CHAPTER 34

Stereopsis

If you have two eyes, what you see in each eye is somewhat different, but most people will not see doubled images. The same point will appear in somewhat different places in the left eye and in the right eye. Most people will *fuse* these images into a single representation. This representation ascribes the shift in position to depth. In 1956, the belief that the process of fusion had something to do with object properties was laid to rest by a demonstration due to Bela Julesz. Most people who look at appropriately constructed random patterns will see depth (Figure 34.1).

Recovering 3D structure from a pair of images is known as *stereopsis*. There are many ways to get a pair of images. Section 21.3 assumes that you have a specialized camera setup, configured to make stereopsis easy. More complex camera geometries appear in Section 21.3, and what happens when you have more than two cameras starts in Section 21.3 and continues in Section 21.3

34.1 STEREO IN AN EASY GEOMETRY

There are some important geometric constraints that apply to stereopsis. You cannot use this procedure to reconstruct depth from a single image. So, for example, if you have only one camera, you will need to move it to get a second image. You cannot reconstruct anything that appears in only one of two images. So, for example, if your eyes are arranged so the fields of view overlap, you might have stereo, but if the fields of view do not overlap, you cannot have stereo. A rule of thumb says that predator eyes have overlapping fields of view (making it easier to gauge attacks, etc.) and prey eyes have non-overlapping fields of view (so they can see more stuff). Even if the fields of view of your eyes (cameras, etc) overlap, there may be points you see in one but not the other (Figure 34.3).

34.1.1 An Easy Geometry

Figure **??** shows an easy camera geometry. The second camera is translated parallel to the camera plane, and there is no rotation. The first camera is a canonical camera, with focal point at the origin and image plane at $Z = 1$. This camera maps (X, Y, Z) in 3D to $(X/Z, Y/Z)$. The second camera is obtained by translating the first camera along the X axis. It has focal point at $(B, 0, 0)$. In turn, it maps (X, Y, Z) *in the world coordinate system* to $(X/Z - B/Z, Y/Z)$. The point being viewed appears at different locations in the two images. The difference in position is known as the *disparity*. In this configuration, the disparity is $-B/Z$. The key question is which point in camera 1 corresponds to which point in camera 2. Disparity follows from correspondences, and depth follows from disparity.

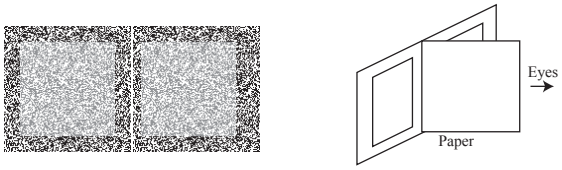

FIGURE 34.1: *The simplest* random dot stereogram *shows each eye a slightly different random pattern. Start with a random pattern (I thresholded a collection of IID normal random variables). Now select some of the pattern – the block indicated on the* **left** *side – and shift it to form the pattern on the* **right** *side. This forms the stereogram of Figure* **??**. *Most people will see a figure in depth if they view this pair properly. Your left eye should see the left side and your right eye should see the right side. Some people can achieve this by just looking at the figure and letting their eyes drift. I can do this, but find it trying, and usually prefer to hold up a sheet of paper blocking each eye's view of the other side, as shown on the* **far right**.

34.1.2 Contraints on Matches

Correspondence is harder than it might seem, because there may be points in camera 1 that have no corresponding point in camera 2, and vice versa (Figure 34.3). There are further geometric constraints on the match that are useful, if not universally true. A *uniqueness constraint* requires that each point in one image have no more than one matching point in the other. As Figure **??** demonstrates, this constraint does not always apply, but the conditions where it is violated can be ignored. An *ordering* constraint requires that matched points appear in the same order in right and left image. As Figure **??** demonstrates, this constraint does not always apply. The conditions where it is violated are usually ignored, apparently safely.

For a match to be valid, the camera must be able to see the point in both views. Points do not float in space, but lie on surfaces. If the surface is tilted too much toward one or another camera, the surface itself will make the point invisible in the other camera. Figure 34.5 illustrates this idea. This results in a constraint on the surface orientation. Surface orientation is given by the derivative of depth, so there is some constraint on the acceptable disparity gradient. The details depend on cameras, viewpoint and so on, and are usually ignored. The constraint typically appears as a threshold on acceptable disparity gradient and is known as the *disparity gradient constraint*.

In the epipolar geometry of Figure 34.4, this constraint boils down to the following. Assume K is the threshold. If (x_1, y_1) matches (x_1', y_1'), then $(x_1+\Delta x, y_1)$ can match only points in the range from $x_1' + (1 - K)\Delta x$ to $x_1' + (1 + K)\Delta x$.

34.1.3 Simple Matching

The simplest procedure is to produce a representation of each pixel in each view – for example, a vector of filter outputs – and then match these. The simplest match from camera 1 to camera 2 proceeds as follows. Choose a point in camera 1, and

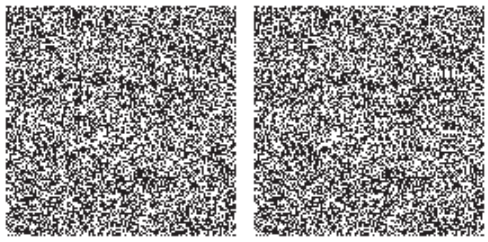

FIGURE 34.2: *A much more complicated random dot stereogram constructed with this method. Most people will see a set of boxes forming a pyramid hovering over the plane.*

obtain its representation. Proceed along the corresponding epipolar line in camera 2, comparing the representation of each point to that of the point in camera 1, and choose the best point. This procedure will give you a match in camera 2 for *any* point in camera 1, which can't be right because there are some points in camera 1 that do not have a match in camera 2. You can improve the procedure by matching both ways. For each point $\mathbf{x}_1$ in 1, find the best match $\mathbf{b}_2$ in 2. For each point $\mathbf{x}_2$ in 2, find the best match $\mathbf{b}_1$ in 1. If the matches are consistent – that is, if $\mathbf{x}_1 = \mathbf{b}_1 \Leftrightarrow \mathbf{x}_2 = \mathbf{b}_2$ – keep the match.

34.1.4 Dynamic Programming

Matching points individually like this doesn't take into account interactions between matches. These are important, because the points you match lie on surfaces, which are likely to be at least somewhat smooth. Further, the disparity gradient constraint limits these interactions. You can obtain improved matching using dynamic programming along each epipolar line.

Assume you have N locations on the left epipolar line given by $y = y_o$. At each location, you must determine the corresponding location on the right epipolar line. Assume the ordering constraint applies. Set up a grid of pairs of (image 1 location, image 2 location) for this pair of epipolar lines. Figure 34.8 shows this grid for a very small (8 pixels across) image. Each node represents a possible correspondence, so each node is associated with a cost. For example, the node representing a correspondence between (i, y_o) and (j, y_o) is associated with a cost comparing some image representation at (i, y_o) in the left image with the same representation at (j, y_o) in the right image. The representation could be pixel

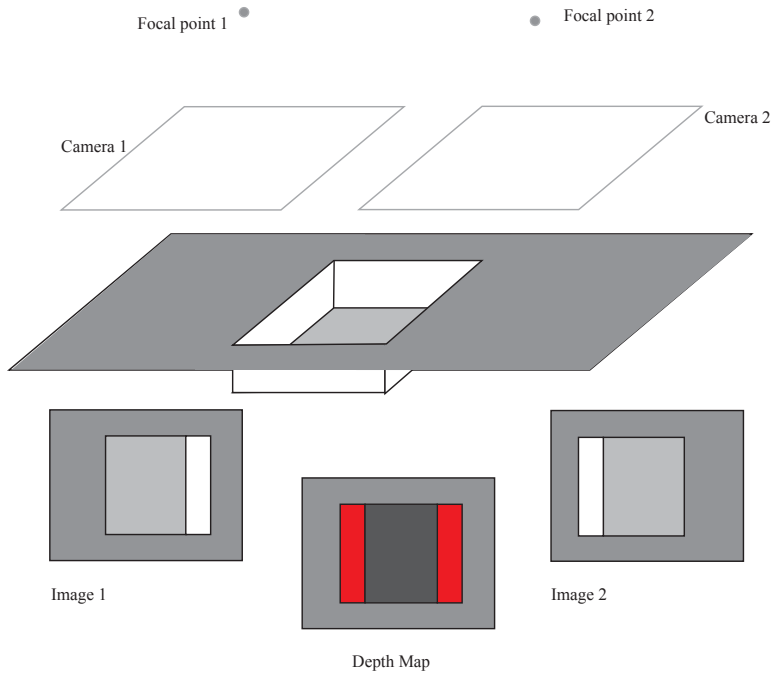

FIGURE 34.3: **Top** *shows two pinhole cameras viewing a rectangular depression in a flat surface. As the images show, camera on the left can see the right wall, and that on the right can see the left wall. This means that these walls cannot be reconstructed directly using trigonometry, and so the depth map will have holes in it. The depth map here is shown with a fairly common convention, where nearer surfaces are lighter and farther surfaces are darker. I have marked the holes in* **red**.

intensity, pixel color, or a vector of filter outputs. Using color or intensity yields what is known a *photometric consistency loss*. Write $C(i, j, y_o)$ for the cost that compares the image at (i, y_o) to the image at (j, y_o). Very often

$$C(i, j, y_o) = \|\mathcal{I}(i, y_o) - \mathcal{I}(j, y_o)\|^2.$$

A solution will be a directed path through the nodes representing points that correspond (**blue** in Figure 34.8). If i corresponds to j, the node (i, j) appears in a solution.

You can insert directed edges from a node to any node that could be the next node in a solution. Some of the many edges are shown in Figure 34.8, with edges that cannot occur in **red**. Matching proceeds along the left epipolar line, so directed edges can never go backward (which is why edge A is in red in that figure). The ordering constraint means that directed edges can never go down (which is why edge B is in red in that figure). The uniqueness constraint means that directed edges cannot be vertical (which is why edge C is red in that figure). The disparity gradient constraint means that other directed edges are impossible as well, because

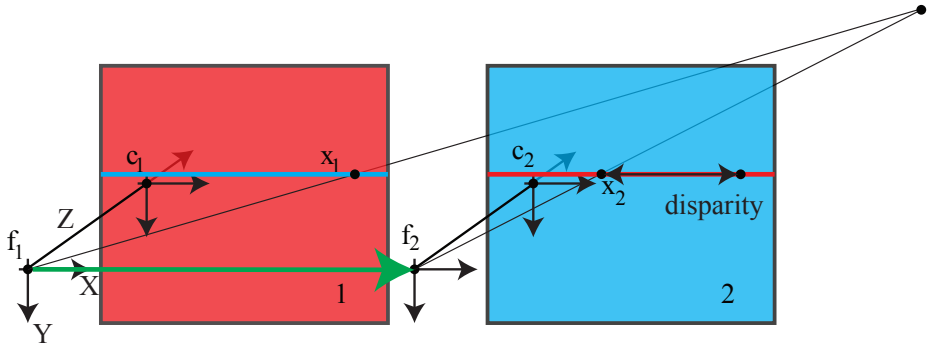

FIGURE 34.4: *In the simplest geometry for stereo, the second (***blue** *image plane) camera has been translated along the X axis in the first camera's coordinate system. Both cameras are calibrated, so you can work in world coordinates, and the two image planes are coplanar. This geometry means the epipolar lines in each image are the lines of constant y. The epipolar line corresponding to a point (x_1, y_1) in camera 1 is the line $y = y_1$ in camera 2. The* **blue** *line in the* **red** *image plane is the epipolar line corresponding to $\mathbf{x}_2$; the* **red** *line in the* **blue** *image plane is the epipolar line corresponding to $\mathbf{x}_1$. This means that when you pass from camera 1 to camera 2, $\mathbf{x}_1$ is at the same height but has "slid" along the epipolar line. The distance it has moved is the disparity. image plane*

they imply too large a disparity (which is why edge D is in red in that figure). **exercises** Smoothness implies that large changes in disparity are deprecated, so you can associate a cost with a legal edge from (i, j) to (k, l) like

$$C(i, j; k, l) = \alpha \left[(j - i) - (k - l)\right]^2$$

Finding the best match is now the dynamic programming problem of finding the lowest cost path from start column to end column in the directed graph. **exercises**

34.1.5 Stereo as a Conditional Random Field

The dynamic programming procedure has a problem. It obtains the best solution for each row independent of each other. This means that solutions will have a characteristic streaky appearance, because the solution for row j ignores the solution for row $j - 1$ (Figure 34.8). You could try and fix this by solving for row 1; then solving for row 2 conditioned on row 1 (you would add some terms to the cost function, but it isn't worth expanding them here); and so on. The difficulty with this approach is the disparity map you obtain depends on the row you start with. You could equally start at the bottom of the image and work your way up, and come up with a somewhat different disparity map.

An alternative is to set up a much harder optimization problem that takes into account pixels in rows above and below the current pixel. Do this by setting up a graph where each vertex corresponds to a pixel location, and each edge corresponds

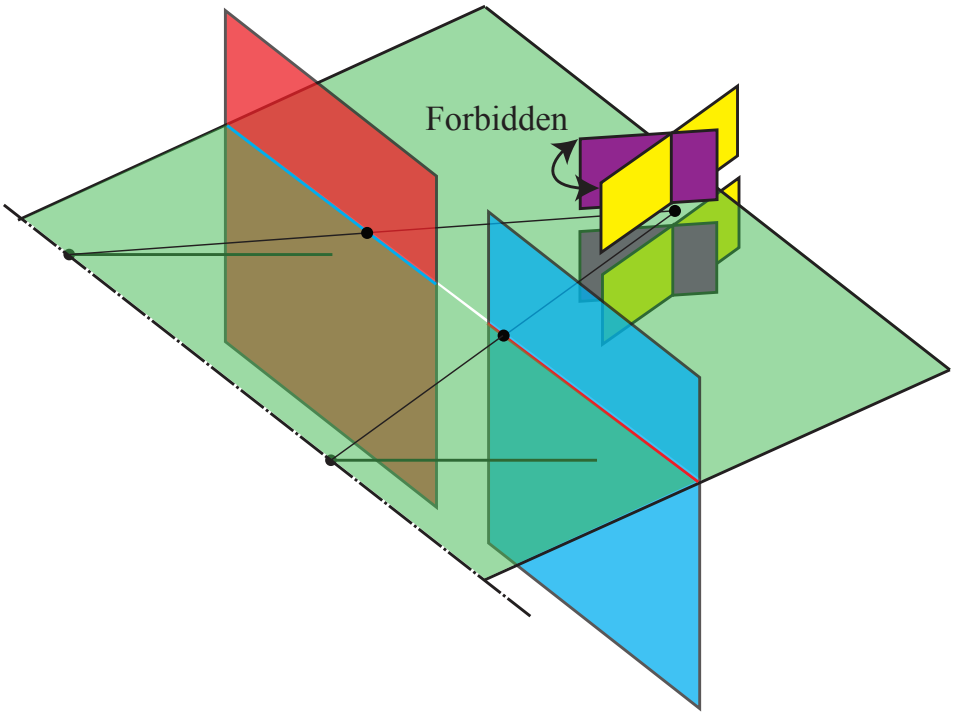

FIGURE 34.5: *There is some limit on the disparity gradient that depends on the particular point being viewed. This figure shows the cameras of Figure 34.4 viewing a point. This point lies on a surface, as points do. The surface on which the point lies cannot obstruct the view from either camera (or you wouldn't be able to see the point). This means that some orientations (*forbidden *in the figure) are not possible. You can compute surface orientation from the gradient of depth, and so from the gradient of disparity. This means that, near a matched point, some disparities cannot be possible. A detailed expression is tricky to use, and it is usual to require that disparity gradients be below some threshold.*

to a pair of pixels whose disparity you want to compare. Write $\mathcal{N}(i,j)$ for the set of pixels that are connected to the pixel at i, j by an edge – these are *neighbors* to the pixel at i, j.

$\mathcal{N}(i,j)$ could just be

$$\{(i+1,j)\}$$

(which would get us the dynamic programming case). Alternatively, it could be

$$\{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\}$$

(the *four neighbors*). It could be

$$\{(i+1,j),(i-1,j),(i,j+1),(i,j-1),(i-1,j-1),(i+1,j-1),(i-1,j+1),(i+1,j+1)\}$$

(the *eight neighbors*).

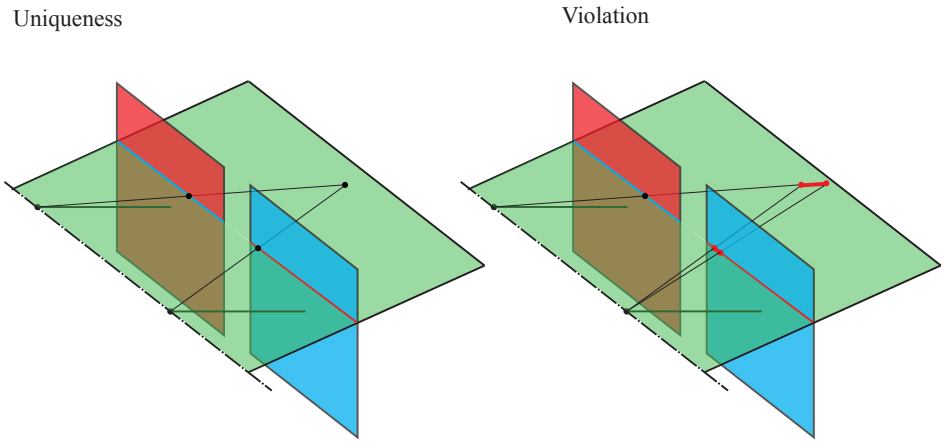

FIGURE 34.6: *The uniqueness constraint requires that matches be unique. Each point on one side should have at most one match on the other (as in the* **left** *drawing). It can be violated if you view a straight object in exactly the right viewing position (***right** *- the* **red** *interval), but this requires a special viewpoint. It is usual to ignore this effect and assume that the constraint applies.*

You want to know a disparity value for each pixel in the image. Write $\delta_{i,j}$ for the disparity at i, j in the left image. A photometric consistency loss follows easily. Very often

$$C(\delta; i, j) = \|\mathcal{I}(i, j) - \mathcal{I}(i + \delta, j)\|^2.$$

Associated with each edge will be a cost function. This cost function penalizes pairs of disparities across the edge that are undesirable. Now write a cost function that compares disparity across pairs of pixels. For example, the cost function could be

$$B(u, v) = \begin{cases} \|u - v\|^2 & \text{if } \|u - v\| < k \\ L & \text{otherwise} \end{cases}$$

for L some large value. Now quantize the disparity to a fixed set of V values. You must choose the $\delta_{i,j}$ to minimize

$$\sum_{i,j} \left[\sum_{u \text{ in values}} \left[\mathbb{I}_{[\delta_{i,j}=u]}(\delta_{i,j}) C(u; i, j) \right] \right] + \sum_{i,j} \left[\sum_{k,l \in \mathcal{N}(i,j)} \left[\sum_{u,v \text{ in values}} \left[\mathbb{I}_{[\delta_{i,j}=u]}(\delta_{i,j}) \mathbb{I}_{[\delta_{k,l}=v]}(\delta_{k,l}) B(u, v) \right] \right] \right].$$

34.1.6 Rectification

34.2 STEREO BY REGRESSION

34.3 USING STEREO GEOMETRY WITH ACTIVE SOURCES

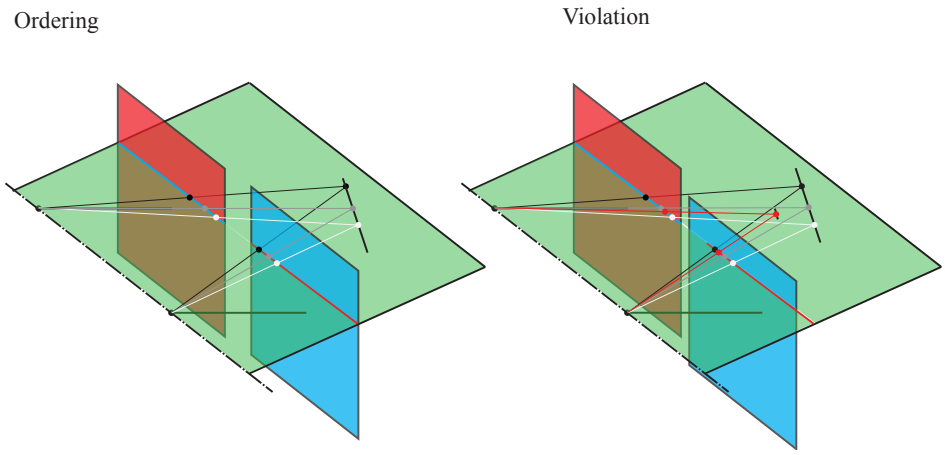

FIGURE 34.7: *The ordering constraint requires that matches appear in the same order on left and right, as on the* **left**, *where the points are matched in order* **light gray**, **mid gray**, **black** *on both sides. This constraint can be violated, as on the* **right** *side. Violation requires looking "into" a hole in a surface. Here the* **red** *point lies on a surface in front of the main piece, but the other points are still visible to both cameras. Points appear in a different order on the two sides. It is usual to ignore this effect and assume that the constraint applies.*

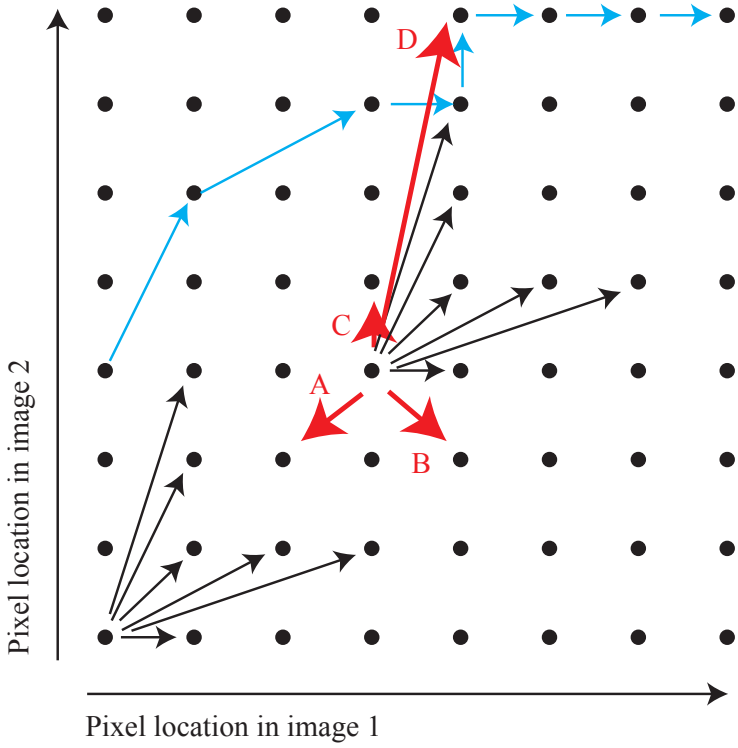

FIGURE 34.8: *Drawing the matching process for a single epipolar line as a graph exposes matching as a dynamic programming problem. The details appear in the text.*

MANY IMAGES