



(a)



(b)



(c)



(d)

**Figure 4.1** A variety of feature detectors and descriptors can be used to analyze, describe and match images: (a) point-like interest operators (Brown, Szeliski, and Winder 2005) © 2005 IEEE; (b) region-like interest operators (Matas, Chum, Urban *et al.* 2004) © 2004 Elsevier; (c) edges (Elder and Goldberg 2001) © 2001 IEEE; (d) straight lines (Sinha, Steedly, Szeliski *et al.* 2008) © 2008 ACM.

Feature detection and matching are an essential component of many computer vision applications. Consider the two pairs of images shown in Figure 4.2. For the first pair, we may wish to *align* the two images so that they can be seamlessly stitched into a composite mosaic (Chapter 9). For the second pair, we may wish to establish a dense set of *correspondences* so that a 3D model can be constructed or an in-between view can be generated (Chapter 11). In either case, what kinds of *features* should you detect and then match in order to establish such an alignment or set of correspondences? Think about this for a few moments before reading on.

The first kind of feature that you may notice are specific locations in the images, such as mountain peaks, building corners, doorways, or interestingly shaped patches of snow. These kinds of localized feature are often called *keypoint features* or *interest points* (or even *corners*) and are often described by the appearance of patches of pixels surrounding the point location (Section 4.1). Another class of important features are *edges*, e.g., the profile of mountains against the sky, (Section 4.2). These kinds of features can be matched based on their orientation and local appearance (edge profiles) and can also be good indicators of object boundaries and *occlusion* events in image sequences. Edges can be grouped into longer *curves* and *straight line segments*, which can be directly matched or analyzed to find *vanishing points* and hence internal and external camera parameters (Section 4.3).

In this chapter, we describe some practical approaches to detecting such features and also discuss how feature correspondences can be established across different images. Point features are now used in such a wide variety of applications that it is good practice to read and implement some of the algorithms from (Section 4.1). Edges and lines provide information that is complementary to both keypoint and region-based descriptors and are well-suited to describing object boundaries and man-made objects. These alternative descriptors, while extremely useful, can be skipped in a short introductory course.

## 4.1 Points and patches

Point features can be used to find a sparse set of corresponding locations in different images, often as a pre-cursor to computing camera pose (Chapter 7), which is a prerequisite for computing a denser set of correspondences using stereo matching (Chapter 11). Such correspondences can also be used to align different images, e.g., when stitching image mosaics or performing video stabilization (Chapter 9). They are also used extensively to perform object instance and category recognition (Sections 14.3 and 14.4). A key advantage of keypoints is that they permit matching even in the presence of clutter (occlusion) and large scale and orientation changes.

Feature-based correspondence techniques have been used since the early days of stereo

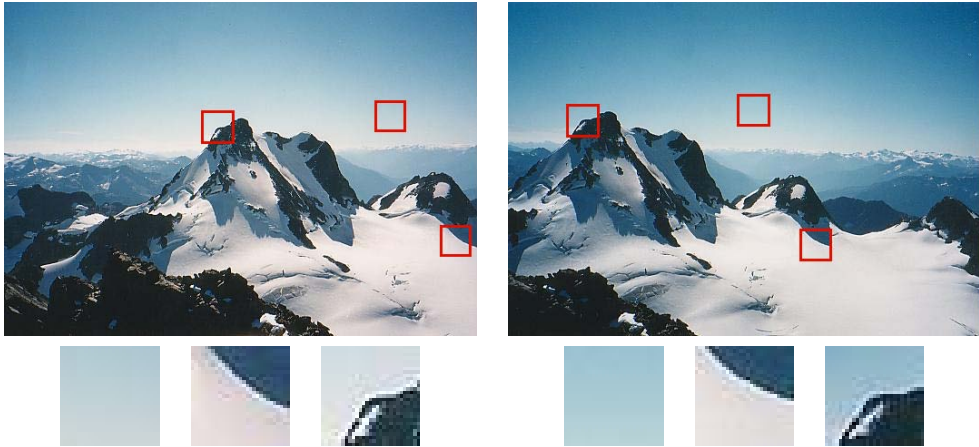


**Figure 4.2** Two pairs of images to be matched. What kinds of feature might one use to establish a set of *correspondences* between these images?

matching (Hannah 1974; Moravec 1983; Hannah 1988) and have more recently gained popularity for image-stitching applications (Zoghلامي, Faugeras, and Deriche 1997; Brown and Lowe 2007) as well as fully automated 3D modeling (Beardsley, Torr, and Zisserman 1996; Schaffalitzky and Zisserman 2002; Brown and Lowe 2003; Snavely, Seitz, and Szeliski 2006).

There are two main approaches to finding feature points and their correspondences. The first is to find features in one image that can be accurately *tracked* using a local search technique, such as correlation or least squares (Section 4.1.4). The second is to independently detect features in all the images under consideration and then *match* features based on their local appearance (Section 4.1.3). The former approach is more suitable when images are taken from nearby viewpoints or in rapid succession (e.g., video sequences), while the latter is more suitable when a large amount of motion or appearance change is expected, e.g., in stitching together panoramas (Brown and Lowe 2007), establishing correspondences in *wide baseline stereo* (Schaffalitzky and Zisserman 2002), or performing object recognition (Fergus, Perona, and Zisserman 2007).

In this section, we split the keypoint detection and matching pipeline into four separate stages. During the *feature detection* (extraction) stage (Section 4.1.1), each image is searched for locations that are likely to match well in other images. At the *feature description* stage (Section 4.1.2), each region around detected keypoint locations is converted into a more compact and stable (invariant) *descriptor* that can be matched against other descriptors. The



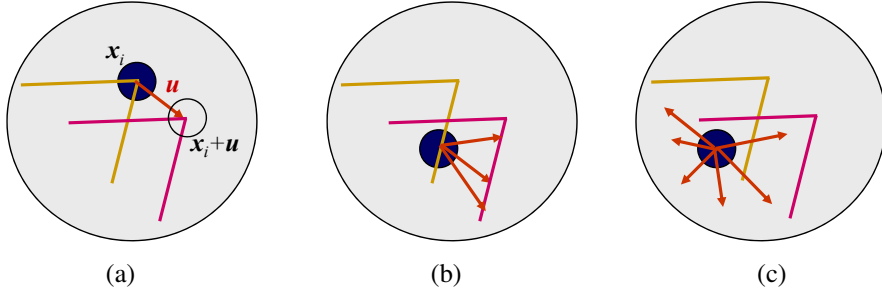
**Figure 4.3** Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.

*feature matching* stage (Section 4.1.3) efficiently searches for likely matching candidates in other images. The *feature tracking* stage (Section 4.1.4) is an alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

A wonderful example of all of these stages can be found in David Lowe’s (2004) paper, which describes the development and refinement of his *Scale Invariant Feature Transform* (SIFT). Comprehensive descriptions of alternative techniques can be found in a series of survey and evaluation papers covering both feature detection (Schmid, Mohr, and Bauckhage 2000; Mikolajczyk, Tuytelaars, Schmid *et al.* 2005; Tuytelaars and Mikolajczyk 2007) and feature descriptors (Mikolajczyk and Schmid 2005). Shi and Tomasi (1994) and Triggs (2004) also provide nice reviews of feature detection techniques.

### 4.1.1 Feature detectors

How can we find image locations where we can reliably find correspondences with other images, i.e., what are good features to track (Shi and Tomasi 1994; Triggs 2004)? Look again at the image pair shown in Figure 4.3 and at the three sample *patches* to see how well they might be matched or tracked. As you may notice, textureless patches are nearly impossible to localize. Patches with large contrast changes (gradients) are easier to localize, although straight line segments at a single orientation suffer from the *aperture problem* (Horn and Schunck 1981; Lucas and Kanade 1981; Anandan 1989), i.e., it is only possible to align the patches along the direction *normal* to the edge direction (Figure 4.4b). Patches with



**Figure 4.4** Aperture problems for different image patches: (a) stable (“corner-like”) flow; (b) classic aperture problem (barber-pole illusion); (c) textureless region. The two images  $I_0$  (yellow) and  $I_1$  (red) are overlaid. The red vector  $\mathbf{u}$  indicates the displacement between the patch centers and the  $w(\mathbf{x}_i)$  weighting function (patch window) is shown as a dark circle.

gradients in at least two (significantly) different orientations are the easiest to localize, as shown schematically in Figure 4.4a.

These intuitions can be formalized by looking at the simplest possible matching criterion for comparing two image patches, i.e., their (weighted) summed square difference,

$$E_{\text{WSSD}}(\mathbf{u}) = \sum_i w(\mathbf{x}_i) [I_1(\mathbf{x}_i + \mathbf{u}) - I_0(\mathbf{x}_i)]^2, \quad (4.1)$$

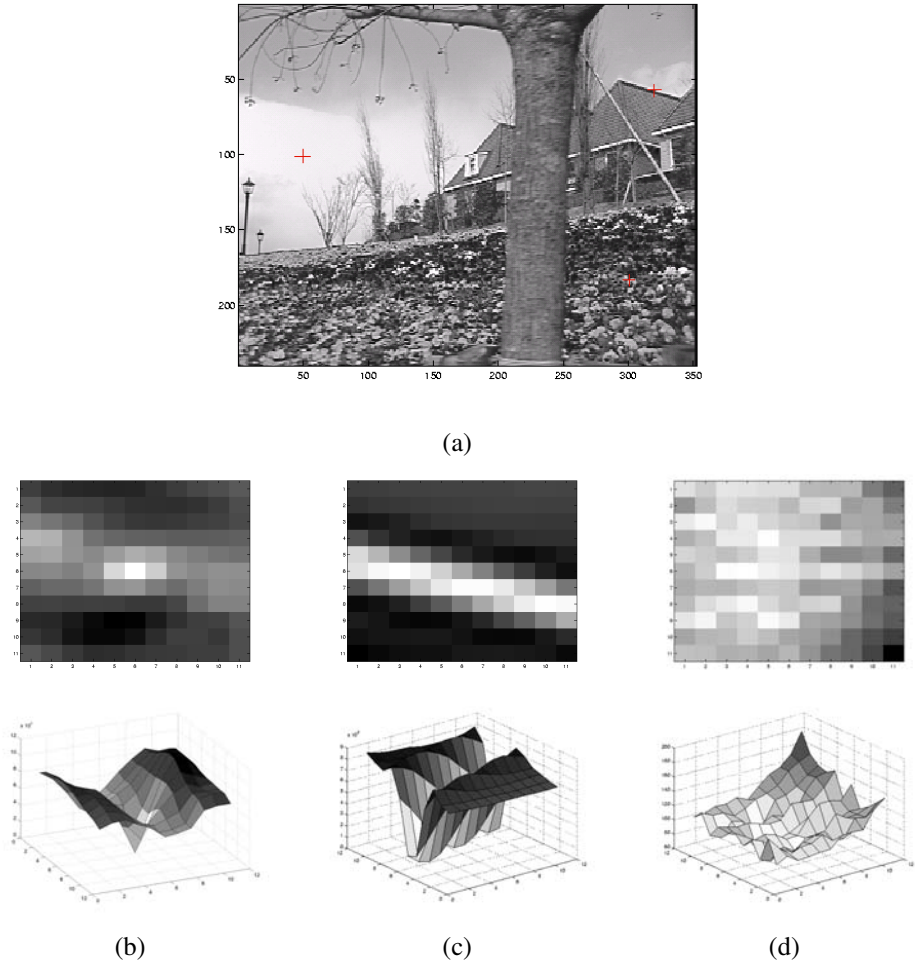
where  $I_0$  and  $I_1$  are the two images being compared,  $\mathbf{u} = (u, v)$  is the *displacement* vector,  $w(\mathbf{x})$  is a spatially varying weighting (or window) function, and the summation  $i$  is over all the pixels in the patch. Note that this is the same formulation we later use to estimate motion between complete images (Section 8.1).

When performing feature detection, we do not know which other image locations the feature will end up being matched against. Therefore, we can only compute how stable this metric is with respect to small variations in position  $\Delta\mathbf{u}$  by comparing an image patch against itself, which is known as an *auto-correlation function* or *surface*

$$E_{\text{AC}}(\Delta\mathbf{u}) = \sum_i w(\mathbf{x}_i) [I_0(\mathbf{x}_i + \Delta\mathbf{u}) - I_0(\mathbf{x}_i)]^2 \quad (4.2)$$

(Figure 4.5).<sup>1</sup> Note how the auto-correlation surface for the textured flower bed (Figure 4.5b and the red cross in the lower right quadrant of Figure 4.5a) exhibits a strong minimum, indicating that it can be well localized. The correlation surface corresponding to the roof edge (Figure 4.5c) has a strong ambiguity along one direction, while the correlation surface corresponding to the cloud region (Figure 4.5d) has no stable minimum.

<sup>1</sup> Strictly speaking, a correlation is the *product* of two patches (3.12); I’m using the term here in a more qualitative sense. The weighted sum of squared differences is often called an *SSD surface* (Section 8.1).



**Figure 4.5** Three auto-correlation surfaces  $E_{AC}(\Delta u)$  shown as both grayscale images and surface plots: (a) The original image is marked with three red crosses to denote where the auto-correlation surfaces were computed; (b) this patch is from the flower bed (good unique minimum); (c) this patch is from the roof edge (one-dimensional aperture problem); and (d) this patch is from the cloud (no good peak). Each grid point in figures b–d is one value of  $\Delta u$ .



Using a Taylor Series expansion of the image function  $I_0(\mathbf{x}_i + \Delta \mathbf{u}) \approx I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u}$  (Lucas and Kanade 1981; Shi and Tomasi 1994), we can approximate the auto-correlation surface as

$$E_{AC}(\Delta \mathbf{u}) = \sum_i w(\mathbf{x}_i) [I_0(\mathbf{x}_i + \Delta \mathbf{u}) - I_0(\mathbf{x}_i)]^2 \quad (4.3)$$

$$\approx \sum_i w(\mathbf{x}_i) [I_0(\mathbf{x}_i) + \nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u} - I_0(\mathbf{x}_i)]^2 \quad (4.4)$$

$$= \sum_i w(\mathbf{x}_i) [\nabla I_0(\mathbf{x}_i) \cdot \Delta \mathbf{u}]^2 \quad (4.5)$$

$$= \Delta \mathbf{u}^T \mathbf{A} \Delta \mathbf{u}, \quad (4.6)$$

where

$$\nabla I_0(\mathbf{x}_i) = \left( \frac{\partial I_0}{\partial x}, \frac{\partial I_0}{\partial y} \right)(\mathbf{x}_i) \quad (4.7)$$

is the *image gradient* at  $\mathbf{x}_i$ . This gradient can be computed using a variety of techniques (Schmid, Mohr, and Bauckhage 2000). The classic ‘‘Harris’’ detector (Harris and Stephens 1988) uses a  $\begin{bmatrix} -2 & -1 & 0 & 1 & 2 \end{bmatrix}$  filter, but more modern variants (Schmid, Mohr, and Bauckhage 2000; Triggs 2004) convolve the image with horizontal and vertical derivatives of a Gaussian (typically with  $\sigma = 1$ ).

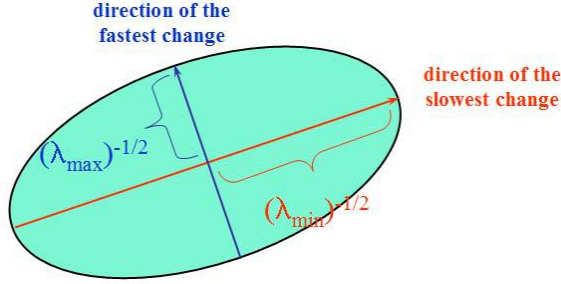
The auto-correlation matrix  $\mathbf{A}$  can be written as

$$\mathbf{A} = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \quad (4.8)$$

where we have replaced the weighted summations with discrete convolutions with the weighting kernel  $w$ . This matrix can be interpreted as a tensor (multiband) image, where the outer products of the gradients  $\nabla I$  are convolved with a weighting function  $w$  to provide a per-pixel estimate of the local (quadratic) shape of the auto-correlation function.

As first shown by Anandan (1984; 1989) and further discussed in Section 8.1.3 and (8.44), the inverse of the matrix  $\mathbf{A}$  provides a lower bound on the uncertainty in the location of a matching patch. It is therefore a useful indicator of which patches can be reliably matched. The easiest way to visualize and reason about this uncertainty is to perform an eigenvalue analysis of the auto-correlation matrix  $\mathbf{A}$ , which produces two eigenvalues ( $\lambda_0, \lambda_1$ ) and two eigenvector directions (Figure 4.6). Since the larger uncertainty depends on the smaller eigenvalue, i.e.,  $\lambda_0^{-1/2}$ , it makes sense to find maxima in the smaller eigenvalue to locate good features to track (Shi and Tomasi 1994).

**Förstner–Harris.** While Anandan and Lucas and Kanade (1981) were the first to analyze the uncertainty structure of the auto-correlation matrix, they did so in the context of associating certainties with optic flow measurements. Förstner (1986) and Harris and Stephens



**Figure 4.6** Uncertainty ellipse corresponding to an eigenvalue analysis of the auto-correlation matrix  $\mathbf{A}$ .

(1988) were the first to propose using local maxima in rotationally invariant scalar measures derived from the auto-correlation matrix to locate keypoints for the purpose of sparse feature matching. (Schmid, Mohr, and Bauckhage (2000); Triggs (2004) give more detailed historical reviews of feature detection algorithms.) Both of these techniques also proposed using a Gaussian weighting window instead of the previously used square patches, which makes the detector response insensitive to in-plane image rotations.

The minimum eigenvalue  $\lambda_0$  (Shi and Tomasi 1994) is not the only quantity that can be used to find keypoints. A simpler quantity, proposed by Harris and Stephens (1988), is

$$\det(\mathbf{A}) - \alpha \operatorname{trace}(\mathbf{A})^2 = \lambda_0 \lambda_1 - \alpha (\lambda_0 + \lambda_1)^2 \quad (4.9)$$

with  $\alpha = 0.06$ . Unlike eigenvalue analysis, this quantity does not require the use of square roots and yet is still rotationally invariant and also downweights edge-like features where  $\lambda_1 \gg \lambda_0$ . Triggs (2004) suggests using the quantity

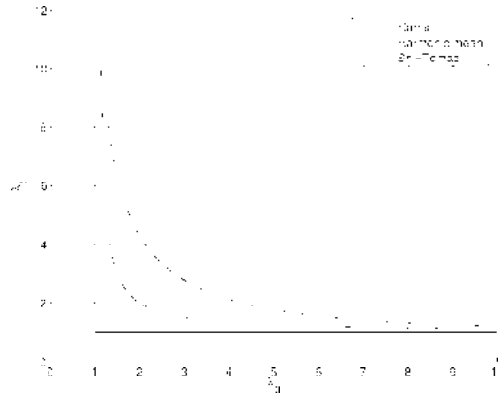
$$\lambda_0 - \alpha \lambda_1 \quad (4.10)$$

(say, with  $\alpha = 0.05$ ), which also reduces the response at 1D edges, where aliasing errors sometimes inflate the smaller eigenvalue. He also shows how the basic  $2 \times 2$  Hessian can be extended to parametric motions to detect points that are also accurately localizable in scale and rotation. Brown, Szeliski, and Winder (2005), on the other hand, use the harmonic mean,

$$\frac{\det \mathbf{A}}{\operatorname{tr} \mathbf{A}} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}, \quad (4.11)$$

which is a smoother function in the region where  $\lambda_0 \approx \lambda_1$ . Figure 4.7 shows isocontours of the various interest point operators, from which we can see how the two eigenvalues are blended to determine the final interest value.

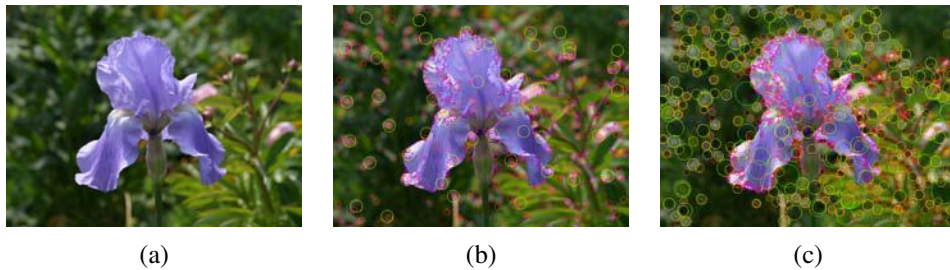




**Figure 4.7** Isocontours of popular keypoint detection functions (Brown, Szeliski, and Winder 2004). Each detector looks for points where the eigenvalues  $\lambda_0, \lambda_1$  of  $\mathbf{A} = w * \nabla I \nabla I^T$  are both large.

1. Compute the horizontal and vertical derivatives of the image  $I_x$  and  $I_y$  by convolving the original image with derivatives of Gaussians (Section 3.2.3).
2. Compute the three images corresponding to the outer products of these gradients. (The matrix  $\mathbf{A}$  is symmetric, so only three entries are needed.)
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas discussed above.
5. Find local maxima above a certain threshold and report them as detected feature point locations.

**Algorithm 4.1** Outline of a basic feature detection algorithm.



**Figure 4.8** Interest operator responses: (a) Sample image, (b) Harris response, and (c) DoG response. The circle sizes and colors indicate the scale at which each interest point was detected. Notice how the two detectors tend to respond at complementary locations.

The steps in the basic auto-correlation-based keypoint detector are summarized in Algorithm 4.1. Figure 4.8 shows the resulting interest operator responses for the classic Harris detector as well as the difference of Gaussian (DoG) detector discussed below.

**Adaptive non-maximal suppression (ANMS).** While most feature detectors simply look for local maxima in the interest function, this can lead to an uneven distribution of feature points across the image, e.g., points will be denser in regions of higher contrast. To mitigate this problem, Brown, Szeliski, and Winder (2005) only detect features that are both local maxima and whose response value is significantly (10%) greater than that of all of its neighbors within a radius  $r$  (Figure 4.9c–d). They devise an efficient way to associate suppression radii with all local maxima by first sorting them by their response strength and then creating a second list sorted by decreasing suppression radius (Brown, Szeliski, and Winder 2005). Figure 4.9 shows a qualitative comparison of selecting the top  $n$  features and using ANMS.

**Measuring repeatability.** Given the large number of feature detectors that have been developed in computer vision, how can we decide which ones to use? Schmid, Mohr, and Bauckhage (2000) were the first to propose measuring the *repeatability* of feature detectors, which they define as the frequency with which keypoints detected in one image are found within  $\epsilon$  (say,  $\epsilon = 1.5$ ) pixels of the corresponding location in a transformed image. In their paper, they transform their planar images by applying rotations, scale changes, illumination changes, viewpoint changes, and adding noise. They also measure the *information content* available at each detected feature point, which they define as the entropy of a set of rotationally invariant local grayscale descriptors. Among the techniques they survey, they find that the improved (Gaussian derivative) version of the Harris operator with  $\sigma_d = 1$  (scale of the derivative Gaussian) and  $\sigma_i = 2$  (scale of the integration Gaussian) works best.