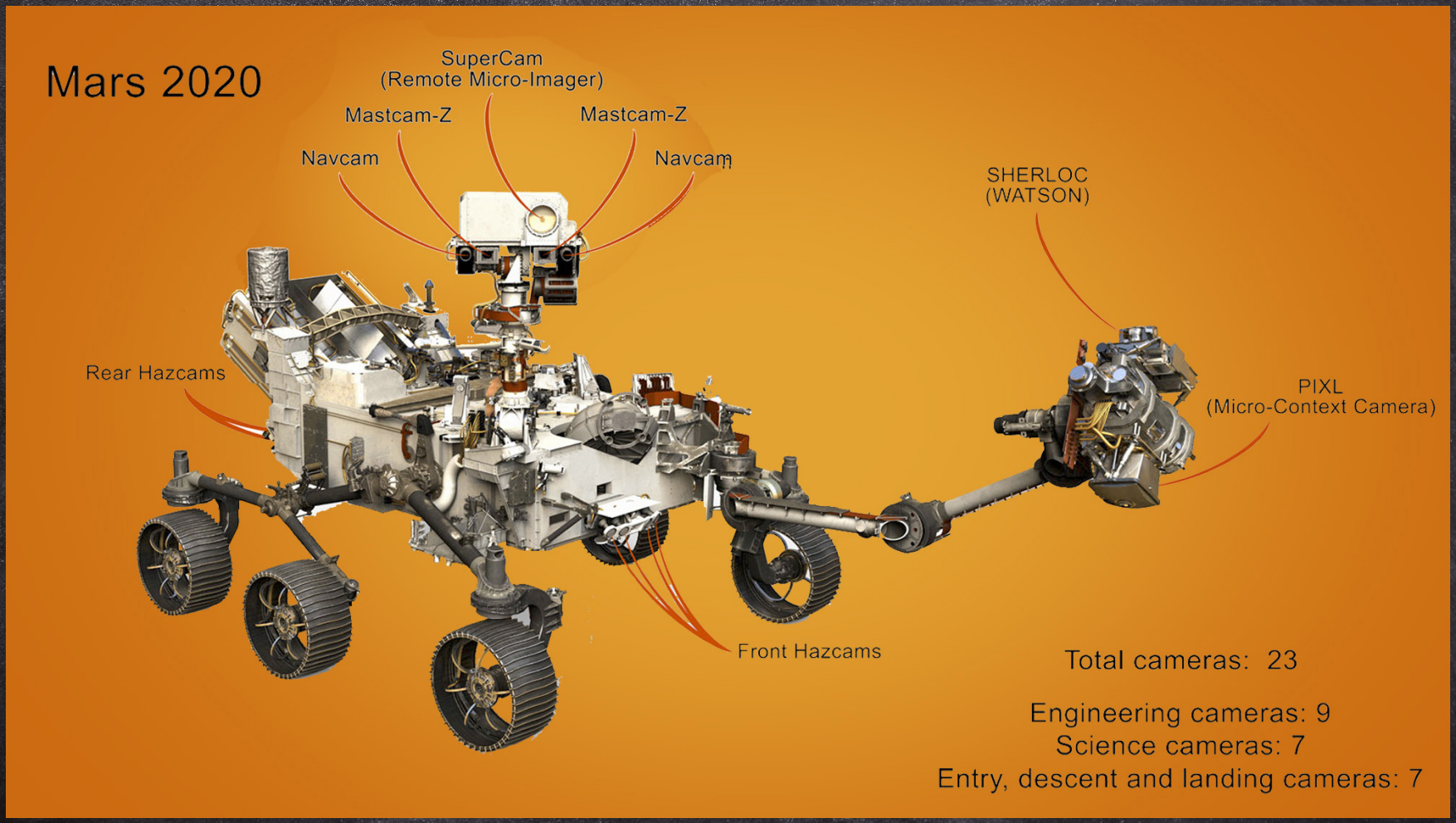
# 计算视觉与模式识别

# 立体视觉




Mars 2020
SuperCam
(Remote Micro-Imager)
Mastcam-Z
Mastcam-Z
Navcam
Navcam
SHERLOC
(WATSON)
Rear Hazcams
PIXL
(Micro-Context Camera)
Front Hazcams
Total cameras: 23
Engineering cameras: 9
Science cameras: 7
Entry, descent and landing cameras: 7

# 双目融合



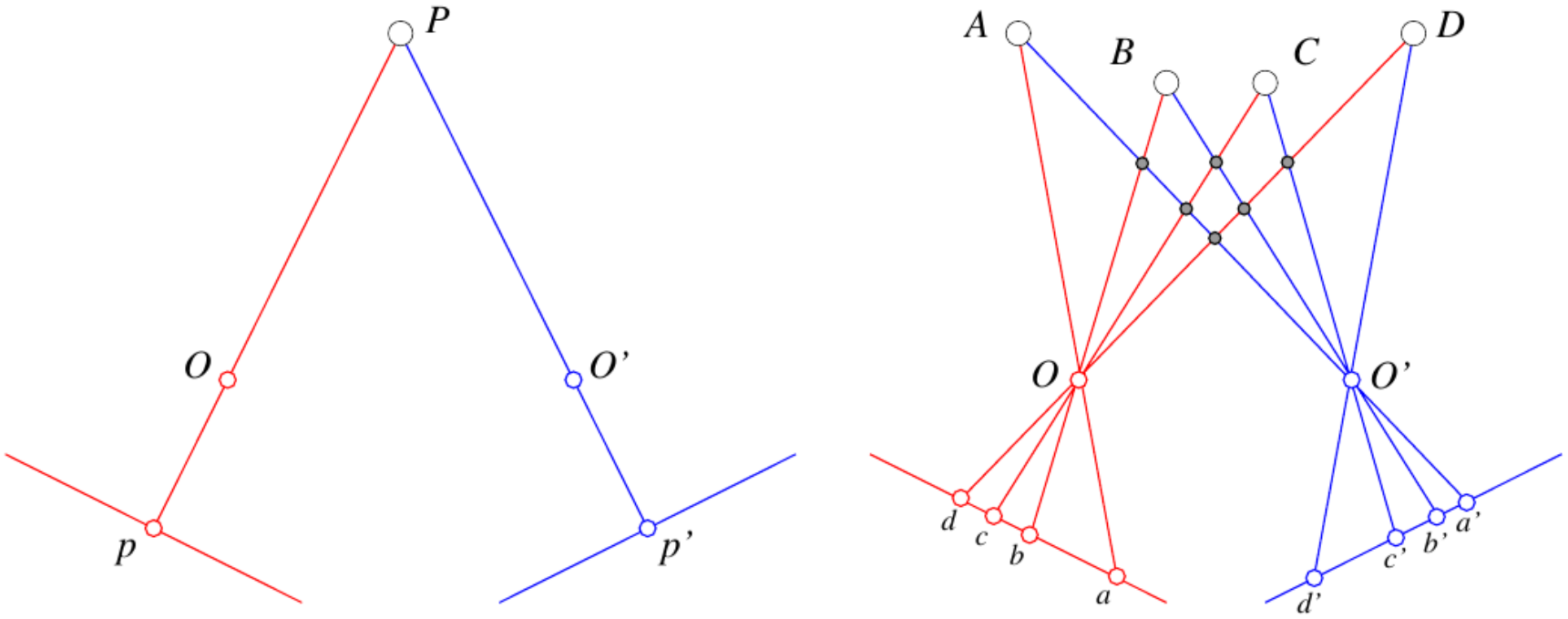



**Figure 13.3.** The binocular fusion problem: in the simple case of the diagram shown on the left, there is no ambiguity and stereo reconstruction is a simple matter. In the more usual case shown on the right, any of the four points in the left picture may, a priori, match any of the four points in the right one. Only four of these correspondences are correct, the other ones yielding the incorrect reconstructions shown as small grey discs.

$$z\boldsymbol{p} = \mathcal{M}\boldsymbol{P}$$
$$z'\boldsymbol{p}' = \mathcal{M}'\boldsymbol{P}$$

得

$$\boldsymbol{p} \times \mathcal{M}\boldsymbol{P} = \mathbf{0}$$
$$\boldsymbol{p}' \times \mathcal{M}'\boldsymbol{P} = \mathbf{0}$$

即

$$([\boldsymbol{p}]_\times \mathcal{M})\boldsymbol{P} = 0$$
$$([\boldsymbol{p}']_\times \mathcal{M}')\boldsymbol{P} = 0$$

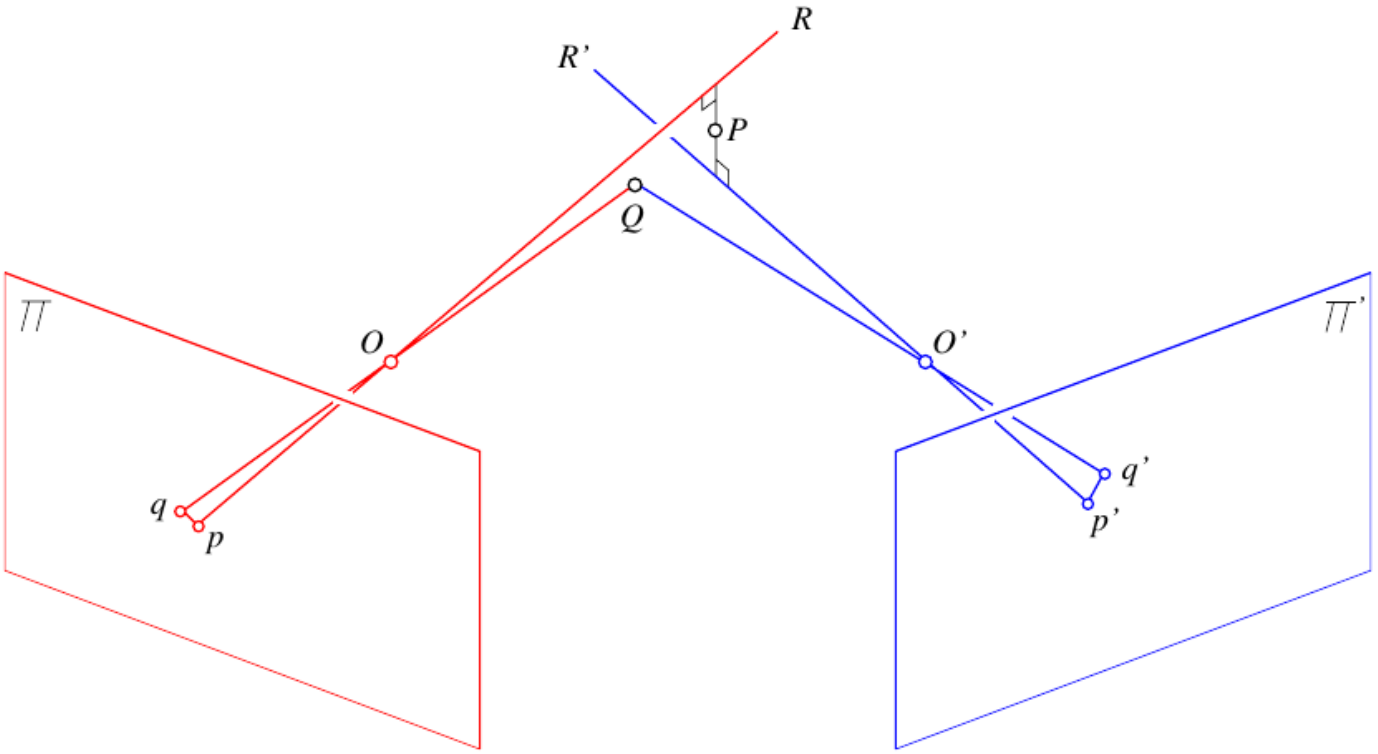



**Figure 13.4.** Triangulation in the presence of measurement errors. See text for details.

# 图像校正



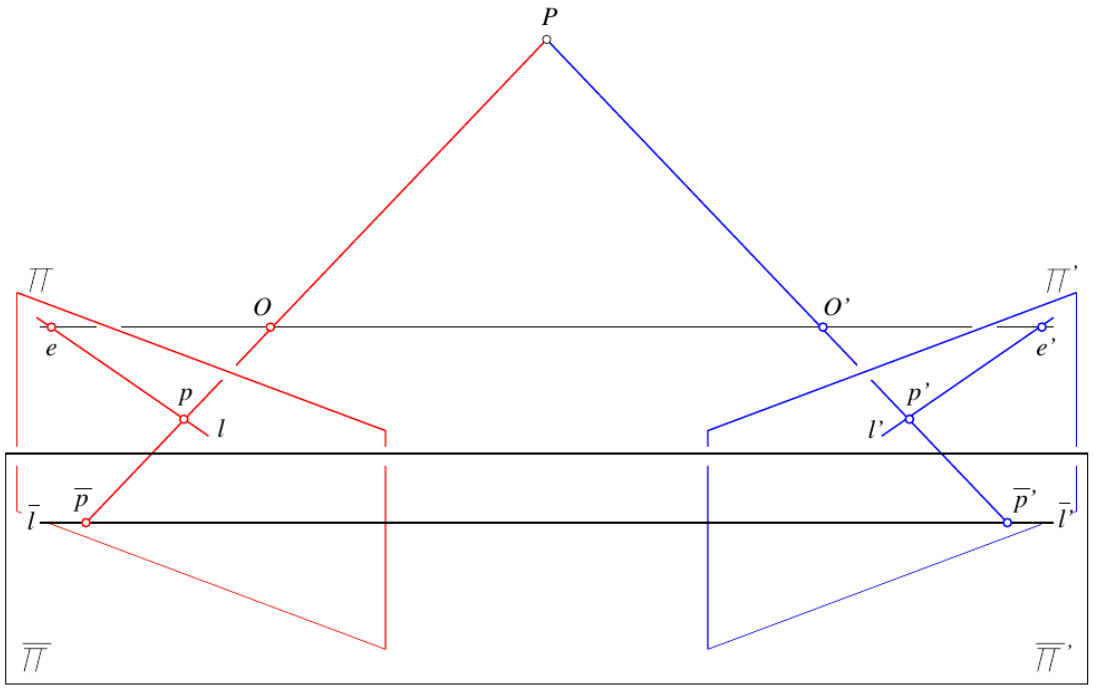



**Figure 13.5.** A rectified stereo pair: the two image planes $\Pi$ and $\Pi'$ are reprojected onto a common plane $\bar{\Pi} = \bar{\Pi}'$ parallel to the baseline. The epipolar lines $l$ and $l'$ associated with the points $p$ and $p'$ in the two pictures map onto a common scanline $\bar{l} = \bar{l}'$ also parallel to the baseline and passing through the reprojected points $\bar{p}$ and $\bar{p}'$. The rectified images are easily constructed by considering each input image as a polyhedral mesh and using texture mapping to render the projection of this mesh into the plane $\bar{\Pi} = \bar{\Pi}'$.

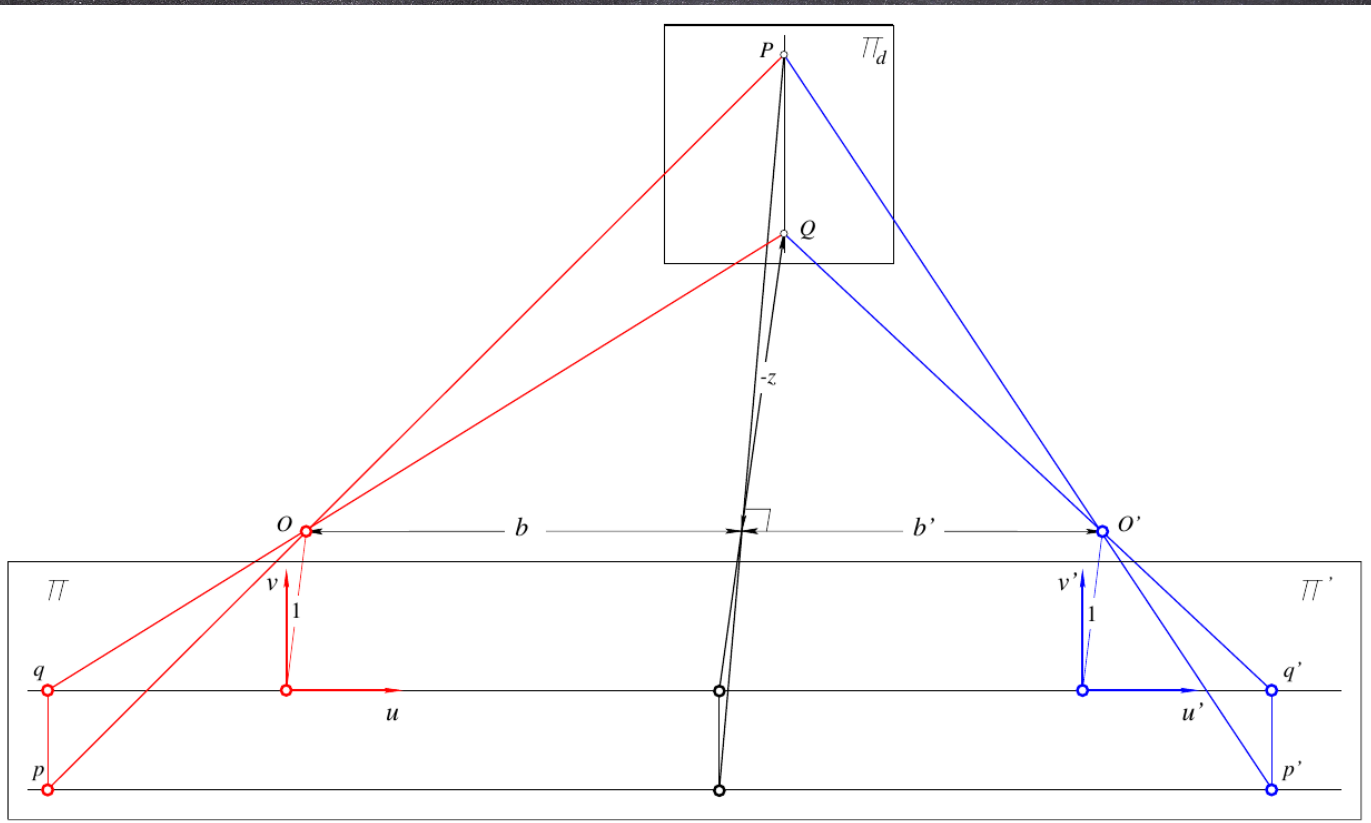



**Figure 13.6.** Triangulation for rectified images: the rays associated with two points $p$ and $p'$ on the same scanline are by construction guaranteed to intersect in some point $P$. As shown in the text, the depth of $P$ relative to the coordinate system attached to the left camera is inversely proportional to the disparity $d = u' - u$. In particular, the preimage of all pairs of image points with constant disparity $d$ is a *frontoparallel* plane $\Pi_d$ (i.e., a plane parallel to the camera retinas).

# 人眼立体视觉



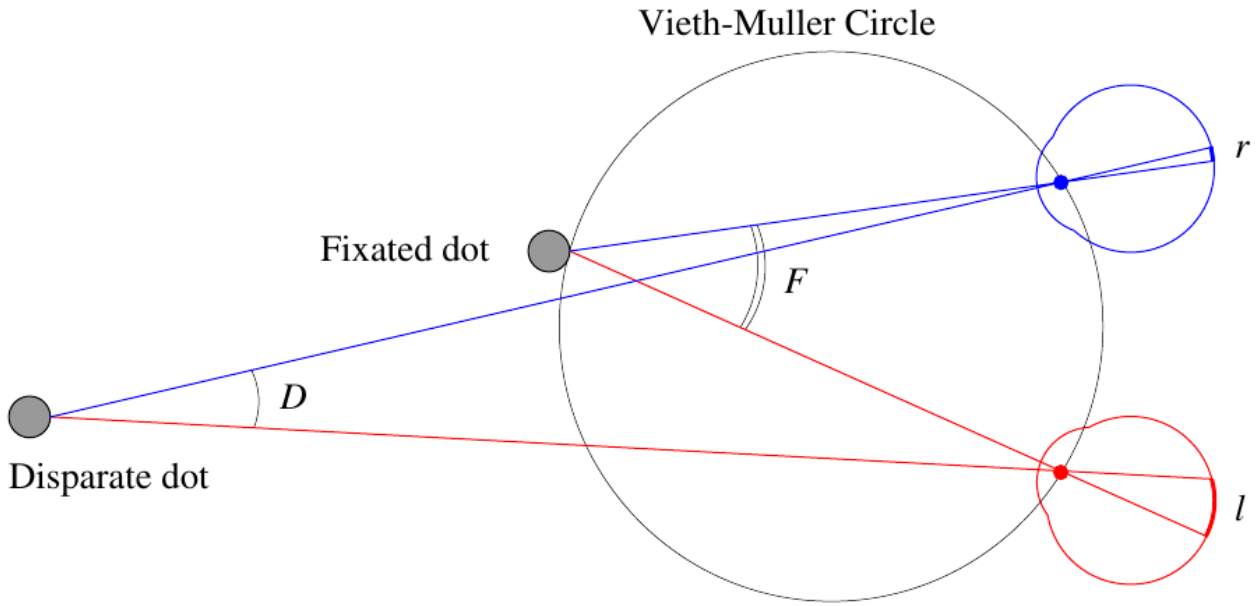



**Figure 13.7.** This diagram depicts a situation similar to that of the sailor in Figure 13.1. The close-by dot is fixated by the eyes, and it projects onto the center of their foveas, with no disparity. The two images of the far dot deviate from this central position by different amounts, indicating a different depth.

# 随机点体视图



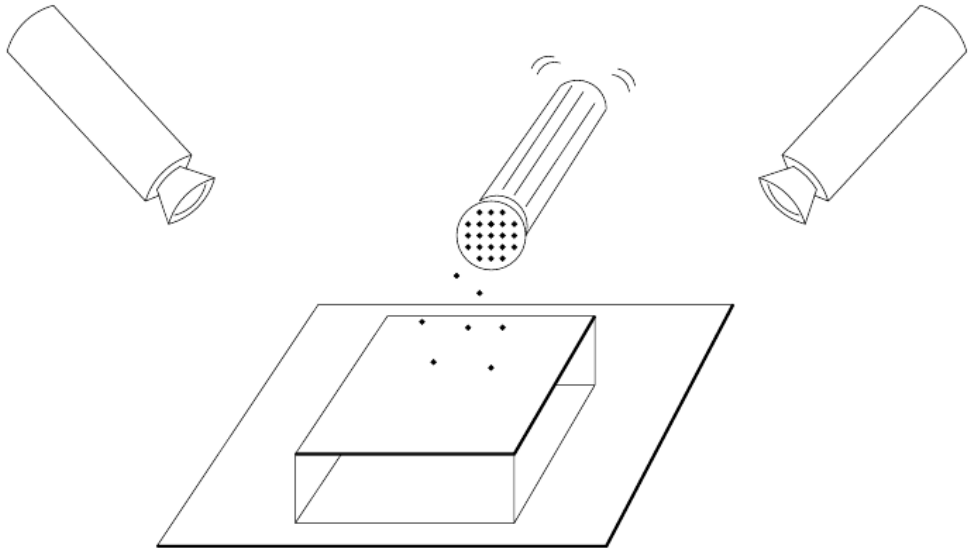


**Figure 13.8.** Creating random dot stereograms by shaking pepper over a pair of plates observed by two cameras. In the experiments presented in [Julesz, 1960], the two images are of course synthesized by a computer using a random-number generator to decide the dot locations and pixel intensities, that can either be binary values as in the situation described in the text, or more generally random values in the 0..15 range. The two pictures have the same random background and differ in a central region by a constant horizontal offset.

# Marr-Poggio algorithm



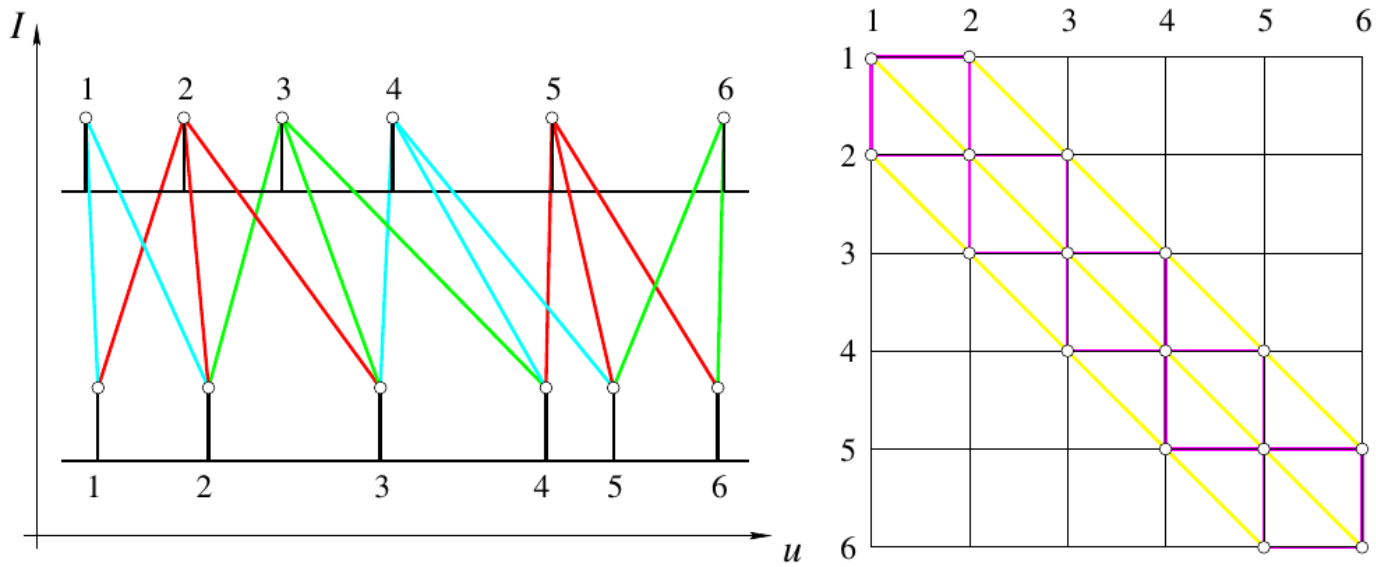



**Figure 13.9.** A cooperative approach to stereopsis: the Marr-Poggio algorithm [1976]. The left part of the figure shows two intensity profiles along the same scanline of two images. The spikes correspond to black dots. The line segments joining the two profiles indicate possible matches between dots given some maximum disparity range. These matches are also shown in the right part of the figure, where they form the nodes of a graph. The vertical and horizontal arcs of this graph join nodes associated with the same dot in the left or right image. The diagonal arcs join nodes with similar disparities.

# 随机点体视图融合



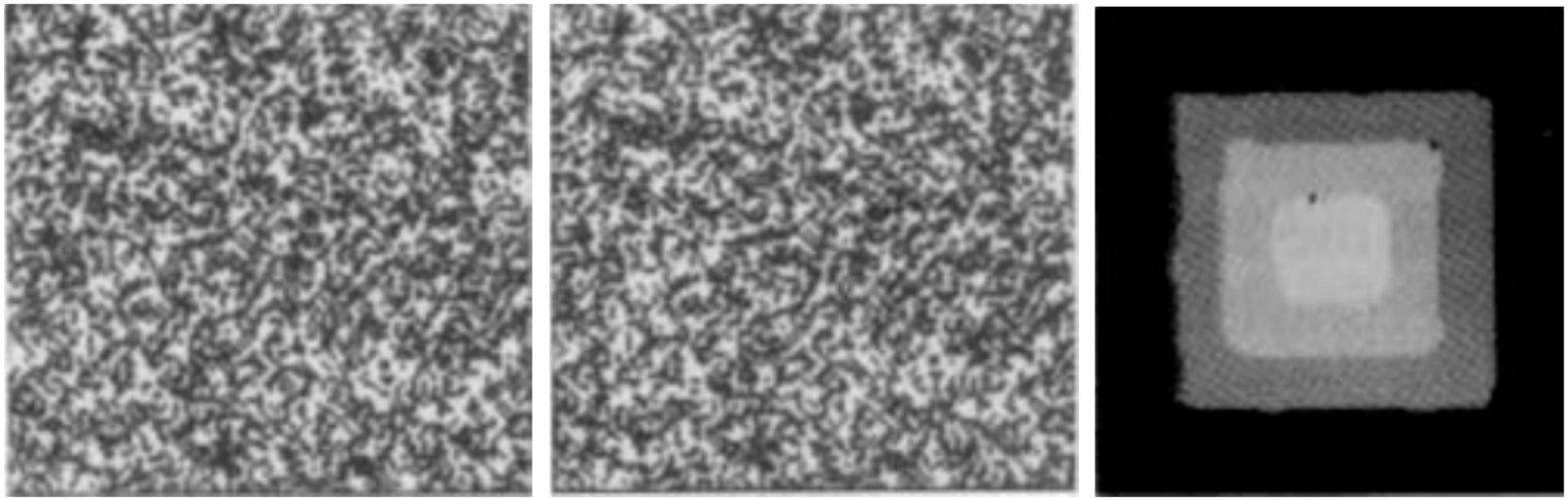


**Figure 13.10.** From left to right: a random dot stereogram depicting four planes at varying depth (a "wedding cake") and the disparity map obtained after 14 iterations of the Marr-Poggio cooperative algorithm. Reprinted from [Marr, 1982], Figure 3-7.

# 相关法找对应点



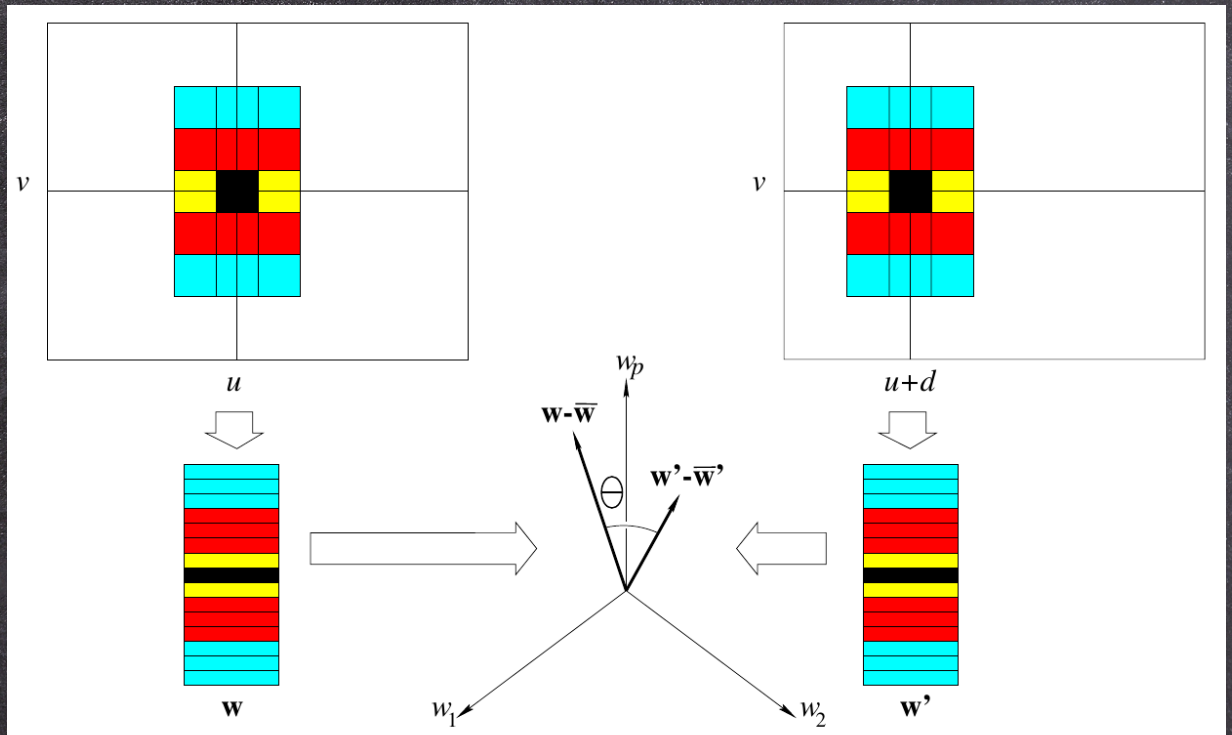



**Figure 13.11.** Correlation of two $3 \times 5$ windows along corresponding epipolar lines. The second window position is separated from the first one by an offset $d$. The two windows are encoded by vectors $\boldsymbol{w}$ and $\boldsymbol{w}'$ in $\mathbb{R}^{15}$, and the correlation function measures the cosine of the angle $\theta$ between the vectors $\boldsymbol{w} - \bar{\boldsymbol{w}}$ and $\boldsymbol{w}' - \bar{\boldsymbol{w}}'$ obtained by substracting from the components of $\boldsymbol{w}$ and $\boldsymbol{w}'$ the average intensity in the corresponding windows.

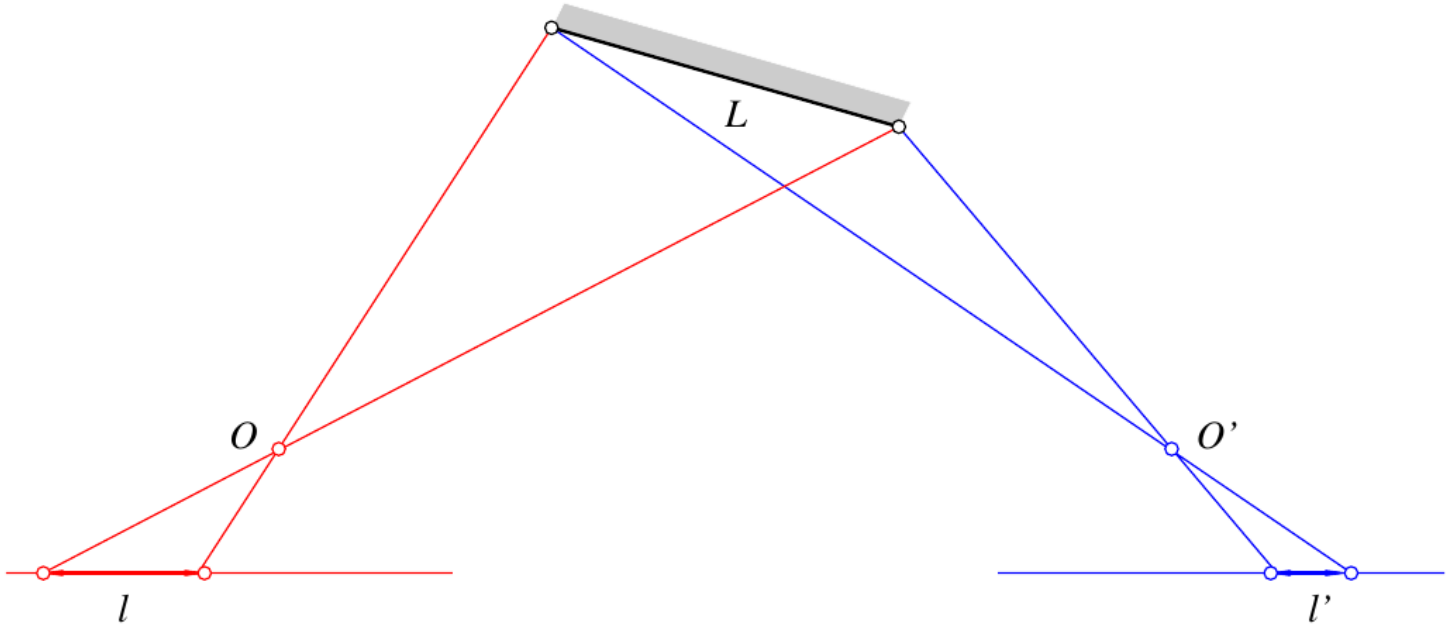



**Figure 13.12.** The foreshortening of non-frontoparallel surfaces is different for the two cameras: a surface segment with length $L$ projects onto two image segments of different lengths $l$ and $l'$.

# 边缘匹配



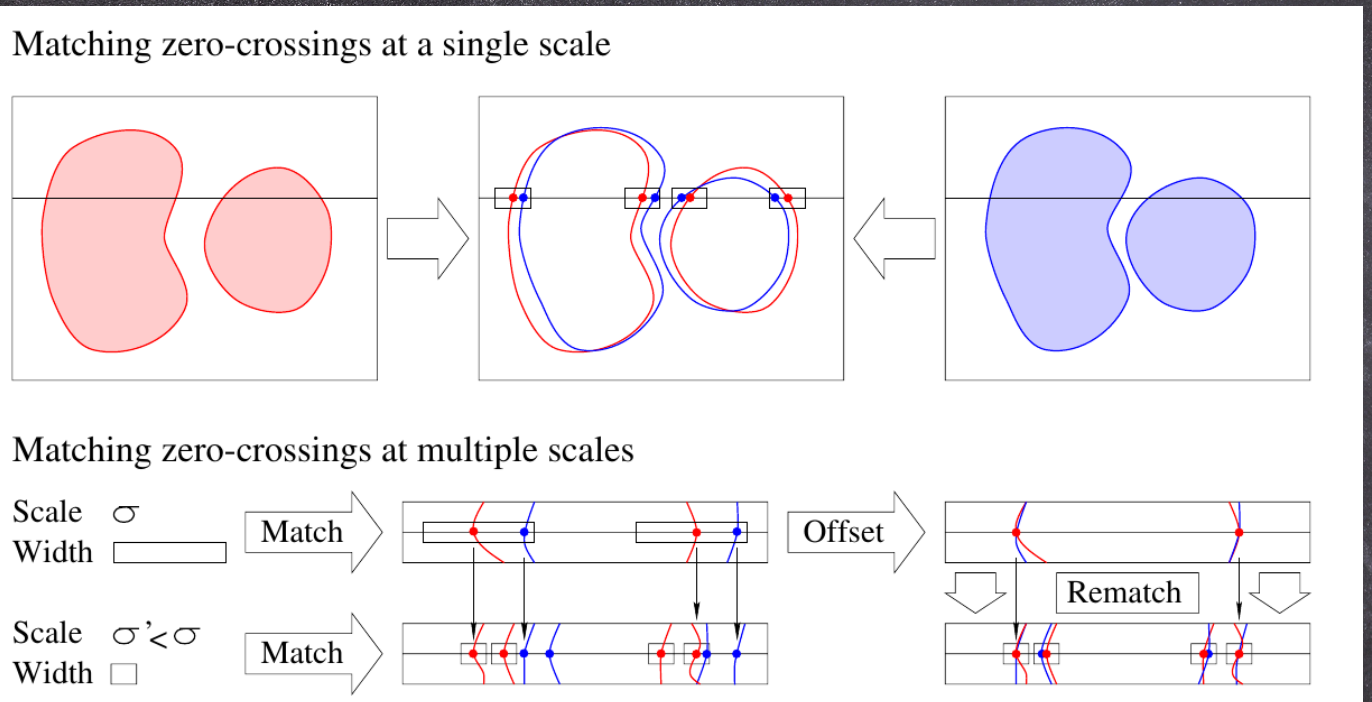



**Figure 13.14.** Multi-scale matching of zero crossings: the eye movements (or equivalently the image offsets used in matching) are controlled by seeking image regions that have been assigned a disparity value at a scale $\sigma'$ but not at a scale $\sigma < \sigma'$. These values are used to refine the eye positions and bring the corresponding regions within matchable range. The disparity value associated with a region can be found by various methods, for example by averaging the disparity values found at each matched zero crossing within it.

# 次序约束



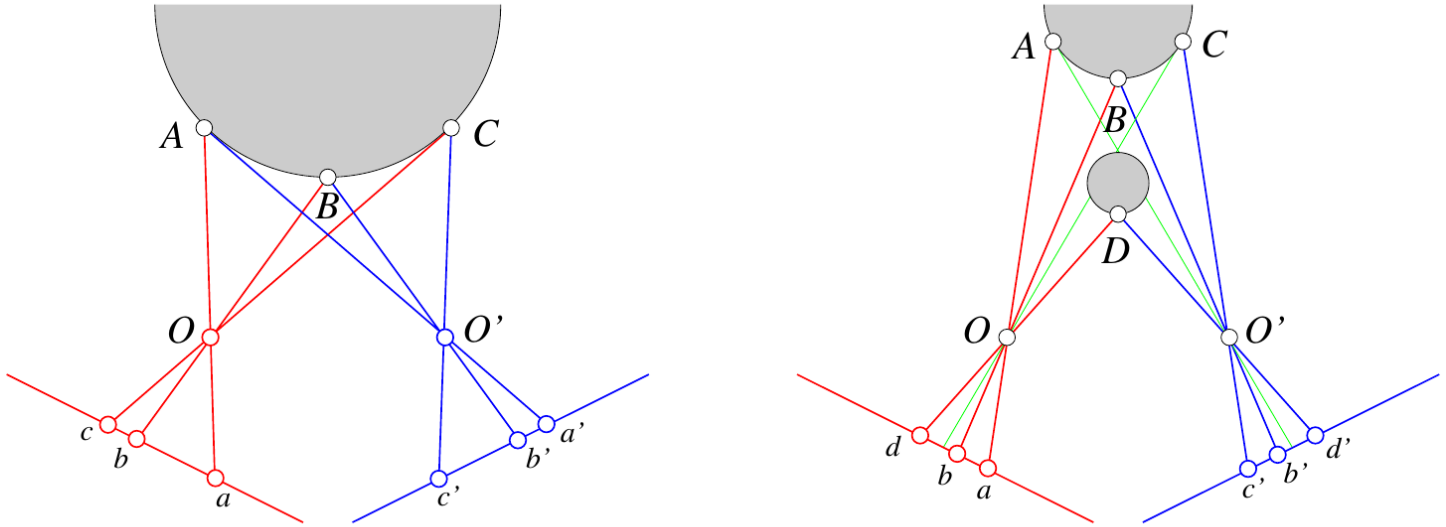



**Figure 13.16.** Ordering constraints. In the (usual) case shown in the left part of the diagram, the order of feature points along the two (oriented) epipolar lines is the same, and it is the inverse of the order of the scene points along the curve where the observed surface intersects the epipolar plane. 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.

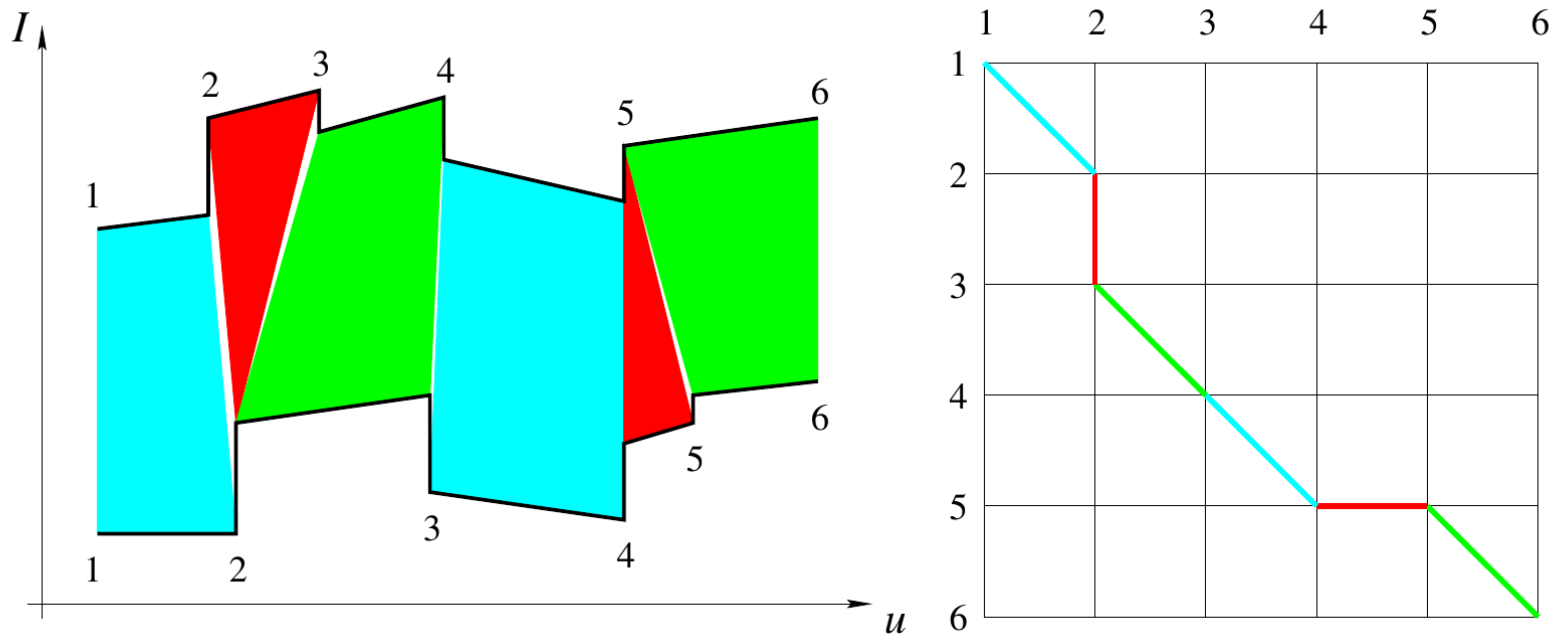



**Figure 13.17.** Dynamic programming and stereopsis: the left part of the figure shows two intensity profiles along matching epipolar lines. The polygons joining the two profiles indicate matches between successive intervals (some of the matched intervals may have zero length). The right part of the 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.

# Dynamic Programming

- m, n : edge points, (the endpoints of the scanlines are included.)
- Inferior-Neighbors(k, l): neighbors (i, j) of t (k, l) $i \leq k$, $j \leq l$,
- Arc-Cost(i, j, k, l): cost of matching the intervals (i, k) and (j, l).
- C(1, 1): should be initialized with a value of zero.
  - % Loop over all nodes (k, l) in ascending order.
  - % Construct optimal path by following backward pointers from (m, n).

# Dynamic Programming

- m, n : edge points, (the endpoints of the scanlines are included.)
- Inferior-Neighbors(k, l): neighbors (i, j) of t (k, l) $i \leq k$, $j \leq l$,
- Arc-Cost(i, j, k, l): cost of matching the intervals (i, k) and (j, l).
- 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 cost C(k, l) and backward pointer B(k, l).

  $C(k, l) \leftarrow +\infty$; B(k, l) $\leftarrow$ nil;

  - % Loop over all inferior neighbors (i, j) of (k, l).

endfor;

endfor;

- % Construct optimal path by following backward pointers from (m, n).

# Dynamic Programming

- m, n : edge points, (the endpoints of the scanlines are included.)
- Inferior-Neighbors(k, l): neighbors (i, j) of t (k, l) $i \leq k$, $j \leq l$,
- Arc-Cost(i, j, k, l): cost of matching the intervals (i, k) and (j, l).
- 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 cost C(k, l) and backward pointer B(k, l).

  $C(k, l) \leftarrow +\infty$; B(k, l) $\leftarrow$ nil;

  % Loop over all inferior neighbors (i, j) of (k, l).

  for (i, j) $\in$ Inferior-Neighbors(k, l) do

% Compute cost, update backward pointer

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).

# Dynamic Programming

- m, n : edge points, (the endpoints of the scanlines are included.)
- Inferior-Neighbors(k, l): neighbors (i, j) of t (k, l) $i \leq k$, $j \leq l$,
- Arc-Cost(i, j, k, l): cost of matching the intervals (i, k) and (j, l).
- 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 cost C(k, l) and backward pointer B(k, l).

  $C(k, l) \leftarrow +\infty$; B(k, l) $\leftarrow$ nil;

  % Loop over all inferior neighbors (i, j) of (k, l).

  for (i, j) $\in$ Inferior-Neighbors(k, l) do

% Compute cost, update backward pointer

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

$(i, j) \leftarrow B(i, j)$;

$P \leftarrow \{(i, j)\} \cup P$

endwhile.

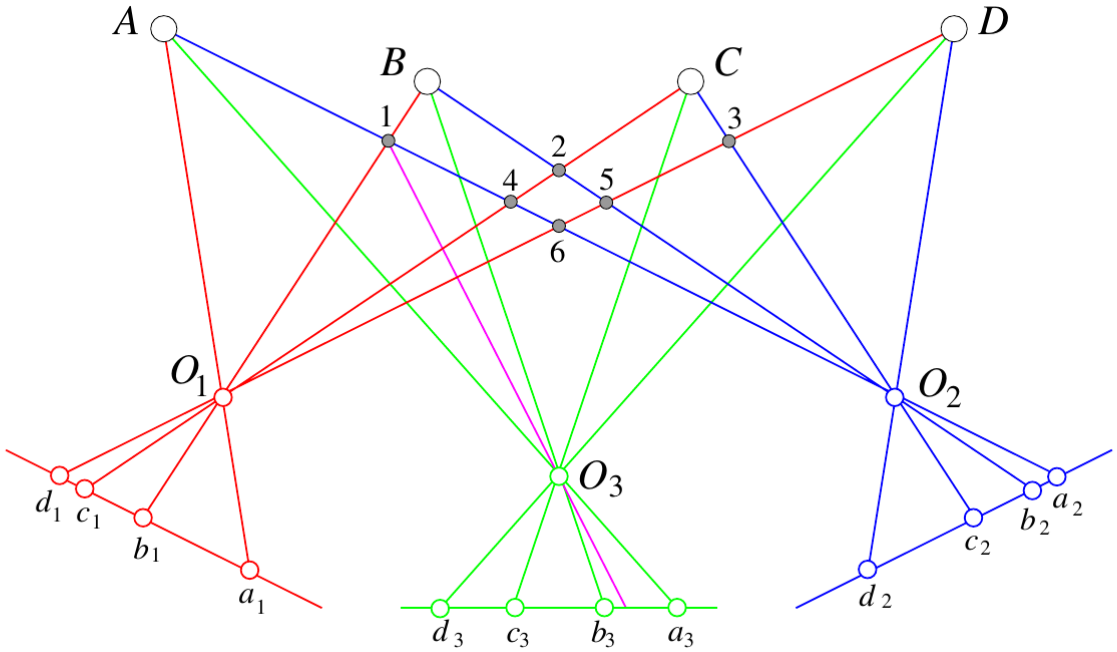



**Figure 13.19.** The small grey discs indicate the incorrect reconstructions associated with the left and right images of four points. The addition of a central camera removes the matching ambiguity: none of the corresponding rays intersects any of the six discs. Alternatively, matches between points in the first two images can be checked by reprojecting the corresponding three-dimensional point in the third image. For example, the match between $b_1$ and $a_2$ is obviously wrong since there is no feature point in the third image near the reprojection of the hypothetical reconstruction numbered 1 in the diagram.

# 曲线切线预测



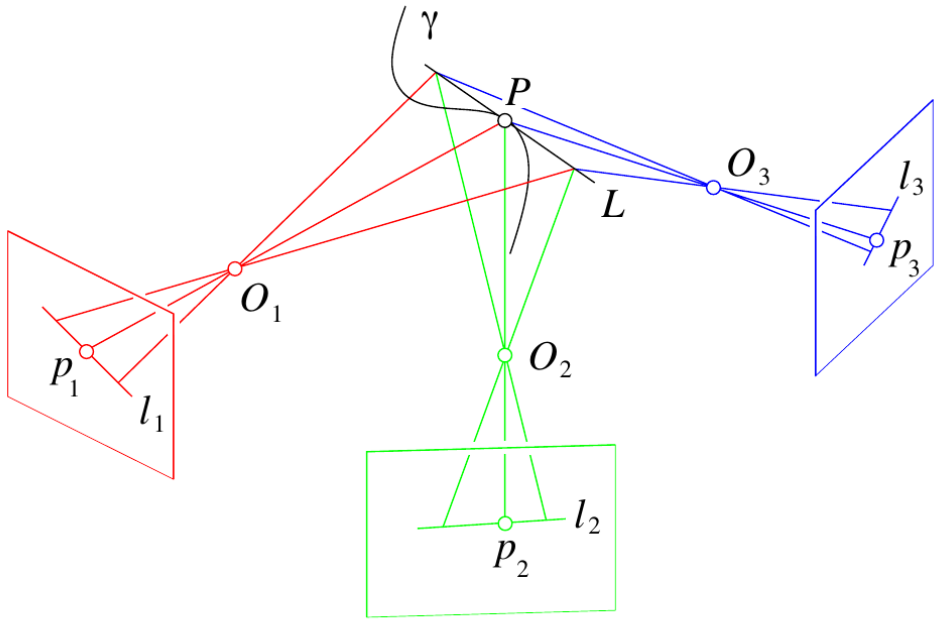



**Figure 13.20.** Given matches between the points $p_1$ and $p_2$ and their tangents $l_1$ and $l_2$ in two images, it is possible to predict both the position of the corresponding point $p_3$ and tangent $l_3$ in a third image.

- Note

# 曲线切线预测



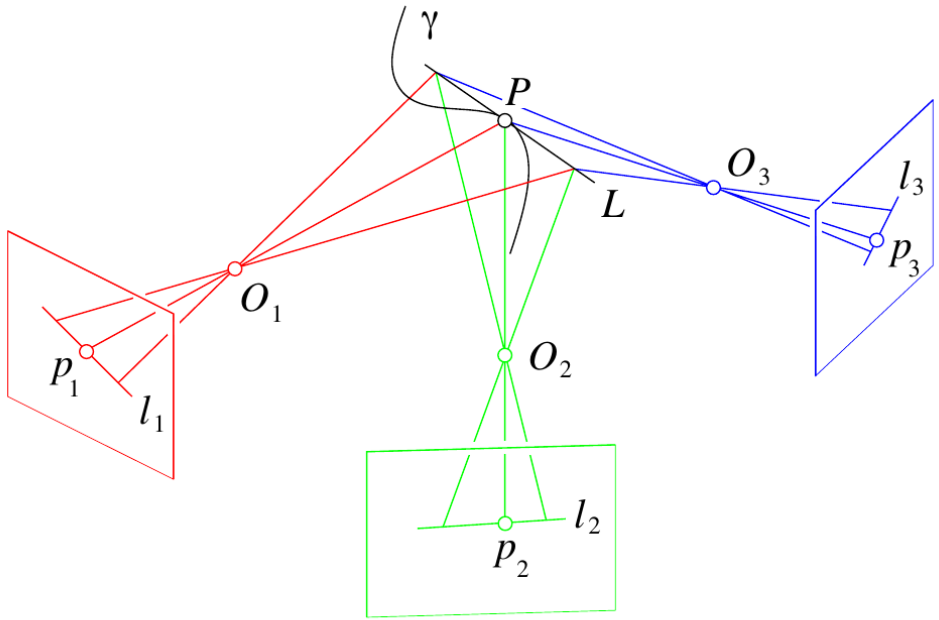



**Figure 13.20.** Given matches between the points $p_1$ and $p_2$ and their tangents $l_1$ and $l_2$ in two images, it is possible to predict both the position of the corresponding point $p_3$ and tangent $l_3$ in a third image.

- Note

$$\boldsymbol{l}_1 \;\propto\; \begin{pmatrix} \boldsymbol{l}_2^T \mathcal{G}_1^1 \boldsymbol{l}_3 \\ \boldsymbol{l}_2^T \mathcal{G}_1^2 \boldsymbol{l}_3 \\ \boldsymbol{l}_2^T \mathcal{G}_1^3 \boldsymbol{l}_3 \end{pmatrix}$$

$$\mathcal{G}_1^i \;=\; \boldsymbol{t}_2 \boldsymbol{R}_3^{iT} - \boldsymbol{R}_2^i \boldsymbol{t}_3^T \qquad i = 1, 2, 3$$