Stereo Imaging

Depth calculation of image pairs

JIN LINHAO

In practice, stereo imaging involves four steps when using two cameras.

1. Mathematically remove radial and tangential lens distortion; this is called undistortion. The outputs of this step are undistorted images.
2. Adjust for the angles and distances between cameras, a process called rectification. The outputs of this step are images that are row-aligned and rectified.
3. Find the same features in the left and right camera views, a process known as correspondence. The output of this step is a disparity map, where the disparities are the differences in x-coordinates on the image planes of the same feature viewed in the left and right cameras: $x_l - x_r$.
4. If we know the geometric arrangement of the cameras, then we can turn the disparity map into distances by triangulation. This step is called reprojection, and the output is a depth map.

Camera Model

The relation that maps the points Qi in the physical world with coordinates (X_i, Y_i, Z_i) to the points on the projection screen with coordinates (x_i, y_i) is called a projective transform. When working with such transforms, it is convenient to use what are known as homogeneous coordinates. The homogeneous coordinates associated with a point in a projective space of dimension n are typically expressed as an (n + 1)-dimensional vector (e.g., x, y, z becomes x, y, z, w), with the additional restriction that any two points whose values are proportional are equivalent. In our case, the image plane is the projective space and it has two dimensions, so we will represent points on that plane as three dimensional vectors q = (q_1, q_2, q_3). Recalling that all points having proportional values in the projective space are equivalent, we can recover the actual pixel coordinates by dividing through by q_3. This allows us to arrange the parameters that define our camera (i.e., f_x, f_y, c_x, and c_y) into a single 3-by-3 matrix, which we will call the camera intrinsics matrix.
The projection of the points in the physical world into the camera is now summarized by the following simple form:

$$q = MQ, \quad \text{where} \quad q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Multiplying this out, you will find that w = Z and so, since the point q is in homogeneous coordinates, we should divide through by w (or Z) in order to recover our earlier definitions.

Epipolar Geometry

The basic geometry of a stereo imaging system is referred to as epipolar geometry as shown in Figure 12-8.

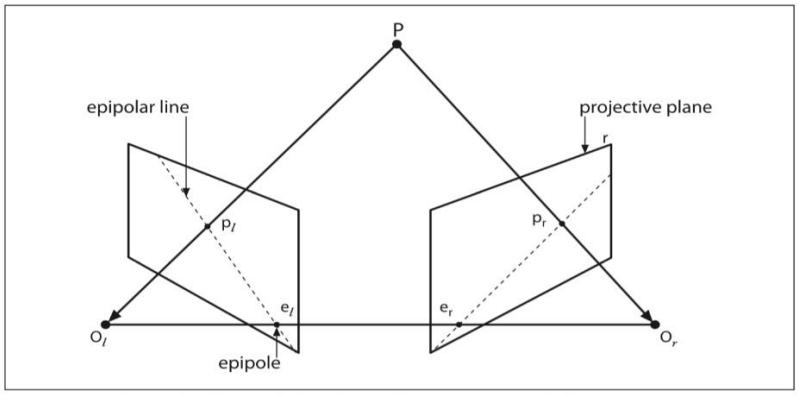

Figure 12-8. The epipolar plane is defined by the observed point P and the two centers of projection, O_l and O_r; the epipoles are located at the point of intersection of the line joining the centers of projection and the two projective planes

Here are some facts about epipolar geometry:

- Every 3D point in view of the cameras is contained in an epipolar plane that inter- sects each image in an epipolar line.
- Given a feature in one image, its matching view in the other image must lie along the corresponding epipolar line. This is known as the epipolar constraint.
- The epipolar constraint means that the possible two-dimensional search for match- ing features across two imagers becomes a one-dimensional search along the epi- polar lines once we know the epipolar geometry of the stereo rig. This is not only a vast computational savings, it also allows us to reject a lot of points that could otherwise lead to spurious correspondences.
- Order is preserved. If points A and B are visible in both images and occur horizon- tally in that order in one imager, then they occur horizontally in that order in the other imager.

The Essential and Fundamental Matrices

The essential matrix E contains information about the translation and rotation that relate the two cameras in physical space (see Figure 12-9), and fundamental matrix F contains the same information as E in addition to information about the intrinsics of both cameras. Because F embeds information about the intrinsic parameters, it relates the two cameras in pixel coordinates.

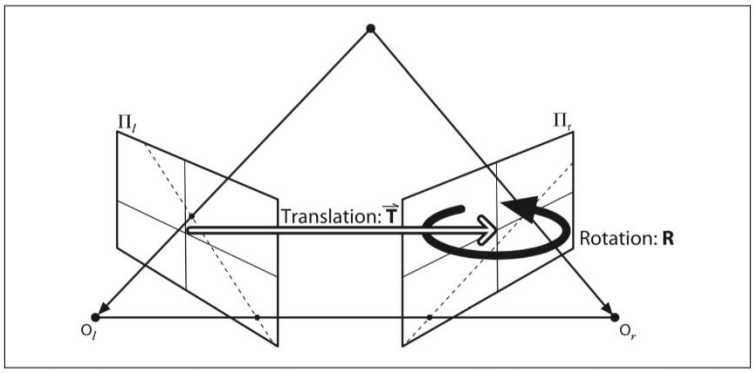

Figure 12-9. The essential geometry of stereo imaging is captured by the essential matrix E, which contains all of the information about the translation T and the rotation R, which describe the location of the second camera relative to the first in global coordinates

Essential Matrix

For calculation of Essential Matrix, choose the coordinates centered on O_l of the left camera. the location of the observed point is P_l and the origin of the other camera is located at T. The point P as seen by the right camera is P_r in that camera's coordinates, where $P_r = R(P_l - T)$. recall that the equation for all points x on a plane with normal vector n and passing through point a obey the following constraint:

$$(\mathbf{x}-\mathbf{a})\cdot\mathbf{n}=0$$

Recall that the epipolar plane contains the vectors P_l and T; thus, if we had a vector (e.g., $P_l \times T$) perpendicular to both, then we could use that for n in our plane equation. Thus an equation for all possible points Pl through the point T and containing both vectors would be:

$$(P_l - T)^{\mathrm{T}}(T \times P_l)=0$$

Substituting the transition between P_l and P_r yields:

$$P_r = R(P_l - T)$$

$$(R^{\mathrm{T}}P_r)^{\mathrm{T}}(T \times P_l)=0$$

Rewrite the cross product,

$$T \times P_l = SP_l \quad \Rightarrow \quad S=\begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

This leads to our first result. Making this substitution for the cross product gives:

$$(P_r)^{\mathrm{T}} RSP_l = 0$$

This product RS is what we define to be the essential matrix E, which leads to the compact equation:

$$(P_r)^T E P_l = 0$$

Simply substitute using the projection equations $p_l = f_l P_l / Z_l$ and $p_r = f_r P_r / Z_r$ and then divide the whole thing by $Z_l Z_r / f_l f_r$ to obtain our final result of relation between the points on the respective imager:

$$p_r^T E p_l = 0$$

Essential matrix E turn out to be a 3-by-3 matrix with rank two, so the equation above ends up being an equation for a line. There are five parameters in the essential matrix—three for rotation and two for the direction of translation (scale is not set)—along with two other constraints. The two additional constraints are:

1. The determinant is zero since it is rank-deficient
2. Its two nonzero singular values are equal because the matrix S is skew-symmetric and R is a rotational matrix

Fundamental matrix

To introduce intrinsic information about the two cameras, we substitute q and the camera intrinsics matrix that relates them for p, the pixel coordinates. Recall that q=Mp (where M is the camera intrinsics matrix), the equation for E becomes:

$$q_r^T (M_r^{-1})^T E M_l^{-1} q_l = 0$$

Define fundamental matrix F as:

$$F = (M_r^{-1})^T E M_l^{-1}$$

So that

$$q_r^T F q_l = 0$$

The fundamental matrix F is just like the essential matrix E, except that F operates in image pixel coordinates whereas E operates in physical coordinates.* Just like E, the fundamental matrix F is of rank 2. The fundamental matrix has seven parameters, two for each epipole and three for the homography that relates the two image planes (the scale aspect is missing from the usual four parameters).

Undistortion

Radial distortions arise as a result of the shape of lens, whereas tangential distortions arise from the assembly process of the camera as a whole. For radial distortions, the distortion is 0 at the (optical) center of the imager and increases as we move toward the periphery. In practice, this distortion is small and can be characterized by the first few terms of a Taylor series expansion around r = 0. For cheap web cameras, we generally use the first two such terms; the first of which is conventionally called k_1 and the second k_2. For highly distorted cameras such as fish-

eye lenses we can use a third radial distortion term k_3. In general, the radial location of a point on the imager will be rescaled according to the following equations:

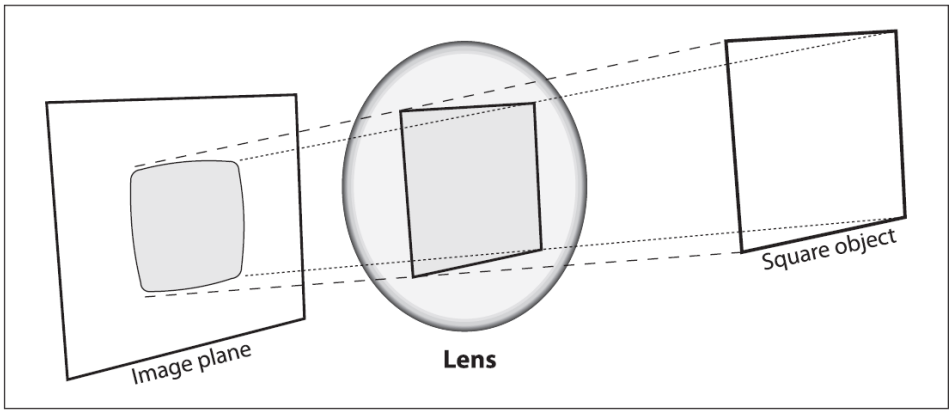

Figure 11-3. Radial distortion: rays farther from the center of a simple lens are bent too much compared to rays that pass closer to the center; thus, the sides of a square appear to bow out on the image plane (this is also known as barrel distortion)

$$x_{\text{corrected}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Here, (x, y) is the original location (on the imager) of the distorted point and ($x_{\text{corrected}}$, $y_{\text{corrected}}$) is the new location as a result of the correction.
The second-largest common distortion is tangential distortion. This distortion is due to manufacturing defects resulting from the lens not being exactly parallel to the imaging plane. Tangential distortion is minimally characterized by two additional parameters, p_1 and p_2, such that:

$$x_{\text{corrected}} = x + [2p_1 y + p_2(r^2 + 2x^2)]$$
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 x]$$

Thus in total there are five distortion coefficients that we require. Because all five are necessary in most of the OpenCV routines that use them, they are typically bundled into one distortion vector; this is just a 5-by-1 matrix containing k_1, k_2, p_1, p_2, and k_3.

There are two things that one often wants to do with a calibrated camera. The first is to correct for distortion effects, and the second is to construct three-dimensional representations of the images it receives. For undistortion, calibration patterns are used to make straight line straight to compensate for the effects of radial lens distortion. How the correction of the distortion is done obviously depends on the used distortion model. We use a simplified version of Brown's model (barrel distortion) to formulate the undistortion:

The raw-camera-image contains distorted image points:

$$\mathbf{x}_D = (x_D, y_D)$$

We seek the undistorted image points:

$$\mathbf{x}_U = (x_U, y_U)$$

Brown's model is given as:

$$\mathbf{x}_U = \mathbf{x}_D + L(r) \cdot (\mathbf{x}_D - \mathbf{x}_C)$$

Where

$$L(r) = K_1 r^2 + K_2 r^4 + \ldots$$
$$r = \sqrt{(x_D - x_C)^2 + (y_D - y_C)^2}$$

Stereo Calibration

Now back to the first three parts, Stereo calibration is the process of computing the geometrical relationship between the two cameras in space. In contrast, stereo rectification is the process of "correcting" the individual images so that they appear as if they had been taken by two cameras with row-aligned image planes.
Stereo calibration depends on finding the rotation matrix R and translation vector T between the two cameras. For any given 3D given point P in object coordinates, single-camera calibration can be used for two cameras to put P in the camera coordinates $P_l = R_l*P + T$ and $P_r = R_r*P + T_r$ for the left and right camera respectively. It it also given by the previous calculation that $P_l = RT*(P_r - T)$. Solving the three equation,

$$R = R_r(R_l)^T$$

$$T = T_r - RT_l$$

To be clear on what stereo calibration gives you: the rotation matrix will put the right camera in the same plane as the left camera; this makes the two image planes coplanar but not row-aligned.

Stereo Rectification

We want the image rows between the two cameras to be aligned after rectification so that stereo correspondence (finding the same point in the two different camera views) will be more reliable and computationally tractable. Note that reliability and computational efficiency are both enhanced by having to search only one row for a match with a point in the other image. The result of aligning horizontal rows within a common image plane containing each image is that the epipoles themselves are then located at infinity. That is, the image of the center of projection in one image is parallel to the other image plane. But because there are an infinite number of possible frontal parallel planes to choose from, we will need to add more constraints. These include maximizing view overlap and/or minimizing distortion, choices that are made by the algorithms discussed in what follows.
There are many ways to compute our rectification terms, of which OpenCV implements two: (1) Hartley's algorithm [Hartley98], which can yield uncalibrated stereo using just the fundamental matrix; and (2) Bouguet's algorithm, which uses the rotation and translation parameters from two calibrated cameras. Hartley's algorithm can be used to derive structure

from motion recorded by a single camera but may (when stereo rectified) produce more distorted images than Bouguet's calibrated algorithm. In situations where you can employ calibration patterns—such as on a robot arm or for security camera installations—Bouguet's algorithm is the natural one to use.

Uncalibrated stereo rectification: Hartley's algorithm

Hartley's algorithm attempts to find homographies that map the epipoles to infinity while minimizing the computed disparities between the two stereo images; it does this simply by matching points between two image pairs. Thus, we bypass having to compute the camera intrinsics for the two cameras because such intrinsic information is implicitly contained in the point matches. Hence we need only compute the fundamental matrix, which can be obtained from any matched set of seven or more points between the two views of the scene. The advantage of Hartley's algorithm is that online stereo calibration can be performed simply by observing points in the scene. The disadvantage is that we have no sense of image scale.

Assuming we have the fundamental matrix F, which required seven or more points to compute, Hartley's algorithm proceeds as follows (see Hartley's original paper [Hartley98] for more details).

1. We use the fundamental matrix to compute the two epipoles via the relations $Fe_l = 0$ and $(e_r)^T F = 0$ for the left and right epipoles, respectively.
2. We seek a first homography H_r , which will map the right epipole to the 2D homogeneous point at infinity $(1, 0, 0)^T$. Since a homography has seven constraints (scale is missing), and we use three to do the mapping to infinity, we have 4 degrees of freedom left in which to choose our H_r . These 4 degrees of freedom are mostly freedom to make a mess since most choices of H_r will result in highly distorted images. To find a good H_r, we choose a point in the image where we want minimal distortion to happen, allowing only rigid rotation and translation not shearing there. A reasonable choice for such a point is the image origin and we'll further assume that the epipole $(e_r)^T = (f , 0, 1)$ lies on the x-axis (a rotation matrix will accomplish this below). Given these coordinates, the matrix

$$G = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/k & 0 & 1 \end{pmatrix}$$

   will take such an epipole to infinity.
3. For a selected point of interest in the right image (we chose the origin), we compute the translation T that will take that point to the image origin (0 in our case) and the rotation R that will take the epipole to $(e_r)^T = (f , 0, 1)$. The homography we want will then be $H_r = GRT$.
4. We next search for a matching homography H_l that will send the left epipole to infinity and align the rows of the two images. Sending the left epipole to infinity is easily done by using up three constraints as in step 2. To align the rows, we just use the fact that aligning the rows minimizes the total distance between all matching points between the two images. That is, we find the Hl that minimizes the total disparity in left -right matching points $\sum_i d(H_l p_i^l, H_r p_i^r)$.These two homographies define the stereo rectification.

If our cameras have roughly the same parameters and are set up in an approximately horizontally aligned frontal parallel configuration, then our eventual rectified outputs from Hartley's algorithm will look very much like the calibrated case described next.
If we know the size or the 3D geometry of objects in the scene, we can obtain the same results as the calibrated case.

Calibrated stereo rectification: Bouguet's algorithm

Given the rotation matrix and translation (R, T) between the stereo images, Bouguet's algorithm for stereo rectification simply attempts to minimize the amount of change reprojection produces for each of the two images (and thereby minimize the resulting reprojection distortions) while maximizing common viewing area.
To minimize image reprojection distortion, the rotation matrix R that rotates the right camera's image plane into the left camera's image plane is split in half between the two cameras; we call the two resulting rotation matrixes r_l and r_r for the left and right camera, respectively. Each camera rotates half a rotation, so their principal rays each end up parallel to the vector sum of where their original principal rays had been pointing. Such a rotation puts the cameras into coplanar alignment but not into row alignment. To compute the Rrect that will take the left camera's epipole to infinity and align the epipolar lines horizontally, a rotation matrix is created by starting with the direction of the epipole e_1 itself. Taking the principal point (c_x, c_y) as the left image's origin, the (unit normalized) direction of the epipole is directly along the translation vector between the two cameras' centers of projection:

$$e_1 = \frac{T}{\|T\|}$$

The next vector, e_2, must be orthogonal to e1 but is otherwise unconstrained. For e_2, choosing a direction orthogonal to the principal ray (which will tend to be along the image plane) is a good choice. This is accomplished by using the cross product of e_1 with the direction of the principal ray and then normalizing so that generating another unit vector:

$$e_2 = \frac{[-T_y \; T_x \; 0]^T}{\sqrt{T_x^2 + T_y^2}}$$

The third vector is just orthogonal to e_1 and e_2; it can be found using the cross product:

$$e_3 = e_1 \times e_2$$

Our matrix that takes the epipole in the left camera to infinity is then:

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}$$

This matrix rotates the left camera about the center of projection so that the epipolar lines become horizontal and the epipoles are at infinity. The row alignment of the two cameras is then achieved by setting:

$$R_l = R_{\text{rect}} r_l$$

$$R_r = R_{\text{rect}} r_r$$

We will also compute the rectified left and right camera matrices M_{rect_l} and M_{rect_r} but return them combined with projection matrices P_l and P_r:

$$P_l = M_{rect\_l} P_l' = \begin{bmatrix} f_{x\_l} & \alpha_l & c_{x\_l} \\ 0 & f_{y\_l} & c_{y\_l} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

And

$$P_r = M_{rect\_r} P_r' = \begin{bmatrix} f_{x\_r} & \alpha_r & c_{x\_r} \\ 0 & f_{y\_r} & c_{y\_r} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

(here α_l and α_r allow for a pixel skew factor that in modern cameras is almost always 0). The projection matrices take a 3D point in homogeneous coordinates to a 2D point in homogeneous coordinates as follows:

$$P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

where the screen coordinates can be calculated as (x/w, y/w). Points in two dimensions can also then be reprojected into three dimensions given their screen coordinates and the camera intrinsics matrix. The reprojection matrix is:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/T_x & (c_x - c_x')/T_x \end{bmatrix}$$

Here the parameters are from the left image except for cx', which is the principal point x coordinate in the right image. If the principal rays intersect at infinity, then $cx = cx'$ and the term in the lower right corner is 0. Given a two-dimensional homogeneous point and its associated disparity d, we can project the point into three dimensions using:

$$Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

The 3D coordinates are then (X/W, Y/W, Z/W).

Rectification Map

The process of rectification is illustrated in Figure 12-11. As shown by the equation flow in that figure, the actual rectification process proceeds backward from (c) to (a) in a process known as reverse mapping. For each integer pixel in the rectified image (c), we find its coordinates in the undistorted image (b) and use those to look up the actual (floating-point) coordinates in the raw image (a). The floating-point coordinate pixel value is then interpolated from the nearby integer pixel locations in the original source image, and that value is used to fill in the rectified integer pixel location in the destination image (c). After the rectified image is filled in, it is typically cropped to emphasize the overlapping areas between the left and right images.

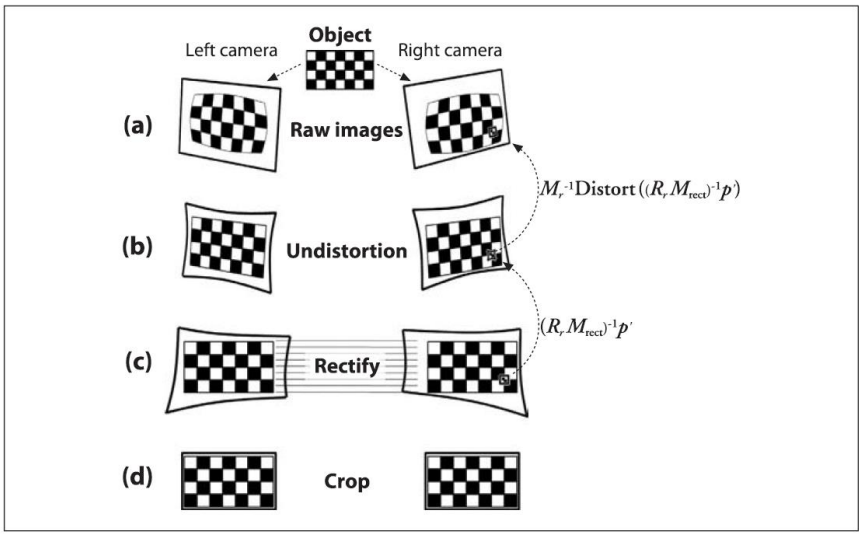

Figure 12-11. Stereo rectification: for the left and right camera, the raw image (a) is undistorted (b) and rectified (c) and finally cropped (d) to focus on overlapping areas between the two cameras; the rectification computation actually works backward from (c) to (a)

Stereo Correspondence

Stereo correspondence—matching a 3D point in the two different camera views—can be computed only over the visual areas in which the views of the two cameras overlap. Once the physical coordinates of the cameras or the size of objects in the scene are known, the depth measurements from the triangular disparity measures $d = x_l - x_r$ between the corresponding points in the two different camera views.

OpenCV implements a fast and effective block-matching stereo algorithm. Which works by using "sum of absolute differences" window to find matching between the left and right stereo rectified images. This algorithm finds only strongly matching (high-texture) points between the two images. Thus, in a highly textured scene such as might occur outdoors in a forest, every pixel might have computed depth. In a very low-textured scene, such as an indoor hallway, very few points might register depth. There are three stages to the block-matching stereo correspondence algorithm, which works on undistorted, rectified stereo image pairs:

1. Prefiltering to normalize image brightness and enhance texture.
2. Correspondence search along horizontal epipolar lines using an SAD window.
3. Postfiltering to eliminate bad correspondence matches.

Correspondence is computed by a sliding SAD window. For each feature in the left image, we search the corresponding row in the right image for a best match. After rectification, each row

is an epipolar line, so the matching location in the right image must be along the same row (same y-coordinate) as in the left image; this matching location can be found if the feature has enough texture to be detectable and if it is not occluded in the right camera's view). If the left feature pixel coordinate is at (x_0, y_0) then, for a horizontal frontal parallel camera arrangement, the match (if any) must be found on the same row and at, or to the left of, x_0; see Figure 12-13.

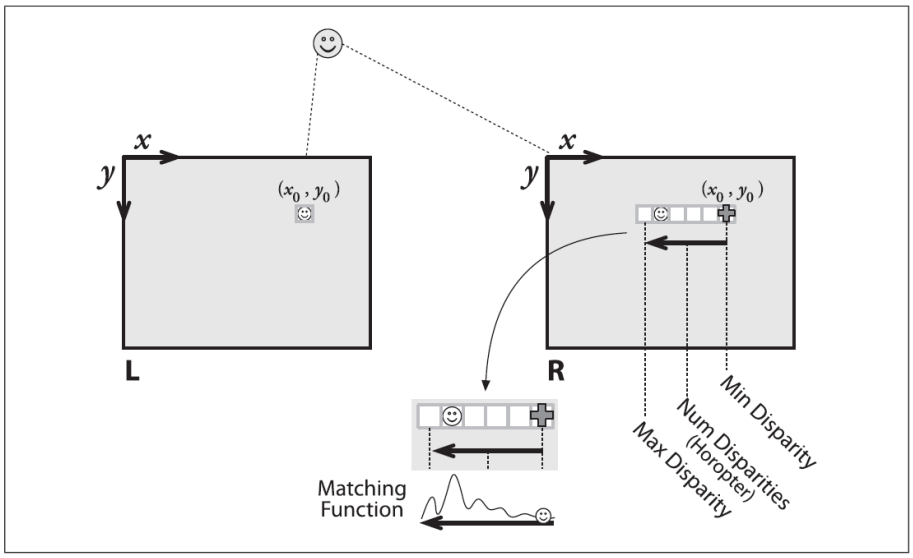

Figure 12-13. Any right-image match of a left-image feature must occur on the same row and at (or to the left of) the same coordinate point, where the match search starts at the minDisparity point (here, 0) and moves to the left for the set number of disparities; the characteristic matching function of window-based feature matching is shown in the lower part of the figure

Depth Maps from 3D Reprojection

Assume that we have a perfectly undistorted, aligned, and measured stereo image as shown in Figure 12-4: two cameras whose image planes are exactly coplanar with each other, with exactly parallel optical axes.

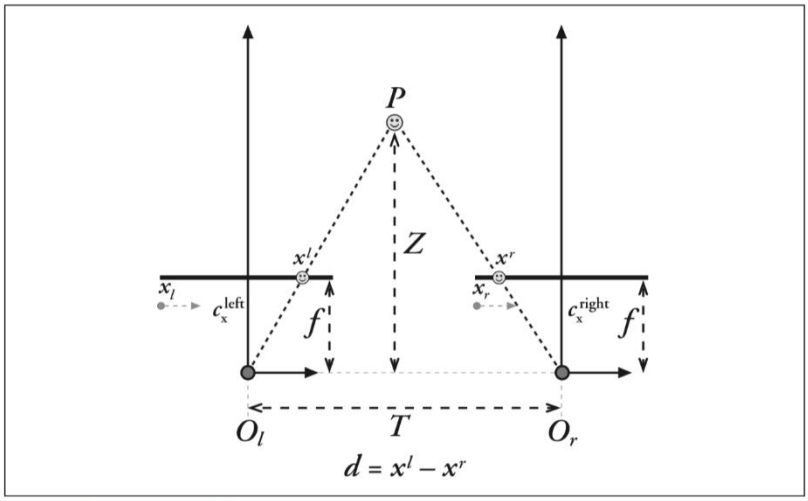

Figure 12-4. With a perfectly undistorted, aligned stereo rig and known correspondence, the depth Z can be found by similar triangles; the principal rays of the imagers begin at the centers of projection O_l and O_r and extend through the principal points of the two image planes at c_l and c_r

we can easily derive the depth Z by using similar triangles. Referring to the figure, we have:

$$\frac{T-(x^l-x^r)}{Z-f}=\frac{T}{Z} \quad \Rightarrow \quad Z=\frac{fT}{x^l-x^r}$$

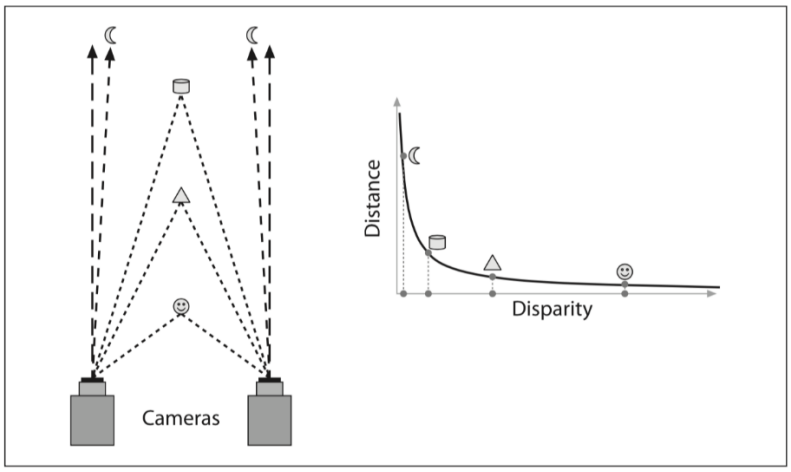

Figure 12-5. Depth and disparity are inversely related, so fine-grain depth measurement is restricted to nearby objects

From Figure 12.5, it is obvious that depth is inversely proportional to disparity. Furthermore, then have a nonlinear relationship.

Recall the 4-by-4 reprojection matrix Q introduced in the section on calibrated stereo rectification. Also recall that, given the disparity d and a 2D point (x, y), we can derive the 3D depth using

$$Q\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix}=\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

where the 3D coordinates are then (X/W, Y/W, Z/W). Remarkably, Q encodes whether or not the cameras' lines of sight were converging (cross eyed) as well as the camera baseline and the principal points in both images. As a result, we need not explicitly account for converging or frontal parallel cameras and may instead simply extract depth by matrix multiplication.