Computer Vision

A MODERN APPROACH

Second Edition

Forsyth • Ponce

ALWAYS LEARNING

PEARSON

COMPUTER VISION
A MODERN APPROACH

SECOND EDITION

DAVID A. FORSYTH
University of Illinois at Urbana-Champaign

JEAN PONCE
Ecole Normale Supérieure

International Edition contributions by

SOUMEN MUKHERJEE
RCC Institute of Information Technology

ARUP KUMAR BHATTACHARJEE
RCC Institute of Information Technology

PEARSON

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

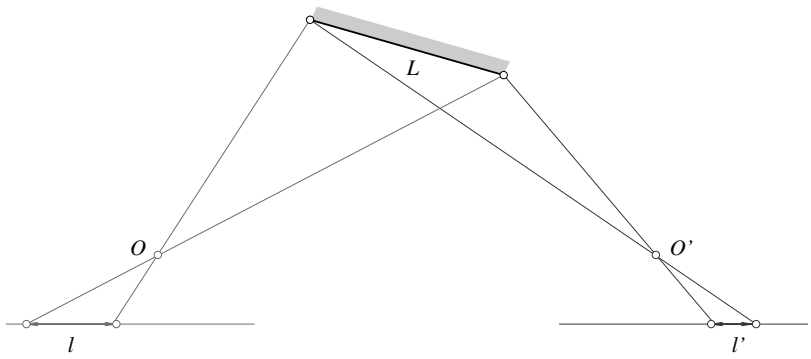

FIGURE 7.10: The foreshortening of an oblique plane is not the same for the left and right cameras: $l/L \neq l'/L$.

correlation function is equivalent to minimizing

$$|\frac{1}{||\boldsymbol{w} - \bar{\boldsymbol{w}}||}(\boldsymbol{w} - \bar{\boldsymbol{w}}) - \frac{1}{||\boldsymbol{w}' - \bar{\boldsymbol{w}}'||}(\boldsymbol{w}' - \bar{\boldsymbol{w}}')|^2,$$

or equivalently the sum of the squared differences between the pixel values of the two windows after they have been submitted to the corresponding normalization process. Second, although the calculation of the normalized correlation function at every pixel of an image for some range of disparities is computationally expensive, it can be implemented efficiently using recursive techniques (see problems). Third, other functions, such as the *sum of absolute difference* $\sum_{i=1}^{p} |w_i - w_i'|$, can be used to measure the discrepancy between two brightness patterns, and they may give better results in certain situations (Scharstein and Szeliski 2002). Finally, a major problem with correlation-based techniques for establishing stereo correspondences is that they implicitly assume that the observed surface is (locally) parallel to the two image planes, since the foreshortening of (oblique) surfaces depends on the position of the cameras observing them (Figure 7.10).

This suggests a two-pass algorithm where initial estimates of the disparity are used to warp the correlation windows to compensate for unequal amounts of foreshortening in the two pictures. For example, Devernay and Faugeras (1994) propose to define a warped window in the right image for each rectangle in the left one, using the disparity in the center of the rectangle and its derivatives. An optimization process is used to find the values of the disparity and its derivatives that maximize the correlation between the left rectangle and the right window, using interpolation to retrieve appropriate values in the right image. Figure 7.11 illustrates this approach with an example.

7.4.2 Multi-Scale Edge Matching

Slanted surfaces pose problems to correlation-based matchers. Other arguments against correlation can be found in Julesz (1960) and Marr (1982), suggesting that correspondences should be found at a variety of scales, with matches between (hopefully) physically significant image features such as edges preferred to matches be-

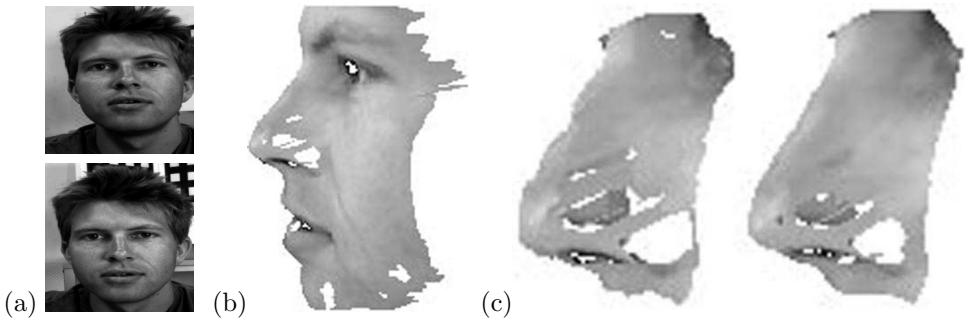

FIGURE 7.11: Correlation-based stereo matching: (a) a pair of stereo pictures; (b) a texture-mapped view of the reconstructed surface; (c) comparison of the regular (**left**) and refined (**right**) correlation methods in the nose region. The latter clearly gives better results. *Reprinted from "Computing Differential Properties of 3D Shapes from Stereopsis Without 3D Models," by F. Devernay and O.D. Faugeras, Proc. IEEE Conference on Computer Vision and Pattern Recognition, (1994).* © *1994 IEEE.*

tween raw pixel intensities. These principles are implemented in Algorithm 7.1, which is due to Marr and Poggio (1979).

1. Convolve the two (rectified) images with $\nabla^2 G_\sigma$ filters of increasing standard deviations $\sigma_1 < \sigma_2 < \sigma_3 < \sigma_4$.
2. Find zero crossings of the Laplacian along horizontal scanlines of the filtered images.
3. For each filter scale σ, match zero crossings with the same parity and roughly equal orientations in a $[-w_\sigma, +w_\sigma]$ disparity range, with $w_\sigma = 2\sqrt{2}\sigma$.
4. Use the disparities found at larger scales to offset the images in the neighborhood of matches and cause unmatched regions at smaller scales to come into correspondence.

Algorithm 7.1: The Marr–Poggio (1979) Multi-Scale Binocular Fusion Algorithm.

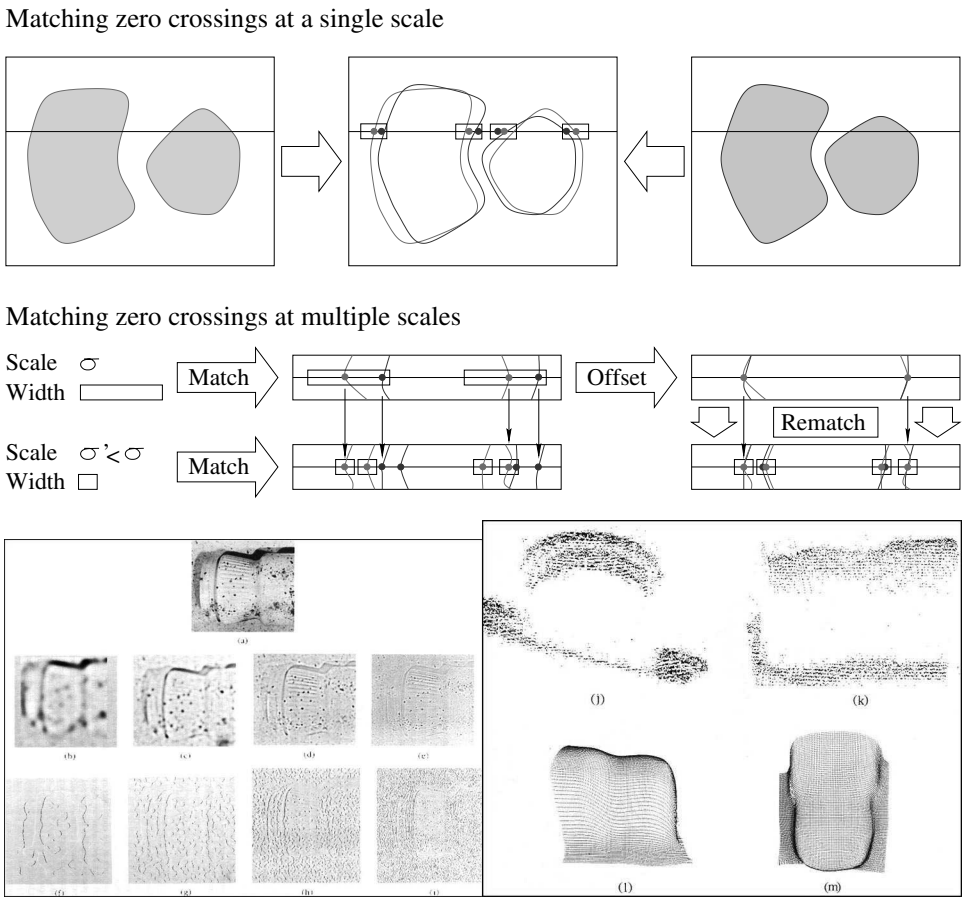

FIGURE 7.12: **Top:** Single-Scale matching. **Middle:** Multi-Scale matching. **Bottom:** Results. **Bottom left:** The input data (including one of the input pictures, the output of four $\nabla^2 G_\sigma$ filters, and the corresponding zero crossings). **Bottom right:** Two views of the disparity map constructed by the matching process and two views of the surface obtained by interpolating the reconstructed points. *Reprinted from Vision: A Computational Investigation into the Human Representation and Processing of Visual Information, by David Marr, © 1982 by David Marr. Reprinted by permission of Henry Holt and Company, LLC.*

Matches are sought at each scale in the $[-w_\sigma, w_\sigma]$ disparity range, where $w_\sigma = 2\sqrt{2}\sigma$ is the width of the central negative portion of the $\nabla^2 G_\sigma$ filter. This choice is motivated by psychophysical and statistical considerations. In particular, assuming that the convolved images are white Gaussian processes, Grimson (1981*a*) showed that the probability of a false match occurring in the $[-w_\sigma, +w_\sigma]$ disparity range of a given zero crossing is only 0.2 when the orientations of the matched features are within 30° of each other. A simple mechanism can be used to disambiguate the multiple potential matches that might still occur within the matching range. See Grimson (1981*a*) for details. Of course, limiting the search for matches to the $[-w_\sigma, +w_\sigma]$ range prevents the algorithm from matching *correct* pairs of zero crossings whose disparity falls outside this interval. Since w_σ is proportional to the scale σ at which matches are sought, eye movements (or equivalently image offsets) controlled by the disparities found at large scales must be used to bring large-disparity pairs of zero crossings within matchable range at a fine scale. This process occurs in Step 4 of Algorithm 7.1 and is illustrated by Figure 7.12 (top). Once matches have been found, the corresponding disparities can be stored in a buffer called the $2\frac{1}{2}$-*dimensional sketch* by Marr and Nishihara (1978). This algorithm has been implemented by Grimson (1981*a*), and extensively tested on random dot stereograms and natural images. An example appears in Figure 7.12 (bottom).

7.5 GLOBAL METHODS FOR BINOCULAR FUSION

The stereo fusion techniques presented in the previous section are purely local, in the sense that they match brightness or edge patterns around individual pixels, but ignore the constraints that may link nearby points. In contrast, we present in this section two *global* approaches to stereo fusion, that formulate this problem as the minimization of a single energy function incorporating *ordering* or *smoothness* constraints among adjacent pixels.

7.5.1 Ordering Constraints and Dynamic Programming

It is reasonable to assume that the order of matching image features along a pair of epipolar lines is the inverse of the order of the corresponding surface attributes along the curve where the epipolar plane intersects the observed object's boundary (Figure 7.13, left). This is the so-called *ordering constraint* introduced in the early 1980s (Baker & Binford 1981; Ohta & Kanade 1985). Interestingly enough, it might not be satisfied by real scenes, in particular when small solids occlude parts of larger ones (Figure 7.13, right) or more rarely, at least in robot vision, when transparent objects are involved. Despite these reservations, the ordering constraint remains a reasonable one, and it can be used to devise efficient algorithms relying on *dynamic programming* (Forney 1973; Aho, Hopcroft, & Ullman 1974) to establish stereo correspondences (see Figure 7.14 and Algorithm 7.2).

Specifically, let us assume that a number of feature points (say, edgels) have been found on corresponding epipolar lines. Our objective here is to match the intervals separating those points along the two intensity profiles (Figure 7.14, left). According to the ordering constraint, the order of the feature points must be the same, although the occasional interval in either image may be reduced to a single

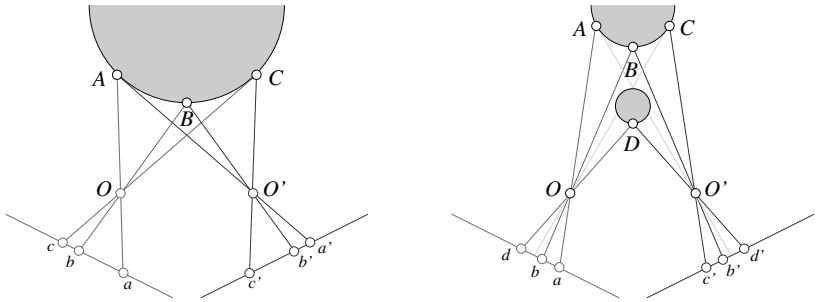

FIGURE 7.13: Ordering constraints. In the (usual) case shown in the **left** part of the diagram, the order of feature points along the two (oriented) epipolar lines is the same. In the case shown in the **right** part of the figure, a small object lies in front of a larger one. Some of the surface points are not visible in one of the images (e.g., A is not visible in the right image), and the order of the image points is not the same in the two pictures: b is on the right of d in the left image, but b' is on the left of d' in the right image.

point corresponding to missing correspondences associated with occlusion and/or noise.

This setting allows us to recast the matching problem as the optimization of a path's cost over a graph whose nodes correspond to pairs of left and right image features; and arcs represent matches between left and right intensity profile intervals bounded by the features of the corresponding nodes (Figure 7.14, right). The cost of an arc measures the discrepancy between the corresponding intervals (e.g., the squared difference of the mean intensity values). This optimization problem can be solved, exactly and efficiently, using dynamic programming (Algorithm 7.2). As given, this algorithm has a computational complexity of $O(mn)$, where m and n denote the number of edge points on the matched left and right scanlines, respectively.[2] Variants of this approach have been implemented by Baker and Binford (1981), who combine a coarse-to-fine intra-scanline search procedure with a cooperative process for enforcing inter-scanline consistency, and Ohta and Kanade (1985), who use dynamic programming for both intra- and inter-scanline optimization, the latter procedure being conducted in a three-dimensional search space. Figure 7.15 shows a sample result taken from Ohta and Kanade (1985).

7.5.2 Smoothness Constraints and Combinatorial Optimization over Graphs

Dynamic programming is a *combinatorial optimization* algorithm aimed at minimizing an error function (a path cost) over some discrete variables (correspondences between pairs of features). It was used in the previous section to incorporate ordering constraints in the matching process. We now present a different approach to stereo fusion that relies instead on smoothness constraints, and a different combinatorial optimization technique aimed at minimizing certain energy functions defined over graphs.

[2] Our version of the algorithm assumes that all edges are matched. To account for noise and edge-detection errors, it is reasonable to allow the matching algorithm to skip a bounded number of edges, but this does not change its asymptotic complexity (Ohta and Kanade 1985).

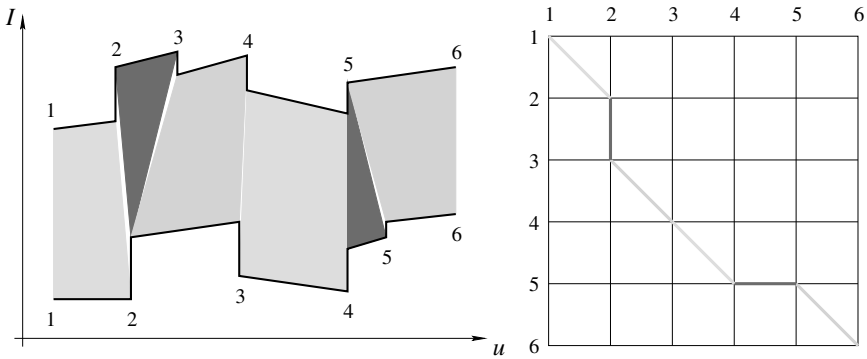

FIGURE 7.14: Dynamic programming and stereopsis: The **left** part of the figure shows two intensity profiles along matching epipolar lines. The polygons joining the two profiles indicate matches between successive intervals (some of the matched intervals may have zero length). The **right** part of the diagram represents the same information in graphical form: an arc (thick line segment) joins two nodes (i, i') and (j, j') when the intervals (i, j) and (i', j') of the intensity profiles match each other.

We assume the scanlines have m and n edge points, respectively (the endpoints of the scanlines are included for convenience). Two auxiliary functions are used: Inferior-Neighbors(k, l) returns the list of neighbors (i, j) of the node (k, l) such that $i \leq k$ and $j \leq l$, and Arc-Cost(i, j, k, l) evaluates and returns the cost of matching the intervals (i, k) and (j, l). For correctness, $C(1, 1)$ should be initialized with a value of zero.

```
% Loop over all nodes (k,l) in ascending order.
for k = 1 to m do
  for l = 1 to n do
    % Initialize optimal cost C(k,l) and backward pointer B(k,l).
    C(k,l) ← +∞; B(k,l) ← nil;
    % Loop over all inferior neighbors (i,j) of (k,l).
    for (i,j) ∈ Inferior-Neighbors(k,l) do
      % Compute new path cost and update backward pointer if necessary.
      d ← C(i,j) + Arc-Cost(i,j,k,l);
      if d < C(k,l) then C(k,l) ← d; B(k,l) ← (i,j) endif;
      endfor;
    endfor;
  endfor;
% Construct optimal path by following backward pointers from (m,n).
P ← {(m,n)}; (i,j) ← (m,n);
while B(i,j) ≠ nil do (i,j) ← B(i,j); P ← {(i,j)} ∪ P endwhile.
```

Algorithm 7.2: A Dynamic-Programming Algorithm for Establishing Stereo Correspondences Between Two Corresponding Scanlines.