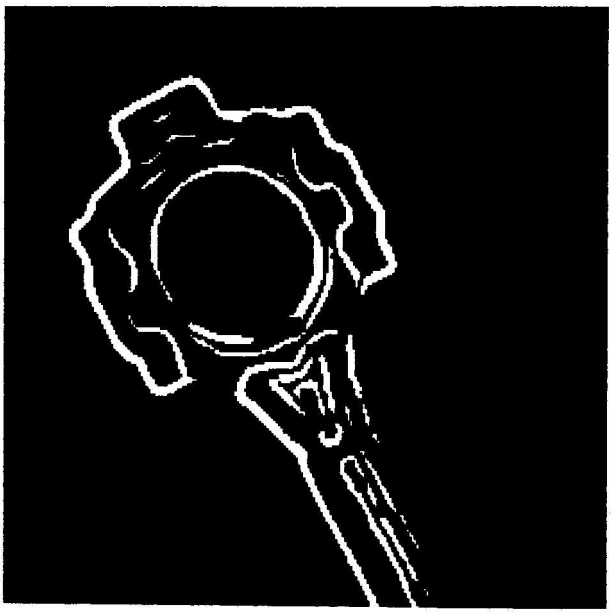

Figure 5.20: Edges obtained with facet model edge detector.

## 5.6 Gaussian Edge Detection

The essential idea in detecting step edges is to find points in the sampled image that have locally large gradient magnitudes. Much of the research work in step edge detection is devoted to finding numerical approximations to the gradient that are suitable for use with real images. The step edges in real images are not perfectly sharp since the edges are smoothed by the low-pass filtering inherent in the optics of the camera lens and the bandwidth limitations in the camera electronics. The images are also severely corrupted by noise from the camera and unwanted detail in the scene. An approximation to the image gradient must be able to satisfy two conflicting requirements: (1) the approximation must suppress the effects of noise, and (2) the approximation must locate the edge as accurately as possible. There is a trade-off between noise suppression and localization. An edge detection operator can reduce noise by smoothing the image, but this will add uncertainty to the location of the edge; or the operator can have greater sensitivity to the presence of edges, but this will increase the sensitivity of the operator to noise. The type of linear operator that provides the best compromise between noise immunity and localization, while retaining the advantages of Gaussian filtering, is the first derivative of a Gaussian. This operator corresponds to

smoothing an image with a Gaussian function and then computing the gradient. The gradient can be numerically approximated by using the standard finite-difference approximation for the first partial derivatives in the $x$ and $y$ directions listed in Section 5.1. The operator that is the combination of a Gaussian smoothing filter and a gradient approximation is not rotationally symmetric. The operator is symmetric along the edge and antisymmetric perpendicular to the edge (along the line of the gradient). This means that the operator is sensitive to the edge in the direction of steepest change, but is insensitive to the edge and acts as a smoothing operator in the direction along the edge.

### 5.6.1 Canny Edge Detector

The Canny edge detector is the first derivative of a Gaussian and closely approximates the operator that optimizes the product of signal-to-noise ratio and localization. The Canny edge detection algorithm is summarized by the following notation. Let $I[i,j]$ denote the image. The result from convolving the image with a Gaussian smoothing filter using separable filtering is an array of smoothed data,

$$S[i,j] = G[i,j;\sigma] \star I[i,j], \tag{5.43}$$

where $\sigma$ is the spread of the Gaussian and controls the degree of smoothing.

The gradient of the smoothed array $S[i,j]$ can be computed using the $2\times2$ first-difference approximations (Section 5.1) to produce two arrays $P[i,j]$ and $Q[i,j]$ for the $x$ and $y$ partial derivatives:

$$\begin{aligned} P[i,j] &\approx (S[i,j+1] - S[i,j] \\ &\quad + S[i+1,j+1] - S[i+1,j])/2 \end{aligned} \tag{5.44}$$

$$\begin{aligned} Q[i,j] &\approx (S[i,j] - S[i+1,j] \\ &\quad + S[i,j+1] - S[i+1,j+1])/2. \end{aligned} \tag{5.45}$$

The finite differences are averaged over the $2 \times 2$ square so that the $x$ and $y$ partial derivatives are computed at the same point in the image. The magnitude and orientation of the gradient can be computed from the standard formulas for rectangular-to-polar conversion:

$$M[i,j] = \sqrt{P[i,j]^2 + Q[i,j]^2} \tag{5.46}$$

$$\theta[i,j] = \arctan(Q[i,j], P[i,j]), \tag{5.47}$$

where the arctan function takes two arguments and generates an angle over the entire circle of possible directions. These functions must be computed efficiently, preferably without using floating-point arithmetic. It is possible to compute the gradient magnitude and orientation from the partial derivatives by table lookup. The arctangent can be computed using mostly fixed-point arithmetic[2] with a few essential floating-point calculations performed in software using integer and fixed-point arithmetic [59, chap. 11]. Sedgewick [218, p. 353] provides an algorithm for an integer approximation to the gradient angle that may be good enough for many applications.

### Nonmaxima Suppression

The magnitude image array $M[i,j]$ will have large values where the image gradient is large, but this is not sufficient to identify the edges, since the problem of finding locations in the image array where there is rapid change has merely been transformed into the problem of finding locations in the magnitude array $M[i,j]$ that are local maxima. To identify edges, the broad ridges in the magnitude array must be thinned so that only the magnitudes at the points of greatest local change remain. This process is called nonmaxima suppression, which in this case results in thinned edges.

Nonmaxima suppression thins the ridges of gradient magnitude in $M[i,j]$ by suppressing all values along the line of the gradient that are not peak values of a ridge. The algorithm begins by reducing the angle of the gradient $\theta[i,j]$ to one of the four sectors shown in Figure 5.21,

$$\zeta[i,j] = \text{Sector}(\theta[i,j]). \tag{5.48}$$

The algorithm passes a $3\times3$ neighborhood across the magnitude array $M[i,j]$. At each point, the center element $M[i,j]$ of the neighborhood is compared with its two neighbors along the line of the gradient given by the sector value $\zeta[i,j]$ at the center of the neighborhood. If the magnitude array value $M[i,j]$ at the center is not greater than both of the neighbor magnitudes along the gradient line, then $M[i,j]$ is set to zero. This process thins the broad ridges of gradient magnitude in $M[i,j]$ into ridges that are only one pixel wide. The

[2]In this context, fixed-point arithmetic is like integer arithmetic except that the number carries an implicit scale factor that assumes that the binary point is to the left of the number. Fixed-point arithmetic can be implemented using integer arithmetic on many machines.

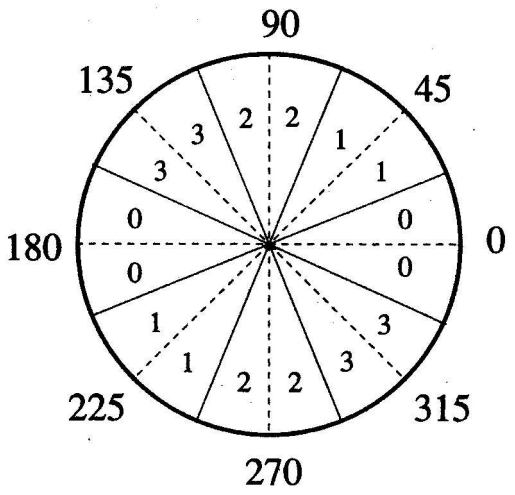


Figure 5.21: The partition of the possible gradient orientations into sectors for nonmaxima suppression is shown. There are four sectors, numbered 0 to 3, corresponding to the four possible combinations of elements in a $3 \times 3$ neighborhood that a line must pass through as it passes through the center of the neighborhood. The divisions of the circle of possible gradient line orientations are labeled in degrees.

values for the height of the ridge are retained in the nonmaxima-suppressed magnitude.

Let

$$N[i,j] = \text{nms}(M[i,j], \zeta[i,j]) \tag{5.49}$$

denote the process of nonmaxima suppression. The nonzero values in $N[i,j]$ correspond to the amount of contrast at a step change in the image intensity. In spite of the smoothing performed as the first step in edge detection, the nonmaxima-suppressed magnitude image $N[i,j]$ will contain many false edge fragments caused by noise and fine texture. The contrast of the false edge fragments is small.

### Thresholding

The typical procedure used to reduce the number of false edge fragments in the nonmaxima-suppressed gradient magnitude is to apply a threshold to $N[i,j]$. All values below the threshold are changed to zero. The result of applying a threshold to the nonmaxima-suppressed magnitude is an array

of the edges detected in the image $I[i, j]$. There will still be some false edges because the threshold $\tau$ was too low (false positives), and portions of actual contours may be missing (false negatives) due to softening of the edge contrast by shadows or because the threshold $\tau$ was too high. Selecting the proper threshold is difficult and involves some trial and error. A more effective thresholding scheme uses two thresholds.

The double thresholding algorithm takes the nonmaxima-suppressed image, $N[i, j]$, and applies two thresholds $\tau_1$ and $\tau_2$, with $\tau_2 \approx 2\tau_1$, to produce two thresholded edge images $T_1[i, j]$ and $T_2[i, j]$. Since image $T_2$ was formed with a higher threshold, it will contain fewer false edges; but $T_2$ may have gaps in the contours (too many false negatives). The double thresholding algorithm links the edges in $T_2$ into contours. When it reaches the end of a contour, the algorithm looks in $T_1$ at the locations of the 8-neighbors for edges that can be linked to the contour. The algorithm continues to gather edges from $T_1$ until the gap has been bridged to an edge in $T_2$. The algorithm performs edge linking as a by-product of thresholding and resolves some of the problems with choosing a threshold. The Canny edge detection algorithm is outlined in Algorithm 5.1.

The edge detection algorithm presented in this section has been run on several test images. Figure 5.22 shows the image of a connecting rod. Figures 5.23 and 5.24 present the results of applying the edge detection algorithm summarized in this section to the test image in Figure 5.22. In Figure 5.23, a $7 \times 7$ Gaussian filter was used to smooth the image before computing the gradient; in Figure 5.24, a $31 \times 31$ Gaussian filter was used. The nonmaxima-suppressed gradient magnitude for the smaller filter size exhibits excellent fine detail in the edges but suffers from excessive unwanted edge fragments

**Algorithm 5.1 Canny Edge Detection**

1. *Smooth the image with a Gaussian filter.*
2. *Compute the gradient magnitude and orientation using finite-difference approximations for the partial derivatives.*
3. *Apply nonmaxima suppression to the gradient magnitude.*
4. *Use the double thresholding algorithm to detect and link edges.*

Figure 5.22: A test image of a connecting rod. The image was acquired by a Reticon 256 × 256 area CCD array camera.

due to noise and fine texture. For the larger filter size, there are fewer unwanted edge fragments, but much of the detail in the edges has been lost. This illustrates the trade-off between edge localization and noise immunity.

## 5.7 Subpixel Location Estimation

In many applications, it is necessary to estimate the location of an edge to better than the spacing between pixels (subpixel resolution). The methods for obtaining subpixel resolution for gradient and second-order edge detection algorithms are very different and will be considered separately.

First, consider the output of a second-order edge detector such as the Laplacian of Gaussian. The edge is signaled by a zero crossing between pixels. In principle, the edge position can be computed to subpixel resolution by using linear interpolation. In practice, the output of second-order edge detection schemes, even with Gaussian presmoothing, is too noisy to allow any simple interpolation method to provide accurate results.

Obtaining subpixel resolution in edge location after edge detection with a gradient-based scheme is both practical and efficient. The result of applying a Gaussian smoothing filter and first derivative to an ideal step edge is a profile that is exactly the same shape as the Gaussian filter used for smoothing. If