Chapter 5

Filtering and Enhancing Images

This chapter describes methods to enhance images for either human consumption or for further automatic operations. Perhaps we need to reduce noise in the image; or, certain image details need to be emphasized or suppressed. Chapter 1 already introduced two methods of image filtering: first, in the blood cell image, isolated black or white pixels were removed from larger uniform regions; secondly, it was shown how a contrast operator could enhance the boundaries between different objects in the image, i.e. improve the constrast between the pictures and the wall.

This chapter is about *image processing*, since the methods take an input image and create another image as output. Other appropriate terms often used are *filtering*, *enhancement*, or *conditioning*. The major notion is that the image contains some signal or structure, which we want to extract, along with uninteresting or unwanted variation, which we want to suppress. If decisions are made about the image, they are made at the level of a single pixel or its local neighborhood. We have already seen how we might label an image pixel as *object* versus *background* or *boundary* versus *not boundary*.

Image processing has both theory and methods that can fill several books. Only a few classical image processing concepts are treated here in detail. Most methods presented use the important notion that each pixel of the output image is computed from a local neighborhood of the corresponding pixel in the input image. However, a few of the enhancement methods are global in that all of the input image pixels are used in some way in creating the output image. The two most important concepts presented are those of (1) *matching* an image neighborhood with a pattern or mask (*correlation*) and (2) *convolution*, a single method that can implement many useful filtering operations.

5.1 What needs fixing?

Before launching into the methods of this chapter, it's useful to review some of the problems that need them. Two general categories of problems follow.

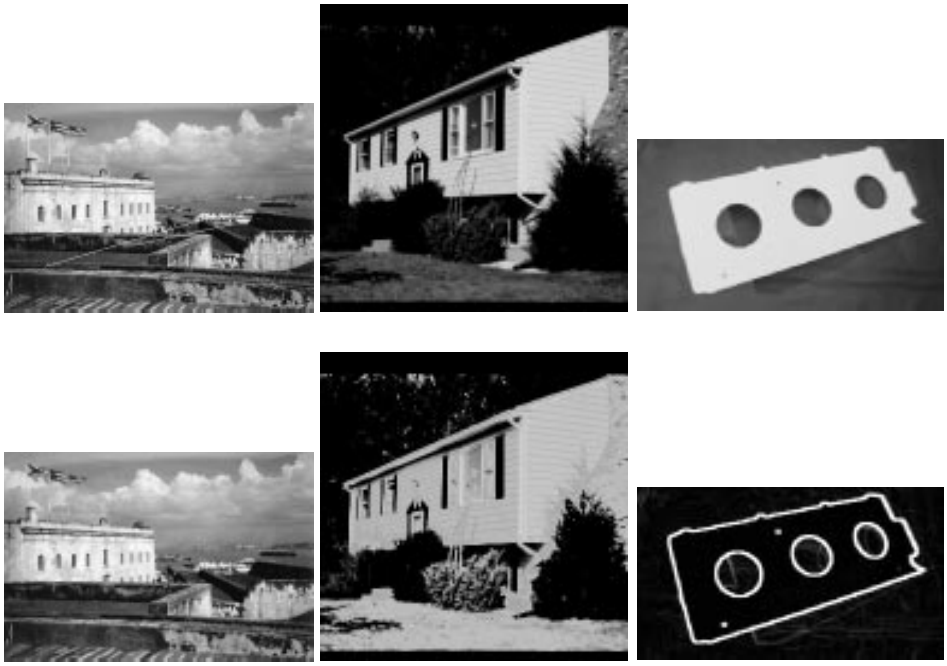


Figure 5.1: (Left) Scratches from original photo of San Juan are removed; (center) intensities of house photo with dark areas are rescaled to show better detail in some regions; (right) image of airplane part has edges enhanced to support automatic recognition and measurement.

5.1.1 An Image needs Improvement

- On safari in Africa you got a shot of a lion chasing an antelope. Unfortunately, the sun was behind your main actor and much of your picture is too dark. The picture can be enhanced by boosting the lower intensities and not the higher ones.
- An old photo has a long bright scratch, but is otherwise fine. The photo can be digitized and the scratch removed.
- A paper document needs to be scanned and converted into a text file. Before applying character recognition, noise pixels need to be cleaned from the background and dropouts in the characters need to be filled.

5.1.2 Low-level features must be detected

- 0.12 inch diameter wire is made by a process that requires (closed loop) feedback from a vision sensor that constantly measures the wire diameter. The two sides of the wire can be located using an edge operator which accurately identifies the boundary between the wire and its background.
- An automatic pilot system for a car can steer by constantly monitoring the white lines on the highway. The white lines can be found in frames of a forward-looking video camera by finding two edges with opposite contrast and similar direction.

Figure 5.2: (Left) Original sensed fingerprint; (center) image enhanced by detection and thinning of ridges; (right) identification of special features called "minutia", which can be used for matching to millions of fingerprint representations in a database (contributed by Shaoyun Chen and Anil Jain).

- A paper blueprint needs to be converted into a CAD (computer-aided-design) model. Part of the process involves finding the blueprint lines as dark streaks in the image about 1 pixel wide.

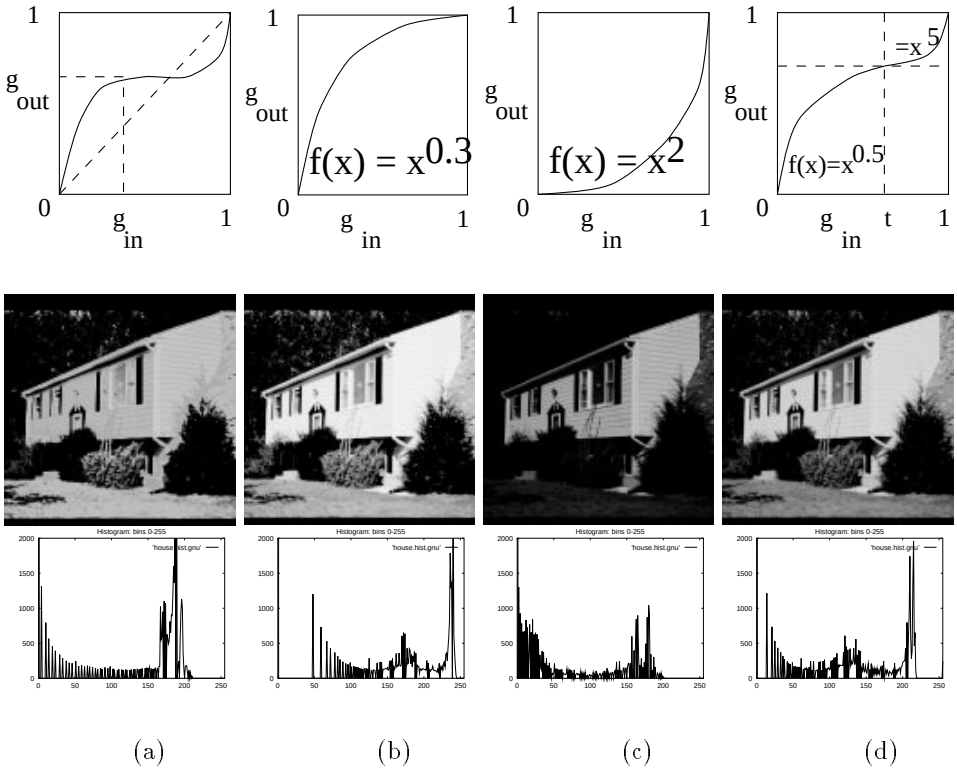
This chapter deals mostly with traditional methods of *image enhancement* and somewhat with *image restoration*. Before moving on, it is important to define and distinguish these terms.

1 Definition **Image enhancement** *operators improve the detectability of important image details or objects by man or machine. Example operations include noise reduction, smoothing, contrast stretching, and edge enhancement.*

2 Definition **Image restoration** *attempts to retore a degraded image to an ideal condition. This is possible only to the extent that the physical processes of ideal image formation and image degradation are understood and can be modeled. The process of restoration involves inverting the degradation process to return to an ideal image.*

5.2 Grey level mapping

It is common to *enhance* images by changing the intensity values of pixels. Most software tools for image processing have several options for changing the appearance of an image by transforming the pixels via a single function that maps an input grey value into a new output value. It is easy to extend this so that a user can indicate several different image regions and apply a separate mapping function to each. Remapping the grey values is often called *stretching* because it is common to stretch the grey values of an image that is too dark onto the full set of grey values available. Figure 5.3 shows a picture whose intensity values are stretched according to four different mapping functions. Figure 5.3(a) shows the result of applying an ad hoc function, which was designed by the user of an interactive enhancement tool. The user defines the grey level mapping function $g_{out} = f(g_{in})$ using the computer mouse: typically the image tool fits a smooth spline through points chosen by the user. Figure 5.3(b) and (c) show remapping using the function $f(x) = x^{1/\gamma}$: this function nonlinearly boosts or reduces intensities according to whether $\gamma > 1$ or $\gamma < 1$.



(a) (b) (c) (d)

Figure 5.3: (Top row) Intensity mapping functions f and (center row) output images resulting from applying f to the original image from Figure 5.1 and (bottom row) resulting grey level histograms. (a) House image pixels remapped according to a user-defined smooth function that stretches the low range of intensities and compresses the higher range ; (b) Gamma correction with $f(x) = x^{0.3}$ boosts dark pixels more than bright ones; (c) Gamma correction with $f(x) = x^{2.0}$ reduces bright pixels more than dark pixels; (d) Use of two different Gamma values to boost dark pixels $< t$ and reduce bright pixels $> t$. Different scene objects appear with different clarity in the several images.

This operation (called *Gamma correction*) might be the proper model to *restore* an image to an original form after undergoing known physical distortion. Figure 5.3(d) shows that a more complex mapping can be defined by using different functions for darker pixels and brighter pixels according to some defining threshold t. All of the functions in Figure 5.3 stretch or extend at least some range of intensities to show more variation in the output. Image variation will be increased in the intensity ranges where the slope of function $f(x)$ is greater than 1.

3 Definition *A* **point operator** *applied to an image is an operator in which the output pixel is determined only by the input pixel,* $Out[x,y] = f(In[x,y])$*: possibly function* f *depends upon some global parameters.*

Figure 5.4: Histogram equalization maps the grey scale such that the output image uses the entire range available and such that there are approximately the same number of pixels of each grey value in the output image: images after histogram equalization are at the right.

4 DEFINITION *A* **contrast stretching** *operator is a point operator that uses a piecewise smooth function* $f(In[x, y])$ *of the input grey level to enhance important details of the image.*

Because point operators map one input pixel to one output pixel, they can be applied to an entire pixel array in any sequential order or can be applied in parallel. Ad hoc mappings of intensities, including nonmonotonic ones as in Figure 5.3, can be very useful in enhancing images for general human consumption in graphics and journalism. However, in certain domains, such as radiology, one must be careful not to alter a meaningful intensity scale to which human experts and intricate sensors are carefully calibrated. Finally, we note that the performance of some algorithms for machine vision might not be changed by monotonic grey scale stretching ($f(g_2) > f(g_1)$ whenever grey level $g_2 > g_1$), although humans monitoring the process might appreciate the enhanced images.

5.2.1 Histogram equalization

Histogram equalization is often tried to enhance an image. The two requirements on the operator are that (a) the output image should use all available grey levels and (b) the output image has approximately the same number of pixels of each grey level. Requirement (a) makes good sense but (b) is ad hoc and its effectiveness must be judged empirically. Figure 5.4 shows images transformed by histogram equalization. One can see that the remapping of grey levels does indeed change the appearance of some regions; for example, the texture of the shrub in front of the house is better defined. The face image had been cropped from a larger image and the cropped window did not have many lower intensity pixels. Requirement (b) will cause large homogeneous regions, such as sky, to be remapped into more grey levels and hence to show more texture. This may or may not help in image interpretation.

Requirements (a) and (b) mean that the target output image uses all grey values $z = z_1, z = z_2, \ldots, z = z_n$ and that each grey level z_k is used approximately $q = (R \times C)/n$ times, where R, C are the number of rows and columns of the image. The input image histogram $H_{in}[]$ is all that is needed in order to define the stretching function f. $H_{in}[i]$ is the number of pixels of the input image having grey level z_i. The first grey level threshold t_1 is found by advancing i in the input image histogram until approximately q_1 pixels are accounted

for: all input image pixels with grey level $z_k < t_1 - 1$ will be mapped to grey level z_1 in the output image. Threshold t_1 is formally defined by the following computational formula:

$$\sum_{i=1}^{t_1-1} H_{in}[i] \leq q_1 < \sum_{i=1}^{t_1} H_{in}[i].$$

This means that t_1 is the smallest grey level such that the original histogram contains no more than q pixels with lower grey values. The k-th threshold t_k is defined by continuing the iteration:

$$\sum_{i=1}^{t_k-1} H_{in}[i] \leq (q_1 + q_2 + \ldots + q_k) < \sum_{i=1}^{t_k} H_{in}[i].$$

A practical implementation of the mapping f is a *lookup table* that is easily obtained from the above process. As the above formula is computed, the threshold values t_k are placed (possibly repetitively) in array $T[]$ as long as the inequality holds: thus we'll have function $z_{out} = f(z_{in}) = T[z_{in}]$.

Exercise 1

An input image of 200 pixels has the following histogram: $H_{in} = [0, 0, 20, 30, 5, 5, 40, 40, 30, 20, 10, 0, 0, 0, 0, 0]$. (a) Using the formula for equalizing the histogram (of 15 grey levels), what would be the output image value $f(8)$? (b) Repeat question (a) for $f(11)$. (c) Give the lookup table $T[]$ implementing f for transforming all input image values.

Exercise 2 An algorithm for histogram equalization

Use pseudocode to define an algorithm for histogram equalization. Be sure to define all the data items and data structures being used.

Exercise 3 A histogram equalization program

(a) Using the pseudocode from the previous exercise, implement and test a program for histogram equalization. (b) Describe how well it works on different images.

It is often the case that the range of output image grey levels is larger than the range of input image grey levels. It is thus impossible for any function f to remap grey levels onto the entire output range. If an approximately uniform output histogram is really wanted, a random number generator can be used to map an input value z_{in} to some neighborhood of $T[z_{in}]$. Suppose that the procedure described above calls for mapping $2q$ pixels of level g to output level g_1 and 0 pixels to level $g_1 + 1$. We can then simulate a coin flip so that an input pixel of level g has a 50-50 chance of mapping to either g_1 or $g_1 + 1$.

5.3 Removal of Small Image Regions

Often, it is useful to remove small regions from an image. A small region might just be the result of noise, or it might represent low level detail that should be suppressed from the image description being constructed. Small regions can be removed by changing single pixels or by removing components after connnected components are extracted.

Exercise 4

Give some arguments for and against "randomizing" function f in order to make the output histogram more uniform.

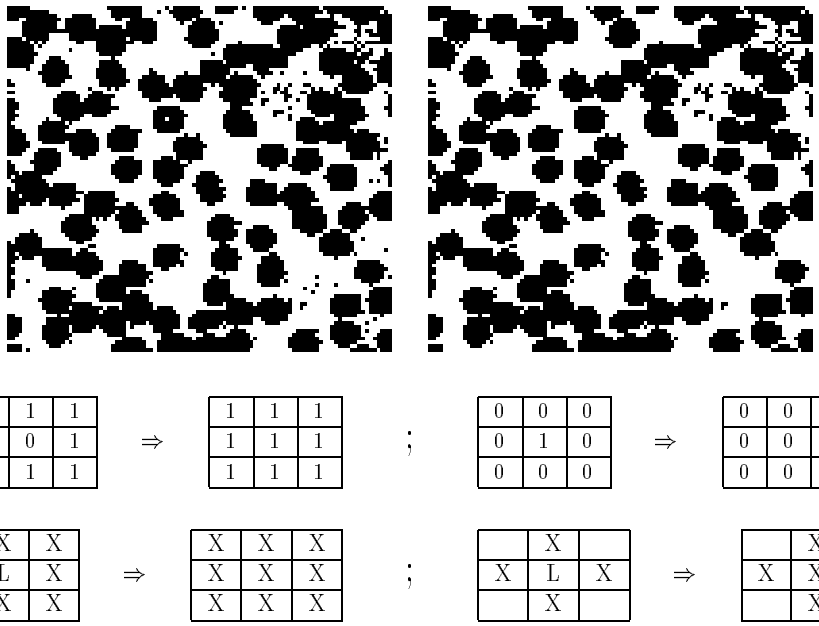


Figure 5.5: Binary image of red blood cells (top left) with salt and pepper noise removed (top right). Middle row: templates showing how binary pixel neighborhoods can be cleaned. Bottom row: templates defining isolated pixel removal for a general labeled input image; (bottom left) 8-neighborhood decision and (bottom right) 4-neighborhood decision.

5.3.1 Removal of Salt-and-Pepper Noise

The introductory chapter briefly discussed how single anomalous pixels can be removed from otherwise homogeneous regions and methods were extended in Chapter 3. The presence of single dark pixels in bright regions, or single bright pixels in dark regions, is called *salt and pepper noise*. The analogy to real life is obvious. Often, salt and pepper noise is the natural result of creating a binary image via thresholding. Salt corresponds to pixels in a dark region that somehow passed the threshold for bright, and pepper corresponds to pixels in a bright region that were below threshold. Salt and pepper might be classification errors resulting from variation in the surface material or illumination, or perhaps noise in the analog/digital conversion process in the frame grabber. In some cases, these isolated pixels are not classification errors at all, but are tiny details contrasting with the larger neighborhood, such as a button on a shirt or a clearing in a forest, etc. and it may be that the description needed for the problem at hand prefers to ignore such detail.

Figure 5.5 shows the removal of salt and pepper noise from a binary image of red blood cells. The operations on the input image are expressed in terms of *masks* given at the bottom of the figure. If the input image neighborhood matches the mask at the left, then it is changed into the neighborhood given by the mask at the right. Only two such masks

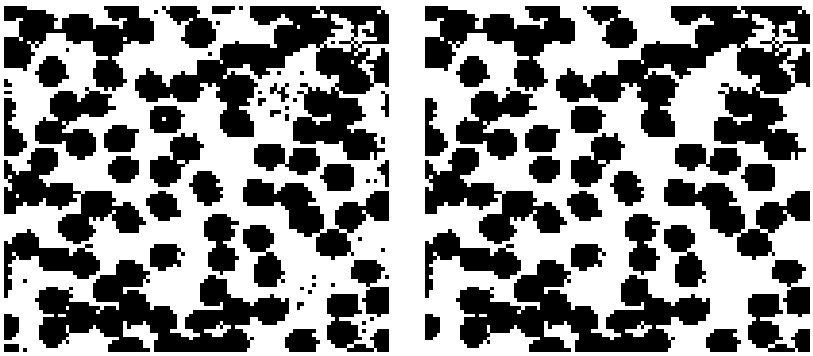


Figure 5.6: Original blood cell image (left) with salt and pepper noise removed and components with area 12 pixels or less removed (right).

are needed for this approach. In case the input image is a labeled image created by use of a set of thresholds or some other classification procedure, a more general mask can be used. As shown in the bottom row of Figure 5.5, any pixel label 'L' that is isolated in an otherwise homogenous 8-neighborhood is coerced to the majority label 'X'. 'L' may be any of the k labels used in the image. The figure also shows that either the eight or four neighborhood can be used for making the decision; in the case of the 4-neighborhood, the four corner pixels are not used in the decision. As shown in Chapter 3, use of different neighborhoods can result in different output images, as would be the case with the red blood cell image. For example, some of the black pixels at the bottom left of Figure 5.5 that are only 8-connected to a blood cell region could be cleaned using the 4-neighborhood mask.

5.3.2 Removal of Small Components

Chapter 3 discussed how to extract the connected components of a binary image and defined a large number of features that could be computed from the set of pixels comprising a single component. The description computed from the image is the set of components, each representing a region extracted from the background, and the features computed from that region. An algorithm can remove any of the components from this description based on computed features; for example, components with a small number of pixels or components that are very thin can be removed. This processing could remove the small noise region toward the upper right in the blood cell image of Figure 5.5 (top right). Once small regions are culled from the description it may not be necessary, or possible, to generate the corresponding output image. If an output image is necessary, information must be saved in order to return to the input image and correctly recode the pixels from the changed regions. Figure 5.6 shows the blood cell image after removal of salt and pepper noise and components of area 12 pixels or less.

5.4 Image Smoothing

Often, an image is composed of some underlying ideal structure, which we want to detect and describe, together with some random noise or artifact, which we would like to remove.

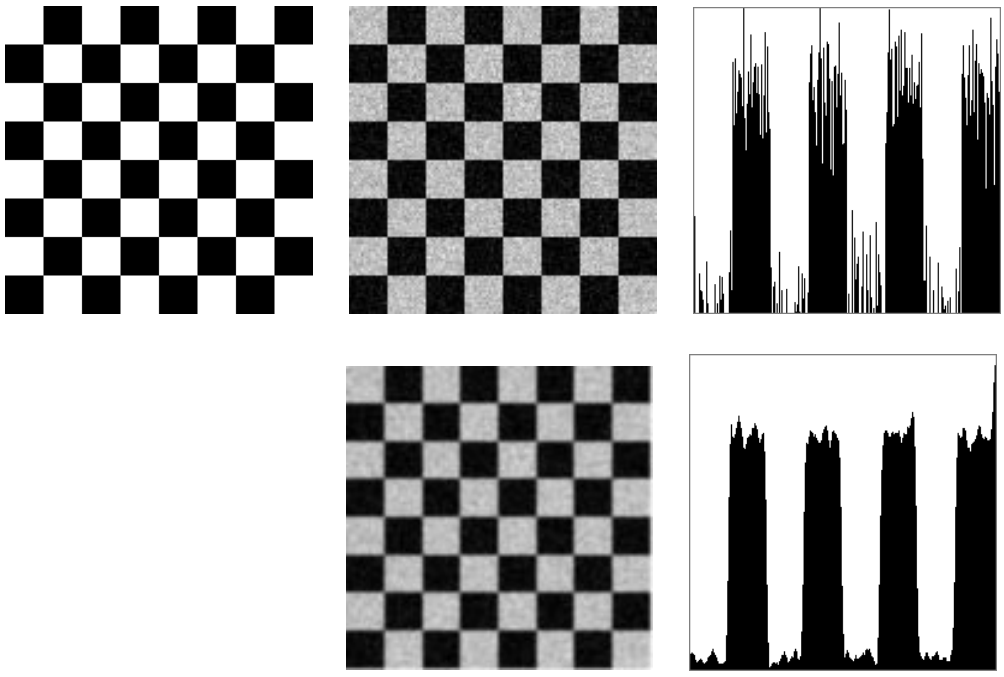


Figure 5.7: Ideal image of checkerboard (top left) with pixel values of 0 in the black squares and 255 in the white squares; (top center) image with added Gaussian noise of standard deviation 30; (top right) pixel values in a horizontal row 100 from the top of the noisy image; (bottom center) noise averaged using a 5x5 neighborhood centered at each pixel; (bottom right) pixels across image row 100 from the top.

For example, a simple model is that pixels from the image region of a uniform object have value $g_r + N(0, \sigma)$, where g_r is some expected grey level for ideal imaging conditions and $N(0, \sigma)$ is Gaussian noise of mean 0 and standard deviation σ. Figure 5.7(top left) shows an ideal checkerboard with uniform regions. Gaussian noise has been added to the ideal image to create the noisy image in the center: note that the noisy values have been clipped to remain within the interval $[0, 255]$. At the top right is a plot of the pixel values across a single (horizontal) row of the image.

Noise which varies randomly above and below a nominal brightness value for a region can be reduced by averaging a neighborhood of values.

$$OutputImage[r, c] \; = \; average \; of \; some \; neighborhood \; of \; InputImage[r, c] \tag{5.1}$$

$$Out[r, c] \; = \; (\sum_{i=-2}^{+2} \sum_{j=-2}^{+2} In[r + i, c + j])/25 \tag{5.2}$$

Equation 5.2 defines a smoothing filter that averages 25 pixel values in a 5x5 neighborhood of the input image pixel in order to create a smoothed output image. Figure 5.7(bottom center) illustrates its use on the checkerboard image: the image row shown at the bottom right of the figure is smoother than the input image row shown at the top right. This row is not actually formed by averaging just within that row, but by using pixel values from five

rows of the image. Also note that while the smooth image is cleaner than the original, it is not as sharp.

5 Definition (Box filter) *Smoothing an image by equally weighting a rectangular neighborhood of pixels is called using a box filter.*

Rather than weight all input pixels equally, it is better to reduce the weight of the input pixels with increasing distance from the center pixel $I[x_c, y_c]$. The Gaussian filter does this and is perhaps the most commonly used of all filters. Its desirable properties are discussed more fully below.

6 Definition (Gaussian filter) *When a Gaussian filter is used, pixel* $[x, y]$ *is weighted according to*

$$g(x, y) \;=\; \frac{1}{\sqrt{2\pi}\;\sigma} e^{-\frac{d^2}{2\sigma^2}}$$

where $d = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ *is the distance of the neighborhood pixel* $[x, y]$ *from the center pixel* $[x_c, y_c]$ *of the output image where the filter is being applied.*

Later in this chapter, we will develop in more detail both the theory and methods of smoothing and will express edge detection within the same framework. Before proceeding in that direction, we introduce the useful and intuitive *median filter*.

5.5 Median Filtering

Averaging is sure to produce a better estimate of $I[x, y]$ when the average is taken over a homogeneous neighborhood with zero-mean noise. When the neighborhood straddles the boundary between two such regions, the estimate uses samples from two intensity populations, resulting in blurring of the boundary. A popular alternative is the *median filter*, which replaces a pixel value with the *median* value of the neighborhood.

7 Definition (median) *Let* $\mathbf{A}[i]_{i=0\ldots(n-1)}$ *be a sorted array of* n *real numbers. The median of the set of numbers in* $\mathbf{A}$ *is* $\mathbf{A}[(n-1)/2]$

Sometimes, the cases of n being odd versus even are differentiated. For odd n, the array has a unique middle as defined above. If n is even, then we can define that there are two medians, at $\mathbf{A}[n/2]$ and $\mathbf{A}[n/2-1]$, or one median, which is the average of these two values. Although sorted order was used to define the median, the n values do not have to be fully sorted in practice to determine the median. The well-known quicksort algorithm can easily be modified so that it only recurses on the subarray of $\mathbf{A}$ that contains the $(n+1)/2$-th element; as soon as a sort pivot is placed in that array position the median of the entire set is known.

Figure 5.8 shows that the median filter can smooth noisy regions yet better preserve the structure of boundaries between them. When a pixel is actually chosen from one of the white squares, but near the edge, it is likely that the majority of the values in its neighborhood are white pixels with noise: if this is true, then neighboring pixels from a black square will not be used to determine the output value. Similarly, when computing

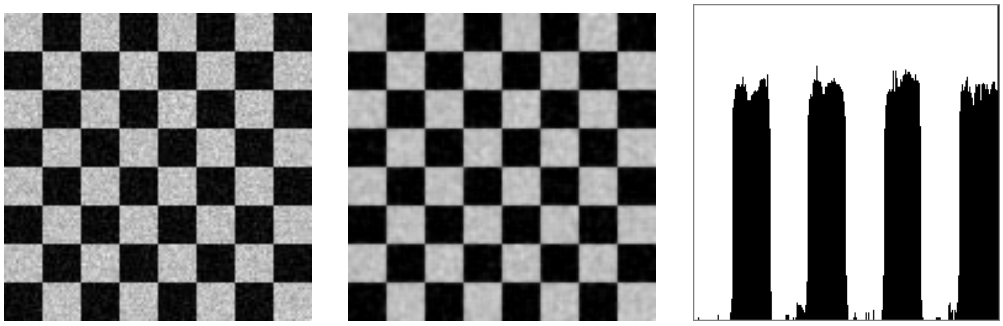


Figure 5.8: (Left) Noisy checkerboard image; (center) result of setting output pixel to the median value of a 5x5 neighborhood centered at the pixel; (right) display of pixels across image row 100 from the top; compare to Figure 5.7.

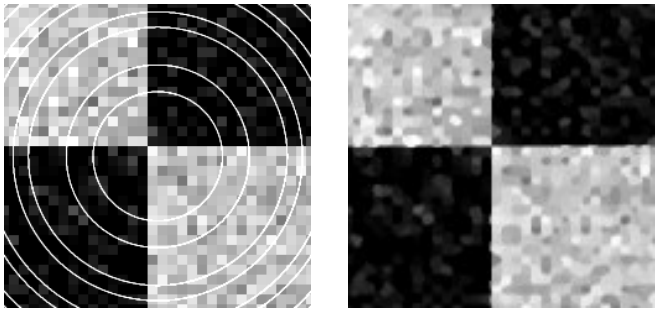


Figure 5.9: (Left) Input image contains both Gaussian noise and bright ring artifacts added to four previously uniform regions; (right) result of applying a 7x7 median filter.

the output pixel on the black side of an edge, it is highly likely that a majority of those pixels are black with noise, meaning that any neighborhood samples from the adjacent white region will not be used any further in computing the output value. Thus, unlike smoothing using averaging, the median filter tends to preserve edge structure while at the same time smoothing uniform regions. Median filtering can also remove salt-and-pepper noise and most other small artifacts that effectively replace a few ideal image values with noise values *of any kind*. Figure 5.9 shows how structured artifact can be removed while at the same time reducing variation in uniform regions and preserving boundaries between regions.

Computing the median requires more computation time than computing a neighborhood average, since the neighborhood values must be partially sorted. Moreover, median filtering is not so easily implemented in special hardware that might be necessary for real-time processing, such as in a video pipeline. However, in many image analysis tasks, its value in image enhancement is worth the time spent.

5.5.1 Computing an Output Image from an Input Image

Now that examples have been presented showing what kinds of image enhancements will be done, it's important to consider how these operations on images can be performed. Different

Exercise 5 Modifying Quicksort 1

(a) Find pseudocode for the traditional quicksort algorithm from one of the many data structures and algorithms texts. Modify the algorithm to return only the median as soon as it can be decided. (b) What is the computational effort of finding the median relative to performing the entire sort? (b) Implement your algorithm in some programming language and test it on some example images.

Exercise 6 Modifying Quicksort 2

Consider using the quicksort algorithm from above to detect steps in the picture function, such as the steps from black to white squares in the checkerboard images. Suppose the median of a neighborhood about **I**[r,c] has just been found by placing a pivot value in array position **A**[n/2]. Describe how the rest of the array can be processed to decide whether or not the pixel at [r,c] is or is not on the boundary between regions of different brightness.

options for controlling the filtering of an input image to produce an enhanced output image are represented by the generic algorithm below.

Algorithm 1 shows simple sequential program control which considers each pixel of the output image **G** in raster scan order and computes the value of $\mathbf{G}[\mathbf{r},\mathbf{c}]$ using a neighborhood of pixels around $\mathbf{F}[\mathbf{r},\mathbf{c}]$. It should be clear that the pixels of output image **G** could be computed in a random order rather than in row and column order, and, in fact, could all be computed in parallel. This holds because the input image is unchanged by any of the neighborhood computations. Secondly, the procedure compute_using_neighbors could be implemented to perform either boxcar or median filtering. For boxcar filtering, the procedure need only add up the $w \times h$ pixels of the neighborhood of $\mathbf{F}[\mathbf{r},\mathbf{c}]$ and then divide by the number of pixels $w \times h$. To implement a median filter, the procedure could copy those $w \times h$ pixel values into a local array **A** and partially sort it to obtain their median value.

Control could also be arranged so that only h rows of the image were in main memory at any one time. Outputs $\mathbf{G}[\mathbf{r},\mathbf{c}]$ would be computed only for the middle row r. Then, a new row would be input to replace the oldest row in memory and the next output row of $\mathbf{G}[\mathbf{r},\mathbf{c}]$ would be computed. This process is repeated until all possible output rows are computed. Years ago, when computers had small main memories, the primary storage for images was on disk and many algorithms had to process images a few rows at a time. Today, such control is still of interest because it is used in image processing boards implementing a pipelined architecture.

5.6 Detecting Edges using Differencing Masks

Image points of high contrast can be detected by computing intensity differences in local image regions. Typically, such points form the border between different objects or scene parts. In this section, we show how to do this using neighborhood templates or *masks*. We start by using one-dimensional signals: this helps develop both the intuition and formalism, and is also very important in its own right. The 1D signals could just be rows or columns of a 2D image. The section ends by studying more general 2D situations.

Compute output image pixel G[r,c] from neighbors of input image pixel F[r,c].

F[**r**, **c**] is an input image of MaxRow rows and MaxCol columns;
F is unchanged by the algorithm.
G[**r**, **c**] is the output image of MaxRow rows and MaxCol columns.
The border of **G** are all those pixels whose neighborhoods
are not wholly contained in **G**.
w and h are the width and height, in pixels, defining a neighborhood.

```
procedure enhance_image(F,G,w,h);
{
for r := 0 to MaxRow - 1
   for c := 0 to MaxCol - 1
      {
      if [r,c] is a border pixel then G[r,c] := F[r,c];
      else G[r,c] := compute_using_neighbors ( F, r, c, w, h );
      } ;
}
procedure compute_using_neighbors ( IN, r, c, w, h )
{
using all pixels within w/2 and h/2 of pixel IN[r,c],
compute a value to return to represent IN[r,c]
}
```

Algorithm 1: Compute output image pixel G[r,c] from neighbors of input image pixel F[r,c].

Exercise 7

Implement Algorithm 1 in some programming language. Code both the boxcar and median filtering operations and test them on some images such as in Figure 5.9.

5.6.1 Differencing 1D Signals

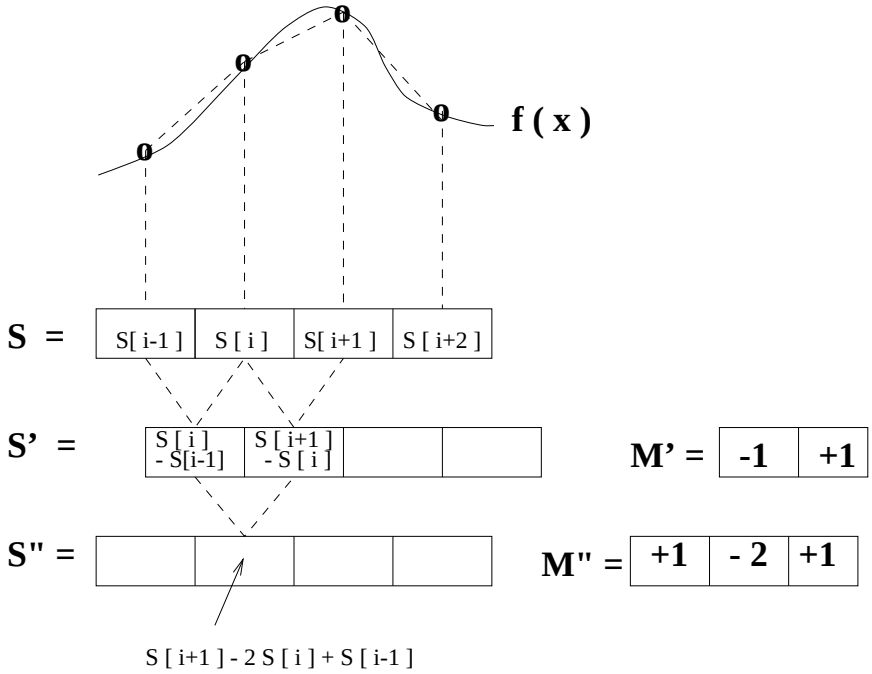


Figure 5.10: (Left) The first (**S'**) and second (**S"**) difference signals are scaled approximations to the first and second derivatives of the signal **S**. (Right) Masks **M'** and **M"** represent the derivative operations.

Figure 5.10 shows how masks can be used to compute representations of the derivatives of a signal. Given that the signal **S** is a sequence of samples from some function f, then $f'(x_i) \approx (f(x_i) - f(x_{i-1})/(x_i - x_{i-1})$. Assuming that the sample spacing is $\Delta x = 1$, the derivative of $f(x)$ can be approximated by applying the mask $\mathbf{M}' = [-\mathbf{1}, \mathbf{1}]$ to the samples in **S** as shown in Figure 5.10 to obtain an output signal **S'**. As the figure shows, it's convenient to think of the values of **S'** as occurring in between the samples of **S**. A high absolute value of **S'**[i] indicates where the signal is undergoing rapid change or *high contrast*. Signal **S'** itself can be differentiated a second time using mask $\mathbf{M}'$ to produce output **S"** which corresponds to the second derivative of the original function f. The important result illustrated in Figure 5.10 and derived in the equations below is that the approximate second derivative can be computed by applying the mask $\mathbf{M}''$ to the original sequence of samples **S**.

$$
\begin{aligned}
S'[i] &= -S[i-1] + S[i] && (5.3)\\
mask\ \mathbf{M}' &= [-1, +1] && (5.4)\\
S''[i] &= -S'[i] + S'[i+1] && (5.5)\\
&= -(S[i] - S[i-1]) + (S[i+1] - S[i]) && (5.6)\\
&= S[i-1] - 2S[i] + S[i+1] && (5.7)
\end{aligned}
$$

$$mask\ \mathbf{M}'' \quad = \quad [1,\ -2,\ 1] \tag{5.8}$$

$$\mathbf{mask\ M\ =\ [-1, 0, 1]}$$

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	12	12	0	0	0	0

(a) S_1 is an upward step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	-12	-12	0	0	0	0

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	$\otimes$	M	0	0	0	3	6	6	6	3	0	0

(c) S_3 is an upward ramp

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	0	0	0	12	0	-12	0	0	0	0

(d) S_4 is a bright impulse or "line"

Figure 5.11: Cross correlation of four special signals with first derivative edge detecting mask $[-1, 0, 1]$; (a) upward step edge, (b) downward step edge, (c) upward ramp, and (d) bright impulse. Note that, since the coordinates of $\mathbf{M}$ sum to zero, output must be zero on a constant region.

If only points of *high contrast* are to be detected, it is very common to use the absolute value after applying the mask at signal position $\mathbf{S[i]}$. If this is done, then the first derivative mask can be either $\mathbf{M}' = [-1, +1]$ or $[+1, -1]$ and the second derivative mask can be either $\mathbf{M}'' = [+1, -2, +1]$ or $[-1, +2, -1]$. A similar situation exists for 2D images as we shall soon see. Whenever only magnitudes are of concern, we will consider these patterns to be the same, and whenever the sign of the change is important, we will consider them to be different.

Use of another common first derivative mask is shown in Figure 5.11. This mask has 3 coordinates and is centered at signal point $\mathbf{S[i]}$ so that it computes the signal difference across the adjacent values. Because $\Delta x = 2$, it will give a high estimate of the actual derivative unless the result is divided by 2. Moreover, this mask is known to give a response on perfect step edges that is 2 samples wide, as is shown in Figure 5.11(a)-(b). Figure 5.12 shows the response of the second derivative mask on the sample signals. As Figure 5.12 shows, signal contrast is detected by a *zero-crossing*, which localizes and amplifies the change between two successive signal values. Taken together, the first and second derivative signals reveal much of the local signal structure. Figure 5.13 shows how smoothing of signals can be put into

the same framework as differencing: the boxed table below compares the general properties of smoothing versus differencing masks.

Some properties of derivative masks follow:

- Coordinates of derivative masks have opposite signs in order to obtain a high response in signal regions of high contrast.
- The sum of coordinates of derivative masks is zero so that a zero response is obtained on constant regions.
- First derivative masks produce high absolute values at points of high contrast.
- Second derviatve masks produce zero-crossings at points of high contrast.

For comparison, smoothing masks have these properties:

- Coordinates of smoothing masks are positive and sum to one so that output on constant regions is the same as the input.
- The amount of smoothing and noise reduction is proportional to the mask size.
- Step edges are blurred in proportion to the mask size.

5.6.2 Difference Operators for 2D Images

Contrast in the 2D picture function $f(x, y)$ can occur in any direction. From calculus, we know that the maximum change occurs along the direction of the gradient of the function, which is in the direction $[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$ in the picture plane. Figure 5.14 shows that this is quite intuitive when one considers discrete approximations in digital images. We can estimate the contrast at image location $I[x, y]$ along the x-direction by computing $(I[x+1, y] - I[x-1, y])/2$, which is the change in intensity across the left and right neighbors of pixel [x,y] divided by $\Delta x = 2$ pixel units. For the neighborhood shown in Figure 5.14, the constrast in the x direction would be estimated as $(64 - 14)/2 = 25$. Since our pixel values are noisy and since the edge might actually cut across the pixel array at any angle, it should help to average 3 different estimates of the contrast in the neighborhood of [x,y]:

$$\begin{aligned} \partial f/\partial x \equiv f_x \;\approx\; & \tfrac{1}{3}\,[\,(I[x+1,y] - I[x-1,y])/2 \\ & +\;(I[x+1,y-1] - I[x-1,y-1])/2 \\ & +\;(I[x+1,y+1] - I[x-1,y+1])/2)\,] \end{aligned} \tag{5.9}$$

This estimates the contrast in the x-direction by equally weighting the contrast across the row y with the row below and above it. The contrast in the y-direction can be estimated similarly:

$$\begin{aligned} \partial f/\partial y \equiv f_y \;\approx\; & \tfrac{1}{3}\,[\,(I[x,y+1] - I[x,y-1])/2 \\ & +\;(I[x-1,y+1] - I[x-1,y-1])/2 \\ & +\;(I[x+1,y+1] - I[x+1,y-1])/2)\,] \end{aligned} \tag{5.10}$$

mask M = $[-1, 2, -1]$

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	0	0	0	0	-12	12	0	0	0	0

(a) S_1 is an upward step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	$\otimes$	M	0	0	0	0	12	-12	0	0	0	0

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	$\otimes$	M	0	0	0	-3	0	0	0	3	0	0

(c) S_3 is an upward ramp

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	0	0	0	-12	24	-12	0	0	0	0

(d) S_4 is a bright impulse or "line"

Figure 5.12: Cross correlation of four special signals with second derivative edge detecting mask $\mathbf{M} = [-\mathbf{1}, \mathbf{2}, -\mathbf{1}]$; (a) upward step edge, (b) downward step edge, (c) upward ramp, and (d) bright impulse. Since the coordinates of $\mathbf{M}$ sum to zero, response on constant regions is zero. Note how a *zero-crossing* appears at an output position where different trends in the input signal join.

box smoothing mask M = [1/3, 1/3, 1/3]

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	12	12	12	12	16	20	24	24	24	24

(a) S_1 is an upward step edge

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	12	12	12	16	16	16	12	12	12	12

(d) S_4 is a bright impulse or "line"

Gaussian smoothing mask M = [1/4, 1/2, 1/4]

S_1			12	12	12	12	12	24	24	24	24	24
S_1	$\otimes$	M	12	12	12	12	15	21	24	24	24	24

(a) S_1 is an upward step edge

S_4			12	12	12	12	24	12	12	12	12	12
S_4	$\otimes$	M	12	12	12	15	18	15	12	12	12	12

(d) S_4 is a bright impulse or "line"

Figure 5.13: (Top two rows) Smoothing of step and impulse with box mask $[1/3, 1/3, 1/3]$; (bottom two rows) smoothing of step and impulse with Gaussian mask $[1/4, 1/2, 1/4]$.

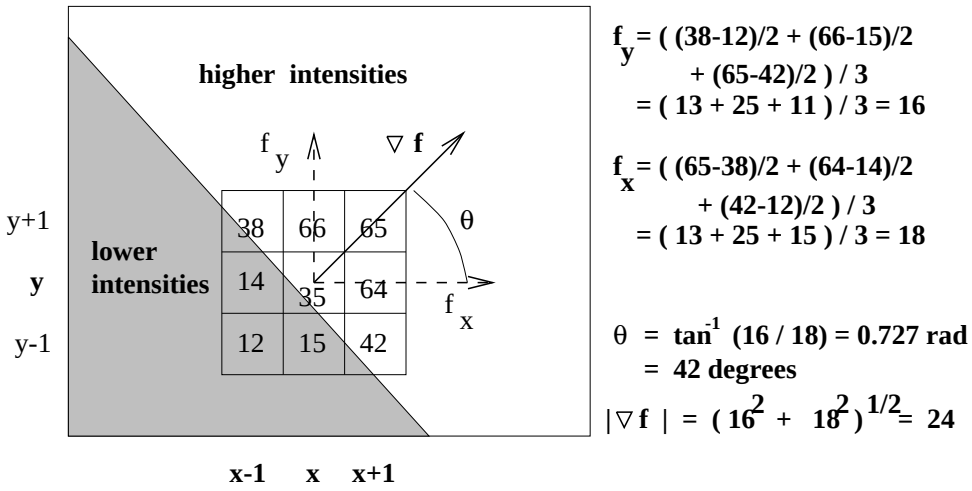


Figure 5.14: Estimating the magnitude and direction of contrast at I[x,y] by estimating the gradient magnitude and direction of picture function $f(x, y)$ using the discrete samples in the image array.

Often, the division by 6 is ignored to save computation time, resulting in scaled estimates. Masks for these two contrast operators are denoted by M_x and M_y at the top of Figure 5.15. The gradient of the picture function can be estimated by applying the masks to the 8-neighborhood $N_8[x, y]$ of pixel $[x, y]$ as given in Equations 5.11 to 5.14. These masks define the *Prewitt operator* credited to Dr Judith Prewitt who used them in detecting boundaries in biomedical images.

$$\frac{\partial f}{\partial x} \approx (1/6)(M_x \circ N_8[x, y]) \tag{5.11}$$

$$\frac{\partial f}{\partial y} \approx (1/6)(M_y \circ N_8[x, y]) \tag{5.12}$$

$$| \nabla f | \approx \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2} \tag{5.13}$$

$$\theta \approx tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}) \tag{5.14}$$

The operation $M \circ N$ is defined formally in the next section: operationally, mask M is overlaid on image neighborhood N so that each intensity N_{ij} can be multiplied by weight M_{ij}; finally all these products are summed. The middle row of Figure 5.15 shows the two analogous *Sobel masks*; their derivation and interpretation is the same as for the Prewitt masks except that the assumption is that the center estimate should be weighted twice as much as the estimate to either side.

The *Roberts masks* are only 2x2; hence they are both more efficient to use and more locally applied. Often referred to as *the Roberts cross operator*, these masks actually compute a gradient estimate at the center of a 4-neighborhood and not at a center pixel. Moreover, the actual coordinate system in which the operator is defined is rotated 45° off the standard row direction. Application of the Roberts cross operator is shown in Figure 5.16:

Prewitt: $M_x =$ | -1 | 0 | 1 | / | -1 | 0 | 1 | / | -1 | 0 | 1 | ; $M_y =$ | 1 | 1 | 1 | / | 0 | 0 | 0 | / | -1 | -1 | -1 |

Sobel: $M_x =$ | -1 | 0 | 1 | / | -2 | 0 | 2 | / | -1 | 0 | 1 | ; $M_y =$ | 1 | 2 | 1 | / | 0 | 0 | 0 | / | -1 | -2 | -1 |

Roberts: $M_x =$ | 0 | 1 | / | -1 | 0 | ; $M_y =$ | 1 | 0 | / | 0 | -1 |

Figure 5.15: 3x3 masks used to estimate the gradient of picture function $f(x, y)$: (top row) Prewitt; (middle row) Sobel; (bottom row) Roberts.

the original input image is shown in the upper left, while the highest 10% of the responses of the operator are shown at the lower left. Qualitatively, these results are typical of the several small-neighborhood operators – many pixels on many edges are detected but some gaps exist. Also, there are strong responses on the highly textured regions corresponding to plants. The Roberts results should be compared to the results obtained by combining the simple 1D row and column masks shown in Figure 5.16(b-d) and (f-h). It is common to avoid computing a square root in order to compute gradient magnitude; alternatives are $max(\mid \frac{\partial f}{\partial x} \mid, \mid \frac{\partial f}{\partial y} \mid)$, $\mid \frac{\partial f}{\partial x} \mid + \mid \frac{\partial f}{\partial y} \mid$, or $(\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2)/2$. With these estimates, one must be careful when trying to interpret the actual gradient or gradient direction. Figure 5.17(b) shows the results of using the Sobel 3x3 operator to compute mean square gradient magnitude and Figure 5.17(c) shows an encoding of the gradient direction. The small squares in the original image are 8x8 pixels: the Sobel operator represents many, but not all, of the image edges.

Exercise 8

If a true gradient magnitude is needed, why might the Sobel masks provide a faster solution than the Prewitt masks?

Exercise 9 *Optimality of Prewitt masks

The problem is to prove that Prewitt masks give the weights for the best fitting plane approximating the intensity surface in a 3×3 neighborhood, assuming all 9 samples have equal weight. Suppose that the 9 intensities $I[r + i, c + j]; i, j = -1, 0, 1$ of a 3×3 image neighborhood are fit by the least squares planar model $I[r, c] = z = pr + qc + z_0$. (Recall that the 9 samples are equally spaced in terms of r and c.) Show that the Prewitt masks compute the estimates of p and q as the partial derivatives of the least squares planar fit of the intensity function.

Figure 5.18 (top) shows plots of the intensites of two rows of the room scene at the left. The lower row cuts across four dark regions as seen in the image and the plot; (1) the coat on the chair at the left (columns 20 to 80), (2) Dr Prewitt's chair and dress in the

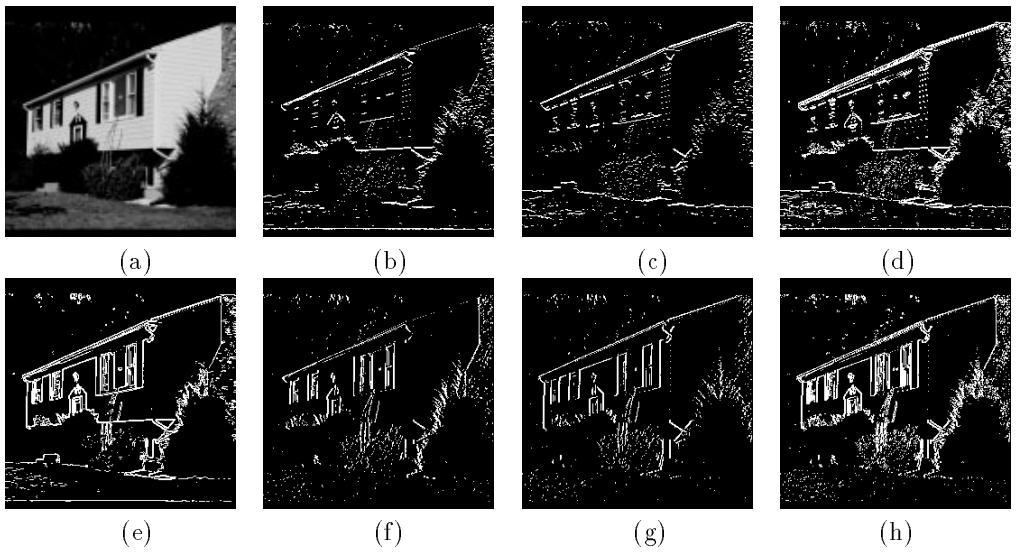
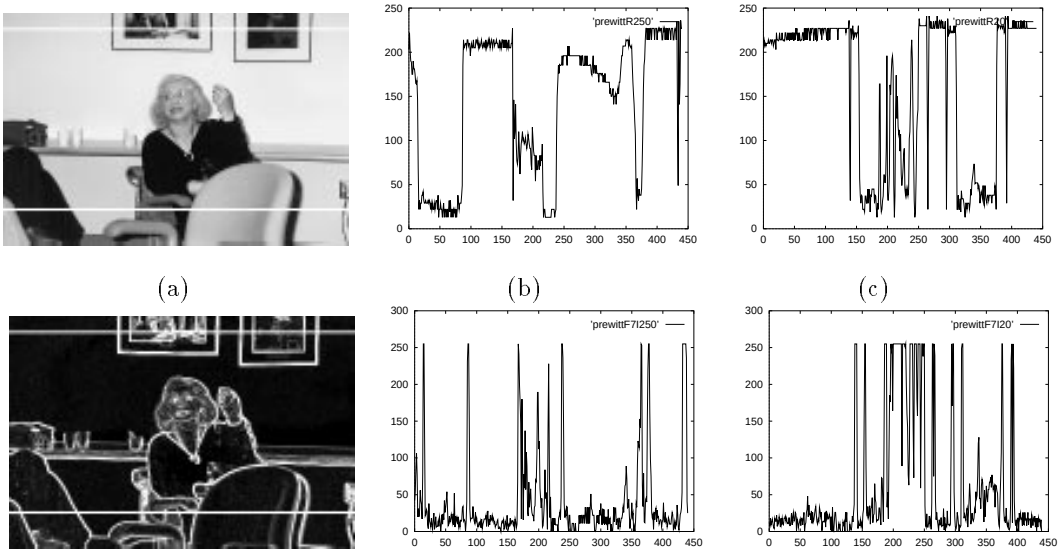


Figure 5.16: (a) Original house image; (b) top 5% of the responses from horizontal edge mask $[+1, -1]^{\ t}$; (c) top 5% of the responses from horizontal edge mask $[-1, +1]^{\ t}$; (d) images of (b) and (c) ORed together; (e) top 10% of the sum of the absolute values of the two Roberts mask responses; (f) top 5% of the responses from the vertical edge mask $[-1, +1]$; (g) top 5% of the responses from the vertical edge mask $[+1, -1]$; (h) images (f) and (g) ORed together.

center (columns 170 to 240), (3) the shadow of the rightmost chair (columns 360 to 370) and (4) the electric wires (column 430). Note that the transitions between dark and bright are sharp except for the boundary between the chair and its shadow, which ramps down from brightness 220 to 20 over about 10 pixels. The upper row profile, shown at the top right, shows sharp transitions where it cuts across the picture frames, mat, and pictures. The picture at the left shows much more intensity variation than the one at the right. The bottom row of Figure 5.18 shows the results of applying the 3x3 Prewitt gradient operator to the original image. The sum of the absolute values of the column and row gradients f_x and f_y is plotted for the same two image rows shown at the top of the figure. The highest values of the Prewitt operator correspond well with the major boundaries crossed; however, the several medium level spikes from Dr Prewitt's chair between columns 170 and 210 are harder to interpret. The contrasts in the upper row, graphed at the bottom right, are interpeted similarly – major object boundaries correspond well to the object boundaries of the picture frame and mat; however, there is a lot of intensity variation in the leftmost picture. Generally, gradient operators work well for detecting the boundary of isolated objects, although some problems are common. Boundaries sometimes drop out due to object curvature or soft shadows: on the other hand, good contrast often produces boundaries that are several pixels wide, necessitating a thinning step afterward. Gradient operators will also respond to textured regions, as we shall study in more detail in Chapter 7.

(a) (b) (c)

Figure 5.17: (a) Image of noisy squares and rings; (b) mean square response of 3x3 Sobel operator; (c) coding of gradient direction computed by 3x3 Sobel operator.



(a) (b) (c)

(d) (e) (f)

Figure 5.18: (a) Image of Judith Prewitt with two rows selected; (b) plot of intensities along selected lower row; (c) plot of intensities along selected upper row; (d) "gradient image" showing result of $\mid f_x \mid + \mid f_y \mid$ using the Prewitt 3x3 operator; (e) plot of selected lower row of gradient image; (f) plot of selected upper row of gradient image.

5.7 Gaussian Filtering and LOG Edge Detection

The Gaussian function has important applications in many areas of mathematics, including image filtering. In this section, we highlight the characteristics that make it useful for smoothing images or detecting edges after smoothing.

8 DEFINITION *A* **Gaussian function** *of one variable with spread σ is of the following form, where c is some scale factor.*

$$g(x) = ce^{-\frac{x^2}{2\sigma^2}} \tag{5.15}$$

A Gaussian function of two variables is

$$g(x,y) = ce^{-\frac{(x^2+y^2)}{2\sigma^2}} \tag{5.16}$$

These forms have the same structure as the normal distribution defined in Chapter 4, where the constant c was set so that the area under the curve would be 1. To create a mask for filtering, we usually make c a large number so that all mask elements are integers. The Gaussian is defined centered on the origin and thus needs no location parameter μ as does the normal distribution: an image processing algorithm will translate it to wherever it will be applied in a signal or image. Figure 5.19 plots the Gaussian of one variable along with its first and second derivatives, which are also important in filtering operations. The derivation of these functions is given in Equations 5.17 to 5.22. The area under the function $g(x)$ is 1, meaning that it is immediately suitable as a smoothing filter that does not affect constant regions. $g(x)$ is a positive even function; $g'(x)$ is just $g(x)$ multiplied by odd function $-x$ and scaled down by σ^2. More structure is revealed in $g''(x)$. Equation 5.21 shows that $g''(x)$ is the difference of two even functions and that the central lobe will be negative with $x \approx 0$. Using Equation 5.22, it is clear that the zero crossings of the second derviative occur at $x = \pm\sigma$, in agreement with the plots in Figure 5.19.

$$g(x) = \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{5.17}$$

$$g'(x) = \frac{-1}{\sqrt{2\pi}\ \sigma^3} x e^{-\frac{x^2}{2\sigma^2}} \tag{5.18}$$

$$= \frac{-x}{\sigma^2}\ g(x) \tag{5.19}$$

$$g''(x) = \left(\frac{x^2}{\sqrt{2\pi}\ \sigma^5} - \frac{1}{\sqrt{2\pi}\ \sigma^3}\right)\ e^{-\frac{x^2}{2\sigma^2}} \tag{5.20}$$

$$= \frac{x^2}{\sigma^4}\ g(x)\ -\ \frac{1}{\sigma^2}\ g(x) \tag{5.21}$$

$$= \left(\frac{x^2}{\sigma^4}\ -\ \frac{1}{\sigma^2}\right)\ g(x) \tag{5.22}$$

Understanding the properties of the 1D Gaussian, we can now intuitively create the corresponding 2D function $g(x,y)$ and its derivatives by making the substitution $r = \sqrt{x^2+y^2}$. This creates the 2D forms by just spinning the 1D form about the vertical axis yielding *isotropic* functions which have the same 1D Gaussian cross section in any cut through the origin. The second derivative form is well known as a *sombrero* or *Mexican hat*. From

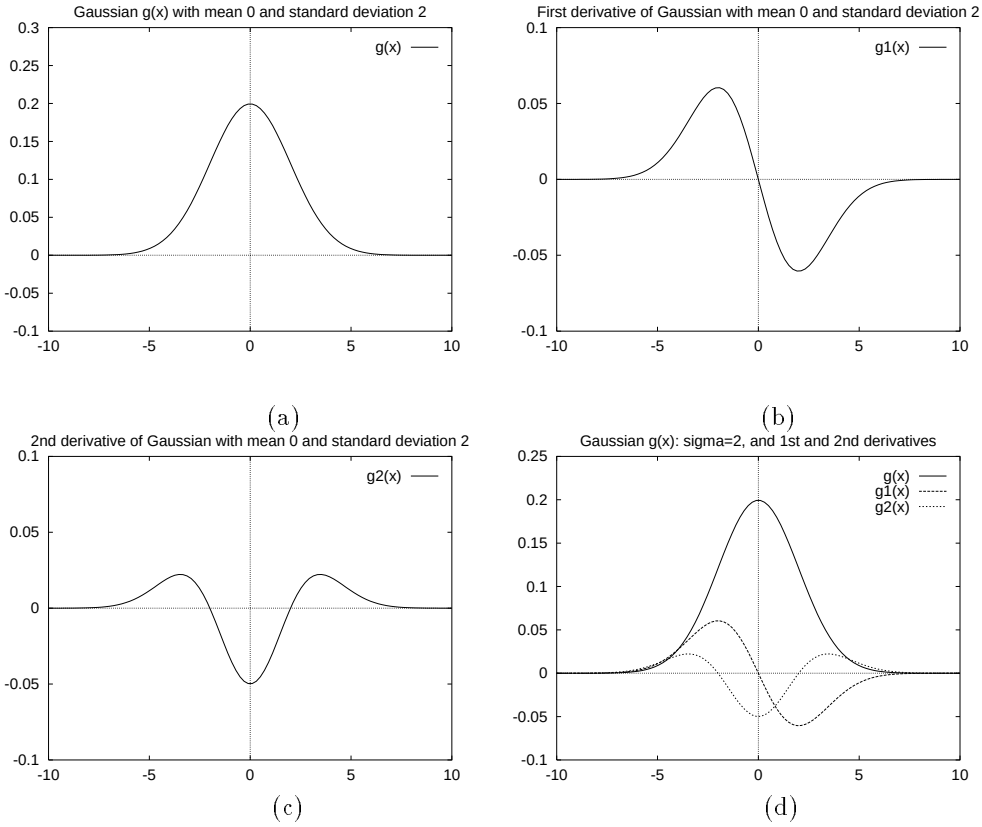


Figure 5.19: (a) Gaussian $g(x)$ with spread $\sigma = 2$; (b) first derivative $g'(x)$; (c) second derivative $g''(x)$, which looks like the cross section of a sombrero upside down from how it would be worn; (d) all three plots superimposed to show how the extreme slopes of $g(x)$ align with the extremas of $g'(x)$ and the zero crossings of $g''(x)$.

the mathematical derivations, the cavity for the head will be pointed upward along the $z = g(x, y)$ axis; however, it is usually displayed and used in filtering with the cavity pointed downward, or equivalently, with the center lobe positive and the rim negative.

$$\mathbf{G_{3\times 3}} = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}\ ; \qquad \mathbf{G_{7\times 7}} = \begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 3 & 7 & 9 & 7 & 3 & 1 \\ \hline 3 & 12 & 26 & 33 & 26 & 12 & 3 \\ \hline 7 & 26 & 55 & 70 & 55 & 26 & 7 \\ \hline 9 & 33 & 70 & 90 & 70 & 33 & 9 \\ \hline 7 & 26 & 55 & 70 & 55 & 26 & 7 \\ \hline 3 & 12 & 26 & 33 & 26 & 12 & 3 \\ \hline 1 & 3 & 7 & 9 & 7 & 3 & 1 \\ \hline \end{array}$$

Figure 5.20: (Left) A 3×3 mask approximating a Gaussian obtained by matrix multiplication $[1,2,1]^t \otimes [1,2,1]$; (Right) a 7×7 mask approximating a Gaussian with $\sigma^2 = 2$ obtained by using Equation 5.16 to generate function values for integers x and y and then setting $c = 90$ so that the smallest mask element is 1.

Some Useful Properties of Gaussians

1. Weight decreases smoothly to zero with distance from the origin, meaning that image values nearer the central location are more important than values that are more remote; moreover, the spread parameter σ determines how broad or focused the neighborhood will be. 95% of the total weight will be contained within 2σ of the center.

2. Symmetry about the abscissa; flipping the function for convolution produces the same kernel.

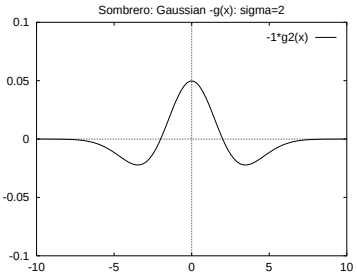
3. Fourier transformation into the frequency domain produces another Gaussian form, which means convolution with a Gaussian mask in the spatial domain reduces high frequency image trends smoothly as spatial frequency increases.

4. The second derivative of a 1D Gaussian $g''(x)$ has a smooth center lobe of negative area and two smooth side lobes of positive area: the zero crossings are located at $-\sigma$ and $+\sigma$, corresponding to the inflection points of $g(x)$ and the extreme points of $g'(x)$.

5. A second derivative filter based on the Laplacian of the Gaussian is called a **LOG filter**. A LOG filter can be approximated nicely by taking the difference of two Gaussians $g''(x) \approx c_1 e^{-\frac{x^2}{2\sigma_1{}^2}} - c_2 e^{-\frac{x^2}{2\sigma_2{}^2}}$, which is often called a **DOG** filter (for **D**ifference **O**f **G**aussians). For a positive center lobe, we must have $\sigma_1 < \sigma_2$; also, σ_2 must be carefully related to σ_1 to obtain the correct location of zero crossings and so that the total negative weight balances the total positive weight.

6. The LOG filter responds well to intensity differences of two kinds – small blobs coinciding with the center lobe, and large step edges very close to the center lobe.

Two different masks for Gaussian smoothing are shown in Figure 5.20. Masks for edge detection are given below.



0	-1	0
-1	4	-1
0	-1	0

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

Figure 5.21: (Top row) Cross section of the LOG filter and a 3×3 mask approximating it; (bottom row) input image and result of applying the mask to it.

5.7.1 Detecting Edges with the LOG Filter

Two different masks implementating the LOG filter are given in Figures 5.21 and 5.22. The first is 3×3 mask: the smallest possible implementation, detects image details nearly the size of a pixel. The 11×11 mask computes a response by integrating the input of 121 pixels and thus responds to larger image features and not smaller ones. Integrating 121 pixels can take a lot more time than integrating 9 of them done in software.

Exercise 10 Properties of the LOG filter

Suppose that the 9 intensities of a 3×3 image neighborhood are perfectly fit by the planar model $I[r,c] = z = pr + qc + z_0$. (Recall that the 9 samples are equally spaced in terms of r and c.) Show that the simple LOG mask

0	-1	0
-1	4	-1
0	-1	0

has zero response on such a neighborhood. This means that the LOG filter has zero response on both constant regions and ramps.

5.7.2 On Human Edge-Detection

We now describe an artificial neural network (ANN) architecture which implements the LOG filtering operation in a highly parallel manner. The behavior of this network has been shown

0	0	0	-1	-1	-2	-1	-1	0	0	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	-2	-7	-15	-22	-23	-22	15	-7	-2	0
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-2	-9	-23	-1	103	178	103	-1	-23	-9	-2
-1	-8	-22	-14	52	103	52	-14	-22	-8	-1
-1	-4	-15	-24	-14	-1	-14	-24	-15	-4	-1
0	-2	-7	-15	-22	-23	-22	15	-7	-2	0
0	0	-2	-4	-8	-9	-8	-4	-2	0	0
0	0	0	-1	-1	-2	-1	-1	0	0	0

Figure 5.22: An 11×11 mask approximating the Laplacian of a Gaussian with $\sigma^2 = 2$. (From Harlalick and Shapiro, Volume I, page 349.)

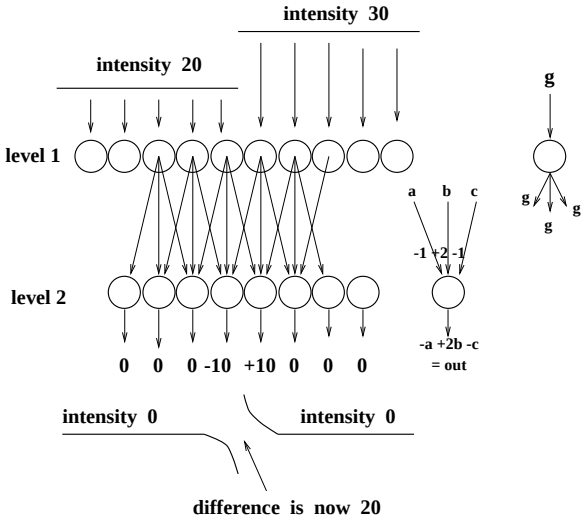


Figure 5.23: Producing the *Mach band effect* using an ANN architecture. Intensity is sensed by cells of the retina (level 1) which then stimulate integrating cells in a higher level layer (level 2).

Figure 5.24: Seven constant stripes generated with grey levels $31 + 32k$, $k = 1, 7$. Due to the Mach band effect, humans perceive scalloped, or concave, panels.

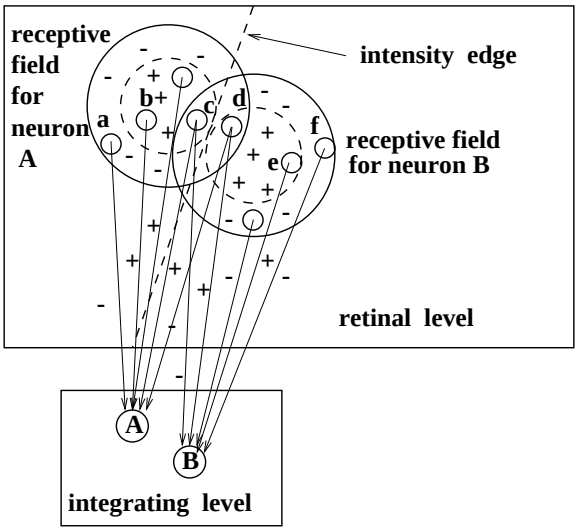


Figure 5.25: A 3D ANN architecture for LOG filtering.

to simulate some of the known behaviors of the human visual system. Moreover, invasive techniques have also shown that the visual systems of cats and monkeys produce electrical signals consistent with the behavior of the neural network architecture. Figure 5.23 shows the processing of 1D signals. A step edge is sensed at various points by cells of the retinal array. These level 1 cells stimulate the integrating cells at level 2. Each physical connection between level 1 cell i and level 2 cell j has an associated weight w_{ij} which is multiplied by the stimulus being communicated before it is summed in cell j. The output of cell j is $y_j = \sum_{i=1}^{N} w_{ij} x_i$ where x_i is the output of the i-th first level cell and N is the total number of first level cells. (Actually, we need only account for those cells i directly connected to the second level cell j). By having the weights associated with each connection, it is possible, and common, to have the same cell i give positive input to cell j and negative input to cell $k \neq j$. Figure 5.23 shows that each cell j of the second level computes its output as $-a + 2b - c$, corresponding to the mask $[-1, 2, -1]$: the weight 2 is applied to the central input, while the (inhibitory) inputs a and b are each weighted by -1.

This kind of architecture can be defined for any kind of mask and permits a highly parallel implementation of cross correlation for filtering or feature detection. The psychologist Mach noted that humans perceive an edge between two regions as if it were pulled apart to exaggerate the intensity difference, as is done in Figure 5.23. Note that this architecture and mask create the *zero crossing* at the location of the edge between two cells, one of which produces positive output and the other negative output. The Mach band effect changes the perceived shape of joining surfaces and is evident in computer graphics systems that display polyhedral objects via shaded faces. Figure 5.24 shows seven constant regions, stepping from grey level 31 to 255 in steps of 32. Do you perceive concave panels in 3D such as on a Doric column from a Greek temple?

Figure 5.25 extends Figure 5.23 to 2D images. Each set of retinal cells connected to integrating cell j comprise what is called the *receptive field* of that cell. To perform edge detection via a second derivative operation, each receptive field has a center set of cells

which have positive weights w_{ij} with respect to cell j and a surrounding set of cells with negative weights. Retinal cells b and c are in the center of the receptive field of integrating cell A, whereas retinal cells a and d are in the surround and provide inhibitory input. Retinal cell d is in the center of the receptive field of integrating cell B, however, and cell c is in its surround. The sum of the weights from the center and the surround should be zero so that the integrating cell has neutral output on constant regions. Because the center and surround are circular, output will not be neutral whenever a straight region boundary just nips the center area, regardless of the angle. Thus, each integrating cell is an *isotropic* edge detector cell. Additionally, if a small region contrasting with the background images within the center of the receptive field the integrating cell will also respond, making it a spot detector as well. Figure 5.21 shows the result of convolving the smallest of the LOG masks with an image containing two regions. The result at the right of the figure shows how the operation determines the boundary between the regions via zero crossings. An 11×11 mask corresponding to the Laplacian of a Gaussian with $\sigma^2 = 2$ is shown in Figure 5.22. The tiny mask is capable of finding the boundary between tiny regions and is sensitive to high curvature bouundaries but will also respond to noise texture. The larger mask performs a lot of smoothing and will only respond to boundaries between larger regions with smoother perimeter.

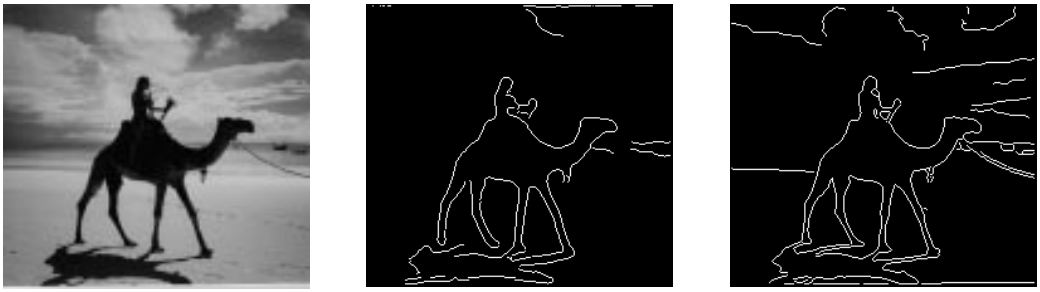
Exercise 11

Give more detailed support to the above arguments that the integrating cells shown in Figure 5.25 (a) respond to contrasting spots or blobs that image within the center of the field and (b) respond to boundaries between two large regions that just barely cross into the center of the field.

5.7.3 Marr-Hildreth Theory

David Marr and Ellen Hildreth proposed LOG filtering to explain much of the low level behavior of human vision. Marr proposed that the objective of low level human visual processing was the construction of a *primal sketch* which was a 2D description containing lines, edges, and blobs. (The primal sketches derived from the two eyes would then be processed further to derive 3D interpretations of the scene.) To derive a primal sketch, Marr and Hildreth proposed an organization based on LOG filtering with 4 or 5 different spreads σ. The mathematical properties outlined above explained the results of both perceptual experiments on humans and invasive experiments with animals. LOG filters with large σ would detect broad edges while those with small σ would focus on small detail. Coordination of the output from the different scales could be done at a higher level, perhaps with the large scale detections guiding those at the small scale. Subsequent work has produced different practical *scale space* methods of integrating output from detectors of different sizes.

Figure 5.26 shows an image processed at two different levels of Gaussian smoothing. The center image shows good representation of major objects and edges, whereas the rightmost image shows both more image detail and more noise. Note that the ship and part of the sand/water boundary is represented in the rightmost image but not in the center image. Marr's primal sketch also contained descriptions of *virtual lines*, which are formed by similar detected features organized along an image curve. These might be the images of a dotted line, a row of shrubs, etc. A synthetic image containing a virtual line is shown in Figure 5.27



original smoothed $\sigma = 4$ smoothed $\sigma = 1$

Figure 5.26: An input image (a) is smoothed using Gaussian filters of size (b) $\sigma = 4$ and (c) $\sigma = 1$ before performing edge detection. More detail and more noise is shown for the smaller filter (photo by David Shaffer 1998).

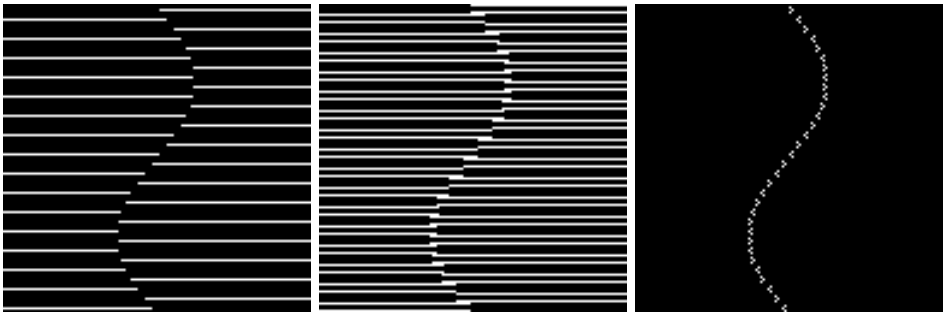


Figure 5.27: (Left) A virtual line formed by the termination points of a line texture – perhaps this is two pieces of wrapping paper overlaid; (center) output from a specific 4×4 LOG filter responds to both lines and endpoints; (right) a different 3×3 LOG filter responds only to the endpoints.

along with the output of two different LOG filters. Both LOG filters give a response to the ends of the stripes; one is sensitive to the edges of the stripes as well, but the other is not. Figure 5.28 shows the same principle in a real image that was thresholded to obtain an artistic texture. Recent progress in researching the human visual system and brain has been rapid: results seem to complicate the interpretations of earlier work on which Marr and Hildreth based their mathematical theory. Nevertheless, use of variable-sized Gaussian and LOG filters is firmly established in computer vision.

5.8 The Canny Edge Detector

The Canny edge detector is a very popular and effective operator, and it is important to introduce it here, although details are placed in Chapter 10. The Canny operator first smoothes the intensity image and then produces extended contour segments by following high gradient magnitudes from one neighborhood to another. Figure 5.29 shows edge detection in fairly difficult outdoor images. The contours of the arch of St. Louis are detected

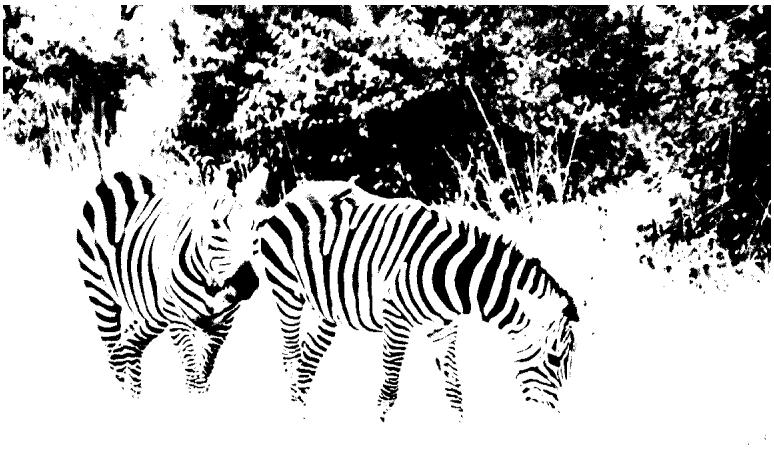


Figure 5.28: Picture obtained by thresholding: object boundaries are formed by virtual curves formed by the ends of stripes denoting cross sections of generalized cylinders. (Original photo by Eleanor Harding.

quite well in Figure 5.29: using a parameter of $\sigma = 1$ detects some of the metal seams of the arch as well as some internal variations in the tree, whereas use of $\sigma = 4$ detects only the exterior boundaries of these objects. As shown in the bottom row of Figure 5.29, the operator isolates many of the *texture elements* of the checkered flags. For comparison, use of the Roberts operator with a low threshold on gradient magnitude is shown: more texture elements of the scene (grass and fence) are evident, although they have less structure than those in the Canny output. The algorithm for producing contour segments is treated in detail in Chapter 10.

5.9 *Masks as Matched Filters

Here we give a theoretical basis for the concept that the response of a mask to a certain image neighborhood is proportional to how similar that neighborhood is to the mask. The important practical result of this is that we now know how to design masks to detect specific features – we just design a mask that is similar to the feature[s] we want to detect. This will serve us well in both edge and texture detection and also in detecting other special patterns such as holes or corners. We introduce the concepts using 1-dimensional signals, which are important in their own right, and which might correspond to rows or columns or any other cut through a 2D image. The concepts and mathematical theory immediately extend to the 2D case.

5.9.1 The Vector Space of all Signals of n Samples

For a given $n \geq 1$, the set of all vectors of n real coordinates forms a *vector space*. The practical and powerful vector space operations which we use are summarized below. The reader has probably already worked with vectors with $n = 2$ or $n = 3$ when studying

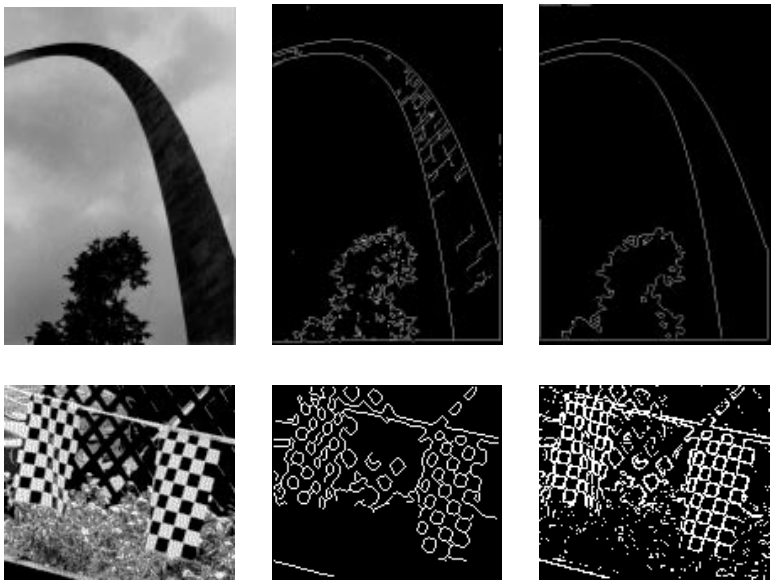


Figure 5.29: (Top left) Image of the great arch at St Louis; (top center) results of Canny operator with $\sigma = 1$; (top right) results of Canny operator with $\sigma = 4$; (bottom left) image with textures; (bottom center) results of Canny operator with $\sigma = 1$; (bottom right) results of Roberts operator thresholded to pass the top 20% of the pixels in gradient magnitude.

analytical geometry or calculus. For $n = 2$ or $n = 3$, the notion of vector length is the same as that used in plane geometry and 3D analytical geometry since Euclid. In the domain of signals, length is related to *energy* of the signal, defined as the squared length of the signal, or equivalently, just the sum of the squares of all coordinates. Signal energy is an extremely useful concept as will be seen.

9 DEFINITION *The* **energy** *of signal* $S = [s_1, s_2, ..., s_n]$ *is* $\| S \|^2 = {s_1}^2 + {s_2}^2 + \ldots + {s_n}^2$.

Note that in many applications, the full range of real-valued signals do not arise because negative values are impossible for the coordinates. For example, a 12-dimensional vector recording the rainfall at a particular location for each of the 12 months should have no negative coordinates. Similarly, intensities along an image row are commonly kept in a non-negative integer range. Nevertheless, the vector space interpretation is still useful, as we shall see. Often, the mean signal value is subtracted from all coordinates in making an interpretation, and this shifts about half of the coordinate values below zero. Moreover, it is quite common to have negative values in masks, which are templates or models of the shape of parts of signals.

Basic definitions for a vector space with a defined vector length.

Let U and V be any two vectors; u_i and v_i be real numbers denoting the coordinates of these vectors; and let a, b, c, etc. be any real numbers denoting scalars.

10 DEFINITION *For vectors* $U = [u_1, u_2, ..., u_n]$ *and* $V = [v_1, v_2, ...v_n]$ *their* **vector sum** *is the vector* $U \oplus V = [u_1 + v_1, u_2 + v_2, ..., u_n + v_n]$.

11 DEFINITION *For vector* $V = [v_1, v_2, ...v_n]$ *and real number (scalar)* a *the* **product of the vector and scalar** *is the vector* $aV = [av_1, av_2, ..., av_n]$.

12 DEFINITION *For vectors* $U = [u_1, u_2, ..., u_n]$ *and* $V = [v_1, v_2, ..., v_n]$ *their* **dot product, or scalar product** *is the real number* $U \circ V = u_1v_1 + u_2v_2 + ... + u_nv_n$.

13 DEFINITION *For vector* $V = [v_1, v_2, ...v_n]$ *its* **length, or norm**, *is the nonnegative real number* $\| V \| = V \circ V = (v_1v_1 + v_2v_2 + ... + v_nv_n)^{1/2}$.

14 DEFINITION *Vectors* U *and* V *are* **orthogonal** *if and only if* $U \circ V = 0$.

15 DEFINITION *The* **distance between vectors** $U = [u_1, u_2, ..., u_n]$ *and* $V = [v_1, v_2, ...v_n]$ *is the length of their difference* $d(U, V) = \| U - V \|$.

16 DEFINITION *A* **basis** *for a vector space of dimension* n *is a set of* n *vectors* $\{ w_1, w_2, ..., w_n \}$ *that are independent and that span the vector space. The spanning property means that any vector* V *can be expressed as a linear combination of basis vectors:* $V = a_1w_1 \oplus a_2w_2 \oplus ... \oplus a_nw_n$. *The independence property means that none of the basis vectors* w_i *can be represented as a linear combination of the others.*

Properties of vector spaces that follow from the above definitions.

1. $U \oplus V = V \oplus U$
2. $U \oplus (V \oplus W) = (U \oplus V) \oplus W$
3. There is a vector O such that for all vectors V, $O \oplus V = V$
4. For every vector V, there is a vector $(-1)V$ such that $V \oplus (-1)V = O$
5. For any scalars a, b and any vector V, $a(bV) = (ab)V$
6. For any scalars a, b and any vector V, $(a + b)V = aV \oplus bV$
7. For any scalar a and any vectors U, V, $a(U \oplus V) = aU \oplus aV$
8. For any vector V, $1V = V$
9. For any vector V, $(-1V) \circ V = -\| V \|^2$

Exercise 12
Choose any 5 of the 9 listed properties of vector spaces and show that each is true.

5.9.2 Using an Orthogonal Basis

Two of the most important results of the study of vector spaces are that (1) every vector can be expressed in only one way as a linear combination of the basis vectors, and (2) any set of basis vectors must have exactly n vectors. Having an *orthogonal basis* allows us a much stronger interpretation of the representation of any vector V as is shown in the following example.

Example of representing a signal as a combination of basis signals.

Consider the vector space of all $n = 3$ sample signals $[v_1, v_2, v_3]$. Represented in terms of the *standard basis*, any vector $V = [v_1, v_2, v_3] = v_1[1,0,0] \oplus v_2[0,1,0] \oplus v_3[0,0,1]$. The standard basis vectors are orthogonal and have unit length; such a set is said to be *orthonormal*. We now study a different set of basis vectors $\{w_1, w_2, w_3\}$, where $w_1 = [-1,0,1]$, $w_2 = [1,1,1]$, and $w_3 = [-1,2,-1]$. Any two of the basis vectors are orthogonal, since $w_i \circ w_j = 0$ for $i \neq j$ (show this). Scaling these to unit length, yields the new basis $\{\frac{1}{\sqrt{2}}[-1,0,1], \frac{1}{\sqrt{3}}[1,1,1], \frac{1}{\sqrt{6}}[-1,2,-1]\}$.
Now represent the signal $S = [10,15,20]$ in terms of the orthogonal basis.
$[10,15,20]$ is relative to the *standard basis*.

$$\begin{aligned}
S \circ w_1 &= \frac{1}{\sqrt{2}}(-10+0+20) \\
S \circ w_2 &= \frac{1}{\sqrt{3}}(10+15+20) \\
S \circ w_3 &= \frac{1}{\sqrt{6}}(-10+30-20) \\
S &= (S \circ w_1)w_1 \oplus (S \circ w_2)w_2 \oplus (S \circ w_3)w_3 \\
S &= (10/\sqrt{2})w_1 \oplus (45/\sqrt{3})w_2 \oplus 0w_3 \\
\| S \|^2 &= 100 + 225 + 400 = 725 \\
&= (10/\sqrt{2})^2 + (45/\sqrt{3})^2 + 0^2 = 725
\end{aligned} \tag{5.23}$$

The last two equations show that when an orthonormal basis is used, it is easy to account for the total energy by summing the energy associated with each basis vector.

The example above showed how to represent the signal $[10,15,20]$ in terms of a basis of three given vectors $\{w_1, w_2, w_3\}$, which have special properties as we have seen. In general, let any signal $S = [a_1, a_2, a_3] = a_1 w_1 \oplus a_2 w_2 \oplus a_3 w_3$. Then $S \circ w_i = a_1(w_1 \circ w_i) \oplus a_2(w_2 \circ w_i) \oplus a_3(w_3 \circ w_i) = a_i(w_i \circ w_i) = a_i$, since $w_i \circ w_j$ is 0 when $i \neq j$ and 1 when $i = j$. So, it is very convenient to have an orthonormal basis: we can easily account for the energy in a signal separately in terms of its energy components associated with each basis vector. Suppose we repeat the above example using the signal $S_2 = [-5,0,5]$, which can be obtained by subtracting the mean signal value of S; $S_2 = S \oplus (-1[15,15,15])$. S_2 is the same

as $S \circ w_1$ since the component along $[1,1,1]$ has been removed. S_2 is just a scalar multiple of w_1: $S_2 = (10/\sqrt{2})w_1 = (10/\sqrt{2})((1/\sqrt{2}))[-1,0,1] = [-5,0,5]$ and we will say that S_2 has the same pattern as w_1. If w_1 is taken to be a filter, then it matches the signal S_2 very well; in some sense, it also matches the signal S very well. We will explore this idea further, but before moving on, we note that there are many different useful orthonormal bases for an n-dimensional space of signal vectors.

Exercise 13

(a) Following the boxed example above, represent the vector $[10,14,15]$ in terms of the basis $\{\frac{1}{\sqrt{2}}[-1,0,1],\ \frac{1}{\sqrt{3}}[1,1,1],\ \frac{1}{\sqrt{6}}[-1,2,-1]\}$. (b) Now represent $[10,19,10]$: to which basis vector is it most similar? Why?

From the properties of vectors and the dot product, the *Cauchy-Schwartz Inequality* of Equation 5.24 is obtained. Its basic meaning is that the dot product of unit vectors must lie between -1 and 1. Thus we have a ready measure to determine the similarity between two vectors: note that if $U = V$, then $+1$ is obtained and if $U = -V$, then -1 is obtained. The *normalized dot product* is used to define the *angle between two vectors*. This angle is the same angle that one can compute in 2D or 3D space using trigonometry. For $n \geq 3$ the angle, or its cosine, is taken to be an abstract measure of similarity between two vectors; if their normalized dot product is 0, then they are dissimilar; if it is 1, then they are maximally similar scaled versions of each other; if it is -1, then one is a negative scaled version of the other, which may or may not be similar, depending on the problem domain.

5.9.3 Cauchy-Schwartz inequality

$$\textit{For any two nonzero vectors } U \textit{ and } V,\ -1 \leq \frac{U \circ V}{\| U \| \| V \|} \leq +1 \qquad (5.24)$$

17 DEFINITION *Let U and V be any two nonzero vectors, then the* **normalized dot product of** *U and V is defined as* $(\frac{U \circ V}{\|U\| \|V\|})$

18 DEFINITION *Let U and V be any two nonzero vectors, then the* **angle between** *U and V is defined as* $cos^{-1}(\frac{U \circ V}{\|U\| \|V\|})$

Exercise 14

Sketch the following five vectors and compute the normalized dot product, or *cos* of the angle between each pair of the vectors: $[5,5]$, $[10,10]$, $[-5,5]$, $[-5,-5]$, $[-10,10]$. Which pairs are perpendicular? Which pairs have the same direction? Which have opposite directions? Compare the relative directions to value of the normalized dot product.

5.9.4 The Vector Space of $m \times n$ Images

The set of all $m \times n$ matrices with real-valued elements is a *vector space* of dimension $m \times n$. Here we interpret the vector space theory in terms of masks and image regions and show

how it applies. In this section, our model of an image is that of an image function over a discrete domain of $m \times n$ sample points $I[x, y]$. We work mainly with 2×2 and 3×3 matrices, but everything easily generalizes to images or masks of any size.

5.9.5 A Roberts basis for 2×2 neighborhoods

The structure of a 2×2 neighborhood of an intensity image can be interpreted in terms of the basis shown in Figure 5.30, which we shall call *the Roberts basis*. Two of the four basis vectors were shown in Figure 5.15. As the exercise below shows, any 2×2 neighborhood of real intensity values can be expressed uniquely as a sum of these four basis vectors, each scaled as needed. The relative size of the scale factor directly indicates the amount of similarity between the image neighborhood and that basis vector and thus can be used to interpret the neighborhood structure. Several examples are given in Figure 5.30.

Exercise 15

Verify that the Roberts basis vectors shown in Figure 5.30 are orthonormal.

Exercise 16

Consider the vector space of all 2x2 images with real-valued pixels. (a) Determine the values of the a_j so that the image $\begin{array}{|c|c|}\hline 10 & 5 \\ \hline 5 & 0 \\ \hline \end{array}$ is represented as a linear combination of the four Roberts basis images W_j. (b) Explain why we can always find unique a_j for any such 2x2 image.

Exercise 17

Suppose that the 2x2 image $\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}$ has energy e_1, e_2, e_3, e_4 along the respective four Roberts basis vectors W_1, W_2, W_3, W_4 respectively. What are the formulas for computing the four e_1 in terms of a, b, c, d?

5.9.6 The Frei-Chen basis for 3x3 neighborhoods

Usually, masks used for image processing are 3×3 or larger. A *standard basis* for 3×3 image neighborhoods is given in Figure 5.31. One advantage of the standard basis is that it is obvious how to expand any image neighborhood using this basis. However, the obvious expansion tells nothing about the 2D structure of the neighborhood. The Frei-Chen basis, shown in Figure 5.32, consists of a set of orthonormal masks which enable simple interpretation of the structure of the 3×3 neighborhood.

Representation of an image neighborhood in terms of the Frei-Chen basis allows intepretation of the energy as gradient, ripple, line, etc. Energy will be high when the intensity structure is similar to a basis vector, or mask. Each of the basis vectors has a specially

Roberts basis: $W_1 = \frac{1}{2}$ | 1 | 1 | / | 1 | 1 | $W_2 = \frac{1}{\sqrt{2}}$ | 0 | 1 | / | -1 | 0 | $W_3 = \frac{1}{\sqrt{2}}$ | +1 | 0 | / | 0 | -1 | $W_4 = \frac{1}{2}$ | -1 | 1 | / | 1 | -1 |

Constant region: | 5 | 5 | / | 5 | 5 | $= \frac{20}{2}(\frac{1}{2}$ | 1 | 1 | / | 1 | 1 | $) = 10W_1 \oplus 0W_2 \oplus 0W_3 \oplus 0W_4$

Step Edge: | -1 | +1 | / | -1 | +1 | $= 0W_1 \oplus \frac{2}{\sqrt{2}}W_2 \oplus \frac{2}{\sqrt{2}}W_3 \oplus 0W_4$

Step Edge: | +1 | +1 | / | -3 | +1 | $= 0W_1 \oplus \frac{4}{\sqrt{2}}W_2 \oplus 0W_3 \oplus \frac{-4}{2}W_4$

Line: | 0 | 8 | / | 8 | 0 | $= 8W_1 \oplus 0W_2 \oplus 0W_3 \oplus 8W_4$

Figure 5.30: (Top row) A basis for all 2x2 images which contains the two Roberts gradient masks; (row 2) constant region is a multiple of the constant image; (row 3) vertical step edge has energy only along the gradient masks; (row 4) diagonal step edge has most energy along the matching gradient mask; (row 5) line pattern has energy along constant mask W_1 and line mask W_4 .

| 1 | 0 | 0 | / | 0 | 0 | 0 | / | 0 | 0 | 0 |

| 0 | 1 | 0 | / | 0 | 0 | 0 | / | 0 | 0 | 0 |

| 0 | 0 | 1 | / | 0 | 0 | 0 | / | 0 | 0 | 0 |

...

| 0 | 0 | 0 | / | 0 | 0 | 0 | / | 0 | 1 | 0 |

| 0 | 0 | 0 | / | 0 | 0 | 0 | / | 0 | 0 | 1 |

Nine "standard" basis vectors for the space of all 3x3 matrices.

| 9 | 5 | 0 | / | 5 | 0 | 0 | / | 0 | 0 | 0 |

$= 9$ | 1 | 0 | 0 | / | 0 | 0 | 0 | / | 0 | 0 | 0 |

$+5$ | 0 | 1 | 0 | / | 0 | 0 | 0 | / | 0 | 0 | 0 |

$+5$ | 0 | 0 | 0 | / | 1 | 0 | 0 | / | 0 | 0 | 0 |

Figure 5.31: Any 3×3 matrix can be represented as a sum of no more than 9 scaled standard matrices: (top row) nine basis vectors and (bottom row) representation of an example matrix in terms of the basis.

gradient: $\mathbf{W_1} = 1/\sqrt{8}\begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}$ $\mathbf{W_2} = 1/\sqrt{8}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$

ripple: $\mathbf{W_3} = 1/\sqrt{8}\begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}$ $\mathbf{W_4} = 1/\sqrt{8}\begin{bmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix}$

line: $\mathbf{W_5} = 1/2\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$ $\mathbf{W_6} = 1/2\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$

Laplacian: $\mathbf{W_7} = 1/6\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$ $\mathbf{W_8} = 1/6\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$

constant: $\mathbf{W_9} = 1/3\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Figure 5.32: The Frei-Chen basis for the set of all 3x3 images with real intensity values.

designed structure. The basis vectors $\mathbf{W_1}$ and $\mathbf{W_2}$ are similar to the Prewitt and Sobel gradient masks, while the basis vectors $\mathbf{W_7}$ and $\mathbf{W_8}$ are similar to the common 3x3 Laplacian mask. The line masks will respond strongly to one-pixel wide lines passing through a 3x3 neighborhood, while the two ripple masks model two perpendicular waves with two peaks, two troughs, and three zero-crossings. The elements of the vectors differ slightly from the masks that we formerly designed in isolation because of the requirement that the set be orthogonal.

Algorithm 2 computes a binary image which detects intensity neighborhood structure with significant energy in a specified subspace. To detect edges, one could select pixels by their neighborhood energy along the basis vectors $\mathbf{W_1}$, $\mathbf{W_2}$, which would be indicated by setting $\mathbf{S} = \{1, 1, 0, 0, 0, 0, 0, 0, 0\}$. An example of the calculations projecting the intensity neighborhood on the Frei-Chen basis vectors is also given below.

Detect neighborhoods with high energy in a selected subspace.

$\mathbf{F}[\mathbf{r},\mathbf{c}]$ is an input intensity image; $\mathbf{F}$ is unchanged by the algorithm.
$\mathbf{S}$ is a bit vector such that $\mathbf{S[j]} = \mathbf{1}$ if and only if $\mathbf{W_j}$ is included in the subspace of interest.
thresh is a threshold on the fraction of energy required.
noise is the noise energy level.
$\mathbf{G}[\mathbf{r},\mathbf{c}]$ is the output image, a binary image where $\mathbf{G}[\mathbf{r},\mathbf{c}] = 1$ indicates that $\mathbf{F}[\mathbf{r},\mathbf{c}]$ has above threshold energy in the specified subspace $\mathbf{S}$.

```
procedure detect_neighborhoods(F, G, S, thresh, noise);
{
      for r := 0 to MaxRow - 1
      for c := 0 to MaxCol - 1
        {
          if [r,c] is a border pixel then G[r, c] := 0;
          else G[r, c] := compute_using_basis ( F, r, c, S, thresh, noise);
        } ;
}
 procedure compute_using_basis( IN, r, c, thresh, noise )
{
      N[r, c] is the 3x3 neighborhood centered at pixel [r,c] of IN[].
      average_energy := N[r, c] ∘ W9;
      subspace_energy := 0.0;
      for j := 1 to 8
      {
        if (S[j]) subspace_energy := subspace_energy + (N[r, c] ∘ Wj)^2;
      }
      if subspace_energy < noise return 0;
      if subspace_energy/((N[r, c] ∘ N[r, c])-average_energy) < thresh return 0;
      else return 1;
}
```

Algorithm 2: Compute detection image $\mathbf{G}[\mathbf{r},\mathbf{c}]$ from input image $\mathbf{F}[\mathbf{r},\mathbf{c}]$ and subspace $\mathbf{S}$.

Example of representing an intensity neighborhood using the Frei-Chen basis.

Consider the intensity neighborhood $\mathbf{N} =$

10	10	10
10	10	5
10	5	5

We find the component of this vector along each basis vector using the dot product as before. Since the basis is orthonormal, the total image energy is just the sum of the component energies and the structure of $\mathbf{N}$ can be interpreted in terms of the components.

$$
\begin{aligned}
N \circ W_1 &= \frac{5+5\sqrt{2}}{\sqrt{8}} \approx 4.3; \quad energy \approx 18 \\
N \circ W_2 &= \frac{5+5\sqrt{2}}{\sqrt{8}} \approx 4.3; \quad energy \approx 18 \\
N \circ W_3 &= 0; \quad energy = 0 \\
N \circ W_4 &= \frac{5\sqrt{2}-10}{\sqrt{8}} \approx -1; \quad energy \approx 1 \\
N \circ W_5 &= 0; \quad energy = 0 \\
N \circ W_6 &= 2.5; \quad energy \approx 6 \\
N \circ W_7 &= 2.5; \quad energy \approx 6 \\
N \circ W_8 &= 0; \quad energy = 0 \\
N \circ W_9 &= 25; \quad energy = 625
\end{aligned}
$$

The total energy in $\mathbf{N}$ is $\mathbf{N} \circ \mathbf{N} = 675$, 625 of which is explained just by the average intensity level along $\mathbf{W_9}$. The energy along all other components is 50 of which 36, or 72%, is along the gradient basis vectors $\mathbf{W_1}$ and $\mathbf{W_2}$. Thus, the neighborhood center would be marked as a detected feature in case the gradient subspace is of interest.

Exercise 18

Verify that the set of nine vectors shown in Figure 5.32 is an orthonormal set.

Exercise 19

(a) Represent the intensity neighborhood

0	0	1
0	1	0
1	0	0

in terms of the basis vectors shown in Figure 5.32. Is all the energy distributed along the *line* basis vectors $\mathbf{W_5}$ and $\mathbf{W_6}$? (b) Repeat the question (a) for the intensity neighborhood

0	1	0
0	1	0
0	1	0

.

Exercise 20

(a) Represent the intensity neighborhood $\begin{array}{|c|c|c|}\hline 10 & 10 & 10 \\ \hline 10 & 20 & 10 \\ \hline 10 & 10 & 10 \\ \hline \end{array}$ in terms of the basis vectors shown in Figure 5.32. Is all the energy distributed along certain basis vectors as you expect? (b) Would the interpretation of the intensity neighborhood $\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$ be different: why or why not? (c) What kind of image neighborhoods give responses to only W_7 and W_8?

Exercise 21

Write a program to implement the detection of pixels using the Frei-Chen basis as outlined in the algorithm above. Allow the user of the program to input the subspace $\mathbf{S}$ of interest as a string of 9 bits. The user should also be able to input the noise energy level and the threshold determining the minimum energy required in the selected subspace. Test your program on some real images and also on some test patterns such as those in the exercises above.

5.10 *Convolution and Cross Correlation

The previous sections showed how many useful detections can be performed by matching a mask or pattern image to an image neighborhood. Also, we saw that image smoothing could be put in the same framework. In this section, we give definitions for the important operations of *cross correlation* and *convolution*, which formalize the notion of moving a mask around the image and recording the dot product of the mask with each image neighborhood.

5.10.1 Defining operations via Masks

We begin by redefining simple image smoothing as the *cross correlation* of the image with a smoothing mask. A boxcar filter computes the ouptut image pixel as an equal weighting of a neighborhood of pixels of the corresponding input image pixel. This is equivalent to performing a dot product of an $m \times n$ image pattern of weights $\frac{1}{mn}$ as is shown in Figure 5.33 for a 3×3 mask. Assuming that m and n are both odd and the division by 2 ignores the remainder, Equation 5.25 defines the dot product used to compute the value of the output pixel $G[x, y]$ from input image $F[]$ using mask $H[]$. In this formulation, mask H is centered at the origin so that $H[0, 0]$ is the center pixel of the mask: it is obvious how H is used to weight the pixels from the neighborhood of $\mathbf{F}[\mathbf{x}, \mathbf{y}]$. An alternate formulation convenient to compute all output pixels of G results from a simple change of variables in Equation 5.25 yielding Equation 5.26 which can use masks $H[]$ with even dimensions.

19 DEFINITION *The* **cross correlation** *of image $F[x,y]$ and mask $H[x,y]$ is defined as*

$$\begin{aligned} G[x,y] &= F[x,y] \otimes H[x,y] \\ &= \sum_{i=-w/2}^{w/2} \sum_{j=-h/2}^{h/2} F[x+i, y+j] H[i,j] \end{aligned} \tag{5.25}$$

For implementation of this computational formula:
mask $H[x,y]$ is assumed to be centered over the origin, so negative coordinates make sense;
image $F[x,y]$ need not be centered at the origin;
result $G[x,y]$ must be defined in an alternate manner when $H[]$ does not completely overlap $F[]$.

An alternate formulation does not require a mask with odd dimensions, but should be viewed as an entire image transformation and not just an operation centered on pixel $G[x,y]$.

$$G[x,y] = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} F[x+i, y+j] H[i,j] \tag{5.26}$$

Exercise 22

Suppose that an image **F**[] has all 0 pixels except for a single 1 pixel at the image center. What output image **G**[] results from convolving **F**[] with the 3×3 boxcar shown in Figure 5.33?

Exercise 23

Design a single mask to detect edge elements making an angle of 30^0 with the X-axis. The mask should not respond strongly to edge elements of other directions or to other patterns.

Exercise 24 Corner detection

(a) Design a set of four 5×5 masks to detect the corners of any rectangle aligned with the image axes. The rectangle can be brighter or darker than the background. (b) Are your masks orthogonal? (c) Specify a decision procedure to detect a corner and justify why it would work.

5.10.2 The Convolution Operation

20 DEFINITION *The* **convolution** *of functions $f(x,y)$ and $h(x,y)$ is defined as*

$$g(x,y) = f(x,y) \star h(x,y) \equiv \int_{x'=-\infty}^{+\infty} \int_{y'=-\infty}^{+\infty} f(x',y') h(x-x', y-y') dx' dy' \tag{5.27}$$

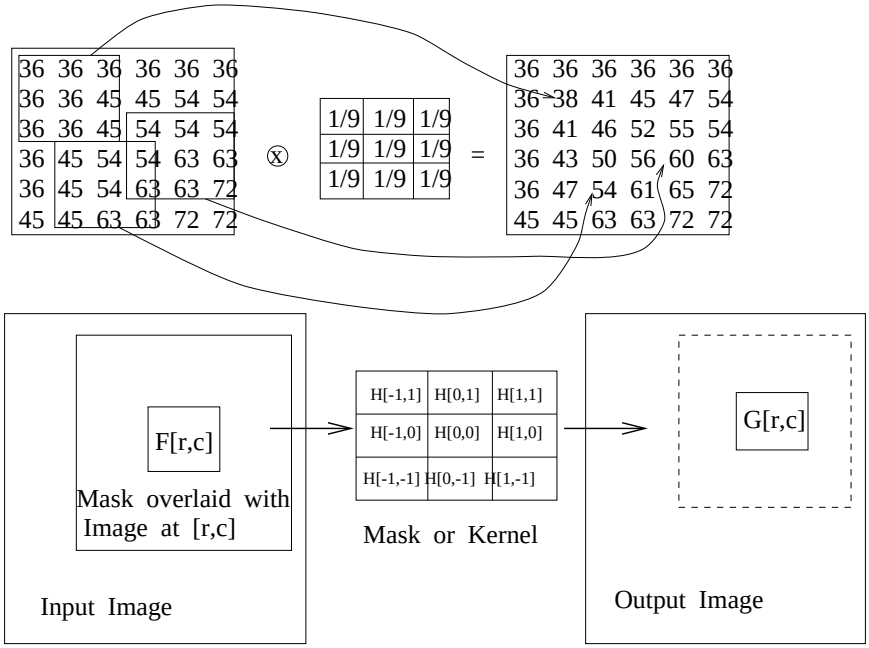


Figure 5.33: Smoothing an image using a 3×3 boxcar filter can be viewed as computing a dot product between every input image neighborhood and a mask (boxcar) that is a small image of equal values.

Convolution is closely related to cross correlation and is defined formally in terms of continuous picture functions in Equation 5.27. In order for the integrals to be defined and to have practical use, the 2D image functions $f(x, y)$ and $h(x, y)$ should have zero values outside of some finite rectangle in the $xy-plane$ and have finite volume under their surface. For filtering, *kernel* function $h(x, y)$ will often be zero outside some rectangle that is much smaller than the rectangle that supports $f(x, y)$. For a global analysis of the spatial frequency of f, the rectangle supporting h will include all of the support of f. More details are given in the optional section of Fourier analysis below. Figure 5.34 shows an interpretation of steps in computing the value of $g(x)$ from the two functions being convolved in the case of 1D signals. Kernel function $h(x)$ is first flipped about the origin and then translated to the point x at which $g(x)$ is being computed. $g(x)$ is then computed by integrating the products of the input function $f(x')$ times the relocated kernel $h(x' - x)$; since, that function is zero outside of the interval $[a, b]$, integration can be confined to this finite interval. Convolution is easily carried out with digital images by using discrete sums of products instead of the continuous integrals defined above.

Cross correlation translates the mask or kernel directly to the image point $[x, y]$ without flipping it; otherwise it is the same operation used in convolution. Conceptually, it is easier not to have to think of flipping the kernel, but rather just placing it at some image location. If the kernel is symmetric so that the flipped kernel is the same as the original kernel, then the results of convolution and correlation will be identical. While this is the case for smoothing masks and other isotropic operators, many edge detection masks are

Exercise 25 Rectangle detection

Use the corner detection procedure of the previous exercise to implement a program that detects rectangles in an image. (Rectangle sides are assumed to align with the sides of the image.) The first step should detect candidate rectangle corners. A second step should extract subsets of four candidate corners that form a proper rectangle according to geometric constraints. An optional third step might further check the four corners to make sure that the intensity within the candidate rectangle is uniform and contrasting with the backgorund. What is the expected behavior of your program if it is given a noisy checkerboard image? Test your program on a noisy checkerboard, such as in Figure 5.7 and an image of a building with rectangular windows, such as in Figure 5.42.

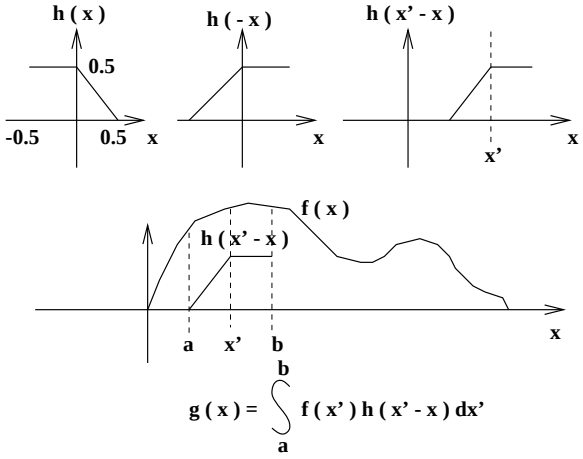


Figure 5.34: Computing convolution $g(x)$ of signal $f(x)$ with kernel $h(x)$ denoted $g(x) = f(x) \star h(x)$. For any domain point x, kernel h is flipped and then translated to x; then, $g(x)$ is computed by summing all the products of $f(x)$ and the values of the flipped and translated h.

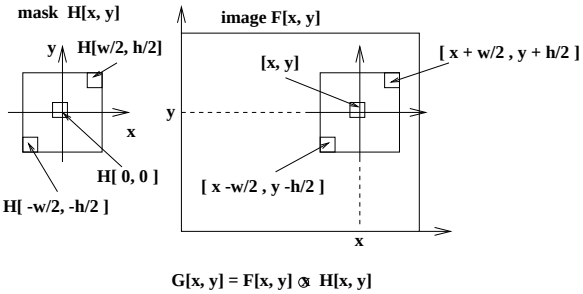


Figure 5.35: Computing the cross correlation $G[x, y]$ of image $F[x, y]$ with mask $H[x, y]$: $G[x, y] = F[x, y] \otimes H[x, y]$. To compute $G[x, y]$, the mask $H[x, y]$ is centered on input image point $F[x, y]$ to sum all the products of image values of F and the corresponding overlaid weights of H.

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

$\otimes$

0	-1	0
-1	4	-1
0	-1	0

$=$

0	0	0	0	0	0
0	0	-5	-5	-5	0
0	-5	10	5	5	0
0	-5	10	0	0	0
0	0	-10	10	0	0
0	0	0	0	0	0

Cross correlation with a LOG mask produces zero-crossings at boundaries.

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

$\otimes$

-1	0	+1
-1	0	+1
-1	0	+1

$=$

0	0	0	0	0	0
0	5	5	0	0	0
0	10	10	0	0	0
0	10	15	5	0	0
0	5	10	10	5	0
0	0	0	0	0	0

Cross correlation with column derivative mask detects column boundaries.

5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

$\otimes$

+1	+1	+1
0	0	0
-1	-1	-1

$=$

0	0	0	0	0	0
0	-5	-10	-15	-15	0
0	-5	-10	-15	-15	0
0	5	5	5	0	0
0	5	10	10	5	0
0	0	0	0	0	0

Cross correlation with row derivative mask detects column boundaries.

Figure 5.36: Cross correlation of an image with various masks to enhance regions boundaries: (top) second derivative operator produces a zero-crossing at boundaries; (center) column (x) derivative operator detects changes across columns; (right) column (y) derivative operator detects changes across rows.

asymmetric. Despite the formal difference between convolution and cross correlation, those who do image processing often loosely refer to either operation as "convolution" because of their similarity. The many masks used in this chapter, and others, have been presented assuming that they would not be flipped before being applied to an image. *Normalized cross correlation* will normalize by dividing $G[x, y]$ by the magnitudes of both $F[x, y]$ and $H[x, y]$ so that the result can be interpreted as a result of matching the structure of F with the structure of H that is independent of scale factors, as has been discussed in previous sections.

Exercise 26

Given $H =$

1	0	-3
0	4	0
-3	0	1

and $F =$

5	5	5	5	5	5
5	5	5	5	8	8
5	5	5	8	8	8
5	5	8	8	8	8
8	8	8	8	8	8

,

compute $G = F \otimes H$.

Exercise 27 Point spread

Given kernel $H =$

1	2	1
2	5	2
1	2	1

, what is the result of convolving it with an image $F[x, y]$ which has $F[x_0, y_0] = 1$ and all other pixels 0?

Exercise 28

Suppose that function $h(x)$ takes value 1 for $-1/2 \leq x \leq 1/2$ and is zero elsewhere, and suppose that function $f(x)$ takes value 1 for $10 \leq x \leq 20$ and zero elsewhere. (a) Sketch the two functions f and h. (b) Compute and sketch the function $g(x) = f(x) \star h(x)$. (c) Compute and sketch the function $g(x) = h(x) \star h(x)$.

5.10.3 Possible parallel implementations

One can see from the definition of convolution, that $g(x_1, y_1)$ can be computed independently of $g(x_2, y_2)$; in fact, all the integrations can be done at the same time in parallel. Also, each integration computing an individual value $g(x, y)$ can form all of the products simultaneously, enabling a highly parallel system. Various computer architectures can be envisioned to perform all or some operations in parallel.

5.11 *Analysis of spatial frequency using sinusoids

Fourier analysis is very important in signal processing; its theory and practice fills many books. We give only a brief snapshot here, building on the notions of a vector space already introduced.

Exercise 29

The *pointilist* school of painters created their paintings by dabbing their paintbrush perpendicularly on the canvas and creating one dab/point of color at a time. Each dab is similar to a single pixel in a digital image. The human viewer of the artwork would stand back and a smooth picture would be perceived. Implement a program to do pointilist art. The program should present a palette of colors and some other options, such as choosing paintbrush size or whether '+' or 'XOR' would be used for dabbing, etc. When your program is working, create a painting of a starry night. Your program should work with an external file representing the painting in progress so that a user could save the session for later continuation.

Exercise 30

Implement a program to convolve a mask with an image. The program should read both the image and mask from input files in the same format. Perhaps you can test it on the artwork produced by the program from the previous exercise.

Exercise 31

Suppose in the search for extra terrestrial intelligence (SETI) deep space is scanned with the hope that interesting signals would be found. Suppose signal **S** is a concatenation of the first 100 prime numbers in binary form and that **R** is some received signal that is much longer than **S**. Assume that **R** has noise and has real valued coordinates. To find if **S** is embedded in **R**, would cross correlation or normalized cross correlation work? Why?

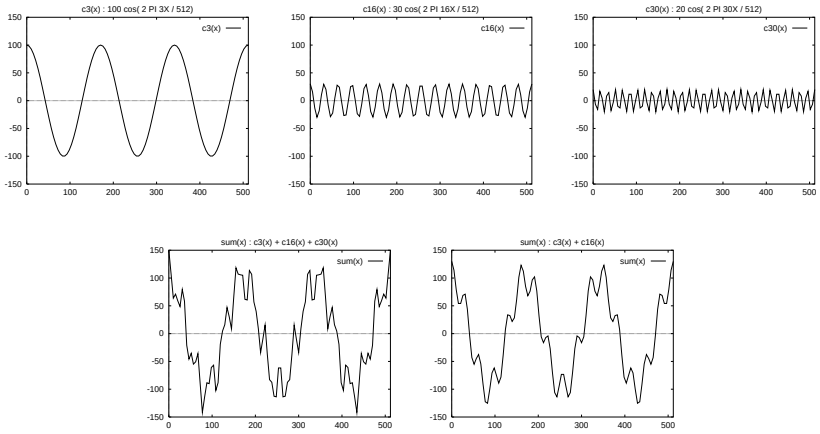


Figure 5.37: (Top row) Three sinusoids, $100cos(2\pi\frac{3x}{512})$, $30cos(2\pi\frac{16x}{512})$ and $20cos(2\pi\frac{30x}{512})$; (bottom row left) sum of all three and (bottom row right) sum of first two.

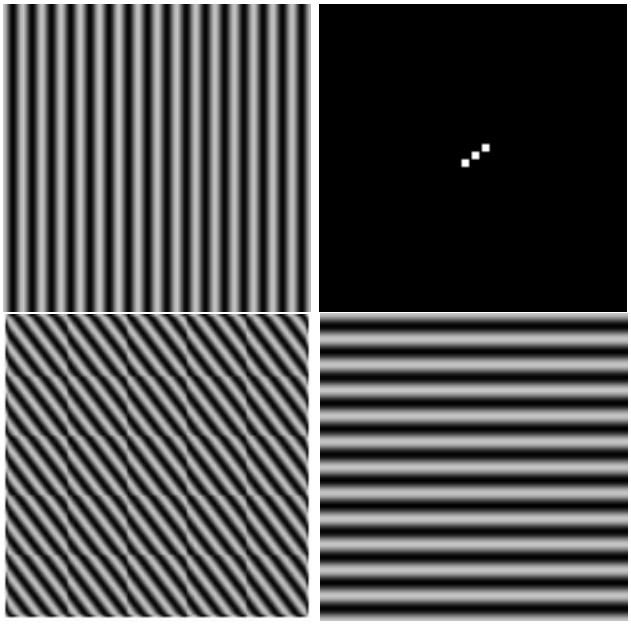


Figure 5.38: Different sinusoidal picture functions in the spatial domain $[x, y]$: The top left image was generated by the formula $100cos(2\pi(16x/512)) + 100$ and has 16 cycles along the x-axis. The bottom right image was generated by the formula $100cos(2\pi(12y/512)) + 100$ and has 12 cycles along the y-axis. The bottom left image was generated by the formula $100cos(2\pi(16x/512 + 12y/512)) + 100$: note how the waves of the bottom left image align with those of the top left and bottom right. The Fourier power spectrum is shown at the top right.

The mathematician Fourier imagined the surface of the sea as a sum of sine waves. Large waves caused by the tide or ships had long wavelengths (low frequency), while smaller waves caused by the wind or dropped objects, etc. had short wavelengths (high frequency). The top row of Figure 5.37 shows three pure waves of 3, 16, and 30 cycles across a 1D space of $x \in [0, 512]$: in the bottom row are two functions, one which sums all three waves and one which sums only the first two. Similar sets of waves can be used to create 2D picture functions or even 3D density functions.

The Fourier theory in analysis shows how most real surfaces or real functions can be represented in terms of a basis of sinusoids. The energy along the basis vectors can be interpreted in terms of the structure of the represented surface (function). This is most useful when the surface has repetitive patterns across large regions of the surface; such as do city blocks of an aerial image of an urban area, waves on a large body of water, or the texture of a large forest or farm field. The idea is to expand the entire image, or various windows of it, using a Fourier basis and then filter the image or make decisions about it based on the image energy along various basis vectors. For example, high frequency noise can be removed by subtracting from the image all the components along high frequency sine/cosine waves. Equivalently, we can reconstruct our spatial image by adding up only the low frequency waves and ignoring the high frequency waves.

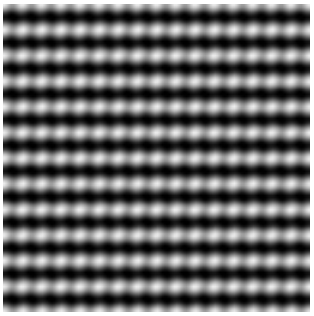


Figure 5.39: A picture function formed by the sum $I[x,y] = 100E_i + 30E_j + 10E_k$ where E_i is as in the bottom right of Figure 5.38, E_j is as in the top left, and E_k is as in the bottom left. (Perhaps it models an orchard or foam padding?)

5.11.1 A Fourier Basis

For an intuitive introduction, let us assume an orthonormal set of sinusoidal basis images (or picture functions) $E_k \approx E_{u,v}(x,y)$. Presently, k and u,v are integers which define a finite set of basis vectors. It will soon become clear how parameters u,v determine the basis vectors, but right now, we'll use the single index k to keep our focus on the fundamental concept. The bottom row of Figure 5.37 shows two signals formed by summing three or two of the pure cosine waves shown in the top row of the figure. More complex functions can be created by using more pure cosine waves. Figure 5.39 shows the result of summing scaled versions of the three "pure" waves in Figure 5.38 to create a new picture function. By using the Fourier basis functions E_k, any picture function can be represented as $I[x,y] = \sum_{k=0}^{N-1} a_k E_k[x,y]$. As in previous sections, a_k is a measure of the similarity between $I[x,y]$ and $E_k[x,y]$ and the energy of image $I[x,y]$ along that particular component wave. Useful image processing operations can be done by operating on the values of a_k rather than on the individual intensities $I[x,y]$. Three major operations are described below.

Important image processing operations using the Fourier basis:

1. The Fourier basis can be used to **remove high frequency noise** from the image or signal. The signal f is represented as $\sum_k a_k E_k$. The values of a_k for the high frequency sinusoids E_k are set to zero and a new signal $\hat{f}$ is computed by summing the remaining basis functions with $a_k \neq 0$.

2. The Fourier basis can be used to **extract texture features** that can be used to classify the type of object in an image region. After representing the image, or image region, in the Fourier basis, various a_k can be used to compute features to be used for the classification decision. Regions of waves on water or crops organized by rows are amenable to such a process. The a_k are useful for determining both the frequency and direction of textured regions.

3. The Fourier basis can also be used for **image compression**. A sender can send a subset of the a_k and a receiver can reconstruct the approximate picture by summing the known sinusoidal components. All of the a_k can be transmitted, if needed, in either

order of energy, or in order of frequency: the receiver can terminate the transmission at any time based on the content received up to that point.

Our goal here is to produce a useful basis of picture functions and an understanding of how to use it in practice. Some important mathematical background using continuous functions is needed. For this development, lets assume that the origin of our coordinate system is at the center of our picture function, which is defined for a square region of the xy-plane. When we proceed to a digital version, the picture will be represented by digital image $I[x, y]$ with N^2 samples. First, we establish a set of sinusoids of different frequency as an orthogonal basis for continuous signals f. If m, n are any two different integers, then the two cosine waves with these frequency parameters are orthogonal over the interval $[-\pi, \pi]$. The reader should work through the following exercises to verify that the function set $\{\mathbf{1}, sin(mx), cos(nx), \ldots\}$ is an orthogonal set of functions on $[-\pi, \pi]$. The orthogonality of the cosine waves follows from Equation 5.28, since $sin(k\pi) = 0$ for all integers k.

$$\begin{aligned} \int_{-\pi}^{\pi} cos(m\theta)cos(n\theta)d\theta &= \frac{sin(m-n)\pi}{2(m-n)} + \frac{sin(m+n)(-\pi)}{2(m+n)} \\ &= 0 \; for \;\; m^2 \neq n^2 \end{aligned} \tag{5.28}$$

N symmetrically spaced samples of these cosine functions will create a set of vectors that will be orthogonal in the sense defined previously by the dot product.

Exercise 32

Consider the set of all continuous functions f defined on the interval $x \in [x_1, x_2]$. Show that this set of functions $f, g, h, \ldots$, together with scalars $a, b, c, \ldots$ forms a vector space by arguing that the following properties hold.

$$\begin{aligned} f \oplus g = g \oplus f \qquad & (f \oplus g) \oplus h = f \oplus (g \oplus h) \\ c(f \oplus g) = cf \oplus cg \qquad & (a+b)f = af \oplus bf \\ (ab)f = a(bf) \qquad & 1f = f \\ 0f = \mathbf{0} & \end{aligned}$$

5.11.2 2D Picture Functions

21 Definition *The complex valued picture function*

$$\begin{aligned} E_{u,v}(x, y) &\equiv e^{-j\ 2\pi(ux+vy)} \\ &= cos(2\pi(ux+vy)) \; - jsin(2\pi(ux+vy)) \end{aligned} \tag{5.30}$$

where u and v are spatial frequency parameters as shown in Figure 5.38 and $j = \sqrt{-1}$.

Using a complex value is a convenience that allows us to separately account for both a cosine and sine wave of the same frequency: the sine wave has the same structure as the

Exercise 33

For the space of continuous functions of $x \in [x_1, x_2]$, as in the exercise above, define a dot product and corresponding norm as follows.

$$f \circ g \;=\; \int_a^b f(x)g(x)dx \quad ; \quad \|f\| = \sqrt{f \circ f} \tag{5.29}$$

Argue why the following four properties of a dot product hold.

$$\begin{aligned} (f \oplus g) \circ h &= (f \circ g) + (g \circ h) \\ f \circ f &\geq 0 \\ f \circ f = 0 \Longleftrightarrow f &= \mathbf{0} \\ f \circ g &= g \circ f \\ (cf) \circ g &= c(f \circ g) \end{aligned}$$

Exercise 34 Odd and even functions

A function is an *even function* iff $f(-x) = f(x)$ and a function is an *odd function* if $f(-x) = -f(x)$. (a) Show that $cos(mx)$ is an even function and that $sin(nx)$ is an odd function, where m, n are nonzero integers. (b) Let f and g be odd and even continuous functions on $[-L, L]$, respectively. Argue that $\int_{-L}^{L} f(x)g(x)dx = 0$.

Exercise 35

Using the definition of the dot product given in Exercise 33 above, show that the set of sinusoidal functions f_k defined below are orthogonal on the interval $[-\pi, \pi]$.
$f_0(x) = 1$; $f_1(x) = sin(x)$; $f_2(x) = cos(x)$; $f_3(x) = sin(2x)$; $f_4(x) = cos(2x)$; $f_5(x) = sin(3x)$; $f_6(x) = cos(3x)$; ...

cosine wave but is 1/4 wavelength out of phase with the cosine wave. When one of these basis functions correlates highly with a picture function, it means that the picture function has high energy in frequency u, v. The *Fourier transform*, converts a picture function into an array of these correlations. We start with the integral form and then give the discrete summation for digital images below.

22 DEFINITION *The 2D* **Fourier Transform** *transforms a spatial function $f(x, y)$ into the u, v frequency domain.*

$$\begin{aligned} F(u,v) &\equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)E_{u,v}(x,y)dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j\ 2\pi(ux+vy)}dxdy \end{aligned} \tag{5.31}$$

The function f must be well-behaved; specifically, picture functions have the following properties which make the above formula well-defined.
$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mid f(x,y) \mid dxdy$ is finite; moreover, $f(x, y) = 0$ outside of some rectangle R so the infinite limits in the formula can be replaced by the limits of R. Also, f has a finite number of extrema and no infinite discontinuities in R.

Exercise 36

What is the special meaning of $F(0, 0)$, where $F(u, v)$ is the Fourier transform of picture function $f(x, y)$?

Often, we want to work with the *power spectrum*, which combines energy from sine and cosine waves of the same frequency components u, v. A power spectrum is shown at the top right of Figure 5.38.

23 DEFINITION *The* **Fourier power spectrum** *is computed as*

$$P(u,v) \equiv (\ Real(F(u,v))^2\ +\ Imaginary(F(u,v))^2\ )^{1/2} \tag{5.32}$$

Figure 5.40 shows how the actual wavelength of a 2D sinusoid relates to the wavelengths projected along each axis. u is the frequency along the X-axis in cycles per unit length and $1/u$ is the wavelength. v is the frequency along the Y-axis and $1/v$ is the wavelength. λ is the wavelength of the sinusoid along its natural axis, or direction of travel. By computing the area of the triangle shown at the right in Figure 5.40 in two different ways, we obtain the following formula which will help us interpret what the power spectrum tells us about the frequency content of the original image.

$$\begin{aligned} \lambda\sqrt{(1/u)^2 + (1/v)^2} &= (1/u)(1/v) \\ \lambda &= \frac{1}{\sqrt{u^2+v^2}} \end{aligned} \tag{5.33}$$

Let's assume that the both the width and height of our picture have length 1. In Figure 5.38 there are $u = 16$ cycles across the width of the picture, so the wavelength is $1/u = 1/16$. Similarly, we have $1/v = 1/12$. Applying the formula in Equation 5.33, we obtain $\lambda = 1/20$. By counting the number of waves in Figure 5.38 at the bottom left, we see

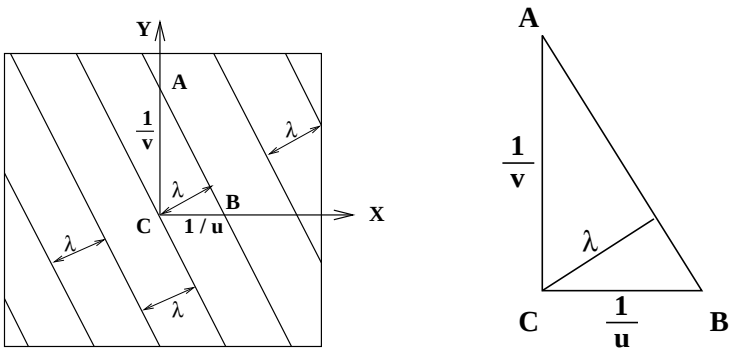


Figure 5.40: Relationships among the wavelengths $1/u_0$ and $1/v_0$ of a sinusoid intercepted by the X and Y axes and the wavelength λ of the 2D wave.

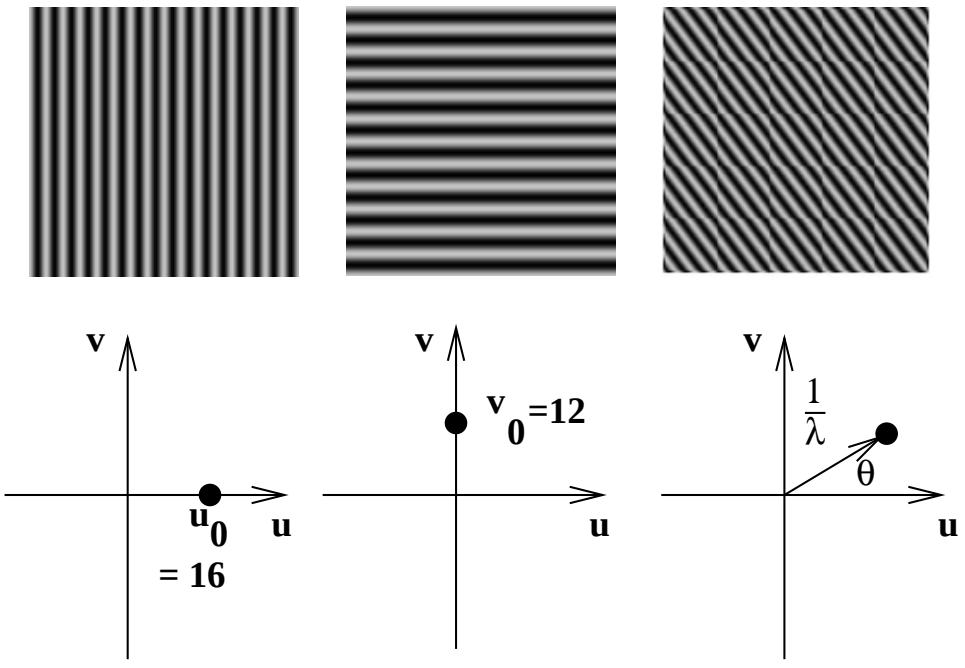


Figure 5.41: (Top row) three sinusoidal waves and (bottom row) their principle response in the power spectrum.

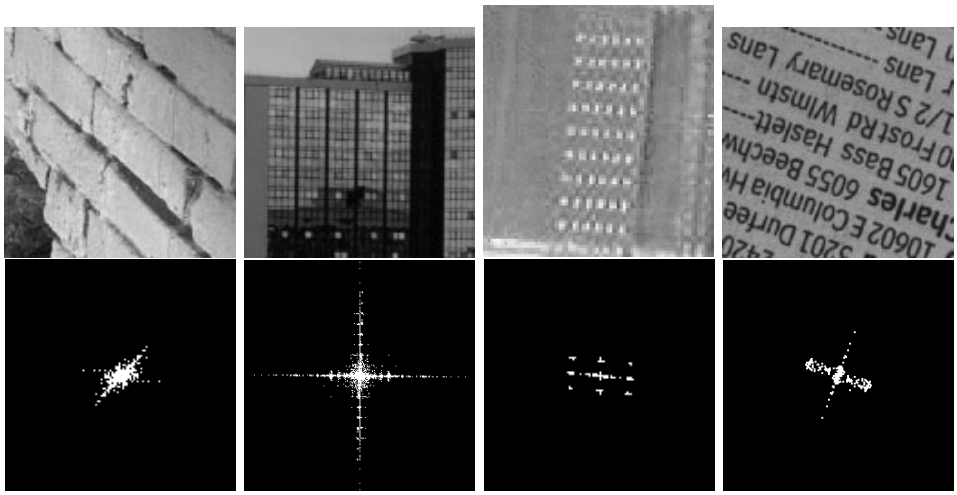


Figure 5.42: Four images (above) and their power spectrums (below). The power spectrum of the brick texture shows energy in many sinusoids of many frequencies, but the dominant direction is perpendicular to the 6 dark seams running about 45 degrees with the X-axis. There is noticable energy at 0 degrees with the X axis, due to the several short vertical seams. The power spectrum of the building shows high frequency energy in waves along the X-direction and the Y-direction. The third image is an aerial image of an orchard: the power spectrum shows the rows and columns of the orchard and also the "diagnonal rows". The far right image, taken from a phone book, shows high frequency power at about 60^o with the X-axis, which represents the texture in the lines of text. Energy is spread more broadly in the perpendicular direction also in order to model the characters and their spacing.

27 waves across the 1.4 unit diagonal, yielding the expected frequency $27/1.4 \approx 20$ along the actual direction of the 2D wave.

Figure 5.41 shows the principle response in the power spectrum of the three sinusoidal picture functions originally shown in Figure 5.38. The power spectrum at the top right of Figure 5.38 actually shows heavy responses at three points, not just one. First of all, note that $F(0,0)$ is just the total energy in $f(x,y)$; since each sinusoid of Figure 5.38 has mean of 100 and not 0, they have significant average energy at *zero frequency*. Secondly, it is evident from the defintion that $P(-u,-v) = P(u,v)$, so the power spectrum is symmetric about the origin $u = 0, v = 0$. Figure 5.42 shows the power spectrums of four real images.

The power spectrum need not be interpreted as an image, but rather as a 2D display of the power in the original image versus the frequency parameters u and v. In fact, a transform can be computed via optics, and thus it can be realized as a physical image. In Chapter 2, there was brief mention of a sensor array partitioned into sectors and rings (refer back to the "ROSA" configuration in the relevant figure from Chapter 2). Since the power spectrum is symmetrical about the x-axis, as shown in Figure 5.42, the sectors can be used to sample directional power, while the rings sample frequency bands without regard to direction. Such sampling can also be done in software. In any case, if n_r rings and n_s sectors are sampled, one obtains $n_r + n_s$ features that may be useful for classification of the image neighborhood from which they were obtained.

5.11.3 Discrete Fourier Transform

The discrete Fourier transform, or DFT, which is immediately applicable to digital images is defined in Equation 5.34. We already know that a basis for the set of $N \times N$ images with real intensity values must have N^2 basis vectors. Each of these is determined by a pair of frequency parameters u, v which range from 0 to $N-1$ as used in the formulas below.

24 DEFINITION *The* **Discrete Fourier Transform (DFT)** *transforms an image of $N \times N$ spatial samples $I[x,y]$ into an $N \times N$ array $F[u,v]$ of coefficents used in its frequency reresentation.*

$$F[u,v] \equiv \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x,y] e^{\frac{-2\pi j}{N}(xu+yv)} \tag{5.34}$$

To compute the single frequency domain element ("pixel") $F[u,v]$, we just compute the dot product between the entire image $I[x,y]$ and a "mask" $E_{u,v}[x,y]$, which is usually not actually created, but computed implicitly in terms of u, v and the *cos* and *sin* functions as needed. We also define an inverse transform to transform a frequency domain representation $F[u,v]$ into a spatial image $I[x,y]$. Although it is useful to display the transform F as a 2D image, it might be less confusing to insist that it is NOT really an image. We have used the formal term *frequency representation* instead.

25 DEFINITION *The* **Inverse Discrete Fourier Transform (IDFT)** *transforms an $N \times N$ frequency representation $F[u,v]$ into an $N \times N$ image $I[x,y]$ of spatial samples.*

$$I[x,y] \equiv \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F[u,v] e^{\frac{+2\pi j}{N}(ux+vy)} \tag{5.35}$$

If $F[x,y]$ was computed by "forward" transforming $I[x,y]$ in the first place, we want the inverse transform to return to that original image. This property does hold for the pair of definitions given above: proof is left to the guided exercises below. We first focus the discussion on the practical use of the **DFT & IDFT**. For storage or communication of an image, it might be useful to transform it into its frequency representation; the input image can be recovered by using the inverse transform. In image processing, it is common to perform some enhancement operation on the frequency representation before transforming back to recover the spatial image. For example, high frequencies can be reduced or even removed by reducing, or zeroing, the elements of $F[u,v]$ that represent high frequency waves. The *convolution theorem* below gives an elegant theoretical interpretation of this intuitive process.

Exercise 37 Some basics about complex numbers

Use the definition $e^{j\omega} = cos\ \omega + j\ sin\ \omega$. (a) Show that $(e^{j\omega})^n = cos(n\omega) + j\ sin(n\omega)$. (b) Show that $x = e^{j\frac{2\pi k}{N}}$ is a solution to the equation $x^N - 1 = 0$ for $k = 0, 1, \ldots, N-1$. (c) Let $x_0 = 1 = e^{j\frac{2\pi 0}{N}}, \ldots, x_k = e^{j\frac{2\pi k}{N}}$ be the N roots of the equation $x^N - 1 = 0$. Show that $x_1 + x_2 + x_3 + \ldots + x_{N-1} = 0$.

Exercise 38 Proof of invertability of DFT/IDFT transforms

We want to prove that substituting the $F[u,v]$ from Equation 5.34 into Equation 5.35 returns the exact original value $I[x,y]$. Consider the following summation, where x, y, s, t are integer parameters in the range $[0, N-1]$ as used in the transform definitions:

$$G(x,y,s,t) \;=\; \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} e^{\frac{j\,2\pi}{N}((x-s)u \,+\, (y-t)v)}.$$

(a) Show that if $s = x$ and $t = y$, then $G(x,y,s,t) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} 1 \;=\; N^2$. (b) Show that if $s \neq x$ or $t \neq y$ then $G(x,y,s,t) = 0$.
(c) Now show the major result that the inverse transform applied to the transform returns the original image.

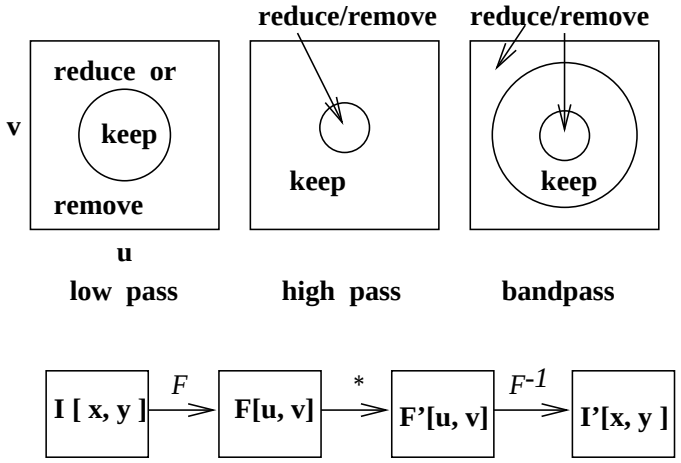


Figure 5.43: Bandpass filtering can be performed by a Fourier transformation into the frequency domain ($\mathbf{F}[\mathbf{u},\mathbf{v}]$) followed by multiplication (*) by a bandpass filter. The multiplication can set the "coefficients" u, v of various frequencies to zero, as shown in the top row. This modified frequency representation is then inverse transformed to obtain the modified spatial image $\mathbf{I'}[\mathbf{x},\mathbf{y}]$.

5.11.4 Bandpass Filtering

Bandpass filtering is a common image processing operation performed in the frequency domain and is sketched in Figure 5.43. The **DFT** is used to transform the image into its frequency representation where the coefficients for some frequencies are reduced, perhaps to zero, while others are preserved. A sketch of the *low pass filter* is given at the left of Figure 5.43; intuitively, the high frequencies are erased and then the altered frequency representation is inverse transformed via Equation 5.35 to obtain a smoothed version of the original image. Instead of *erasing* elements of the frequency representation, we can take the dot product of $F[u,v]$ with a 2D Gaussian, which will weight the low frequency components high and weight the high frequency components low. Figure 5.43 also shows how the frequency representation would be altered to accomplish high pass and bandpass filtering. The theory of convolution below provides more insight to these operations.

5.11.5 Discussion of the Fourier Transform

The *Fast Fourier Transform* saves computational time by sharing common operations for different pairs of u, v and is usually used on square images of size $2^m \times 2^m$. Despite its common use in image processing, the Fourier Transform can cause unwanted degradation in local features of images because it is a *global transform* that uses all image pixels in every computation of $F[u,v]$. For example, to represent hair or grass, or some other fine feature, high frequency waves must be used. First, such high frequencies might be filtered out as noise; however, even if not filtered out, a high frequency response $F[u_h, v_h]$ would be computed by the dot product of a high frequency wave with the entire image. Since the region of hair or grass would only be a small part of the entire image, the value of the dot product could be high or low, depending on the rest of the image content. In the last ten years, there has been a strong movement toward using *wavelets* instead of image-size waves. Wavelets allow more sensitivity to local image variation, yet retain some major advantages of using global sinusoids. JPEG and other picture compression schemes use cosine wave representations on subimages to reduce the data size. Sometimes such compression must be suppressed in order to preserve needed local image detail.

5.11.6 *The Convolution Theorem

In this section, we sketch a proof of the important *convolution theorem* , which tells us that convolution of two functions in the spatial domain is equivalent to pointwise multiplication of their frequency representations. This equivalence has great practical utility as has already been seen.

Convolution Theorem:

If $f(x,y)$ and $h(x,y)$ are well-behaved functions of spatial parameters x, y, then $\mathbf{F}(f(x,y) \star h(x,y)) \equiv \mathbf{F}((f \star h)(x,y)) = \mathbf{F}(f(x,y))\mathbf{F}(h(x,y)) = \mathbf{F}(u,v)\mathbf{H}(u,v)$, where $\mathbf{F}$ is the Fourier transform operator and $\star$ is the convolution operator.

Before sketching a proof in the 1D case, we make the interpretation that is commonly used in signal processing: convolution can be carried out without applying a mask $h(x,y)$ to all points of an image $f(x,y)$.

Filter image $f(x,y)$ with mask $h(x,y)$

(1) Fourier transform the image $f(x,y)$ to obtain its frequency rep. $F(u,v)$.
(2) Fourier transform the mask $h(x,y)$ to obtain its frequency rep. $H(u,v)$
(3) multiply $F(u,v)$ and $H(u,v)$ pointwise to obtain $F'(u,v)$
(4) apply the inverse Fourier transform to $F'(u,v)$ to obtain the filtered image $f'(x,y)$.

Algorithm 3: Filtering image $f(x,y)$ with mask $h(x,y)$ using the Fourier transform

For commonly used filters h, the transform H is likely to be available either in closed functional form or as a stored array in memory, thus short cutting step (2). Signal processing texts typically contain a table of *transform pairs* $< h, H >$ described both graphically and in functional form so that the user can select a filter with appropriate properties. We now sketch a proof of the convolution theorem for the 1D case; these same steps can be followed for the 2D case. An intermediate step is to show what happens to the transform when the function is shifted.

Shift Theorem: $\mathbf{F}(f(x-x_0)) = e^{-j\ 2\pi u x_0}\mathbf{F}(f(x))$

By definition $\mathbf{F}(f(x-x_0)) \equiv \int_{-\infty}^{+\infty} f(x-x_0)e^{-j\ 2\pi ux}dx$.
Making the change of variable $x' = x - x_0$, we obtain

$$\begin{aligned}\mathbf{F}(f(x-x_0)) &= \int_{-\infty}^{+\infty} f(x')e^{-j\ 2\pi u(x'+x_0)}dx' \\ &= \int_{-\infty}^{+\infty} e^{-j\ 2\pi ux_0}f(x')e^{-j\ 2\pi ux'}dx' \\ &= e^{-j\ 2\pi ux_0}\mathbf{F}(f(x)),\end{aligned} \tag{5.36}$$

since the first factor is constant with respect to the variable of integration x'. Note that

$$|\ e^{-j\ 2\pi ux_0}\ |^2 = cos^2(2\pi ux_0) + sin^2(2\pi ux_0) = 1, \tag{5.37}$$

so shifting the function does not change the energy of $f(x)$ or $\mathbf{F}(u)$.

We can now use the above result to sketch the proof of the convolution theorem.

$$\mathbf{F}((f \star h)(x)) \equiv \int_{x=-\infty}^{x=+\infty} (\int_{t=-\infty}^{t=+\infty} f(t)h(x-t)dt)e^{-j\ 2\pi ux}dx. \tag{5.38}$$

Using the good behavior assumed for functions f and h allows the order of integration to be interchanged.

$$\mathbf{F}((f \star h)(x)) = \int_{t=-\infty}^{t=+\infty} f(t)\ (\int_{x=-\infty}^{x=+\infty} h(x-t)e^{-j\ 2\pi ux}dx)\ dt \tag{5.39}$$

Using the shift theorem,

$$\int_{x=-\infty}^{x=+\infty} h(x-t)e^{-j\ 2\pi ux}dx = e^{-j\ 2\pi ut}\mathbf{H}(u), \tag{5.40}$$

where $\mathbf{H}(u)$ is the Fourier transform of $h(x)$. We now have that

$$\begin{aligned} \mathbf{F}((f \star h)(x)) &= \int_{t=-\infty}^{t=+\infty} f(t)(e^{-j\ 2\pi ut}\mathbf{H}(u))\ dt \\ &= \mathbf{H}(u) \int_{t=-\infty}^{t=+\infty} f(t)e^{-j\ 2\pi ut}\ dt \\ &= \mathbf{H}(u)\mathbf{F}(u) = \mathbf{F}(u)\mathbf{H}(u) \end{aligned} \tag{5.41}$$

and the theorem is proved for the 1D case.

Exercise 39

Follow the steps given above for proving the Shift Theorem and Convolution Theorem in the 1D case and extend them to the 2D case.

5.12 Summary and Discussion

This was a long chapter covering several methods and many examples: it is important to review major concepts. Methods were presented to enhance appearance of images for either human consumption or for automatic processing. A few methods were given for remapping intensity levels to enhance the appearance scene objects; it was shown that often some image regions were improved at the expense of degrading others. The methods were presented only for grey level images; however, most can be extended to color images, as you may already know from using some image enhancing tool. Edge enhancement was also presented as a method for human consumption. Hopefully, the artists toolbox has been expanded.

The most important concept of the chapter is that of using a mask or kernel to define a local structure to be applied throughout the image. Convolution and cross correlation are two very powerful and related techniques which compute a result at $I[x, y]$ by taking the sum of the pointwise products of the input image intensity and a corresponding mask value. These are linear operations that are pervasive in both theory and practice. Within this context, it was shown that the response to a particular mask at a particular image point (the *correlation*) is a measure of how similar the structure of the mask is to the structure of the image neighborhood. This notion provides a practical method for designing masks, or filters, for a variety of tasks, such as smoothing, edge detection, corner detection, or even texture detection.

There is a large literature on edge detection, and several different schemes were covered in this chapter. It should be clear that specifc edge detector output can be very useful for particular machine vision tasks. The dream of many developers has not been realized, however — to produce a single uniform solution to the low level vision problem of representing significant object boundaries by some description based on detected edges. Perhaps it is unrealistic to expect this. After all, given an image of a car, how does the low level system know whether or not a car is present, is of interest, is moving or still, or whether we are interested in inspecting for scratches in its paint or the fit of its doors versus just recognizing the make? Many of the edgemaps extracted in our examples seemed very promising for various applications. Indeed they are – many methods of subsequent chapters will be based on representations built using edge input. The reader should not be overly optimistic, however, because the reader has interpeted the images of this chapter using much higher

level organization and knowledge of objects and the world. The higher level methods that we develop must be tolerant, because our edgemaps have gaps, noise, and multiple levels of structure, which will challenge the algorithms that build from them.

5.13 References

Because few of the modern specialized journals existed when early work on image processing and computer vision was done, research papers are scattered across many different journals. Larry Roberts published his seminal thesis work in recognition of 3D block-world objects in 1965. Part of that work was the detection of edges using the operator which now bears his name. Roberts could not have anticipated the large amount of edge detection work which would soon follow. Other early work was published by Prewitt in 1970 and Kirsch in 1971. The Kirsch masks are now regarded as corner detectors rather than edge detectors. Recent work by Shin *et al* (1998) supports the current popularity of the Canny step edge detector (1986) by showing that it is one of the best in both performance and efficiency, at least in a structure-from-motion task. The paper by Huertas and Medioni (1986) takes a deep practical look at implementing LOG filters and shows how to use them to locate edges to subpixel accuracy. The work on edge detection can easily be extended to 3D volume images as has been done by Zucker and Hummel (1981).

The book by Kreider *et al* (1966) gives required linear algebra background focused on its use in the analysis of functions, such as our picture functions: chapters 1,2 and 7 are most relevant. Chapters 9 and 10 give background on Fourier series approximation of 1D signals and are useful background for those wanting to explore the use of Fourier analysis of 2D picture functions. One can consult the optics text by Hecht and Lajac (1974) for Fourier interpretations of images as a superposition of waves. The book on algorithms by Cormen *et al* (1990) gives two algorithms for selecting the i-th smallest of n numbers: the theoretical bound on one of them is $O(n)$ effort, which puts median filtering in the same theoretical complexity class as boxcar filtering.

1. J. Canny (1986), *A computational approach to edge detection*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6 (Nov 1986)679-698.

2. T. Cormen, C. Leiserson, and R. Rivest (1990) *Introduction to Algorithms*, MIT Press, Cambridge, Mass.

3. W. Frei and C-C. Chen (1977) *Fast Boundary Detection: A Generalization and New Algorithm*, IEEE Trans. on Computers, Vol. C-26, No. 10, (October 1977)pp988-998.

4. E. Hecht and A. Lajac (1974), **Optics**, Addison-Wesley. A. Huertas and G. Medioni (1986), *Detection of Intensity Changes with Subpixel Accuracy Using Laplacian-Gaussian Masks* **IEEE-T-PAMI**, Vol. 8, No. 5 (Sept 1986)651-664.

5. R. Kirsch (1971) *Computer Determination of the Constituent Structure of Biological Images*, **Computers and Biomedical Research**, Vol. 4, No. 3 (June 1971) pp315-328.

6. D. Kreider, R. Kuller, D. Ostberg, and F. Perkins (1966) **An Introduction to Linear Analysis**, Addison-Wesley, New York.

7. J. Prewitt (1970) *Object Enhancement and Extraction*, in **Picture Processing and Psychopictorics**, B. Lipkin and A. Rosenfeld (eds), Academic Press, N.Y. (1970)pp75-149.

8. L. Roberts (1965) *Machine Perception of Three-dimensional Solids*, in **Optical and Electro-Optical Information Processing**, J. Tippett *et al* (eds) MIT Press,Cambridge, Mass. (1965)pp159-197.

9. M. Shin, D. Goldgof, and K. Bowyer (1998) *An Objective Comparison Methodology of Edge Detection Algorithms using a Structure from Motion Task*, in **Empirical Evaluation Techniques in Computer Vision**, K. Bowyer and P. Philips, eds., IEEE Computer Society Press, Los Alamitos, CA.

10. S. Zucker and R. Hummel, *A three-dimensional edge operator*, **IEEE-T-PAMI**, Vol.3 (May 1981)324-331.