CHAPTER 32

Stereopsis: The Basics

Bolt two calibrated cameras together in a rig. Ensure the cameras are arranged so that at least some of what you see in the left camera is also visible in the right camera. You should know the translation and rotation between the cameras (you built the rig). Obtain left and right images at the same time, so nothing moves. Then you can recover the 3D configuration of any point you can see in both images using triangulation (Section 36.3). Most people have an arrangement like this in their heads (two eyes in known relative configuration).

You can extend this construction. Assume you have two images of a scene where nothing moves, obtained using cameras with known calibration. You can assume that these images were taken at the same time because nothing moved in the scene. Assume further you can see some parts of the scene in both images. Then you can: recover the essential matrix (Section 36.3); recover the relative configuration of the cameras up to scale (Section 36.3); choose a scale; and reconstruct by triangulation.

The underlying trick here is you need to be able to tell which points in the left image correspond to which in the right. You want to construct correspondences for as many points as possible, because that will lead to a dense 3D reconstruction. Constructing dense correspondences for two images like this is usually called *stereopsis* or *stereo*. If there are more images, the process is often called *multi-view stereopsis*.

Stereopsis is at the root of a very rich and interesting tree. Section 36.3 assumes that you have a specialized camera setup, configured to make stereopsis easy. More complex camera geometries appear in Section 36.3. What happens when things could move in the scene appears in Chapter 36.3. What happens when you have more than two cameras starts in Chapter 36.3 and continues in Chapter 36.3.

32.1 STEREO IN AN EASY GEOMETRY

You cannot use stereo 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 but you will see more stuff. The result is a useful biological rule of thumb. Generally, 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).

32.1.1 An Easy Geometry

Figure **??** shows an easy camera geometry. It is quite usual to build camera rigs so that the cameras are in this geometry. In this arrangement, the two cameras

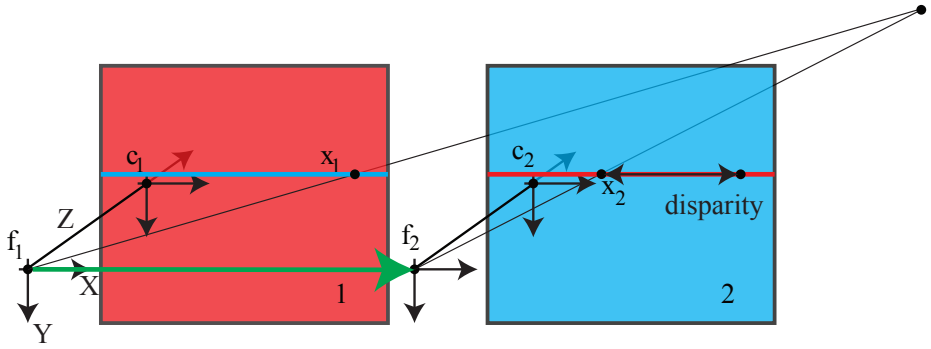


FIGURE 32.1: *In the simplest geometry for stereo, the second (***blue** *image plane) camera has been translated along the X axis in the first camera's coordinate system. Both cameras are calibrated, so you can work in world coordinates, and the two image planes are coplanar. This geometry means the 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.*

are the same. They are calibrated, so you can treat 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 $(x, y) = (X/Z, Y/Z)$. The second camera is obtained by translating the first camera along the X axis. It has focal point at $(B, 0, 0)$.

Definition: 32.1 *Baseline*

The distance between the two focal points in a stereo setup is known as the *baseline*.

The configuration has baseline B. The second camera maps (X, Y, Z) *in the world coordinate system* to $(X/Z - B/Z, Y/Z)$. This configuration has the extremely useful property that the epipolar lines are the lines $y =$ constant. You will occasionally see this configuration referred to as an *epipolar configuration*. The point being viewed appears at different locations in the two images. The difference in position is known as the *disparity*.

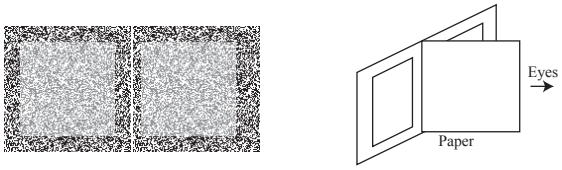


FIGURE 32.2: *To build a random dot stereogram of a block, I started with a random pattern, selected the block indicated on the* **left** *side, and shifted it to form the pattern on the* **right** *side. Most people will see a figure in depth if they view this pair properly. It is usually easier to hold up a sheet of paper blocking each eye's view of the other side, as shown on the* **far right**.

Definition: 32.2 *Disparity*

Two cameras with different focal points view a point in 3D. The point being viewed will appear at different points in these two cameras. The difference in location is referred to as the *disparity*. Disparity is always inversely related to depth, and proportional to baseline.

Notice there are two disparities: you can compute the disparity in image two with respect to image one, or the disparity in image one with respect to image two. At corresponding points in images one and two, the disparity will be the same up to a sign. BUT corresponding points are NOT in the same place (which is the whole point of disparity) so the disparity maps look somewhat dissimilar.

Remember this: *There is a practical maximum to the size of the disparity, otherwise the object pierces the camera lens. There is some resolution limit for position in the image, and you cannot measure disparities below this limit. In turn, very large depths are difficult to estimate without a large baseline. Large baselines create other engineering problems. For example, a stereo rigs with a large baseline is difficult to move around, presents increased risks of collisions, and so on.*

Stereo is common, but not universal, in people. If you have two eyes, what you see in each eye is somewhat different, but most people will not see doubled images. Instead, most people will interpret the disparity as a cue to depth, and *fuse* the two images into a single representation. This representation ascribes the shift in position to depth. The effect was first recorded by Charles Wheatstone in 1838 (see *Contributions to the Physiology of Vision.—Part the First. On some remarkable, and hitherto unobserved, Phenomena of Binocular Vision.* by Charles Wheatstone). How stereo works was controversial for some time.

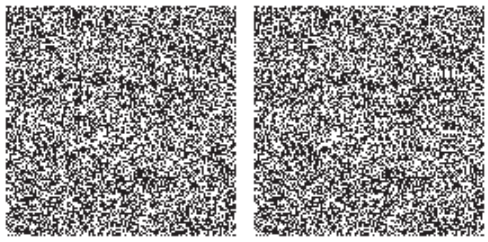


FIGURE 32.3: *A much more complicated random dot stereogram. Most people will see a set of boxes forming a pyramid hovering over the plane.*

32.1.2 Random Dot Stereograms

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. 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 for Figure 32.2). Use this for the left eye. Now select some of the pattern, and shift it to get the pattern for the right eye. Show the two patterns in a way that the left eye sees its pattern and the right eye sees its (Figure 32.2).

Most people will see a figure in depth if they view a pair of images created like this 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 as shown in Figure 32.2. Figure 32.3 shows a much more challenging random dot stereogram.

32.1.3 Photometric Consistency and Simple Matching

All stereo matching exploits the idea that the points that match in left and right image "look like" one another. This *photometric consistency* constraint is very general; details in the box below.

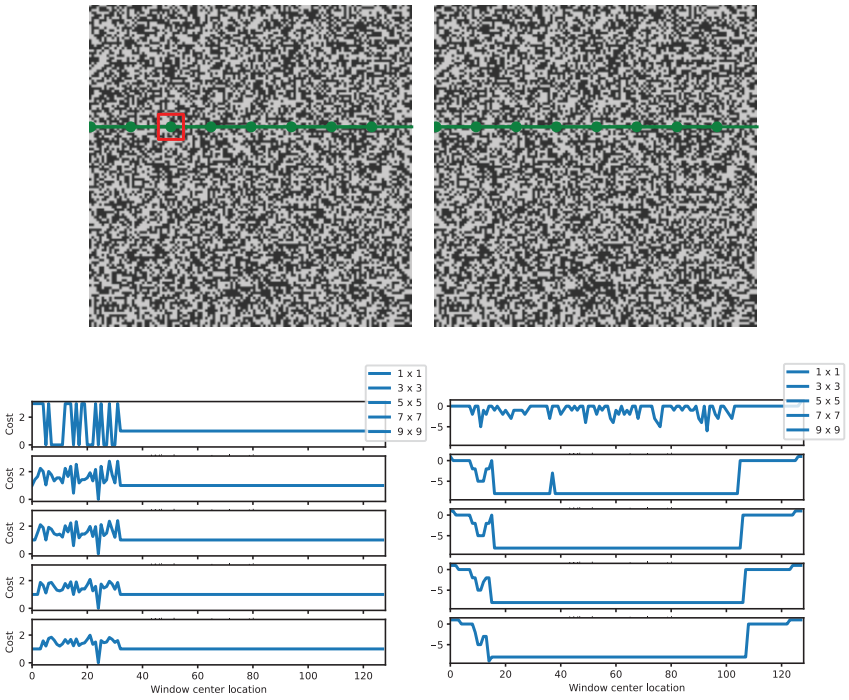


FIGURE 32.4: *Take windows at the location indicated by the* **red** *box in the random dot stereogram on the* **top left**, *and match using weighted sum of squared differences to all locations with negative disparity along the epipolar line (***green***) in the other image (***top right***). This yields the matching cost on the* **bottom left**, *computed for various window sizes (***inset***). The smallest matching cost yields the best match, and so an estimate of disparity.* **Bottom right** *shows the disparity for each point on the epipolar line on the* **top left**.

Definition: 32.3 *Photometric consistency*

The *photometric consistency constraint* is the very powerful assumption is that the intensity of a point on an object does not change when the object or the camera moves.

Photometric consistency doesn't always apply (what you see in a mirror changes when you move your head). It is a powerful constraint because most surfaces are, approximately, matte and so photometric consistency applies at most points. Photometric consistency implies that matching points look the same, suggesting you can obtain matches with a simple procedure: for each point, look at all the acceptable matches, and choose the one that is most similar. By construction, photometric consistency applies to random dot stereograms.

This means that correspondence is extremely easy to establish in a random dot stereogram. Small windows of a random pattern almost never look like one another *unless* they match. Figure 32.4 illustrates a very simple matching procedure applied to a random dot stereogram. The possible correspondences all have negative disparities. For the figure, I used weighted sum of squared differences to score the match, but this is not the only possibility. The weights are gaussian, with a σ about half the size of the window.

Procedure: 32.1 *The simplest stereo matching*

For each point along an epipolar line in the left image:

- construct a window around the point;
- compare this window to all possible correspondences in the right image (more below);
- take the best matching window.

Figure 32.4 shows the costs for matching a particular point along an epipolar line and also the disparities computed for all points on that line. Matching a 1×1 window is a bad idea, because this is a single pixel and there are many bad matches. By the time the window is 3×3, the matches are mostly good. The stereogram of Figure 32.4 shows a box shaped depression on a flat background, so the disparities computed in Figure 32.4 make sense.

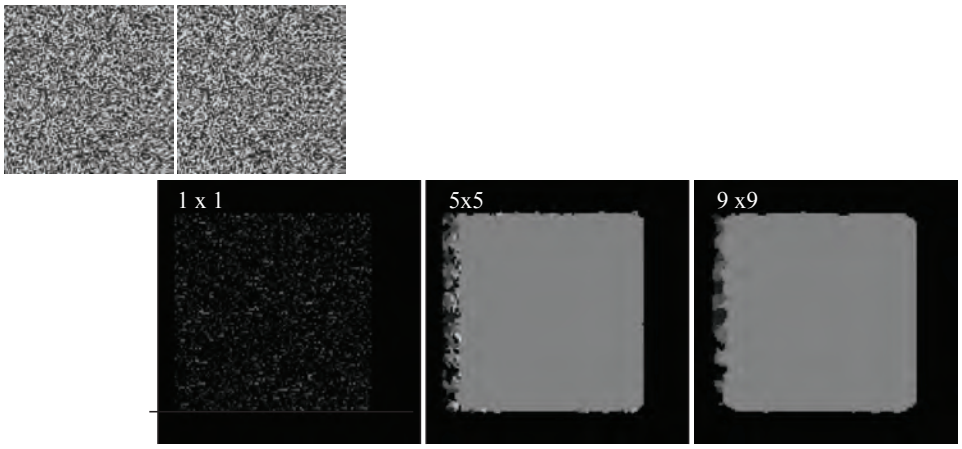


FIGURE 32.5: *Disparity maps for the random dot stereogram of Figure 32.4, for a simple window matching procedure, using window sizes from* 1×1 *to* 9×9*; details in text.*

Figure 32.5 shows disparity maps (the disparity at each pixel for the left hand image, estimated using the simple matching procedure) for three window sizes. You

should notice the tendency to make errors at depth (or disparity) discontinuities. This is because it may not be possible to match a window correctly at these points **exercises**.

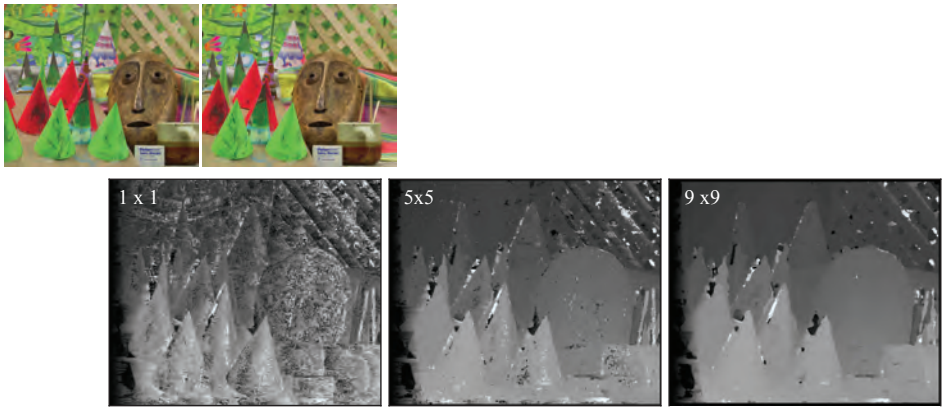


FIGURE 32.6: *Disparity maps for a stereo pair of cones, from the Middlebury datasets, for a simple window matching procedure, using window sizes from* 1×1 *to* 9×9*; details in text.* Image credit: *Stereo pair from the Middlebury dataset at* `https://vision.middlebury.edu/stereo/data/`*, disparities are mine.*

Random dot stereograms tend to flatter simple matching procedures, because the patterns are so distinctive. Figure 32.8 illustrates the basic difficulty encountered using this matching procedure in more challenging cases. Small windows match too easily, meaning the disparity field is noisy. Big windows tend to oversmooth the disparity **exercises**.

32.1.4 Improving Simple Matching

The simple matching procedure will give you a match in camera 2 for *any* point in camera 1. This simply can't be right because there are some points in camera 1 that do not have a match in camera 2 (Figure 32.7).

> **Remember this:** *There are usually points that can be seen in one image but not the other. This means that not every point in an image will have a match in the other. It also means that the disparity in image two with respect to image one and the disparity in image one with respect to image two have a quite complicated relationship. For many points, the disparities are equivalent up to sign. For others, one but not the other is meaningful.*

You can improve the procedure by matching both ways, as below. However, as Figure 32.8 suggests, there are significant difficulties that remain. When you

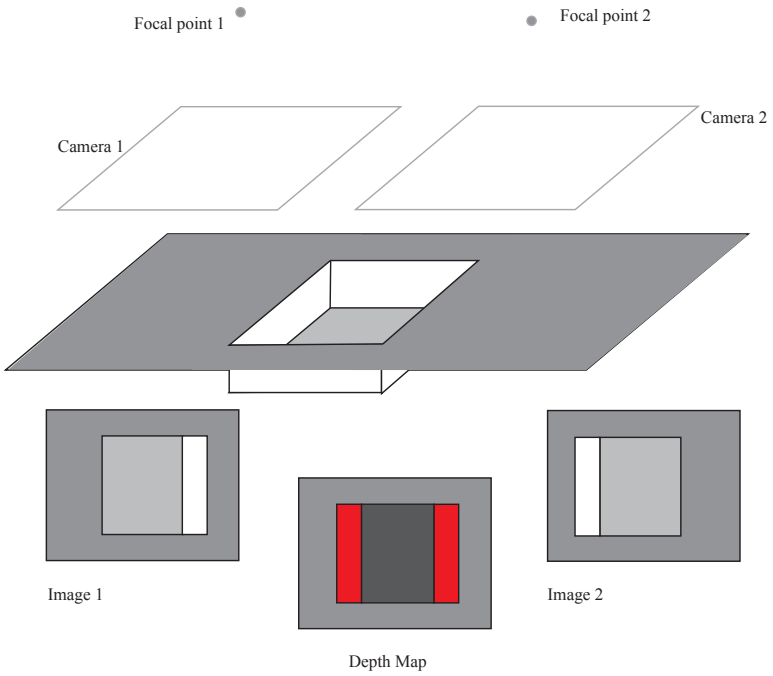


FIGURE 32.7: **Top** *shows two pinhole cameras viewing a rectangular depression in a flat surface. As the images show, camera on the left can see the right wall, and that on the right can see the left wall. This means that these walls cannot be reconstructed directly using trigonometry, and so the depth 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**.

use small windows, matches are ambiguous and so you end up with errors. When you use big windows, disparity maps tend to be oversmoothed. Fixing these effects requires more complex matching procedures (Section 33.1).

Procedure: 32.2 *Improved stereo matching*

For each point $\mathbf{x}_1$ in image 1, find the best match $\mathbf{b}_2(\mathbf{x}_1)$ in image 2 using procedure 32.1. Now for each point $\mathbf{x}_2$ in 2, find the best match $\mathbf{b}_1(\mathbf{x}_2)$ in 1. Now find pairs $(\mathbf{x}_1, \mathbf{x}_2)$ where $\mathbf{b}_2(\mathbf{x}_1) = \mathbf{x}_2$ *and* $\mathbf{b}_1(\mathbf{x}_2) = \mathbf{x}_1$. These are matches.

32.1.5 Coarse-to-Fine Matching

If left and right images are large, you may need to search a large range of locations in the right image to find the best match to a left image point. You can control

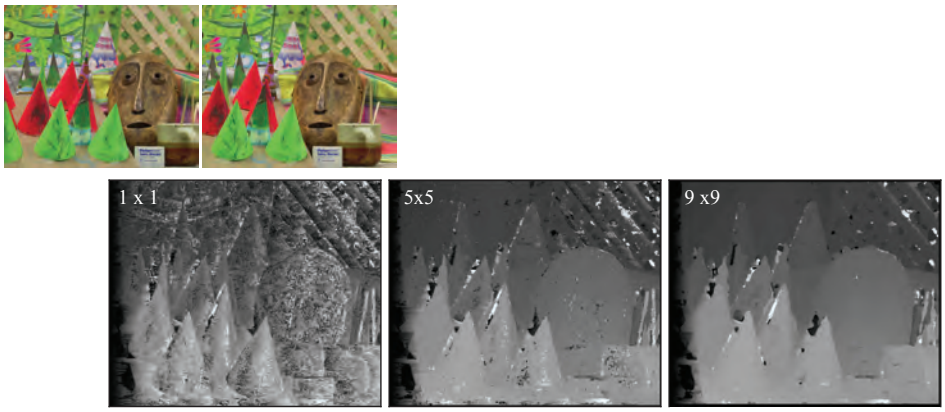


FIGURE 32.8: *Disparity maps for a stereo pair of cones, from the Middlebury datasets, for a simple window matching procedure, using window sizes from* 1×1 *to* 9×9*; details in text.* Image credit: *Stereo pair from the Middlebury dataset at* `https://vision.middlebury.edu/stereo/data/`*, disparities are mine.*

this issue with the coarse-to-fine search of Section 15.1.3.

The photometric consistency constraint means the disparity computed for a smoothed and downsampled stereo image pair should be a fair guide to the disparity for the original pair. The smoothing window on (say) the left image may cover some points not visible in the right image (Figure 32.9). This means you need to be careful turning coarse scale matches into finer scale matches. You can manage this issue by searching outside the neighborhood you might expect. For example, assume you smooth and resample by half to make a coarser layer of your pyramid. In principle, you could search only four locations when you refine to the fine scale as in Section 15.1.3; instead, you might search a larger window. I have sketched the procedure below.

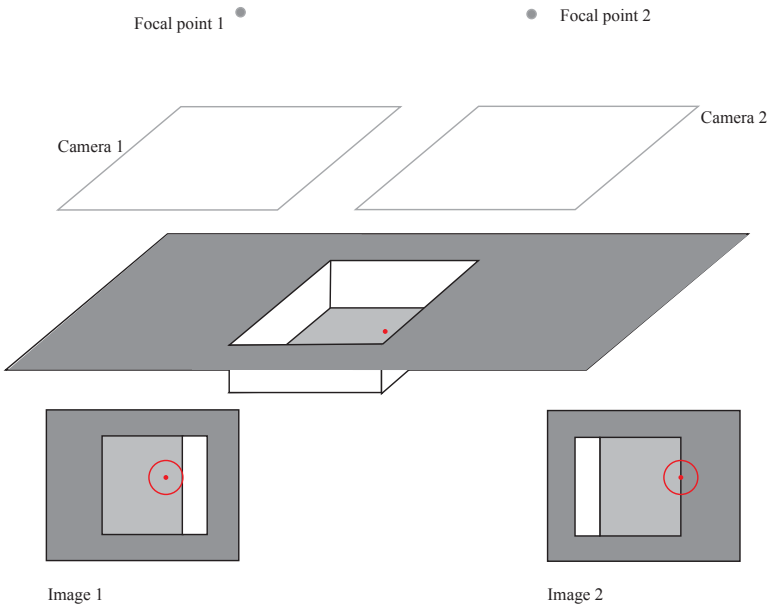


FIGURE 32.9: *Coarse-to-fine matching requires care. The* **red** *point on the surface (***top***) appears as a point in the high resolution versions of left and right images (***bottom***). When you smooth those images by a weighted average over a large window (the* **red** *circle), the points that contribute to the smoothed result mostly correspond. But some points do not, because of the depth discontinuity. This means that the corresponding points in left and right low resolution versions may not have exactly the same intensity.*

Procedure: 32.3 *Coarse to fine matching, in outline*

You have a left and right image in a rectified pair. You wish to match points in the left image to points in the right image. Build a Gaussian pyramid for left and right images, as in Procedure 3.6. Write d for the downsampling factor and N for the number of layers. A point $\mathbf{x}_l^{(1)}$ in the original left image – that is, the finest layer – appears at $\mathbf{x}_l^{(2)} = \lfloor \mathbf{x}_l / d \rfloor$ in the next coarser layer, and at $\mathbf{x}_l^{(j)} = \lfloor \mathbf{x}_l^{(j-1)} / d \rfloor$ in the j'th layer ($j > 1$).
Choose a neighborhood size s for searching. You want $s > d$. Now obtain a match for $\mathbf{x}_l^{(1)}$ by:

- Form $\mathbf{x}_l^{(N)}$ and find a best matching point in the right image at $\mathbf{b}_r^{(N)}$.
- Now iterate from $j = N$ to $j = 2$:
  - Given $\mathbf{b}_r^{(j)}$, you know $\mathbf{x}_l^{(j-1)}$ (from above). Find $\mathbf{b}_r^{(j-1)}$ by searching in a $s \times s$ neighborhood around $d\mathbf{b}_r^{(j)}$.

You can use this outline for right to left matching as well. You can also use it for the improved matching (just substitute for the matching in Procedure 32.2).

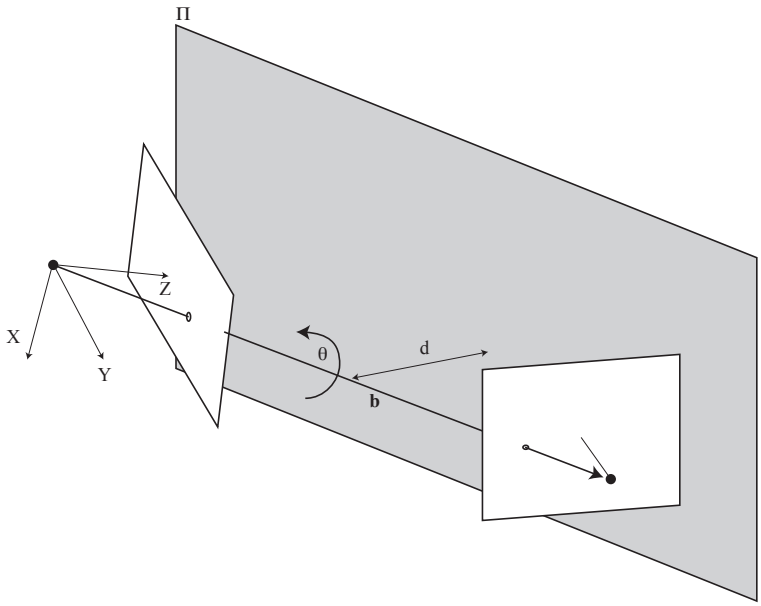


FIGURE 32.10: *There is a two parameter family of planes that are parallel to the baseline between two cameras. One parameter, d, is the distance from baseline to the plane; the other, θ chooses how the plane is rotated around the baseline.*

32.2 RECTIFICATION: GETTING TO THE EASY GEOMETRY

You are given a pair of images, obtained with cameras whose calibration is known. You know the transformation from first to second camera coordinate system, either because the cameras are on a known rig, or because you used odometry. Odometry will yield the translation between focal points only up to scale. For the moment, ignore this issue. *Rectification* is the process of constructing a pair of images that are equivalent and that appear to have been obtained in an epipolar configuration.

To rectify, you must construct homographies that map each image onto a virtual image plane. To rectify into an epipolar configuration, the virtual image plane must be parallel to **b exercises**. There is a two parameter family of such planes **exercises**. One parameter – d – chooses the perpendicular distance from the baseline to the virtual image plane **exercises**. The other parameter – θ – chooses the rotation of the plane around the baseline **exercises**. Figure 32.10 shows the geometric setup.

Figure 32.11 illustrates some notation. Because the intrinsics are known, you can map from camera coordinates to world coordinates. Choose a world coordinate system so that the first camera is a standard camera. The second camera is $\left[\mathcal{R}^T \mid -\mathcal{R}^T\mathbf{b}\right]$ (check Section 29.2.2 if you're uncertain). The vector $\mathbf{b}$ is the translation from left focal point to right focal point. Write $\mathbf{t}$ for a unit vector parallel to

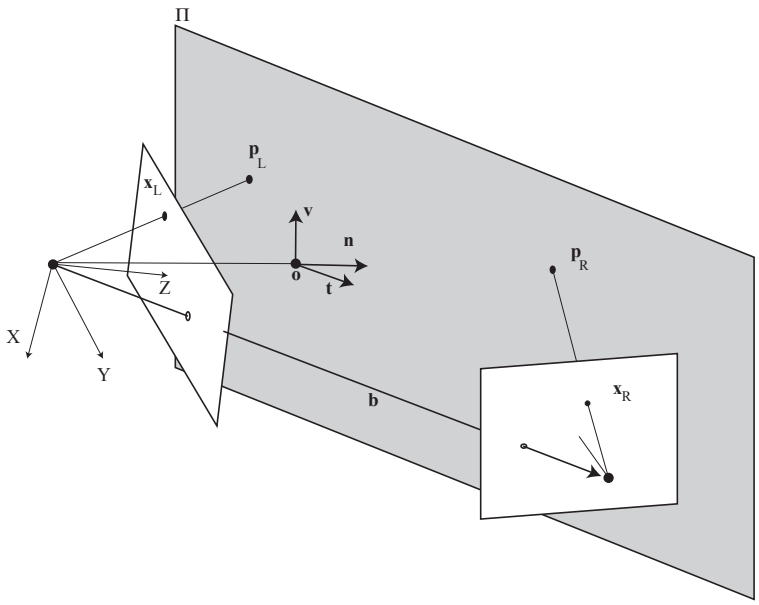


FIGURE 32.11: *The two cameras and virtual image plane of Figure 32.10, together with notation for rectification.*

$\mathbf{b}$; $\mathbf{n}(\theta)$ for the unit vector normal to the plane; and $\mathbf{v}(\theta)$ for $\mathbf{t} \times \mathbf{n}(\theta)$. Use $\mathbf{t}$ and $\mathbf{v}$ as coordinate directions on the virtual image plane.

Now construct the point $\mathbf{o} = d\mathbf{n}$ which lies on the virtual image plane **exercises**. Use this as the origin of a coordinate system on the virtual image plane. The coordinate directions are given by $\mathbf{v}$ and $\mathbf{t}$.

Because both cameras are calibrated, it is enough to ignore intrinsics and assume that each is a standard camera **exercises**. You can show that the homography from the first camera to the virtual image plane is

$$\begin{bmatrix} d\mathbf{t}^T \\ d\mathbf{v}^T(\theta) \\ \mathbf{n}(\theta)^T \end{bmatrix}.$$

exercises. Working in camera two's coordinate system, you can also show that the homography from the second camera to the virtual image plane is

$$\begin{bmatrix} d\mathbf{t}^T\mathcal{R} \\ d\mathbf{v}^T(\theta)\mathcal{R} \\ \mathbf{n}(\theta)^T\mathcal{R} \end{bmatrix}.$$

exercises. Notice that all d does is scale the virtual images, so it isn't particularly interesting. You can parametrize $\mathbf{n}(\theta)$ by choosing some plane to be the $\theta = 0$ plane. For this plane, you have $\mathbf{n}_0$ and $\mathbf{v}_0 = \mathbf{t} \times \mathbf{n}_0$. Then $\mathbf{n}(\theta) = \cos\theta\mathbf{n}_0 + \sin\theta\mathbf{v}_0$. One useful criterion for choosing the best θ is to require that the homographies be "as

affine as possible". You can achieve this by ensuring that the bottom row of each homography looks most like $\mathbf{e}_3^T = (0, 0, 1)$. This gives the optimization criterion

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \left[\cos\theta(\mathbf{n}_0^T\mathbf{e}_3 + \mathbf{n}_0^T\mathcal{R}\mathbf{e}_3) + \sin\theta(\mathbf{v}_0^T\mathbf{e}_3 + \mathbf{v}_0^T\mathcal{R}\mathbf{e}_3)\right]^2.$$

exercises. There are two solutions **exercises**, but one places the virtual image plane behind the cameras.

Remember this: *You can rectify a calibrated stereo pair by choosing a virtual image plane parallel to the baseline. This produces a pair of homographies. It is usual to choose the homographies so that they are "as affine as possible", using a simple optimization problem.*

32.3 YOU SHOULD

32.3.1 remember these terms:

Baseline 591
Disparity 592
Photometric consistency 594

32.3.2 remember these facts:

A variety of engineering considerations apply to disparity 592
There are usually points that can be seen in one image but not the other 596
You can rectify a calibrated stereo pair with a straightforward geometric construction 602

32.3.3 remember these procedures:

The simplest stereo matching 595
Improved stereo matching 597
Coarse to fine matching, in outline 600

32.3.4 be able to:

EXERCISES

QUICK CHECKS

32.1. Section 32.1.2 has: "the tendency to make errors at depth (or disparity) discontinuities.. is because it may not be possible to match a window correctly at these points **exercises**." Explain.

32.2. Section 32.1.2 has: "Small windows match too easily, meaning the disparity field is noisy. Big windows tend to oversmooth the disparity **exercises**." Explain.

32.3. Section 32.1.2 has: "To rectify into an epipolar configuration, the virtual image plane must be parallel to **b**." Use epipolar geometry to explain why.

32.4. Section 32.1.2 has: "There is a two parameter family of such planes."

32.5. Section 32.1.2 has: " One parameter – d – chooses the perpendicular distance from the baseline to the virtual image plane." Explain.

32.6. Section 32.1.2 has: "The other parameter – θ – chooses the rotation of the plane around the baseline."

32.7. Section 32.1.2 has: "Because both cameras are calibrated, it is enough to assume that each is a standard camera."

LONGER PROBLEMS

32.8. This problem constructs the homographies that rectify a stereo pair onto a virtual image plane

(a) Show that, in the notation of Section 32.1.2, the point $\mathbf{o} = d\mathbf{n}$ lies on the virtual image plane.

(b) Show that, in the notation of Section 32.1.2, the homography from the first camera to the virtual image plane is

$$\begin{bmatrix} d\mathbf{t}^T \\ d\mathbf{v}^T(\theta) \\ \mathbf{n}(\theta)^T \end{bmatrix}.$$

(c) Construct a coordinate system in the virtual image plane for the second camera. You need only move the origin.

(d) Working in this coordinate system and camera two's coordinate system, show that the homography from the second camera to the virtual image plane is

$$\begin{bmatrix} d\mathbf{t}^T\mathcal{R} \\ d\mathbf{v}^T(\theta)\mathcal{R} \\ \mathbf{n}(\theta)^T\mathcal{R} \end{bmatrix}.$$

(e) Section 32.2 uses the optimization criterion

$$\hat{\theta} = \underset{\theta}{\text{argmin}} \left[\cos\theta(\mathbf{n}_0^T\mathbf{e}_3 + \mathbf{n}_0^T\mathcal{R}\mathbf{e}_3) + \sin\theta(\mathbf{v}_0^T\mathbf{e}_3 + \mathbf{v}_0^T\mathcal{R}\mathbf{e}_3)\right]^2.$$

Why does this correspond to requiring the homographies are "as affine as possible"?

(f) Section 32.2 states: "There are two solutions, but one places the virtual image plane behind the cameras." Explain.

PROGRAMMING EXERCISES

32.9. Write code to reproduce Figure 32.8. In this figure, disparity for the left image is computed by matching windows to the right image. Use your codebase to answer the following questions.

- **(a)** When you compute the photometric loss, do you get better results using color or intensity?
- **(b)** When you compute the photometric loss, do you get better results using RGB color or LAB color?
- **(c)** When you compute the photometric loss, is there any advantage to weighting the window?
- **(d)** Do you get improvements by using Procedure 32.2?