

Early Processing

3

3.1 RECOVERING INTRINSIC STRUCTURE

The imaging process confounds much useful physical information into the gray-level array. In this respect, the imaging process is a collection of degenerate transformations. However, this information is not irrevocably lost, because there is much spatial redundancy: Neighboring pixels in the image have the same or nearly the same physical parameters. A collection of techniques, which we call *early processing*, exploits this redundancy in order to undo the degeneracies in the imaging process. These techniques have the character of transformations for changing the image into “parameter images” or *intrinsic images* [Barrow and Tenenbaum 1978; 1981] which reflect the spatial properties of the scene. Common intrinsic parameters are surface discontinuities, range, surface orientation, and velocity.

In this chapter we neglect high-level internal model information even though it is important and can affect early processing. Consider the case of the perceived central edge in Fig. 3.1a. As shown by Fig. 3.1b, which shows portions of the same image, the central edge of Fig. 3.1a is not present in the data. Nevertheless, the human perceiver “sees” the edge, and one reasonable explanation is that it is a product of an internal block model. Model-directed activity is taken up in later chapters. These examples show how high level models (e.g., circles) can affect low-level processors (e.g., edge finders). However, for the purposes of study it is often helpful to neglect these effects. These simplifications make it easier to derive the fundamental constraints between the physical parameters and gray levels. Once these are understood, they can be modified using the more abstract structures of later chapters.

Most early computer vision processing can be done with parallel computations whose inputs tend to be spatially localized. When computing intrinsic images

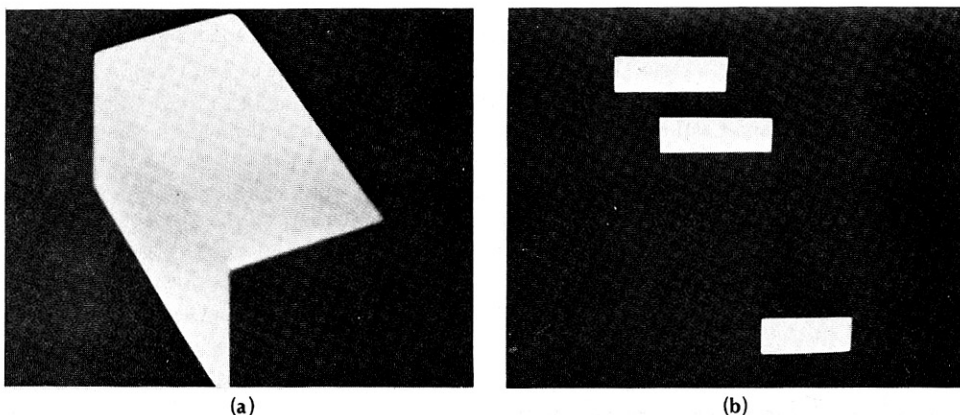


Fig. 3.1 (a) A perceived edge. (b) Portions of image in (a) showing the lack of image data.

the parallel computations are iterated until the intrinsic parameter measurements converge to a set of values. A computation that falls in the parallel-iterative category is known in computer vision as *relaxation* [Rosenfeld et al. 1976]. Relaxation is a very general computational technique that is useful in computer vision. Specific examples of relaxation computations appear throughout the book; general observations on relaxation appear in Chapter 12.

This chapter covers six categories of early processing techniques:

1. *Filtering* is a generic name for techniques of changing image gray levels to enhance the appearance of objects. Most often this means transformations that make the intensity discontinuities between regions more prominent. These transformations are often dependent on gross object characteristics. For example, if the objects of interest are expected to be relatively large, the image can be blurred to erase small intensity discontinuities while retaining those of the object's boundary. Conversely, if the objects are relatively small, a transformation that selectively removes large discontinuities may be appropriate. Filtering can also compensate for spatially varying illumination.
2. *Edge operators* detect and measure very local discontinuities in intensity or its gradient. The result of an edge operator is usually the magnitude and orientation of the discontinuity.
3. *Range transforms* use known geometry about stereo images to infer the distance of points from the viewer. These transforms make use of the inverse perspective transform to interpret how points in three-dimensional space project onto stereo pairs. A correspondence between points in two stereo images of known geometry determines the range of those points. Relative range may also be derived from local correspondences without knowing the imaging geometry precisely.
4. *Surface orientation* can be calculated if the source illumination and reflectance properties of the surface are known. This calculation is sometimes called

“shape from shading.” Surface orientation is particularly simple to calculate when the source illumination can be controlled.

5. *Optical flow*, or velocity fields of image points, can be calculated from local temporal and spatial variations in sequences of gray-level images.
6. A *pyramid* is a general structure for representing copies of the image at multiple resolutions. A pyramid is a “utility structure” which can dramatically improve the speed and effectiveness of many early processing and later segmentation algorithms.

3.2 FILTERING THE IMAGE

Filtering is a very general notion of transforming the image intensities in some way so as to enhance or deemphasize certain features. We consider only transforms that leave the image in its original format: a spatial array of gray levels. Spurred on by the needs of planetary probes and aerial reconnaissance, filtering initially received more attention than any other area of image processing and there are excellent detailed reference works (e.g., [Andrews and Hunt 1977; Pratt 1978; Gonzalez and Wintz 1977]). We cannot afford to examine these techniques in great detail here; instead, our intent is to describe a set of techniques that conveys the principal ideas.

Almost without exception, the best time to filter an image is at the image formation stage, before it has been sampled. A good example of this is the way chemical stains improve the effectiveness of microscopic tissue analysis by changing the image so that diagnostic features are obvious. In contrast, filtering after sampling often emphasizes random variations in the image, termed *noise*, that are undesirable effects introduced in the sampling stage. However, for cases where the image formation process cannot be changed, digital filtering techniques do exist. For example, one may want to suppress low spatial frequencies in an image and sharpen its edges. An image filtered in this way is shown in Fig. 3.2.

Note that in Fig. 3.2 the work of recognizing real-world objects still has to be done. Yet the edges in the image, which constitute object boundaries, have been made more prominent by the filtering operation. Good filtering functions are not easy to define. For example, one hazard with Fourier techniques is that sharp edges in the filter will produce unwanted “ringing” in the spatial domain, as evidenced by Fig. 2.5. Unfortunately, it would be too much of a digression to discuss techniques of filter design. Instead, the interested reader should refer to the references cited earlier.

3.2.1 Template Matching

Template matching is a simple filtering method of detecting a particular feature in an image. Provided that the appearance of this feature in the image is known accu-

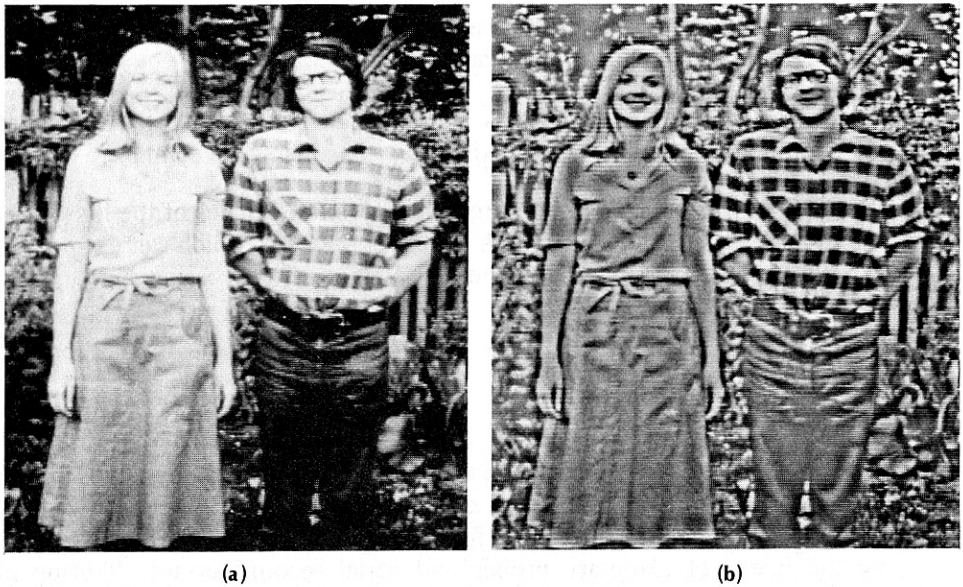


Fig. 3.2 Effects of high frequency filtering. (a) Original image. (b) Filtered image.

rately, one can try to detect it with an operator called a *template*. This template is, in effect, a subimage that looks just like the image of the object. A similarity measure is computed which reflects how well the image data match the template for each possible template location. The point of maximal match can be selected as the location of the feature. Figure 3.3 shows an industrial image and a relevant template.

Correlation

One standard similarity measure between a function $f(\mathbf{x})$ and a template $t(\mathbf{x})$ is the Euclidean distance $d(\mathbf{y})$ squared, given by

$$d(\mathbf{y})^2 = \sum_{\mathbf{x}} [f(\mathbf{x}) - t(\mathbf{x} - \mathbf{y})]^2 \quad (3.1)$$

By $\sum_{\mathbf{x}}$ we mean $\sum_{x=-M}^M \sum_{y=-N}^N$, for some M, N which define the size of the template extent. If the image at point \mathbf{y} is an exact match, then $d(\mathbf{y}) = 0$; otherwise, $d(\mathbf{y}) > 0$. Expanding the expression for d^2 , we can see that

$$d^2(\mathbf{y}) = \sum_{\mathbf{x}} [f^2(\mathbf{x}) - 2f(\mathbf{x})t(\mathbf{x} - \mathbf{y}) + t^2(\mathbf{x} - \mathbf{y})] \quad (3.2)$$

Notice that $\sum_{\mathbf{x}} t^2(\mathbf{x} - \mathbf{y})$ is a constant term and can be neglected. When $\sum_{\mathbf{x}} f^2(\mathbf{x})$ is approximately constant it too can be discounted, leaving what is called the *cross correlation* between f and t .

$$R_{ft}(\mathbf{y}) = \sum_{\mathbf{x}} f(\mathbf{x})t(\mathbf{x} - \mathbf{y}) \quad (3.3)$$

This is maximized when the portion of the image “under” t is identical to t .

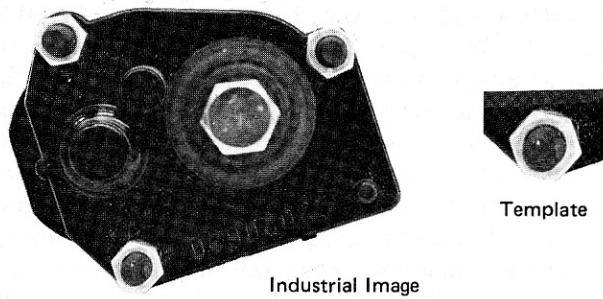


Fig. 3.3 An industrial image and template for a hexagonal nut.

One may visualize the template-matching calculations by imagining the template being *shifted* across the image to different offsets; then the superimposed values at this offset are *multiplied* together, and the products are *added*. The resulting sum of products forms an entry in the “correlation array” whose coordinates are the offsets attained by the source template.

If the template is allowed to take *all* offsets with respect to the image such that some overlap takes place, the correlation array is larger than either the template or the image. An $n \times n$ image with an $m \times m$ template yields an $(n + m - 1 \times n + m - 1)$ correlation array. If the template is not allowed to shift off the image, the correlation array is $(n - m + 1 \times n - m + 1)$; for $m < n$. Another form of correlation results from computing the offsets modulo the size of the image; in other words, the template “wraps around” the image. Being shifted off to the right, its right portion reappears on the left of the image. This sort of correlation is called *periodic* correlation, and those with no such wraparound properties are called *aperiodic*. We shall be concerned exclusively with aperiodic correlation. One can always modify the input to a periodic correlation algorithm by padding the outside with zeros so that the output is the aperiodic correlation.

Figure 3.4 provides an example of (aperiodic) “shift, add, multiply” template matching. This figure illustrates some difficulties with the simple correlation measure of similarity. Many of the advantages and disadvantages of this measure stem from the fact that it is linear. The advantages of this simplicity have mainly to do with the existence of algorithms for performing the calculation efficiently (in a transform domain) for the entire set of offsets. The disadvantages have to do with

Template	Image	Correlation
1 1 1	1 1 0 0 0	7 4 2 x x
1 1 1	1 1 1 0 0	5 3 2 x x
1 1 1	1 0 1 0 0	2 1 9 x x
	0 0 0 0 0	x x x x x
	0 0 0 0 8	x x x x x
		x = undefined

Fig. 3.4 (a) A simple template. (b) An image with noise. (c) The aperiodic correlation array of the template and image. Ideally peaks in the correlation indicate positions of good match. Here the correlation is only calculated for offsets that leave the template entirely within the image. The correct peak is the upper left one at 0, 0 offset. The “false alarm” at offset 2, 2 is caused by the bright “noise point” in the lower right of the image.

the fact that the metric is sensitive to properties of the image that may vary with the offset, such as its average brightness. Slight changes in the shape of the object, its size, orientation, or intensity values can also disturb the match.

Nonetheless, the idea of template matching is important, particularly if Eq. (3.3) is viewed as a *filtering* operation instead of an algorithm that does all the work of object detection. With this viewpoint one chooses one or more templates (filters) that transform the image so that certain features of an object are more readily apparent. These templates generally highlight subparts of the objects. One such class of templates is edge templates (discussed in detail in Section 3.3).

We showed in Section 2.2.4 that convolution and multiplication are Fourier transform pairs. Now note that the correlation operation in (3.3) is essentially the same as a convolution with a function $t'(\mathbf{x}) \equiv t(-\mathbf{x})$. Thus in a mathematical sense cross correlation and convolution are equivalent. Consequently, if the size of the template is sufficiently large, it is cheaper to perform the template matching operation in the spatial frequency domain, by the same transform techniques as for filtering.

Normalized Correlation

A crucial assumption in the development of Eq. (3.3) was that the image energy covered by the matching template at any offset was constant; this leads to a linear correlation matching technique. This assumption is approximately correct if the average image intensity varies slowly compared to the template size, but a bright spot in the image can heavily influence the correlation by affecting the sum of products violently in a small area (Fig. 3.4). Even if the image is well behaved, the range of values of the metric can vary with the size of the matching template. Are there ways of normalizing the correlation metric to make it insensitive to these variations?

There is a well-known treatment of the normalized correlation operation. It has been used for a variety of tasks involving registration and stereopsis of images [Quam and Hannah 1974]. Let us say that two input images are being matched to find the best offset that aligns them.

Let $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ be the images to be matched. q_2 is the patch of f_2 (possibly all of it) that is to be matched with a similar-sized patch of f_1 . q_1 is the patch of f_1 that is covered by q_2 when q_2 is offset by \mathbf{y} .

Let $E()$ be the expectation operator. Then

$$\sigma(q_1) = [E(q_1^2) - (E(q_1))^2]^{1/2} \quad (3.4)$$

$$\sigma(q_2) = [E(q_2^2) - (E(q_2))^2]^{1/2} \quad (3.5)$$

give the standard deviations of points in patches q_1 and q_2 . (For notational convenience, we have dropped the spatial arguments of q_1 and q_2 .) Finally, the normalized correlation is

$$N(\mathbf{y}) = \frac{E(q_1 q_2) - E(q_1)E(q_2)}{\sigma(q_1)\sigma(q_2)} \quad (3.6)$$

and $E(q_1 q_2)$ is the expected value of the product of intensities of points that are superimposed by the translation by \mathbf{y} .

The normalized correlation metric is less dependent on the local properties of the reference and input images than is the unnormalized correlation, but it is sensitive to the signal-to-noise content of the images. High uncorrelated noise in the two images, or the image and the reference, decreases the value of the correlation. As a result, one should exercise some care in interpreting the metric. If the noise properties of the image are known, one indication of reliability is given by the “(signal + noise)-to-noise” ratio. For the normalized correlation to be useful, the standard deviation of the patches of images to be matched (i.e., of the areas of image including noise) should be significantly greater than that of the noise. Then a correlation value may be considered significant if it is approximately equal to the theoretically expected one. Consider uncorrelated noise of identical standard deviation, in a patch of true value $f(x, y)$. Let the noise component of the image be $n(x, y)$. Then the theoretical maximum correlation is

$$1 - \frac{\sigma^2(n)}{\sigma^2(f+n)} \quad (3.7)$$

In matching an idealized, noise-free reference pattern, the best expected value of the cross correlation is

$$\frac{\sigma(f)}{\sigma(f+n)} \quad (3.8)$$

If the noise and signal characteristics of the data are known, the patch size may be optimized by using that information and the simple statistical arguments above. However, such considerations leave out the effects of systematic, nonstatistical error (such as imaging distortions, rotations, and scale differences between images). These systematic errors grow with patch size, and may swamp the statistical advantages of large patches. In the worst case, they may vitiate the advantages of the correlation process altogether.

Since correlation is expensive, it is advantageous to ensure that there is enough information in the patches chosen for correlation before the operation is done. One way to do this is to apply a cheap “interest operator” before the relatively expensive correlation. The idea here is to make sure that the image varies enough to give a usable correlation image. If the image is of uniform intensity, even its correlation with itself (autocorrelation) is flat everywhere, and no information about where the image is registered with itself is derivable. The “interest operator” is a way of finding areas of image with high variance. In fact, a common and useful interest measure is exactly the (directional) variance over small areas of image. One directional variance algorithm works as follows.

The Moravec interest operator [Moravec 1977] produces candidate match points by measuring the distinctness of a local piece of the image from its surround. To explain the operator, we first define a variance measure at a pixel (\mathbf{x}) as

$$\text{var}(x, y) = \left\{ \sum_{k, l \text{ in } s} [f(x, y) - f(x + k, y + l)]^2 \right\}^{1/2} \quad (3.9)$$

$$s = \left\{ (0, a), (0, -a), (a, 0), (-a, 0) \right\}$$

where a is a parameter. Now the interest operator value is initially the minimum of itself and surrounding points:

$$\text{IntOpVal}(\mathbf{x}) := \min_{y < 1} [\text{var}(\mathbf{x} + \mathbf{y})] \quad (3.10)$$

Next a check is made to see if the operator is a local maximum by checking neighbors again. Only local maxima are kept.

$$\begin{aligned} \text{IntOpVal}(\mathbf{x}) &:= 0 \text{ if} \\ \text{IntOpVal}(\mathbf{x}) &\geq \text{IntOpVal}(\mathbf{x} + \mathbf{y}) \\ &\text{for } \mathbf{y} \leq 1 \end{aligned} \quad (3.11)$$

Finally, candidate points are chosen from the IntOpVal array by thresholding.

$$\mathbf{x} \text{ is a candidate point iff } \text{IntOpVal}(\mathbf{x}) > T \quad (3.12)$$

The threshold is chosen empirically to produce some fraction of the total image points.

3.2.2 Histogram Transformations

A gray-level histogram of an image is a function that gives the frequency of occurrence of each gray level in the image. Where the gray levels are quantized from 0 to n , the value of the histogram at a particular gray level p , denoted $h(p)$, is the number or fraction of pixels in the image with that gray level. Figure 3.5 shows an image with its histogram.

A histogram is useful in many different ways. In this section we consider the histogram as a tool to guide gray-level transformation algorithms that are akin to filtering. A very useful image transform is called *histogram equalization*. Histogram equalization defines a mapping of gray levels p into gray levels q such that the distribution of gray levels q is uniform. This mapping stretches contrast (expands the

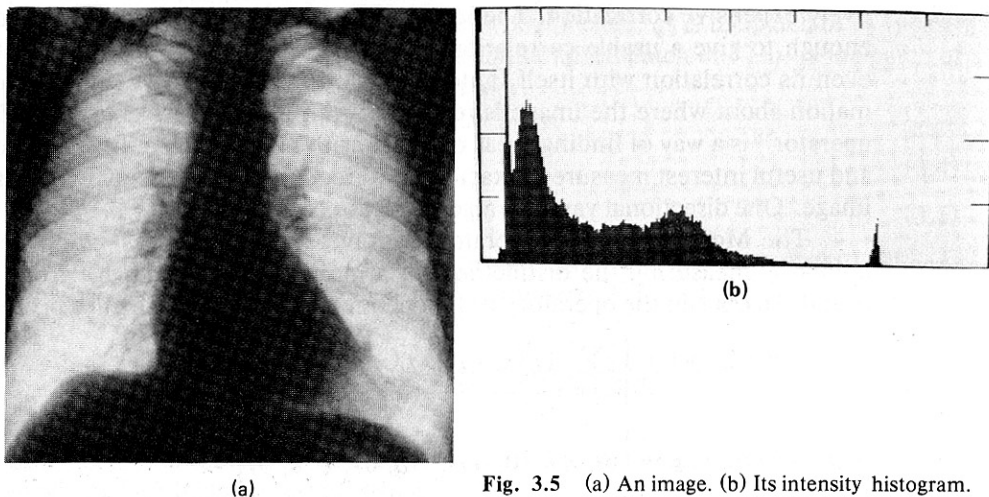


Fig. 3.5 (a) An image. (b) Its intensity histogram.

range of gray levels) for gray levels near histogram maxima and compresses contrast in areas with gray levels near histogram minima. Since contrast is expanded for most of the image pixels, the transformation usually improves the detectability of many image features.

The histogram equalization mapping may be defined in terms of the *cumulative* histogram for the image. To see this, consider Fig. 3.6a. To map a small interval of gray levels dp onto an interval dq in the general case, it must be true that

$$g(q) dq = h(p) dp \quad (3.13)$$

where $g(q)$ is the new histogram. If, in the histogram equalization case, $g(q)$ is to be uniform, then

$$g(q_2) = \frac{N^2}{M} \quad (3.14)$$

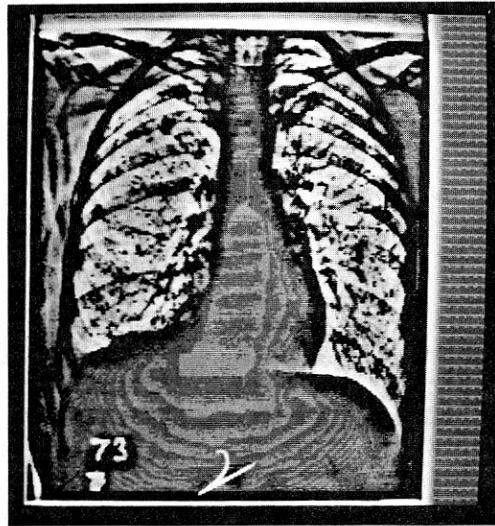
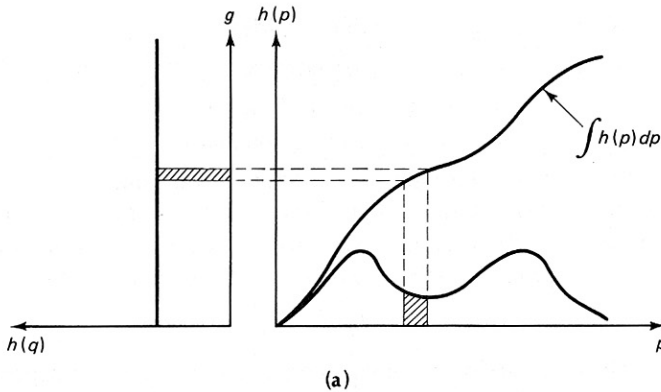


Fig. 3.6 (a) Basis for a histogram equalization technique. (b) Results of histogram equalization.

where N^2 is the number of pixels in the image and M is the number of gray levels. Thus combining Eqs. (3.13) and (3.14) and integrating, we have

$$g(q) = \frac{M}{N^2} \int_0^q h(p) dp \quad (3.15)$$

But Eq. (3.15) is simply the equation for the normalized cumulative histogram. Figure 3.6b shows the histogram-equalized image.

3.2.3 Background Subtraction

Background subtraction can be another important filtering step in early processing. Many images can have slowly varying background gray levels which are incidental to the task at hand. Examples of such variations are:

- Solution gradients in cell slides
- Lighting variations on surfaces in office scenes
- Lung images in a chest radiograph

Note that the last example is only a “background” in the context of looking for some smaller variations such as tumors or pneumoconiosis.

Background subtraction attempts to remove these variations by first approximating them (perhaps analytically) with a background image f_b and then subtracting this approximation from the original image. That is, the new image f_n is

$$f_n(\mathbf{x}) = f(\mathbf{x}) - f_b(\mathbf{x}) \quad (3.16)$$

Various functional forms have been tried for analytic representations of slowly varying backgrounds. In the simplest cases, $f_b(\mathbf{x})$ may be a constant,

$$f_b(\mathbf{x}) = c \quad (3.17)$$

or linear,

$$f_b(\mathbf{x}) = \mathbf{m} \cdot \mathbf{x} + c \quad (3.18)$$

A more sophisticated background model is to use a low-pass filtered variant of the original image:

$$f_b(\mathbf{x}) = \mathcal{F}^{-1}[H(\mathbf{u}) F(\mathbf{u})] \quad (3.19)$$

where $H(\mathbf{u})$ is a low-pass filtering function. The problem with this technique is that it is global; one cannot count on the “best” effect in any local area since the filter treats all parts of the image identically. For the same reason, it is difficult to design a Fourier filter that works for a number of very different images.

A workable alternative is to approximate $f_b(\mathbf{x})$, using *splines*, which are piecewise polynomial approximation functions. The mathematics of splines is treated in Chapter 8 since they find more general application as representations of shape. The filtering application is important but specialized. The attractive feature of a spline approximation for filtering is that it is *variation diminishing* and *spatially variant*. The spline approximation is guaranteed to be “smoother” than the origi-

nal function and will approximate the background differently in different parts of the image. The latter feature distinguishes the method from Fourier-domain techniques which are spatially invariant. Figure 3.7 shows the results of spline filtering.

3.2.4 Filtering and Reflectance Models

Leaving the effects of imaging geometry implicit (Section 2.2.2), the definitions in Section 2.2.3 imply that the image irradiance (gray level) at the image point x' is proportional to the product of the scene irradiance E and the *reflectance* r at its corresponding world point x .

$$f(x') = E(x)r(x) \quad (3.20)$$

The irradiance at x is the sum of contributions from all illumination sources, and the reflectance is that portion of the irradiance which is reflected toward the observer (camera). Usually E changes slowly over a scene, whereas r changes quickly over edges, due to varying face angles, paint, and so forth. In many cases one would like to detect these changes in r while ignoring changes in E . One way of doing this is to filter the image $f(x')$ to eliminate the slowly varying component. However, as f is the *product* of illumination and reflectance, it is difficult to define an operation that selectively diminishes E while retaining r . Furthermore, such an operation must retain the positivity of f . One solution is to take the logarithm of Eq. (3.20). Then

$$\log f = \log E + \log r \quad (3.21)$$

Equation (3.21) shows two desirable properties of the logarithmic transformation: (1) the logarithmic image is positive in sign, and (2) the image is a superposition of the irradiance component and reflectance component. Since reflectance is an in-

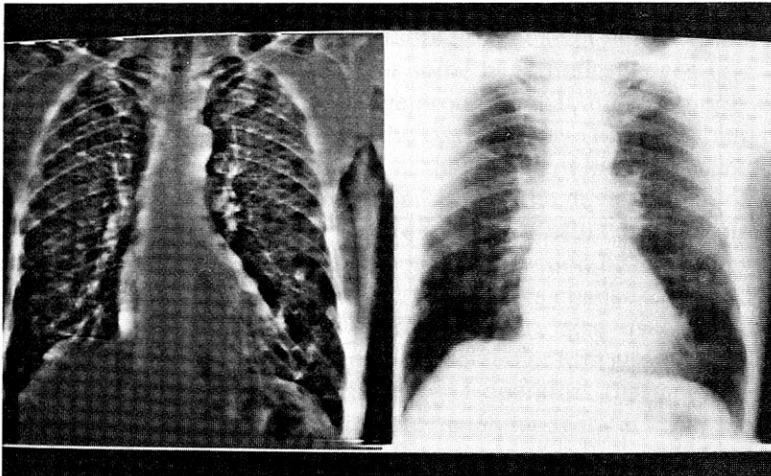
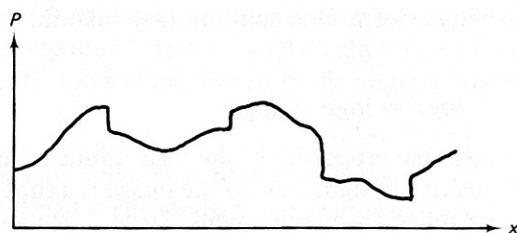


Fig. 3.7 The results of spline filtering to remove background variation.

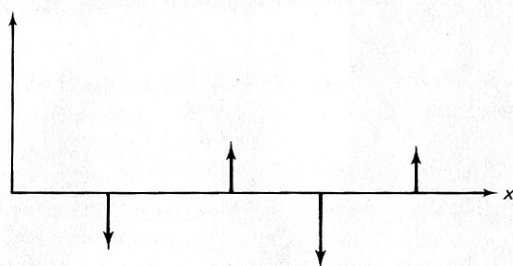
trinsic characteristic of objects, the obvious goal of image analysis is to recognize the reflectance component under various conditions of illumination. Since the separation of two components is preserved under linear transformations and the irradiance component is usually of low spatial frequency compared to the reflectance component, filtering techniques can suppress the irradiance component of the signal relative to the reflectance component.

If the changes in r occur over very short distances in the images, r may be isolated by a three-step process [Horn 1974]. First, to enhance reflectance changes, the image function is differentiated (Section 3.3.1). The second step removes the low irradiance gradients by thresholding. Finally, the resultant image is integrated to obtain an image of perceived "lightness" or reflectance. Figure 3.8 shows these steps for the one-dimensional case.

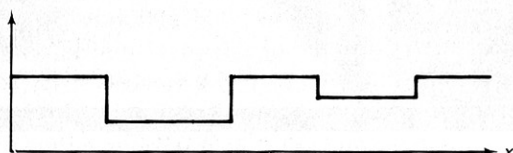
A basic film parameter is density, which is proportional to the logarithm of transmitted intensity; the logarithmically transformed image is effectively a *density image*. In addition to facilitating the extraction of lightness, another advantage of the density image is that it is well matched to our visual experience. The ideas for many image analysis programs stem from our visual inspection of the image. However, the human visual system responds logarithmically to light intensity and also enhances high spatial frequencies [Stockham 1972]. Algorithms derived from



(a)



(b)



(c)

Fig. 3.8 Steps in processing an image to detect reflectance. (a) Original image. (b) Differentiation followed by thresholding. (c) Integration of function in (b).

introspective reasoning about the perceived image (which has been transformed by our visual system) will not necessarily be successful when applied to an unmodified intensity image. Thus one argument for using a density transformation followed by high spatial frequency emphasis filtering is that the computer is then “seeing” more like the human image analyzer.

3.3 FINDING LOCAL EDGES

Boundaries of objects tend to show up as intensity discontinuities in an image. Experiments with the human visual system show that boundaries in images are extremely important; often an object can be recognized from only a crude outline [Attneave 1954]. This fact provides the principal motivation for representing objects by their boundaries. Also, the boundary representation is easy to integrate into a large variety of object recognition algorithms.

One might expect that algorithms could be designed that find the boundaries of objects directly from the gray-level values in the image. But when the boundaries have complicated shapes, this is difficult. Much greater success has been obtained by first transforming the image into an intermediate image of *local* gray-level discontinuities, or edges, and then composing these into a more elaborate boundary. This strategy reflects the principle: When the gap between representations becomes too large, introduce intermediate representations. In this case, boundaries that are highly model-dependent may be decomposed into a series of local edges that are highly model-independent.

A local edge is a small area in the image where the local gray levels are changing rapidly in a simple (e.g., monotonic) way. An *edge operator* is a mathematical operator (or its computational equivalent) with a small spatial extent designed to detect the presence of a local edge in the image function.

It is difficult to specify a priori which local edges correspond to relevant boundaries in the image. Depending on the particular task domain, different local changes will be regarded as likely edges. Plots of gray level versus distance along the direction perpendicular to the edge for some hypothetical edges (Fig. 3.9a-e) demonstrate some different kinds of “edge profiles” that are commonly encountered. Of course, in most practical cases, the edge is noisy (Fig. 3.9d) and may appear as a composite of profile types. The fact that different kinds of edge operators perform best in different task domains has prompted the development of a variety of operators. However, the unifying feature of most useful edge operators is that they compute a *direction* which is aligned with the direction of maximal gray-level change, and a *magnitude* describing the severity of this change. Since edges are a high-spatial-frequency phenomenon, edge finders are also usually sensitive to high-frequency noise, such as “snow” on a TV screen or film grain.

Operators fall into three main classes: (1) operators that approximate the mathematical gradient operator, (2) template matching operators that use multiple templates at different orientations, and (3) operators that fit local intensities with parametric edge models. Representative examples from the first two of these categories appear in this section. The computer vision literature abounds with edge