it is important to locate the edge position with *subpixel precision*. For instance, in commercial laser scanners (Chapter 2) an accuracy of about 0.25mm is a common target, and half a pixel may correspond to less than 0.5mm. The easiest way to achieve subpixel resolution is to locate the peak of a parabola interpolating three values in the output of CANNY_ENHANCER, namely at the edge pixel and at its two neighbours along the edge normal. Of course, any information available on edge profiles and noise should be exploited to improve on the parabola method.

4.3 Point Features: Corners

Although the mathematics of edge detection may seem involved, edges can be characterized intuitively in geometric terms: they are the projection of object boundaries, surface marks, and other interesting elements of a scene. We now give an example of image features that can be characterized more easily than edges in mathematical terms, but do not correspond necessarily to any geometric entities of the observed scene. These features can be interpreted as *corners*, but not only in the sense of intersections of image lines; they capture corner structures in patterns of intensities. Such features prove stable across sequences of images, and are therefore interesting to track objects across sequences (Chapter 8).

How do we detect corner features? Consider the spatial image gradient, $[E_x, E_y]^\top$ (the subscripts indicate partial differentiation, e.g., $E_x = \frac{\partial E}{\partial x}$). Consider a generic image point, p, a neighbourhood Q of p, and a matrix, C, defined as

Q is a parameter!

$$C = \begin{bmatrix} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{bmatrix}, \tag{4.9}$$

where the sums are taken over the neighbourhood Q. This matrix characterizes the *structure* of the grey levels. How?

The key to the answer is in the eigenvalues of C and their geometric interpretation. Notice that C is symmetric, and can therefore be diagonalized by a rotation of the coordinate axes; thus, with no loss of generality, we can think of C as a diagonal matrix:

$$C = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

The two eigenvalues, λ_1 and λ_2, are both nonnegative (why?); let us assume $\lambda_1 \geq \lambda_2$. The geometric interpretation of λ_1 and λ_2 can be understood through a few particular cases. First, consider a perfectly uniform Q: the image gradient vanishes everywhere, C becomes the null matrix, and we have $\lambda_1 = \lambda_2 = 0$. Second, assume that Q contains an ideal black and white step edge: we have $\lambda_2 = 0$, $\lambda_1 > 0$, and the eigenvector associated with λ_1 is parallel to the image gradient. Note that C is rank deficient in both cases, with rank 0 and 1 respectively. Third, assume that Q contains the corner of a black square against a white background: as there are two principal directions in Q, we expect $\lambda_1 \geq \lambda_2 > 0$, and the larger the eigenvalues, the stronger (higher contrast) their corresponding image lines. At this point, you have caught on with the fact that *the eigenvectors encode edge directions, the eigenvalues edge strength.* A corner is identified

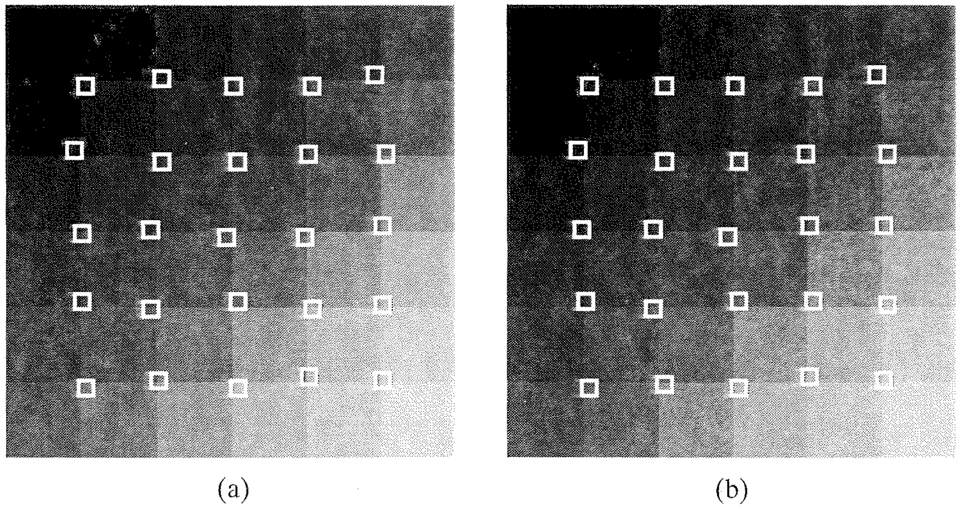


Figure 4.8 Corners found in a 8-bit, synthetic checkerboard image, corrupted by two realizations of synthetic Gaussian noise of standard deviation 2. The corner is the bottom right point of each 15×15 neighbourhood (highlighted).

by two strong edges; therefore, as $\lambda_1 \geq \lambda_2$, *a corner is a location where the smaller eigenvalue,* λ_2, *is large enough.*

Time for examples. Figure 4.8 shows the corners found in a synthetic image of a checkerboard, with and without additive noise. Figure 4.9 shows the corners found in the image of a building, and the histogram of the λ_2 values. The shape of this histogram is rather typical for most natural images. If the image contains uniform regions, or many almost ideal step edges, the histogram has a second peak at $\lambda_2 = 0$. The tail (right) of the histogram is formed by the points for which λ_2 is large, which are precisely the points (or, equivalently, the neighbourhoods) we are interested in. Figure 4.10 shows another example with a road scene.

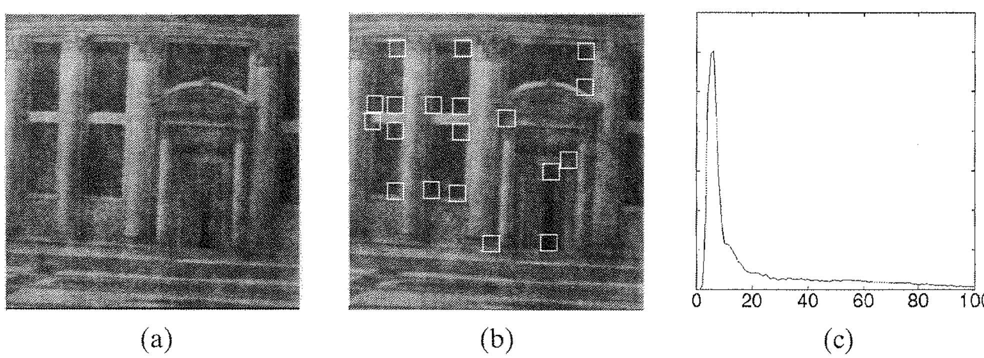
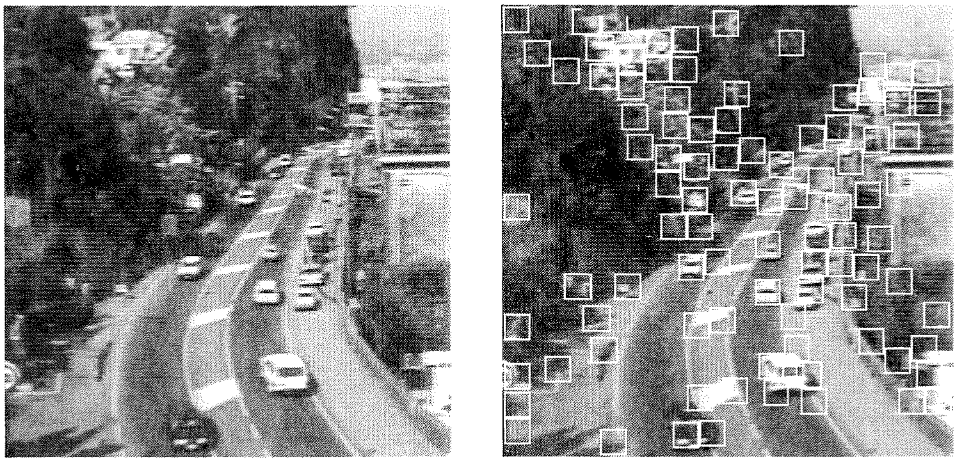


Figure 4.9 (a): original image of a building. (b): the 15×15 pixel neighbourhoods of some of the image points for which $\lambda_2 > 20$. (c): histogram of λ_2 values across the image.



(a) (b)

Figure 4.10 (a): image of an outdoor scene. The corner is the bottom right point of each 15×15 neighbourhood (highlighted). (b): corners found using a 15×15 neighbourhood.

We reiterate that our feature points include high-contrast image corners and T-junctions generated by the intersection of object contours (as the corners in Figure 4.8), but also corners of the local intensity pattern not corresponding to obvious scene features (as some of the corners in Figure 4.10). In general terms, at corners points *the intensity surface has two well-pronounced, distinctive directions, associated to eigenvalues of C both significantly larger than zero.*

We now summarize the procedure for locating this new type of image features.

Algorithm CORNERS

The input is formed by an image, I, and two parameters: the threshold on λ_2, τ, and the linear size of a square window (neighbourhood), say $2N + 1$ pixels.

1. Compute the image gradient over the entire image I;
2. For each image point p:
   (a) form the matrix C of (4.9) over a $(2N + 1) \times (2N + 1)$ neighbourhood Q of p;
   (b) compute λ_2, the smaller eigenvalue of C;
   (c) if $\lambda_2 > \tau$, save the coordinates of p into a list, L.
3. Sort L in decreasing order of λ_2.
4. Scanning the sorted list top to bottom: for each current point, p, delete all points appearing further on in the list which belong to the neighbourhood of p.

The output is a list of feature points for which $\lambda_2 > \tau$ and whose neighbourhoods do not overlap.

Algorithm CORNERS has two main parameters: the threshold, τ, and the size of the neighbourhood, $(2N + 1)$. The threshold, τ, can be estimated from the histogram of λ_2 (Exercise 4.6), as the latter has often an obvious valley near zero (Figure 4.9).

☞ Notice that such valley is not *always* present (Exercise 4.7).

Unfortunately, there is no simple criterion for the estimation of the optimal size of the neighbourhood. Experience indicates that choices of N between 2 and 10 are adequate in most practical cases.

☞ In the case of corner points, the value of N is linked to the location of the corner within the neighbourhood. As you can see from Figure 4.9, for relatively large values of N the corner tends to move away from the neighbourhood center (see Exercise 4.8 for a quantitative analysis of this effect).

4.4 Surface Extraction from Range Images

Many 3-D objects, especially man-made, can be conveniently described in terms of the shape and position of the surfaces they are made of. For instance, you can describe a cone as an object formed by two surface patches, one conical and one planar, the latter perpendicular to the axis of the former. Surface-based descriptions are used for object classification, pose estimation, and reverse engineering, and are ubiquitous in computer graphics.

As we have seen in Chapter 2, range images are basically a sampled version of the visible surfaces in the scene. Therefore, ignoring the distortions introduced by sensor imperfections, *the shape of the image surface*[9] *and the shape of the visible scene surfaces are the same*, and any geometric property holding for one holds for the other too. This section presents a well-known method to find patches of various shapes composing the visible surface of an object. The method, called *HK segmentation*, partitions a range image into regions of homogeneous shape, called *homogeneous surface patches*, or just *surface patches* for short.[10] The method is based on differential geometry; Appendix, section A.5 gives a short summary of the basic concepts necessary.

☞ The solution to several computer vision problems involving 3-D object models are simpler when using 3-D features than 2-D features, as image formation must be taken into account for the latter.

[9] That is, the image values regarded as a surface defined on the image plane.

[10] Notice that surface patches are the basic ingredients for building a surface-based CAD model of an object automatically.