity constraint over the disparities in a region surrounding the left-image zero-crossing in question. The validity of desiring that the disparities be continuous in value in a local region comes from the assumption that the scene consists of object points that vary continuously. There is no theoretically correct choice for the size of the neighborhood to be used for the disambiguation of multiple matches. However, in Kak (20) a 10×10 neighborhood for 256×256 images was used successfully. After disambiguation has taken place, the disparity map is updated.

We will now discuss some important ancillary issues that would be relevant to a modern-day implementation of the MPG algorithm. The first one of these deals with the concept of vertical disparity. Psychophysical experiments have demonstrated that when a pair of stereo images is not row-registered, human subjects are still able to make the correct matches when there is only a small amount of vertical difference (disparity) between the corresponding points. A similar capability may be given to a computer implementation, which has the additional desired effect of correcting for small vertical displacements of the images that can result from less than perfect row registration. This can be accomplished by using two-dimensional search windows spanning a few rows above and below the apparent epipolar line.

Another issue is that of figural continuity that can be used to further constrain the matching between the zero-crossings. Figural continuity is

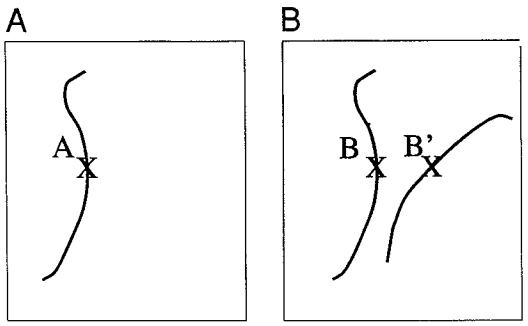

Figure 8.15. The match pair (A, B) will pass the figural continuity constraint but the match pair (A, B') will fail the figural continuity constraint: (A) left image; (B) right image.

derived from the notion that surfaces vary smoothly in a scene and thus disparity values should be continuous (30). Its computer implementation is based on the realization that the contours on a scene surface will project into each image as continuous contours with approximately the same shape. Although the exact nature of the implementation varies from system to system, the common denominator consists of comparing the shapes, say, by comparing chain codes, of the zero-crossing contours. The idea is illustrated in Fig. 8.15. In order for the zero-crossing A in the left image to be considered matchable to the zero-crossing B in the right image, the figural continuity constraint demands that the shape of the zero-crossing contour passing through A be similar to the shape of the zero-crossing contour passing through B. A local shape comparison would reject a match between the point A and the point B' also shown in Fig. 8.15. Mayhew and Frisby (30) have advanced psychophysical evidence to support the conjecture that the stereopsis in the human visual system also uses a figural continuity constraint.

It is interesting to note that the region-based disparity continuity constraint and the figural continuity constraint play a complementary role in the fusion process. For those regions of object surfaces that are away from the boundaries, it makes intuitive sense to use a region-based disparity constraint for disambiguation. However, such a constraint will serve no purpose in the vicinity of depth discontinuities. Since figural continuity constraint is applied only along contours, it is less sensitive to the problems caused by depth discontinuities. The reader's attention is also drawn to a recent contribution by Fleck (9) who has taken scene topology into account and presented a generalization of the figural continuity constraint.

In addition to the region-based disparity and figural continuity constraints, it is also possible to apply what has been called a *disparity gradient constraint* (37). Pollard *et al.* have shown that there is an upper bound to the maximum difference in the disparities of two potential matches as a function of their separation in the image space. Therefore,

potential matches that violate this upper bound can be discarded. The reader is referred to Baunegg (5) for a recent implementation of the MPG algorithm that includes an implementation of the disparity gradient constraint.

4.1.2. Low-Level Features besides Zero-Crossings

At this point, the reader is probably wondering whether low-level features besides the zero-crossings of the LOG output can be used for stereopsis. Mayhew and Frisby (30) claim that, in addition to the zero-crossings, points located at the peaks of the LOG-filtered images are also necessary low-level features. They show psychophysical evidence that humans use these peaks in binocular fusion to perceive the depth in some scenes. If only LOG zero-crossings are used in such scenes, incorrect depth perception results. Shown in Fig. 8.16(A) are the images of a stereo pair. The brightness function along one row of the images is shown in Fig. 8.16(B). The output of the LOG operator for a single channel MPG implementation is shown in Fig. 8.16(C) for each image. Figure 8.16(D) shows the brightness values along one row in each of the images in part (C). Figures 8.16(E) and (F) show the zero crossings and the peaks detected along a single row of the images in (C). When a human subject is shown the stereo pair of Fig. 8.16(A), the perceived depth profile is approximately as shown in Fig. 8.17(A). The depth profile constructed from just the LOG zero-crossings is presented in Fig. 8.17(B), while the profile constructed by using just the peaks of the LOG output are shown in Fig. 8.17(C). Finally, Fig. 8.17(D) shows the depth profile constructed by using both the LOG zero-crossings and peaks. The similarity of this computed depth profile to the perceived depth profile of Fig. 8.17(A) lends credence to the claim of Mayhew and Frisby.

Besides the LOG operator, it is also possible to use other operators to extract points of high gray-level variance. For example, the Movarec interest operator has been used and also the Sobel, Roberts, and Prewitt, along with other first-derivative operators, have been used for edge point detection [see Rosenfeld and Kak (40) for a general discussion of edge detection]. Marr and Poggio (28) and Medioni and Nevatia (31) suggest the use of oriented masks to extract edge points. However, it is the opinion of the authors that the extraction of zero-crossings from a LOG-filtered image is in general a good choice because the circular symmetry of the LOG filter is supported by psychophysical evidence (30) and because convolution with a single LOG filter is computationally more efficient than using multiple-oriented filters.

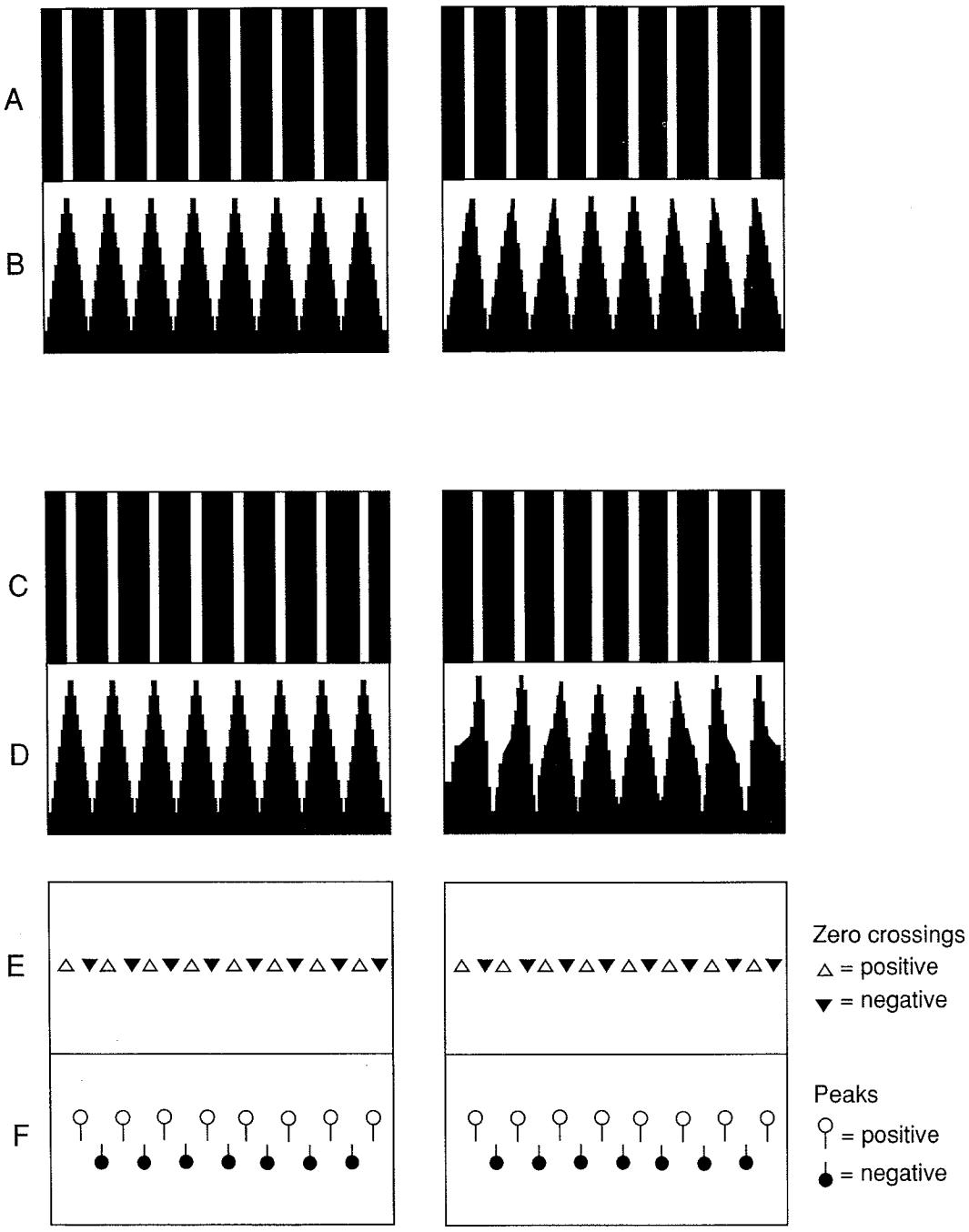

Figure 8.16. (A) Left and right stereo images; (B) brightness function of each image in (A) along a single row; (C) resulting images after convolution of filter with images in (A); (D) brightness function of each image in (C) along a single row; (E) extracted zero-crossings from each image in (C) along a single row; (F) extracted peaks from each image in (C) along a single row. From Mayhew and Frisby (30), with permission from the authors and Elsevier Science Publishers BV.

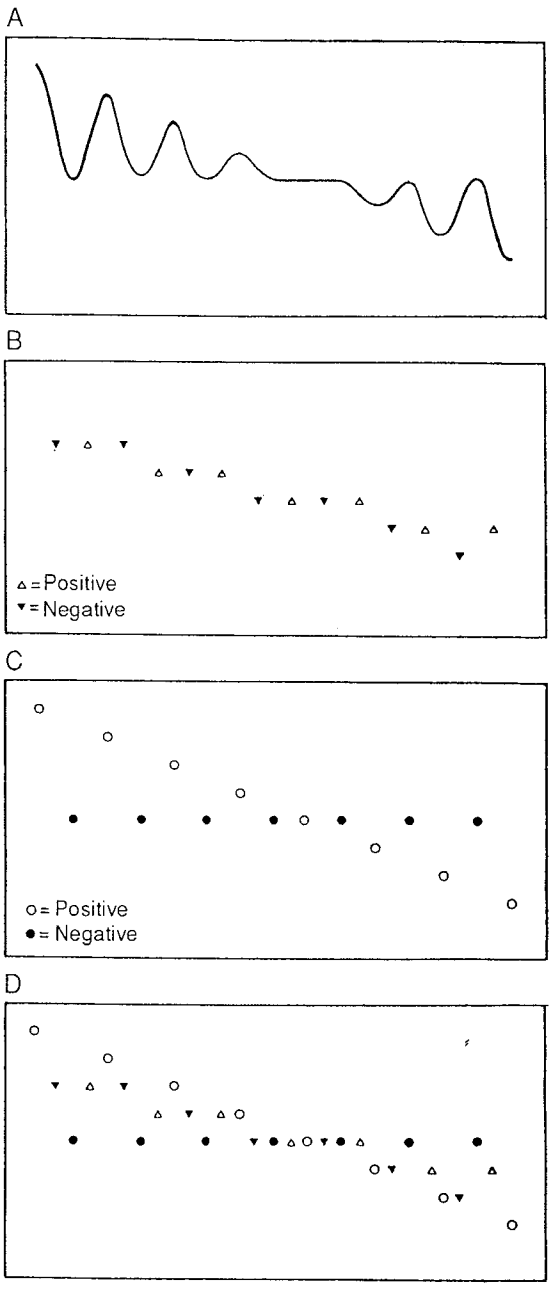

Figure 8.17. Depth profiles: (A) perceived depth profile (along a single row); (B) depth profile constructed from zero-crossings alone; (C) depth profile constructed from peaks alone; (D) depth profile constructed from zero-crossings and peaks. From Mayhew and Frisby (30), with permission from the authors and Elsevier Science Publishers BV.

4.1.3. Other Matching Techniques for Low-Level Features

Other methods have been advanced for solving the correspondence problem in stereo vision. In what follows, we will briefly review some of the more prominent of the alternatives to the MPG algorithm.

4.1.3.1. Relaxation-Based Matching

Probablistic relaxation is a process whereby the probability of a candidate match is iteratively updated depending on the probabilities of other neighboring matches. Thus, the disambiguation process in not performed in one step but in an iterative fashion. As discussed in Marr and Poggio (29) and Julesz and Chang (19), there is psychophysical evidence that a cooperative process between local matches occurs, which is used to arrive at a globally consistent set of matches. In addition, it has been shown that the closer the disparity values are of two matches in a neighborhood, the stronger will be their cooperation. The evidence of such cooperative processes validates the use of relaxation techniques for the disambiguation of multiple matches.

Most relaxation schemes for disambiguation will iterate a fixed number of times or until the desired matching results are obtained. An example of a desired result is that for each set of multiple matches, there exists one match that has a probability measure greater than the other probability measures by some fixed threshold. One of the nice features of a relaxation based method is that the update procedure is a function only of the probability values in a discrete neighborhood and thus can be implemented in a parallel fashion. In this way, the computation involved in each iteration can be greatly reduced.

The use of probabilistic relaxation in stereo was first advanced by Barnard and Thompson (4). They use a region-based disparity continuity constraint in updating the probabilities of the matches. For such schemes, the situation where a region encompasses a physical discontinuity will cause the same difficulty as it did for the original MPG algorithm. In a more recent contribution, Kim and Aggarwal (21) have proposed a modification of the Barnard and Thompson algorithm in which both the initial probabilities of the matches and the update procedures use the disparity information along the local zero-crossings contour. We will next discuss this algorithm.

Kim and Aggarwal's stereo vision algorithm (21) begins with the detection of the zero-crossings of the LOG-filtered stereo images. Next, initial matches are set up in the same manner as in the MPG algorithm for a single channel, and each candidate match is assigned a weight value. Suppose we are considering the ith zero-crossing in the left image, which has n candidate matches in the right image. Let $w_i(d_j)$ denote the weight assigned to the match of the ith left-image zero-crossing (x_i, y_i) and its

jth candidate (x_j, y_j) in the right image where d_j is the corresponding disparity. This weight is a function of how similar the left and right zero-crossings are in terms of the local orientations of their zero-crossing contours (w^1) and an intensity gradient measure (w^2) as follows:

$$w_i(d_j) = a^* w^1(d_j) + b^* w^2(d_j), \tag{21}$$

where a and b are constants and

$$\begin{aligned} w^1(d_j) &= \frac{1}{1 + |DP_{ij}|}, \\ w^2(d_j) &= \frac{1}{1 + \left|G_l(x_i, y_i) - G_r(x_j, y_j)\right|}, \\ G_l(x_i, y_i) &= \frac{\text{intensity}(x_i + 1, y_i) - \text{intensity}(x_i - 1, y_i)}{2}, \\ G_r(x_j, y_j) &= \frac{\text{intensity}(x_j + 1, y_j) - \text{intensity}(x_j - 1, y_j)}{2}. \end{aligned} \tag{22}$$

where DP_{ij} is a measure of the difference in orientation of the zero-crossing contours that fall in the 3×3 neighborhoods of the zero-crossings of the left image (x_i, y_i) and the right image (x_j, y_j), respectively. This can be easily measured by looking at the zero-crossing patterns in each 3×3 neighborhood. Figure 8.18 shows some examples of possible zero-crossing patterns that can occur. Assume that $D_1(x, y)$ and $D_2(x, y)$ represent the position of the first and second zero-crossings found in a counterclockwise traversal starting in the east direction of a 3×3 neighborhood centered at the zero-crossing (x, y) where the position assignments are shown in Fig. 8.18(I). For example, in Fig. 8.18(A) D_1 is 3 and D_2 is 7. The difference in local orientations is measured as follows:

$$DP_{ij} = DIFF_1 + DIFF_2,$$

where

$$DIFF_1 = \left|D_1(x_i, y_i) - D_1(x_j, y_j)\right|$$

$$DIFF_2 = \left|D_2(x_i, y_i) - D_2(x_j, y_j)\right|$$

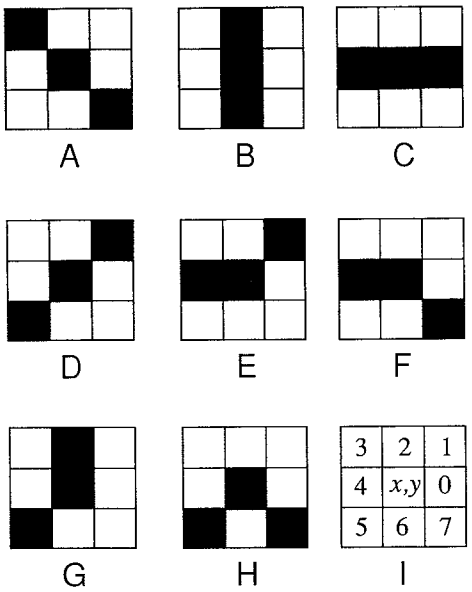

Figure 8.18. Parts (A)-(H) are examples of 3×3 zero-crossing patterns than can occur. Part (I) illustrates the position assignment given to the neighboring zero-crossing points.

and

$$\text{If } (DIFF_k > 4) \text{ then } DP_{ij} = |8 - DIFF_k|.$$

So far, we have assumed that each neighborhood will have two zero-crossings in it but, if that is not the case then the value of *DIFF* will be set to 20, which is a number large enough to ensure that the resulting weight w^1 will be small, thus indicating that the matched zero-crossings do not have similar surrounding zero-crossing patterns (i.e., orientations).

Next, initial probability values are assigned to each candidate match as a function of the weight values that we calculated above as follows:

$$P_i^0(d_j) = \frac{w_i(d_j)}{w_i(no\ match) + \sum_{k=1}^{n} w_i(d_k)}, \qquad d_j = no\ match, d_1, \ldots, d_n, \tag{23}$$

where $w_i(no\ match)$ is equal to $1 - \max[w_i(d_j)]$. Thus, the initial probability of the jth candidate match of the ith zero-crossing in the left image is the weight of this match normalized by the sum of the weights of all the other candidate matches where there are n such candidates.

The iterative step of the relaxation-based matching procedure updates the probability of each match with supporting matches that pass a regional

disparity continuity constraint and a disparity gradient constraint. The following equations describe the update procedure at the $(k+1)$th iteration:

$$P_i^{k+1}(d_j) = \frac{\tilde{P}_i^{k+1}(d_j)}{\tilde{P}_i^{k+1}(no\ match) + \sum_{r=1}^{n} \tilde{P}_i^{k+1}(d_r)},$$

$$d_j = no\ match, d_1, \ldots, d_n, \tag{24}$$

where

$$\begin{aligned} \tilde{P}_i^{k+1}(d_j) &= P_i^k(d_j) + cF\big(P_i^k(d_j)\big)P_S^k - dP_i^k(d_j)I(P_{FS}) \\ \tilde{P}_i^{k+1}(no\ match) &= P_i^k(no\ match) \end{aligned} \tag{25}$$

and

$$P_F^k = \max\big[P_f^k(d_j - 1), P_f^k(d_j), P_f^k(d_j + 1)\big]$$

$$P_S^k = \max\big[P_s^k(d_j - 1), P_s^k(d_j), P_s^k(d_j + 1)\big]$$

$$F\big[P_i^k(d_j)\big] = \begin{cases} \big[P_i^k(d_j)\big]^2 & \text{if} \quad 0.0 < P_i^k(d_j) \le 0.5 \\ \big[P_i^k(d_j)\big]\big(1 - P_i^k(d_j)\big) & \text{if} \quad 0.5 < P_i^k(d_j) < 1.0 \end{cases}$$

$$I(P_{FS}) = \begin{cases} 0 & \text{if} \quad P_F^k + P_S^k \neq 0 \\ 1 & \text{if} \quad P_F^k + P_S^k = 0 \end{cases}$$

Note that the subscripts f and s indicate the two zero-crossings found in the 3×3 neighborhood of the left image zero-crossing in question where f denotes the first zero-crossing found in the counterclockwise scan of the neighborhood and s denotes the second zero-crossing found. Therefore, $P_s^k(d_j)$ is the probability at the kth iteration that the second zero-crossing in the neighborhood has a match of disparity d_j. If there does not exist a first or second zero-crossing, then probabilities involving them are set to zero. The third term in Eq. (25) implements a disparity gradient constraint. The value $I(P_{FS})$ is set to 1, which activates a decrease in support, if both of the two neighboring zero-crossing produce a disparity gradient with respect to the zero-crossing match in question that is larger than 1. Because the two neighboring zero-crossings must lie on the same local contour, this is much like the limit of one in the disparity gradient limit constraint presented in Pollard *et al.* (37). The second term of Eq. (25) increases the support of a match if a neighboring zero-crossing has a

nonzero probability for disparities similar in value to the disparity of the match in question. This is what the authors refer to as a "regional" disparity constraint even though only a single neighboring zero-crossing point is considered.

4.1.3.2. Dynamic-Programming-Based Matching

Dynamic programming is a problem solving technique that is commonly applied to problems for which a recursive algorithm would tend to solve many of the same subproblems over and over again. The basic idea is that small subproblems are solved and their results are stored. Then larger subproblems and eventually the problem itself are solved by looking up the results of the smaller subproblems. Edge-point matching can be solved using dynamic programming when the problem is viewed as finding an optimal path through the set of all nodes where a node represents a possible match of a left-image edge point and a right-image edge point. Below we discuss an algorithm that is typical of the stereo algorithms that utilize dynamic programming.

Ohta and Kanade (35) present an algorithm that uses dynamic programming in a two-stage fashion for solving the stereo correspondence problem. The first stage finds the optimal set of matches along each epipolar scan line pair. Each such intra–scan line search is interpreted as the problem of finding a matching path on a two-dimensional (2D) search plane whose axes are the left and right scan lines being searched. Figure 8.19 illustrates the search space where a vertical line represents an edge-point position on the left-image scan line and a horizontal line represents the edge-point position on the right-image scan line. For convenience the 0th vertical and horizontal lines denote the leftmost pixel of each scan line; the Nth vertical line of the search space represents the right most pixel of the left image scan line and the Mth horizontal line represents the right most pixel of the right image scan line. An intersection point of a vertical and a horizontal line is referred to as a *node* and can be thought of as representing a potential match between the two corresponding edge points. Now, the goal is to find an optimal path between the nodes $(0, 0)$ and (N, M) in Fig. 8.19, where every node on this path represents an accepted match between two edge points.

Finding the optimal path from any node (p_l, p_r) to the starting node $(0, 0)$ is considered to be a subproblem in their dynamic-programming-based algorithm. However, to use dynamic programming there must exist an ordering from smaller to larger subproblems: for the intra–scan line search problem this is accomplished by the following constraint: When considering the match of two edge points (p_l, p_r), the edge points that are to the left of p_l on the left-image scan line and the edge points to the left of p_r on the right-image scan line should have already been processed. This constraint is referred to as the *left-to-right ordering of matches*. This

constraint is valid if the objects and their edge points in the scene retain a left-to-right ordering in the two stereo images and this will be true for most scenes.

When we say that we want the optimal path, we mean that the cost of this path is smaller than all of the other possible paths where the cost of a path is defined to be the sum of the costs of its primitive paths. A primitive path is defined as a link directly connecting two nodes. A link such as link c in Fig. 19 represents a continuous interval of pixels along each scan line. A measure of the cost of such a link is defined as how much the two intervals (one on the left-image scan line and one of the right-image scan line) are similar in terms of their gray-level values. Because the intervals may be of different lengths, the cost is defined in terms of the average gray-level variance of each interval with respect to the average of their mean gray-level values as follows:

Suppose $a_1, \ldots, a_k \in$ left-image interval and

$$b_1, \ldots, b_t \in \text{right-image interval},$$

$$m = \frac{1}{2}\left(\frac{1}{k}\sum_{i=1}^{k} a_i + \frac{1}{t}\sum_{j=1}^{t} b_i\right) = \text{average of means},$$

$$\sigma^2 = \frac{1}{2}\left(\frac{1}{k}\sum_{i=1}^{k} a_i - m^2 + \frac{1}{t}\sum_{j=1}^{t} b_i - m^2\right) = \text{average of variances } wrt\, m,$$

$$\text{Cost of link} = \sigma^2\sqrt{k^2 + t^2}\,. \tag{26}$$

This cost measure will be smaller for smaller values of the variance σ^2 that will occur for intervals of similar gray-scale content.

A cost measure of the optimal path from node (N, M) to node $(0, 0)$ is determined iteratively by adding the cost of each added primitive path to the already known optimal partial path. To clarify the ordering of nodes and how this iterative processing occurs, we define the distance of a node (i, j) as being equal to $i + j$. At the first iteration, the optimal path to the node $(0, 0)$ from each node that is at a distance of 1 is determined and these paths will be primitive by definition. At the second iteration the optimal path to the node $(0, 0)$ from each node at a distance of 2 is determined using the partial optimal paths calculated at the previous iteration. This iterative process continues until the (N, M) node is reached. The italicized number next to each node in Fig. 8.19 is the distance from the node to the node $(0, 0)$.

Although an optimal path of matches for each scan line pair can be found using the above dynamic programming procedure, the formation of

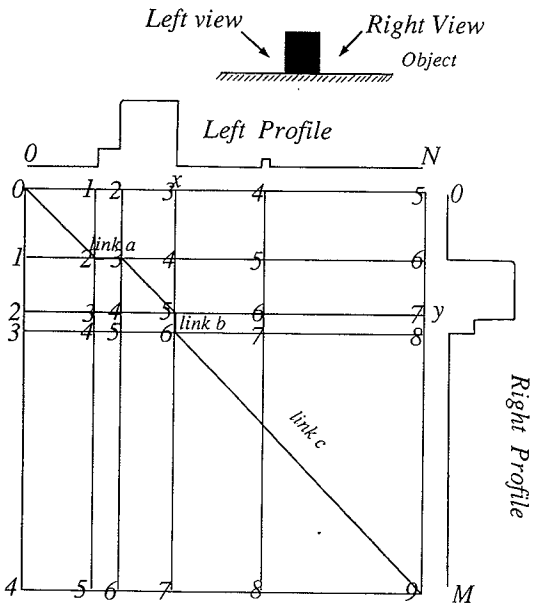

Figure 8.19. This illustrates the 2D search plane used in intra–scan line search. Along the top is a gray-scale profile of a row in the left stereo image, and along the right side is the gray-scale profile of the corresponding row in the right image. The vertical lines indicate positions of edge points in the left profile, and similarly, the horizontal lines indicate the positions of edge points in the right profile. Intersections of these lines denote possible edge point matches. Italicized numbers at intersection points denote their visitation order in the intra–scan line search procedure.

these matches does not take into account the figural constraints that must consider edge points across scan lines. To get around this shortcoming, Ohta and Kanade propose a dynamic programming approach that integrates intra–scan line search with inter–scan line constraints. Inter–scan line search is interpreted as searching in a 3D space that is constructed from stacking the 2D intra–scan line search spaces in order of the scan line numbers. The goal now is to find an optimal surface of matches (nodes) that minimizes the intra–scan line costs and at the same time satisfies inter–scan line consistency. A similar left-to-right ordering of edge contours of the two images is assumed and the reader is referred to Ohta and Kanade (35) for details.

The assumption of left-to-right ordering of edge points or contours in an image is not valid for scenes where neighboring long thin vertical objects exist at varying depths. In this case, the projections of these objects may actually reverse position in a left–right sense between the two camera images. For example, hold up your two index fingers in front of you at very different depths but very close in terms of a projected distance along a horizontal line (see Fig. 8.20). Now, alternate looking through your left eye and your right eye and you will observe that the fingers project in a

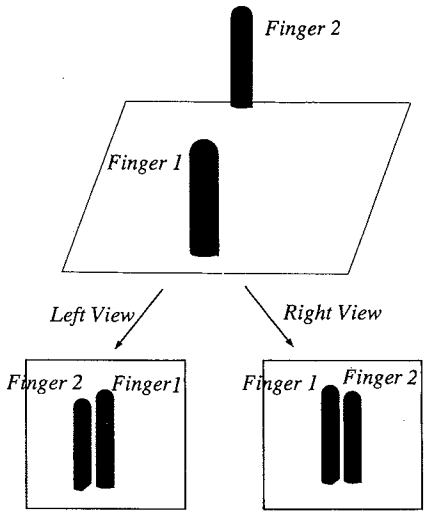

Figure 8.20. Scene where the reversal of the left-to-right ordering of the objects occurs between the two stereo images.

different left–right order for each eye. This is one limitation of the dynamic programming approach that the previous two matching paradigms do not suffer from.

The advantage of using a dynamic programming approach is that an optimal set of matches are obtained. The disadvantages are the large execution time typically needed for dynamic programming algorithms and the large amount of storage needed for the results of subproblems. In Ohta and Kanade (36), an attempt is made to reduce the processing time algorithm by implementing portions of the algorithm in hardware. A reader interested in another stereo vision system that uses dynamic programming is referred to Lloyd (24).

4.2. High-Level-Feature-Based Stereo Vision

As the reader should have surmised by now, one of the difficulties with the use of low-level features for stereo, such as the zero-crossings produced by an LOG operator, is the problem of disambiguation, meaning the difficulty associated with deciding which candidate match to accept if there is more than one contender. This difficulty with disambiguation is ameliorated when higher-level features are used. In the rest of this section, we will review four contributions, by Medioni and Nevatia (31),

Tanaka and Kak (44), Boyer and Kak (6), and Horaud and Skordas (16), for matching straight line segments in two images. Another contribution we will discuss after these is by Nasrabadi and Liu (34) on the matching of curved segments. There is an interesting tradeoff between the use of relational constraints in the matching process and the computational efficiency of the algorithms presented in Tanaka and Kak (44) and Medioni and Nevatia (31), on one hand, and those presented in Boyer and Kak (6), Horaud and Skordas (16), and Nasrabadi and Liu (34) on the other. In comparison to Tanaka and Kak (44) and Medioni and Nevatia (31), the algorithms of Boyer and Kak (6), Horaud and Skordas (16), and Nasrabadi and Liu (34) have a stronger relational flavor, but at the expense of decreased computational efficiency. Algorithms that utilize relationships between features come under the rubrik of *structural stereopsis*, and those that do not, we will simply refer to as *edge-based stereopsis*.

4.2.1. Edge-Based Stereopsis

Medioni and Nevatia (31) propose an iterative method for solving the correspondence problem using straight line segments as features. These features are composed of zero-crossing points and each feature is described by its endpoints, orientation, and the average gray-level variation across the line. Initially, all right-image lines that fall in the search space for a given left image line and have similar attribute values with respect to the attribute values of the left-image line are considered to be candidates for matching (and vice versa). The search space is bounded in one direction by the epipolar lines of the left-image line's endpoints and in the perpendicular direction by a span w, which is a function of the maximum expected disparity in the scene. Every match, (l_i, r_j), where l_i is a left-image line and r_j is a right-image line, is assigned a value, $v(l_i, r_j)$, that measures how well the disparity of the other line matches in neighborhoods of both l_i and r_j agree with the average disparity d_{ij}, computed along the matched lines (l_i, r_j). This evaluation function implements a region-based disparity continuity constraint and is set up so that a small value indicates a better match. The following algorithm describes the iterative scheme presented in Medioni and Nevatia (31) that removes matches from the initial match set using the measure v.

```
M={set of initially constructed matches}
∀l_i Q(l_i)=initially constructed matches for l_i
∀r_i Q(r_i)=initially constructed matches for r_i
t=0
```

```
Until (Termination_Condition)

   {

      Calculate v^t(l_i, r_j) for all (l_i, r_j) ∈ M using only sur-
    rounding matching segments in Q.

      ∀ l_i { Eliminate any r_j ∈ Q(l_i) that does not form a
              ``preferred'' match }
      ∀ r_i{ Eliminate any l_j ∈ Q(r_i) that does not form a
              ``preferred'' match }

    t++

   }
```

where a match (l_i, r_j) is ``preferred'' if

$$\forall\, r_k \text{ where } (l_i, r_k) \in M \text{ and } r_k \text{ and } r_j \text{ overlap then } v^t(l_i, r_j) < v^t(l_i, r_k)$$

AND

$$\forall\, l_h \text{ where } (l_h, r_j) \in M \text{ and } l_h \text{ and } l_i \text{ overlap then } v^t(l_i, r_j) < v^t(l_h, r_j)$$

A simple termination condition requiring the number of iterations t to be equal to three is used and the matches with the smallest v values are accepted. An advantage of this method is that more than one preferred match is allowed for a particular line segment. For example, line segment l_i of the left image is allowed the following two matches (l_i, r_j) and (l_i, r_s) if r_j and r_s do not overlap. This capability handles the situation where a line segment in one image corresponds to more than one line segment in the other image. A drawback of this system is that the number of iterations needed to totally disambiguate the match set is unknown and stopping at a prechosen number will not guarantee that the results yield the best match set. Another problem that can arise is a result of the assumption that a region-based disparity constraint is desirable for line features. Depending on the size of the neighborhood and the type of objects in the scene this assumption can lead to incorrect results. For example, consider the situation where neighboring lines are produced from the physical edges of two nearby objects that are at very different depths. These lines should produce very different disparities, and they will incorrectly inhibit each other as valid matches using a regional disparity constraint. It is our opinion that the use of relational constraints such as collinearity and adjacency are more appropriate than region-based disparity constraints for use in segment feature matching systems.

Tanaka and Kak (44) present a hierarchical stereo algorithm in which one step of this hierarchy involves straight line feature matching. Like the previous algorithm, this algorithm also allows an edge in one image to correspond to multiple edges in the second image. Since the algorithm does not take into account any relational constraints during the matching of the line segments, it is computationally efficient. (Note that, unlike the Nevatia–Medioni algorithm, not taking into account relational constraints while matching straight line segments does not degrade the performance of stereo fusion since the edge matching takes place in a larger hierarchical framework.) An outline of the algorithm follows (it is assumed that the canonical camera configuration is used and the images are row-registered):

Feature Extraction

1) For each image extract straight line segments using modified Freeman criteria so that each segment is S pixels long. Represent a line longer than S pixels by overlapping segments such that their starting pixels are spaced s pixels apart. Segments that are nearly parallel to epipolar lines are removed from the image.

Generate Initial Candidate Matches

2) Consider every right image line segment, R, to be a candidate match for a left image line segment, L, if:

$$Start\_row(R) = Start\_row(L) \pm t$$
$$Start\_col(R) = Start\_col(L) \mp d_{max}$$

The value of t is 1. (t accounts for the possible misregistration of the epipolar lines and perspective effects that can result from the fact that the two cameras may not have truly parallel axes.) The value of d_{max} is the maximum expected disparity.

Disambiguation of Multiple Candidates

3) Evaluate each candidate match (L, R) by comparing the similarity of their orientations using chain code descriptions of the line segments. Let $L(i)$ and $R(i)$ be the i-th element of the chain codes along L and R. L is similar to R if

$$\forall\, i$$
$$if(L(i) == R(i))\ \{similarity{+}{+}\}$$
$$if(L(i) == R(i-1)\ OR\ (L(i) == R(i+1)))$$
$$\{similarity = similarity + w\}$$

The second test is needed in the case where the chain codes of diagonal lines are being compared.

Consider the two following chain codes: 1212121212... and 2121212121... which represent lines at approximately 45°. These two segments would be considered to have similar orientations by a human observer. The second test ensures that the algorithm will also judge them as similar.

4) If *similarity* is larger than a threshold then accept the match (L, R) and delete L from left image. If no match for L, delete L from left image.

5) Go to step 2 until no more edges exist in the left image.

Disparity Calculation

6) For every match (L, R) found, compute the disparity values along the segments as follows: $D(i) = L_{col}(i) - R_{col}(i)$ where col indicates the column value of the current chain code position.

In the work reported by Tanaka and Kak (44), the length parameter S was set to 41 and s to 4 pixels; these choices were dictated by empirical considerations. Also, the threshold in step 4 was empirically chosen to be 50 pixels. It is assumed in this algorithm that dominant edges, meaning edges that are long and straight, are few and will in most cases result in unique matches. The scene lines constructed by fusing long edges then serve as anchors for testing hypotheses regarding the presence or absence of planar surfaces in the scene.

For scenes in which the objects have edges far apart in comparison to d_{max}, this algorithm will yield good results and do so in linear time with respect to the number of edges in either image. As we will see in the next two examples, it is possible to construct straight line and contour matching algorithms that are capable of matching straight line edges even when they do not obey the d_{max} separation constraint, but by entailing exponential computational burden.

4.2.2. *Structural Stereopsis*

While the previous two approaches enforce constraints on the continuity of disparity along line segments, the approaches described in this section use the relationships between the line segments to improve the robustness of stereopsis. In this way, the structure present in the images is explicitly used; hence the name *structural stereopsis*.

Boyer and Kak (6) present a system that seeks to find a mapping between the set of straight line edge segments in the left image to the set

of straight line edge segments in the right image that best preserves the relational constraints between the matched line segments. Each line segment is described by attributes such as its length in pixels, and mean orientation of the edge segment. The relationships used between two segments are as follows: the orientation and the length of the line that connects the centroids of the two segments in question, and the length of the line that connects the closest pair of endpoints of the two segments in question.

In the Boyer–Kak approach, a mapping from the segments in the left image to those in the right image is constructed by minimizing a cost function that has two components; the first component measures the dissimilarity of the attributes of the matched segments, and the second measures the inconsistency of the relations between the matched segments. The cost function is formulated using information-theoretic considerations, which means specifying probabilities for the image-to-image distortion of the attributes of the segments and their relations. *Nilmapping*, which consists of assigning nils to the segments of the left image, (necessitated by the fact that in the presence of occlusions there may not exist a right-image correspondent for a left-image segment), plays an important role in the discovery of a best-mapping function. The mapping function itself is constructed by employing a backtracking search; the structure of the search space is such that each level of the search tree corresponds to one left-image segment and the different nodes at a level represent the different possible right-image candidate segments. A principal shortcoming of the Boyer–Kak approach is that the system is incapable of dealing with the fragmentation of segments. What that means is that if a left-image segment shows up as two or more disconnected segments in the right image, this system would be incapable of discovering that fact. Another problem, which we will not describe here in any detail for reasons of space limitation, is caused by large depths of field in a scene. Both these shortcomings of the Boyer–Kak method have been rectified in a recent contribution by Kosaka and Kak (22).

In Horaud and Skordas (16), structural stereopsis is implemented with a graph-theoretical approach where straight line edge segments are used as features. In graph-theoretic approaches, the set of line segments and their relations in each image are represented by a graph data structure, in particular the relational attributed variety. In general, a relational attributed graph (RAG) is constructed for each image where a node of the graph represents a segment and the arcs indicate relationships between segments of the connected nodes. Each node in the graph has a set of attributes that describe the structure of the edge segment assigned to the node. The problem of establishing correspondences between the line segments in the two images then becomes an exercise in finding the largest

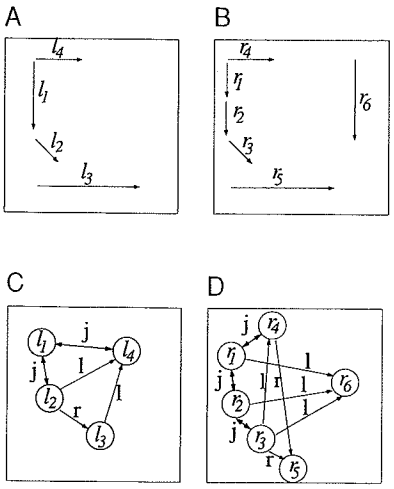

Figure 8.21. Example relational attributed graphs (RAGs) are constructed for a set of stereo images: (A) straight line segments found in the left image; (B) straight line segments found in the right image; (C) RAG of left image; (D) RAG of the right image. Note in (C) and (D) the meanings of the labels on the links are as follows: j = junction, l = left-of, r = right-of, c = collinear. Also note that for every link with label l (or r) there is another link in the other direction with the label r (or l) that is not drawn.

double-subgraph isomorphism.[1] Note that "double" subgraph isomorphisms are required rather than subgraph isomorphisms because a segment in one image may correspond to more than one segment in the other image as a result of occlusion, illumination differences, and so on that can occur between the images and effect the extraction of segments. Another reason for stating the problem as a double-subgraph isomorphism problem is that a relation between a particular pair of segments in one image may be different from the relation for the corresponding pair of segments in the other image. For example, left-to-right ordering is a common relationship used, and, as illustrated by Fig. 8.20, a reversal of this ordering can occur for thin vertical objects that are spatially close in the image domain but separated by a large depth.

Horaud and Skordas (16) use adjacency, collinearity, and intersection for the relations. *Adjacency* is indicated by whether a line segment is directly adjacent to another segment in a left or right sense. For a given line segment, these left and right adjacencies are detected by scanning in a direction perpendicular to the orientation of the segment and recording

[1]As stated in Ballard and Brown (3), the problem of solving "double" subgraph isomorphisms consists of finding all isomorphisms between the subgraphs of a graph and the subgraphs of the other graphs.

hits with other line segments. *Collinearity* is simply a measure that indicates that two line features lie on the same line in the image plane. Finally, an intersection point (junction) occurs when two line segments intersect. Figures 8.21(A) and (B) show, for the purpose of illustration, a schematic of line segments that could be the output of an edge detector applied to the two images of a stereo pair. The arrowheads on the line segments indicate the orientation of the underlying edges with respect to the black-to-white transitions. Figures 8.21(C) and (D) are the corresponding relational attributed graphs. As explained in the legend, the labels correspond to the different relations.

Horaud and Skordas (16) solve the problem of finding the largest double-subgraph isomorphism by using what they call a *correspondence graph* (GC). A CG has a node for every feasible match between the lines of the left and right images. Arcs are connected between two nodes in the CG if they are compatible, meaning that all the relations concerning the lines in question in the RAGs match or at least do not dispute each other. For example, say we have two nodes of the CG called $n_1 = (l_1, r_2)$ and $n_2 = (l_2, r_3)$ then the two nodes are compatible if the relations (arcs in RAG) between l_1 and l_2 exist between r_2 and r_3 or at least do not conflict and vice versa. Using the CG derived from a stereo pair of images, the solution to the correspondence problem is to find the "best" maximal clique.[2]

As Horaud and Skordas discuss, there may be more than one largest maximal clique, and in practice even if there is a unique largest maximal clique, it may not result in the best stereo matching. Thus, as presented below, a benefit value is calculated for each node and the benefit of a clique is the sum of the benefits of its nodes. Consequently, the "best" maximal clique is defined as the maximal clique with the largest benefit. The following steps outline the process of matching line features in their system:

1) Create nodes of CG

for every *l*, a line of the left image, find all of the possible lines of right image, *r*, that can match given the epipolar line, position and orientation constraints. The search space for finding candidate lines in right image is bounded by the epipolar lines of the endpoints of line *l*. The position constraint delimits the span along the epipolar line that a valid candidate right line must be in and this is a function of the field of

[2]A "clique" is a completely connected subgraph; each node is connected to all the other nodes of the subgraph. A maximal clique is a clique that cannot be extended to include any other nodes of the graph.

views of the cameras (no expected disparity information is used). The orientation constraint requires that two matching lines be within 30° of each other in orientation.

2) For each node=$(l,r)\in$CG calculate its benefit:

$$Benefit(node)=1/4\left(\frac{\min(C_l,C_r)}{\max(C_l,C_r)}+\frac{\min(L_l,L_r)}{\max(L_l,L_r)}+\frac{\min(\Theta_l,\Theta_r)}{\max(\Theta_l,\Theta_r)}+\frac{\min(N_l,N_r)}{\max(N_l,N_r)}\right)$$

where C_i=contrast across line i.
L_i=length of line i.
Θ_i=orientation of line i.
N_i=# of relations line i has with other lines of same image.

3) Remove nodes with $Benefit(node)<$Threshold
Note that Threshold is empirically chosen.

4) Place arcs between nodes in CG that are compatible.

5) Find maximal cliques in CG.

6) Take maximal clique with max $Benefit(clique)$
where $Benefit(\mathbf{clique})=\sum_{nodes\in clique} Benefit(node)$

An advantage of this system is that it allows one edge in an image to match to two or more edges in the other image. The major drawback of this algorithm is that finding all of the maximal cliques (step 5) in the CG is an NP-hard problem, and as such this step will be of exponential time complexity in terms of the number of nodes in the CG. For one example of an office scene it took 26 minutes to run the line correspondence algorithm on a 11/780 VAX. The authors attempt to reduce the actual time spent in finding the maximal cliques by removing nodes in the CG that have a low benefit value (step 3). There are also other heuristic methods that possess smaller time complexities, however at the risk of not finding all of the maximal cliques and therefore possibly not finding the best stereo match.

Nasrabadi and Liu (34) present an approach, similar to the one proposed by Horaud and Skordas (16), in which curve segments consisting of zero-crossing points are used as features. Like the Horaud–Skordas algorithm, this algorithm forms a correspondence graph and finds its maximal cliques. Each curve segment is described by its centroid location, length,

and Hough transform. The Hough transform is used here to describe the shape of a curve segment. After a RAG is formed for each image a CG is constructed, but only the distance between the curve segments' centroids is considered to determine whether two nodes of a CG are compatible. Unlike the previous system, this system simply chooses the largest maximal clique. Nasrabadi and Liu have implemented a coarse-to-fine processing strategy using curve segments derived from zero-crossing contours generated by LOG operators at different scales. As was the case in the MPG algorithm, the use of LOG operators also allows the authors to implement vergence control.

Nasrabadi and Liu argue that using curve segments is better than straight line segments since the extraction of straight lines via a piecewise segmentation process can introduce problems with edge localization. They extract semantically significant curve segments through a curve tracing routine that traces only zero-crossings that are above a chosen gradient threshold. Tracing is iteratively started at a maximum gradient zero-crossing point and the current curve is ended when the next zero-crossing in the trace is at an orientation[3] 10° different from the orientation of the previously visited zero-crossing or if it has a gradient magnitude that is below a certain threshold. The lengths of the extracted zero-crossings will be a function of the gradient and orientation thresholds chosen. The larger curve segments generally correspond to the physical edges of the objects in the scene or significant (nonrandom) texture patterns on the objects and therefore small curve segments are ignored.

In the algorithm reported in Nasrabadi and Liu (34), the disparities along the matching curve segments are calculated after the largest maximal clique is found. The system attempts to resolve the ambiguity that arises when part of a matching curve is occluded or when some of the pixels on a curve are along the epipolar line by assigning the disparity of these pixels to be equal to the disparity at the curve segment's centroid.

4.3. Incorporation of Object-Level Information in Stereo Vision

There are two distinctive ways in which object-level information is used in stereo vision systems. The first is in the use of very high level features in matching such as vertices, junctions, and surfaces. The advantage of using ultra-high-level features is that the probability of mismatches occurring is very low because of the sparseness of these features in the scene. A second use of object-level information is in the area of surface reconstruction where knowledge of the surface characteristics can be used to correct

[3]"Orientation" here refers to the direction of the maximum gray-level variation at the zero-crossing point in question.

mismatches. Surface reconstruction techniques are usually applied to dense disparity maps that are produced from the matching of lower-level features. Below we describe a few systems that illustrate each of the ways in which object-level information is used.

4.3.1. *Object-Derived High-Level Features*

In the preceding sections, the goal of the stereo system has been to recover the depth information of any type of scene. However, in this section we relax this goal and now only wish to recover the depth information of a particular type of scene, namely, one populated with particular objects. In this sense these stereo vision systems will be domain-dependent and will most probably not function well with a scene from a different type of domain. We will describe two systems that differ in the breadth of their domains. The first system is more specific in that it is only for scenes of a particular set of objects. The importance of discussing such a system is that it is very fast and tuned to the eventual recognition of the objects in question, which is often the goal of many industrial projects. The reader is also referred to Mohan and Nevatia (33), which briefly discusses a stereo system that matches ribbons where a ribbon is a surface-like feature that is extracted by the system using perceptual organization techniques. The system of Mohan and Nevatia (33) differs from the two systems presented in this section mainly in that it uses perceptual organization techniques for feature extraction.

Currently, we are working on a stereo vision system for the recognition and pose estimation of a set of industrial objects. There are four unique types of objects, three of which are shown in Fig. 8.22(A). Because our goal is very specific, we are able to adopt an object/domain-dependent approach where we use object-derived high-level features for stereo matching. The system takes on a hierarchical flavor in the sense that first objects are monocularly recognized, then these objects are matched, then the corners of each pair of matched objects are matched, and these are used to create a sparse disparity map. "Corners" are defined as the physical vertices of the objects. Matching the objects takes place first because there are only a few objects in the scene and the chance of mismatching is very low. Subsequently, the matching of corners can take place using constraints from the previous object-level matching. More specifically, corners of a particular left-image object must match with corners of the matching right-image object. This method works well for the type of scenes we are dealing with and is much faster than a low-level-feature-based stereo vision algorithm. Figure 8.23 illustrates the processing steps of this system.

Objects are monocularly detected through the use of their boundary edges. These edges are detected via Sobel edge detection and thinning

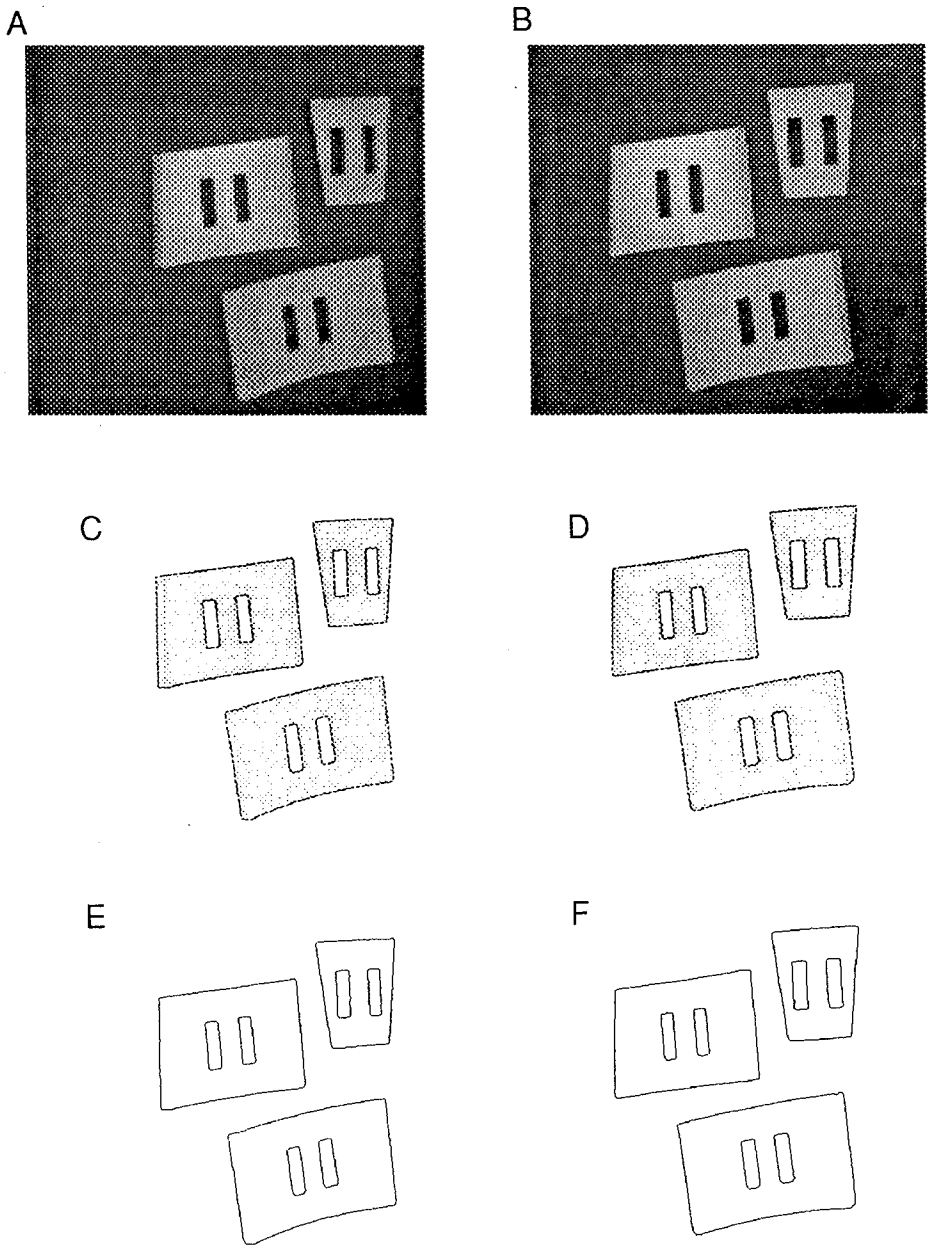

Figure 8.22. (A), (B) Original gray-scale stereo image pair; (C), (D) thresholded images convolved with Sobel operator; (E), (F) results after thinning edges (*Figure continues.*)

and a skeleton tracing procedure. First, the stereo pair of images are thresholded in order to remove the background noise. The results of convolving the thresholded images with a Sobel operator are shown in Figs. 8.22(C) and (D), and the output after thinning is shown in Figs. 8.22(E) and (F). The resulting images are passed to a skeleton tracing procedure that removes skeletons that do not correspond to object boundary edges. During skeleton tracing, the corners and their connecting linear boundary edges are found where corners are defined to be points of high curvature. Objects can be monocularly recognized from the information

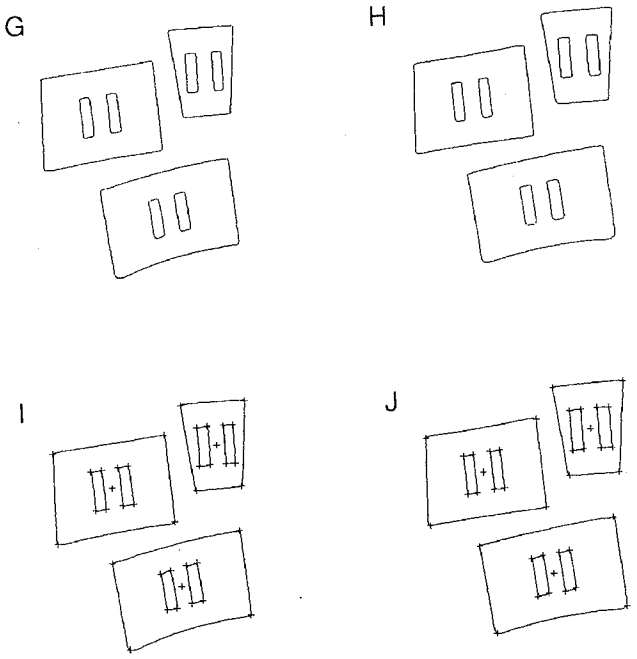

Figure 8.22. (*Continued*) (G), (H) results after skeleton tracing; (I), (J) corners found in the images are marked by cross symbol, and the object centers are also marked.

gleaned during this processing step. At this point, each detected object has three skeletons associated with it. The largest skeleton corresponds to the object's outside boundary edges, and the two smaller skeletons are the bounding edges of the two grasping holes inside of the object. Figures 8.22(I) and (J) show the corners that are detected for our example.

Because the detected objects are large with respect to the image size, both left-to-right and top-to-bottom ordering of the objects can be used to make object matches between the images. Actually, the objects can be matched with respect to their centers, where a "center" is defined as the centroid point of the pixels in the object's outside bounding edges. First, all of the object center points must be reprojected so that the points lie in image planes of the canonical camera configuration (see Section 3 for discussion on reprojection). Now, a search window can be set up in the right image surrounding each object center point in the left image with a $\pm d_{\max}$ span along the horizontal direction and a $\pm t$ span in the vertical direction. As before, $d_{\max}$ is the maximum expected disparity, and a small value of t such as 1 accounts for errors in the reprojection process. In our scenes, there is only one candidate right-image object center in this window. If this candidate is of the same object type, then the two objects are considered to be matched.

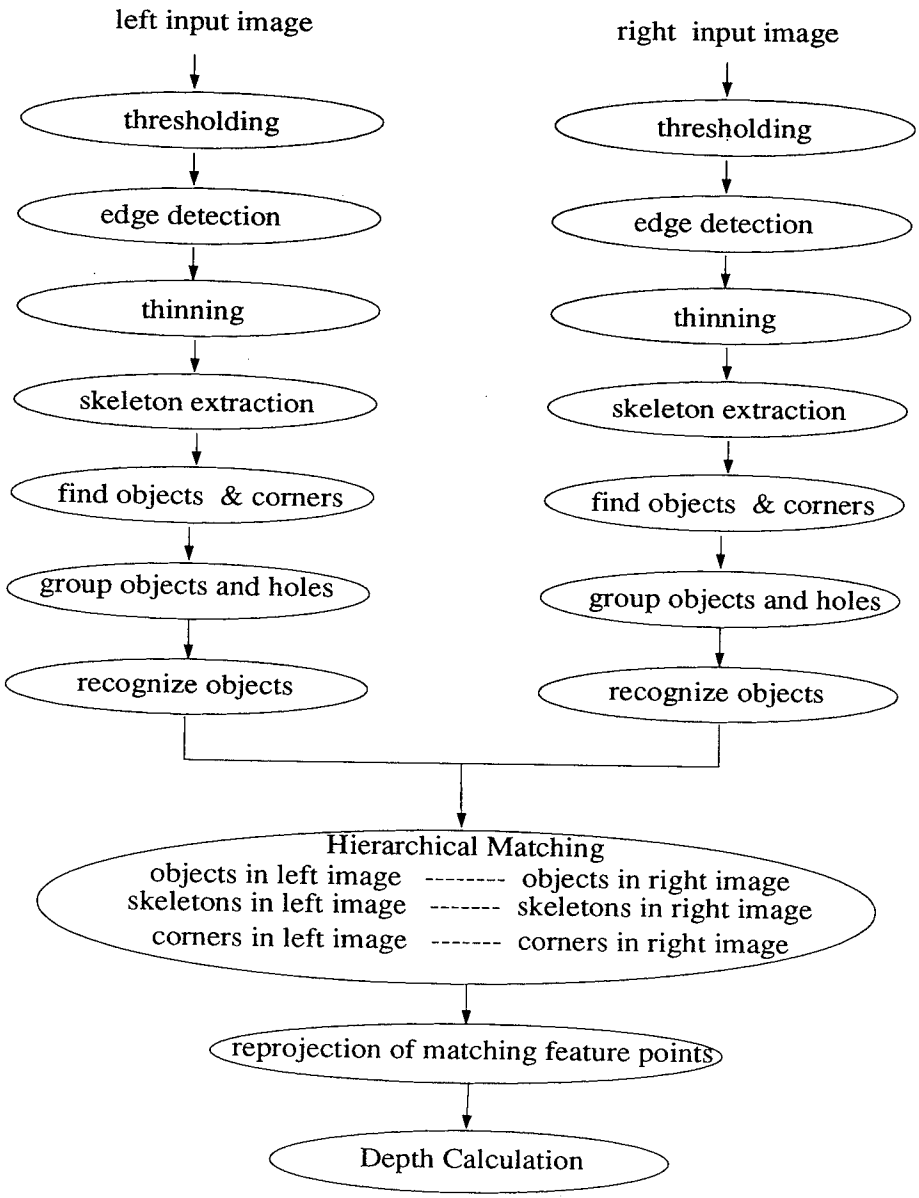

Figure 8.23. Block diagram illustrating the processing steps in the high-level-feature-based stereo vision algorithm for the eventual recognition of a set of industrial parts.

Each object has at most three skeletons associated with it, and each of these skeletons can have at most four corners. Before corner matching can occur, matching of the set of skeletons of each previously found pair of matched objects must take place. It is very obvious that the two largest skeletons that are the outside bounding edges of the matched objects must match. Therefore, only the two inside skeletons of each object remain to be matched, and this is easily accomplished with the use of a vector comparison test that maintains the left-to-right and top-to-bottom ordering that exists for these skeletons. Vectors are created from the center of each inside skeleton to the center of its object. Then a skeleton of the

left-image object will match to the skeleton of the corresponding right-image object whose vector is most similar in orientation. Because of the very large difference in the orientation of these vectors, this scheme efficiently captures the left-to-right and top-to-bottom ordering of the inside skeletons. A similar matching scheme for the corners of each pair of matched skeletons takes place, but this time the vectors are with respect to the centers of the skeletons that contain them.

Lim and Binford (23) present a system that is in some ways similar to the previous one in that it uses a hierarchy of scene features. While it is true that the set of objects for which this system is geared is larger, we believe that the system will perform best for polygonal objects that do not occlude one another in the scene. The high-level features used are bodies, surfaces, curves, junctions, and edgels. A "body" is a set of connected surfaces in the scene and a surface is described by a set of boundary curves that are connected at junction points. These curves are listed in order of a directional traversal of the surface's boundary. "Surfaces" can be either open or closed. A closed surface is a surface whose boundary curves form a closed loop. An open surface does not form such a loop and can occur as a result of an occluding surface or breaks in the curves due to noisy data or illumination patterns. "Curves" are either straight line or conic edge segments and junctions are points where curves intersect. Finally "edgels" are loosely defined here as short linear edge segments.

Monocular detection of the features in each image begins with the detection of edgels, where an edgel is characterized by its direction and position. Next, edgels are grouped into ordered sets that correspond to extended edges and curves are fit to these sets where a curve can be either a straight line or a conic section. Junctions are detected in the image as points where more than two curves intersect; curves are broken up at junctions. Now a curve tracing procedure takes place to find the surfaces. Unlike the previous system, this tracing routine is not fine-tuned to any particular set of objects, and thus surfaces can be found that are actually a composite of more than one surface or only part of a surface in the scene. Finally, bodies are detected as groups of surfaces that share edges, and as such, a body may consist of more than one object in the scene.

Matching occurs in a top–down fashion in the following order: bodies, surfaces, curves, junctions. The higher-levels of matching produce constraints for the lower levels that greatly reduce the number of match candidates. The matching process begins by matching bodies according to how similar they are in terms of the number of surfaces they contain, the order of these surfaces, and the relative positions of the bodies in each image. For each pair of matched bodies, their surfaces are matched on the basis of characteristics such as the number of junctions and the relative positions of the surfaces in the body. Finally, for each surface match, their curves and junctions are matched according to ordering constraints. Dis-

parity and depth values are computed at the matched curves and junctions.

The authors mention that a subsequent step of surface interpolation could take place on the sparse disparity map. However, this can happen only if each surface can be recognized from its boundary curves along with its pose, as was accomplished by the system described previously.

4.3.2. Surface Reconstruction Techniques in Stereo Vision

Surface reconstruction can be a process that is totally independent from the matching process in stereo vision and takes place after an unambiguous disparity or depth map has been produced. More recently, surface reconstruction has been incorporated into the feature matching stage of stereo vision systems and used to disambiguate multiple matches. This integration of surface reconstruction obviously assumes that some types of surfaces exist in the scene and as such utilizes object-level information. In this section, we describe two systems that use surface reconstruction to perform disambigution of multiple matches. The main difference between the two systems lies in the fact that in the first, surface reconstruction takes place only in selected local regions, whereas in the second, surface reconstruction is attempted everywhere.

Surface reconstruction, also called *surface fitting* and *surface interpolation*, is the process of finding the surface that most of the 3D points in question will lie on or close to. In stereo vision systems, these 3D points are either the points of the disparity map or its corresponding depth map. It is important to note that fitting a particular type of surface to a set of depth points is equivalent to fitting the same type of surface to the corresponding set of disparity points. Commonly, planar or quadratic surfaces, in mathematical terms called the *first-* and *second-order surfaces*, are fit to the data. Higher-order surfaces are not used because accurate fitting requires a larger density of data points, which seldom is available.

Tanaka and Kak (44) present a hierarchical stereo vision algorithm that, unlike the systems described in Section 4.3.1, produces a dense disparity map. The following is a list of the processing steps of this system in order of their execution: (1) straight line matching, (2) application of geometric constraints, (3) curve segment matching, and (4) zero-crossing point matching. The denser disparity map is a result of the second processing step, which we will discuss below, and the last step, where an MPG-type algorithm is invoked for pixels in the disparity map that are not assigned values from the previous three steps. We will now discuss the step of the algorithm that deals with the application of geometric constraints. The algorithm assumes that the surfaces in the scene are all planar and the orientation of each planar surface is one of a known set of orientations.

Evidently, this assumption severely limits any practical usefulness of the algorithm, but the algorithm is of historical interest since it is one of the first stereo algorithms that imposes top–down constraints on the stereo fusion process. Here is a description of the algorithm:

```
For each θ∈{Finite Set of Assumed Orientations}
    1) Grow/Fit a 16 pixel wide region to a plane of ori-
    entation θ that contains the pair of matched straight
    lines in question.

    2) The goodness of the fit is evaluated by matching
    the zero-crossings in this 16 pixel region from the
    finest channel. The disparity values corresponding to
    the planar strip produced in step 1 are used to center
    the search windows used in matching. The number of
    zero-crossing matches is recorded.

The planar strip (θ) that leads to the largest number of
zero-crossing matches is regarded as the best fit and the
corresponding disparity values are stored in the final dis-
parity map of the scene.
```

Hoff and Ahuja (15) describe a system that integrates low-level-feature-based matching with surface reconstruction at every point in the image. An MPG-type algorithm is used to produce zero-crossing matches at three different resolutions and a coarse-to-fine matching strategy is used. All of the multiple zero-crossing matches produced are kept, and surface fitting is used for disambiguation where the points that lie on or closest to the surface are kept. Both planar and quadratic surfaces are locally fit to the data. In addition, a final global surface fitting operation takes place after occluding and ridge edges in the scene have been detected.

First, a sparse grid is created at a spacing of w_{2D}, which is the size of the LOG operator used to find the zero-crossing points. A planar surface is fit to a circular patch centered at each grid point with a radius in the range of w_{2D} to $2w_{2D}$. The two best fit planes are kept. A fit is accepted only if the number of unmatchable points in the region is less than an empirically determined threshold and if the sum of the squared errors is less than another empirically determined threshold. The squared error at a point is the square of the distance from the point to the surface. Points whose distances to the plane being fit are greater than what is commonly called an "outlier distance" are considered unmatchable and are not included in the sum of the squared errors.

One way in which a planar surface can be fit to a set of data points is through the use of a least-squares method where all of the combinations of points are tried for each surface fitting. The data set that has a planar patch fit with the minimum squared error and passes the threshold tests mentioned above is considered as the best-fitting plane. The total number

of these combinations is exponential in the number of ambiguous points. However, the system in Hoff and Ahuja (15) avoids this exponential complexity by using a Hough transform of the data points as the surface fitting procedure. A surface can be described by a set of parameters, as in $z = ax + by + c$ for a planar surface. The 3D planar parameter space, also called the *Hough space*, spanned by the parameters a, b, and c, is discretized and a Hough transform is computed for all of the data points. The maximum point in the Hough accumulator space yields the parameters of the best-fitting plane. The amount of time it takes to find the best planar fit using the Hough transform method is a function of the size of the 3D Hough space, specifically, the coarseness of the discretization of the parameters used. While using this Hough technique will decrease the time spent in finding the best-fitting plane, it is also true that it is at the cost of finding the best-fitting plane from only a discrete set of planes.

Next, for each grid point, a quadratic patch is fit to the points of the planar patches at this grid point and the planar patches of neighboring grid points. Therefore, the quadratic patches are fit to a larger local region of the scene; this is acceptable since the previous fittings of planar patches would have removed many of the ambiguous matches. This time a least-squares method is used, but only the two most compatible sets of the planes covering this larger region are used as input to the quadratic patch fitting procedure. Two neighboring planar patches are compatible if the differences in depth and orientation between them are less than empirically chosen thresholds. Only the two most compatible sets of planes in the region are used, and consequently the exponential time complexity that can occur for surface fitting is avoided. The quadratic patch with the minimum squared error that also is less than an empirically determined threshold is taken as the best-fitting quadratic patch. If the quadratic patch with the minimum squared error does not pass the threshold test, then no patch is fit to the corresponding grid point.

After quadratic fitting, all of the ambiguous matches are resolved. Before a final global surface interpolation can take place, the occluding and ridge contours in the scene must be detected. This is performed by fitting bipartite circular planar patches to each grid point. Bipartite circular planar patches are formed by dividing a circular patch into two halves by cutting along a diameter at a particular orientation. If the two planes differ in depth or orientation by more than some threshold, then an occluding or ridge edge may exist along the split circular patch's diameter. The final surface interpolation follows the calculation of a weighted average of the surrounding quadratic surface points at each grid point where the weights reflect the distance from the surrounding surface point to the center grid point in question. This interpolation step does not cross edges when averaging in the surrounding surface points.

The main disadvantage of using surface reconstruction techniques at every point in the scene is the computational expense. For example, for

the finest LOG channel for a 512×512 pixel image, it took this system approximately 3 hours to perform planar patch fitting on a Sun 3/160 workstation. The authors suggest special dedicated hardware to implement the Hough transform as a way of significantly improving this time and parallel processing techniques where each processor can be assigned the task of a single planar patch fitting. In addition, this system also runs the risk of spending too much time performing surface fitting in areas where no surfaces exist or where background surfaces exist which may be unimportant to the subsequent recognition tasks. As suggested by the Tanaka–Kak system (44), it may be desirable to perform surface fitting only over local areas that are most likely to contain semantically significant object surfaces such as the areas surrounding straight line features.

5. DEPTH RESOLUTION AND DESIGN OF STEREO CAMERA CONFIGURATION

The process of designing a stereo camera configuration is a function of many considerations such as the desired depth resolution, the typical scene depths, and the portion of the scene you wish to capture. The parameters of a stereo camera configuration are the baseline length b, the focal length of the cameras f, the angle of the optical axes, and the image sampling intervals in the images. We assume that the cameras have the same focal length and sampling intervals simply because there is no reason for these parameters to differ, and if they did, it would complicate feature extraction and matching. In the following paragraphs, we will attempt to elucidate the interdependencies of the system parameters with the task-dependent considerations.

In Section 3, we briefly mentioned two types of configurations: the canonical and convergent camera configurations shown in Figs. 8.4 and 8.6, respectively. In the canonical configuration, the optical axes are parallel, whereas in the convergent configuration they are not parallel. Before feature matching can occur for the convergent case, a reprojection process must take place as described in Section 3. This reprojection process utilizes estimates of the cameras' positions and orientations and because these are only estimates, errors can be introduced. Therefore, whenever possible the canonical configuration should be used. Unfortunately, for given b and f values chosen to achieve a desired depth resolution, the cameras may be spaced too far apart to view enough of the same portion of the scene for the particular task in question. In this case, a convergent camera system must be utilized.

A design criterion, which often is the most important, is the desired depth resolution. The depth at a point is defined as the z component of the (x, y, z) value recovered from the corresponding stereo point match as

described by Eq. (17). The depth resolution, which we will denote as Δz, is defined as the smallest detectable change in depth and is a function of the smallest detectable change in disparity. The smallest displacement of a point in an image is one pixel unless subpixel detection of points is performed. Therefore, the smallest change in disparity, which we denote as Δd, is equal to δu, the spacing between pixels, or in the case of subpixel calculations, an appropriate fraction of δu. Using Eq. (17), we can solve for Δd as follows, where d = disparity:

$$d = \frac{bf}{z}, \qquad d + \Delta d = \frac{bf}{z + \Delta z},$$

$$\Delta d = \frac{bf}{z + \Delta z} - \frac{bf}{z},$$

$$\Delta d(z + \Delta z)z = bfz - bf(z + \Delta z),$$

$$\Delta z = \frac{-\Delta d z^2}{bf + \Delta d z}. \tag{27}$$

Observe that the depth resolution is inversely proportional to b and f and directly proportional to the square of the distance to the scene. This last relationship leads to the fact that for a given b and f the depth values computed for objects that are closer are more accurate than for objects that are farther away. Another way to think about this is that closer object points will project farther away from the optical axis of each camera and thus produce a larger disparity (see Fig. 8.3). Therefore, it is not possible to calculate the depth of distant object points whose disparities are less than the smallest detectable disparity. Supposedly, this phenomenon is also responsible for the observation that, through the mechanism of binocular fusion, humans cannot perceive depth beyond approximately 60 m. Fortunately, in a camera-based system, we can set the b and f parameters so that appropriate disparity resolution is achieved for the perception of any depth; of course, some values of b and f may not be practically feasible. Given that the maximum depth value $z_{\max}$ is known, the following equation describes the bounds on Δz:

$$\frac{-z_{\max}^2 \Delta u}{bf + z_{\max} \Delta u} \leq \Delta z \leq \frac{z_{\max}^2 \Delta u}{bf - z_{\max} \Delta u} \tag{28}$$

If either the baseline length b or the focal length f is increased, then the two images will view less of the same portion of the scene. When relatively large objects populate the scene and there is a task-specific requirement that the objects be all viewable from both cameras, then bounds must be placed on how large b and f are allowed to be. The values of these two parameters are also limited by the extent of occlusion acceptable to the system.

In addition to depth resolution, there is also a similar notion of resolution in the x and y directions. However, the constraints placed on the parameters of the stereo configuration are usually not as strict as those created by the desired depth resolution. This is especially true when the x,y span of the objects is smaller than the depth of the objects.

Dhond and Aggarwal (8) derive a similar expression for Δz for the convergent camera configuration where the second camera's optical axis is rotated by a pan angle ϕ with respect to the optical axis of the first camera. This expression is as follows:

$$\frac{\Delta z}{\Delta d} = \cos^2(\phi - \beta)[A + B],$$

where

$$\beta = \tan^{-1}\left[\frac{b - (zu_1/f)}{z}\right], \qquad A = \frac{z^2}{bf}, \qquad \text{and}$$

$$B = \frac{b}{f}\left[1 - \frac{zu_1}{bf}\right]^2$$

Note that in this equation u_1 indicates the horizontal coordinate of the point in image 1 for a particular match pair. Therefore, Δz for the convergent camera setup is a function of the coordinates of the pixels used in a matched pair, besides, of course, being a function of z, b, f, and ϕ. In other words, for a given set of z, b, f, and ϕ values, Δz will vary according to the position of the edge point in image 1 that is being matched.

6. TRINOCULAR STEREO VISION

Trinocular stereo vision is a relatively new technique for depth recovery. It is similar in many ways to binocular stereo vision except for the fact that now three cameras are used. The addition of the third camera helps to reduce the ill effects of occlusion and adds additional epipolar constraints that can be used to produce more robust disparity calculations. More specifically, a third image may capture regions in the scene visible in image 1 but occluded in image 2, or vice versa. Also, the additional epipolar constraints, discussed below, can be used to reject erroneous matches that might otherwise be accepted by a binocular stereo algorithm.

Figure 8.24 illustrates a trinocular camera configuration. Notice that the scene point P is projected into each of the three image planes at p_1, p_2, and p_3, respectively. Now, given the point p_1, an epipolar line can be

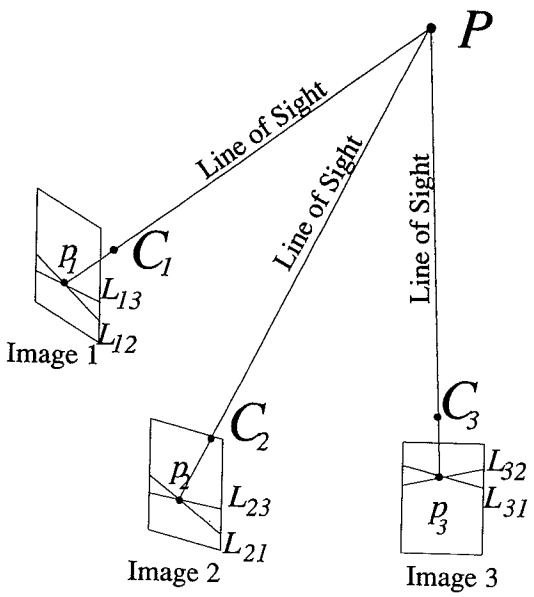

Figure 8.24. Illustration of a trinocular camera configuration.

drawn in each of the other image planes; these lines are denoted by L_{21} and L_{31} for the second and third images, respectively. Given an initial match between p_1 and p_2, the epipolar constraints dictate that the corresponding point in the third image should be located at the intersection of the epipolar lines, L_{31} and L_{32}. This extra set of epipolar constraints can be used to find the point p_3 in the third image. If such a point is found in the third image and the local properties of this point are similar to those observed at p_1 and p_2, then (p_1, p_2, p_3) can be accepted as a triplet of candidate matching points.

An initially matched pair p_1, p_2 is selected in a manner similar to how matched pairs are formed for regular binocular stereo, that is, by using local properties to establish similarity between the two points. If two points p_1 and p_2 are strongly similar, one can go ahead and accept them for disparity calculations. But, if for point p_1 there exist multiple candidates in the second image, searching in the third image for the corresponding point p_3 with similar properties can be used as a method of disambiguation. In this way, trinocular matching can reduce the number of mismatches that may result from a binocular stereo algorithm.

Given a triplet of matches, (p_1, p_2, p_3), the depth of the corresponding scene point is ideally recovered by finding the 3D intersection point of the three lines of sight (see Fig. 8.24). A line of sight for the point p_i is the 3D line that goes through the point p_i on the image plane and the focal point C_i. Because we are dealing with discrete images, the three lines may not intersect at a single point. Therefore, the corresponding scene point is taken to be the 3D point that minimizes the sum of the squared distances

to the three lines of sight. Since we are using more than the two lines of sights available to a binocular stereo vision system, it can be said that the calculation of a 3D point in the scene will be on average more accurate.

In the next subsection, we present two trinocular vision algorithms, one that utilizes edgels (edge points) and the other, which uses straight lines as features. In the following subsection, we discuss the rectification of the three trinocular images that result in the epipolar lines being parallel to the axes of the image coordinate frames. In the last subsection, we will discuss the added computational complexity of trinocular vision with respect to binocular stereo vision and also discuss the quantitative improvement in depth recovery.

6.1. Example Trinocular Stereo Vision Systems

In this section, we present two algorithms that are indicative of the work that has been accomplished in trinocular stereo vision. We present two algorithms, one that uses low level features and the other, high-level features.

Ito and Ishi (17) describe a trinocular stereo vision system that uses edgels (edge points) as features for matching. In this system, the edge image extracted from one of the images is treated as a "base" image. The goal is to recover the depth of all of the points in the "base" image. Unlike the more recent trinocular vision systems, image rectification is not performed, and thus the epipolar lines will generally not fall along the scan lines of the images, as shown in Fig. 8.24. The next step in their procedure is to determine all of the epipolar lines corresponding to the edgels in the base image. Now, the matching procedure outlined below is invoked where image 1 is the base image. In the following procedure, a two-sided correlation coefficient of a candidate match between two points p_a and p_b is defined as a function of the correlation of the gray-scale information in the neighborhoods surrounding each point. A large correlation coefficient indicates that the matched points have neighborhoods with similar gray-scale content. Similarly, a one-sided correlation coefficient is defined as a function of the correlation of the gray-scale information only on one side of the neighborhoods that surround the candidate matching points. The reader is referred to Ito and Ishi (17) for further implementation details.

1) For each $p_1 \in$ image 1 find all of the candidate matches, p_2^i, in image 2 that have 2-sided correlation coefficients above a certain threshold.

For each $(p_1, p_2^i) \in$ {Matches to p_1}

{

Find the corresponding point, p_3^i, in image 3 using epipolar constraints. Compute the 2-sided correlation coefficient of the points p_1 and p_3^i

}

Retain the triplet match (p_1, p_2^i, p_3^i) which has the largest correlation coefficient

2) For all unmatched $p_1 \in$ Image 1 find all of the candidate matches, p_3^i, in image 3 that have 2-sided correlation coefficients above a certain threshold.

For each $(p_1, p_3^i) \in$ {Matches to p_1}

{

Find the corresponding point, p_2^i, in image 2 using epipolar constraints. Compute the 2-sided correlation coefficient of the points p_1 and p_2^i

}

Retain the triplet match (p_1, p_3^i, p_2^i) which has the largest correlation coefficient

3) Do the same as step 1 except use 1-sided correlation coefficients where necessary.

4) Do the same as step 2 except use 1-sided correlation coefficients where necessary.

5) For any points in image 1 that are unmatched, use a binocular stereopsis algorithm with the other images to produce matches. The search region along an epipolar line is bounded using the information of the surrounding disparity values calculated from matches previously found in steps 1–4.

It is interesting to note that through the actions of steps 3–5 the ill effects of occlusion may be reduced. Consider the situation in Fig. 8.25, where in image 1 the point p_1, because of occlusion, can be compared only in terms of one side of its neighborhood with one side of the neighborhoods of the corresponding points p_2 and p_3 of images 2 and 3, respec-

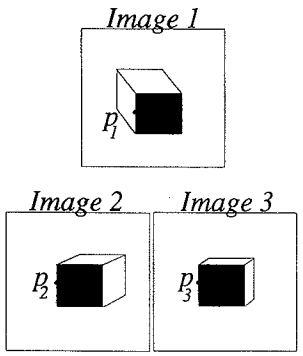

Figure 8.25. In this scene, the neighborhood of point p_1 can only be compared on one side of the neighborhoods of the points p_2 and p_3.

tively. Steps 3 and 4, perform a one-sided correlational test that will result in this triplet of points forming a match. Observe in Fig. 8.26, that the point p_1 has a corresponding point in image 3 but none in image 2. This is a result of occlusion and is handled by step 5 of the algorithm.

In (2, 14), a trinocular stereo vision algorithm is described that matches straight line segments. This algorithm uses different combinations of the images to set up the candidate matches. This helps to reduce the occurrence of missed matches that can result from the fact that one segment in an image may correspond to more than one segment in another image as a result of occlusion and other factors (14). Below is an outline of the algorithm called, `Trinocular`.

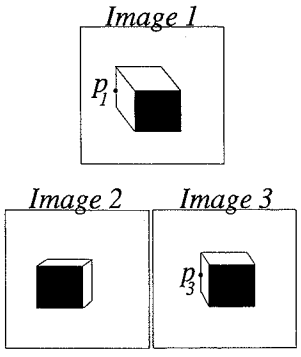

Figure 8.26. In this scene, the point p_1 has a corresponding point in image 3 but not in image 2 because of occlusion.

```
Trinocular

{
  Matcher(1,2,3)
  Matcher(2,1,3)
  Matcher(3,2,1)
}

where

Matcher(i,j,k)

{
  For each unmatched segment S_i ∈ image i

  {
      calculate |α_i| where |α_i| is the angle formed by S_i and
      the corresponding epipolar line in image j

  if |α_i| > 45°

        Generate_Hypotheses (i,j,k,S_i)

  else

        Generate_Hypotheses (i,k,j,S_i)
 }
}

Generate_Hypotheses (i,j,k,S_i)

{
  perform a search in image j for candidate matches with
  S_i.
  if a match is found then check validity by looking for a
  corresponding segment S_k in image k.
}
```

Generate_Hypotheses (i, j, k, S_i) is the procedure that finds the candidate matches for the segment S_i in image i. Basically, this procedure operates by first calculating the midpoint, a_i, of the straight line segment, S_i (see Fig. 8.27). Then any segment S_j that intersects with the epipolar line of the point a_i in image j (denoted as L_{ji}) and also has similar geometric characteristics such as orientation and length with respect to S_i

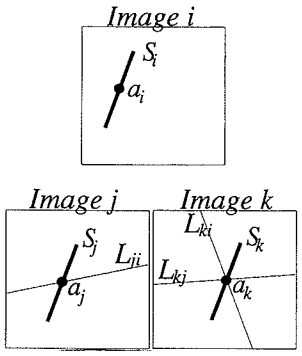

Figure 8.27. Illustration of epipolar line constraints used in the matching of segment S_i.

is considered to be a candidate match. For each candidate match (S_i, S_j), the epipolar lines L_{ki} and L_{kj}, associated with points a_i and a_j, respectively, are used to find point a_k in image 3. This point should lie on the corresponding segment in the third image. A neighborhood of 3×3 pixels is searched and if a segment, which we will call S_k, is found in this region that also has similar characteristics with respect to S_i and S_j, then the triplet (S_i, S_j, S_k) is kept as a candidate match.

As discussed in Hansen *et al.* (14), before the procedure `Trinocular` is invoked, the three images are rectified so that the epipolar lines become parallel to the axes of the image coordinate frames. This allows for fast epipolar line search. We will discuss this procedure in more detail in the next subsection.

Subsequent to the `Trinocular` procedure, the 3D line segments are reconstructed from the triple segment matches. Because each pairwise reconstruction of the 3D line from a triplet segment match will differ, the 3D line that minimizes the least-squared error is chosen. Actually, a Kalman filter approach that computes a recursive weighted least-squares solution is used. Finally, a comparison involving neighboring 3D segments is invoked to remove erroneous matches.

6.2. Trinocular Image Rectification

An example camera configuration for a trinocular vision system is shown in Fig. 8.24. Rectification of the three images is important because it allows for fast epipolar search. It is best to reproject the camera images, or the feature points extracted therefrom, into planes that would correspond

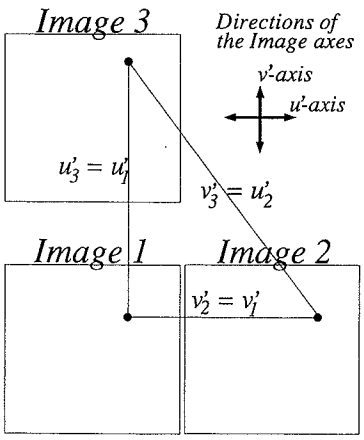

Figure 8.28. Resulting images from a trinocular system after image rectification has been performed.

to a trinocular canonical configuration (see Fig. 8.28). As a result of this reprojection, the images will be coplanar and lie in the plane containing the focal points of the three cameras. This reprojection process uses the position of the focal points of the cameras in 3D space (C), and the perspective transformation matrices T. As a result, the horizontal scan lines of images 1 and 2 will be row-registered; that is, the nth row of image 1 will be coincident with the nth row of image 2. In addition, the columns of images 1 and 3 will be registered after reprojection. Therefore, the following relations on the newly rectified images will be true where u'_i is the coordinate of a pixel along the horizontal axis of the ith image and v'_i is the coordinate of the same pixel along the vertical axis of the ith image:

$$v'_2 = v'_1, \qquad u'_3 = u'_1, \qquad v'_3 = u'_2.$$

Consequently, the epipolar line of point (u'_1, v'_1) in image 2 is the line $v'_2 = v'_1$ and in image 3 is the line $u'_3 = u'_1$. In the following reprojection process, we will denote the transformation matrix of camera i as T_i and the focal point of camera i as C_i.

For each point $p = (u, v)$ in image i, the reprojected point $p' = (u', v')$ is computed as follows:

$$\begin{bmatrix} a \\ b \\ w \end{bmatrix} = R_i \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \text{and} \qquad u' = \frac{a}{w}, \quad v' = \frac{b}{w},$$

where

$$R_i = \begin{bmatrix} (C_{i-1} \times C_i)^t \\ (C_i \times C_{i+1})^t \\ (C_1 \times C_2 + C_2 \times C_3 + C_3 \times C_1)^t \end{bmatrix} \cdot \begin{bmatrix} t_2 \times t_3 \\ t_3 \times t_1 \\ t_1 \times t_2 \end{bmatrix}^t$$

where t_j denotes the jth row of the matrix T_i. Note that in this procedure $i + 1 = 1$ if $i = 3$ and $i - 1 = 3$ if $i = 1$. The reader is referred to Ayache and Hansen (1) for details.

6.3. Comparison of Trinocular and Binocular Stereopsis

A quantitative analysis of the reduction in the number of stereo matching errors for a trinocular system over a binocular system is performed in (8). Both real digital elevation map (DEM) data and computer-generated random-dot stereograms are used. The trinocular systems use an extra image of each test scene besides the two images used by the binocular system. A subset of the data points for which ground truth is known is processed by each stereo algorithm. The percentage of mismatched points is argued by Dhond and Aggarwal (8) to be an estimate of the probability of a mismatch occuring for a scene point with no a priori knowledge of its true depth. The computational complexity of a generic trinocular stereo algorithm and a generic binocular stereo algorithm are evaluated.

The error involved in a match for which the actual 3D point is known is calculated as follows:

$$\varepsilon = |d_{\text{calc}} - d_{\text{truth}}|.$$

The value d_{calc} is the disparity calculated by the stereo algorithm, and d_{truth} is the corresponding disparity of the actual 3D point that is found by using the perspective projections of the known 3D point into images 1 and 2 and measuring the resultant displacement. It is important to note that in this study the disparity of a triplet match (p_1, p_2, p_3), produced from the trinocular algorithm, is defined to be the disparity between points p_1 and p_2. If the error ε is greater than a chosen threshold τ, the match is considered to be erroneous.

Different values of the threshold τ were used, and for values exceeding 5 pixels, the trinocular stereo algorithm cut in half the number of matching errors produced by the binocular stereo algorithm. This was true for both the DEM data and the random-dot stereograms. Even for smaller

disparity thresholds, the trinocular stereo algorithm always produced a smaller number of matching errors. It should be emphasized that the depth in a trinocular stereo system is usually calculated using all three points in the match triplet, and as such the error in the depth values may be less on average than if only two of the points are used as was performed in this study.

Dhond and Aggarwal (8) also evaluate the computational complexity of a generic trinocular stereo algorithm and of a generic binocular stereo algorithm in terms on the number of multiplications (M), additions (A), and comparisons (C) for each matched point in image 1. The following numbers were obtained:

Binocular 76M, 130A, 110C

Trinocular 96M, 162A, 115C

Thus, in the worst case the increase in computational cost of the trinocular algorithm over the binocular algorithm is approximately one-fourth.

In conclusion, if an increase in computational complexity of one-fourth is acceptable, an decrease of up to 50% in the number of matching errors can be obtained by using a trinocular stereo vision system. It should be stressed that these performance measures were calculated using the particular trinocular and binocular stereo algorithms in Dhond and Aggarwal (8) and that for other algorithms these values may change.

7. ILLUMINATION EFFECTS

In this section, we will briefly discuss the effects of scene illumination on the stereo vision process. In addition, we will discuss the use of a special lighting technique referred to as *unstructured lighting*.

The locations of the scene light sources will effect the edge information that is extracted from a view of the scene. For example, consider Figs. 8.29(A–D), where the light sources used are placed at four different angles. Each image in Fig. 8.29 shows only the strongest zero-crossings detected in the image. Notice that a different set of zero-crossing contours appear in each image. In some cases, contours exist in one image, but not in another image. In other cases, the shapes of the corresponding contours have altered between images. From the fact that different edge images will result from the movement of the light sources, we can deduce that for different views, different edge patterns in the commonly viewed scene regions may result.

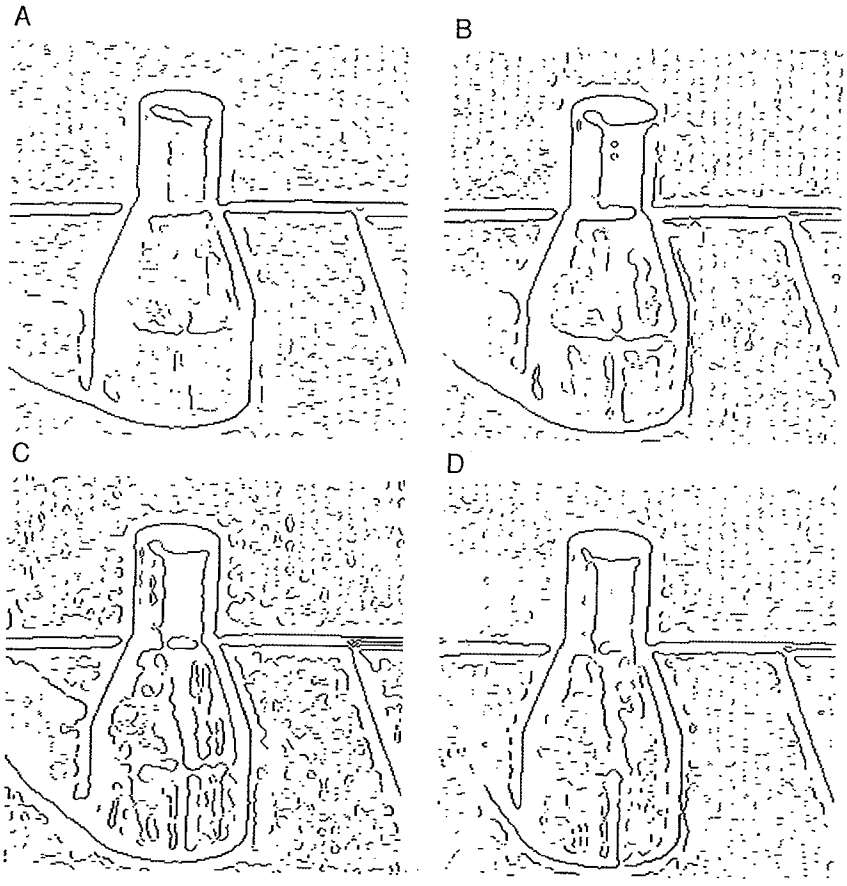

Figure 8.29. Zero-crossings extracted from the same scene with the illumination sources at different angles: (A) illumination 1; (B) illumination 2; (C) illumination 3; (D) illumination 4.

Most of the features used in stereo vision algorithms are constructed from the edgels detected in the images of the scene. To produce dense depth maps, it would be ideal to have features everywhere in the image. Unfortunately, in areas of approximately uniform gray level, few edge points will be detected. For example, consider Fig. 8.30(A), which shows a gray-level image of a scene and notice that there is little gray-level variation in the surfaces in the image. In Fig. 8.30(B), the corresponding zero-crossing image is shown, and as we expect, there are very few zero-crossings detected inside of the fairly uniform gray-level regions of the scene.

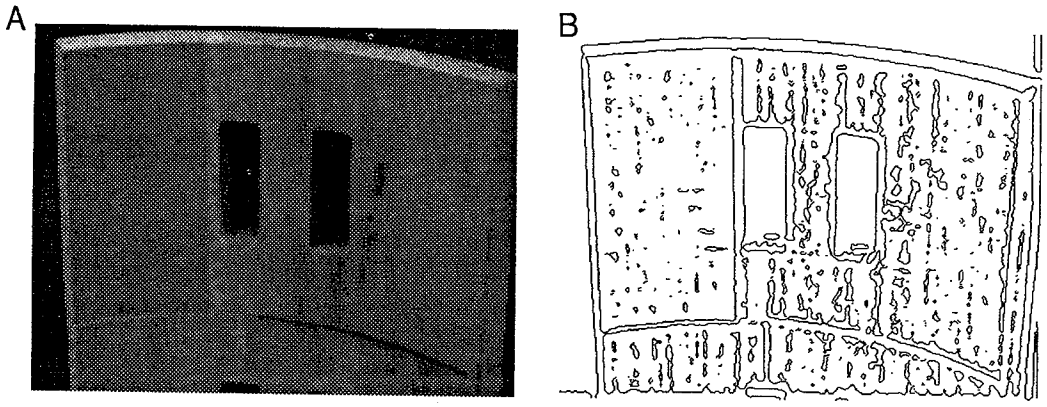

Figure 8.30. (A) Gray-scale image of objects against a black background; (B) zero-crossings extracted from the LOG-filtered image with $w_{2D} = 8$.

One lighting technique, called *unstructured* or *textured lighting*, can be used to induce edge features in regions that would appear to have constant gray levels under normal room lighting. Unstructured lighting of a scene is accomplished by illuminating the scene with a randomly textured pattern of light. Such a textured pattern of light can be produced by illuminating the scene with a slide projector where the slide is an image of a random-dot gray-scale pattern. Figure 8.31(A) shows the scene in Fig. 8.30(A) but with unstructured lighting, and Fig. 8.31(B) illustrates the resulting increase in the number of zero-crossings detected. Figure 8.32 shows the results of stereopsis on both the normal and unstructured illumination of a scene. As in the previous example, superior results are obtained for the unstructured lighting case.

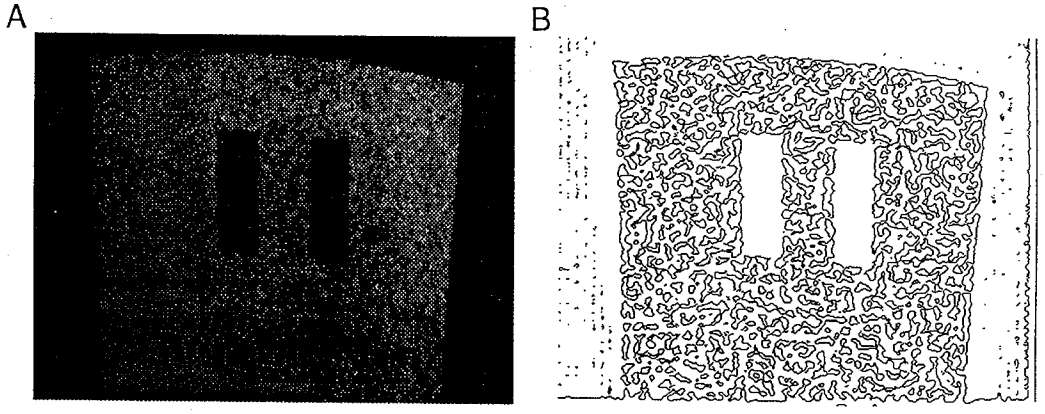

Figure 8.31. (A) Objects of Fig. 8.30(A) illuminated with unstructured light; (B) zero-crossings extracted from the LOG-filtered image with $w_{2D} = 8$.

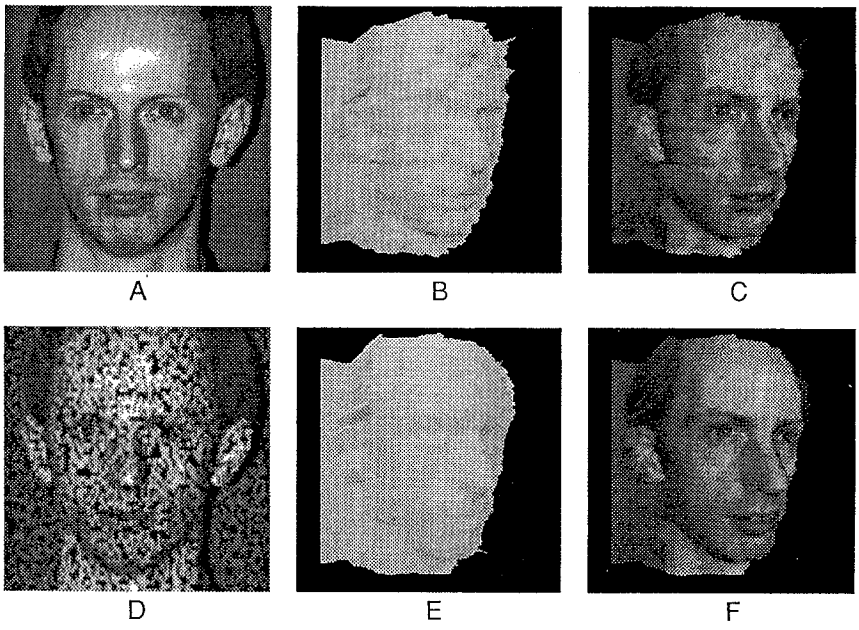

Figure 8.32. (A) Left image under normal illumination; (B) recovered depth map (for normal illumination case) rendered with Gouraud shading; (C) recovered depth map (for normal illumination case) rendered with texture mapping; (D) left image under unstructured illumination; (E) recovered depth map (for unstructured illumination case) rendered with Gouraud shading; (F) recovered depth map (for unstructured illumination case) rendered with texture mapping. Images were supplied by P. Siebert and C. Urquhart at the Turing Institute Ltd. in Glasgow, UK (under the DTI project IED3/1/2109). See Siebert and Urquhart (41) for a previous contribution from the authors on the use of unstructured lighting.

8. CORRECTION AND ANALYSIS OF ERRORS

The errors that can occur in a stereo vision system can be classified into three types. The first, which we will call *quantization error*, is a consequence of the fact that images are 2D discrete signals and thus a scene point is projected to a pixel in the image. Thus, a scene point's location can be displaced by as much as $\pm \frac{1}{2}$ pixel. The second type of error, which we will refer to as *camera distortion error*, occurs as a result of camera sensor and lens distortions. This type of error is difficult to model because they are systematic and not random. More specifically, it is a function of the particular camera or lens that is being used. The last type of error we refer to as *matching error*, which results from incorrect matching of stereo

features. In this section, we will discuss each type of error and its effects. In addition, we discuss methods for correcting these errors.

8.1. Quantization Errors

Quantization error is defined as the change in the calculated depth value at a point as a function of the corresponding change in the disparity value on account of the quantization or digitization of the images. Equation (27) describes this relation, where Δd is the change in disparity and Δz is the quantization error. While Eq. (27) is useful, the quantization error can also be modeled stochastically. In the next few paragraphs, we present the stochastic model as derived in Rodrigues and Aggarwal (39).

We can consider Δz to be a function of the random variables Δd and z in Eq. (27). We assume a canonical camera configuration as shown in Fig. 8.4. Thus $\Delta d = \Delta x_l - \Delta x_r$, where Δx_l is the quantization error of the x_l position and Δx_r is the quantization error of the x_r position. We assume that Δx_l and Δx_r are independent of each other and of z. This assumption is true if the disparity $(x_l - x_r)$ is not too small (39). In addition, Δx_l and Δx_r are assumed to be uniformly distributed between $-\delta u/2$ and $\delta u/2$, which corresponds to a displacement of up to $\pm \frac{1}{2}$ pixel. The probability density functions (pdf) of Δx_l and Δx_r are

$$f_{\Delta x_l}(\Delta x_l) = \frac{1}{\delta u}, \quad \text{for} \quad -\frac{\delta u}{2} \leq \Delta x_l \leq \frac{\delta u}{2};$$

$$f_{\Delta x_r}(\Delta x_r) = \frac{1}{\delta u}, \quad \text{for} \quad -\frac{\delta u}{2} \leq \Delta x_r \leq \frac{\delta u}{2} \tag{29}$$

Using the fact that the pdf of the sum of two independent variables is the convolution of their pdfs we can derive the following:

$$f_{\Delta d}(\Delta d) = f_{\Delta x_l}(\Delta x_l) * f_{\Delta x_r}(\Delta x_r)$$

$$= \begin{cases} \dfrac{\delta u + \Delta d}{(\delta u)^2} & -\delta u \leq \Delta d \leq 0 \\ \dfrac{\delta u - \Delta d}{(\delta u)^2} & 0 \leq \Delta d \leq \delta u \end{cases} \tag{30}$$

Next, using Eq. (27) and (30) we can show that the following is true:

$$f_{\Delta z}(\Delta z \mid z) = f_{\Delta d}(\Delta d)\left|\frac{d(\Delta d)}{d(\Delta z)}\right|$$

$$= \frac{bf}{(z+\Delta z)^2}\begin{cases}\dfrac{\delta u + \Delta d}{(\delta u)^2} & -\delta u \le \Delta d \le 0 \\ \dfrac{\delta u - \Delta d}{(\delta u)^2} & 0 \le \Delta d \le \delta u\end{cases}$$

Our goal is to find the pdf of Δz, which we do using the above conditional probability as follows:

$$f_{\Delta z}(\Delta z) = \int_{-\infty}^{\infty} f_{\Delta z}(\Delta z \mid z) f_z(z)\, dz$$

$$= \begin{cases}\dfrac{bf}{\delta u^2}\displaystyle\int_{-\infty}^{\infty} \frac{z^2\,\delta u + z\,\delta u\,\Delta z + bf\,\Delta z}{z(z+\Delta z)^3} g^-(z) f_z(z)\, dz & -\delta u \le \Delta d \le 0 \\ \dfrac{bf}{\delta u^2}\displaystyle\int_{-\infty}^{\infty} \frac{z^2\,\delta u + z\,\delta u\,\Delta z - bf\,\Delta z}{z(z+\Delta z)^3} g^+(z) f_z(z)\, dz & 0 \le \Delta d \le \delta u\end{cases} \tag{31}$$

where

$$g^+(z) = \begin{cases}1 & \text{if } -\delta u \le \Delta d \le 0 \\ & \quad \text{equivalently } z \ge \dfrac{-\delta u\,\Delta z + \sqrt{\delta u^2\,\Delta z^2 + 4bf\,\delta u\,\Delta z}}{2\delta u} \\ 0 & \text{otherwise}\end{cases},$$

$$g^-(z) = \begin{cases}1 & \text{if } 0 \le \Delta d \le \delta u \\ & \quad \text{equivalently } z \ge \dfrac{-\delta u\,\Delta z + \sqrt{\delta u^2\,\Delta z^2 - 4bf\,\delta u\,\Delta z}}{2\delta u} \\ 0 & \text{otherwise}\end{cases}.$$

Notice that $f_z(z)$ is the pdf of the random variable z and this is domain-dependent. Assuming that the distribution of depth values in the

scene is uniformly distributed between $z_{\min}$ and $z_{\max}$, the following is true:

$$f_z(z) = \frac{1}{z_{\max} - z_{\min}}, \qquad z_{\min} \le z \le z_{\max}.$$

After plugging this into Eq. (31) and integrating, we obtain

For $0 < \Delta z \le z_{\max}^2 \dfrac{\delta u}{bf - z_{\max}\delta u}$,

$$f_{\Delta z}(\Delta z) \doteq \frac{bf}{2\delta u^2 \Delta z^2 (z_{\max} - z_{\min})} \times \left[\frac{4bfz - 2\delta u \Delta z^2}{z + \Delta z} - \frac{bfz^2}{(z + \Delta z)^2} + 2bf \ln\left(1 + \frac{\Delta z}{z}\right) \right]_{z_0}^{z_2}; \tag{32}$$

For $-z_{\max}^2 \dfrac{\delta u}{bf + z_{\max}\delta u} \le \Delta z < 0$,

$$f_{\Delta z}(\Delta z) = \frac{bf}{2\delta u^2 \Delta z^2 (z_{\max} - z_{\min})} \times \left[\frac{bfz^2}{(z + \Delta z)^2} - \frac{4bfz - 2\delta u \Delta z^2}{z + \Delta z} - 2bf \ln\left(1 + \frac{\Delta z}{z}\right) \right]_{z_1}^{z_2}; \tag{33}$$

For $\Delta z = 0$,

$$f_{\Delta z}(0) = \frac{bf}{z_{\min} z_{\max} \delta u}, \tag{34}$$

where

$$z_2 = z_{\max} \quad \text{and} \quad z_0 = \max\{z_{\min}, z^+\} \quad \text{and} \quad z_1 = \max\{z_{\min}, z^-\}$$

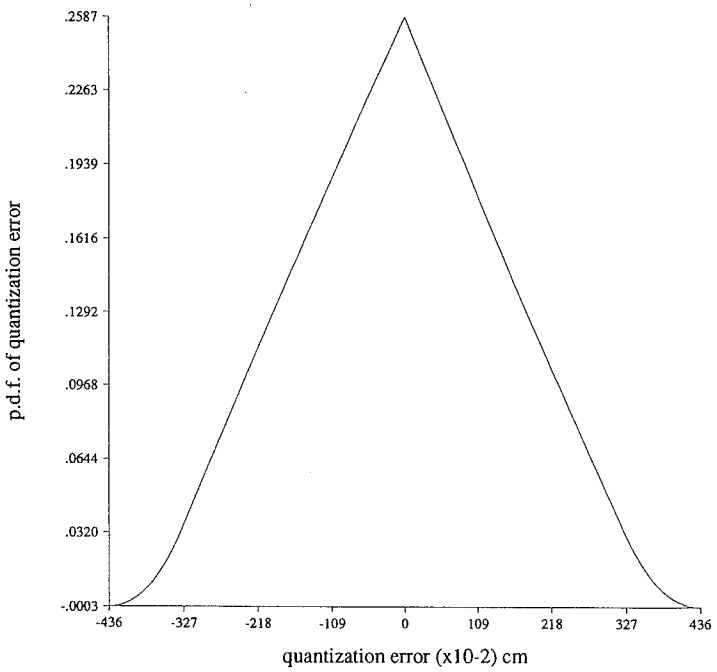

Figure 8.33. Plot of the probability density function of the quantization error.

and

$$z^{+} = \frac{-\delta u z + \sqrt{\delta u^2 \Delta z^2 + 4bf\,\delta u\,\Delta z}}{2\delta u}$$

$$z^{-} = \frac{-\delta u z + \sqrt{\delta u^2 \Delta z^2 - 4bf\,\delta u\,\Delta z}}{2\delta u}$$

Figure 8.33 is a plot of the pdf, $f_{\Delta z}$, where the following parameters were used: $b = 53.8$ cm, $f = 16$ mm, $z_{\min} = 187.49$ cm, $z_{\max} = 158.79$ cm, $\delta u = 0.00153$ cm.

In addition to the pdf of the quantization error, the expected value of the magnitude of the quantization error was derived in Rodrigues and Aggarwal (39). The following is an approximation to this expected value under the assumption that the depth is uniformly distributed between $z_{\min}$ and $z_{\max}$:

$$E[\,|\Delta z|\,] = \frac{\delta u}{9bf}\left(z_{\min}^2 + z_{\min} z_{\max} + z_{\max}^2\right) \tag{35}$$

For the stereo parameters used to create Fig. 33, the expected value is 1.297 cm.

There are three ways to reduce quantization error. The first is to perform subpixel detection of features. Unfortunately, subpixel detection of features, as mentioned before, is appropriate only when the features are relatively well separated. Second, as discussed in Section 5, the camera configuration parameters can be altered, and if possible the cameras could be moved closer to the objects in the scene. Finally, a practical solution would be to use a high-resolution camera where the pixel size δu is smaller. In any case, there will always be some quantization error existent in the system.

8.2. Camera Distortion Errors

We have defined this type of error to be a consequence of the distortions present in the lens and camera sensors. This type of error will vary from camera to camera and lens to lens. A practical solution to reducing camera distortion errors is to buy a fine-quality lens and camera. One place in which the effect of camera distortion errors can be reduced is in the reprojection process of the stereo images that is performed for a convergent camera configuration (see Section 3). The reprojection process uses information derived in camera calibration. Typically most systems use the pinhole model of the camera for calibration. However, other models can be used, such as the one described in (45), which incorporates radial lens distortion and therefore produces more accurate calibration results. In practice, camera distortion errors in general do not contribute as much to the total error as do the quantization and matching errors.

8.3. Matching Errors

Unlike quantization errors, the errors that result from mismatching have not been adequately modeled since they can result from a number of very different factors that are difficult to identify. Examples of these factors are missing features in one image, and the local invalidity of the particular matching constraints used such as a regional disparity continuity constraint. Therefore, instead of pursuing the difficult task of modeling the statistics of these errors, researchers have attempted to develop algorithms to detect matching errors. Often the goodness of a stereo matching algorithm is evaluated by counting the number of incorrect matches formed for different scenes.

In some sense, one can think of the algorithms used for disambiguation of matches as algorithms that detect and correct matching errors. However, in this section we will concentrate on methods that detect and correct matching errors after a unique disparity map has been produced.

One method of detecting and correcting matching errors uses domain-dependent knowledge in the guise of surface reconstruction. As discussed in Section 4.3.2, if surface boundaries can be found (for either local surface patches or entire surface regions) and the type of the underlying surface can be recognized, surface reconstruction can take place in these bounded regions. Points that are outliers with respect to the best-fit surface are replaced by the corresponding disparity values of the fit surface. An outlier is a point whose distance to the fitted surface is too large, ostensibly a result of mismatching. In the same manner, holes in the disparity map can be filled in. This procedure will work only if surfaces exist in the scene and they can be detected, meaning that the physical discontinuities of the scene should be detectable. In addition, the type(s) of the surfaces must be predefined, and as such this method will work in only some domains. The reader is referred to Cochran *et al.* (7) and Sinha *et al.* (42, 43) for details on some surface reconstruction techniques that have been used on stereo data.

Another method is described in Mohan *et al.* (32) using the figural continuity constraint to correct mismatches along contours. This algorithm will work on the output of any low-level-feature-based stereo algorithm. It is important to note that even if figural continuity is used as a constraint in a low-level-feature-based algorithm, it will only be used locally and thus a long contour can have some of its points mismatched. Mohan *et al.* (32) classify matching errors along contours into two types. Type 1 errors correspond to matching errors on a contour where the majority of matches on this contour are correct. Type 2 errors on a contour indicate the situation where the majority of the matches on the contour are incorrect. On the basis of figural continuity alone, only type 1 errors can be detected and corrected. Unfortunately, this algorithm runs the risk of trying to correct type 2 errors and thus producing contours with an even larger majority of erroneous matches.

In Mohan *et al.* (32), only linear segments are used. The fact that the disparity along a pair of matched linear segments varies linearly in proportion to the length of the segment is used as a constraint. First, the linear segments in each image are detected. For each linear segment, a 2D plot of all of its matched points is constructed. The disparity of each matched point on the segment is plotted as a function of its distance to the end of the segment. Next, for each such plot a straight line is fit to the plot since it is known that in the absence of matching errors the plotted points should form a straight line. This fitting is accomplished via a modified Hough transform, and the reader is referred to Mohan *et al.* (32) for

details. This best-fit line is then used to calculate with subpixel accuracy the disparity at each point along the line, and these are the values retained in the final disparity map.

One advantage of this method is that it has a linear time complexity, $O(N)$, where N is the number of line segments in the two images. A disadvantage of this algorithm is that erroneous matches along straight lines only will be corrected. In addition, the number of type 2 errors along a contour may increase if originally a sufficiently large number of type 2 errors form a straight line in the 2D disparity-length plot.

ACKNOWLEDGMENT

Work in section 4.3.1 was supported by Hitachi Construction Machinery Corporation, Limited and in collaboration with Zhaohui Li.

REFERENCES

1. N. Ayache and C. Hansen, Rectification of images for binocular and trinocular stereovision. *Proc. Int. Conf. Pattern Recognition, 9th*, Rome, Italy, *1988*, pp. 11–16 (1988).
2. N. Ayache and F. Lustman, Fast and reliable passive trinocular stereovision. *Proc. Int. Conf. Comput. Vision, 1st*, London, *1987*, pp. 422–427 (1987).
3. D. Ballard and C. Brown, "Computer Vision." Prentice-Hall, Englewood Cliffs, NJ, 1982.
4. S. Barnard and W. Thompson, Disparity analysis of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-2**(4), 333–340 (1980).
5. D. Baunegg, Stereo feature matching in disparity space. *Proce IEEE Int. Conf. Robotics Autom.*, Cincinnati, OH, *1990*, pp. 796–803 (1990).
6. K. Boyer and A. Kak, Structural stereopsis for 3-D vision. *IEEE Trans. Pattern Anal. Mach. Intell.* **10**(2), 144–166 (1988).
7. S. Cochran, G. Medioni, and R. Nevatia, Correcting matches and inferring surface patches inpassive stereo. *Proc. IEEE Comput. Soc. Workshop Comput. Vision*, Miami Beach, FL, *1987*, pp. 333–335 (1987).
8. U. Dhond and J. Aggarwal, A cost-benefit analysis of a third camera for stereo correspondence. *Int. J. Comput. Vision* **6**(1), 39–58 (1991).
9. M. Fleck, A topological stereo matcher. *Int. J. Comput. Vision* **6**, 197–226 (1991).
10. S. Ganapathy, Decomposition of transformation matrices for robot vision. *Proc. IEEE Int. Conf. Robot. Autom., 1984*, pp. 130–139 (1984).
11. W. E. L. Grimson, A computer implementation of a theory of human stereo vision. *Philos. Trans. R. Soc. London, Ser. B* **292**, 217–253 (1981).
12. W. E. L. Grimson, Computational experiments with a feature based stereo algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-7**(1), 17–34 (1985).
13. W. E. L. Grimson, "From Images to Surfaces: A Study of the Human Early Visual System." MIT Press, Cambridge, MA, 1981.
14. C. Hansen, N. Ayache, and F. Lustman, Towards real-time trinocular stereo. *Proc. Int. Conf. Comput. Vision, 2nd*, Tampa, FL, *1988*, pp. 129–133 (1988).

15. W. Hoff and N. Ahuja, Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection. *IEEE Trans. Pattern Anal. Mach. Intel.* **PAMI-11**(2), 121–136 (1989).
16. R. Horaud and T. Skordas, Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-11**(11), 1168–1180 (1989).
17. M. Ito and A. Ishi, Three-view stereo analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**(4), 524–532 (1986).
18. B. Julesz, Binocular depth perception of computer-generated patterns. *Bell Syst. Tech. J.* **39**, 1125–1162 (1960).
19. B. Julesz and J. Chang, Interaction between pools of binocular disparity detectors tuned to different disparities. *Biol. Cybernet.* **22**, 107–119 (1976).
20. A. C. Kak, Depth perception for robots. *In* "Handbook of Industrial Robotics" (S. Y. Nof, ed.) Chapter 16. Wiley, New York, 1985.
21. Y. C. Kim and J. K. Aggarwal, Positioning three-dimensional objects using stereo images. *IEEE J. Robotics Autom.* **RA-3**(4), 361–373 (1987).
22. A. Kosaka and A. Kak, A useful generalization of the theory of structural stereopsis for 3D robot vision. Submitted.
23. H. Lim and T. Binford, Stereo vision correspondence: A hierarchical approach. *Proc. DARPA Image Understanding Workshop*, Los Angeles, *1987*, pp. 234–240 (1987).
24. S. Lloyd, Stereo matching using intra- and inter-row dynamic programming. *Pattern Recognition Lett.* **4**, 273–277 (1986).
25. C. Lopez-Abadia and A. Kak, "Vision-Guided Mobile Robot Navigation," Tech. Rep. TR-EE 89-34. Purdue University, West Lafayette, IN, 1989.
26. D. Marr and T. Poggio, A computational theory of human stereo vision. *Proc. R. Soc. London, Ser. B* **204**, 301–328 (1979).
27. D. Marr and E. Hildreth, Theory of edge detection. *Proc. R. Soc. London, Ser. B* **207**, 187–217 (1980).
28. D. Marr and T. Poggio, A theory of human stereopsis. *Proc. R. Soc. London, Ser. B* **204**, 301–328 (1979).
29. D. Marr and T. Poggio, Cooperative computation of stereo disparity. *Science* **194**, 283–287 (1976).
30. J. E. W. Mayhew and J. P. Frisby, Psychophysical and computational studies towards a theory of human stereopsis. *Artif. Intell.* **17**, 349–385 (1981).
31. G. Medioni and R. Nevatia, Segment-based stereo matching. *Comput. Vision, Graphics Image Process.*, **31**, 2–18 (1985).
32. R. Mohan, G. Medioni, and R. Nevatia, Stereo error detection, correction, and evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* **11**(2), 113–120 (1989).
33. R. Mohan and R. Nevatia, Perceptual organization for scene segmentation and description. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(6), 616–633 (1992).
34. N. Nasrabadi and Y. Liu, Stereo vision correspondence using a multichannel graph matching technique. *Image Vision Comput.* 7(4), 237–245 (1989).
35. Y. Ohta and T. Kanade, Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Anal. Mach. Intell.* 7(2), 139–154 (1985).

36. Y. Ohta and T. Kanade, A high speed stereo matching system based on dynamic programming. *Proc. Int. Conf. Comput. Vision, 1st*, London, *1987*, pp. 335–342 (1987).
37. S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby, PMF: A stereo correspondence algorithm using a disparity gradient limit. *Perception* **14**, 449–470 (1981).
38. K. Prazdny, Detection of binocular disparities. *Biol. Cybernet.* **52**, 93–99 (1985).
39. J. Rodrigues and J. Aggarwal, Stochastic analysis of stereo quantization error. *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(5), 467–470 (1990).
40. A. Rosenfeld and A. C. Kak, "Digital Picture Processing." Academic Press, New York, 1982.
41. J. Siebert and C. Urquhart, Active stereo: Texture enhanced reconstruction. *Electron. Lett.* **26**(26), 427–429 (1990).
42. S. Sinha, S. Moezzi, and B. Schunck, Robust stereo vision. *Proc. SPIE — Opt. Illum., Image Sens. Mach. Vision V* **1835**, 259–266 (1990).
43. S. Sinha, S. Moezzi, and B. Schunck, A two-stage algorithm for discontinuity-preserving surface reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* **14**(6), 36–55 (1992).
44. S. Tanaka and A. C. Kak, A rule-based approach to binocular stereopsis. *In* "Analysis and Interpretation of Range Images" (R. C. Jain and A. K. Jain, eds.), Chapter 2. Springer-Verlag, Berlin, 1990.
45. R. Y. Tsai, A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE J. Robotics Autom.*, **RA-3**(4), 323–344 (1987).