Digital Image Processing

Second Edition

Instructor's Manual

Rafael C. Gonzalez
Richard E. Woods

Digital Image Processing

Second Edition

Instructor's Manual

Rafael C. Gonzalez
Richard E. Woods

Prentice Hall

Upper Saddle River, NJ 07458

www.prenhall.com/gonzalezwoods

or

www.imageprocessingbook.com

Revision history

10 9 8 7 6 5 4 3 2 1

Preface

This manual contains detailed solutions to all problems in *Digital Image Processing*, 2nd Edition. We also include a suggested set of guidelines for using the book, and discuss the use of computer projects designed to promote a deeper understanding of the subject matter. The notation used throughout this manual corresponds to the notation used in the text.

The decision of what material to cover in a course rests with the instructor, and it depends on the purpose of the course and the background of the students. We have found that the course outlines suggested here can be covered comfortably in the time frames indicated when the course is being taught in an electrical engineering or computer science curriculum. In each case, no prior exposure to image processing is assumed. We give suggested guidelines for one-semester courses at the senior and first-year graduate levels. It is possible to cover most of the book in a two-semester graduate sequence.

The book was completely revised in this edition, with the purpose not only of updating the material, but just as important, making the book a better teaching aid. To this end, the instructor will find the new organization to be much more flexible and better illustrated. Although the book is self contained, we recommend use of the companion web site, where the student will find detailed solutions to the problems marked with a star in the text, review material, suggested projects, and images from the book. One of the principal reasons for creating the web site was to free the instructor from having to prepare materials and handouts beyond what is required to teach from the book.

Computer projects such as those described in the web site are an important part of a course on image processing. These projects give the student hands-on experience with algorithm implementation and reinforce the material covered in the classroom. The projects suggested at the web site can be implemented on almost any reasonably-equipped multi-user or personal computer having a hard copy output device.

1 Introduction

The purpose of this chapter is to present suggested guidelines for teaching material from this book at the senior and first-year graduate level. We also discuss use of the book web site. Although the book is totally self-contained, the web site offers, among other things, complementary review material and computer projects that can be assigned in conjunction with classroom work. Detailed solutions to all problems in the book also are included in the remaining chapters of this manual.

Teaching Features of the Book

Undergraduate programs that offer digital image processing typically limit coverage to one semester. Graduate programs vary, and can include one or two semesters of the material. In the following discussion we give general guidelines for a one-semester senior course, a one-semester graduate course, and a full-year course of study covering two semesters. We assume a 15-week program per semester with three lectures per week. In order to provide flexibility for exams and review sessions, the guidelines discussed in the following sections are based on forty, 50-minute lectures per semester. The background assumed on the part of the student is senior-level preparation in mathematical analysis, matrix theory, probability, and computer programming.

The suggested teaching guidelines are presented in terms of general objectives, and not as time schedules. There is so much variety in the way image processing material is taught that it makes little sense to attempt a breakdown of the material by class period. In particular, the organization of the present edition of the book is such that it makes it much easier than before to adopt significantly different teaching strategies, depending on course objectives and student background. For example, it is possible with the new organization to offer a course that emphasizes spatial techniques and covers little or no transform material. This is not something we recommend, but it is an option that often is attractive in programs that place little emphasis on the signal processing aspects of the field and prefer to focus more on the implementation of spatial techniques.

The companion web site

www.prenhall.com/gonzalezwoods

or

www.imageprocessingbook.com

is a valuable teaching aid, in the sense that it includes material that previously was covered in class. In particular, the review material on probability, matrices, vectors, and linear systems, was prepared using the same notation as in the book, and is focused on areas that are directly relevant to discussions in the text. This allows the instructor to assign the material as independent reading, and spend no more than one total lecture period reviewing those subjects. Another major feature is the set of solutions to problems marked with a star in the book. These solutions are quite detailed, and were prepared with the idea of using them as teaching support. The on-line availability of projects and digital images frees the instructor from having to prepare experiments, data, and handouts for students. The fact that most of the images in the book are available for downloading further enhances the value of the web site as a teaching resource.

One Semester Senior Course

A basic strategy in teaching a senior course is to focus on aspects of image processing in which both the inputs and outputs of those processes are images. In the scope of a senior course, this usually means the material contained in Chapters 1 through 6. Depending on instructor preferences, wavelets (Chapter 7) usually are beyond the scope of coverage in a typical senior curriculum). However, we recommend covering at least some material on image compression (Chapter 8) as outlined below.

We have found in more than two decades of teaching this material to seniors in electrical engineering, computer science, and other technical disciplines, that one of the keys to success is to spend at least one lecture on motivation and the equivalent of one lecture on review of background material, as the need arises. The motivational material is provided in the numerous application areas discussed in Chapter 1. This chapter was totally rewritten with this objective in mind. Some of this material can be covered in class and the rest assigned as independent reading. Background review should cover probability theory (of one random variable) before histogram processing (Section 3.3). A brief review of vectors and matrices may be required later, depending on the material covered. The review material included in the book web site was designed for just this purpose.

Chapter 2 should be covered in its entirety. Some of the material (such as parts of Sections 2.1 and 2.3) can be assigned as independent reading, but a detailed explanation of Sections 2.4 through 2.6 is time well spent.

Chapter 3 serves two principal purposes. It covers image enhancement (a topic of significant appeal to the beginning student) and it introduces a host of basic spatial processing tools used throughout the book. For a senior course, we recommend coverage of Sections 3.2.1 through 3.2.2; Section 3.3.1; Section 3.4; Section 3.5; Section 3.6; Section 3.7.1, 3.7.2 (through Example 3.11), and 3.7.3. Section 3.8 can be assigned as independent reading, depending on time.

Chapter 4 also discusses enhancement, but from a frequency-domain point of view. The instructor has significant flexibility here. As mentioned earlier, it is possible to skip the chapter altogether, but this will typically preclude meaningful coverage of other areas based on the Fourier transform (such as filtering and restoration). The key in covering the frequency domain is to get to the convolution theorem and thus develop a tie between the frequency and spatial domains. All this material is presented in very readable form in Section 4.2. "Light" coverage of frequency-domain concepts can be based on discussing all the material through this section and then selecting a few simple filtering examples (say, low- and highpass filtering using Butterworth filters, as discussed in Sections 4.3.2 and 4.4.2). At the discretion of the instructor, additional material can include full coverage of Sections 4.3 and 4.4. It is seldom possible to go beyond this point in a senior course.

Chapter 5 can be covered as a continuation of Chapter 4. Section 5.1 makes this an easy approach. Then, it is possible give the student a "flavor" of what restoration is (and still keep the discussion brief) by covering only Gaussian and impulse noise in Section 5.2.1, and a couple of spatial filters in Section 5.3. This latter section is a frequent source of confusion to the student who, based on discussions earlier in the chapter, is expecting to see a more objective approach. It is worthwhile to emphasize at this point that spatial enhancement and restoration are the same thing when it comes to noise reduction by spatial filtering. A good way to keep it brief and conclude coverage of restoration is to jump at this point to inverse filtering (which follows directly from the model in Section 5.1) and show the problems with this approach. Then, with a brief explanation regarding the fact that much of restoration centers around the instabilities inherent in inverse filtering, it is possible to introduce the "interactive" form of the Wiener filter in Eq. (5.8-3) and conclude the chapter with Examples 5.12 and 5.13.

Chapter 6 on color image processing is a new feature of the book. Coverage of this

chapter also can be brief at the senior level by focusing on enough material to give the student a foundation on the physics of color (Section 6.1), two basic color models (RGB and CMY/CMYK), and then concluding with a brief coverage of pseudocolor processing (Section 6.3).

We typically conclude a senior course by covering some of the basic aspects of image compression (Chapter 8). Interest on this topic has increased significantly as a result of the heavy use of images and graphics over the Internet, and students usually are easily motivated by the topic. Minimum coverage of this material includes Sections 8.1.1 and 8.1.2, Section 8.2, and Section 8.4.1. In this limited scope, it is worthwhile spending one-half of a lecture period filling in any gaps that may arise by skipping earlier parts of the chapter.

One Semester Graduate Course (No Background in DIP)

The main difference between a senior and a first-year graduate course in which neither group has formal background in image processing is mostly in the scope of material covered, in the sense that we simply go faster in a graduate course, and feel much freer in assigning independent reading. In addition to the material discussed in the previous section, we add the following material in a graduate course.

Coverage of histogram matching (Section 3.3.2) is added. Sections 4.3, 4.4, and 4.5 are covered in full. Section 4.6 is touched upon briefly regarding the fact that implementation of discrete Fourier transform techniques requires non-intuitive concepts such as function padding. The separability of the Fourier transform should be covered, and mention of the advantages of the FFT should be made. In Chapter 5 we add Sections 5.5 through 5.8. In Chapter 6 we add the HSI model (Section 6.3.2) , Section 6.4, and Section 6.6. A nice introduction to wavelets (Chapter 7) can be achieved by a combination of classroom discussions and independent reading. The minimum number of sections in that chapter are 7.1, 7.2, 7.3, and 7.5, with appropriate (but brief) mention of the existence of fast wavelet transforms. Finally, in Chapter 8 we add coverage of Sections 8.3, 8.4.2, 8.5.1 (through Example 8.16), Section 8.5.2 (through Example 8.20) and Section 8.5.3.

If additional time is available, a natural topic to cover next is morphological image processing (Chapter 9). The material in this chapter begins a transition from methods whose inputs and outputs are images to methods in which the inputs are images, but the outputs are attributes about those images, in the sense defined in Section 1.1. We

recommend coverage of Sections 9.1 through 9.4, and some of the algorithms in Section 9.5.

One Semester Graduate Course (with Background in DIP)

Some programs have an undergraduate course in image processing as a prerequisite to a graduate course on the subject. In this case, it is possible to cover material from the first eleven chapters of the book. Using the undergraduate guidelines described above, we add the following material to form a teaching outline for a one semester graduate course that has that undergraduate material as prerequisite. Given that students have the appropriate background on the subject, independent reading assignments can be used to control the schedule.

Coverage of histogram matching (Section 3.3.2) is added. Sections 4,3, 4.4, 4.5, and 4.6 are added. This strengthens the student's background in frequency-domain concepts. A more extensive coverage of Chapter 5 is possible by adding sections 5.2.3, 5.3.3, 5.4.3, 5.5, 5.6, and 5.8. In Chapter 6 we add full-color image processing (Sections 6.4 through 6.7). Chapters 7 and 8 are covered as in the previous section. As noted in the previous section, Chapter 9 begins a transition from methods whose inputs and outputs are images to methods in which the inputs are images, but the outputs are attributes about those images. As a minimum, we recommend coverage of binary morphology: Sections 9.1 through 9.4, and some of the algorithms in Section 9.5. Mention should be made about possible extensions to gray-scale images, but coverage of this material may not be possible, depending on the schedule. In Chapter 10, we recommend Sections 10.1, 10.2.1 and 10.2.2, 10.3.1 through 10.3.4, 10.4, and 10.5. In Chapter 11we typically cover Sections 11.1 through 11.4.

Two Semester Graduate Course (No Background in DIP)

A full-year graduate course consists of the material covered in the one semester undergraduate course, the material outlined in the previous section, and Sections 12.1, 12.2, 12.3.1, and 12.3.2.

Projects

One of the most interesting aspects of a course in digital image processing is the pictorial

nature of the subject. It has been our experience that students truly enjoy and benefit from judicious use of computer projects to complement the material covered in class. Since computer projects are in addition to course work and homework assignments, we try to keep the formal project reporting as brief as possible. In order to facilitate grading, we try to achieve uniformity in the way project reports are prepared. A useful report format is as follows:

Page 1: Cover page.

- Project title
- Project number
- Course number
- Student's name
- Date due
- Date handed in
- Abstract (not to exceed 1/2 page)

Page 2: One to two pages (max) of technical discussion.

Page 3 (or 4): Discussion of results. One to two pages (max).

Results: Image results (printed typically on a laser or inkjet printer). All images must contain a number and title referred to in the discussion of results.

Appendix: Program listings, focused on any original code prepared by the student. For brevity, functions and routines provided to the student are referred to by name, but the code is not included.

Layout: The entire report must be on a standard sheet size (e.g., 8.5×11 inches), stapled with three or more staples on the left margin to form a booklet, or bound using clear plastic standard binding products.

Project resources available in the book web site include a sample project, a list of suggested projects from which the instructor can select, book and other images, and MATLAB functions. Instructors who do not wish to use MATLAB will find additional software suggestions in the Support/Software section of the web site.

2 Problem Solutions

Problem 2.1

The diameter, x, of the retinal image corresponding to the dot is obtained from similar triangles, as shown in Fig. P2.1. That is,

$$\frac{(d/2)}{0.2} = \frac{(x/2)}{0.014}$$

which gives $x = 0.07d$. From the discussion in Section 2.1.1, and taking some liberties of interpretation, we can think of the fovea as a square sensor array having on the order of 337,000 elements, which translates into an array of size 580×580 elements. Assuming equal spacing between elements, this gives 580 elements and 579 spaces on a line 1.5 mm long. The size of each element and each space is then $s = [(1.5\text{mm})/1,159] = 1.3 \times 10^{-6}$ m. If the size (on the fovea) of the imaged dot is less than the size of a single resolution element, we assume that the dot will be invisible to the eye. In other words, the eye will not detect a dot if its diameter, d, is such that $0.07(d) < 1.3 \times 10^{-6}$ m, or $d < 18.6 \times 10^{-6}$ m.

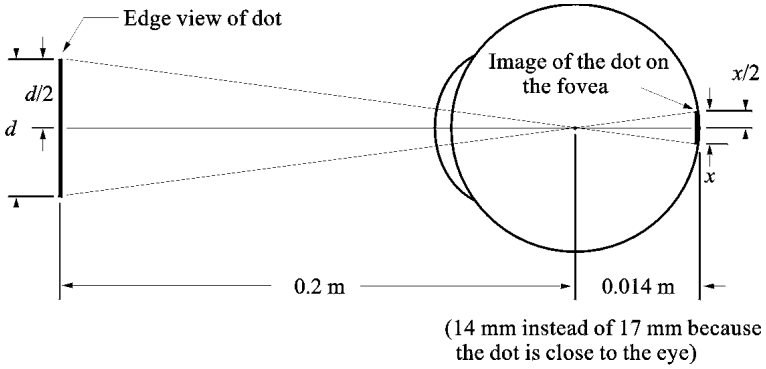


Figure P2.1

Problem 2.2

Brightness adaptation.

Problem 2.3

$\lambda = c/v = 2.998 \times 10^8(m/s)/60$(1/s)$= 4.99 \times 10^6m= 5000$ Km.

Problem 2.4

(a) From the discussion on the electromagnetic spectrum in Section 2.2, the source of the illumination required to see an object must have wavelength the same size or smaller than the object. Because interest lies only on the boundary shape and not on other spectral characteristics of the specimens, a single illumination source in the far ultraviolet (wavelength of .001 microns or less) will be able to detect all objects. A far-ultraviolet camera sensor would be needed to image the specimens. (b) No answer required since the answer to (a) is affirmative.

Problem 2.5

From the geometry of Fig. 2.3, 7mm$/35$mm$= z/500$mm, or $z = 100$ mm. So the target size is 100 mm on the side. We have a total of 1024 elements per line, so the resolution of 1 line is $1024/100 = 10$ elements/mm. For line pairs we divide by 2, giving an answer of 5 lp/mm.

Problem 2.6

One possible solution is to equip a monochrome camera with a mechanical device that sequentially places a red, a green, and a blue pass filter in front of the lens. The strongest camera response determines the color. If all three responses are approximately equal, the object is white. A faster system would utilize three different cameras, each equipped with an individual filter. The analysis would be then based on polling the response of each camera. This system would be a little more expensive, but it would be faster and more reliable. Note that both solutions assume that the field of view of the camera(s) is such that it is completely filled by a uniform color [i.e., the camera(s) is(are) focused on

a part of the vehicle where only its color is seen. Otherwise further analysis would be required to isolate the region of uniform color, which is all that is of interest in solving this problem].

Problem 2.7

The image in question is given by

$$\begin{aligned} f(x,y) &= i(x,y)r(x,y) \\ &= 255e^{-[(x-x_0)^2+(y-y_0)^2]}(1.0) \\ &= 255e^{-[(x-x_0)^2+(y-y_0)^2]} \end{aligned}$$

A cross section of the image is shown in Fig. P2.7(a). If the intensity is quantized using m bits, then we have the situation shown in Fig. P2.7(b), where $\triangle G = (255+1)/2^m$. Since an abrupt change of 8 gray levels is assumed to be detectable by the eye, it follows that $\triangle G = 8 = 256/2m$, or $m = 5$. In other words, 32, or fewer, gray levels will produce visible false contouring.

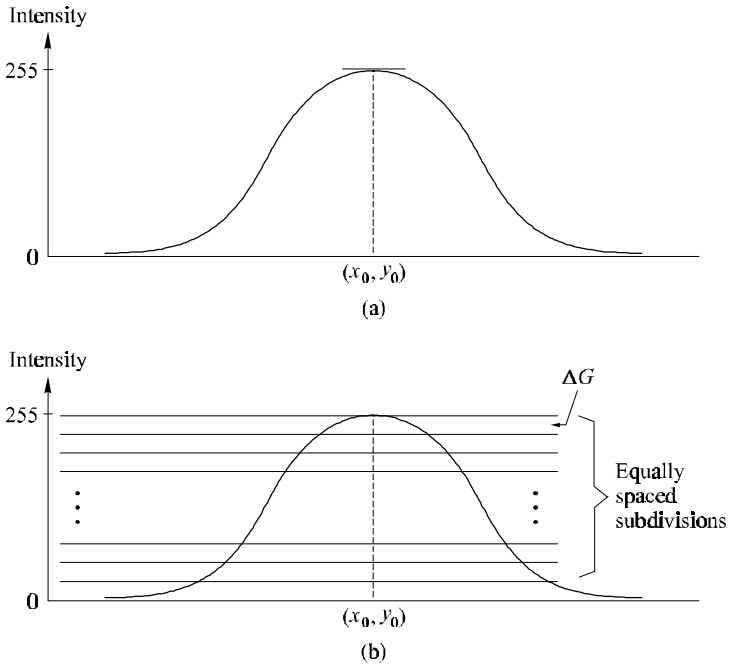


Figure P2.7

Problem 2.8

The use of two bits ($m = 2$) of intensity resolution produces four gray levels in the range 0 to 255. One way to subdivide this range is to let all levels between 0 and 63 be coded as 63, all levels between 64 and 127 be coded as 127, and so on. The image resulting from this type of subdivision is shown in Fig. P2.8. Of course, there are other ways to subdivide the range $[0, 255]$ into four bands.

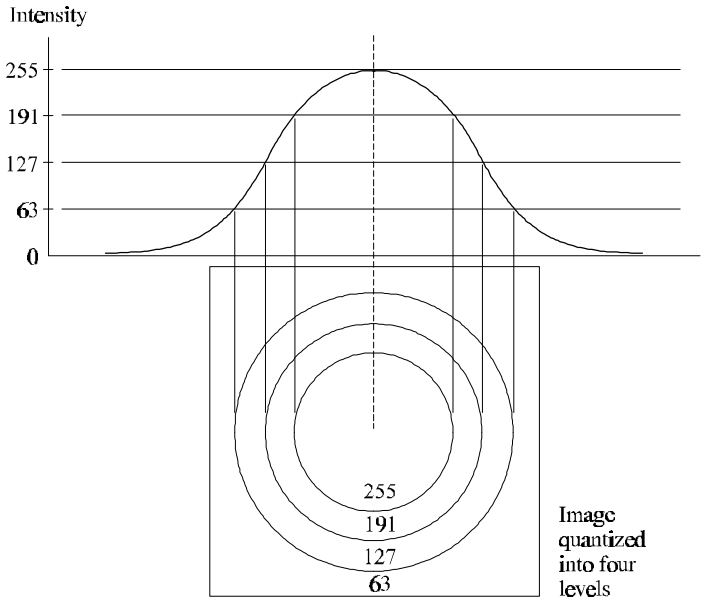


Figure P2.8

Problem 2.9

(a) The total amount of data (including the start and stop bit) in an 8-bit, 1024×1024 image, is $(1024)^2 \times [8 + 2]$ bits. The total time required to transmit this image over a At 56K baud link is $(1024)^2 \times [8 + 2]/56000 = 187.25$ sec or about 3.1 min. (b) At 750K this time goes down to about 14 sec.

Problem 2.10

The width-to-height ratio is 16/9 and the resolution in the vertical direction is 1125 lines (or, what is the same thing, 1125 pixels in the vertical direction). It is given that the

resolution in the horizontal direction is in the 16/9 proportion, so the resolution in the vertical direction is $(1125) \times (16/9) = 2000$ pixels per line. The system "paints" a full 1125×2000, 8-bit image every 1/30 sec for each of the red, green, and blue component images. There are 7200 sec in two hours, so the total digital data generated in this time interval is $(1125)(2000)(8)(30)(3)(7200) = 1.166 \times 10^{13}$ bits, or 1.458×10^{12} bytes (i.e., about 1.5 terrabytes). These figures show why image data compression (Chapter 8) is so important.

Problem 2.11

Let p and q be as shown in Fig. P2.11. Then, (a) S_1 and S_2 are not 4-connected because q is not in the set $N_4(p)$; (b) S_1 and S_2 are 8-connected because q is in the set $N_8(p)$; (c) S_1 and S_2 are *m*-connected because (i) q is in $N_D(p)$, and (ii) the set $N_4(p) \cap N_4(q)$ is empty.

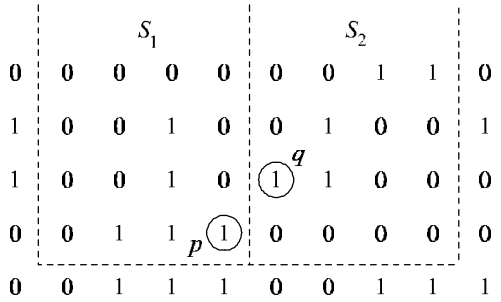


Figure P2.11

Problem 2.12

The solution to this problem consists of defining all possible neighborhood shapes to go from a diagonal segment to a corresponding 4-connected segment, as shown in Fig. P2.12. The algorithm then simply looks for the appropriate match every time a diagonal segment is encountered in the boundary.

Problem 2.13

The solution to this problem is the same as for Problem 2.12 because converting from an *m*-connected path to a 4-connected path simply involves detecting diagonal segments and converting them to the appropriate 4-connected segment.

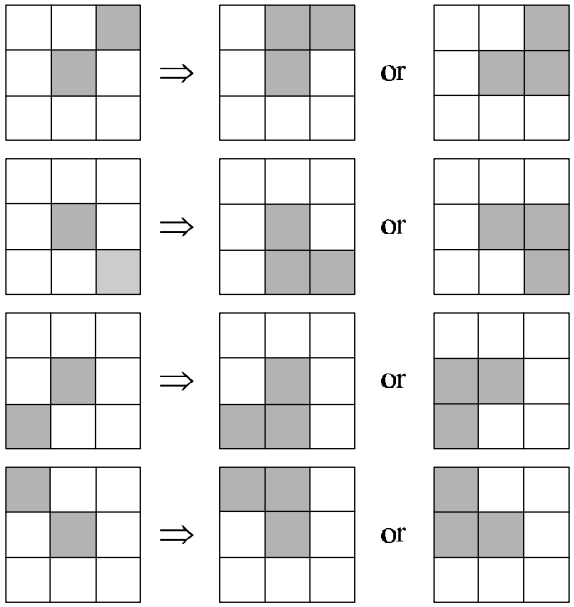


Figure P2.12

Problem 2.14

A region R of an image is composed of a set of connected points in the image. The boundary of a region is the set of points that have one or more neighbors that are not in R. Because boundary points also are part of R, it follows that a point on the boundary has at least one neighbor in R and at least one neighbor not in R. (If the point in the boundary did not have a neighbor in R, the point would be disconnected from R, which violates the definition of points in a region.) Since all points in R are part of a connected component (see Section 2.5.2), all points in the boundary are also connected and a path (entirely in R) exists between any two points on the boundary. Thus the boundary forms a closed path.

Problem 2.15

(a) When $V = \{0, 1\}$, 4-path does not exist between p and q because it is impossible to get from p to q by traveling along points that are both 4-adjacent and also have values from V. Figure P2.15(a) shows this condition; it is not possible to get to q. The shortest 8-path is shown in Fig. P2.15(b); its length is 4. The length of the shortest m- path (shown dashed) is 5. Both of these shortest paths are unique in this case. (b) One

possibility for the shortest 4-path when $V = \{1, 2\}$ is shown in Fig. P2.15(c); its length is 6. It is easily verified that another 4-path of the same length exists between p and q. One possibility for the shortest 8-path (it is not unique) is shown in Fig. P2.15(d); its length is 4. The length of a shortest *m*-path (shown dashed) is 6. This path is not unique.

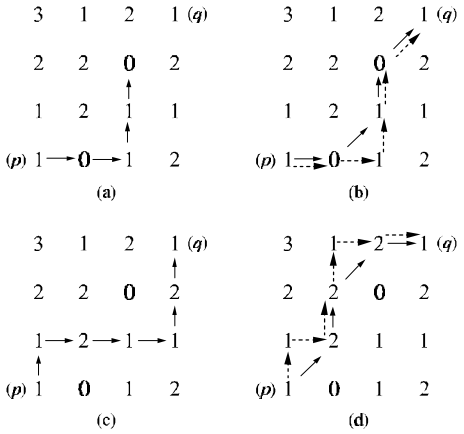


Figure P2.15

Problem 2.16

(a) A shortest 4-path between a point p with coordinates (x, y) and a point q with coordinates (s, t) is shown in Fig. P2.16, where the assumption is that all points along the path are from V. The length of the segments of the path are $|x - s|$ and $|y - t|$, respectively. The total path length is $|x - s| + |y - t|$, which we recognize as the definition of the D_4 distance, as given in Eq. (2.5-16). (Recall that this distance is independent of any paths that may exist between the points.) The D_4 distance obviously is equal to the length of the shortest 4-path when the length of the path is $|x - s| + |y - t|$. This occurs whenever we can get from p to q by following a path whose elements (1) are from V, and (2) are arranged in such a way that we can traverse the path from p to q by making turns in at most two directions (e.g., right and up). (b) The path may of may not be unique, depending on V and the values of the points along the way.

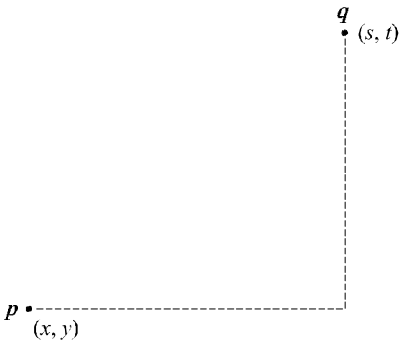


Figure P2.16

Problem 2.17

(a) The D_8 distance between p and q (see Fig. P2.16) is defined as $\max(|x-s|, |y-t|)$. Recall that the D_8 distance (unlike the Euclidean distance) counts diagonal segments the same as horizontal and vertical segments, and, as in the case of the D_4 distance, is independent of whether or not a path exists between p and q. As in the previous problem, the shortest 8-path is equal to the D_8 distance when the path length is $\max(|x-s|, |y-t|)$. This occurs when we can get from p to q by following a path whose elements (1) are from V, and (2) are arranged in such a way that we can traverse the path from p to q by by traveling diagonally in only one direction and, whenever diagonal travel is not possible, by making turns in the horizontal or vertical (but not both) direction. (b) The path may of may not be unique, depending on V and the values of the points along the way.

Problem 2.18

With reference to Eq. (2.6-1), let H denote the neighborhood sum operator, let S_1 and S_2 denote two different small subimage areas of the same size, and let $S_1 + S_2$ denote the corresponding pixel-by-pixel sum of the elements in S_1 and S_2, as explained in Section 2.5.4. Note that the size of the neighborhood (i.e., number of pixels) is not changed by this pixel-by-pixel sum. The operator H computes the sum of pixel values is a given neighborhood. Then, $H(aS_1 + bS_2)$ means: (1) multiplying the pixels in each of the subimage areas by the constants shown, (2) adding the pixel-by-pixel values from S_1 and S_2 (which produces a single subimage area), and (3) computing the sum of the values of all the pixels in that single subimage area. Let ap_1 and bp_2 denote two arbitrary (but

corresponding) pixels from $aS_1 + bS_2$. Then we can write

$$\begin{aligned} H(aS_1 + bS_2) &= \sum_{p_1 \in S_1 \text{ and } p_2 \in S_2} ap_1 + bp_2 \\ &= \sum_{p_1 \in S_1} ap_1 + \sum_{p_2 \in S_2} bp_2 \\ &= a \sum_{p_1 \in S_1} p_1 + b \sum_{p_2 \in S_2} p_2 \\ &= aH(S_1) + bH(S_2) \end{aligned}$$

which, according to Eq. (2.6-1), indicates that H is a linear operator.

Problem 2.19

The median, ζ, of a set of numbers is such that half the values in the set are below ζ and the other half are above it. A simple example will suffice to show that Eq. (2.6-1) is violated by the median operator. Let $S_1 = \{1, -2, 3\}$, $S_2 = \{4, 5, 6\}$, and $a = b = 1$. In this case H is the median operator. We then have $H(S_1 + S_2) =$median$\{5, 3, 9\} = 5$, where it is understood that $S_1 + S_2$ is the element-by-corresponding-element sum of S_1 and S_2. Next, we compute $H(S_1) =$ median$\{1, -2, 3\} = 1$ and $H(S_2) =$ median$\{4, 5, 6\} = 5$. Then, since $H(aS_1 + bS_2) \neq aH(S_1) + bH(S_2)$, it follows that Eq. (2.6-1) is violated and the median is a nonlinear operator.

Problem 2.20

The geometry of the chips is shown in Fig. P2.20(a). From Fig. P2.20(b) and the geometry in Fig. 2.3, we know that

$$\Delta x = \frac{\lambda \times 80}{\lambda - z}$$

where Δx is the side dimension of the image (assumed square since the viewing screen is square) impinging on the image plane, and the 80 mm refers to the size of the viewing screen, as described in the problem statement. The most inexpensive solution will result from using a camera of resolution 512×512. Based on the information in Fig. P2.20(a), a CCD chip with this resolution will be of size $(16\mu) \times (512) = 8$ mm on each side. Substituting $\Delta x = 8$ mm in the above equation gives $z = 9\lambda$ as the relationship between the distance z and the focal length of the lens, where a minus sign was ignored because it is just a coordinate inversion. If a 25 mm lens is used, the front of the lens will have to be located at approximately 225 mm from the viewing screen so that the size of the

image of the screen projected onto the CCD image plane does not exceed the 8 mm size of the CCD chip for the 512×512 camera. This value for z is reasonable, but it is obvious that any of the other given lens sizes would work also; the camera would just have to be positioned further away.

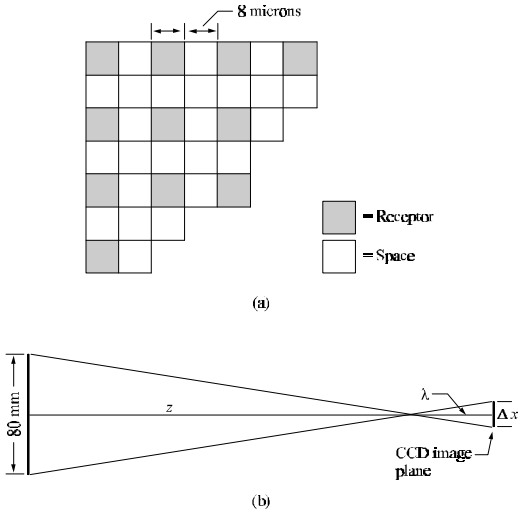


Figure P2.20

Assuming a 25 mm lens, the next issue is to determine if the smallest defect will be imaged on, at least, a 2×2 pixel area, as required by the specification. It is given that the defects are circular, with the smallest defect having a diameter of 0.8 mm. So, all that needs to be done is to determine if the image of a circle of diameter 0.8 mm or greater will, at least, be of size 2×2 pixels on the CCD imaging plane. This can be determined by using the same model as in Fig. P2.20(b) with the 80 mm replaced by 0.8 mm. Using $\lambda = 25$ mm and $z = 225$ mm in the above equation yields $\Delta x = 100\ \mu$. In other words, a circular defect of diameter 0.8 mm will be imaged as a circle with a diameter of 100 μ on the CCD chip of a 512×512 camera equipped with a 25 mm lens and which views the defect at a distance of 225 mm.

If, in order for a CCD receptor to be activated, its area has to be excited in its entirety, then, it can be seen from Fig. P2.20(a) that to guarantee that a 2×2 array of such receptors will be activated, a circular area of diameter no less than $(6)(8) = 48\ \mu$ has to be imaged onto the CCD chip. The smallest defect is imaged as a circle with diameter of 100 μ, which is well above the 48 μ minimum requirement.

Thus, it is concluded that a CCD camera of resolution 512×512 pixels, using a 25 mm lens and imaging the viewing screen at a distance of 225 mm, is sufficient to solve the problem posed by the plant manager.

3 Problem Solutions

Problem 3.1

(a) General form: $s = T(r) = Ae^{-Kr^2}$. For the condition shown in the problem figure, $Ae^{-KL_0^2} = A/2$. Solving for K yields

$$\begin{aligned} -KL_0^2 &= \ln(0.5) \\ K &= 0.693/L_0^2. \end{aligned}$$

Then,

$$s = T(r) = Ae^{-\frac{0.693}{L_0^2}r^2}.$$

(b) General form: $s = T(r) = B(1 - e^{-Kr^2})$. For the condition shown in the problem figure, $B(1 - e^{-KL_0^2}) = B/2$. The solution for K is the same as in (a), so

$$s = T(r) = B(1 - e^{-\frac{0.693}{L_0^2}r^2})$$

(c) General form: $s = T(r) = (D - C)(1 - e^{-Kr^2}) + C$.

Problem 3.2

(a) $s = T(r) = \frac{1}{1+(m/r)^E}$.

(b) See Fig. P3.2.

(c) We want the value of s to be 0 for $r < m$, and s to be 1 for values of $r > m$. When $r = m$, $s = 1/2$. But, because the values of r are integers, the behavior we want is

$$s = T(r) = \begin{cases} 0.0 & \text{when } r \leq m - 1 \\ 0.5 & \text{when } r = m \\ 1.0 & \text{when } r \geq m + 1. \end{cases}$$

The question in the problem statement is to find the smallest value of E that will make the threshold behave as in the equation above. When $r = m$, we see from (a) that $s = 0.5$, regardless of the value of E. If C is the smallest positive number representable

in the computer, and keeping in mind that s is positive, then any value of s less than $C/2$ will be called 0 by the computer. To find out the smallest value of E for which this happens, simply solve the following equation for E, using the given value $m = 128$:

$$\frac{1}{1 + [m/(m-1)]^E} < C/2.$$

Because the function is symmetric about m, the resulting value of E will yield $s = 1$ for $r \geq m + 1$.

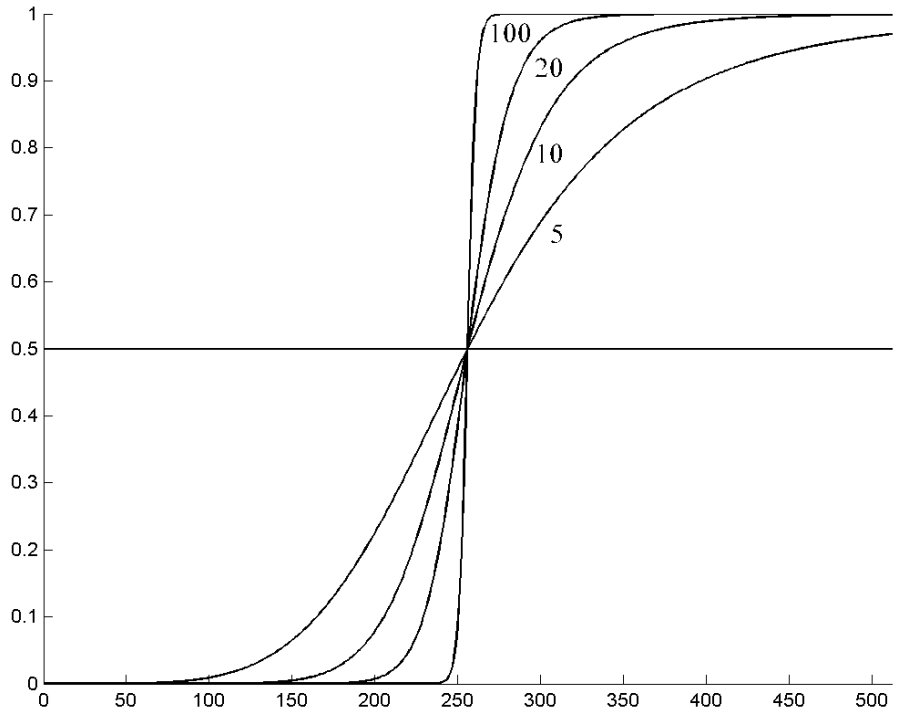


Figure P3.2

Problem 3.3

The transformations required to produce the individual bit planes are nothing more than mappings of the truth table for eight binary variables. In this truth table, the values of the 7th bit are 0 for byte values 0 to 127, and 1 for byte values 128 to 255, thus giving the transformation mentioned in the problem statement. Note that the given transformed values of either 0 or 255 simply indicate a binary image for the 7th bit plane. Any other two values would have been equally valid, though less conventional.

Continuing with the truth table concept, the transformation required to produce an image of the 6th bit plane outputs a 0 for byte values in the range [0, 63], a 1 for byte values in the range [64, 127], a 0 for byte values in the range [128, 191], and a 1 for byte values in the range [192, 255]. Similarly, the transformation for the 5th bit plane alternates between eight ranges of byte values, the transformation for the 4th bit plane alternates between 16 ranges, and so on. Finally, the output of the transformation for the 0th bit plane alternates between 0 and 255 depending as the byte values are even or odd. Thus, this transformation alternates between 128 byte value ranges, which explains why an image of the 0th bit plane is usually the busiest looking of all the bit plane images.

Problem 3.4

(a) The number of pixels having different gray level values would decrease, thus causing the number of components in the histogram to decrease. Since the number of pixels would not change, this would cause the height some of the remaining histogram peaks to increase in general. Typically, less variability in gray level values will reduce contrast.
(b) The most visible effect would be significant darkening of the image. For example, dropping the highest bit would limit to 127 the brightest level in an 8-bit image. Since the number of pixels would remain constant, the height of some of the histogram peaks would increase. The general shape of the histogram would now be taller and narrower, with no histogram components being located past 127.

Problem 3.5

All that histogram equalization does is remap histogram components on the intensity scale. To obtain a uniform (flat) histogram would require in general that pixel intensities be actually redistributed so that there are L groups of n/L pixels with the same intensity, where L is the number of allowed discrete intensity levels and n is the total number of pixels in the input image. The histogram equalization method has no provisions for this type of (artificial) redistribution process.

Problem 3.6

Let n be the total number of pixels and let n_{r_j} be the number of pixels in the input image

with intensity value r_j. Then, the histogram equalization transformation is

$$s_k = T(r_k) = \sum_{j=0}^{k} n_{r_j}/n = \frac{1}{n}\sum_{j=0}^{k} n_{r_j}.$$

Since every pixel (and no others) with value r_k is mapped to value s_k, it follows that $n_{s_k} = n_{r_k}$. A second pass of histogram equalization would produce values v_k according to the transformation

$$v_k = T(s_k) = \frac{1}{n}\sum_{j=0}^{k} n_{s_j}.$$

But, $n_{s_j} = n_{r_j}$, so

$$v_k = T(s_k) = \frac{1}{n}\sum_{j=0}^{k} n_{r_j} = s_k$$

which shows that a second pass of histogram equalization would yield the same result as the first pass. We have assumed negligible round-off errors.

Problem 3.7

The general histogram equalization transformation function is

$$s = T(r) = \int_0^r p_r(w)\, dw.$$

There are two important points to which the student must show awareness in answering this problem. First, this equation assumes only positive values for r. However, the Gaussian density extends in general from $-\infty$ to ∞. Recognition of this fact is important. Once recognized, the student can approach this difficulty in several ways. One good answer is to make some assumption, such as the standard deviation being small enough so that the area of the curve under $p_r(r)$ for negative values of r is negligible. Another is to scale up the values until the area under the negative tail is negligible. The second major point is to recognize is that the transformation function itself,

$$s = T(r) = \frac{1}{\sqrt{2\pi}\sigma}\int_0^r e^{-\frac{(w-m)^2}{2\sigma^2}} dw$$

has no closed-form solution. This is the cumulative distribution function of the Gaussian density, which is either integrated numerically, or its values are looked up in a table. A third, less important point, that the student should address is the high-end values of r. Again, the Gaussian PDF extends to $+\infty$. One possibility here is to make the same

assumption as above regarding the standard deviation. Another is to divide by a large enough value so that the area under the positive tail past that point is negligible (this scaling reduces the standard deviation).

Another principal approach the student can take is to work with histograms, in which case the transformation function would be in the form of a summation. The issue of negative and high positive values must still be addressed, and the possible answers suggested above regarding these issues still apply. The student needs to indicate that the histogram is obtained by sampling the continuous function, so some mention should be made regarding the number of samples (bits) used. The most likely answer is 8 bits, in which case the student needs to address the scaling of the function so that the range is $[0,\ 255]$.

Problem 3.8

We are interested in just one example in order to satisfy the statement of the problem. Consider the probability density function shown in Fig. P3.8(a). A plot of the transformation $T(r)$ in Eq. (3.3-4) using this particular density function is shown in Fig. P3.8(b). Because $p_r(r)$ is a probability density function we know from the discussion in Section 3.3.1 that the transformation $T(r)$ satisfies conditions (a) and (b) stated in that section. However, we see from Fig. P3.8(b) that the inverse transformation from s back to r is not single valued, as there are an infinite number of possible mappings from $s = 1/2$ back to r. It is important to note that the reason the inverse transformation function turned out not to be single valued is the gap in $p_r(r)$ in the interval $[1/4,\ 3/4]$.

Problem 3.9

(a) We need to show that the transformation function in Eq. (3.3-8) is monotonic, single-valued, and that its values are in the range [0, 1]. From Eq. (3.3-8),

$$\begin{aligned} s_k &= T(r_k) = \sum_{j=0}^{k} p_r(rj) \\ &= \sum_{j=0}^{k} \frac{nj}{n} \qquad k = 0, 1, \ldots, L-1. \end{aligned}$$

Because all the $p_r(r_j)$ are positive, it follows that $T(r_k)$ is monotonic. Because all the $p_r(r_j)$ are finite, and the limit of summation is finite, it follows that $T(r_k)$ is of finite

slope and thus us a single-valued function. Finally, since the sum of all the $p_r(r_j)$ is 1, it follows that $0 \leq s_k \leq 1$.

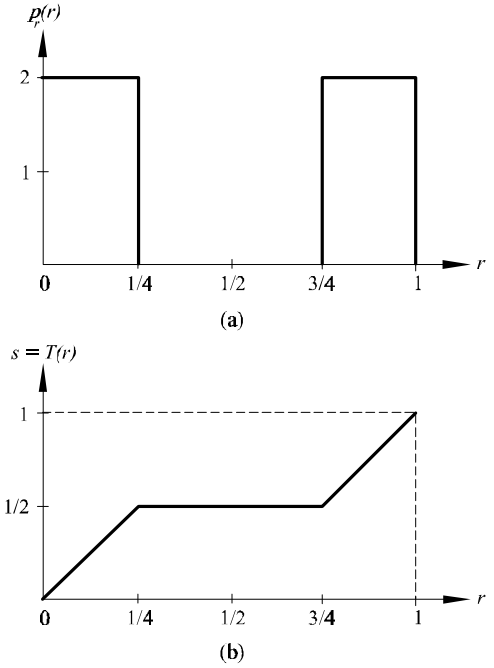


Figure P3.8.

(b) From the discussion in Problem 3.8, it follows that if an image has missing gray levels the histogram equalization transformation function given above will be constant in the interval of the missing gray levels. Thus, in theory, the inverse mapping will not be single-valued in the discrete case either. In practice, assuming that we wanted to perform the inverse transformation, this is not important for the following reason: Assume that no gray-level values exist in the open interval (a, b), so that r_a is the last gray level before the empty gray-level band begins and r_b is the first gray level right after the empty band ends. The corresponding mapped gray levels are s_a and s_b. The fact that no gray levels r exist in interval (a, b) means that no gray levels will exist between s_a and s_b either, and, therefore, there will be no levels s to map back to r in the bands where the multi-valued inverse function would present problems. Thus, in practice, the issue of the inverse not being single-valued is not an issue since it would not be needed. Note that mapping back from s_a and s_b presents no problems, since $T(r_a)$ and $T(r_b)$ (and thus their inverses) are different. A similar discussion applies if there are more than one band empty of gray levels.

(c) If none of the gray levels $r_k,\ k = 1, 2, \ldots, L-1$, are 0, then $T(r_k)$ will be strictly monotonic. This implies that the inverse transformation will be of finite slope and this will be single-valued.

Problem 3.10

First, we obtain the histogram equalization transformation:

$$s = T(r) = \int_0^r p_r(w)\, dw = \int_0^r (-2w + 2)\, dw = -r^2 + 2r.$$

Next we find

$$v = G(z) = \int_0^z p_z(w)\ dw = \int_0^z 2w\ dw = z^2.$$

Finally,

$$z = G^{-1}(v) = \pm\sqrt{v}.$$

But only positive gray levels are allowed, so $z = \sqrt{v}$. Then, we replace v with s, which in turn is $-r^2 + 2r$, and we have

$$z = \sqrt{-r^2 + 2r}.$$

Problem 3.11

The value of the histogram component corresponding to the *k*th intensity level in a neighborhood is

$$p_r(r_k) = \frac{n_k}{n}$$

for $k = 1, 2, \ldots, K-1$,where n_k is the number of pixels having gray level value r_k, n is the total number of pixels in the neighborhood, and K is the total number of possible gray levels. Suppose that the neighborhood is moved one pixel to the right. This deletes the leftmost column and introduces a new column on the right. The updated histogram then becomes

$$p'_r(r_k) = \frac{1}{n}[n_k - n_{L_k} + n_{R_k}]$$

for $k = 0, 1, \ldots, K-1$, where n_{L_k} is the number of occurrences of level r_k on the left column and n_{R_k} is the similar quantity on the right column. The preceding equation can

be written also as

$$p'_r(r_k) = p_r(r_k) + \frac{1}{n}[n_{R_k} - n_{L_k}]$$

for $k = 0, 1, \ldots, K-1$. The same concept applies to other modes of neighborhood motion:

$$p'_r(r_k) = p_r(r_k) + \frac{1}{n}[b_k - a_k]$$

for $k = 0, 1, \ldots, K-1$, where a_k is the number of pixels with value r_k in the neighborhood area deleted by the move, and b_k is the corresponding number introduced by the move.

Problem 3.12

The purpose of this simple problem is to make the student think of the meaning of histograms and arrive at the conclusion that histograms carry no information about spatial properties of images. Thus, the only time that the histogram of the images formed by the operations shown in the problem statement can be determined in terms of the original histograms is when one or both of the images is (are) constant. In (d) we have the additional requirement that none of the pixels of $g(x, y)$ can be 0. Assume for convenience that the histograms are not normalized, so that, for example, $h_f(r_k)$ is the number of pixels in $f(x, y)$ having gray level r_k, assume that all the pixels in $g(x, y)$ have constant value c. The pixels of both images are assumed to be positive. Finally, let u_k denote the gray levels of the pixels of the images formed by any of the arithmetic operations given in the problem statement. Under the preceding set of conditions, the histograms are determined as follows:

(a) The histogram $h_{\text{sum}}(u_k)$ of the sum is obtained by letting $u_k = r_k + c$, and $h_{\text{sum}}(u_k) = h_f(r_k)$ for all k. In other words, the values (height) of the components of h_{sum} are the same as the components of h_f, but their locations on the gray axis are shifted right by an amount c.

(b) Similarly, the histogram $h_{\text{diff}}(u_k)$ of the difference has the same components as h_f but their locations are moved left by an amount c as a result of the subtraction operation.

(c) Following the same reasoning, the values (heights) of the components of histogram $h_{\text{prod}}(u_k)$ of the product are the same as h_f, but their locations are at $u_k = c \times r_k$. Note that while the spacing between components of the resulting histograms in (a) and (b) was not affected, the spacing between components of $h_{\text{prod}}(u_k)$ will be spread out by an amount c.

(d) Finally, assuming that $c \neq 0$, the components of $h_{\text{div}}(u_k)$ are the same as those of h_f, but their locations will be at $u_k = r_k/c$. Thus, the spacing between components of $h_{\text{div}}(u_k)$ will be compressed by an amount equal to $1/c$.

The preceding solutions are applicable if image $f(x, y)$ also is constant. In this case the four histograms just discussed would each have only one component. Their location would be affected as described (a) through (c).

Problem 3.13

Using 10 bits (with one bit being the sign bit) allows numbers in the range -511 to 511. The process of repeated subtractions can be expressed as

$$\begin{aligned} d_K(x,y) &= a(x,y) - \sum_{k=1}^{K} b(x,y) \\ &= a(x,y) - K \times b(x,y) \end{aligned}$$

where K is the largest value such that $d_K(x, y)$ does not exceed -511 at any coordinates (x, y), at which time the subtraction process stops. We know nothing about the images, only that both have values ranging from 0 to 255. Therefore, all we can determine are the maximum and minimum number of times that the subtraction can be carried out and the possible range of gray-level values in each of these two situations.

Because it is given that $g(x, y)$ has at least one pixel valued 255, the maximum value that K can have before the subtraction exceeds -511 is 3. This condition occurs when, at some pair of coordinates (s, t), $a(s, t) = b(s, t) = 255$. In this case, the possible range of values in the difference image is -510 to 255. The latter condition can occur if, at some pair of coordinates (i, j), $a(i, j) = 255$ and $b(i, j) = 0$.

The minimum value that K will have is 2, which occurs when, at some pair of coordinates, $a(s, t) = 0$ and $b(s, t) = 255$. In this case, the possible range of values in the difference image again is -510 to 255. The latter condition can occur if, at some pair of coordinates (i, j), $a(i, j) = 255$ and $b(i, j) = 0$.

Problem 3.14

Let $g(x, y)$ denote the golden image, and let $f(x, y)$ denote any input image acquired during routine operation of the system. Change detection via subtraction is based on computing the simple difference $d(x, y) = g(x, y) - f(x, y)$. The resulting image

$d(x, y)$ can be used in two fundamental ways for change detection. One way is use a pixel-by-pixel analysis. In this case we say that $f(x, y)$ is "close enough" to the golden image if all the pixels in $d(x, y)$ fall within a specified threshold band $[T_{min}, T_{max}]$ where T_{min} is negative and T_{max} is positive. Usually, the same value of threshold is used for both negative and positive differences, in which case we have a band $[-T, T]$ in which all pixels of $d(x, y)$ must fall in order for $f(x, y)$ to be declared acceptable. The second major approach is simply to sum all the pixels in $|d(x, y)|$ and compare the sum against a threshold S. Note that the absolute value needs to be used to avoid errors cancelling out. This is a much cruder test, so we will concentrate on the first approach.

There are three fundamental factors that need tight control for difference-based inspection to work: (1) proper registration, (2) controlled illumination, and (3) noise levels that are low enough so that difference values are not affected appreciably by variations due to noise. The first condition basically addresses the requirement that comparisons be made between corresponding pixels. Two images can be identical, but if they are displaced with respect to each other, comparing the differences between them makes no sense. Often, special markings are manufactured into the product for mechanical or image-based alignment

Controlled illumination (note that "illumination" is not limited to visible light) obviously is important because changes in illumination can affect dramatically the values in a difference image. One approach often used in conjunction with illumination control is intensity scaling based on actual conditions. For example, the products could have one or more small patches of a tightly controlled color, and the intensity (and perhaps even color) of each pixels in the entire image would be modified based on the actual versus expected intensity and/or color of the patches in the image being processed.

Finally, the noise content of a difference image needs to be low enough so that it does not materially affect comparisons between the golden and input images. Good signal strength goes a long way toward reducing the effects of noise. Another (sometimes complementary) approach is to implement image processing techniques (e.g., image averaging) to reduce noise.

Obviously there are a number if variations of the basic theme just described. For example, additional intelligence in the form of tests that are more sophisticated than pixel-by-pixel threshold comparisons can be implemented. A technique often used in this regard is to subdivide the golden image into different regions and perform different (usually more than one) tests in each of the regions, based on expected region content.

Problem 3.15

(a) From Eq. (3.4-3), at any point (x, y),

$$\overline{g} = \frac{1}{K}\sum_{i=1}^{K} g_i = \frac{1}{K}\sum_{i=1}^{K} f_i + \frac{1}{K}\sum_{i=1}^{K} \eta_i.$$

Then

$$E\{\overline{g}\} = \frac{1}{K}\sum_{i=1}^{K} E\{f_i\} + \frac{1}{K}\sum_{i=1}^{K} E\{\eta_i\}.$$

But all the f_i are the same image, so $E\{f_i\} = f$. Also, it is given that the noise has zero mean, so $E\{\eta_i\} = 0$. Thus, it follows that $E\{\overline{g}\} = f$, which proves the validity of

Eq. (3.4-4).

(b) From (a),

$$\overline{g} = \frac{1}{K}\sum_{i=1}^{K} g_i = \frac{1}{K}\sum_{i=1}^{K} f_i + \frac{1}{K}\sum_{i=1}^{K} \eta_i.$$

It is known from random-variable theory that the variance of the sum of uncorrelated random variables is the sum of the variances of those variables (Papoulis [1991]). Since the elements of f are constant and the η_i are uncorrelated, then

$$\sigma^2_{\overline{g}} = \sigma^2_f + \frac{1}{K^2}[\sigma^2_{\eta_1} + \sigma^2_{\eta_2} + \cdots + \sigma^2_{\eta_K}].$$

The first term on the right side is 0 because the elements of f are constants. The various $\sigma^2_{\eta_i}$ are simply samples of the noise, which is has variance σ^2_η. Thus, $\sigma^2_{\eta_i} = \sigma^2_\eta$ and we have

$$\sigma^2_{\overline{g}} = \frac{K}{K^2}\sigma^2_\eta = \frac{1}{K}\sigma^2_\eta$$

which proves the validity of Eq. (3.4-5).

Problem 3.16

With reference to Section 3.4.2, when $i = 1$ (no averaging), we have

$$\overline{g}(1) = g_1 \text{ and } \sigma^2_{\overline{g}(1)} = \sigma^2_\eta.$$

When $i = K$,

$$\overline{g}(K) = \frac{1}{K}\sum_{i=1}^{K} g_i \quad \text{and} \quad \sigma^2_{\overline{g}(K)} = \frac{1}{K}\sigma^2_\eta.$$

We want the ratio of $\sigma^2_{\bar{g}(K)}$ to $\sigma^2_{\bar{g}(1)}$ to be 1/10, so

$$\frac{\sigma^2_{\bar{g}(K)}}{\sigma^2_{\bar{g}(1)}} = \frac{1}{10} = \frac{\frac{1}{K}\sigma^2_\eta}{\sigma^2_\eta}$$

from which we get $K = 10$. Since the images are generated at 30 frames/s, the stationary time required is 1/3 s.

Problem 3.17

(a) Consider a 3×3 mask first. Since all the coefficients are 1 (we are ignoring the 1/9 scale factor), the net effect of the lowpass filter operation is to add all the gray levels of pixels under the mask. Initially, it takes 8 additions to produce the response of the mask. However, when the mask moves one pixel location to the right, it picks up only one new column. The new response can be computed as

$$R_{\text{new}} = R_{\text{old}} - C_1 + C_3$$

where C_1 is the sum of pixels under the first column of the mask before it was moved, and C_3 is the similar sum in the column it picked up after it moved. This is the basic box-filter or moving-average equation. For a 3×3 mask it takes 2 additions to get C_3 (C_1 was already computed). To this we add one subtraction and one addition to get R_{new}. Thus, a total of 4 arithmetic operations are needed to update the response after one move. This is a recursive procedure for moving from left to right along one row of the image. When we get to the end of a row, we move down one pixel (the nature of the computation is the same) and continue the scan in the opposite direction.

For a mask of size $n \times n$, $(n-1)$ additions are needed to obtain C_3, plus the single subtraction and addition needed to obtain R_{new}, which gives a total of $(n+1)$ arithmetic operations after each move. A brute-force implementation would require $n^2 - 1$ additions after each move.

(b) The computational advantage is

$$A = \frac{n^2 - 1}{n+1} = \frac{(n+1)(n-1)}{(n+1)} = n - 1.$$

The plot of A as a function of n is a simple linear function starting at $A = 1$ for $n = 2$.

Problem 3.18

One of the easiest ways to look at repeated applications of a spatial filter is to use super-

position. Let $f(x,\, y)$ and $h(x,\, y)$ denote the image and the filter function, respectively. Assuming square images of size $N \times N$ for convenience, we can express $f(x,\, y)$ as the sum of at most N^2 images, each of which has only one nonzero pixel (initially, we assume that N can be infinite). Then, the process of running $h(x,\, y)$ over $f(x,\, y)$ can be expressed as the following convolution:

$$h(x,y) * f(x,y) = h(x,y) * [f_1(x,y) + f_2(x,y) + \cdots f_{N^2}(x,y)]\,.$$

Suppose for illustrative purposes that $f_i(x,y)$ has value 1 at its center, while the other pixels are valued 0, as discussed above (see Fig. P3.18a). If $h(x,y)$ is a 3×3 mask of 1/9's (Fig. P3.18b), then convolving $h(x,y)$ with $f_i(x,y)$ will produce an image with a 3×3 array of 1/9's at its center and 0's elsewhere, as shown in Fig. P3.18(c). If $h(x,y)$ is now applied to this image, the resulting image will be as shown in Fig. P3.18(d). Note that the sum of the nonzero pixels in both Figs. P3.18(c) and (d) is the same, and equal to the value of the original pixel. Thus, it is intuitively evident that successive applications of $h(x,y)$ will "diffuse" the nonzero value of $f_i(x,y)$ (not an unexpected result, because $h(x,y)$ is a blurring filter). Since the sum remains constant, the values of the nonzero elements will become smaller and smaller, as the number of applications of the filter increases. The overall result is given by adding all the convolved $f_k(x,y)$, for $k = 1, 2, ..., N^2$. The net effect of successive applications of the lowpass spatial filter $h(x,y)$ is thus seen to be more and more blurring, with the value of each pixel "redistributed" among the others. The average value of the blurred image will be thus be the same as the average value of $f(x,y)$.

It is noted that every iteration of blurring further diffuses the values outwardly from the starting point. In the limit, the values would get infinitely small, but, because the average value remains constant, this would require an image of infinite spatial proportions. It is at this junction that border conditions become important. Although it is not required in the problem statement, it is instructive to discuss in class the effect of successive applications of $h(x,y)$ to an image of finite proportions. The net effect is that, since the values cannot diffuse outward past the boundary of the image, the denominator in the successive applications of averaging eventually overpowers the pixel values, driving the image to zero in the limit. A simple example of this is given in Fig. P3.18(e), which shows an array of size 1×7 that is blurred by successive applications of the 1×3 mask $h(y) = \frac{1}{3}[1, 1, 1]$. We see that, as long as the values of the blurred 1 can diffuse out, the sum, S, of the resulting pixels is 1. However, when the boundary is met, an assumption must be made regarding how mask operations on the border are treated. Here, we used the commonly made assumption that pixel value immediately past the boundary are 0. The mask operation does not go beyond the boundary, however. In this example, we

see that the sum of the pixel values begins to decrease with successive applications of the mask. In the limit, the term $1/(3)^n$ would overpower the sum of the pixel values, yielding an array of 0's.

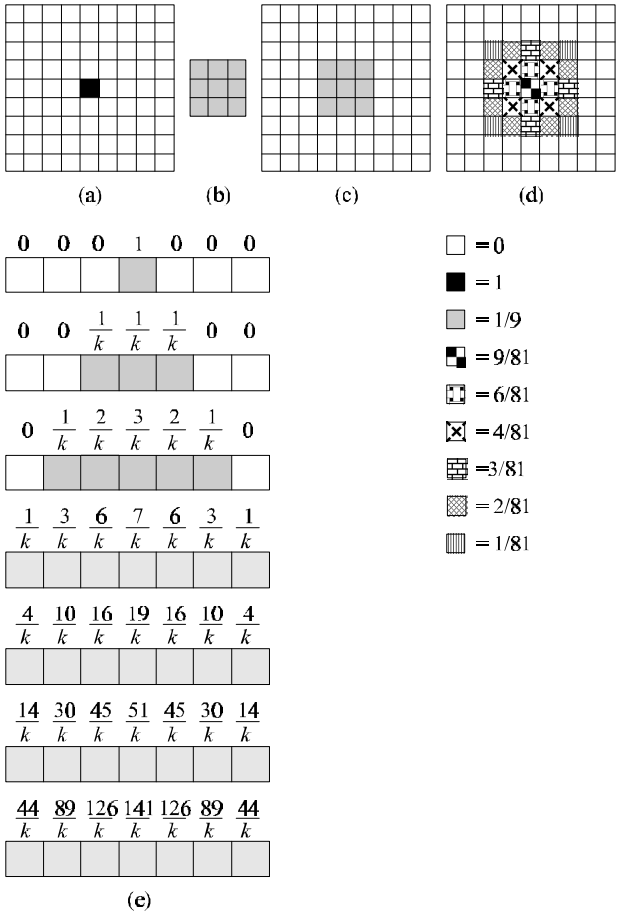


Figure P3.18

Problem 3.19

(a) There are n^2 points in an $n \times n$ median filter mask. Since n is odd, the median value, ζ, is such that there are $(n^2 - 1)/2$ points with values less than or equal to ζ and the same number with values greater than or equal to ζ. However, since the area A (number of points) in the cluster is less than one half n^2, and A and n are integers, it follows that A is always less than or equal to $(n^2 - 1)/2$. Thus, even in the extreme case when all cluster points are encompassed by the filter mask, there are not enough

points in the cluster for any of them to be equal to the value of the median (remember, we are assuming that all cluster points are lighter or darker than the background points). Therefore, if the center point in the mask is a cluster point, it will be set to the median value, which is a background shade, and thus it will be "eliminated" from the cluster. This conclusion obviously applies to the less extreme case when the number of cluster points encompassed by the mask is less than the maximum size of the cluster.

(b) For the conclusion reached in (a) to hold, the number of points that we consider cluster (object) points can never exceed $(n^2-1)/2$. Thus, two or more different clusters cannot be in close enough proximity for the filter mask to encompass points from more than one cluster at any mask position. It then follows that no two points from different clusters can be closer than the diagonal dimension of the mask minus one cell (which can be occupied by a point from one of the clusters). Assuming a grid spacing of 1 unit, the minimum distance between any two points of different clusters then must greater than $\sqrt{2}(n-1)$. In other words, these points must be separated by at least the distance spanned by $n-1$ cells along the mask diagonal.

Problem 3.20

(a) Numerically sort the n^2 values. The median is

$$\zeta = [(n^2+1)/2]\text{-th largest value.}$$

(b) Once the values have been sorted one time, we simply delete the values in the trailing edge of the neighborhood and insert the values in the leading edge in the appropriate locations in the sorted array.

Problem 3.21

(a) The most extreme case is when the mask is positioned on the center pixel of a 3-pixel gap, along a thin segment, in which case a 3×3 mask would encompass a completely blank field. Since this is known to be the largest gap, the next (odd) mask size up is guaranteed to encompass some of the pixels in the segment. Thus, the smallest mask that will do the job is a 5×5 averaging mask.

(b) The smallest average value produced by the mask is when it encompasses only two pixels of the segment. This average value is a gray-scale value, not binary, like the rest of the segment pixels. Denote the smallest average value by A_{min}, and the binary values

of pixels in the thin segment by B. Clearly, A_{min} is less than B. Then, setting the binarizing threshold slightly smaller than A_{min} will create one binary pixel of value B in the center of the mask.

Problem 3.22

From Fig. 3.35, the vertical bars are 5 pixels wide, 100 pixels high, and their separation is 20 pixels. The phenomenon in question is related to the horizontal separation between bars, so we can simplify the problem by considering a single scan line through the bars in the image. The key to answering this question lies in the fact that the distance (in pixels) between the onset of one bar and the onset of the next one (say, to its right) is 25 pixels. Consider the scan line shown in Fig. P3.22. Also shown is a cross section of a 25×25 mask. The response of the mask is the average of the pixels that it encompasses. We note that when the mask moves one pixel to the right, it loses on value of the vertical bar on the left, but it picks up an identical one on the right, so the response doesn't change. In fact, the number of pixels belonging to the vertical bars and contained within the mask does not change, regardless of where the mask is located (as long as it is contained within the bars, and not near the edges of the set of bars). The fact that the number of bar pixels under the mask does not change is due to the peculiar separation between bars and the width of the lines in relation to the 25-pixel width of the mask This constant response is the reason no white gaps is seen in the image shown in the problem statement. Note that this constant response does not happen with the 23×23 or the 45×45 masks because they are not "synchronized" with the width of the bars and their separation.

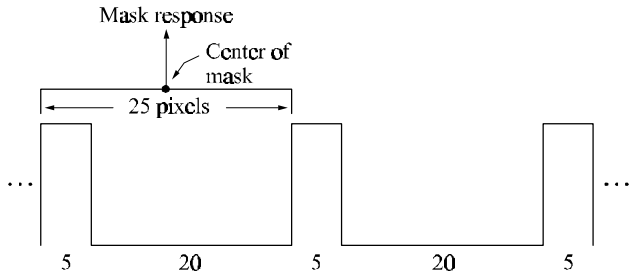


Figure P3.22

Problem 3.23

There are at most q^2 points in the area for which we want to reduce the gray level of each pixel to one-tenth its original value. Consider an averaging mask of size $n \times n$ encompassing the $q \times q$ neighborhood. The averaging mask has n^2 points of which we are assuming that q^2 points are from the object and the rest from the background. Note that this assumption implies separation between objects at least the area of the mask all around each object. The problem becomes intractable unless this assumption is made. This condition was not given in the problem statement on purpose in order to force the student to arrive at that conclusion. If the instructor wishes to simplify the problem, this should then be mentioned when the problem is assigned. A further simplification is to tell the students that the gray level of the background is 0.

Let B represent the gray level of background pixels, let a_i denote the gray levels of points inside the mask and o_i the levels of the objects. In addition, let S_a denote the set of points in the averaging mask, S_o the set of points in the object, and S_b the set of points in the mask that are not object points. Then, the response of the averaging mask at any point on the image can be written as

$$\begin{aligned} R &= \frac{1}{n^2}\sum_{a_i \in S_a} a_i \\ &= \frac{1}{n^2}\left[\sum_{o_j \in S_o} o_j + \sum_{a_k \in S_b} a_k\right] \\ &= \frac{1}{n^2}\left[\frac{q^2}{q^2}\sum_{o_j \in S_o} o_j\right] + \frac{1}{n^2}\left[\sum_{a_k \in S_b} a_k\right] \\ &= \frac{q^2}{n^2}\overline{Q} + \frac{1}{n^2}\left[(n^2 - q^2)B\right] \end{aligned}$$

where $\overline{Q}$ denotes the average value of object points. Let the maximum expected average value of object points be denoted by $\overline{Q}_{\max}$. Then we want the response of the mask at any point on the object under this maximum condition to be less than one-tenth $\overline{Q}_{\max}$, or

$$\frac{q^2}{n^2}\overline{Q}_{\max} + \frac{1}{n^2}\left[(n^2 - q^2)B\right] < \frac{1}{10}\overline{Q}_{\max}$$

from which we get the requirement

$$n > q\left[\frac{10(\overline{Q}_{\max} - B)}{(\overline{Q}_{\max} - 10B)}\right]^{1/2}$$

for the minimum size of the averaging mask. Note that if the background gray-level is 0, we the minimum mask size is $n < \sqrt{10}q$. If this was a fact specified by the instructor,

or student made this assumption from the beginning, then this answer follows almost by inspection.

Problem 3.24

The student should realize that both the Laplacian and the averaging process are linear operations, so it makes no difference which one is applied first.

Problem 3.25

The Laplacian operator is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

for the unrotated coordinates and as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2}.$$

for rotated coordinates. It is given that

$$x = x' \cos\theta - y' \sin\theta \quad \text{and} \quad y = x' \sin\theta + y' \cos\theta$$

where θ is the angle of rotation. We want to show that the right sides of the first two equations are equal. We start with

$$\begin{aligned} \frac{\partial f}{\partial x'} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x'} \\ &= \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta. \end{aligned}$$

Taking the partial derivative of this expression again with respect to x' yields

$$\frac{\partial^2 f}{\partial x'^2} = \frac{\partial^2 f}{\partial x^2}\cos^2\theta + \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)\sin\theta\cos\theta + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)\cos\theta\sin\theta + \frac{\partial^2 f}{\partial y^2}\sin^2\theta.$$

Next, we compute

$$\begin{aligned} \frac{\partial f}{\partial y'} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial y'} \\ &= -\frac{\partial f}{\partial x}\sin\theta + \frac{\partial f}{\partial y}\cos\theta. \end{aligned}$$

Taking the derivative of this expression again with respect to y' gives

$$\frac{\partial^2 f}{\partial y'^2} = \frac{\partial^2 f}{\partial x^2}\sin^2\theta - \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial y}\right)\cos\theta\sin\theta - \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial x}\right)\sin\theta\cos\theta + \frac{\partial^2 f}{\partial y^2}\cos^2\theta.$$

Adding the two expressions for the second derivatives yields

$$\frac{\partial^2 f}{\partial x'^2} + \frac{\partial^2 f}{\partial y'^2} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

which proves that the Laplacian operator is independent of rotation.

Problem 3.26

Unsharp masking is high-boost filtering [Eq. (3.7-11)] with $A = 1$. Figure P3.26 shows the two possible solutions based on that equation. The left and right masks correspond to the first and second line in the equation, respectively.

1	1	1
1	−7	1
1	1	1

(a)

−1	−1	−1
−1	9	−1
−1	−1	−1

(b)

Problem 3.26.

Problem 3.27

Consider the following equation:

$$\begin{aligned} f(x,y) - \nabla^2 f(x,y) &= f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) \\ &\quad + f(x,y-1) - 4f(x,y)] \\ &= 6f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) \\ &\quad + f(x,y-1) + f(x,y)] \\ &= 5\{1.2f(x,y) - \\ &\quad \frac{1}{5}[f(x+1,y) + f(x-1,y) + f(x,y+1) \\ &\quad + f(x,y-1) + f(x,y)]\} \\ &= 5\left[1.2f(x,y) - \overline{f}(x,y)\right] \end{aligned}$$

where $\overline{f}(x,y)$ denotes the average of $f(x,y)$ in a predefined neighborhood that is centered at (x,y) and includes the center pixel and its four immediate neighbors. Treating the constants in the last line of the above equation as proportionality factors, we may write

$$f(x,y) - \nabla^2 f(x,y) \sim f(x,y) - \overline{f}(x,y).$$

The right side of this equation is recognized as the definition of unsharp masking given in Eq. (3.7-7). Thus, it has been demonstrated that subtracting the Laplacian from an

image is proportional to unsharp masking.

Problem 3.28

(a) From Problem 3.25,

$$\frac{\partial f}{\partial x'} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta$$

and

$$\frac{\partial f}{\partial y'} = -\frac{\partial f}{\partial x}\sin\theta + \frac{\partial f}{\partial y}\cos\theta$$

from which it follows that

$$\left(\frac{\partial f}{\partial x'}\right)^2 + \left(\frac{\partial f}{\partial y'}\right)^2 = \left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2$$

or

$$\left[\left(\frac{\partial f}{\partial x'}\right)^2 + \left(\frac{\partial f}{\partial y'}\right)^2\right]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}.$$

Thus, we see that the magnitude of the gradient is an isotropic operator.

(b) From Eq. (3.7-12), (3.7-14) and the preceding results,

$$|G_x| = \left|\frac{\partial f}{\partial x}\right| \qquad |G_y| = \left|\frac{\partial f}{\partial y}\right|,$$

$$|G_{x'}| = \left|\frac{\partial f}{\partial x'}\right| = \left|\frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta\right|,$$

and

$$|G_{y'}| = \left|\frac{\partial f}{\partial y'}\right| = \left|-\frac{\partial f}{\partial x}\sin\theta + \frac{\partial f}{\partial y}\cos\theta\right|.$$

Clearly, $|G_{x'}| + |G_{y'}| \neq |G_x| + |G_y|$.

Problem 3.29

It is given that the range of illumination stays in the linear portion of the camera response range, but no values for the range are given. The fact that images stay in the linear range simply says that images will not be saturated at the high end or be driven in the low end to such an extent that the camera will not be able to respond, thus losing image information irretrievably. The only way to establish a benchmark value for illumination

is when the variable (daylight) illumination is not present. Let $f_0(x, y)$ denote an image taken under artificial illumination only, with no moving objects (e.g., people or vehicles) in the scene. This becomes the standard by which all other images will be normalized. There are numerous ways to solve this problem, but the student must show awareness that areas in the image likely to change due to moving objects should be excluded from the illumination-correction approach.

One simple way is to select various representative subareas of $f_0(x, y)$ not likely to be obscured by moving objects and compute their average intensities. We then select the minimum and maximum of all the individual average values, denoted by, $\overline{f}_{\min}$ and $\overline{f}_{\max}$. The objective then is to process any input image, $f(x, y)$, so that its minimum and maximum will be equal to $\overline{f}_{\min}$ and $\overline{f}_{\max}$, respectively. The easiest way to do this is with a linear transformation function of the form

$$f_{\text{out}}(x, y) = af(x, y) + b.$$

where f_{out} is the output image. It is easily verified that the output image will have the required minimum and maximum values if we choose

$$a = \frac{\overline{f}_{\max} - \overline{f}_{\min}}{f_{\max} - f_{\min}}$$

and

$$b = \frac{\overline{f}_{\min} f_{\max} - \overline{f}_{\max} f_{\min}}{f_{\max} - f_{\min}}$$

where $f_{\max}$ and $f_{\min}$ are the maximum and minimum values of the input image.

Note that the key assumption behind this method is that all images stay within the linear operating range of the camera, thus saturation and other nonlinearities are not an issue. Another implicit assumption is that moving objects comprise a relatively small area in the field of view of the camera, otherwise these objects would overpower the scene and the values obtained from $f_0(x, y)$ would not make a lot of sense. If the student selects another automated approach (e.g., histogram equalization), he/she must discuss the same or similar types of assumptions.

4 Problem Solutions

Problem 4.1

By direct substitution of $f(x)$ [Eq. (4.2-6)] into $F(u)$ [Eq. (4.2-5)]:

$$\begin{aligned} F(u) &= \frac{1}{M}\sum_{x=0}^{M-1}\left[\sum_{r=0}^{M-1}F(r)e^{j2\pi rx/M}\right]e^{-j2\pi ux/M} \\ &= \frac{1}{M}\sum_{r=0}^{M-1}F(r)\sum_{x=0}^{M-1}e^{j2\pi rx/M}e^{-j2\pi ux/M} \\ &= \frac{1}{M}F(u)\,[M] \\ &= F(u) \end{aligned}$$

where the third step follows from the orthogonality condition given in the problem statement. Substitution of $F(u)$ into $f(x)$ is handled in a similar manner.

Problem 4.2

This is a simple problem to familiarize the student with just the manipulation of the 2-D Fourier transform and its inverse. The Fourier transform is linear iff:

$$\Im\left[a_1 f_1(x,y) + a_2 f_2(x,y)\right] = a_1\Im\left[f_1(x,y)\right] + a_2\Im\left[f_2(x,y)\right]$$

where a1 and a2 are arbitrary constants. From the definition of the 2-D transform,

$$\begin{aligned} \Im\left[a_1 f_1(x,y) + a_2 f_2(x,y)\right] &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[a_1 f_1(x,y) + a_2 f_2(x,y)\right] \\ &\quad e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}a_1 f_1(x,y)e^{-j2\pi(ux/M\,+\,vy/N)} \\ &\quad + \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}a_2 f_2(x,y)e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= a_1\Im\left[f_1(x,y)\right] + a_2\Im\left[f_2(x,y)\right] \end{aligned}$$

which proves linearity. The inverse is done in the same way.

Problem 4.3

The inverse DFT of a constant A in the frequency domain is an impulse of strength A in the spatial domain. Convolving the impulse with the image copies (multiplies) the value of the impulse at each pixel location in the image.

Problem 4.4

An important aspect of this problem is to recognize that the quantity $(u^2 + v^2)$ can be replaced by the distance squared, $D^2(u, v)$. This reduces the problem to one variable, which is notationally easier to manage. Rather than carry an award capital letter throughout the development, we define $w^2 \triangleq D^2(u, v) = (u^2 + v^2)$. Then we proceed as follows:

$$H(w) = e^{-w^2/2\sigma^2}.$$

The inverse Fourier transform is

$$\begin{aligned} h(z) &= \int_{-\infty}^{\infty} H(w) e^{j2\pi wz} dw \\ &= \int_{-\infty}^{\infty} e^{-w^2/2\sigma^2} e^{j2\pi wz} dw \\ &= \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}\left[w^2 - j4\pi\sigma^2 wz\right]} dw. \end{aligned}$$

We now make use of the identity

$$e^{-\frac{(2\pi)^2 z^2 \sigma^2}{2}} e^{\frac{(2\pi)^2 z^2 \sigma^2}{2}} = 1.$$

Inserting this identity in the preceding integral yields

$$\begin{aligned} h(z) &= e^{-\frac{(2\pi)^2 z^2 \sigma^2}{2}} \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}\left[w^2 - j4\pi\sigma^2 wz - (2\pi)^2\sigma^4 z^2\right]} dw \\ &= e^{-\frac{(2\pi)^2 z^2 \sigma^2}{2}} \int_{-\infty}^{\infty} e^{-\frac{1}{2\sigma^2}\left[w - j2\pi\sigma^2 z\right]^2} dw. \end{aligned}$$

Next we make the change of variable $r = w - j2\pi\sigma^2 z$. Then, $dr = dw$ and the above integral becomes

$$h(z) == e^{-\frac{(2\pi)^2 z^2 \sigma^2}{2}} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr.$$

Finally, we multiply and divide the right side of this equation by $\sqrt{2\pi}\sigma$:

$$h(z) = \sqrt{2\pi}\sigma e^{-\frac{(2\pi)^2 z^2 \sigma^2}{2}} \left[\frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr\right].$$

The expression inside the brackets is recognized as a Gaussian probability density function, whose integral from $-\infty$ to ∞ is 1. Then,

$$h(z) = \sqrt{2\pi}\sigma e^{-\frac{(2\pi)^2 z^2 \sigma^2}{2}}.$$

Going back to two spatial variables gives the final result:$h(x,y) = \sqrt{2\pi}\sigma\, e^{-2\pi^2\sigma^2(x^2+y^2)}$.

Problem 4.5

The spatial filter is obtained by taking the inverse Fourier transform of the frequency-domain filter:

$$\begin{aligned} h_{\text{hp}}(x,y) &= \Im^{-1}\left[1 - H_{\text{lp}}(u,v)\right] \\ &= \Im^{-1}\left[1\right] - \Im^{-1}\left[H_{\text{lp}}(u,v)\right] \\ &= \delta(0) - \sqrt{2\pi}\sigma\, e^{-2\pi^2\sigma^2(x^2+y^2)} \end{aligned}$$

Problem 4.6

(a) We note first that $(-1)^{x+y} = e^{j\pi(x+y)}$. Then,

$$\begin{aligned} \Im\left[f(x,y)e^{j\pi(x+y)}\right] &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)e^{j\pi(x+y)}\right]e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)e^{-j2\pi(-\frac{xM}{2M}-\frac{yN}{2N})}\right] \\ &\qquad e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi\left(x[u-\frac{M}{2}]/M+y[v-\frac{N}{2}]/N\right)} \\ &= F(u - M/2, v - N/2). \end{aligned}$$

(b) Following the same format as in (a),

$$\begin{aligned} \Im\left[f(x,y)e^{j2\pi(u_0x/M\,+\,v_0y/M)}\right] &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[f(x,y)e^{j2\pi(u_0x/M\,+\,v_0y/M)}\right] \\ &\qquad e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y) \\ &\qquad e^{-j2\pi(x[u-u_0]/M\,+\,y[v-v_0]/N)} \\ &= F(u-u_0, v-v_0) \end{aligned}$$

Similarly,

$$\Im^{-1}\left[F(u,v)e^{-j2\pi(ux_0/M\,+\,vy_0/M)}\right] = f(x-x_0, y-y_0).$$

Problem 4.7

The equally-spaced, vertical bars on the left, lower third of the image.

Problem 4.8

With reference to Eq. (4.4-1), all the highpass filters in discussed in Section 4.4 can be expressed a 1 minus the transfer function of lowpass filter (which we know do not have an impulse at the origin). The inverse Fourier transform of 1 gives an impulse at the origin in the highpass spatial filters.

Problem 4.9

The complex conjugate simply changes j to $-j$ in the inverse transform, so the image on the right is given by

$$\begin{aligned}\Im^{-1}\left[F^*(u,v)\right] &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u.v)e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} F(u.v)e^{j2\pi(u(-x)/M\,+\,v(-y)/N)} \\ &= f(-x,-y)\end{aligned}$$

which simply mirrors $f(x,y)$ about the origin, thus producing the image on the right.

Problem 4.10

If $H(u,v)$ is real and symmetric, then

$$H(u,v) = H^*(u,v) = H^*(-u,-v) = H(-u,-v).$$

The filter in the spatial domain is

$$h(x,y) = \Im^{-1}\left[H(u,v)\right] = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H(u.v)e^{j2\pi(ux/M\,+\,vy/N)}.$$

Then,

$$\begin{aligned} h^*(x,y) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H^*(u.v)e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H^*(-u,-v)e^{j2\pi(ux/M\,+\,vy/N)} \\ &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H(u,v)e^{j2\pi(ux/M\,+\,vy/N)} \\ &= h(x,y) \qquad (\text{real}). \end{aligned}$$

Similarly,

$$\begin{aligned} h(-x,-y) &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H(u,v)e^{-j2\pi(ux/M\,+\,vy/N)} \\ &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H(-u,-v)e^{j2\pi(ux/M\,+\,vy/N)} \\ &= \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} H(u,v)e^{j2\pi(ux/M\,+\,vy/N)} \\ &= h(x,y) \qquad (\text{symmetric}). \end{aligned}$$

Problem 4.11

Starting from Eq. (4.2-30), we easily find the expression for the definition of continuous convolution in one dimension:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)d\alpha.$$

The Fourier transform of this expression is

$$\begin{aligned} \Im\left[f(x) * g(x)\right] &= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)d\alpha\right]e^{-j2\pi ux}dx \\ &= \int_{-\infty}^{\infty} f(\alpha)\left[\int_{-\infty}^{\infty} g(x-\alpha)e^{-j2\pi ux}dx\right]d\alpha. \end{aligned}$$

The term inside the inner brackets is the Fourier transform of $g(x-\alpha)$. But,

$$\Im\left[g(x-\alpha)\right] = G(u)e^{-j2\pi u\alpha}$$

so

$$\begin{aligned} \Im\left[f(x) * g(x)\right] &= \int_{-\infty}^{\infty} f(\alpha)\left[G(u)e^{-j2\pi u\alpha}\right]d\alpha \\ &= G(u)\int_{-\infty}^{\infty} f(\alpha)e^{-j2\pi u\alpha}d\alpha \\ &= G(u)F(u). \end{aligned}$$

This proves that multiplication in the frequency domain is equal to convolution in the spatial domain. The proof that multiplication in the spatial domain is equal to convolution in the spatial domain is done in similar way.

Problem 4.12

(a) The ring in fact has a dark center area as a result of the highpass operation only (the following image shows the result of highpass filtering only). However, the dark center area is averaged out by the lowpass filter. The reason the final result looks so bright is that the discontinuity (edge) on boundaries of the ring are much higher than anywhere else in the image, thus giving an averaged area whose gray level dominates.

(b) Filtering with the Fourier transform is a linear process. The order does not matter.

Figure P4.12

Problem 4.13

(a) One application of the filter gives:

$$\begin{aligned} G(u,v) &= H(u,v)F(u,v) \\ &= e^{-D^2(u,v)/2D_0^2}F(u,v). \end{aligned}$$

Similarly, K applications of the filter would give

$$G_K(u,v) = e^{-KD^2(u,v)/2D_0^2}F(u,v).$$

The inverse DFT of $G_K(u,v)$ would give the image resulting from K passes of the Gaussian filter. If K is "large enough," the Gaussian LPF will become a notch pass filter, passing only $F(0,0)$. We know that this term is equal to the average value of the image. So, there is a value of K after which the result of repeated lowpass filtering will simply produce a constant image. The value of all pixels on this image will be equal to the average value of the original image. Note that the answer applies even as K approaches infinity. In this case the filter will approach an impulse at the origin, and this would still give us $F(0,0)$ as the result of filtering.

(b) To guarantee the result in (a), K has to be chosen large enough so that the filter becomes a notch pass filter (at the origin) for all values of $D(u,v)$. Keeping in mind that increments of frequencies are in unit values, this means

$$H_K(u,v) = e^{-KD^2(u,v)/2D_0^2} = \begin{cases} 1 & \text{if } (u,v) = (0,0) \\ 0 & \text{Otherwise.} \end{cases}$$

Because u and v are integers, the conditions on the second line in this equation are satisfied for all $u > 1$ and/or $v > 1$. When $u = v = 0$, $D(u,v) = 0$, and $H_K(u,v) = 1$, as desired.

We want all values of the filter to be zero for all values of the distance from the origin that are greater than 0 (i.e., for values of u and/or v greater than 0). However, the filter is a Gaussian function, so its value is always greater than 0 for all finite values of $D(u,v)$. But, we are dealing with digital numbers, which will be designated as zero whenever the value of the filter is less than $\frac{1}{2}$ the smallest positive number representable in the computer being used. Assume this number to be k_{min} (don't confuse the meaning of this k with K, which is the number of applications of the filter). So, values of K for which for which the filter function is greater than $0.5 \times k_{\text{min}}$ will suffice. That is, we want the minimum value of K for which

$$e^{-KD^2(u,v)/2D_0^2} < 0.5k_{\text{min}}$$

or

$$\begin{aligned} K &> -\frac{\ln(0.5k_{\text{min}})}{D^2(u,v)/2D_0^2} \\ &> -\frac{2D_0^2\ \ln(0.5k_{\text{min}})}{D^2(u,v)}. \end{aligned}$$

As noted above, we want this equation for hold for all values of $D^2(u,v) > 0$. Since the exponential decreases as a function of increasing distance from the origin, we choose

the smallest possible value of $D^2(u,v)$, which is 1. Tis gives the result

$$K > -2D_0^2 \ln(0.5k_{\min})$$

which gives a positive number because $k_{\min} << 1$. This result guarantees that the lowpass filter will act as a notch pass filter, leaving only the value of the transform at the origin. The image will not change past this value of K.

Problem 4.14

(a) The spatial average is

$$g(x,y) = \frac{1}{4}\left[f(x,y+1) + f(x+1,y) + f(x-1,y) + f(x,y-1)\right].$$

From Eq. (4.6-2),

$$\begin{aligned} G(u,v) &= \frac{1}{4}\left[e^{j2\pi v/N} + e^{j2\pi u/M} + e^{-j2\pi u/M} + e^{-j2\pi v/N}\right] F(u,v) \\ &= H(u,v)F(u,v), \end{aligned}$$

where

$$H(u,v) = \frac{1}{2}\left[\cos(2\pi u/M) + \cos(2\pi v/N)\right]$$

is the filter transfer function in the frequency domain.

(b) To see that this is a lowpass filter, it helps to express the preceding equation in the form of our familiar centered functions:

$$H(u,v) = \frac{1}{2}\left[\cos(2\pi[u - M/2)/M) + \cos(2\pi[v - N/2]/N)\right].$$

Consider one variable for convenience. As u ranges from 0 to M, the value of $\cos(2\pi[u - M/2)/M)$ starts at -1, peaks at 1 when $u = M/2$ (the center of the filter) and then decreases to -1 again when $u = M$. Thus, we see that the amplitude of the filter decreases as a function of distance from the origin of the centered filter, which is the characteristic of a lowpass filter. A similar argument is easily carried out when considering both variables simultaneously.

Problem 4.15

The problem statement gives the form of the difference in the x-direction. A similar expression gives the difference in the y-direction. The filtered function in the spatial domain then is:

$$g(x,y) = f(x,y) - f(x+1,y) + f(x,y) - f(x,y+1).$$

From Eq. (4.6-2),

$$\begin{aligned} G(u,v) &= F(u,v) - F(u,v)e^{j2\pi u/M} + F(u,v) - F(u,v)e^{j2\pi v/N} \\ &= [1 - e^{j2\pi u/M}]F(u,v) + [1 - e^{j2\pi v/N}]F(u,v) \\ &= H(u,v)F(u,v), \end{aligned}$$

where $H(u,v)$ is the filter function:

$$H(u,v) = -2j\left[\sin(\pi u/M)e^{j\pi u/M} + \sin(\pi v/N)e^{j\pi v/N}\right].$$

(b) To see that this is a highpass filter, it helps to express the filter function in the form of our familiar centered functions:

$$H(u,v) = -2j\left[\sin(\pi[u - M/2]/M)e^{j\pi u/M} + \sin(\pi[v - N/2]/N)e^{j\pi v/N}\right].$$

Consider one variable for convenience. As u ranges from 0 to M, $H(u,v)$ starts at its maximum (complex) value of $2j$ for $u = 0$ and decreases from there. When $u = M/2$ (the center of the shifted function), A similar argument is easily carried out when considering both variables simultaneously.. The value of $H(u,v)$ starts increasing again and achieves the maximum value of $2j$ again when $u = M$. Thus, this filter has a value of 0 a the origin and increases with increasing distance from the origin. This is the characteristic of a highpass filter. A similar argument is easily carried out when considering both variables simultaneously.

Problem 4.16

(a) The key for the student to be able to solve the problem is to treat the number of applications (denoted by K) of the highpass filter as 1 minus K applications of the corresponding lowpass filter, so that

$$\begin{aligned} H_K(u,v) &= H_K(u,v)F(u,v) \\ &= \left[1 - e^{-KD^2(u,v)/2D_0^2}\right]H(u,v) \end{aligned}$$

where the Gaussian lowpass filter is from Problem 4.13. Students who start directly with the expression of the Gaussian highpass filter $\left[1 - e^{-KD^2(u,v)/2D_0^2}\right]$ and attempt to raise it to the Kth power will run into a dead end.

The solution to this problem parallels the solution to Problem 4.13. Here, however, the filter will approach a notch filter that will take out $F(0,0)$ and thus will produce an image with zero average values (this implies negative pixels). So, there is a value of K after which the result of repeated highpass filtering will simply produce a constant image.

(b) The problem is to determine the value of K for which

$$H_K(u,v) = 1 - e^{-KD^2(u,v)/2D_0^2} = \begin{cases} 0 & \text{if } (u,v) = (0,0) \\ 1 & \text{Otherwise.} \end{cases}$$

Because u and v are integers, the conditions on the second line in this equation are satisfied for all $u > 1$ and/or $v > 1$. When $u = v = 0$, $D(u,v) = 0$, and $H_K(u,v) = 0$, as desired.

We want all values of the filter to be 1 for all values of the distance from the origin that are greater than 0 (i.e., for values of u and/or v greater than 0). For $H_K(u,v)$ to become 1, the exponential term has to become 0 for values of u and/or v greater than 0. This is the same requirement as in Problem 4.13, so the solution of that problem applies here as well.

Problem 4.17

(a) Express filtering as convolution to reduce all processes to the spatial domain. Then, the filtered image is given by

$$g(x,y) = h(x,y) * f(x,y)$$

where h is the spatial filter (inverse Fourier transform of the frequency-domain filter) and f is the input image. Histogram processing this result yields

$$\begin{aligned} g\prime(x,y) &= T\left[g(x,y)\right] \\ &= T\left[h(x,y) * f(x,y)\right], \end{aligned}$$

where T denotes the histogram equalization transformation. If we histogram-equalize first, then

$$g(x,y) = T\left[f(x,y)\right]$$

and

$$g\prime(x,y) = h(x,y) * T\left[f(x,y)\right].$$

In general, T is a nonlinear function determined by the nature of the pixels in the image from which it is computed. Thus, in general, $T\left[h(x,y) * f(x,y)\right] \neq h(x,y) * T\left[f(x,y)\right]$ and the order does matter.

(b) As indicated in Section 4.4, highpass filtering severely diminishes the contrast of an image. Although high-frequency emphasis helps some, the improvement is usually not dramatic (see Fig. 4.30). Thus, if an image is histogram equalized first, the gain in contrast improvement will essentially be lost in the filtering process. Therefore, the procedure in general is to filter first and histogram-equalize the image after that.

Problem 4.18

The answer is no. The Fourier transform is a linear process, while the square and square roots involved in computing the gradient are nonlinear operations. The Fourier transform could be used to compute the derivatives (as differences—see Prob.4.15), but the squares, square root, or absolute values must be computed directly in the spatial domain.

Problem 4.19

The equation corresponding to the mask in Fig. 4.27(f) is Eq. (3.7-4):

$$g(x,y) = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)] - 4f(x,y).$$

As in Problem 4.15,

$$G(u,v) = H(u,v)F(u,v)$$

where

$$\begin{aligned} H(u,v) &= \left[e^{j2\pi u/M} + e^{-j2\pi u/M} + e^{j2\pi v/N} + e^{-j2\pi v/N} - 4\right] \\ &= 2\left[\cos(2\pi u/M) + \cos(2\pi v/N) - 2\right]. \end{aligned}$$

Shifting the filter to the center of the frequency rectangle gives

$$H(u,v) = 2\left[\cos(2\pi\left[u - M/2\right]/M) + \cos(2\pi\left[v - N/2\right]/N) - 2\right].$$

When $(u,v) = (M/2, N/2)$ (the center of the shifted filter). For values away from the center values of $H(u,v)$ decrease, but this is as expected [see Fig. 4.27(a)] for this particular formulation of the Laplacian.

Problem 4.20

From Eq. (4.4-3), the transfer function of a Butterworth highpass filter is

$$H(u,v) = \frac{1}{1 + \left[\frac{D_0}{D(u,v)}\right]^{2n}}.$$

We want the filter to have a value of γ_L when $D(u,v) = 0$, and approach γ_H for high values of $D(u,v)$. The preceding equation is easily modified to accomplish this:

$$H(u,v) = \gamma_L + \frac{(\gamma_H - \gamma_L)}{1 + \left[\frac{D_0}{D(u,v)}\right]^{2n}}.$$

The value of n controls the sharpness of the transition between γ_L and γ_H.

Problem 4.21

Recall that the reason for padding is to establish a "buffer" between the periods that are implicit in the DFT. Imagine the image on the left being duplicated infinitely many times to cover the xy-plane. The result would be a checkerboard, with each square being in the checkerboard being the image (and the black extensions). Now imagine doing the same thing to the image on the right. The results would be indistinguishable. Thus, either form of padding accomplishes the same separation between images, as desired.

Problem 4.22

(a) Padding an image with zeros increases its size, but not its gray-level content. Thus, the average gray-level of the padded image is lower than that of the original image. This implies that $F(0,0)$ in the spectrum of the padded image is less than $F(0,0)$ in the original image (recall that $F(0,0)$ is the average value of the corresponding image). Thus, we can visualize $F(0,0)$ being lower in the spectrum on the right, with all values away from the origin being lower too, and covering a narrower range of values. That's the reason the overall contrast is lower in the picture on the right.

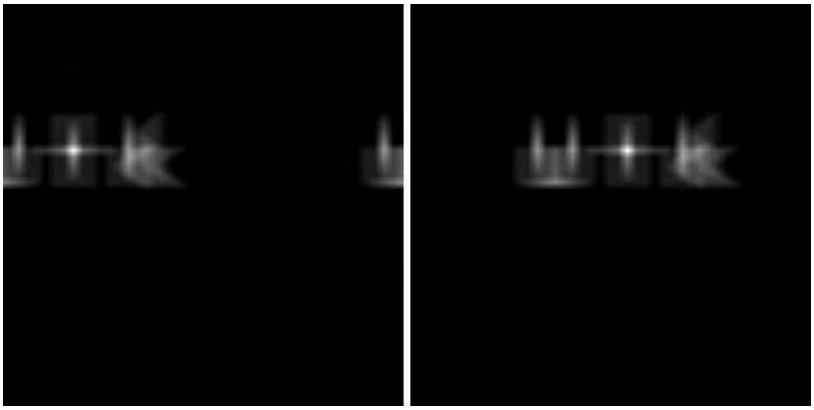
(b) Padding an image with 0's introduces significant discontinuities at the borders of the original images. This process introduces strong horizontal and vertical edges, where the image ends abruptly and then continues with 0 values. These sharp transitions correspond to the strength of the spectrum along the horizontal and vertical axes of the spectrum.

Problem 4.23

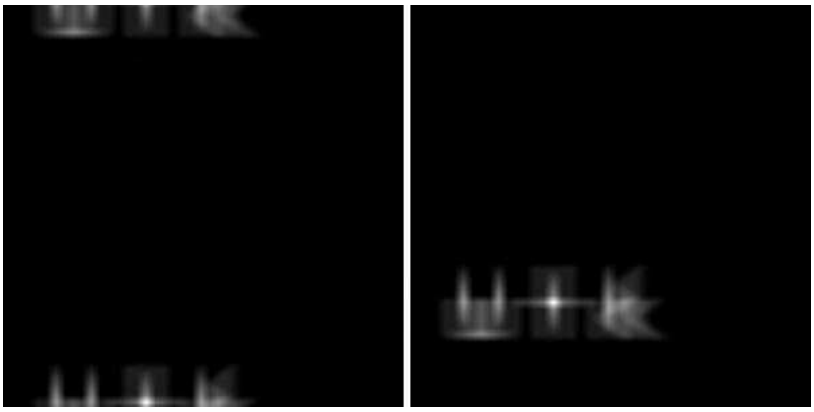
As in problem 4.9, taking the complex conjugate of an image mirrors it in the spatial domain. Thus, we would expect the result to be a mirror image (about both axes) of Fig. 4.41(e).

Problem 4.24

(a) and (b) See Figs. P4.24(a) and (b). (c) and (d) See Figs. P4.24(c) and (d).



Figures P4.24(a) and (b)



Figures P4.24(c) and (d)

Problem 4.25

Because $M = 2^n$, we can write Eqs. (4.6-47) and (4.6-48) respectively as

$$m(n) = \frac{1}{2}Mn$$

and

$$a(n) = Mn.$$

Proof by induction begins by showing that both equations hold for $n = 1$:

$$m(1) = \frac{1}{2}(2)(1) = 1 \qquad \text{and} \qquad a(1) = (2)(1) = 2.$$

We know these results to be correct from the discussion in Section 4.6.6. Next, we assume that the equations hold for n. Then, we are required to prove that they also are true for $n + 1$. From Eq. (4.6-45),

$$m(n + 1) = 2m(n) + 2^n.$$

Substituting $m(n)$ from above,

$$\begin{aligned} m(n+1) &= 2\left(\frac{1}{2}Mn\right) + 2^n \\ &= 2\left(\frac{1}{2}2^n n\right) + 2^n \\ &= 2^n(n+1) \\ &= \frac{1}{2}\left(2^{n+1}\right)(n+1). \end{aligned}$$

Therefore, Eq. (4.6-47) is valid for all n.

From Eq. (4.6-46),

$$a(n+1) = 2a(n) + 2^{n+1}.$$

Substituting the above expression for $a(n)$ yields

$$\begin{aligned} a(n+1) &= 2Mn + 2^{n+1} \\ &= 2(2^n n) + 2^{n+1} \\ &= 2^{n+1}(n+1) \end{aligned}$$

which completes the proof.

Problem 4.26

Consider a single star modeled as an impulse $\delta(x - x_0, y - y_0)$. Then,

$$f(x,y) = K\delta(x - x_0, y - y_0)$$

from which

$$\begin{aligned} z(x,y) &= \ln f(x,y) = \ln K + \ln \delta(x - x_0, y - y_0) \\ &= K' + \delta'(x - x_0, y - y_0). \end{aligned}$$

Taking the Fourier transform of both sides yields

$$\begin{aligned} \Im\left[z(x,y)\right] &= \Im\left[K'\right] + \Im\left[\delta'(x - x_0, y - y_0)\right] \\ &= \delta(0,0) + e^{-2\pi(ux_0 + vy_0)}. \end{aligned}$$

From this result, it is evident that the contribution of illumination is an impulse at the origin of the frequency plane. A notch filter that attenuates only this component will take care of the problem. Extension of this development to multiple impulses (stars) is straightforward. The filter will be the same.

Problem 4.27

The problem can be solved by carrying out the following steps:

1. Perform a median filtering operation.
2. Follow (1) by high-frequency emphasis.
3. Histogram-equalize this result.
4. Compute the average gray level, K_0. Add the quantity $(K - K_0)$ to all pixels.
5. Perform the transformations shown in Fig. P4.27, where r is the input gray level, and R, G, and B are fed into an RGB color monitor.

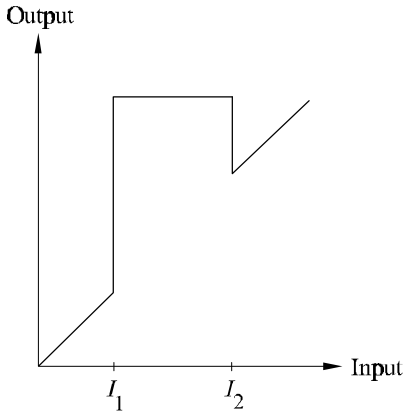


Figure P4.27

5 Problem Solutions

Problem 5.1

The solutions to (a), (b), and (c) are shown in Fig. P5.1, from left to right:

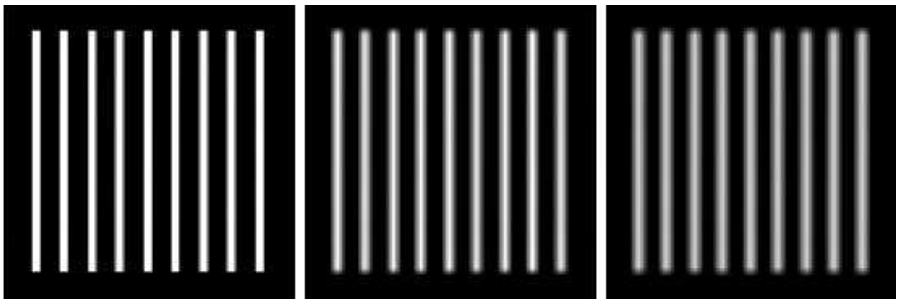


Figure P5.1

Problem 5.2

The solutions to (a), (b), and (c) are shown in Fig. P5.2, from left to right:

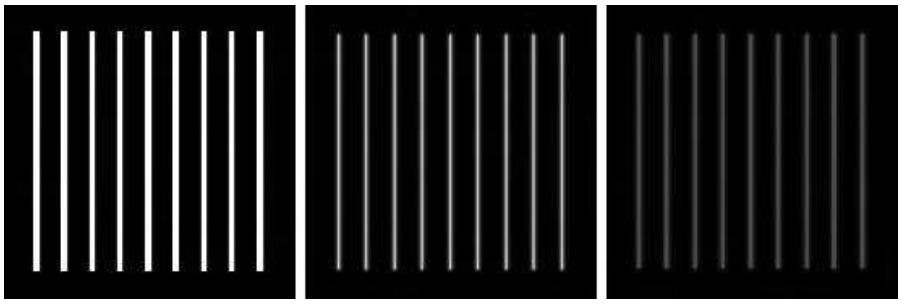


Figure P5.2

Problem 5.3

The solutions to (a), (b), and (c) are shown in Fig. P5.3, from left to right:

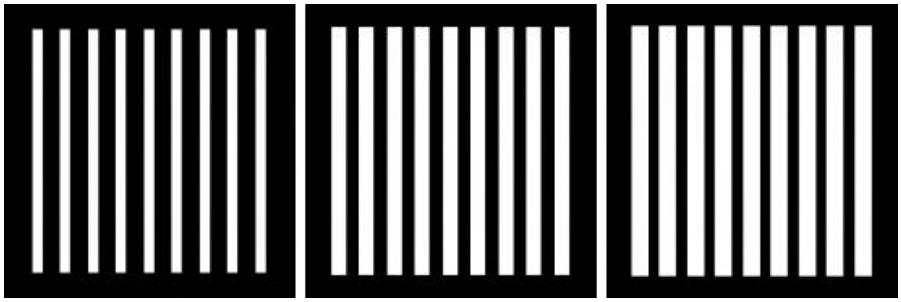


Figure P5.3

Problem 5.4

The solutions to (a), (b), and (c) are shown in Fig. P5.4, from left to right:

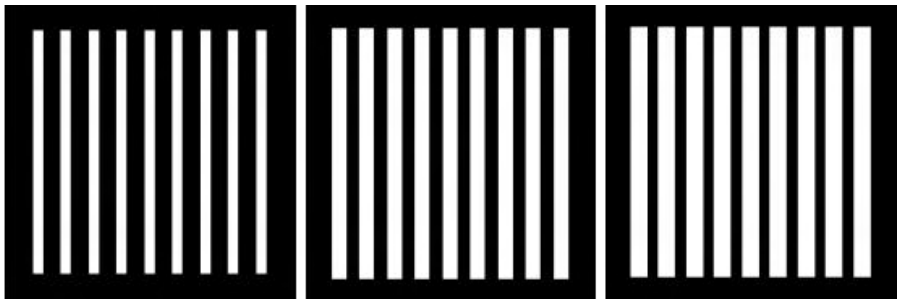


Figure P5.4

Problem 5.5

The solutions to (a), (b), and (c) are shown in Fig. P5.5, from left to right:

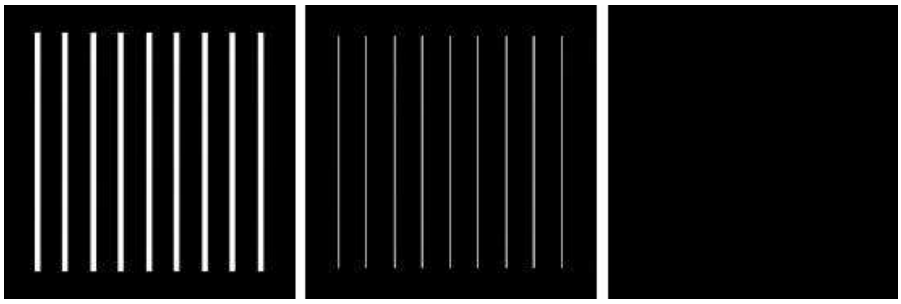


Figure P5.5

Problem 5.6

The solutions to (a), (b), and (c) are shown in Fig. P5.6, from left to right:

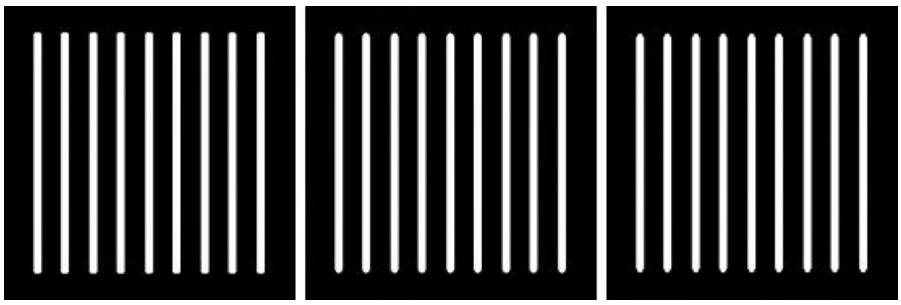


Figure P5.6

Problem 5.7

The solutions to (a), (b), and (c) are shown in Fig. P5.7, from left to right:

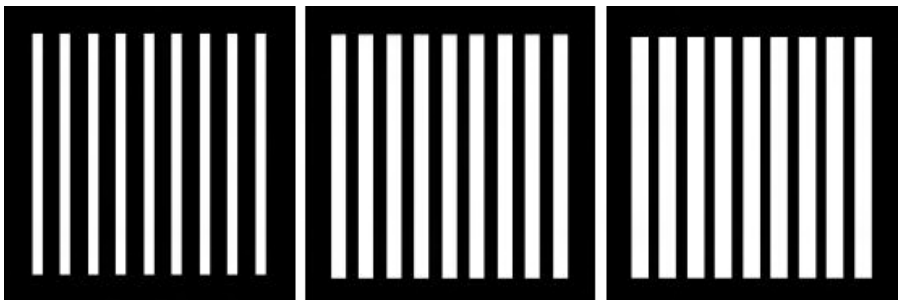


Figure P5.7

Problem 5.8

The solutions to (a), (b), and (c) are shown in Fig. P5.8, from left to right:

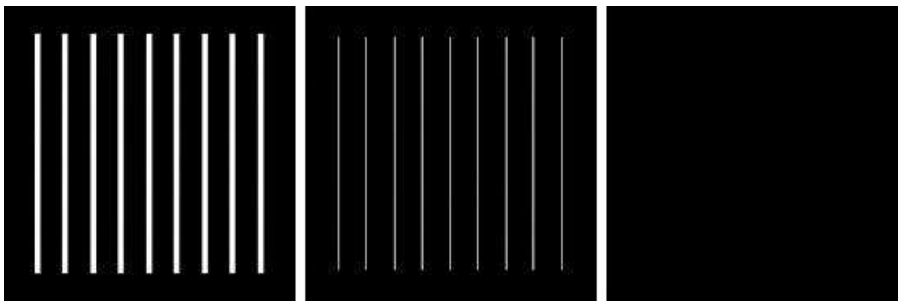


Figure P5.8

Problem 5.9

The solutions to (a), (b), and (c) are shown in Fig. P5.9, from left to right:

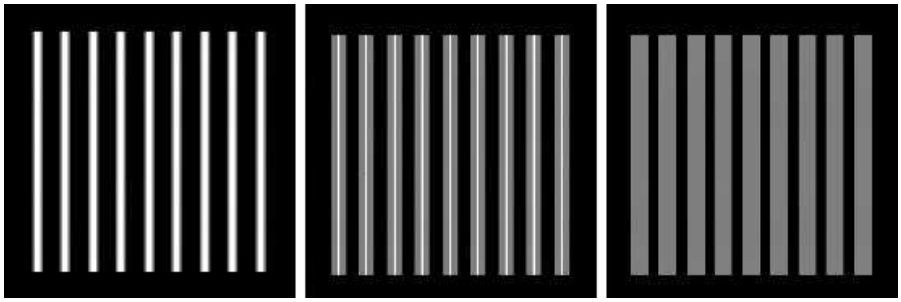


Figure P5.9

Problem 5.10

(a) The key to this problem is that the geometric mean is zero whenever any pixel is zero. Draw a profile of an ideal edge with a few points valued 0 and a few points valued 1. The geometric mean will give only values of 0 and 1, whereas the arithmetic mean will give intermediate values (blur).

(b) Black is 0, so the geometric mean will return values of 0 as long as at least one pixel

in the window is black. Since the center of the mask can be outside the original black area when this happens, the figure will be thickened.

Problem 5.11

The key to understanding the behavior of the contra-harmonic filter is to think of the pixels in the neighborhood surrounding a noise impulse as being constant, with the impulse noise point being in the center of the neighborhood. For the noise spike to be visible, its value must be considerably larger than the value of its neighbors. Also keep in mind that the power in the numerator is 1 plus the power in the denominator.

(a) By definition, pepper noise is a low value (really 0). It is most visible when surrounded by light values. Then center pixel (the pepper noise), will have little influence in the sums. If the area spanned by the filter is approximately constant, the ratio will approach the value of the pixels in the neighborhood—thus reducing the effect of the low-value pixel. For example, here are some values of the filter for a dark point of value 1 in a 3×3 region with pixels of value 100: For $Q = 0.5$, filter $= 98.78$; for $Q = 1$, filter $= 99.88$, for $Q = 2$, filter $= 99.99$; and for $Q = 5$, filter $= 100.00$.

(b) The reverse happens when the center point is large and its neighbors are small. The center pixel will now be the largest. However, the exponent is now negative, so the small numbers will dominate the result. The numerator can then be thought of a constant raised to the power $Q + 1$ and the denominator as a the same constant raised to the power Q. That constant is the value of the pixels in the neighborhood. So the ratio is just that value.

(c) When the wrong polarity is used the large numbers in the case of the salt noise will be raised to a positive power, thus the noise will overpower the result. For salt noise the image will become very light. The opposite is true for pepper noise—the image will become dark.

(d) When $Q = -1$, the value of the numerator becomes equal to the number of pixels in the neighborhood ($m \times n$). The value of the denominator become sum values, each of which is 1 over the value of a pixel in the neighborhood. This is the same as the average of $1/A$, where A is the image average.

(e) In a constant area, the filter returns the value of the pixels in the area, independently of the value of Q.

Problem 5.12

A bandpass filter is obtained by subtracting the corresponding bandreject filter from 1:

$$H_{\text{bp}}(u,v) = 1 - H_{\text{br}}(u,v).$$

Then:

(a) Ideal bandpass filter:

$$H_{\text{Ibp}}(u,v) = \begin{cases} 0 & \text{if } D(u,v) < D_0 - \frac{W}{2} \\ 1 & \text{if } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 0 & D(u,v) > D_0 + \frac{W}{2} \end{cases}.$$

(b) Butterworth bandpass filter:

$$\begin{aligned} H_{\text{Bbp}}(u,v) &= 1 - \frac{1}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - D_0^2}\right]^{2n}} \\ &= \frac{\left[\frac{D(u,v)W}{D^2(u,v) - D_0^2}\right]^{2n}}{1 + \left[\frac{D(u,v)W}{D^2(u,v) - D_0^2}\right]^{2n}}. \end{aligned}$$

(c) Gaussian bandpass filter:

$$\begin{aligned} H_{\text{Gbp}}(u,v) &= 1 - \left[1 - e^{-\frac{1}{2}\left[\frac{D^2(u,v) - D_0^2}{D(u,v)W}\right]^2}\right] \\ &= e^{-\frac{1}{2}\left[\frac{D^2(u,v) - D_0^2}{D(u,v)W}\right]^2}. \end{aligned}$$

Problem 5.13

A notch pass filter is obtained by subtracting the corresponding notch reject filter from 1:

$$H_{\text{np}}(u,v) = 1 - H_{\text{nr}}(u,v).$$

Then:

(a) Ideal notch pass filter:

$$H_{\text{Inp}}(u,v) = \begin{cases} 1 & \text{if } D_1(u,v) \leq D_0 \text{ or } D_2(u,v) \leq D_0 \\ 0 & \text{otherwise} \end{cases}.$$

(b) Butterworth notch pass filter:

$$\begin{aligned} H_{\text{Bnp}}(u,v) &= 1 - \frac{1}{1+\left[\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n} \\ &= \frac{\left[\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n}{1+\left[\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n}. \end{aligned}$$

(c) Gaussian notch pass filter:

$$\begin{aligned} H_{\text{Gnp}}(u,v) &= 1 - \left[1 - e^{-\frac{1}{2}\left[\frac{D_1(u,v)D_2(u,v)}{D_0^2}\right]}\right] \\ &= e^{-\frac{1}{2}\left[\frac{D_1(u,v)D_2(u,v)}{D_0^2}\right]}. \end{aligned}$$

Problem 5.14

We proceed as follows:

$$\begin{aligned} F(u,v) &= \iint_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dx\,dy \\ &= \iint_{-\infty}^{\infty} A\sin(u_0x+v_0y)e^{-j2\pi(ux+vy)}dx\,dy. \end{aligned}$$

Using the exponential definition of the sine function:

$$\sin\theta = \frac{1}{2j}\left(e^{j\theta} - e^{-j\theta}\right)$$

gives us

$$\begin{aligned} F(u,v) &= \frac{-jA}{2}\iint_{-\infty}^{\infty}\left[e^{j(u_0x+v_0y)} - e^{-j(u_0x+v_0y)}\right]e^{-j2\pi(ux+vy)}dx\,dy \\ &= \frac{-jA}{2}\left[\iint_{-\infty}^{\infty} e^{j2\pi(u_0x/2\pi+v_0y/2\pi)}e^{-j2\pi(ux+vy)}dx\,dy\right] - \\ &\quad \frac{jA}{2}\left[\iint_{-\infty}^{\infty} e^{-j2\pi(u_0x/2\pi+v_0y/2\pi)}e^{-j2\pi(ux+vy)}dx\,dy\right]. \end{aligned}$$

These are the Fourier transforms of the functions

$$1 \times e^{j2\pi(u_0x/2\pi+v_0y/2\pi)}$$

and

$$1 \times e^{-j2\pi(u_0x/2\pi+v_0y/2\pi)}$$

respectively. The Fourier transform of the 1 gives an impulse at the origin, and the exponentials shift the origin of the impulse, as discussed in Section 4.6.1. Thus,

$$F(u,v) = \frac{-jA}{2}\left[\delta\left(u-\frac{u_0}{2\pi}, v-\frac{v_0}{2\pi}\right) - \delta\left(u+\frac{u_0}{2\pi}, v+\frac{v_0}{2\pi}\right)\right].$$

Problem 5.15

From Eq. (5.4-19)

$$\sigma^2 = \frac{1}{(2a+1)(2b+1)} \sum\sum \{[g(\gamma) - w\eta(\gamma)] - [\overline{g} - w\overline{\eta}]\}^2$$

where "γ" indicates terms affected by the summations. Letting $K = 1/(2a+1)(2b+1)$, taking the partial derivative of σ^2 with respect to w and setting the result equal to zero gives

$$\begin{aligned}
\frac{\partial \sigma^2}{\partial w} &= K \sum\sum 2\left[g(\gamma) - w\eta(\gamma) - \overline{g} + w\overline{\eta}\right]\left[-\eta(\gamma) + \overline{\eta}\right] = 0 \\
&= K \sum\sum -g(\gamma)\eta(\gamma) + g(\gamma)\overline{\eta} + w\eta^2(\gamma) - w\eta(\gamma)\overline{\eta} + \\
&\quad \overline{g}\eta(\gamma) - \overline{g\eta} - w\overline{\eta}\eta(\gamma) + w\overline{\eta}^2 \\
&= 0 \\
&= -\overline{g\eta} + \overline{g\eta} + w\overline{\eta^2} - w\overline{\eta}^2 + \overline{g\eta} - \overline{g\eta} - w\overline{\eta}^2 + w\overline{\eta}^2 = 0 \\
&= -\overline{g\eta} + \overline{g\eta} + w\left(\overline{\eta^2} - \overline{\eta}^2\right) = 0
\end{aligned}$$

where, for example, we used the fact that

$$\frac{1}{(2a+1)(2b+1)} \sum\sum g(\gamma)\eta(\gamma) = \overline{g\eta}.$$

Solving for w gives us

$$w = \frac{\overline{g\eta} - \overline{g\eta}}{\overline{\eta^2} - \overline{\eta}^2}.$$

Finally, inserting the variables x and y,

$$w(x,y) = \frac{\overline{g(x,y)\eta(x,y)} - \overline{g}(x,y)\overline{\eta}(x,y)}{\overline{\eta^2}(x,y) - \overline{\eta}^2(x,y)}$$

which agrees with Eq. (5.4-21).

Problem 5.16

From Eq. (5.5-13),

$$g(x,y) = \iint_{-\infty}^{\infty} f(\alpha,\beta)h(x-\alpha, y-\beta)\, d\alpha\, d\beta.$$

It is given that $f(x,y) = \delta(x-a)$, so $f(\alpha,\beta) = \delta(\alpha - a)$. Then, using the impulse response given in the problem statement,

$$g(x,y) = \iint_{-\infty}^{\infty} \delta(\alpha - a) e^{-\left[(x-\alpha)^2 + (y-\beta)^2\right]}\, d\alpha\, d\beta$$

$$
\begin{aligned}
&= \iint_{-\infty}^{\infty} \delta(\alpha - a) e^{-\left[(x-\alpha)^2\right]} e^{-\left[(y-\beta)^2\right]} \, d\alpha \, d\beta \\
&= \int_{-\infty}^{\infty} \delta(\alpha - a) e^{-\left[(x-\alpha)^2\right]} \, d\alpha \int_{-\infty}^{\infty} e^{-\left[(y-\beta)^2\right]} \, d\beta \\
&= e^{-\left[(x-a)^2\right]} \int_{-\infty}^{\infty} e^{-\left[(y-\beta)^2\right]} \, d\beta
\end{aligned}
$$

where we used the fact that the integral of the impulse is nonzero only when $\alpha = a$. Next, we note that

$$\int_{-\infty}^{\infty} e^{-\left[(y-\beta)^2\right]} \, d\beta = \int_{-\infty}^{\infty} e^{-\left[(\beta-y)^2\right]} \, d\beta$$

which is in the form of a constant times a Gaussian density with variance $\sigma^2 = 1/2$ or standard deviation $\sigma = 1/\sqrt{2}$. In other words,

$$e^{-\left[(\beta-y)^2\right]} = \sqrt{2\pi(1/2)} \left[\frac{1}{\sqrt{2\pi(1/2)}} e^{-(1/2)\left[\frac{(\beta-y)^2}{(1/2)}\right]} \right].$$

The integral from minus to plus infinity of the quantity inside the brackets is 1, so

$$g(x, y) = \sqrt{\pi} e^{-\left[(x-a)^2\right]}$$

which is a blurred version of the original image.

Problem 5.17

Because the motion in the x- and y-directions are independent (motion is in the vertical (x) direction only at first, and then switching to motion only in the horizontal (y) direction) this problem can be solved in two steps. The first step is identical to the analysis that resulted in Eq. (5.6-10), which gives the blurring function due to vertical motion only:

$$H_1(u, v) = \frac{T_1}{\pi u a} \sin(\pi u a) e^{-j\pi u a},$$

where we are representing linear motion by the equation $x_0(t) = at/T_1$.The function $H_1(u, v)$ would give us a blurred image in the vertical direction. That blurred image is the image that would then start moving in the horizontal direction and to which horizontal blurring would be applied. This is nothing more than applying a second filter with transfer function

$$H_2(u, v) = \frac{T_2}{\pi u b} \sin(\pi u b) e^{-j\pi u b}$$

where we assumed the form $y_0(t) = bt/T_2$ for motion in the y-direction. Therefore, the overall blurring transfer function is given by the product of these two functions:

$$H(u, v) = \frac{T_1 T_2}{(\pi u a)(\pi u b)} \sin(\pi u a) \sin(\pi u b) e^{-j\pi(ua - ub)},$$

and the overall blurred image is

$$g(x,y) = \Im^{-1}\left[H(u,v)F(u,v)\right]$$

where $F(u,v)$ is the Fourier transform of the input image.

Problem 5.18

Following the procedure in Section 5.6.3,

$$\begin{aligned} H(u,v) &= \int_0^T e^{-j2\pi u x_0(t)} dt \\ &= \int_0^T e^{-j2\pi u\left[(1/2)at^2\right]} dt \\ &= \int_0^T e^{-j\pi u a t^2} dt \\ &= \int_0^T \left[\cos(\pi u a t^2) - j\sin(\pi u a t^2)\right] dt \\ &= \sqrt{\frac{T^2}{2\pi u a T^2}}\left[C(\sqrt{\pi u a}T) - jS(\sqrt{\pi u a}T)\right] \end{aligned}$$

where

$$C(x) = \sqrt{\frac{2\pi}{T}}\int_0^x \cos t^2 dt$$

and

$$S(x) = \sqrt{\frac{2}{\pi}}\int_0^x \sin t^2 dt.$$

These are Fresnel cosine and sine integrals. They can be found, for example, the *Handbook of Mathematical Functions*, by Abramowitz, or other similar reference.

Problem 5.19

A basic approach for restoring a rotationally blurred image is to convert the image from rectangular to polar coordinates. The blur will then appear as one-dimensional uniform motion blur along the θ-axis. Any of the techniques discussed in this chapter for handling uniform blur along one dimension can then be applied to the problem. The image is then converted back to rectangular coordinates after restoration. The mathematical solution is simple. For any pixel with rectangular coordinates (x,y) we generate a corresponding pixel with polar coordinates (r,θ), where

$$r = \sqrt{x^2 + y^2}$$

and

$$\theta = \tan^{-1}\left(\frac{y}{x}\right).$$

A display of the resulting image would shown an image that is blurred along the θ-axis and would, in addition, appear distorted due to the coordinate conversion. Since the extent of the rotational blur is known (it is given as $\pi/8$ radians), we can use the same solution we used for uniform linear motion (Section 5.6.3), with $x = \theta$ and $y = r$ to obtain the transfer function. Any of the methods in Sections 5.7 through 5.9 then become applicable.

Problem 5.20

Measure the average value of the background. Set all pixels in the image, except the cross hairs, to that gray level. Denote the Fourier transform of this image by $G(u, v)$. Since the characteristics of the cross hairs are given with a high degree of accuracy, we can construct an image of the background (of the same size) using the background gray levels determined previously. We then construct a model of the cross hairs in the correct location (determined from he given image) using the provided dimensions and gray level of the crosshairs. Denote by $F(u, v)$ the Fourier transform of this new image . The ratio $G(u, v)/F(u, v)$ is an estimate of the blurring function $H(u, v)$. In the likely event of vanishing values in $F(u, v)$, we can construct a radially-limited filter using the method discussed in connection with Fig. 5.27. Because we know $F(u, v)$ and $G(u, v)$, and an estimate of $H(u, v)$, we can also refine our estimate of the blurring function by substituting G and H in Eq. (5.8-3) and adjusting K to get as close as possible to a good result for $F(u, v)$ [the result can be evaluated visually by taking the inverse Fourier transform]. The resulting filter in either case can then be used to deblur the image of the heart, if desired.

Problem 5.21

The key to solving this problem is to recognize that the given function

$$h(r) = \frac{r^2 - \sigma^2}{\sigma^4} e^{-r^2/2\sigma^2}$$

where $r^2 = x^2 + y^2$, is the Laplacian (second derivative with respect to r) of the function

$$h_0(r) = e^{-r^2/2\sigma^2}.$$

That is, $\nabla^2[h_0(r)]$ is equal to the given function. Then we know from Eq. (4.4-7) that,

for a function $f(x, y)$,

$$\Im\left[\nabla^2 f(x,y)\right] = -(u^2+v^2)F(u,v).$$

Thus, we have reduced the problem to finding the Fourier transform of $e^{-r^2/2\sigma^2}$, which is in the form of a Gaussian function. From Table 4.1, we note from the Gaussian transform pair that the Fourier transform of a function of the form $e^{-(x^2+y^2)/2\sigma^2}$ is

$$\Im\left[e^{-(x^2+y^2)/2\sigma^2}\right] = \sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)}.$$

Therefore, the Fourier transform of the given degradation function is

$$\begin{aligned} H(u,v) &= \Im\left[\frac{r^2-\sigma^2}{\sigma^4}e^{-r^2/2\sigma^2}\right] = \Im\left[\nabla^2 h_0(r)\right] \\ &= -(u^2+v^2)F(u,v) \\ &= -\sqrt{2\pi}\sigma(u^2+v^2)e^{-2\pi^2\sigma^2(x^2+y^2)}. \end{aligned}$$

Problem 5.22

This is a simple plugin problem. Its purpose is to gain familiarity with the various terms of the Wiener filter. From Eq. (5.8-3),

$$H_W(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]$$

where

$$\begin{aligned} |H(u,v)|^2 &= H^*(u,v)H(u,v) \\ &= 2\pi\sigma^2(u^2+v^2)^2 e^{-4\pi^2\sigma^2(x^2+y^2)}. \end{aligned}$$

Then,

$$H_W(u,v) = -\left[\frac{\sqrt{2\pi}\sigma(u^2+v^2)e^{-2\pi^2\sigma^2(x^2+y^2)}}{\left[2\pi\sigma^2(u^2+v^2)^2 e^{-4\pi^2\sigma^2(x^2+y^2)}\right] + K}\right].$$

Problem 5.23

This also is a simple plugin problem, whose purpose is the same as the previous problem. From Eq. (5.9-4)

$$\begin{aligned} H_C(u,v) &= \frac{H^*(u,v)}{|H(u,v)|^2 + \gamma|P(u,v)|^2} \\ &= -\frac{\sqrt{2\pi}\sigma(u^2+v^2)e^{-2\pi^2\sigma^2(x^2+y^2)}}{2\pi\sigma^2(u^2+v^2)^2 e^{-4\pi^2\sigma^2(x^2+y^2)} + \gamma|P(u,v)|^2} \end{aligned}$$

where $P(u, v)$ is the Fourier transform of the Laplacian operator [Eq. (5.9-5)]. This is as far as we can reasonably carry this problem. It is worthwhile pointing out to students

that a closed expression for the transform of the Laplacian operator was obtained in Problem 4.19. However, substituting that solution for $P(u,v)$ here would only increase the number of terms in the filter and would not aid at all in simplifying the expression.

Problem 5.24

Because the system is assumed linear and position invariant, it follows that Eq. (5.5-17) holds. Furthermore, we can use superposition and obtain the response of the system first to $F(u,v)$ and then to $N(u,v)$. The sum of the two individual responses gives the complete response. First, using only $F(u,v)$,

$$G_1(u,v) = H(u,v)F(u,v)$$

and

$$|G_1(u,v)|^2 = |H(u,v)|^2 |F(u,v)|^2 .$$

Then, using only $N(u,v)$,

$$G_2(u,v) = N(u,v)$$

and

$$|G_2(u,v)|^2 = |N(u,v)|^2$$

so that

$$\begin{aligned} |G(u,v)|^2 &= |G_1(u,v)|^2 + |G_2(u,v)|^2 \\ &= |H(u,v)|^2 |F(u,v)|^2 + |N(u,v)|^2 . \end{aligned}$$

Problem 5.25

(a) It is given that

$$\left|\hat{F}(u,v)\right|^2 = |R(u,v)|^2 |G(u,v)|^2 .$$

From Problem 5.24,

$$\left|\hat{F}(u,v)\right|^2 = |R(u,v)|^2 \left[|H(u,v)|^2 |F(u,v)|^2 + |N(u,v)|^2\right] .$$

Forcing $\left|\hat{F}(u,v)\right|^2$ to equal $|F(u,v)|^2$ gives

$$R(u,v) = \left[\frac{|F(u,v)|^2}{|H(u,v)|^2 |F(u,v)|^2 + |N(u,v)|^2}\right]^{1/2} .$$

(b)

$$\begin{aligned}\hat{F}(u,v) &= R(u,v)G(u,v) \\ &= \left[\frac{|F(u,v)|^2}{|H(u,v)|^2|F(u,v)|^2 + |N(u,v)|^2}\right]^{1/2} G(u,v) \\ &= \left[\frac{1}{|H(u,v)|^2 + \frac{|N(u,v)|^2}{|F(u,v)|^2}}\right]^{1/2} G(u,v)\end{aligned}$$

and, because $|F(u,v)|^2 = S_f(u,v)$ and $|N(u,v)|^2 = S_\eta(u,v)$,

$$\hat{F}(u,v) = \left[\frac{1}{|H(u,v)|^2 + \frac{S_\eta(u,v)}{S_f(u,v)}}\right]^{1/2} G(u,v).$$

Problem 5.26

One possible solution: (1) Average images to reduce noise. (2) obtain blurred image of a bright, single star to simulate an impulse (the star should be as small as possible in the field of view of the telescope to simulate an impulse as closely as possible. (3) The Fourier transform of this image will give $H(u,v)$. (4) Use a Wiener filter and vary K until the sharpest image possible is obtained.

Problem 5.27

The basic idea behind this problem is to use the camera and representative coins to model the degradation process and then utilize the results in an inverse filter operation. The principal steps are as follows:

1. Select coins as close as possible in size and content as the lost coins. Select a background that approximates the texture and brightness of the photos of the lost coins.

2. Set up the museum photographic camera in a geometry as close as possible to give images that resemble the images of the lost coins (this includes paying attention to illumination). Obtain a few test photos. To simplify experimentation, obtain a TV camera capable of giving images that resemble the test photos. This can be done by connecting the camera to an image processing system and generating digital images, which will be used in the experiment.

3. Obtain sets of images of each coin with different lens settings. The resulting images should approximate the aspect angle, size (in relation to the area occupied by the background), and blur of the photos of the lost coins.

4. The lens setting for each image in (3) is a model of the blurring process for the corresponding image of a lost coin. For each such setting, remove the coin and background and replace them with a small, bright dot on a uniform background, or other mechanism to approximate an impulse of light. Digitize the impulse. Its Fourier transform is the transfer function of the blurring process.

5. Digitize each (blurred) photo of a lost coin, and obtain its Fourier transform. At this point, we have $H(u, v)$ and $G(u, v)$ for each coin.

6. Obtain an approximation to $F(u, v)$ by using a Wiener filter. Equation (5.8-3) is particularly attractive because it gives an additional degree of freedom (K) for experimenting.

7. The inverse Fourier transform of each approximate $F(u, v)$ gives the restored image. In general, several experimental passes of these basic steps with various different settings and parameters are required to obtain acceptable results in a problem such as this.

Problem 5.28

Using triangular regions means three tiepoints, so we can solve the following set of linear equations for six coefficients:

$$\begin{aligned} x' &= c_1 x + c_2 y + c_3 \\ y' &= c_4 x + c_5 y + c_6 \end{aligned}$$

to implement spatial transformations. We also solve the following equation for three coefficients

$$v(x', y') = ax' + by' + c$$

to implement gray level interpolation.

6 Problem Solutions

Problem 6.1

From the figure, $x = 0.43$ and $y = 0.4$. Since $x + y + z = 1$, it follows that $z = 0.17$. These are the trichromatic coefficients. We are interested in tristimulus values X, Y, and Z, which are related to the trichromatic coefficients by Eqs. (6.1-1) through (6.1-3). We note however, that all the tristimulus coefficients are divided by the same constant, so their percentages relative to the trichromatic coefficients are the same as those of the coefficients. Thus, the answer is $X = 0.43$, $Y = 0.40$, and $Z = 0.17$.

Problem 6.2

Denote by c the given color, and let its coordinates be denoted by (x_0, y_0). The distance between c and c_1 is

$$d(c, c_1) = \left[(x_0 - x_1)^2 + (y_0 - y_1)^2\right]^{1/2}.$$

Similarly the distance between c_1 and c_2

$$d(c_1, c_2) = \left[(x_1 - x_2)^2 + (y_1 - y_2)^2\right]^{1/2}.$$

The percentage p_1 of c_1 in c is

$$p_1 = \frac{d(c_1, c_2) - d(c, c_1)}{d(c_1, c_2)} \times 100.$$

The percentage p_2 of c_2 is simply $p_2 = 100 - p_1$. In the preceding equation we see, for example, that when $c = c_1$, then $d(c, c_1) = 0$ and it follows that $p_1 = 100\%$ and $p_2 = 0\%$. Similarly, when $d(c, c_1) = d(c_1, c_2)$, it follows that $p_1 = 0\%$ and $p_2 = 100\%$. Values in between are easily seen to follow from these simple relations.

Problem 6.3

Consider Fig. P6.3, in which c_1, c_2, and c_3 are the given vertices of the color triangle and c is an arbitrary color point contained within the triangle or on its boundary. The key to solving this problem is to realize that any color on the border of the triangle is made up of proportions from the two vertices defining the line segment that contains the point. The contribution to a point on the line by the color vertex opposite this line is 0% .

The line segment connecting points c_3 and c is shown extended (dashed segment) until it intersects the line segment connecting c_1 and c_2. The point of intersection is denoted c_0. Because we have the values of c_1 and c_2, if we knew c_0, we could compute the percentages of c_1 and c_2 contained in c_0 by using the method described in Problem 6.2. Denote the ratio of the content of c_1 and c_2 in c_0 be denoted by R_{12}. If we now add color c_3 to c_0, we know from Problem 6.2 that the point will start to move toward c_3 along the line shown. For any position of a point along this line we could determine the percentage of c_3 and c_0, again, by using the method described in Problem 6.2. What is important to keep in mind that the ratio R_{12} will remain the same for any point along the segment connecting c_3 and c_0. The color of the points along this line is different for each position, but the *ratio* of c_1 to c_2 will remain constant.

So, if we can obtain c_0, we can then determine the ratio R_{12}, and the percentage of c_3, in color c. The point c_0 is not difficult to obtain. Let $y = a_{12}x + b_{12}$ be the straight line containing points c_1 and c_2, and $y = a_{3c}x + b_{3c}$ the line containing c_3 and c. The intersection of these two lines gives the coordinates of c_0. The lines can be determined uniquely because we know the coordinates of the two point pairs needed to determine the line coefficients. Solving for the intersection in terms of these coordinates is straightforward, but tedious. Our interest here is in the fundamental method, not the mechanics of manipulating simple equations so we don not give the details.

At this juncture we have the percentage of c_3 and the ratio between c_1 and c_2. Let the percentages of these three colors composing c be denoted by p_1, p_2, and p_3 respectively. Since we know that $p_1 + p_2 = 100 - p_3$, and that $p_1/p_2 = R_{12}$, we can solve for p_1 and p_2. Finally, note that this problem could have been solved the same way by intersecting one of the other two sides of the triangle. Going to another side would be necessary, for example, if the line we used in the preceding discussion had an infinite slope. A simple test to determine if the color of c is equal to any of the vertices should be the first step in the procedure; in this case no additional calculations would be required.

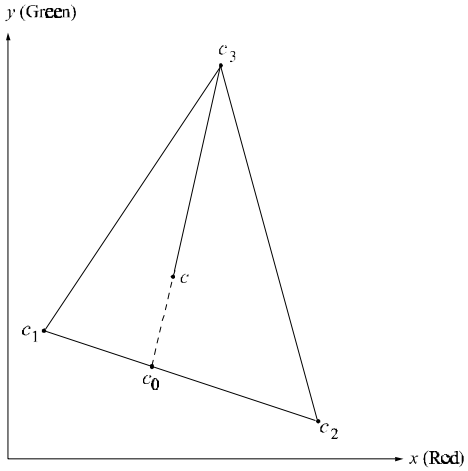


Figure P6.3

Problem 6.4

Use color filters sharply tuned to the wavelengths of the colors of the three objects. Thus, with a specific filter in place, only the objects whose color corresponds to that wavelength will produce a predominant response on the monochrome camera. A motorized filter wheel can be used to control filter position from a computer. If one of the colors is white, then the response of the three filters will be approximately equal and high. If one of the colors is black, the response of the three filters will be approximately equal and low.

Problem 6.5

At the center point we have

$$\frac{1}{2}R + \frac{1}{2}B + G = \frac{1}{2}(R + G + B) + \frac{1}{2}G = \text{midgray} + \frac{1}{2}G$$

which looks to a viewer like pure green with a boot in intensity due to the additive gray component.

Problem 6.6

For the image given, the maximum intensity and saturation requirement means that the RGB component values are 0 or 1. We can create the following table with 0 and 255

representing black and white, respectively:

Table P6.6

Color	R	G	B	Mono R	Mono G	Mono B
Black	0	0	0	0	0	0
Red	1	0	0	255	0	0
Yellow	1	1	0	255	255	0
Green	0	1	0	0	255	0
Cyan	0	1	1	0	255	255
Blue	0	0	1	0	0	255
Magenta	1	0	1	255	0	255
White	1	1	1	255	255	255
Gray	0.5	0.5	0.5	128	128	128

Thus, we get the monochrome displays shown in Fig. P6.6.

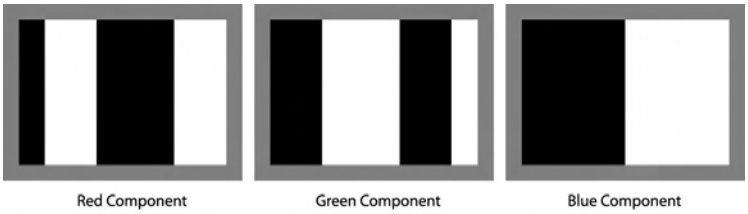


Figure P6.6

Problem 6.7

There are $2^8 = 256$ possible values in each 8-bit image. For a color to be gray, all RGB components have to be equal, so there are 256 shades of gray.

Problem 6.8

(a) All pixel values in the Red image are 255. In the Green image, the first column is all 0's; the second column all 1's; and so on until the last column, which is composed of all 255's. In the Blue image, the first row is all 255's; the second row all 254's, and so on until the last row which is composed of all 0's.

(b) Let the axis numbering be the same as in Fig. 6.7. Then: $(0, 0, 0)$ = white, $(1, 1, 1,)$ = black, $(1, 0, 0)$ = cyan, $(1, 1, 0)$ = blue, $(1, 0, 1)$ = green, $(0, 1, 1)$ = red, $(0, 0, 1)$ = yellow, $(0, 1, 0)$ = magenta.

(c) The ones that do not contain the black or white point are fully saturated. The others decrease in saturation from the corners toward the black or white point.

Problem 6.9

(a) For the image given, the maximum intensity and saturation requirement means that the RGB component values are 0 or 1. We can create Table P6.9 using Eq. (6.2-1):

Table P6.9

Color	R	G	B	C	M	Y	Mono C	Mono M	Mono Y
Black	0	0	0	1	1	1	255	255	255
Red	1	0	0	0	1	1	0	255	255
Yellow	1	1	0	0	0	1	0	0	255
Green	0	1	0	1	0	1	255	0	255
Cyan	0	1	1	1	0	0	255	0	0
Blue	0	0	1	1	1	0	255	255	0
Magenta	1	0	1	0	1	0	0	255	0
White	1	1	1	0	0	0	0	0	0
Gray	0.5	0.5	0.5	0.5	0.5	0.5	128	128	128

Thus, we get the monochrome displays shown in Fig. P6.9(a).

(b) The resulting display is the complement of the starting RGB image. From left to right, the color bars are (in accordance with Fig. 6.32) white, cyan, blue, magenta, red, yellow, green, and black. The middle gray background is unchanged.

Figure P6.9

Problem 6.10

Equation (6.2-1) reveals that each component of the CMY image is a function of a single component of the corresponding RGB image—C is a function of R, M of G, and Y of B. For clarity, we will use a prime to denote the CMY components. From Eq. (6.5-6), we know that

$$s_i = kr_i$$

for $i = 1, 2, 3$ (for the R, G, and B components). And from Eq. (6.2-1), we know that the CMY components corresponding to the r_i and s_i (which we are denoting with primes) are

$$r_i' = 1 - r_i$$

and

$$s_i' = 1 - s_i.$$

Thus,

$$r_i = 1 - r_i'$$

and

$$s_i' = 1 - s_i = 1 - kr_i = 1 - k\left(1 - r_i'\right)$$

so that

$$s_i' = kr_i' + (1 - k).$$

Problem 6.11

(a) The purest green is 00FF00, which corresponds to cell (7, 18).

(b) The purest blue is 0000FF, which corresponds to cell (12, 13).

Problem 6.12

Using Eqs. (6.2-2) through (6.2-4), we get the results shown in Table P6.12. Note that, in accordance with Eq. (6.2-2), hue is undefined when $R = G = B$ since $\theta = \cos^{-1}\left(\frac{0}{0}\right)$. In addition, saturation is undefined when $R = G = B = 0$ since Eq. (6.2-3) yields

$S = 1 - \frac{3\min(0)}{3 \cdot 0} = 1 - \frac{0}{0}$. Thus, we get the monochrome display shown in Fig. P6.12.

Table P6.12

Color	R	G	B	H	S	I	Mono H	Mono S	Mono I
Black	0	0	0	–	0	0	–	–	0
Red	1	0	0	0	1	0.33	0	255	85
Yellow	1	1	0	0.17	1	0.67	43	255	170
Green	0	1	0	0.33	1	0.33	85	255	85
Cyan	0	1	1	0.5	1	0.67	128	255	170
Blue	0	0	1	0.67	1	0.33	170	255	85
Magenta	1	0	1	0.83	1	0.67	213	255	170
White	1	1	1	–	0	1	–	0	255
Gray	0.5	0.5	0.5	–	0	0.5	–	0	128

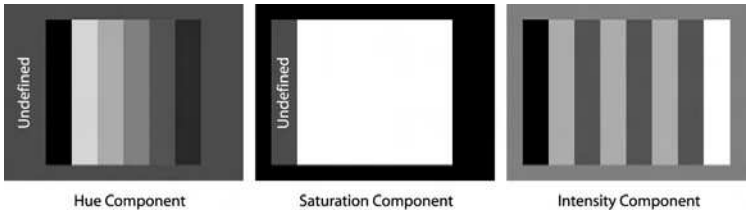


Figure P6.12

Problem 6.13

With reference to the HSI color circle in Fig. 6.14(b), deep purple is found at approximately $270°$. To generate a color rectangle with the properties required in the problem statement, we choose a fixed intensity I, and maximum saturation (these are spectrum colors, which are supposed to be fully saturated), S. The first column in the rectangle uses these two values and a hue of $270°$. The next column (and all subsequent columns) would use the same values of I and S, but the hue would be decreased to $269°$, and so on all the way down to a hue of $0°$, which corresponds to red. If the image is limited to 8 bits, then we can only have 256 variations in hue in the range from $270°$ down to $0°$, which will require a different uniform spacing than one degree increments or, alternatively, starting at a $255°$ and proceed in increments of 1, but this would leave out most of the purple. If we have more than eight bits, then the increments can be smaller. Longer strips also can be made by duplicating column values.

Problem 6.14

There are two important aspects to this problem. One is to approach it in HSI space and the other is to use polar coordinates to create a hue image whose values grow as a function of angle. The center of the image is the middle of whatever image area is used. Then, for example, the values of the hue image along a radius when the angle is 0° would be all 0's. The angle then is incremented by, say, one degree, and all the values along that radius would be 1's, and so on. Values of the saturation image decrease linearly in all radial directions from the origin. The intensity image is just a specified constant. With these basics in mind it is not difficult to write a program that generates the desired result.

Problem 6.15

The hue, saturation, and intensity images are shown in Fig. P6.15, from left to right.

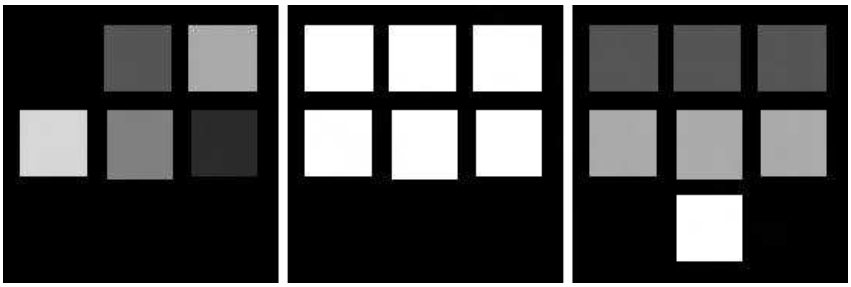


Figure P6.15

Problem 6.16

(a) It is given that the colors in Fig. 6.16(a) are primary spectrum colors. It also is given that the gray-level images in the problem statement are 8-bit images. The latter condition means that hue (angle) can only be divided into a maximum number of 256 values. Since hue values are represented in the interval from 0° to 360° this means that for an 8-bit image the increments between contiguous hue values are now $360/255$. Another way of looking at this is that the entire [0, 360] hue scale is compressed to the range [0, 255]. Thus, for example, yellow (the first primary color we encounter), which

is 60° now becomes 43 (the closest integer) in the integer scale of the 8-bit image shown in the problem statement. Similarly, green, which is 120° becomes 85 in this image. From this we easily compute the values of the other two regions as being 170 and 213. The region in the middle is pure white [equal proportions of red green and blue in Fig. 6.61(a)] so its hue by definition is 0. This also is true of the black background.

(b) The colors are spectrum colors, so they are fully saturated. Therefore, the values shown of 255 applies to all circle regions. The region in the center of the color image is white, so its saturation is 0.

(c) The key to getting the values in this figure is to realize that the center portion of the color image is white, which means equal intensities of fully saturated red, green, and blue. Therefore, the value of both darker gray regions in the intensity image have value 85 (i.e., the same value as the other corresponding region). Similarly, equal proportions of the secondaries yellow, cyan, and magenta produce white, so the two lighter gray regions have the same value (170) as the region shown in the figure. The center of the image is white, so its value is 255.

Problem 6.17

(a) Because the infrared image which was used in place of the red component image has very high gray-level values.

(b) The water appears as solid black (0) in the near infrared image [Fig. 6.27(d)]. Threshold the image with a threshold value slightly larger than 0. The result is shown in Fig. P6.17. It is clear that coloring all the black points in the desired shade of blue presents no difficulties.

(c) Note that the predominant color of natural terrain is in various shades of red. We already know how to take out the water from (b). Thus a method that actually removes the ”background” of red and black would leave predominantly the other man-made structures, which appear mostly in a bluish light color. Removal of the red [and the black if you do not want to use the method as in (b)] can be done by using the technique discussed in Section 6.7.2.

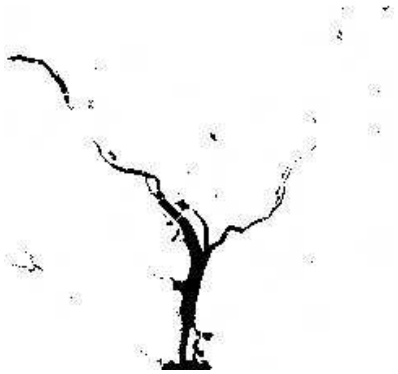


Figure P6.17

Problem 6.18

Using Eq. (6.2-3), we see that the basic problem is that many different colors have the same saturation value. This was demonstrated in Problem 6.12, where pure red, yellow, green, cyan, blue, and magenta all had a saturation of 1. That is, as long as any one of the RGB components is 0, Eq. (6.2-3) yields a saturation of 1.

Consider RGB colors (1, 0, 0) and (0, 0.59, 0), which represent a red and a green. The HSI triplets for these colors [per Eq. (6.4-2) through (6.4-4)] are (0, 1, 0.33) and (0.33, 1, 0.2), respectively. Now, the complements of the beginning RGB values (see Section 6.5.2) are (0, 1, 1) and (1, 0.41, 1), respectively; the corresponding colors are cyan and magenta. Their HSI values [per Eqs. (6.4-2) through (6.4-4)] are (0.5, 1, 0.66) and (0.83, 0.48, 0.8), respectively. Thus, for the red, a starting saturation of 1 yielded the cyan "complemented" saturation of 1, while for the green, a starting saturation of 1 yielded the magenta "complemented" saturation of 0.48. That is, the same starting saturation resulted in two different "complemented" saturations. Saturation alone is not enough information to compute the saturation of the complemented color.

Problem 6.19

The complement of a color is the color opposite it on the color circle of Fig. 6.32. The hue component is the angle from red in a counterclockwise direction normalized by 360 degrees. For a color on the top half of the circle (i.e., $0 \leq H \leq 0.5$), the hue of the complementary color is $H + 0.5$. For a color on the bottom half of the circle (i.e., for

$0.5 \leq H \leq 1$), the hue of the complement is $H - 0.5$.

Problem 6.20

The RGB transformations for a complement [from Fig. 6.33(b)] are:

$$s_i = 1 - r_i$$

where $i = 1, 2, 3$ (for the R, G, and B components). But from the definition of the CMY space in Eq. (6.2-1), we know that the CMY components corresponding to r_i and s_i, which we will denote using primes, are

$$r_i' = 1 - r_i$$
$$s_i' = 1 - s_i.$$

Thus,

$$r_i = 1 - r_i'$$

and

$$s_i' = 1 - s_i = 1 - (1 - r_i) = 1 - (1 - (1 - r_i'))$$

so that

$$s' = 1 - r_i'.$$

Problem 6.21

The RGB transformation should darken the highlights and lighten the shadow areas, effectively compressing all values toward the midtones. The red, green, and blue components should be transformed with the same mapping function so that the colors do not change. The general shape of the curve would be as shown in Fig. P6.21.

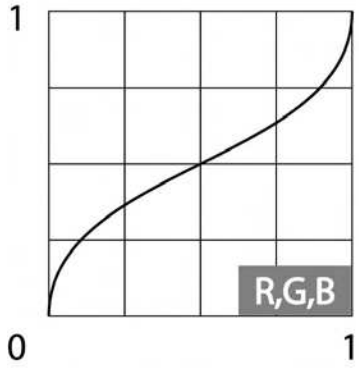


Figure P6.21

Problem 6.22

Based on the discussion is Section 6.5.4 and with reference to the color wheel in Fig. 6.32, we can decrease the proportion of yellow by (1) decreasing yellow, (2) increasing blue, (3) increasing cyan and magenta, or (4) decreasing red and green.

Problem 6.23

The $L^*a^*b^*$ components are computed using Eqs. (6.5-9) through (6.5-12). Reference white is $R = G = B = 1$. The computations are best done in a spreadsheet, as shown in Table P6.23.

Table P6.23

Color	R	G	B	X	Y	Z	$\frac{X}{X_w}$	$\frac{Y}{Y_w}$	$\frac{Z}{Z_w}$	$h\left(\frac{X}{X_w}\right)$	$h\left(\frac{Y}{Y_w}\right)$	$h\left(\frac{Z}{Z_w}\right)$	L^*	a^*	b^*
Ref.	1	1	1	0.95	1.00	1.10	1	1	1	1	1	1	100	0	0
Black	0	0	0	0	0	0	0	0	0	0.14	0.14	0.14	0	0	0
Red	1	0	0	0.59	0.29	0	0.62	0.29	0	0.85	0.66	0.14	83	95	105
Yellow	1	1	0	0.77	0.90	0.07	0.81	0.90	0.06	0.93	0.96	0.40	92	-16	113
Green	0	1	0	0.18	0.61	0.07	0.19	0.61	0.06	0.57	0.85	0.40	51	-136	90
Cyan	0	1	1	0.36	0.71	1.09	0.38	0.71	1	0.73	0.89	1	68	-84	-22
Blue	0	0	1	0.18	0.11	1.02	0.19	0.11	0.94	0.58	0.47	0.98	51	53	-101
Magenta	1	0	1	0.77	0.40	1.02	0.81	0.40	0.94	0.93	0.73	0.98	92	100	-49
White	1	1	1	0.95	1.00	1.10	1	1	1	1	1	1	100	0	0
Gray	0.5	0.5	0.5	0.48	0.50	0.55	0.5	0.5	0.5	0.79	0.79	0.79	76	0	0

Problem 6.24

The conceptually simplest approach is to transform every input image to the HSI color space, perform histogram specification per the discussion in Section 3.3.2 on the intensity (I) component only (leaving H and S alone), and convert the resulting intensity component with the original hue and saturation components back to the starting color space.

Problem 6.25

(a) The boundary between red and green becomes thickened and yellow as a result of blurring between the red and green primaries (recall that yellow is the color between green and red in, for example, Fig. 6.14). The boundary between green and blue is similarly blurred into a cyan color. The result is shown in Fig. P6.25.

(b) Blurring has no effect in this case. The intensity image is constant (at its maximum value) because the pure colors are fully saturated.

Problem 6.26

This is a simple problem to encourage the student to think about the meaning of the elements in Eq. (6.7-2). When $\mathbf{C} = \mathbf{I}$, it follows that $\mathbf{C}^{-1} = \mathbf{I}$ and Eq. (6.7-2) becomes

$$D(\mathbf{z}, \mathbf{a}) = \left[(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a})\right]^{1/2}.$$

But the term inside the brackets is recognized as the inner product of the vector $(\mathbf{z} - \mathbf{a})$ with itself, which, by definition, is equal to the right side of Eq. (6.7-1).

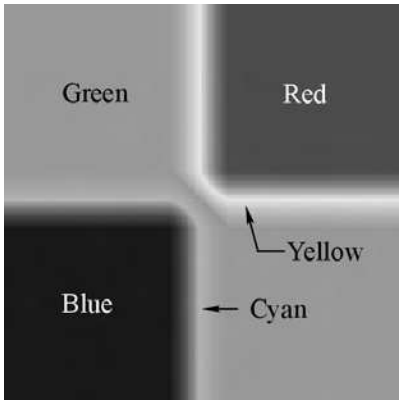


Figure P6.25

Problem 6.27

(a) The cube is composed of 6 intersecting planes in RGB space. The general equation for such planes is

$$a\, z_R + b\, z_G + c\, z_B + d = 0$$

where a, b, c, and d are parameters and the z's are the components of any point (vector) $\mathbf{z}$ in RGB space lying on the plane. If an RGB point $\mathbf{z}$ does not lie on the plane, and its coordinates are substituted in the preceding equation, then equation will give either a positive or a negative value; it will not yield zero. We say that $\mathbf{z}$ lies on the positive or negative side of the plane, depending on whether the result is positive or negative. We can change the positive side of a plane by multiplying its coefficients (except d) by -1. Suppose that we test the point $\mathbf{a}$ given in the problem statement to see whether it is on the positive or negative side each of the six planes composing the box, and change the

coefficients of any plane for which the result is negative. Then, $\mathbf{a}$ will lie on the positive side of all planes composing the bounding box. In fact all points inside the bounding box will yield positive values when their coordinates are substituted in the equations of the planes. Points outside the box will give at least one negative or zero value. Thus, the method consists of substituting an unknown color point in the equations of all six planes. If all the results are positive, the point is inside the box; otherwise it is outside the box. A flow diagram is asked for in the problem statement to make it simpler to evaluate the student's line of reasoning.

(b) If the box is lined up with the RGB coordinate axes, then the planes intersect the RGB coordinate planes perpendicularly. The intersections of pairs of parallel planes establish a range of values along each of the RGB axis that must be checked to see if the if an unknown point lies inside the box or not. This can be done on an image per image basis (i.e., the three component images of an RGB image), designating by 1 a coordinate that is within its corresponding range and 0 otherwise. These will produce three binary images which, when ANDed, will give all the points inside the box.

Problem 6.28

The sketch is an elongated ellipsoidal figure in which the length lined up with the R-axis is 8 times longer that the other two dimensions. In other words, the figure looks like a blimp aligned with the R-axis.

Problem 6.29

Set one of the three primary images to a constant value (say, 0), then consider the two images shown in Fig. P6.29. If we formed an RGB composite image by letting the image on the left be the red component and the image on the right the green component, then the result would be an image with a green region on the left separated by a vertical edge from a red region on the right. To compute the gradient of each component image we take second-order partial derivatives. In this case, only the component of the derivative in the horizontal direction is nonzero. If we model the edge as a ramp edge [Fig. 3.38(b)] then a profile of the derivative image would appear as shown in Fig. P6.29. The magnified view shows clearly that the derivatives of the two images are mirrors of each other. Thus, if we computed the gradient vector of each image and added the results as suggested in the problem statement, the components of the gradient would cancel out, giving a zero gradient for a color image that has a clearly defined edge between two dif-

ferent color regions. This simple example illustrates that the gradient vector of a color image is not equivalent to the result of forming a color gradient vector from the sum of the gradient vectors of the individual component images.

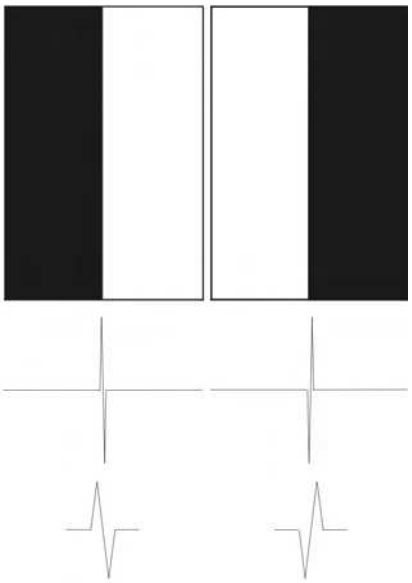


Figure P6.29

7 Problem Solutions

Problem 7.1

Following the explanation in Example 7.1, the decoder is as shown in Fig. P7.1

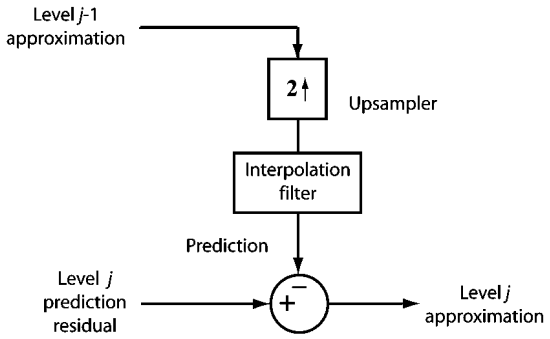


Figure P7.1

Problem 7.2

A mean approximation pyramid is formed by forming 2×2 block averages. Since the starting image is of size 4×4, $J = 2$, and $f(x, y)$ is placed in level 2 of the mean approximation pyramid. The level 1 approximation is (by taking 2×2 block averages over $f(x, y)$ and subsampling):

$$\begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix}$$

and the level 0 approximation is similarly [8.5]. The completed mean approximation pyramid is

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix} \begin{bmatrix} 8.5 \end{bmatrix}.$$

Since no interpolation filtering is specified, pixel replication is used in the generation of the mean prediction residual pyramid levels. Level 0 of the prediction residual pyramid is the lowest resolution approximation, [8.5]. The level 2 prediction residual is obtained by upsampling the level 1 approximation and subtracting it from the level 2 (original image). Thus, we get

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} - \begin{bmatrix} 3.5 & 3.5 & 5.5 & 5.5 \\ 3.5 & 3.5 & 5.5 & 5.5 \\ 11.5 & 11.5 & 13.5 & 13.5 \\ 11.5 & 11.5 & 13.5 & 13.5 \end{bmatrix}$$

$$= \begin{bmatrix} -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \\ -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \end{bmatrix}.$$

Similarly, the level 1 prediction residual is obtained by upsampling the level 0 approximation and subtracting it from the level 1 approximation to yield

$$\begin{bmatrix} 3.5 & 5.5 \\ 11.5 & 13.5 \end{bmatrix} - \begin{bmatrix} 8.5 & 8.5 \\ 8.5 & 8.5 \end{bmatrix} = \begin{bmatrix} -5 & -3 \\ 3 & 5 \end{bmatrix}.$$

The mean prediction residual pyramid is therefore

$$\begin{bmatrix} -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \\ -2.5 & -1.5 & -2.5 & -1.5 \\ 1.5 & 2.5 & 1.5 & 2.5 \end{bmatrix} \begin{bmatrix} -5 & -3 \\ 3 & 5 \end{bmatrix} [8.5].$$

Problem 7.3

The number of elements in a $J+1$ level pyramid is bounded by 4/3 (see Section 7.1.1):

$$2^{2J}\left[1 + \frac{1}{(4)^1} + \frac{1}{(4)^2} + \cdots + \frac{1}{(4)^J}\right] \leq \frac{4}{3}2^{2J}$$

for $J > 0$. We can generate Table P7.3:

Table P7.3

J	Pyramid Elements	Compression Ratio
0	1	1
1	5	$5/4 = 1.25$
2	21	$21/16 = 1.3125$
3	85	$85/86 = 1.328$
$\vdots$		
∞		$4/3 = 1.33$

All but the trivial case $(J = 0)$ are expansions. The expansion factor is a function of and bounded by 4/3 or 1.33.

Problem 7.4

(a) The QMF filters must satisfy Eqs. (7.1-9) and (7.1-10). From Table 7.1, $G_0(z) = H_0(z)$ and $H_1(z) = H_0(-z)$, so $H_1(-z) = H_0(z)$. Thus, beginning with Eq. (7.1-9),

$$\begin{aligned} H_0(-z)G_0(z) + H_1(-z)G_1(z) &= 0 \\ H_0(-z)H_0(z) - H_0(z)H_0(-z) &= 0 \\ 0 &= 0. \end{aligned}$$

Similarly, beginning with Eq. (7.1-10) and substituting for $H_1(z)$, $G_0(z)$, and $G_1(z)$ from rows 2, 3, and 4 of Table 7.1, we get

$$\begin{aligned} H_0(z)G_0(z) + H_1(z)G_1(z) &= 2 \\ H_0(z)H_0(z) + H_0(-z)[-H_0(-z)] &= 2 \\ H_0^2(z) - H_0^2(-z) &= 2 \end{aligned}$$

which is the design equation for the $H_0(z)$ prototype filter in row 1 of the table.

(b) The orthonormal filter proof follows the QMF proof in (a). For Eq. (7.1-9), we get

$$\begin{aligned} H_0(-z)G_0(z) + H_1(z)G_1(z) &= 0 \\ G_0[(-z)^{-1}]G_0(z) + G_1[(-z)^{-1}][-z^{-2K+1}G_0(-z^{-1})] &= 0 \\ G_0(-z^{-1})G_0(z) - z^{-2K+1}G_1(-z^{-1})G_0(-z^{-1}) &= 0 \\ G_0(-z^{-1})G_0(z) - z^{-2K+1}[-(-z^{-1})^{-2K+1}G_0(-[-z^{-1}]^{-1})]G_0(-z^{-1}) &= 0 \\ G_0(-z^{-1})G_0(z) - z^{-2K+1}[z^{2K-1}G_0(z)]G_0(-z^{-1}) &= 0 \\ G_0(-z^{-1})G_0(z) - G_0(z)G_0(-z^{-1}) &= 0. \end{aligned}$$

Similarly, beginning with Eq. (7.1-10),

$$\begin{aligned} H_0(z)G_0(z) + H_1(z)G_1(z) &= 2 \\ G_0(z^{-1})G_0(z) + G_1(z^{-1})G_1(z) &= 2 \\ G_0(z^{-1})G_0(z) + [-(-z^{-1})^{-2K+1}G_0(-[-z^{-1}]^{-1})][-z^{-2K+1}G_0(-z^{-1})] &= 2 \\ G_0(z^{-1})G_0(z) + (-z^{-2K+1})(-z^{-[-2K+1]})G_0(-z)G_0(-z^{-1}) &= 2 \\ G_0(z^{-1})G_0(z) + G_0(-z)G_0(-z^{-1}) &= 2 \end{aligned}$$

which is the design equation for the $G_0(z)$ prototype filter in row 3 of the table.

Problem 7.5

To be biorthogonal, QMF filters must satisfy matrix Eq. (7.1-13). Letting

$$\alpha = \frac{2}{\det[\mathbf{H}_m(z)]}$$

in that expression we can write

$$\begin{aligned} G_0(z) &= \alpha H_1(-z) \\ G_1(z) &= -\alpha H_0(-z) \end{aligned}$$

and see that the QMF filters in column 1 of Table 7.1 do satisfy it with $\alpha = 1$. Thus, QMF filters are biorthogonal. They are not orthonormal, however, since they do not satisfy the requirements of column 3 in Table 7.1. For QMF filters, for instance,

$$H_1(z) = H_0(-z) = -G_1(z)$$

but orthonormality (see column 3) requires that $H_1(z) = G_1(z^{-1})$.

Problem 7.6

Example 7.2 defines $h_0(n)$ for $n = 0, 1, 2, \ldots, 7$ to be about -0.01, 0.03, 0.03, -0.19, -0.03, 0.63, 0.72, 0.23. Using Eq. (7.1-23) with $2K = 8$, we can write

$$\begin{aligned} g_0(7-n) &= h_0(n) \\ g_1(n) &= (-1)^n g_0(7-n). \end{aligned}$$

Thus $g_0(n)$ is time-reversed $h_0(n)$, or 0.23, 0.72, 0.63, -0.03,-0.19, 0.03, 0.03, -0.01. In addtion, $g_1(n)$ is a time-reversed and modulated copy of $g_0(n)$; that is, -0.01, -0.03,0.03, 0.19, -0.03, -0.63, 0.72, -0.23.

To numerically prove the orthonormality of the filters, let $m = 0$ in Eq. (7.1-22):

$$\langle g_i(n) g_j(n) \rangle = \delta(i-j) \text{ with } i, j = \{0, 1\}.$$

Iterating over i and j we get

$$\begin{aligned}\sum_n g_0^2(n) &= \sum_n g_1^2(n) \\ &= 1 \\ \sum_n g_0(n)g_1(n) &= 0.\end{aligned}$$

Substitution of the filter coefficient values into these two equations yields:

$$\begin{aligned}\sum_n g_0(n)g_1(n) &= (0.23)(-0.01) + (0.72)(-0.03) + (0.63)(0.03) + \\ &\quad (-0.03)(0.19) + (-0.19)(-0.03) + (0.03)(-0.63) + \\ &\quad (0.03)(0.72) + (-0.01)(-0.23) \\ &= 0 \\ \sum_n g_0^2(n) &= \sum_n g_1^2(n) \\ &= (\pm 0.23)^2 + (0.72)^2 + (\pm .63)^2 + (-0.03)^2 + (\pm 0.19)^2 + \\ &\quad (0.03)^2 + (\pm 0.03)^2 + (-0.01)^2 \\ &= 1.\end{aligned}$$

Problem 7.7

Reconstruction is performed by reversing the decomposition process; that is, by replacing the downsamplers with upsamplers and the analysis filters by their synthesis filter counterparts, as shown in Fig. P7.7.

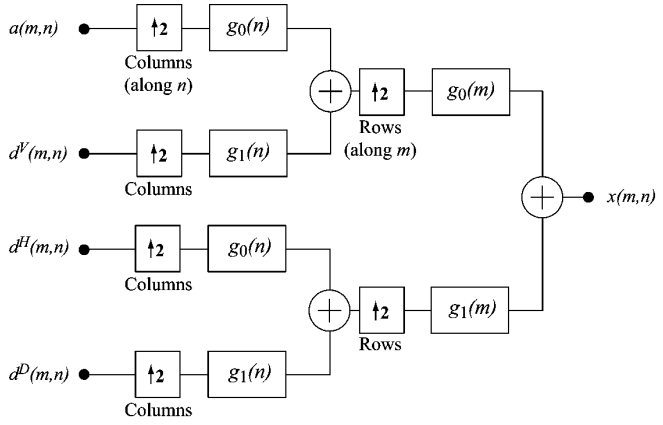


Figure P7.7

Problem 7.8

The Haar transform matrix for $N = 8$ is

$$\mathbf{H}_8 = \frac{1}{\sqrt{8}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{bmatrix}$$

Problem 7.9

(a) Equation (7.1-28) defines the 2×2 Haar transformation matrix as

$$H_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Then, using Eq. (7.1-24), we get

$$\begin{aligned} \mathbf{T} &= \mathbf{HFH} = \left(\frac{1}{\sqrt{2}}\right)^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 5 & 4 \\ -3 & 0 \end{bmatrix}. \end{aligned}$$

(b) First, compute

$$\mathbf{H}_2^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

so that

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Solving this matrix equation yields

$$\begin{aligned} \mathbf{H}_2^{-1} &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \mathbf{H}_2. \end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{F} &= \mathbf{H}^{-1}\mathbf{T}\mathbf{H}^{-1} \\ &= \left(\frac{1}{\sqrt{2}}\right)^2 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ -3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} 3 & -1 \\ 6 & 2 \end{bmatrix}.\end{aligned}$$

Problem 7.10

(a) The basis is orthonormal and the coefficients are computed by the vector equivalent of Eq. (7.2-5):

$$\begin{aligned}\alpha_0 &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= \frac{5\sqrt{2}}{2} \\ \alpha_1 &= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= \frac{\sqrt{2}}{2}\end{aligned}$$

so,

$$\begin{aligned}\frac{5\sqrt{2}}{2}\varphi_0 + \frac{\sqrt{2}}{2}\varphi_1 &= \frac{5\sqrt{2}}{2} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} + \frac{\sqrt{2}}{2} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \\ &= \begin{bmatrix} 3 \\ 2 \end{bmatrix}.\end{aligned}$$

(b) The basis is biorthonormal and the coefficients are computed by the vector equivalent of Eq. (7.2-3):

$$\begin{aligned}\alpha_0 &= \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= 1 \\ \alpha_1 &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= 2\end{aligned}$$

so,

$$\begin{aligned}\varphi_0 + 2\varphi_1 &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2\begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 3 \\ 2 \end{bmatrix}.\end{aligned}$$

(c) The basis is overcomplete and the coefficients are computed by the vector equivalent of Eq. (7.2-3):

$$\begin{aligned}\alpha_0 &= \begin{bmatrix} \frac{2}{3} & 0 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= 2 \\ \alpha_1 &= \begin{bmatrix} -\frac{1}{3} & \frac{\sqrt{3}}{3} \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= -1 + \frac{2\sqrt{3}}{3} \\ \alpha_2 &= \begin{bmatrix} -\frac{1}{3} & -\frac{\sqrt{3}}{3} \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} \\ &= -1 - \frac{2\sqrt{3}}{3}\end{aligned}$$

so,

$$\begin{aligned}2\varphi_0 + \left[-1 + \frac{2\sqrt{3}}{3}\right]\varphi_1 + \left[-1 - \frac{2\sqrt{3}}{3}\right]\varphi_2 &= 2\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \\ &\quad \left[-1 + \frac{2\sqrt{3}}{3}\right]\begin{bmatrix} -\frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} + \\ &\quad \left[-1 - \frac{2\sqrt{3}}{3}\right]\begin{bmatrix} -\frac{1}{2} \\ -\frac{\sqrt{3}}{2} \end{bmatrix} \\ &= \begin{bmatrix} 3 \\ 2 \end{bmatrix}.\end{aligned}$$

Problem 7.11

As can be seen in Fig. P7.11, scaling function $\varphi_{0,0}(x)$ cannot be written as a sum of double resolution copies of itself. Note the gap between $\varphi_{1,0}(x)$ and $\varphi_{1,1}(x)$.

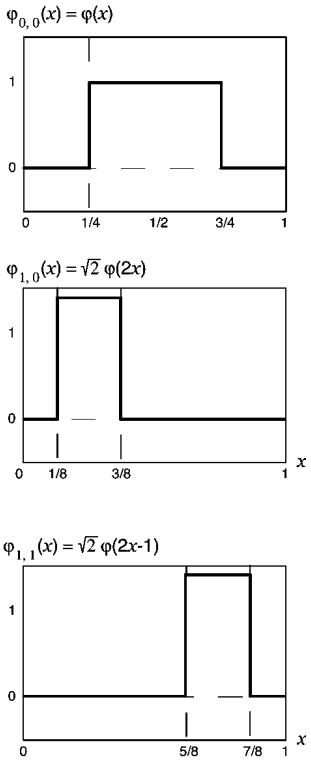


Figure P7.11

Problem 7.12

Substituting $j = 3$ into Eq. (7.2-13) we get

$$\begin{aligned} V_3 &= \overline{\underset{k}{Span}\{\varphi_{3,k}(x)\}} \\ &= \overline{\underset{k}{Span}\{2^{3/2}\varphi(2^3x-k)\}} \\ &= \overline{\underset{k}{Span}\{2\sqrt{2}\varphi(8x-k)\}}. \end{aligned}$$

Using the Haar scaling function in Eq. (7.2-14) we get the results shown in Fig. P7.12.

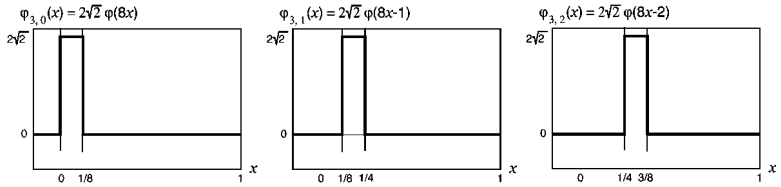


Figure P7.12

Problem 7.13

From Eq. (7.2-19) we find that

$$\begin{aligned}\psi_{3,3}(x) &= 2^{3/2}\psi(2^3x-3)\\ &= 2\sqrt{2}\psi(8x-3)\end{aligned}$$

and using the Haar wavelet function definition from Eq. (7.2-30), obtain the plot shown in Fig. P7.13.

To express $\psi_{3,3}(x)$ as a function of scaling functions, we employ Eq. (7.2-28) and the Haar wavelet vector defined in Example 7.6—that is, $h_\psi(0) = 1/\sqrt{2}$ and $h_\psi(1) = -1/\sqrt{2}$. Thus we get

$$\psi(x) = \sum_n h_\psi(n)\sqrt{2}\varphi(2x-n)$$

so that

$$\begin{aligned}\psi(8x-3) &= \sum_n h_\psi(n)\sqrt{2}\varphi(2[8x-3]-n)\\ &= \frac{1}{\sqrt{2}}\sqrt{2}\varphi(16x-6) + \left(\frac{-1}{\sqrt{2}}\right)\sqrt{2}\varphi(16x-7)\\ &= \varphi(16x-6) - \varphi(16x-7).\end{aligned}$$

Then, since $\psi_{3,3} = 2\sqrt{2}\psi(8x-3)$,

$$\begin{aligned}\psi_{3,3} &= 2\sqrt{2}\psi(8x-3)\\ &= 2\sqrt{2}\varphi(16x-6) - 2\sqrt{2}\varphi(16x-7).\end{aligned}$$

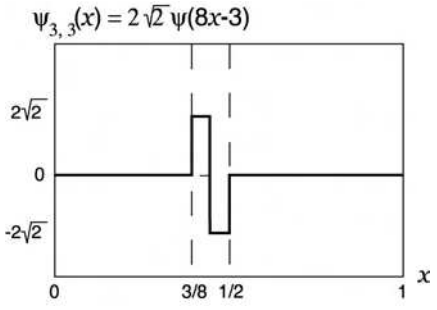


Figure P7.13

Problem 7.14

Using Eq. (7.2-22),

$$\begin{aligned} V_3 &= V_2 \oplus W_2 \\ &= V_1 \oplus W_1 \oplus W_2 \\ &= V_0 \oplus W_0 \oplus W_1 \oplus W_2. \end{aligned}$$

The scaling and wavelet functions are plotted in Fig. P7.14.

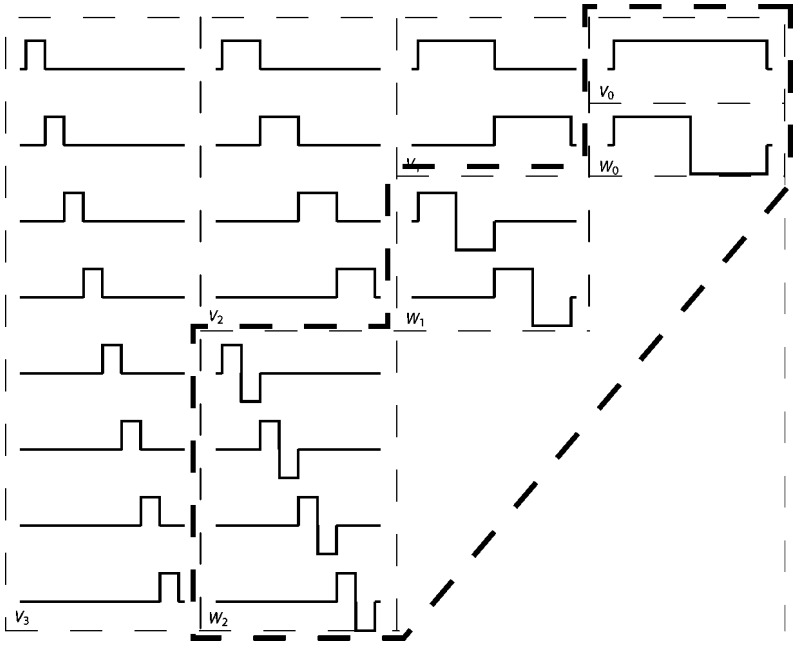


Figure P7.14

Problem 7.15

With $j_0 = 1$ the approximation coefficients are $c_1(0)$ and $c_1(1)$:

$$\begin{aligned} c_1(0) &= \int_0^{1/2} x^2\sqrt{2}dx = \frac{\sqrt{2}}{24} \\ c_1(1) &= \int_{1/2}^{1} x^2\sqrt{2}dx = \frac{7\sqrt{2}}{24}. \end{aligned}$$

Therefore, the V_1 approximation is

$$\frac{\sqrt{2}}{24}\varphi_{1,0}(x) + \frac{7\sqrt{2}}{24}\varphi_{1,1}(x),$$

which, when plotted, is identical to the V_1 approximation in Fig. 7.13(d). The last two coefficients are $d_1(0)$ and $d_1(1)$, which are computed as in the example. Thus, the

expansion is

$$y = \frac{\sqrt{2}}{24}\varphi_{1,0}(x) + \frac{7\sqrt{2}}{24}\varphi_{1,1}(x) + \left[\frac{-\sqrt{2}}{32}\psi_{1,0}(x) - \frac{3\sqrt{2}}{32}\psi_{1,1}(x)\right] + \cdots$$

Problem 7.16

(a) Since $M = 4$, $J = 2$, and $j_0 = 1$, the summations in Eqs. (7.3-5) through (7.3-7) are performed over $x = 0, 1, 2, 3$, $j = 1$, and $k = 0, 1$. Using Haar functions and assuming that they are distributed over the range of the input sequence, we get

$$\begin{aligned}
W_\varphi(1,0) &= \frac{1}{2}\left[f(0)\varphi_{1,0}(0) + f(1)\varphi_{1,0}(1) + f(2)\varphi_{1,0}(2) + f(3)\varphi_{1,0}(3)\right] \\
&= \frac{1}{2}\left[(1)(\sqrt{2}) + (4)(\sqrt{2}) + (-3)(0) + (0)(0)\right] = \frac{5\sqrt{2}}{2} \\
W_\varphi(1,1) &= \frac{1}{2}\left[f(0)\varphi_{1,1}(0) + f(1)\varphi_{1,1}(1) + f(2)\varphi_{1,1}(2) + f(3)\varphi_{1,1}(3)\right] \\
&= \frac{1}{2}\left[(1)(0) + (4)(0) + (-3)(\sqrt{2}) + (0)(\sqrt{2})\right] = \frac{-3\sqrt{2}}{2} \\
W_\psi(1,0) &= \frac{1}{2}\left[f(0)\psi_{1,0}(0) + f(1)\psi_{1,0}(1) + f(2)\psi_{1,0}(2) + f(3)\psi_{1,0}(3)\right] \\
&= \frac{1}{2}\left[(1)(\sqrt{2}) + (4)(-\sqrt{2}) + (-3)(0) + (0)(0)\right] = \frac{-3\sqrt{2}}{2} \\
W_\psi(1,1) &= \frac{1}{2}\left[f(0)\psi_{1,1}(0) + f(1)\psi_{1,1}(1) + f(2)\psi_{1,1}(2) + f(3)\psi_{1,1}(3)\right] \\
&= \frac{1}{2}\left[(1)(0) + (4)(0) + (-3)(\sqrt{2}) + (0)(-\sqrt{2})\right] = \frac{-3\sqrt{2}}{2}
\end{aligned}$$

so that the DWT is $\{5\sqrt{2}/2, -3\sqrt{2}/2, -3\sqrt{2}/2, -3\sqrt{2}/2\}$.

(b) Using Eq. (7.3-7),

$$\begin{aligned}
f(x) &= \frac{1}{2}[W_\varphi(1,0)\varphi_{1,0}(x) + W_\varphi(1,1)\varphi_{1,1}(x) + \\
&\quad W_\psi(1,0)\psi_{1,0}(x) + W_\psi(1,1)\psi_{1,1}(x)]
\end{aligned}$$

which, with $x = 1$, becomes

$$\begin{aligned}
f(1) &= \frac{\sqrt{2}}{4}\left[(5)(\sqrt{2}) + (-3)(0) + (-3)(\sqrt{2}) + (-3)(0)\right] \\
&= \frac{2(\sqrt{2})^2}{4} = 1.
\end{aligned}$$

Problem 7.17

Intuitively, the continuous wavelet transform (CWT) calculates a “resemblance index”

between the signal and the wavelet at various scales and translations. When the index is large, the resemblance is strong; else it is weak. Thus, if a function is similar to itself at different scales, the resemblance index will be similar at different scales. The CWT coefficient values (the index) will have a characteristic pattern. As a result, we can say that the function whose CWT is shown is self-similar—like a fractal signal.

Problem 7.18

(a) The scale and translation parameters are continuous, which leads to the overcompleteness of the transform.

(b) The DWT is a better choice when we need a space saving representation that is sufficient for reconstruction of the original function or image. The CWT is often easier to interpret because the built-in redundancy tends to reinforce traits of the function or image. For example, see the self-similarity of Problem 7.18.

Problem 7.19

The filter bank is the first bank in Fig. (7.17), as shown in Fig. P7.19:

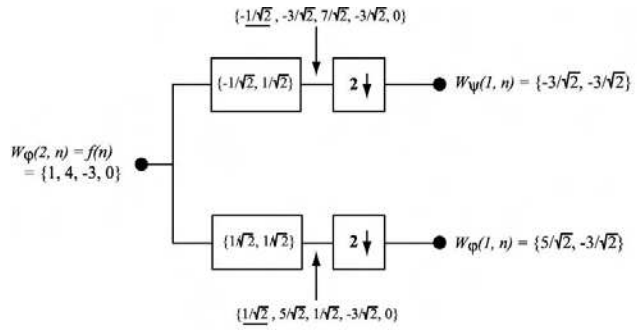


Figure P7.19

Problem 7.20

The complexity is determined by the number of coefficients in the scaling and wavelet vectors—that is, by n in Eqs. (7.2-18) and (72-28). This defines the number of taps in filters $h_\psi(-n)$, $h_\varphi(-n)$, $h_\psi(n)$, and $h_\varphi(n)$.

Problem 7.21

(a) Input $\varphi(n) = \{1, 1, 1, 1, 1, 1, 1, 1\} = \varphi_{0,0}(n)$ for a three-scale wavelet transform with Haar scaling and wavelet functions. Since wavelet transform coefficients measure the similarity of the input to the basis functions, the resulting transform is

$$\{W_\varphi(0,0), W_\psi(0,0), W_\psi(1,0), W_\psi(1,1), W_\psi(2,0), W_\psi(2,1), W_\psi(2,2)$$
$$W_\psi(2,3)\} = \{2\sqrt{2}, 0, 0, 0, 0, 0, 0, 0\}$$

The $W_\varphi(0,0)$ term can be computed using Eq. (7.3-5) with $j_0 = k = 0$.

(b) Using the same reasoning as in part (a), the transform is $\{0, 2\sqrt{2}, 0, 0, 0, 0, 0, 0\}$.

(c) For the given transform, $W_\psi(2,2) = B$ and all other transform coefficients are 0. Thus, the input must be proportional to $\psi_{2,2}(x)$. The input sequence must be of the form $\{0, 0, 0, 0, C, -C, 0, 0\}$ for some C. To determine C, use Eq. (7.3-6) to write

$$\begin{aligned} W_\psi(2,2) &= \frac{1}{\sqrt{8}}\{f(0)\psi_{2,2}(0) + f(1)\psi_{2,2}(1) + f(2)\psi_{2,2}(2) + f(3)\psi_{2,2}(3) + \\ &\quad f(4)\psi_{2,2}(4) + f(5)\psi_{2,2}(5) + f(6)\psi_{2,2}(6) + f(7)\psi_{2,2}(7)\} \\ &= \frac{1}{\sqrt{8}}\{(0)(0) + (0)(0) + (0)(0) + (0)(0) + (C)(2) + (-C)(-2) + \\ &\quad (0)(0) + (0)(0)\} \\ &= \frac{1}{\sqrt{8}}\{2C + 2C\} = \frac{4C}{\sqrt{8}} = \sqrt{2}C. \end{aligned}$$

Because this coefficient is known to have the value B, we have that $\sqrt{2}C = B$ or

$$C = \frac{\sqrt{2}}{2}B.$$

Thus, the input sequence is $\{0, 0, 0, 0, \sqrt{2}B/2, -\sqrt{2}B/2, 0, 0\}$. To check the result substitute these values into Eq. (7.3-6):

$$\begin{aligned} W_\psi(2,2) &= \frac{1}{\sqrt{8}}\{(0)(0) + (0)(0) + (0)(0) + (0)(0) + (\frac{\sqrt{2}}{2}B)(2) + \\ &\quad (-\frac{\sqrt{2}}{2}B)(-2) + (0)(0) + (0)(0)\} \\ &= \frac{1}{\sqrt{8}}\{\sqrt{2}B + \sqrt{2}B\} \\ &= B. \end{aligned}$$

Problem 7.22

They are both multi-resolution representations that employ a single reduced-resolution

approximation image and a series of "difference" images. For the FWT, these "difference" images are the transform detail coefficients; for the pyramid, they are the prediction residuals.

To construct the approximation pyramid that corresponds to the transform in Fig. 7.8(a), we will use the FWT^{-1} 2-d synthesis bank of Fig. 7.22(c). First, place the 64×64 approximation "coefficients" from Fig. 7.8(a) at the top of the pyramid being constructed. Then use it, along with 64×64 horizontal, vertical, and diagonal detail coefficients from the upper-left of Fig. 7.8(a), to drive the filter bank inputs in Fig. 7.22(c). The output will be a 128×128 approximation of the original image and should be used as the next level of the approximation pyramid. The 128×128 approximation is then used with the three 128×128 detail coefficient images in the upper 1/4 of the transform in Fig. 7.8(a) to drive the synthesis filter bank in Fig. 7.22(c) a second time—producing a 256×256 approximation that is placed as the next level of the approximation pyramid. This process is then repeated a third time to recover the 512×512 original image, which is placed at the bottom of the approximation pyramid. Thus, the approximation pyramid would have 4 levels.

Problem 7.23

One pass through the FWT 2-d filter bank of Fig. 7.22(a) is all that is required (see Fig. P7.23):

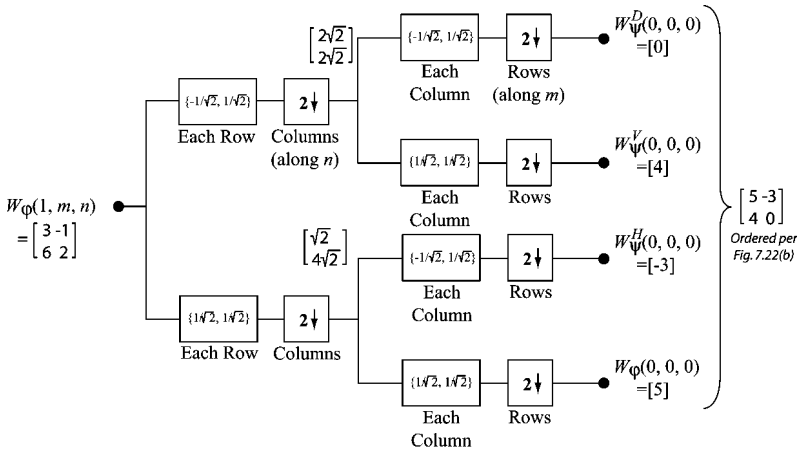


Figure P7.23

Problem 7.24

As can be seen in the sequence of images that are shown, the DWT is not shift invariant. If the input is shifted, the transform changes. Since all original images in the problem are 128×128, they become the $W_{\varphi}(7, m, n)$ inputs for the FWT computation process. The filter bank of Fig. 7.22(a) can be used with $j + 1 = 7$. For a single scale transform, transform coefficients $W_{\varphi}(6, m, n)$ and $W_{\psi}^{i}(6, m, n)$ for $i = H, V, D$ are generated. With Haar wavelets, the transformation process subdivides the image into non-overlapping 2×2 blocks and computes 2-point averages and differences (per the scaling and wavelet vectors). Thus, there are no horizontal, vertical, or diagonal detail coefficients in the first two transforms shown; the input images are constant in all 2×2 blocks (so all differences are 0). If the original image is shifted by 1 pixel, detail coefficients are generated since there are then 2×2 areas that are not constant. This is the case in the third transform shown.

Problem 7.25

The table is completed as shown in Fig. P7.25.

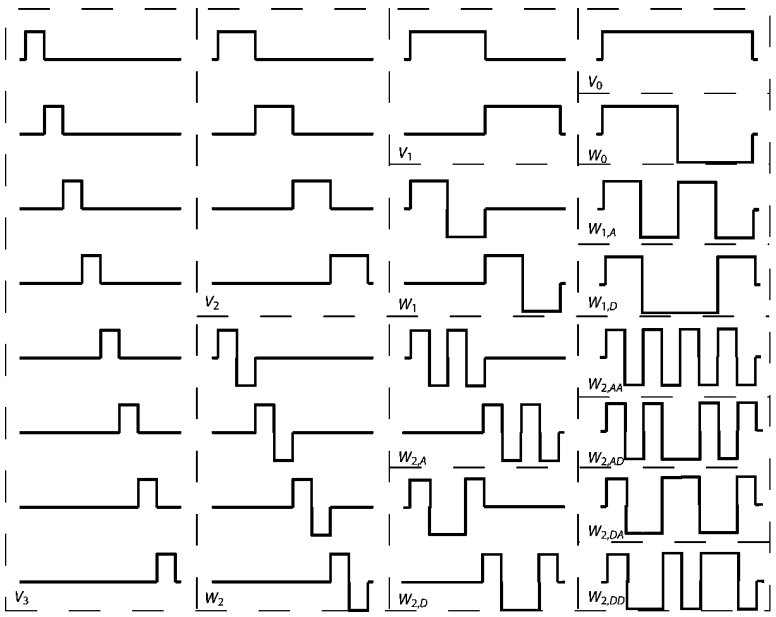


Figure P7.25

The functions are determined using Eqs. (7.2-18) and (7.2-28) with the Haar scaling and

wavelet vectors from Examples 7.5 and 7.6:

$$\begin{aligned}\varphi(x) &= \varphi(2x) + \varphi(2x-1)\\ \psi(x) &= \varphi(2x) - \varphi(2x-1).\end{aligned}$$

Problem 7.26

(a) The analysis tree is shown in Fig. P7.26(a):

(b) The corresponding frequency spectrum is shown in Fig. P7.26(b):

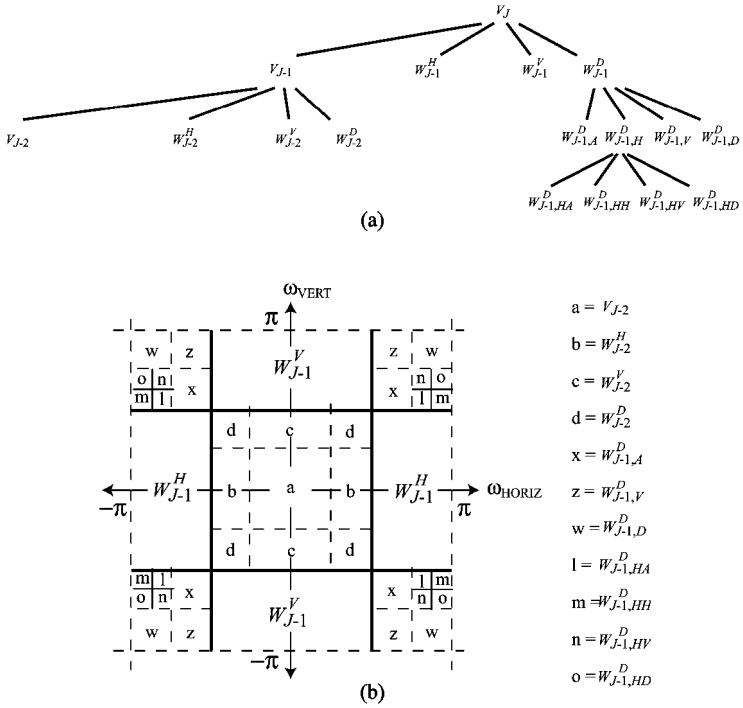


Figure P7.26

Problem 7.27

First use the entropy measure to find the starting value for the input sequence, which is

$$E\{f(n)\} = \sum_{n=0}^{7} f^2(n) \ln\left[f^2(n)\right] = 2.7726.$$

Then perform an iteration of the FWT and compute the entropy of the generated approximation and detail coefficients. They are 2.0794 and 0, respectively. Since their sum is less than the starting entropy of 2.7726, we will use the decomposition.

Because the detail entropy is 0, no further decomposition of the detail is warranted. Thus, we perform another FWT iteration on the approximation to see if it should be decomposed again. This process is then repeated until no further decompositions are called for. The resulting optimal tree is shown in Fig. P7.27:

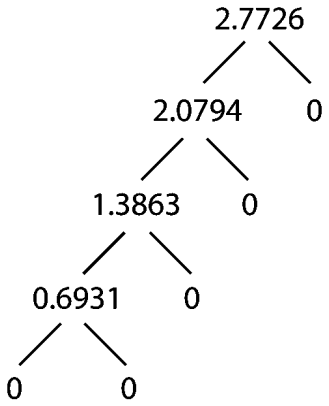


Figure P7.27

8 Problem Solutions

Problem 8.1

(a) A histogram equalized image (in theory) has a gray level distribution which is uniform. That is, all gray levels are equally probable. Eq. (8.1-4) thus becomes

$$L_{avg} = \frac{1}{2^n} \sum_{k=0}^{2^n-1} (r_k)$$

where $1/2n$ is the probability of occurrence of any gray level. Since all levels are equally probable, there is no advantage to assigning any particular gray level fewer bits than any other. Thus, we assign each the fewest possible bits required to cover the $2n$ levels. This, of course is n bits and L_{avg} becomes n bits also:

$$\begin{aligned} L_{avg} &= \frac{1}{2^n} \sum_{k=0}^{2^n-1} (n) \\ &= \frac{1}{2^n} (2^n)\, n \\ &= n. \end{aligned}$$

(b) Since interpixel redundancy is associated with the spatial arrangement of the gray levels in the image, it is possible for a histogram equalized image to contain a high level of interpixel redundancy - or none at all.

Problem 8.2

(a) A single line of raw data contains $n_1 = 2^n$ bits. The maximum run length would be 2^n and thus require n bits for representation. The starting coordinate of each run also requires n bits since it may be arbitrarily located within the 2^n pixel line. Since a run length of 0 can not occur and the run-length pair $(0, 0)$ is used to signal the start of each new line - an additional $2n$ bits are required per line. Thus, the total number of bits

required to code any scan line is

$$\begin{aligned} n_2 &= 2n + N_{avg}(n+n) \\ &= 2n(1+N_{avg}) \end{aligned}$$

where N_{avg} is the average number of run-length pairs on a line. To achieve some level of compression, C_R must be greater than 1. So,

$$\begin{aligned} C_R &= \frac{n_1}{n_2} \\ &= \frac{2^n}{2n(1+N_{avg})} > 1 \end{aligned}$$

and

$$N_{avg} < \frac{2^{n-1}}{n} - 1.$$

(b) For $n = 10$, N_{avg} must be less than 50.2 run-length pairs per line.

Problem 8.3

Table P8.3 shows the data, its 6-bit code, the IGS sum for each step, the actual IGS 3-bit code and its equivalent decoded value, the error between the decoded IGS value and the input values, and the squared error.

Table P8.3

Data	6-bit Code	Sum	IGS Code	Decoded IGS	Error	Sq. Error
		000000				
12	001100	001100	001	8	4	16
12	001100	010000	010	16	-4	16
13	001101	001101	001	8	5	25
13	001101	010010	010	16	-3	9
10	001010	001100	001	8	2	4
13	001101	010001	010	16	-3	9
57	111001	111001	111	56	1	1
54	110110	110111	110	48	6	36

Problem 8.4

The average square error is the sum of the last column of the table in Problem 8.3 divided by 8, the number of data points. This computation yields 116/8 or 14.5. The rms error is then 3.81, the square root of 14.5. The squared signal value (i.e., 6400) is obtained by

summing the squares of column 5 of the table. The rms signal-to-noise ratio is then

$$SNR_{rms} = \sqrt[2]{\frac{6400}{116}} = 7.43$$

Problem 8.5

(a) For the first value of the table (i.e., 0110), substitution into Eq. (8.2-1) gives:

$$\begin{aligned}
h_1 &= b_3 \oplus b_2 \oplus b_0 = 0 \oplus 1 \oplus 0 = 1 \\
h_2 &= b_3 \oplus b_1 \oplus b_0 = 0 \oplus 1 \oplus 0 = 1 \\
h_3 &= b_3 = 0 \\
h_4 &= b_2 \oplus b_1 \oplus b_0 = 1 \oplus 1 \oplus 0 = 0 \\
h_5 &= b_2 = 1 \\
h_6 &= b_1 = 1 \\
h_7 &= b_0 = 0.
\end{aligned}$$

Thus, the encoded value is 1100110. The remaining values of Table 8.2 are treated similarly. The resulting code words are 0011001, 1110000, and 1111111, respectively.

(b) For 1100111, construct the following three bit odd parity word:

$$\begin{aligned}
c_1 &= h_1 \oplus h_3 \oplus h_5 \oplus h_7 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \\
c_2 &= h_2 \oplus h_3 \oplus h_6 \oplus h_7 = 1 \oplus 0 \oplus 1 \oplus 1 = 1 \\
c_4 &= h_4 \oplus h_5 \oplus h_6 \oplus h_7 = 0 \oplus 1 \oplus 1 \oplus 1 = 1.
\end{aligned}$$

A parity word of 111_2 indicates that bit 7 is in error. The correctly decoded binary value is 0110_2. In a similar manner, the parity words for 1100110 and 1100010 are 000 and 101, respectively. The decoded values are identical and are 0110.

Problem 8.6

The conversion factors are computed using the logarithmic relationship

$$\log_a x = \frac{1}{\log_b a} \log_b x.$$

Thus, 1 Hartley = 3.3219 bits and 1 nat = 1.4427 bits.

Problem 8.7

Let the set of source symbols be $\{a_1, a_2, ..., a_q\}$ with probabilities

$$\mathbf{z} = [P(a_1), P(a_2), ..., P(a_q)]^T.$$

Then, using Eq. (8.3-3) and the fact that the sum of all $P(a_i)$ is 1, we get

$$\begin{aligned}\log q - H(\mathbf{z}) &= \sum_{i=1}^{q} P(a_i) \log q + \sum_{i=1}^{q} P(a_i) \log P(a_i) \\ &= \sum_{i=1}^{q} P(a_i) \log qP(a_i).\end{aligned}$$

Using the log relationship from Problem 8.6, this becomes

$$= \log e \sum_{i=1}^{q} P(a_i) \ln qP(a_i).$$

Then, multiplying the inequality $\ln x \leq x - 1$ by -1 to get $\ln 1/x \geq 1 - x$ and applying it to this last result,

$$\begin{aligned}\log q - H(\mathbf{z}) &\geq \log e \sum_{i=1}^{q} P(a_i) \left[1 - \frac{1}{qP(a_i)}\right] \\ &\geq \log e \left[\sum_{i=1}^{q} P(a_i) - \frac{1}{q} \sum_{i=1}^{q} \frac{P(a_i)}{P(a_i)}\right] \\ &\geq \log e\,[1 - 1] \\ &\geq 0\end{aligned}$$

so that

$$\log q \geq H(\mathbf{z}).$$

Therefore, $H(\mathbf{z})$ is always less than, or equal to, $\log q$. Furthermore, in view of the equality condition $(x = 1)$ for $\ln 1/x \geq 1 - x$, which was introduced at only one point in the above derivation, we will have strict equality if and only if $P(a_i) = 1/q$ for all i.

Problem 8.8

The source symbol probabilities are taken directly from $\mathbf{z}$ and are $P(a = 0) = 0.75$ and $P(a = 1) = 0.25$. Likewise, the elements of $\mathbf{Q}$ are the forward transition probabilities $P(b = 0|a = 0) = 2/3$, $P(b = 0|a = 1) = 1/10$, $P(b = 1|a = 0) = 1/3$, and $P(b = 1|a = 1) = 9/10$. The matrix multiplication of Eq. (8.3-6) yields the output probabilities

$$v = Qz = \begin{bmatrix} \frac{2}{3} & \frac{1}{10} \\ \frac{1}{3} & \frac{9}{10} \end{bmatrix} \begin{bmatrix} \frac{3}{4} \\ \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{21}{40} \\ \frac{19}{40} \end{bmatrix}.$$

Thus, $P(b = 0) = 21/40$ and $P(b = 1) = 19/40$. The conditional input probabilities are computed using Bayes' formula

$$P(a_j|b_k) = \frac{P(b_k|a_j)P(a_j)}{P(b_k)}.$$

Thus, $P(a = 0|b = 0) = 20/21$, $P(a = 0|b = 1) = 10/19$, $P(a = 1|b = 0) = 1/21$,

and $P(a=1|b=1)=9/19$. Finally, the joint probabilities are computed using

$$P(a_j,b_k)=P\left(a_j\right)P(b_k|a_j)$$

which yields $P(a=0,b=0)=1/2$, $P(a=0,b=1)=1/4$, $P(a=1,b=0)=1/40$, and $P(a=1,b=1)=9/40$.

Problem 8.9

(a) Substituting the given values of p_{bs} and p_e into the binary entropy function derived in the example, the average information or entropy of the source is 0.811 bits/symbol.

(b) The equivocation or average entropy of the source given that the output has been observed (using Eq. 8.3-9) is 0.75 bits/symbol. Thus, the decrease in uncertainty is 0.061 bits/symbol.

(c) It is the mutual information $I(\mathbf{z},\mathbf{v})$ of the system and is less than the capacity of the channel, which is, in accordance with the equation derived in the example, 0.0817 bits/symbol.

Problem 8.10

(a) The proof proceeds by substituting the elements of $\mathbf{Q}$ into Eq. (8.3-13) and simplifying. The source probabilities are left as variables during the simplification.

$$\begin{aligned}
C &= \max_{\mathbf{z}}\left[I\left[\mathbf{z},\mathbf{v}\right]\right] \\
&= \max_{\mathbf{z}}\sum_{j=1}^{J}\sum_{k=1}^{K}P\left(a_j\right)q_{kj}\log\frac{q_{kj}}{\sum_{i=1}^{J}P(a_j)qki} \\
&= \max_{\mathbf{z}}\left[\sum_{k=1}^{3}P\left(a_1\right)q_{k1}\log\frac{q_{k1}}{\sum_{i=1}^{2}P(a_j)qki}\right. \\
&\qquad \left.+\sum_{k=1}^{3}P\left(a2\right)q_{k2}\log\frac{q_{k2}}{\sum_{i=1}^{2}P(a_j)qki}\right] \\
&= \max_{\mathbf{z}}\left[P\left(a_1\right)\left((1-\beta)\log\frac{1-\beta}{P(a_1)(1-\beta)}+\beta\log\frac{\beta}{P(a_1)(1-\beta)}+0\right)\right. \\
&\qquad \left.+P\left(a_2\right)\left(0+\beta\log\frac{\beta}{P(a_2)(1-\beta)}+(1-\beta)\log\frac{1-\beta}{P(a_2)(1-\beta)}\right)\right] \\
&= \max_{\mathbf{z}}\left[P\left(a_1\right)\left((1-\beta)\log\frac{1}{P(a_1)}+\beta\log\frac{1}{2P(a_1)}\right)\right. \\
&\qquad \left.+P\left(a_2\right)\left(\beta\log\frac{1}{2P(a_2)}+(1-\beta)\log\frac{1}{P(a_2)}\right)\right] \\
&= \max_{\mathbf{z}}\left[-P\left(a_1\right)\left((1-\beta)\log P\left(a_1\right)+\beta\log 2P\left(a_1\right)\right)\right. \\
&\qquad \left.-P\left(a_2\right)\left(\beta\log 2P\left(a_2\right)+(1-\beta)\log P\left(a_2\right)\right)\right]
\end{aligned}$$

$$\begin{aligned} &= \max_{\mathbf{z}} [-P(a_1)((1-\beta)\log P(a_1) + \beta\log 2 + \beta\log P(a_1)) \\ &\quad - P(a_2)(\beta\log 2 + \beta\log P(a_2) + (1-\beta)\log P(a_2))] \\ &= \max_{\mathbf{z}} [-P(a_1)(\log P(a_1) + \beta\log 2) - P(a_2)(\beta\log 2 + \log P(a_2))] \\ &= \max_{\mathbf{z}} [-P(a_1)\log P(a_1) - P(a_2)\log P(a_2) - P(a_1)\beta\log 2) \\ &\quad - P(a_2)\beta\log 2]. \end{aligned}$$

Noting that the first two terms of this sum are the entropy of the source and factoring out the common factor in the last two terms, we get

$$C = \max_{\mathbf{z}} [H(\mathbf{z}) - (P(a_1) + P(a_2))\beta\log 2].$$

Since the sum of the source probabilities is 1 and the maximum entropy of a binary source is also 1 with both symbols equally probable, this reduces to

$$C = 1 - \beta.$$

(b) Substituting 0.5 into the above equation, the capacity of the erasure channel is 0.5. Substituting 0.125 into the equation for the capacity of a BSC given in Section 8.3.2, we find that its capacity is 0.456. Thus, the binary erasure channel with a higher probability of error has a larger capacity to transfer information.

Problem 8.11

(a) The plot is shown in Fig. P8.11.

(b) $D_{\max}$ is σ^2.

(c) If we wish to code the source in this example so that the maximum average encoding-decoding distortion D is $0.75\sigma^2$, we first evaluate $R(D)$ for $D = 0.75\ \sigma^2$. Since $R(0.75\sigma^2) = 0.21$, we know that at least 0.21 code bits per source symbol must be used to achieve the fidelity objective. Thus, this is the maximum possible information compression under this criterion.

Problem 8.12

(a) There are two unique codes.

(b) The codes are: (1) 0, 11, 10 and (2) 1, 00, 01. The codes are complements of one another. They are constructed by following the Huffman procedure for three symbols of arbitrary probability.

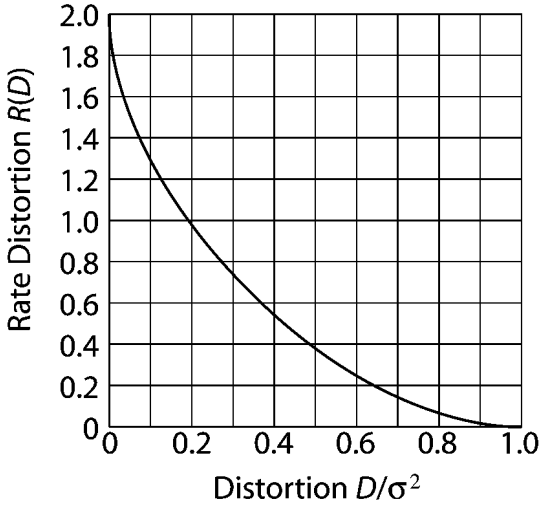


Figure P8.11

Problem 8.13

(a) The entropy is computed using Eq. (8.3-3) and is 2.6508 bits/symbol.

(b) The specific binary codes assigned to each gray level may vary depending upon the arbitrary selection of 1s and 0s assigned at each step of the coding algorithm. The number of bits used for each gray level, however, should be the same for all versions constructed. The construction of Code 2 in Table 8.1 proceeds as follows:

Step 1: Arrange according to symbol probabilities from left to right, as shown in Fig. P8.13(a).

Step 2: Assign code words based on the ordered probabilities from right to left, as shown in Fig. P8.13(b).

Step 3: The codes associated with each gray level are read at the left of the diagram.

(c) - (f) The remaining codes and their average lengths, which are computed using (8.1-4), are shown in Table P8.13. Note that two Huffman shift codes are listed, one of which is the best. In generating these codes, the sum of probabilities 4 - 7 were used as the probability of the shift up symbol. The sum is 0.19, which is equivalent to the probability of symbol r_0. Thus, the two codes shown differ by the ordering of r_0 and the shift symbol during the Huffman coding process.

Table P8.13

r_k	$p_r(r_k)$	B_1-code	2-bit Shift	H. Shift 1	H. Shift 2	Huffman
$r_0 = 0$	0.19	$C0C0$	10	11	000	11
$r_1 = 1/7$	0.25	$C0$	00	01	01	01
$r_2 = 2/7$	0.21	$C1$	01	10	10	10
$r_3 = 3/7$	0.16	$C0C1$	1100	001	001	001
$r_4 = 4/7$	0.08	$C1C0$	1101	00001	1101	0001
$r_5 = 5/7$	0.06	$C1C1$	1110	00010	1110	00001
$r_6 = 6/7$	0.03	$C0C0C0$	111100	00011	11000	000001
$r_7 = 1$	0.02	$C0C0C1$	111101	000001	11001	000000
Length		3.18	2.8	2.75	2.78	2.7

The entropy of the source is $H = 2.65$ from Eq. (8.3-3) and the probabilities from column 2.

Problem 8.14

The arithmetic decoding process is the reverse of the encoding procedure. Start by dividing the [0, 1) interval according to the symbol probabilities. This is shown in Table P8.14. The decoder immediately knows the message 0.23355 begins with an "e", since the coded message lies in the interval [0.2, 0.5). This makes it clear that the second symbol is an "a", which narrows the interval to [0.2, 0.26). To further see this, divide the interval [0.2, 0.5) according to the symbol probabilities. Proceeding like this, which is the same procedure used to code the message, we get "eaii!".

Table P8.14

Symbol	Probability	Range
a	0.2	[0.0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1.0)

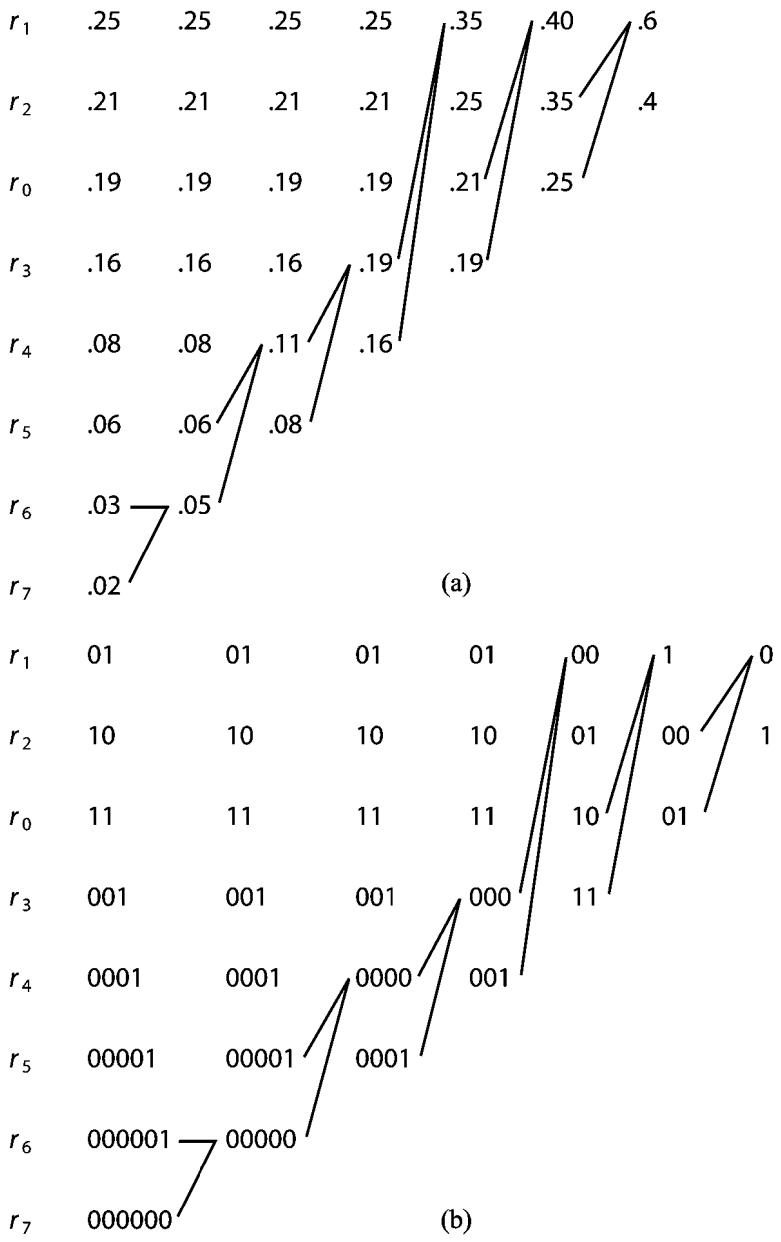

r1 .25 .25 .25 .25 .35 .40 .6
r2 .21 .21 .21 .21 .25 .35 .4
r0 .19 .19 .19 .19 .21 .25
r3 .16 .16 .16 .19 .19
r4 .08 .08 .11 .16
r5 .06 .06 .08
r6 .03 .05
r7 .02
(a)
r1 01 01 01 01 00 1 0
r2 10 10 10 10 01 00 1
r0 11 11 11 11 10 01
r3 001 001 001 000 11
r4 0001 0001 0000 001
r5 00001 00001 0001
r6 000001 00000
r7 000000
(b)

Figure P8.13

Problem 8.15

Assume that the first 256 codes in the starting dictionary are the ASCII codes. If you assume 7-bit ASCII, the first 128 locations are all that are needed. In either case, the ASCII "a" corresponds to location 97. The coding proceeds as shown in Table P8.15.

Table P8.15

Recognized	Character	Output	Dict. Address	Dict. Entry
	a			
a	a	97	256	aa
a	a			
aa	a	256	257	aaa
a	a			
aa	a			
aaa	a	257	258	aaaa
a	a			
aa	a			
aaa	a			
aaaa	a	258	259	aaaaa
a		97		

Problem 8.16

The input to the LZW decoding algorithm for the example in Example 8.12 is

39 39 126 126 256 258 260 259 257 126

The starting dictionary, to be consistent with the coding itself, contains 512 locations– with the first 256 corresponding to gray level values 0 through 255. The decoding algorithm begins by getting the first encoded value, outputting the corresponding value from the dictionary, and setting the "recognized sequence" to the first value. For each additional encoded value, we (1) output the dictionary entry for the pixel value(s), (2) add a new dictionary entry whose content is the "recognized sequence" plus the first element of the encoded value being processed, and (3) set the "recognized sequence" to the encoded value being processed. For the encoded output in Example 8.12, the sequence of operations is as shown in Table P8.16.

Note, for example, in row 5 of the table that the new dictionary entry for location 259 is 126-39, the concatenation of the currently recognized sequence, 126, and the first

element of the encoded value being processed–the 39 from the 39-39 entry in dictionary location 256. The output is then read from the third column of the table to yield

$$\begin{matrix} 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \\ 39 & 39 & 126 & 126 \end{matrix}$$

where it is assumed that the decoder knows or is given the size of the image that was recieved. Note that the dictionary is generated as the decoding is carried out.

Table P8.16

Recognized	Encoded Value	Pixels	Dict. Address	Dict. Entry
	39	39		
39	39	39	256	39-39
39	126	126	257	39-126
126	126	126	258	126-126
126	256	39-39	259	126-39
256	258	126-126	260	39-39-126
258	260	39-39-126	261	126-126-39
260	259	126-39	262	39-39-126-126
259	257	39-126	263	126-39-39
257	126	126	264	39-126-126

Problem 8.17

(a) Using Eq. (8.4-3), form Table P8.17.

Table P8.17

Binary	Gray Code		Binary	Gray Code
0000	0000		1000	1100
0001	0001		1001	1101
0010	0011		1010	1111
0011	0010		1011	1110
0100	0110		1100	1010
0101	0111		1101	1011
0110	0101		1110	1001
0111	0100		1111	1000

(b) The procedure is to work from the most significant bit to the least significant bit using the equations:

$$a_{m-1} = g_{m-1}$$
$$a_i = g_i \oplus a_{i+1} \quad 0 \leq i \leq m-2.$$

The decoded binary value is thus 0101100111010.

Problem 8.18

(a) Using the procedure described in Section 8.4.3, the decoded line is

$$[W\ 1001\ W\ W\ W\ W\ W\ W\ 0000\ 0010\ W\ W\ W\ W\ W\ W]$$

where W denotes four white pixels - i.e., 1111.

(b) - (c) Establish the convention that sub-blocks are included in the code string from left to right. Then, using brackets to clarify the decomposition steps, we get

1 [[W 1001 W W W W W W] [0000 0010 W W W W W W]]
1 [1 [W 1001 W W] [W W W W]] [1 [0000 0010 W W] [W W W W]]]
1 [1 [1 [1 [[W 1001] [W W]] [0]] [1 [1 [[0000 0010] [W W]] [0]]]
1 [1 [1 [1 [1 [W] [1001]] [0]] [0]] [1 [1 [1 [1 [0000] [0010]] [0]] [0]]]
1 [1 [1 [1 [1 [0] [11001]] [0]] [0]] [1 [1 [1 [1 [10000] [10010]] [0]] [0]]]

Thus, the encoded string is 1111011001001111000010010 00, which requires 27 bits. The first encoding required 28 bits.

Problem 8.19

(a) The motivation is clear from Fig. 8.17. The transition at c must somehow be tied to a particular transition on the previous line. Note that there is a closer white to black transition on the previous line to the right of c, but how would the decoder know to use it instead of the one to the left. Both are less than ec. The first similar transition past e establishes the convention to make this decision.

(b) An alternate solution would be to include a special code which skips transitions on the previous line until you get to the closest one.

Problem 8.20

(a) Substituting $\rho_h = 0$ into Eq. (8.5-12) and evaluating it to form the elements of $\mathbf{R}$

and $\mathbf{r}$, we get

$$\mathbf{R} = \sigma^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \text{ and } r = \sigma^2 \begin{bmatrix} \rho \\ \rho^2 \end{bmatrix}.$$

(b) First form the inverse of $\mathbf{R}$,

$$\mathbf{R}^{-1} = \frac{1}{\sigma^2 (1-\rho^2)} \begin{bmatrix} 1 & -\rho \\ -\rho & 1 \end{bmatrix}.$$

Then, perform the matrix multiplication of Eq. (8.5-8):

$$\alpha = \mathbf{R}^{-1}\mathbf{r} = \frac{\sigma^2}{\sigma^2 (1-\rho^2)} \begin{bmatrix} \rho \left(1-\rho^2\right) \\ 0 \end{bmatrix} = \begin{bmatrix} \rho \\ 0 \end{bmatrix}.$$

Thus, $\alpha_1 = \rho$ and $\alpha_2 = 0$.

(c) The variance is computed using Eq. (8.5-11):

$$\sigma_e^2 = \sigma^2 - \boldsymbol{\alpha}^T \mathbf{r} = \begin{bmatrix} \rho & 0 \end{bmatrix} \begin{bmatrix} \rho \\ \rho^2 \end{bmatrix} = \sigma^2 \left(1-\rho^2\right).$$

Problem 8.21

The derivation proceeds by substituting the uniform probability function into Eqs. (8.5-20) - (8.5-22) and solving the resulting simultaneous equations with $L = 4$. Eq. (8.5-21) yields

$$\begin{aligned} s_0 &= 0 \\ s_1 &= \tfrac{1}{2}(t_1 + t_2) \\ s_2 &= \infty. \end{aligned}$$

Substituting these values into the integrals defined by Eq. (8.5-20), we get two equations. The first is (assuming $s_1 \leq A$)

$$\int_{s_0}^{s_1} (s - t_1)\, p(s)\, ds = 0$$

$$\frac{1}{2A} \int_0^{\frac{1}{2}(t_1+t_2)} (s - t_1)\, ds = \frac{s^2}{2} - t_1 s \Bigg|_0^{\frac{1}{2}(t_1+t_2)} = 0$$

$$(t_1 + t_2)^2 - 4t_1 (t_1 + t_2) = 0$$

$$(t_1 + t_2)(t_2 - 3t_1) = 0$$

$$t_1 = -t_2 \text{ and } t_2 = 3t_1.$$

The first of these relations does not make sense since both t_1 and t_2 must be positive. The second relationship is a valid one. The second integral yields (noting that s_1 is less than A so the integral from A to ∞ is 0 by the definition of $p(s)$)

$$\frac{1}{2A} \int_{\frac{1}{2}(t_1+t_2)}^{A} (s - t_2)\, ds = \frac{s^2}{2} - t_2 s \Bigg|_{\frac{1}{2}(t_1+t_2)}^{A} = 0$$

$$4A^2 - 8At_2 - (t_1 + t_2)^2 - 4t_2(t_1 + t_2) = 0.$$

Substituting $t_2 = 3t_1$ from the first integral simplification into this result, we get

$$8t_1^2 - 6At_1 + A^2 = 0$$
$$\left[t_1 - \tfrac{A}{2}\right](8t_1 - 2A) = 0$$
$$t_1 = \tfrac{A}{2} \text{ and } t_1 = \tfrac{A}{4}.$$

Back substituting these values of t_1, we find the corresponding t_2 and s_1 values:

$$t_2 = \tfrac{3A}{2} \text{ and } s_1 = A \text{ for } t_1 = \tfrac{A}{2}$$
$$t_2 = \tfrac{3A}{4} \text{ and } s_1 = \tfrac{A}{2} \text{ for } t_1 = \tfrac{A}{4}.$$

Since $s_1 = A$ is not a real solution (the second integral equation would then be evaluated from A to A, yielding 0 or no equation), the solution is given by the second. That is,

$$s_0 = 0 \quad s_1 = \tfrac{A}{2} \quad s_2 = \infty$$
$$t_1 = \tfrac{A}{4} \quad t_2 = \tfrac{3A}{4}$$

Problem 8.22

Following the procedure in the flow chart of Fig. 8.37, the proper code is

0001 010 1 0011000011 0001

where the spaces have been inserted for readability alone. The coding mode sequence is pass, vertical (1 left), vertical (directly below), horizontal (distances 3 and 4), and pass.

Problem 8.23

(a) - (b) Following the procedure outlined in Section 8.6.2, we obtain the results shown in Table P8.23.

Problem 8.24

Since the T1 transfer rate is 1.544 Mbit/sec, a 6 second transfer will provide

$$(1.544 \times 10^6)(6 \text{ sec}) = 9.264 \times 10^6 \text{ bits}$$

of data. The initial approximation of the X-ray must contain no more than this number of bits. The required compression ratio is thus

$$C_R = \frac{4096 \times 4096 \times 12}{9.264 \times 10^6} = 21.73$$

The JPEG transform coding approach of Section 6.6 can achieve this level of compression and provide reasonably good reconstructions. At the X-ray encoder, the X-ray can be JPEG compressed using a normalization array that yields about a 25:1 compression.

While it is being transmitted over the T1 line to the remote viewing station, the encoder can decode the compressed JPEG data and identify the "differences" between the resulting X-ray approximation and the original X-ray image. Since we wish to transmit these "differences" over a span of 1 minute with refinements every 5 - 6 seconds, there can be no more than

$$\frac{60 \text{ sec}}{6} \text{ to } \frac{60 \text{ sec}}{5} = 10 \text{ to } 12 \text{ refinements.}$$

If we assume that 12 refinements are made and that each refinement corresponds to the "differences" between one of the 12 bits in the original X-ray and the JPEG reconstructed approximation, then the compression that must be obtained per bit (to allow a 6 second average transfer time for each bit) is

$$C_R = \frac{4096 \times 4096 \times 1}{9.264 \times 10^6} = 1.81$$

where, as before, the bottom of the fraction is the number of bits that can be transmitted over a T1 line in 6 seconds. Thus, the "difference" data for each bit must be compressed by a factor just less than 2. One simple way to generate the "difference information" is to XOR the actual X-ray with the reconstructed JPEG approximation. The resulting binary image will contain a 1 in every bit position at which the approximation differs from the original. If the XOR result is transmitted one bit at a time beginning with the MSB and ending with the LSB, and each bit is compressed by an average factor of 1.81:1, we will achieve the performance that is required in the problem statement. To achieve an average error-free bit-plane compression of 1.81:1 (see Section 6.4), the XOR data can be Gray coded, run-length coded, and finally variable-length coded. A conceptual block diagram for both the encoder and decoder are given below. Note that the decoder computes the bit refinements by XORing the decoded XOR data with the reconstructed JPEG approximation.

Table P8.23

DC Coefficient Difference	Two's Complement Value	Code
-7	1...1001	00000
-6	1...1010	00001
-5	1...1011	00010
-4	1...1100	00011
4	0...0100	00100
5	0...0101	00101
6	0...0110	00110
7	0...0111	00111

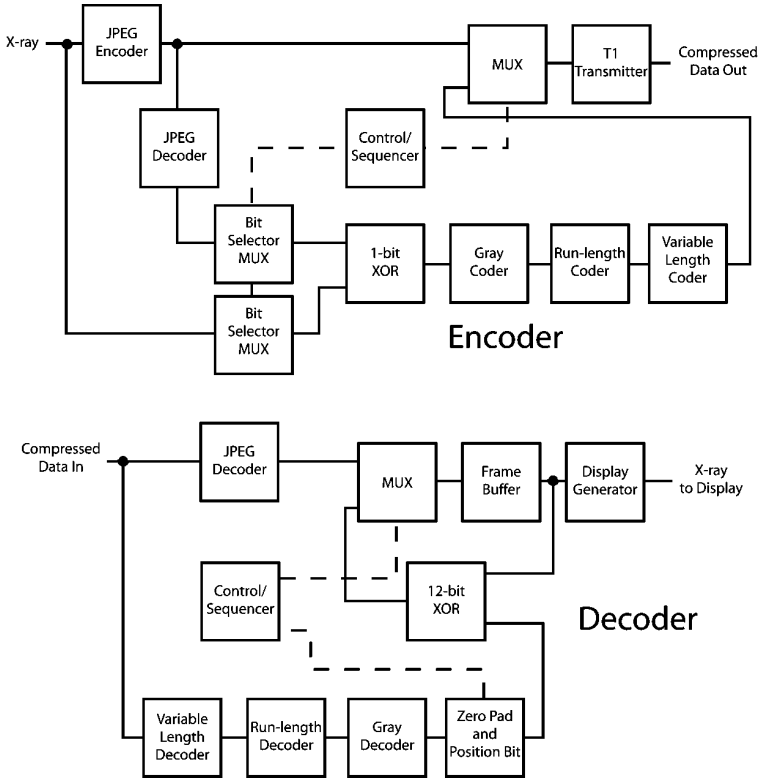


Figure P8.24

Problem 8.25

To demonstrate the equivalence of the lifting based approach and the traditional FWT filter bank method, we simply derive general expressions for one of the odd and even outputs of the lifting algorithm of Eq. (8.6-2). For example, the $Y(0)$ output of step 4 of the algorithm can be written as

$$\begin{aligned} Y_4(0) &= Y_2(0) + \delta[Y_3(-1) + Y_3(1)] \\ &= X(0) + \beta[Y_1(-1) + Y_1(1)] + \delta[Y_3(-1) + Y_3(1)] \end{aligned}$$

where the subscripts on the Y's have been added to identify the step of the lifting algorithm from which the value is generated. Continuing this substitution pattern from earlier steps of the algorithm until $Y_4(0)$ is a function of X's only, we get

$$\begin{aligned} Y(0) &= [1+2\alpha\beta+2\alpha\delta+6\alpha\beta\gamma\delta+2\gamma\delta]\,X(0) \\ &\quad +[\beta+3\beta\gamma\delta+\delta]\,X(1) \\ &\quad +[\alpha\beta+4\alpha\beta\gamma\delta+\alpha\delta+\gamma\delta]\,X(2) \\ &\quad +[\beta\gamma\delta]\,X(3) \\ &\quad +[\alpha\beta\gamma\delta]\,X(4) \\ &\quad +[\beta+3\beta\gamma\delta+\delta]\,X(-1) \\ &\quad +[\alpha\beta+4\alpha\beta\gamma\delta+\alpha\delta+\gamma\delta]\,X(-2) \\ &\quad +[\beta\gamma\delta]\,X(-3) \\ &\quad +[\alpha\beta\gamma\delta]\,X(-4)\,. \end{aligned}$$

Thus, we can form the lowpass analysis filter coefficients shown in Table P8.25-1.

Table P8.25-1

Coefficient Index	Expression	Value
± 4	$\alpha\beta\gamma\delta/K$	0.026748757
± 3	$\beta\gamma\delta/K$	-0.016864118
± 2	$(\alpha\beta+4\alpha\beta\gamma\delta+\alpha\delta+\gamma\delta)/K$	-0.07822326
± 1	$(\beta+3\beta\gamma\delta+\delta)/K$	0.26686411
0	$(1+2\alpha\beta+2\alpha\delta+6\alpha\beta\gamma\delta+2\gamma\delta)/K$	0.60294901

Here, the coefficient expressions are taken directly from our expansion of $Y(0)$ and the division by K is in accordance with step 6 of Eq. (8.6-2). The coefficient values in column 3 are determined by substituting the values of α, β, γ, δ, and K from the text into the expressions of column 2. A similar derivation beginning with

$$Y_3(1) = Y_1(1) + \gamma\,[Y_2(0) + Y_2(2)]$$

yields

$$\begin{aligned} Y(1) &= [\alpha+3\alpha\beta\gamma+\delta]\,X(0) \\ &\quad +[1+2\beta\gamma]\,X(1) \\ &\quad +[\alpha+3\alpha\beta\gamma+\delta]\,X(2) \\ &\quad +[\beta\gamma]\,X(3) \\ &\quad +[\alpha\beta\gamma]\,X(4) \\ &\quad +[\beta\gamma]\,X(-1) \\ &\quad +[\alpha\beta\gamma]\,X(-2) \end{aligned}$$

from which we can obtain the highpass analysis filter coefficients shown in Table P8.25-2

Table P8.25-2

Coefficient Index	Expression	Value
-2	$-K(\alpha\beta\gamma)$	-0.091271762
-1	$-K(\beta\gamma)$	0.057543525
0	$-K(\alpha+3\alpha\beta\gamma+\delta)$	0.591271766
1	$-K(1+2\beta\gamma)$	-1.115087053
2	$-K(\alpha+3\alpha\beta\gamma+\delta)$	0.591271766
3	$-K(\beta\gamma)$	0.057543525
4	$-K(\alpha\beta\gamma)$	-0.091271762

Problem 8.26

From Eq. (8.6-5) and the problem statement, we get that

$$\mu_{2LL} = \mu_0 = 8$$
$$\epsilon_{2LL} = \epsilon_0 + 2 - 2 = \epsilon_0 = 8.$$

Substituting these values into Eq. (8.6-4), we find that for the $2LL$ subband

$$\Delta_{2LL} = 2^{(8+0)-8}\left[1 + \frac{8}{2^{11}}\right] = 1.00390625.$$

Here, we have assumed an 8-bit image so that $R_b = 8$. Likewise, using Eqs. (8.6-5), (8.6-4), and Fig. 8.46 (to find the analysis gain bits for each subband), we get

$$\Delta_{2HH} = 2^{(8+2)-8}\left[1 + \tfrac{8}{2^{11}}\right] = 4.015625$$
$$\Delta_{2HL} = \Delta_{2LH} = 2^{(8+1)-8}\left[1 + \tfrac{8}{2^{11}}\right] = 2.0078125$$
$$\Delta_{1HH} = 2^{(8+2)-8}\left[1 + \tfrac{8}{2^{11}}\right] = 4.015625$$
$$\Delta_{1HL} = \Delta_{1LH} = 2^{(8+1)-8}\left[1 + \tfrac{8}{2^{11}}\right] = 2.0078125.$$

Problem 8.27

The appropriate MPEG decoder is shown in Fig. P8.27.

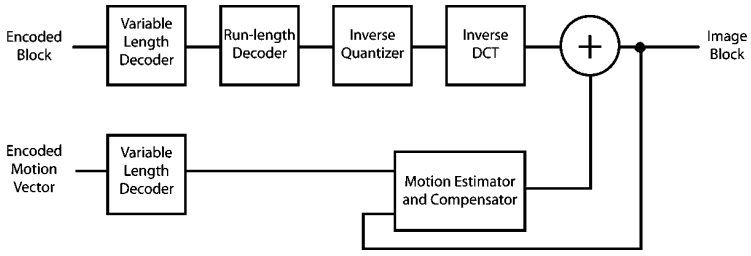


Figure P8.27

9 Problem Solutions

Problem 9.1

(a) Converting a rectangular to a hexagonal grid basically requires that even and odd lines be displaced horizontally with respect to each other by one-half the horizontal distance between adjacent pixels (see the figure in the problem statement). Since in a rectangular grid there are no pixel values defined at the new locations, a rule must be specified for their creation. A simple approach is to double the image resolution in both dimensions by interpolation (see Section 2.4.5). Then, the appropriate 6-connected points are picked out of the expanded array. The resolution of the new image will be the same as the original (but the former will be slightly blurred due to interpolation). Figure P9.1(a) illustrates this approach. The black points are the original pixels and the white points are the new points created by interpolation. The squares are the image points picked for the hexagonal grid arrangement.

(b) Rotations in a 6-neighbor arrangement are invariant to rotations in $60°$ increments.

(c) Yes. Ambiguities arise when there is more than one path that can be followed from one 6-connected pixel to another. Figure P9.1(c) shows an example, in which the 6-connected points of interest are in black.

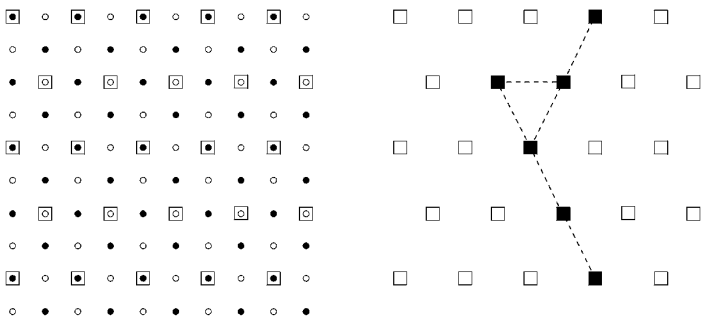


Figure P9.1

Problem 9.2

(a) The answer is shown shaded in Fig. P9.2.

(b) With reference to the sets shown in the problem statement, the answers are, from left to right,

$$(A \cap B \cap C) - (B \cap C);$$
$$(A \cap B \cap C) \cup (A \cap C) \cup (A \cap B); \text{ and}$$
$$\{B \cap (A \cup C)^c\} \cup \{(A \cap C) - [(A \cap C) \cap (B \cap C)]\}.$$

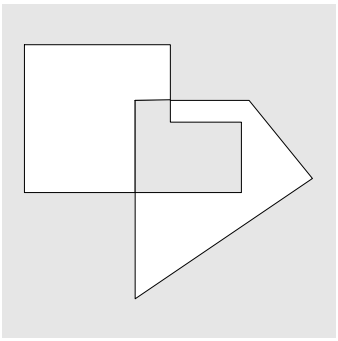


Figure P9.2

Problem 9.3

With reference to the discussion in Section 2.5.2, *m*-connectivity is used to avoid multiple paths that are inherent in 8-connectivity. In one-pixel-thick, fully connected boundaries, these multiple paths manifest themselves in the four basic patterns shown in Fig. P9.3.

The solution to the problem is to use the hit-or-miss transform to detect the patterns and then to change the center pixel to 0, thus eliminating the multiple paths. A basic sequence of morphological steps to accomplish this is as follows:

$$X_1 = A \circledast B^1$$
$$Y_1 = A \cap X_1^c$$

$$
\begin{aligned}
X_2 &= Y_1 \circledast B^2 \\
Y_2 &= Y_1 \cap X_2^c \\
X_3 &= Y_2 \circledast B^3 \\
Y_3 &= Y_2 \cap X_3^c \\
X_4 &= Y_3 \circledast B^4 \\
Y_4 &= Y_3 \cap X_4^c
\end{aligned}
$$

where A is the input image containing the boundary.

(b) Only one pass is required. Application of the hit-or-miss transform using a given B^i finds all instances of occurrence of the pattern described by that structuring element.

(c) The order does matter. For example, consider the sequence of points shown in Fig. P9.3(c). and assume that we are traveling from left to right. If B^1 is applied first, point a will be deleted and point b will remain after application of all other structuring elements.. If, on the other hand, B^3 is applied first, point b will be deleted and point a will remain. Thus, we would end up with different (but of course, acceptable) m-paths.

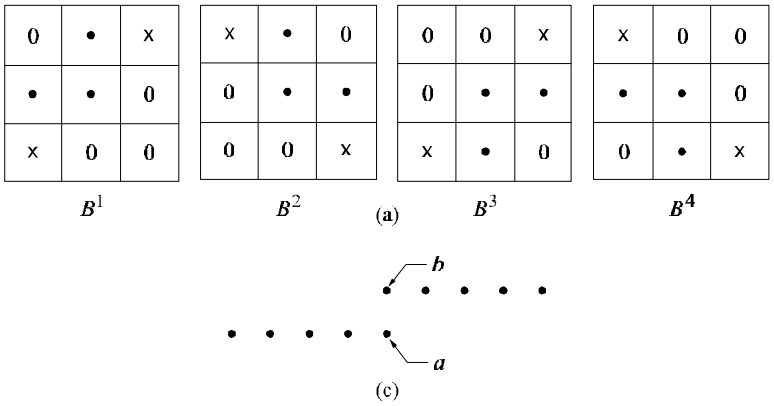


Figure P9.3

Problem 9.4

See Fig. P9.4. Keep in mind that erosion is the set described by the *origin* of the structuring element, such that the structuring element is contained within the set being eroded.

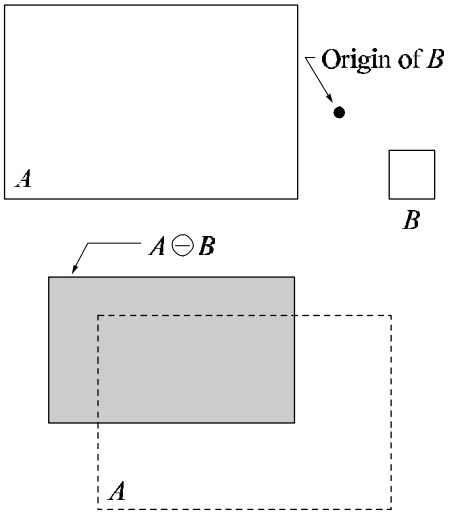


Figure P9.4

Problem 9.5

(a) Erosion is set intersection. The intersection of two convex sets is convex also. See Fig. P9.5 for solutions to parts (b) through (d). Keep in mind that the digital sets in question are the larger black dots. The lines are shown for convenience in visualizing what the continuous sets would be. In (b) the result of dilation is not convex because the center point is not in the set. In (c) we see that the lower right point is not connected to the others. In (d), it is clear that the two inner points are not in the set.

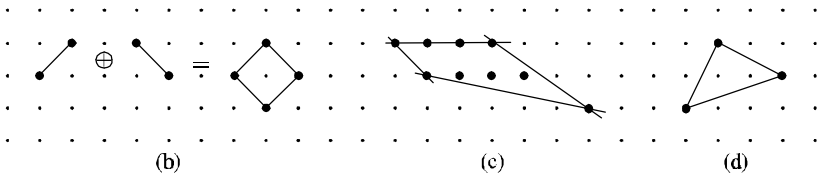


Figure P9.5

Problem 9.6

Refer to Fig. P9.6. The center of each structuring element is shown as a black dot. Solution (a) was obtained by eroding the original set (shown dashed) with the structuring element shown (note that the origin is at the bottom, right). Solution (b) was obtained by eroding the original set with the tall rectangular structuring element shown. Solution

(c) was obtained by first eroding the image shown down to two vertical lines using the rectangular structuring element; this result was then dilated with the circular structuring element. Solution (d) was obtained by first dilating the original set with the large disk shown. Then dilated image was then eroded with a disk of half the diameter of the disk used for dilation.

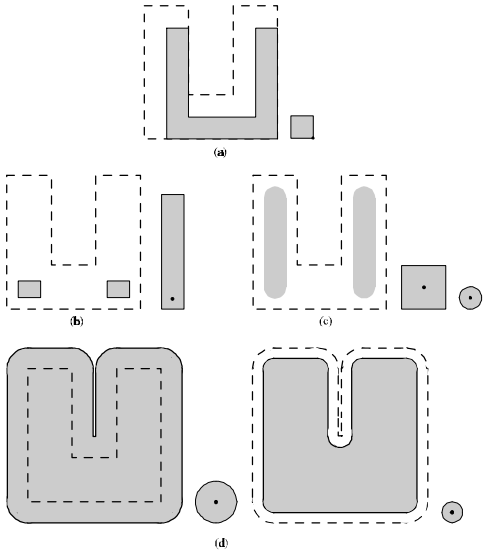


Figure P9.6

Problem 9.7

The solutions to (a) through (d) are shown from top to bottom in Fig. P9.7.

Problem 9.8

(a) The dilated image will grow without bound. (b) A one-element set (i.e., a one-pixel image).

Problem 9.9

(a) The image will erode to one element. (b) The smallest set that contains the structuring element.

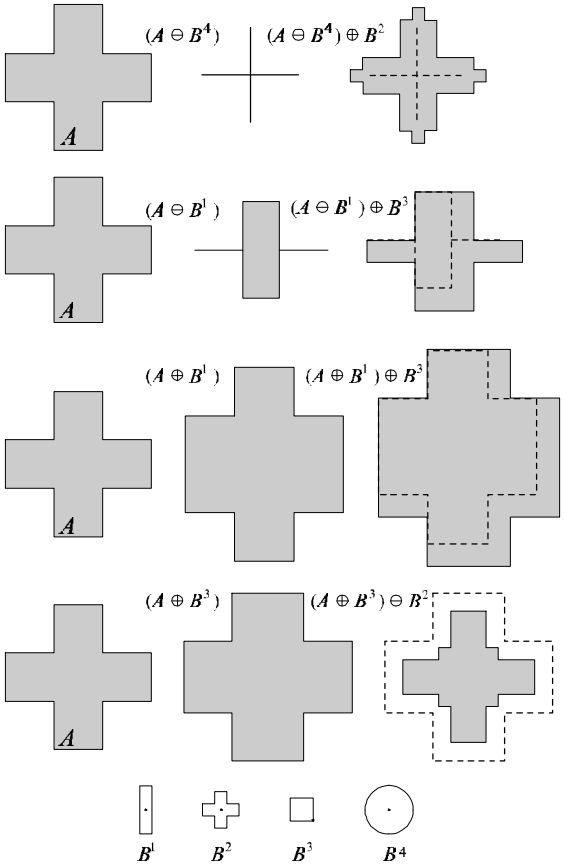


Figure P9.7

Problem 9.10

The approach is to prove that

$$\left\{x \in Z^2 \,\middle|\, (\hat{B})_x \cap A \neq \emptyset\right\} \equiv \left\{x \in Z^2 \,|\, x = a + b \quad \text{for } a \in A \text{ and } b \in B\right\}.$$

The elements of $(\hat{B})_x$ are of the form $x - b$ for $b \in B$. The condition $(\hat{B})_x \cap A \neq \emptyset$ implies that for some $b \in B$, $x - b \in A$, or $x - b = a$ for some $a \in A$ (note in the preceding equation that $x = a + b$). Conversely, if $x = a + b$ for some $a \in A$ and $b \in B$, then $x - b = a$ or $x - b \in A$, which implies that $(\hat{B})_x \cap A \neq \emptyset$.

Problem 9.11

(a) Suppose that $x \in A \oplus B$. Then, for some $a \in A$ and $b \in B$, $x = a + b$. Thus,

$x \in (A)_b$ and, therefore, $x \in \bigcup_{b \in B} (A)_b$. On the other hand, suppose that $x \in \bigcup_{b \in B} (A)_b$. Then, for some $b \in B$, $x \in (A)_b$. However, $x \in (A)_b$ implies that there exists an $a \in A$ such that $x = a + b$. But, from the definition of dilation given in the problem statement, $a \in A$, $b \in B$, and $x = a + b$ imply that $x \in A \oplus B$.

(b) Suppose that $x \in \bigcup_{b \in B} (A)_b$. Then, for some $b \in B$, $x \in (A)_b$. However, $x \in (A)_b$ implies that there exists an $a \in A$ such that $x = a + b$. But, if $x = a + b$ for some $a \in A$ and $b \in B$, then $x - b = a$ or $x - b \in A$, which implies that $x \in \left[(\hat{B})_x \cap A \neq \emptyset\right]$. Now, suppose that $x \in \left[(\hat{B})_x \cap A \neq \emptyset\right]$. The condition $(\hat{B})_x \cap A \neq \emptyset$ implies that for some $b \in B$, $x - b \in A$ or $x - b = a$ (i.e., $x = a + b$) for some $a \in A$. But, if $x = a + b$ for some $a \in A$ and $b \in B$, then $x \in (A)_b$ and, therefore, $x \in \bigcup_{b \in B} (A)_b$.

Problem 9.12

The proof, which consists of proving that

$$\left\{x \in Z^2 \,|\, x + b \in A \text{ , for every } b \in B\right\} \equiv \left\{x \in Z^2 \,|\, (B)_x \subseteq A\right\},$$

follows directly from the definition of translation because the set $(B)_x$ has elements of the form $x + b$ for $b \in B$. That is, $x + b \in A$ for *every* $b \in B$ implies that $(B)_x \subseteq A$. Conversely, $(B)_x \subseteq A$ implies that *all* elements of $(B)_x$ are contained in A, or $x+b \in A$ for every $b \in B$.

Problem 9.13

(a) Let $x \in A \ominus B$. Then, from the definition of erosion given in the problem statement, for every $b \in B$, $x + b \in A$. But, $x + b \in A$ implies that $x \in (A)_{-b}$. Thus, for *every* $b \in B$, $x \in (A)_{-b}$, which implies that $x \in \bigcap_{b \in B} (A)_{-b}$. Suppose now that $x \in \bigcap_{b \in B} (A)_{-b}$. Then, for every $b \in B$, $x \in (A)_{-b}$. Thus, for every $b \in B$, $x + b \in A$ which, from the definition of erosion, means that $x \in A \ominus B$.

(b) Suppose that $x \in A \ominus B = \bigcap_{b \in B} (A)_{-b}$. Then, for every $b \in B$, $x \in (A)_{-b}$, or $x + b \in A$. But, as shown in Problem 9.12, $x + b \in A$ for every $b \in B$ implies that $(B)_x \subseteq A$, so that $x \in A \ominus B = \left\{x \in Z^2 \,|\, (B)_x \subseteq A \right\}$. Similarly, $(B)_x \subseteq A$ implies that all elements of $(B)_x$ are contained in A, or $x + b \in A$ for every $b \in B$ or, as in (a), $x + b \in A$ implies that $x \in (A)_{-b}$. Thus, if for every $b \in B$, $x \in (A)_{-b}$, then $x \in \bigcap_{b \in B} (A)_{-b}$.

Problem 9.14

Starting with the definition of closing,

$$\begin{aligned}(A \bullet B)^c &= [(A \oplus B) \ominus B]^c \\ &= (A \oplus B)^c \oplus \hat{B} \\ &= (A^c \ominus \hat{B}) \oplus \hat{B} \\ &= A^c \circ \hat{B}.\end{aligned}$$

Problem 9.15

(a) Erosion of a set A by B is defined as the set of all values of translates, z, of B such that $(B)_z$ is contained in A. If the origin of B is contained in B, then the set of points describing the erosion is simply all the possible locations of the origin of B such that $(B)_z$ is contained in A. Then it follows from this interpretation (and the definition of erosion) that erosion of A by B is a subset of A. Similarly, dilation of a set C by B is the set of all locations of the origin of $\hat{B}$ such that the intersection of C and $(\hat{B})_z$ is not empty. If the origin of B is contained in B, this implies that C is a subset of the dilation of C by B. Now, from Eq. (9.3-1), we know that $A \circ B = (A \ominus B) \oplus B$. Let C denote the erosion of A by B. It was already established that C is a subset of A. From the preceding discussion, we know also that C is a subset of the dilation of C by B. But C is a subset of A, so the opening of A by B (the erosion of A by B followed by a dilation of the result) is a subset of A.

(b) From Eq. (9.3-3),

$$C \circ B = \bigcup \{(B)_z \,|(B)_z \subseteq C\}$$

and

$$D \circ B = \bigcup \{(B)_z \,|(B)_z \subseteq D\}.$$

Therefore, if $C \subseteq D$, it follows that $C \circ B \subseteq D \circ B$.

(c) From (a), $(A \circ B) \circ B \subseteq (A \circ B)$. From the definition of opening,

$$\begin{aligned}(A \circ B) \circ B &= \{(A \circ B) \ominus B\} \oplus B \\ &= \{[(A \ominus B) \oplus B] \ominus B\} \oplus B \\ &= \{(A \ominus B) \bullet B\} \oplus B \\ &\supseteq (A \ominus B) \oplus B \\ &\supseteq A \circ B.\end{aligned}$$

But, the only way that $(A \circ B) \circ B \subseteq (A \circ B)$ and $(A \circ B) \circ B \supseteq (A \circ B)$ can hold is if $(A \circ B) \circ B = (A \circ B)$. The next to last step in the preceding sequence follows from the fact that the closing of a set by another contains the original set [this is from Problem 9.16(a)].

Problem 9.16

(a) From Problem 9.14, $(A \bullet B)^c = A^c \circ \hat{B}$, and, from Problem 9.15(a), it follows that

$$(A \bullet B)^c = A^c \circ \hat{B} \subseteq A^c.$$

Taking the complement of both sides of this equation reverses the inclusion sign and we have that $A \subseteq (A \bullet B)$, as desired.

(b) From Problem 9.16(b), if $D^c \subseteq C^c$, then $D^c \circ \hat{B} \subseteq C^c \circ \hat{B}$ where we used D^c, C^c, and $\hat{B}$ instead of C, D, and B. From Problem 9.15, $(C \bullet B)^c = C^c \circ \hat{B}$ and $(D \bullet B)^c = D^c \circ \hat{B}$. Therefore, if $D^c \subseteq C^c$ then $(D \bullet B)^c \subseteq (C \bullet B)^c$. Taking complements reverses the inclusion, so we have that if $C \subseteq D$, then $(C \bullet B) \subseteq (D \bullet B)$, as desired.

(c) Starting with the result of Problem 9.15,

$$\begin{aligned}(A \bullet B) \bullet B &= \left\{(A \bullet B)^c \circ \hat{B}\right\}^c \\ &= \left\{(A^c \circ \hat{B}) \circ \hat{B}\right\}^c \\ &= \left\{(A^c \circ \hat{B})\right\}^c \\ &= \{(A \bullet B)^c\}^c \\ &= (A \bullet B).\end{aligned}$$

where the third step follows from Problem 9.15(c) and the fourth step follows from Problem 9.14.

Problem 9.17

The solution is shown in Fig. P9.17. Although the images shown could be sketched by hand, they were done in MATLAB The size of the original is 647×624 pixels. A disk structuring element of radius 11 was used. This structuring element was just large enough to encompass all noise elements, as given in the problem statement. The images shown in Fig. P9.17 are: (a) erosion of the original, (b) dilation of the result, (c) another dilation, and finally (d) an erosion. The main points we are looking for from the student's answer are: The first erosion (leftmost image) should take out all noise elements that do not touch the rectangle, should increase the size of the noise elements completely contained within the rectangle, and should decrease the size of the rectangle. If worked by hand, the student may or may not realize that some "imperfections" are left along the boundary of the object. We do not consider this an important issue because it is scale-dependent, and nothing is said in the problem statement about this. The first dilation (next image) should shrink the noise components that were increased in erosion, should increase the size of the rectangle, and should round the corners. The next dilation should eliminate the internal noise components completely and further increase the size of the rectangle. The final erosion (last image on the right) should then decrease the size of the rectangle. The rounded corners in the final answer are an important point that should be recognized by the student.

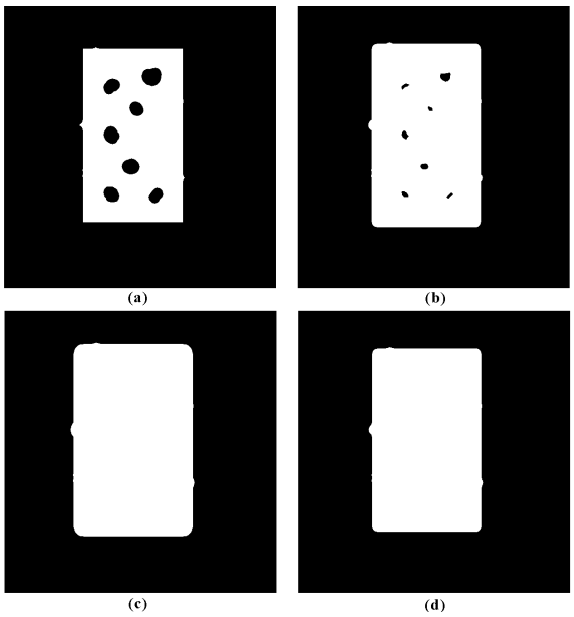


Figure P9.17

Problem 9.18

It was possible to reconstruct the three large squares to their original size because they were not completely eroded and the geometry of the objects and structuring element was the same (i.e., they were squares). This also would have been true if the objects and structuring elements were rectangular. However, a complete reconstruction, for instance, by dilating a rectangle that was partially eroded by a circle, would not be possible.

Problem 9.19

(a) Select a one-pixel border around the image of the T, assuming that the resulting subimage is odd, let the origin be located at the horizontal/vertical midpoint of this subimage (if the dimensions were even, we could just as easily select any other point). The resulting of applying the hit-or-miss transform would be a single point where the two T's were in perfect registration. The location of the point would be the same as the origin of the structuring element.

(b) The hit-or-miss transform and (normalized) correlation are similar in the sense that they produce their maximum value at the location of a perfect match, and also in the mechanics of sliding the template (structuring element) past all locations in the image. Major differences are the lack of a complex conjugate in the hit-or-miss transform, and the fact that this transform produced a single nonzero binary value in this case, as opposed to the multiple nonzero values produced by correlation of the two images.

Problem 9.20

The key difference between the Lake and the other two features is that the former forms a closed contour. Assuming that the shapes are processed one at a time, basic two-step approach for differentiating between the three shapes is as follows:

Step 1. Apply an end-point detector to the object until convergence is achieved. If the result is not the empty set, the object is a Lake. Otherwise it is a Bay or a Line.

Step 2. There are numerous ways to differentiate between a lake and a line. One of the simplest is to determine a line joining the two end points of the object. If the AND of the object and this line contains only two points, the figure is a Bay. Otherwise it is

a line segment. There are pathological cases in which this test will fail, and additional "intelligence" needs to be built into the process, but these pathological cases become less probable with increasing resolution of the thinned figures.

Problem 9.21

(a) The entire image would be filled with 1's. (b) The background would be filled with 1's. (c) See Fig. P9.21.

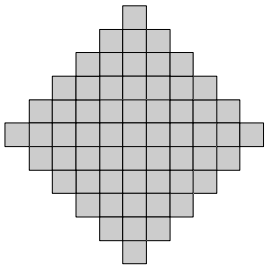


Figure P9.21

Problem 9.22

(a) With reference to the example shown in Fig. P9.22(a), the boundary that results from using the structuring element in Fig. 9.15(c) generally forms an 8-connected path (leftmost figure), whereas the boundary resulting from the structuring element in Fig. 9.13(b) forms a 4-connected path (rightmost figure).

(b) Using a 3×3 structuring element of all 1's would introduce corner pixels into segments characterized by diagonally-connected pixels. For example, square (2,2) in Fig. 9.15(e) would be a 1 instead of a 0. That value of 1 would carry all the way to the final result in Fig. 9.15(i). There would be other 1's introduced that would turn Fig. 9.15(i) into a much more distorted object.

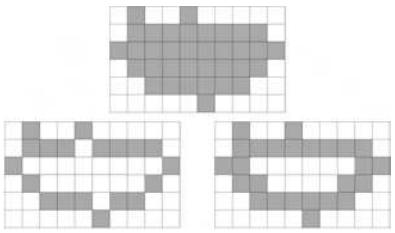


Figure P9.22(a)

Problem 9.23

If spheres are allowed to touch, we can make the simplifying assumption that no spheres touch it in such a way that they create "pockets" of black points surrounded by all white or surrounded by all white and part of the boundary of the image. This situation requires additional preprocessing, as discussed below. With these simplification in mind, the problem reduces first to determining which points are background (black) points. To do this, we pick a black point on the boundary of the image and find all black points connected to it using a connected component algorithm (Section 9.5.3). These connected components are labels with a value different from 1 or 0. The remaining black points are interior to spheres. We can fill all spheres with white by applying the region filling algorithm until all interior black points have been turned into white points. The alert student will realize that if the interior points are already known, they can all be turned simply into white points thus filling the spheres without having to do region filling as a separate procedure.

If the spheres are allowed to touch in arbitrary ways, a way must be found to separate them because they could create "pockets" of black points surrounded by all white or surrounded by all white and part of the boundary of the image. The simplest approach is to separate the spheres by preprocessing. One way to do this is to erode the white components of the image by one pass of a 3×3 mask, effectively creating a black border around the spheres, thus "separating" them. This approach works in this case because the objects are spherical, thus having small areas of contact. To handle the case of spheres touching the border of the image, we simply set all border point to black. We then proceed to find all background points To do this, we pick a point on the boundary of the image (which we know is black due to preprocessing) and find all black points connected to it using a connected component algorithm (Section 9.5.3). These connected components are labels with a value different from 1 or 0. The remaining black points are interior to spheres. We can fill all spheres with white by applying the region filling algorithm until all such interior black points have been turned into white points. The alert student will realize that if the interior points are already known, they can all be turned simply into white points thus filling the spheres without having to do region filling as a separate procedure.

Note that the erosion of white areas makes the black areas interior to the spheres grow, so the possibility exists that such an area near the border of a sphere could grow into the background. This issue introduces further complications that the student may not have the tools to solve yet. We recommend making the assumption that the interior black

areas are small and near the center. Recognition of the potential problem by the student should be sufficient.

Problem 9.24

Denote the original image by A. Create an image of the same size as the original, but consisting of all 0's, call it B. Choose an arbitrary point labeled 1 in A, call it p_1, and apply the algorithm. When the algorithm converges, a connected component has been detected. Label and copy into B the set of all points in A belonging to the connected components just found, set those points to 0 in A and call the modified image A_1. Choose an arbitrary point labeled 1 in A_1, call it p_2, and repeat the procedure just given. If there are K connected components in the original image, this procedure will result in an image consisting of all 0's after K applications of the procedure just given. Image B will contain K labeled connected components.

Problem 9.25

(a) Equation (9.6-1) requires that the (x, y) used in the computation of dilation must satisfy the condition $(x, y) \in D_b$. In terms of the intervals given in the problem statement, this means that x and y must be in the closed interval $x \in [B_{x1}, B_{x2}]$ and $y \in [B_{y1}, B_{y2}]$. It is required also that $(s - x), (t - y) \in D_f$, which means that $(s - x) \in [F_{x1}, F_{x2}]$ and $(t - y) \in [F_{y1}, F_{y2}]$. Since the valid range of x is the interval $[B_{x1}, B_{x2}]$, the valid range of $(s-x)$ is $[s-B_{x1}, s-B_{x2}]$. But, since x must also satisfy the condition $(s - x) \in [F_{x1}, F_{x2}]$, it follows that $F_{x1} \leq s - B_{x1}$ and $F_{x2} \geq s - B_{x2}$, which finally yields $F_{x1} + B_{x1} \leq s \leq F_{x2} + B_{x2}$. Following the same analysis for t yields $F_{y1} + B_{y1} \leq t \leq F_{y2} + B_{y2}$. Since dilation is a function of (s, t), these two inequalities establish the domain of $(f \oplus b)(s, t)$ in the st-plane.

(b) Following a similar procedure yields the following intervals for s and t: $F_{x1} - B_{x1} \leq s \leq F_{x2} - B_{x2}$ and $F_{y1} - B_{y1} \leq t \leq F_{y2} - B_{y2}$. Since erosion is a function of (s, t), these two inequalities establish the domain of $(f \ominus b)(s, t)$ in the st-plane.

Problem 9.26

(a) The noise spikes are of the general form shown in Fig. P9.26(a), with other possibilities in between. The amplitude is irrelevant in this case; only the shape of the noise

spikes is of interest. To remove these spikes we perform an opening with a cylindrical structuring element of radius greater than $R_{\max}$, as shown in Fig. P9.26(b) (see Fig. 9.30 for an explanation of the process). Note that the shape of the structuring element is matched to the known shape of the noise spikes.

(b) The basic solution is the same as in (a), but now we have to take into account the various possible overlapping geometries shown in Fig. P9.26(c). A structuring element like the one used in (a) but with radius slightly larger than $4R_{\max}$ will do the job. Note in (a) and (b) that other parts of the image would be affected by this approach. The bigger $R_{\max}$, the bigger the structuring element that would be needed and, consequently, the greater the effect on the image as a whole.

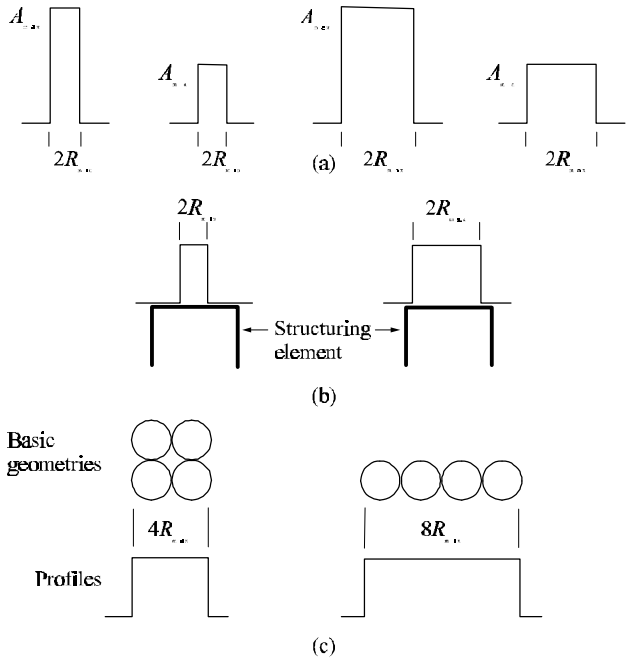


Figure P9.26

Problem 9.27

(a) Color the image border pixels the same color as the particles (white). Call the resulting set of border pixels B. Apply the connected component algorithm. All connected components that contain elements from B are particles that have merged with the border of the image.

(b) It is given that all particles are of the same size (this is done to simplify the problem; more general analysis requires tools from Chapter 11). Determine the area (number of pixels) of a single particle; denote the area by R. Eliminate from the image the particles that were merged with the border of the image. Apply the connected component algorithm. Count the number of pixels in each component. A component is then designated as a single particle if the number of pixels is less than or equal to $R + \varepsilon$, where ε is a small quantity added to account for variations in size due to noise.

(c) Subtract from the image single particles and the particles that have merged with the border, and the remaining particles are overlapping particles.

Problem 9.28

As given in the problem statement, interest lies on deviations from the round in the inner and outer boundaries of the washers. It also is stated that we can ignore errors due to digitizing and positioning. This means that the imaging system has enough resolution so that artifacts will not be introduced as a result of digitization. The mechanical accuracy similarly tells us that no appreciable errors will be introduced as a result of positioning. This is important if we want to do matching without having to register the images.

The first step in the solution is the specification of an illumination approach. Because we are interested in boundary defects, the method of choice is a backlighting system that will produce a binary image. We are assured from the problem statement that the illumination system has enough resolution so that we can ignore defects due to digitizing.

The next step is to specify a comparison scheme. The simplest way to match binary images is to AND one image with the complement of the other. Here, we match the input binary image with the complement of the golden image (this is more efficient than computing the complement of each input image and comparing it to the golden image). If the images are identical (and perfectly registered) the result of the AND operation will be all 0's. Otherwise, there will be 1's in the areas where the two images do not match. Note that this requires that the images be of the same size and be registered, thus the assumption of the mechanical accuracy given in the problem statement.

As noted, differences in the images will appear as regions of 1's in the AND image. These we group into regions (connected components) by using the algorithm given in Section 9.5.3. Once all connected components have been extracted, we can compare them against specified criteria for acceptance or rejection of a given washer. The sim-

plest criterion is to set a limit on the number and size (number of pixels) of connected components. The most stringent criterion is 0 connected components. This means a perfect match. The next level for ”relaxing” acceptance is one connected component with of size 1, and so on. More sophisticated criteria might involve measures like the shape of connected components and the relative locations with respect to each other. These types of descriptors are studied in Chapter 11.

10 Problem Solutions

Problem 10.1

The masks would have the coefficients shown in Fig. P10.1. Each mask would yield a value of 0 when centered on a pixel of an unbroken 3-pixel segment oriented in the direction favored by that mask. Conversely, the response would be a +2 when a mask is centered on a one-pixel gap in a 3-pixel segment oriented in the direction favored by that mask.

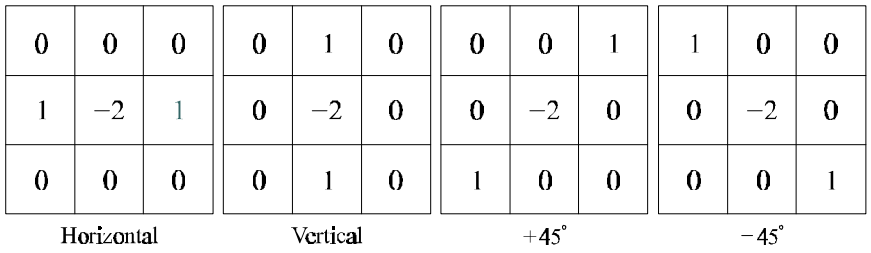


Figure P10.1

Problem 10.2

The key to solving this problem is to find all end points of line segments in the image. End points are those points on a line which have only one 8-neighbor valued 1. Once all end points have been found, the D_8 distance between all pairs of such end points gives the lengths of the various gaps. We choose the smallest distance between end points of every pair of segments and any such distance less than or equal to L satisfies the statement of the problem. This is a rudimentary solution, and numerous embellishments can be added to build intelligence into the process. For example, it is possible for end points of different, but closely adjacent, lines to be less than L pixels apart, and heuristic tests that attempt to sort out things like this are quite useful. Although the problem statement does not call for any such tests, they are normally needed in practice and it is

worthwhile to bring this up in class if this particular problem is assigned as a homework assignment.

Problem 10.3

(a) The lines were thicker than the width of the line detector masks. Thus, when, for example, a mask was centered on the line it "saw" a constant area and gave a response of 0.

(b) Via connectivity analysis.

Problem 10.4

It is given that the location of the edge relative to the size of the mask is such that image border effects can be ignored. Assume that n is odd and keep in mind that an ideal step edge transition takes place between adjacent pixels. Then, the average is 0 until the center of the mask is $(n-1)/2$ pixels or more to the left of the edge. The average is 1 when the center of the mask is further away than $(n-1)/2$ pixels to the right of the edge. When transitioning into the edge, (say from left to right) the average picks up one column of the mask for every pixel that it moves to the right, so the value of the average grows as $n/n^2, 2n/n^2, \ldots, (n-1)\,n/n^2, n^2/n^2$, or $1/n, 2/n, \ldots, (n-1)/n, 1$. This is a simple linear growth with slope equal to $1/n$. Figure P10.4 shows a plot of the original profile and what the profile would look like after smoothing. Thus, we get a ramp edge, as expected.

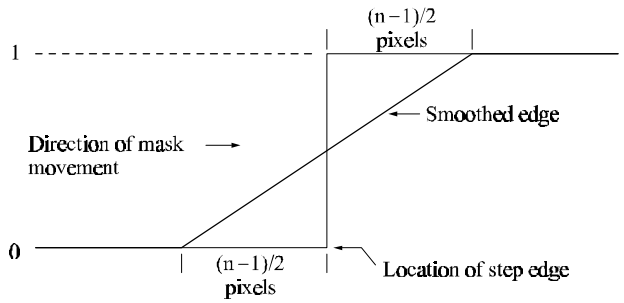


Figure P10.4

Problem 10.5

The gradient and Laplacian (first and second derivatives) are shown in Fig. P10.5.

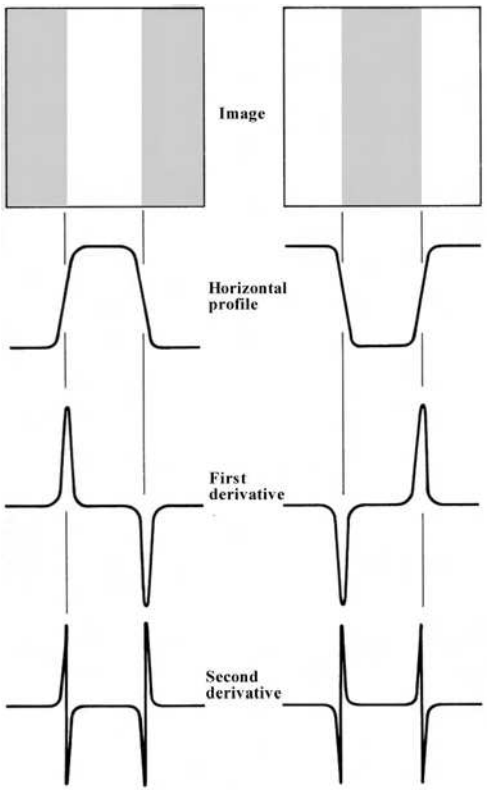


Figure P10.5

Problem 10.6

(a) Inspection of the Sobel masks shows that $G_x = 0$ for edges oriented vertically and $G_y = 0$ for edges oriented horizontally. Therefore, it follows in this case that , for vertical edges, $\nabla f = \sqrt{G_y^2} = |G_y|$, and similarly for horizontal edges.

(b) The same argument applies to the Prewitt masks.

Problem 10.7

Consider first the Sobel masks of Figs. 10.8 and 10.9. The easiest way to prove that

these masks give isotropic results for edge segments oriented at multiples of $45°$ is to obtain the mask responses for the four general edge segments shown in Fig. P10.7, which are oriented at increments of $45°$. The objective is to show that the responses of the Sobel masks are indistinguishable for these four edges. That this is the case is evident from Table P10.1, which shows the response of each Sobel mask to the four general edge segments. We see that in each case the response of the mask that matches the edge direction is $(4a - 4b)$, and the response of the corresponding orthogonal mask is 0. The response of the remaining two masks is either $(3a - 3b)$ or $(3b - 3a)$. The sign difference is not significant because the gradient is computed by either squaring or taking the absolute value of the mask responses. The same line of reasoning applies to the Prewitt masks.

Table P10.7

Edge direction	Horizontal Sobel (G_x)	Vertical Sobel (G_y)	$+45°$ Sobel (G_{45})	$-45°$ Sobel (G_{-45})
Horizontal	$4a - 4b$	0	$3a - 3b$	$3b - 3a$
Vertical	0	$4a - 4b$	$3a - 3b$	$3a - 3b$
$+45°$	$3a - 3b$	$3a - 3b$	$4a - 4b$	0
$-45°$	$3b - 3a$	$3a - 3b$	0	$4a - 4b$

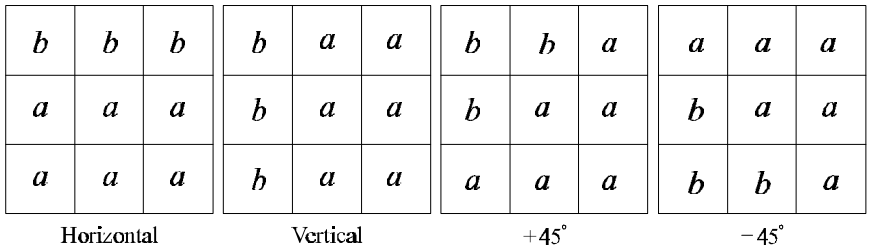


Figure P10.7

Problem 10.8

With reference to Fig. P10.8, consider first the 3×3 smoothing mask mentioned in the problem statement, as well as the general subimage area shown in the figure. Recall that value e is replaced by the response of the 3×3 mask when its center is at that location. Ignoring the 1/9 scale factor, the response of the mask when centered at that location is $(a + b + c + d + e + f + g + h + i)$.

The idea with the one-dimensional mask is the same: We replace the value of a pixel by the response of the mask when it is centered on that pixel. With this in mind, the mask

[1 1 1] would yield the following responses when centered at the pixels with values *b*, *e*, and *h*, respectively: $(a+b+c)$, $(d+e+f)$, and $(g+h+i)$. Next, we pass the mask

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

through these results. When this mask is centered at the pixel with value e, its response will be $[(a+b+c) + (d+e+f) + (g+h+i)]$, which is the same as the result produced by the 3×3 smoothing mask.

Returning now to problem at hand, when the G_x Sobel mask is centered at the pixel with value e, its response is $G_x = (g+2h+i) - (a+2b+c)$. If we pass the one-dimensional differencing mask

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

through the image, its response when its center is at the pixels with values d, e, and f, respectively, would be:$(g-a)$, $(h-b)$, and $(i-c)$. Next we apply the smoothing mask [1 2 1] to these results. When the mask is centered at the pixel with value e, its response would be $[(g-a) + 2(h-b) + (i-c)]$ which is $[(g+2h+i) - (a+2b+c)]$. This is the same as the response of the 3×3 Sobel mask for G_x. The process to show equivalence for G_y is basically the same. Note, however, that the directions of the one-dimensional masks would be reversed in the sense that the differencing mask would be a column mask and the smoothing mask would be a row mask.

1	1	1
1	1	1
1	1	1

Smoothing mask (scaled by 1/9).

a	*b*	*c*
d	*e*	*f*
g	*h*	*i*

Subimage area under the mask at any one time.

Figure P10.8

Problem 10.9

The solution is shown in Fig. P10.9 (negative numbers are shown underlined).

Edge direction							
E	NE	N	NW	W	SW	S	SE
Gradient direction							
N	NW	W	SW	S	SE	E	NE
Compass gradient operators							
1 1 1	1 1 0	1 0 $\underline{1}$	0 $\underline{1}$ $\underline{1}$	$\underline{1}$ $\underline{1}$ $\underline{1}$	$\underline{1}$ $\underline{1}$ 0	$\underline{1}$ 0 1	0 1 1
0 0 0	1 0 $\underline{1}$	1 0 $\underline{1}$	1 0 $\underline{1}$	0 0 0	$\underline{1}$ 0 1	$\underline{1}$ 0 1	$\underline{1}$ 0 1
$\underline{1}$ $\underline{1}$ $\underline{1}$	0 $\underline{1}$ $\underline{1}$	1 0 $\underline{1}$	1 1 0	1 1 1	0 1 1	$\underline{1}$ 0 1	$\underline{1}$ $\underline{1}$ 0

Figure P10.9

Problem 10.10

(a) The solution is shown in Fig. P10.10(a). The numbers in brackets are values of $[G_x, G_y]$. (b) The solution is shown in Fig. P10.10(b). The angle was not computed for the trivial cases in which $G_x = G_y = 0$.. The histogram follows directly from this table. (c) The solution is shown in Fig. P10.10(c).

Problem 10.11

(a) With reference to Eq. (10.1-17), we need to prove that

$$\int_{-\infty}^{\infty} \left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} dr = 0.$$

Expanding this equation results in the expression

$$\int_{-\infty}^{\infty} \left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} dr = \frac{1}{\sigma^4} \int_{-\infty}^{\infty} r^2 e^{-\frac{r^2}{2\sigma^2}} dr - \frac{1}{\sigma^2} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr.$$

Recall from the definition of the Gaussian density that

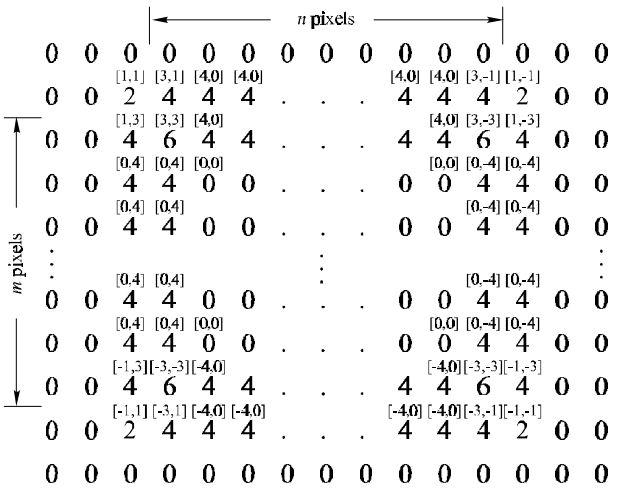
$$\frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{r^2}{2\sigma^2}} dr = 1$$

and, from the definition of the variance of a Gaussian random variable that

$$\mathrm{Var}(r) = \sigma^2 = \int_{-\infty}^{\infty} r^2 e^{-\frac{r^2}{2\sigma^2}} dr.$$

Thus, it follows from the preceding equations that

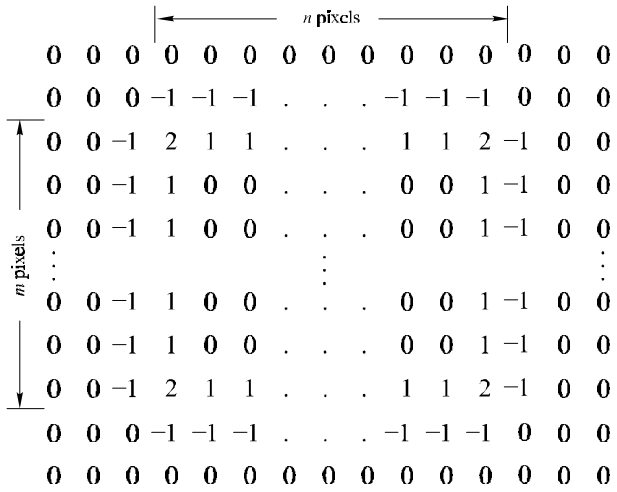
$$\int_{-\infty}^{\infty} \left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} dr = \frac{\sqrt{2\pi\sigma^2}}{\sigma^4}\sigma^2 - \frac{\sqrt{2\pi\sigma^2}}{\sigma^2} = 0.$$



(a)

Angle	No. of points with that angle
0°	2(n−2)
18.4°	2
45°	4
71.6°	2
90°	(m−2)
−18.4°	2
−71.6°	2
−45°	4
−90°	(n−2)

(b)



(c)

Figure P10.10

(b) Suppose that we convolve an image f with $\nabla^2 h$. Using the convolution theorem, this

is the same as multiplying the Fourier transform of f by the Fourier transform of $\nabla^2 h$. The average value of the convolution can be obtained by evaluating the Fourier transform of this product at the origin of the frequency plane [see Eq. (4.2-22)]. But, it was shown in (a) that the average value of $\nabla^2 h$ is zero, which means that its Fourier transform is zero at the origin. From this it follows that the value of the product of the two Fourier transforms is also zero, thus proving that the average value of the convolution of f with $\nabla^2 h$ is zero.

(c) Yes. Consider Eq. (10.1-14), expressed as

$$\nabla^2 f(x,y) = 4f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)].$$

As in (b), we evaluate the average value of a spatial expression by looking at the value of its Fourier transform at the origin. Here, it follows from Eq. (4.6-2) that, if $F(u,v)$ denotes the Fourier transform of $f(x,y)$, then the transforms of all the terms inside the brackets in the above equation are $F(u,v)$ multiplied by appropriate exponential terms. However, the exponential terms have value 1 at the origin, so the net result is $4F(0,0) - 4F(0,0) = 0$, thus proving that the Laplacian obtained by convolving an image with the operator shown in Fig. 10.13 (which implements Eq. (10.1-14)] has an average value of zero. The same zero result is obtained for Eq. (10.1-15).

Problem 10.12

(a) Figure 10.15(g) was obtained from Fig. 10.15(h) which is a binary image, and thus consists of sets of connected components of 1's (see Section 2.5.2 regarding connected components). The boundary of each connected component forms a closed path (Problem 2.14). The contours in Fig. 10.15(g) were obtained by noting transitions of the boundaries of the connected components with the background, and thus form closed paths.

(b) The answer is yes for functions that meet certain mild conditions, and if the zero crossing method is based on rotational operators like the LoG function. Geometrical properties of zero crossings in general are explained in some detail in the paper "On Edge Detection," by V. Torre and T. Poggio, *IEEE Trans. Pattern Analysis and Machine Intell.*, vol. 8, no. 2, pp. 147-163. Looking up this paper and becoming familiar with the mathematical underpinnings of edge detection is an excellent reading assignment for graduate students.

Problem 10.13

(a) Point 1 has coordinates $x = 0$ and $y = 0$. Substituting into Eq. (10.2-3) yields $\rho = 0$, which, in a plot of ρ vs. θ,is a straight line.

(b) Only the origin $(0, 0)$ would yield this result.

(c) At $\theta = +90°$, it follows from Eq. (10.2-3) that $x \cdot (0) + y \cdot (1) = \rho$, or $y = \rho$. At $\theta = -90°$, $x \cdot (0) + y \cdot (-1) = \rho$, or $-y = \rho$. Thus the reflective adjacency.

Problem 10.14

(a) Express $xcos\theta + ysin\theta = \rho$ in the form $x = -(\cot\theta)x + \rho/\sin\theta$. Equating terms with the slope-intercept form, $y = ax + b$, gives $a =$ and $-(\cot\theta)$ and $b = \rho/\sin\theta$. This gives $\theta = \cot^{-1}(a)$ and $\rho = b\sin\theta$. Once obtained from a and b of a given line, the parameters θ and ρ completely specify the normal representation of that line.

(b) $\theta = \cot^{-1}(2) = 26.6°$ and $\rho = (1)\sin\theta = 0.45$.

Problem 10.15

This problem is a natural for the Hough transform, which is set up as follows: The θ axis is divided into six subdivisions, corresponding to the six specified directions and their error bands. For example (since the angle directions specified in the problem statement are with respect to the horizontal) the first band for angle θ extends from $-30°$ to $-20°$, corresponding to the $-25°$ direction and its $\pm 5°$ band. The ρ axis extends from $\rho = -\sqrt{D}$ to $\rho = +\sqrt{D}$, where D is the largest distance between opposite corners of the image, properly calibrated to fit the particular imaging set up used. The subdivisions in the ρ axis are chosen finely enough to resolve the minimum expected distance between tracks that may be parallel, but have different origins, thus satisfying the last condition of the problem statement.

Set up in this way, the Hough transform can be used as a ”filter” to categorize all points in a given image into groups of points in the six specified directions. Each group is then processed further to determine if its points satisfy the criteria for a valid track: (1) each group must have at least 100 points; and (2) it cannot have more than three gaps, each of which cannot be more than 10 pixels long (see Problem 10.2 on the estimation of gaps of a given length).

Problem 10.16

(a) The paths are shown in Fig. P10.16. These paths are as follows:

$$1 : (1,1)(1,2) \rightarrow (2,1)(2,2) \rightarrow (3,1)(3,2)$$

$$2 : (1,1)(1,2) \rightarrow (2,1)(2,2) \rightarrow (3,2)(2,2) \rightarrow (3,2)(3,3)$$

$$3 : (1,1)(1,2) \rightarrow (2,2)(1,2) \rightarrow (2,2)(2,3) \rightarrow (3,2)(3,3)$$

$$4 : (1,1)(1,2) \rightarrow (2,2)(1,2) \rightarrow (2,2)(2,3) \rightarrow (2,2)(3,2) \rightarrow (3,1)(3,2)$$

$$5 : (1,2)(1,3) \rightarrow (2,2)(2,3) \rightarrow (3,2)(3,3)$$

$$6 : (1,2)(1,3) \rightarrow (2,2)(2,3) \rightarrow (2,2)(3,2) \rightarrow (3,1)(3,2)$$

$$7 : (1,2)(1,3) \rightarrow (1,2)(2,2) \rightarrow (2,1)(2,2) \rightarrow (3,1)(3,2)$$

$$8 : (1,2)(1,3) \rightarrow (1,2)(2,2) \rightarrow (2,1)(2,2) \rightarrow (3,2)(2,2) \rightarrow (3,2)(3,3)$$

(b) From Fig. 10.24 and (a), we see that the optimum path is path 6. Its cost is $c = 2 + 0 + 1 + 1 = 4$.

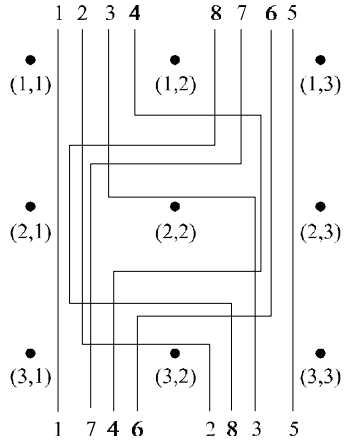


Figure P10.16

Problem 10.17

From Eq. (10.2-6), $c(p,q) = H - [f(p) - f(q)]$. In this case $H = 8$. Assume that p is to the right as the image is traversed from left to right. The possible paths are shown in Fig. P10.17(a). The costs are detailed in Fig. P10.17(b). The graph (with the minimum-cost path shown dashed) is shown in Fig. P10.17(c). Finally, the edge corresponding to the minimum-cost path is shown in Fig. P10.17(d).

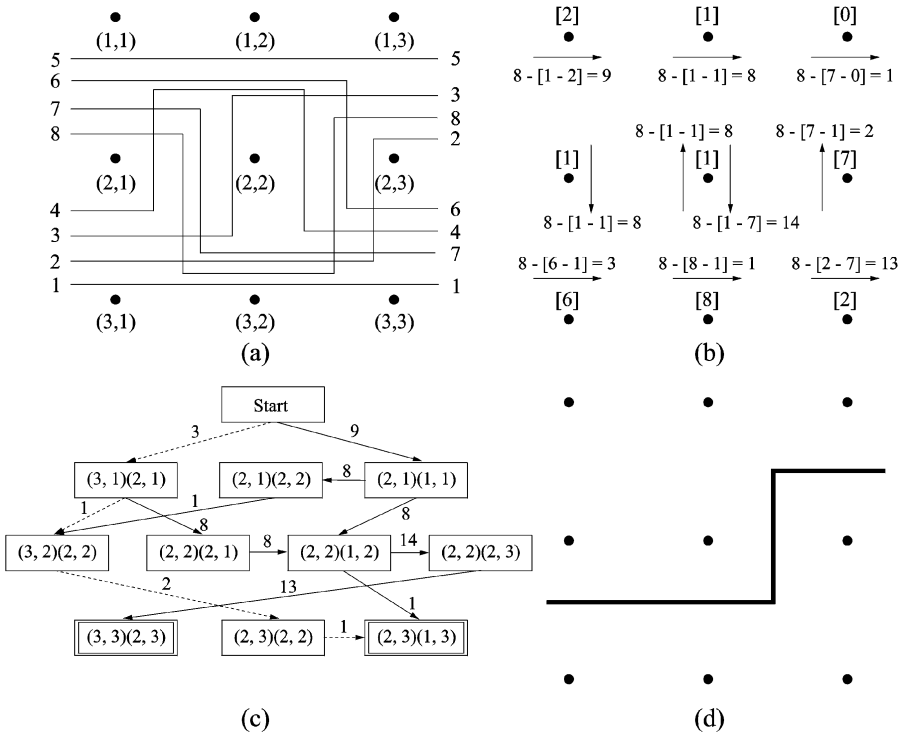


Figure P10.17

Problem 10.18

(a) The number of boundary points between black and white regions is much larger in the image on the right. When the images are blurred, the boundary points will give rise to a larger number of different values for the image on the right, so the histograms of the two blurred images will be different.

(b) To handle border effects, we surround the image with a border of 0's. We assume that the image is of size $N \times N$ (the fact that the image is square is evident from the right image in the problem statement). Blurring is implemented by a 3×3 mask whose coefficients are $1/9$. Figure P10.18 shows the different types of values that the blurred left image (see problem statement) will have. These values are summarized in Table P10.18-1. It is easily verified that the sum of the numbers on the left column of the table is N^2.

Table P10.18-1

No. of Points	Value
$N\left(\frac{N}{2}-1\right)$	0
2	2/9
$N-2$	3/9
4	4/9
$3N-8$	6/9
$(N-2)\left(\frac{N}{2}-2\right)$	1

A histogram is easily constructed from the entries in this table. A similar (tedious, but not difficult) procedure yields the results shown in Table P10.18-2 for the checkerboard image.

Table P10.18-2

No. of Points	Value
$\frac{N^2}{2}-14N+98$	0
28	2/9
$14N-224$	3/9
128	4/9
98	5/9
$16N-256$	6/9
$\frac{N^2}{2}-16N+128$	1

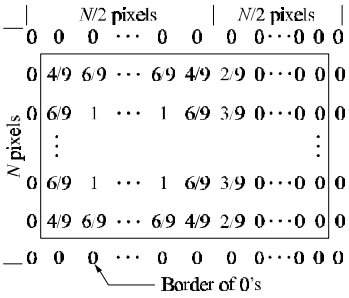


Figure P10.18

Problem 10.19

The gray level profile of one row of the image is shown in Fig. P10.19(a), and the

histogram of the image is shown in Fig. P10.19(b). The gray level profile of one row in the wedge image is shown in Fig. P10.19(c), and its histogram is shown in Fig. P10.19(d). The gray level profile of a row in the product image is shown in Fig. P10.19(e). The histogram of the product is shown in Fig. P10.19(f).

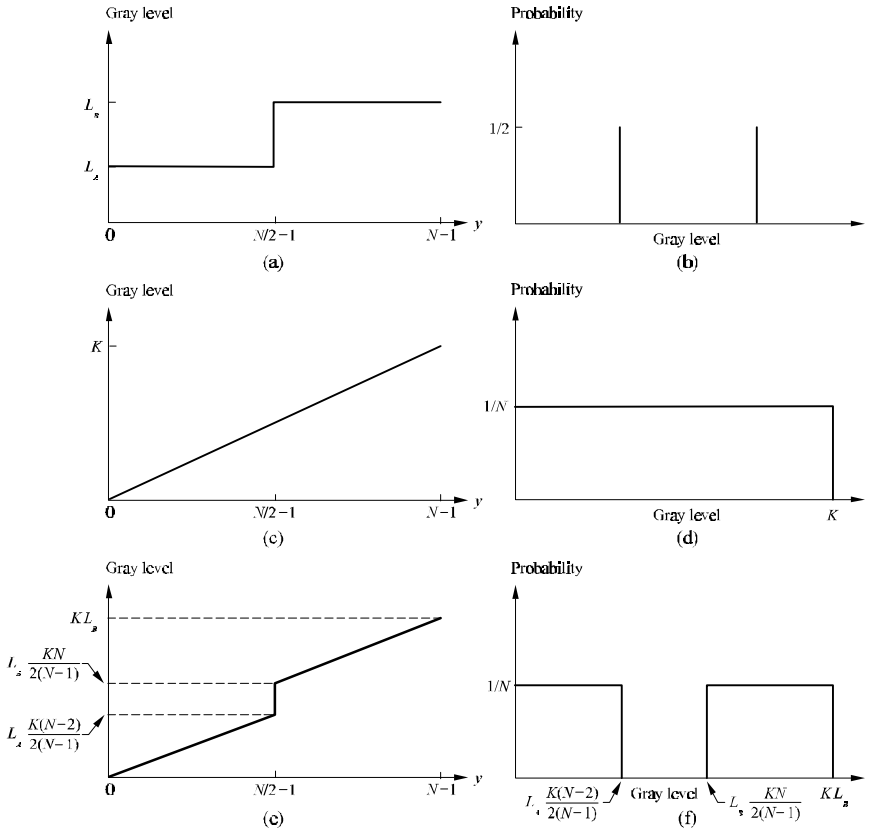


Figure P10.19

Problem 10.20

(a) $A_1 = A_2$ and $\sigma_1 = \sigma_2 = \sigma$, which makes the two modes identical. If the number of samples is not large, convergence to a value at or near the mid point between the two means also requires that a clear valley exist between the two modes. We can guarantee this by assuming that $\sigma << (m_1 + m_2)/2$.

(b) That this condition cannot happen if $A2 \neq 0$. This is easily established by starting the algorithm with an initial value less than m_1. Even if the right mode associated with m_2 is much smaller in size (e.g., $A1 >> A2$ and $\sigma_1 >> \sigma_2$) the average value of the

region to the left of the starting threshold will be smaller than the average of the region to the right because the modes are symmetrical about their mean, and the mode associated with m_2 will bias the data to the right. Thus, the next iterative step will bring the value of the threshold closer to m_1, and eventually to the right of it. This analysis assumes that enough points are available in order to avoid pathological cases in which the algorithm can get "stuck" due to insufficient data that truly represents the shapes assumed in the problem statement.

(c) $\sigma_2 >> \sigma_1$. This will "draw" the threshold toward m_2 during iteration.

Problem 10.21

The illumination function is a bell-shaped surface with its center at $(500, 500)$. The value of illumination at this point is 1, and it decreases radially from there. Draw a series of concentric circles about point $(500, 500)$ so that the value of $i(x, y)$ at each circle is 0.1 less than the circle before. Any two points within these two circles do no differ by more than 10% in illumination. Segment (threshold) the region between adjacent circles. If the distance between circles is greater than 10 pixels, then we are told that the segmentation will be correct. That is, proper segmentation of areas greater than 10×10 pixels is guaranteed in the problem statement, as long as illumination between any two points does not differ by more than 10%. Regions of 10×10 pixels will fit between concentric circles that are more than 10 pixels apart. If the distance between circles is less than 10 pixels, then the segmentation is not guaranteed to be perfect. But, there is nothing that can be done about that because changes in illumination are determined by the illumination function, which is given.

Problem 10.22

From the figure in the problem statement,

$$p_1(z) = \begin{cases} 0 & z < 1 \\ \frac{1}{2}z - \frac{1}{2} & 1 \leq z \leq 3 \\ 0 & z > 3 \end{cases}$$

and

$$p_2(z) = \begin{cases} 0 & z < 0 \\ -\frac{1}{2}z + 1 & 0 \leq z \leq 2 \\ 0 & z > 2 \end{cases} .$$

The optimum threshold is the value $z = T$ for which $P_1 p_1(T) = P_2 p_2(T)$. In this case $P_1 = P_2$, so

$$\frac{1}{2}T - \frac{1}{2} = -\frac{1}{2}T + 1$$

from which we get $T = 1.5$.

Problem 10.23

Keeping the same sense of directions as in Problem 10.22, let $p_2(z)$ be the probability density function given in the problem statement. The key in solving the problem is to recognize that the direction of the "tail" of the Rayleigh function can be reversed as follows:

$$p_1(z) = \begin{cases} \frac{2}{d}(-z+c)e^{-(-z+c)^2/d} & z \le c \\ 0 & z > c \end{cases}.$$

Then, the optimum threshold, T, is found by solving the following equation for T:

$$P_1 p_1(T) = P_2 p_2(T).$$

Substituting the density functions into these equations yields

$$P_1 \frac{2}{d}(-T+c)e^{-(-T+c)^2/d} = P_2 \frac{2}{b}(T-a)e^{-(T-a)^2/d}$$

which must be solved for T to find the optimum threshold. With the exception of some possible additional reformatting (like taking the natural log), this is as far as we normally expect students to carry this problem. However, it is important for the student to state that the solution is valid only in the range $a \le T \le c$.

Problem 10.24

From Eq. (10.3-10),

$$P_1 p_1(T) = P_2 p_2(T).$$

Taking the ln of both sides yields

$$\ln P_1 + \ln p_1(T) = \ln P_2 + \ln p_2(T).$$

But

$$p_1(T) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(T-\mu_1)^2}{2\sigma_1^2}}$$

and

$$p_2(T) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(T-\mu_2)^2}{2\sigma_2^2}}$$

so it follows that

$$\ln P_1 + \ln \frac{1}{\sqrt{2\pi}\sigma_1} - \frac{(T-\mu_1)^2}{2\sigma_1^2} = \ln P_2 + \ln \frac{1}{\sqrt{2\pi}\sigma_2} - \frac{(T-\mu_2)^2}{2\sigma_2^2}$$

$$\ln P_1 - \ln \sigma_1 - \frac{(T-\mu_1)^2}{2\sigma_1^2} - \ln P2 + \ln \sigma_2 + \frac{(T-\mu_2)^2}{2\sigma_2^2} = 0$$

$$\ln \frac{P_1}{P_2} + \ln \frac{\sigma_1}{\sigma_2} - \frac{1}{2\sigma_1^2}(T^2 - 2\mu_1 T + \mu_1^2) + \frac{1}{2\sigma_2^2}(T^2 - 2\mu_2 T + \mu_2^2) = 0$$

$$\ln \frac{\sigma_2 P_1}{\sigma_1 P_2} + T^2\left(\frac{1}{2\sigma_2^2} - \frac{1}{2\sigma_1^2}\right) + T\left(\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2}\right) + \left(\frac{\mu_2^2}{2\sigma_2^2} - \frac{\mu_1^2}{2\sigma_1^2}\right) = 0.$$

From this expression we get

$$AT^2 + BT + C = 0$$

with

$$\begin{aligned} A &= (\sigma_1^2 - \sigma_2^2) \\ B &= 2(\sigma_2^2\mu_1 - \sigma_1^2\mu_2) \end{aligned}$$

and

$$C = \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\sigma_2^2 \ln \frac{\sigma_2 P_1}{\sigma_1 P_2}.$$

Problem 10.25

If $\sigma_1 = \sigma_2 = \sigma$, then $A = 0$ in Eq. (10.3-12) and we have to solve the equation

$$BT + C = 0$$

with

$$B = 2\sigma^2(\mu_1 - \mu_2)$$

and

$$C = \sigma^2(\mu_2^2 - \mu_1^2) + 2\sigma^4 \ln \frac{P_1}{P_2}.$$

Substituting and cancelling terms gives

$$2(\mu_1 - \mu_2)T - (\mu_1 + \mu_2)(\mu_1 - \mu_2) + 2\sigma^2 \ln \frac{P_1}{P_2} = 0$$

or

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln \frac{P_1}{P_2}.$$

Problem 10.26

The simplest solution is to use the given means and standard deviations to form two Gaussian probability density functions, and then to use the optimum thresholding approach discussed in Section 10.3.5 (in particular, see Eqs. (10.3-11) through (10.3-13). The probabilities P_1 and P_2 can be estimated by visual analysis of the images (i.e., by determining the relative areas of the image occupied by objects and background). It is clear by looking at the image that the probability of occurrence of object points is less than that of background points. Alternatively, an automatic estimate can be obtained by

thresholding the image into points with values greater than 200 and less than 110 (see problem statement). Using the given parameters, the results would be good estimates of the relative probability of occurrence of object and background points due to the separation between means, and the relatively tight standard deviations. A more sophisticated approach is to use the Chow-Kaneko procedure discussed in Section 10.3.5.

Problem 10.27

Let m_1 and m_2 denote the mean gray level of objects and background, respectively, and let σ_1 and σ_2 denote the corresponding standard deviations (see the problem statement for specific values). We note that $\pm 2\sigma_2$ about the mean background level gives a range of gray level values from 80 to 140, and that $\pm 2\sigma_1$ about the mean intensity of the objects gives a range of 120 to 280, so a reasonable separation exists between the two gray level populations. Choosing $m_1 = 200$ as the seed value is quite adequate. Regions are then grown by appending to a seed any point that is 8-connected to any point previously appended to that seed, and whose gray level is $m_1 \pm 2\sigma_1$.

Problem 10.28

The region splitting is shown in Fig. P10.28(a). The corresponding quadtree is shown in Fig. P10.28(b).

Problem 10.29

(a) The elements of $T[n]$ are the coordinates of points in the image below the plane $g(x, y) = n$, where n is an integer that represents a given step in the execution of the algorithm. Since n never decreases, the set of elements in $T[n-1]$ is a subset of the elements in $T[n]$. In addition, we note that all the points below the plane $g(x, y) = n - 1$ are also below the plane $g(x, y) = n$, so the elements of $T[n]$ are never replaced. Similarly, $C_n(M_i)$ is formed by the intersection of $C(M_i)$ and $T[n]$, where $C(M_i)$ (whose elements never change) is the set of coordinates of *all* points in the catchment basin associated with regional minimum M_i. Since the elements of $C(M_i)$ never change, and the elements of $T[n]$ are never replaced, it follows that the elements in $C_n(M_i)$ are never replaced either. In addition, we see that $C_{n-1}(M_i) \subseteq C_n(M_i)$.

(b) This part of the problem is answered by the same argument as in (a). Since (1) n

always increases; (2) the elements of neither $C_n(M_i)$ nor $T[n]$ are ever replaced; and (3) $T[n-1] \subseteq T[n]$ and $C_{n-1}(M_i) \subseteq C_n(M_i)$, it follows that the number of elements of both $C_n(M_i)$ and $T[n]$ either increases or remains the same.

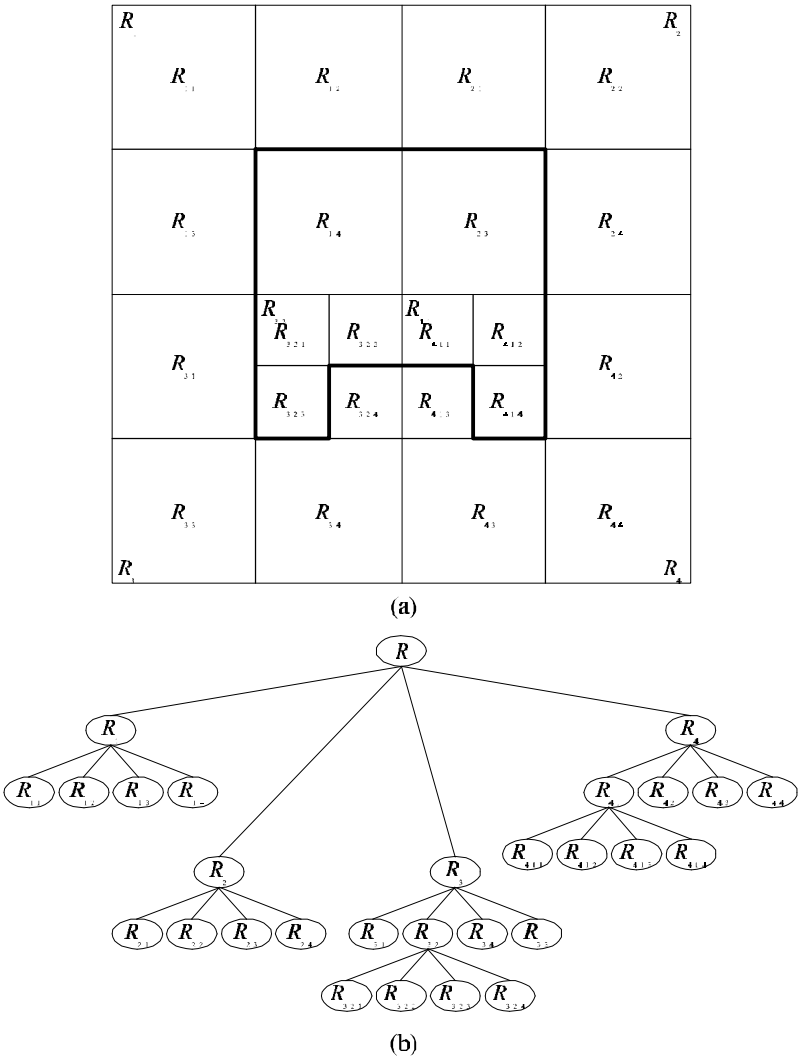


Figure P10.28

Problem 10.30

Using the terminology of the watershed algorithm, a break in a boundary between two catchment basins would cause water between the two basins to merge. However, the

heart of the algorithm is to build a dam higher than the highest gray level in the image any time a break in such boundaries occurs. Since the entire topography is enclosed by such a dam, dams are built any time there is a break that causes water to merge between two regions, and segmentation boundaries are precisely the tops of the dams, it follows that the watershed algorithm always produces closed boundaries between regions.

Problem 10.31

The first step in the application of the watershed segmentation algorithm is to build a dam of height $max + 1$ to prevent the rising water from running off the ends of the function, as shown in Fig. P10.31(b). For an image function we would build a box of height $max + 1$ around its border. The algorithm is initialized by setting $C[1] = T[1]$. In this case, $T[1] = \{g(2)\}$, as shown in Fig. P10.31(c) (note the water level). There is only one connected component in this case: $Q[1] = \{q_1\} = \{g(2)\}$.

Next, we let $n = 2$ and, as shown in Fig. P10.31(d), $T[2] = \{g(2), g(14)\}$ and $Q[2] = \{q_1; q_2\}$, where, for clarity, different connected components are separated by semicolons. We start construction of $C[2]$ by considering each connected component in $Q[2]$. When $q = q_1$, the term $q \cap C[1]$ is equal to $\{g(2)\}$, so condition 2 is satisfied and, therefore, $C[2] = \{g(2)\}$. When $q = q_2$, $q \cap C[1] = \emptyset$ (the empty set) so condition 1 is satisfied and we incorporate q in $C[2]$, which then becomes $C[2] = \{g(2); g(14)\}$ where, as above, different connected components are separated by semicolons.

When $n = 3$ [Fig. P10.31(e)], $T[3] = \{2, 3, 10, 11, 13, 14\}$ and $Q[3] = \{q_1; q_2; q_3\} = \{2, 3; 10, 11; 13, 14\}$ where, in order to simplify the notation we let k denote $g(k)$. Proceeding as above, $q_1 \cap C[2] = \{2\}$ satisfies condition 2, so q_1 is incorporated into the new set to yield $C[3] = \{2, 3; 14\}$. Similarly, $q_2 \cap C[2] = \emptyset$ satisfies condition 1 and $C[3] = \{2, 3; 10, 11; 14\}$. Finally, $q_3 \cap C[2] = \{14\}$ satisfies condition 2 and $C[3] = \{2, 3; 10, 11; 13, 14\}$. It is easily verified that $C[4] = C[3] = \{2, 3; 10, 11; 13, 14\}$.

When $n = 5$ [Fig. P10.31(f)], we have, $T[5] = \{2, 3, 5, 6, 10, 11, 12, 13, 14\}$ and $Q[5] = \{q_1; q_2; q_3\} = \{2, 3; 5, 6; 10, 11, 12, 13, 14\}$ (note the merging of two previously distinct connected components). Is is easily verified that $q_1 \cap C[4]$ satisfies condition 2 and that $q_2 \cap C[4]$ satisfied condition 1. Proceeding with these two connected components exactly as above yields $C[5] = \{2, 3; 5, 6; 10, 11; 13, 14\}$ up to this point. Things get more interesting when we consider q_3. Now, $q_3 \cap C[4] = \{10, 11; 13, 14\}$ which, since it contains two connected components of $C[4]$ satisfies condition 3. As mentioned previously, this is an indication that water from two different basins has merged and a

dam must be built to prevent this. Dam building is nothing more than separating q_3 into the two original connected components. In this particular case, this is accomplished by the dam shown in Fig. P10.31(g), so that now $q_3 = \{q_{31}; q_{32}\} = \{10, 11; 13, 14\}$. Then, $q_{31} \cap C[4]$ and $q_{32} \cap C[4]$ each satisfy condition 2 and we have the final result for $n = 5$, $C[5] = \{2, 3; 5, 6; 10, 11; 13; 14\}$.

Continuing in the manner just explained yields the final segmentation result shown in Fig. P10.31(h), where the "edges" are visible (from the top) just above the water line. A final post-processing step would remove the outer dam walls to yield the inner edges of interest.

Problem 10.32

With reference to Eqs. (10.6-4) and (10.6-3), we see that comparing the negative ADI against a positive, rather than a negative, threshold would yield the image negative of the positive ADI. The result is shown in the left of Fig. P10.32. The image on the right is the positive ADI from Fig. 10.49(b). We have included it here for convenience in making the comparison.

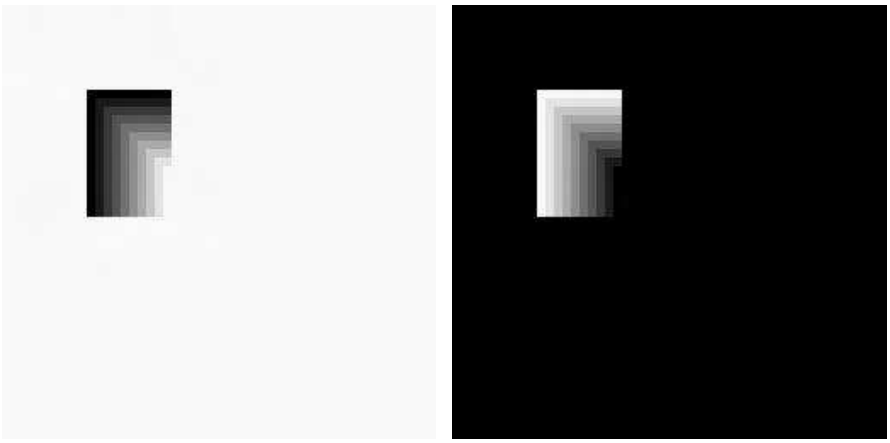


Figure P10.32

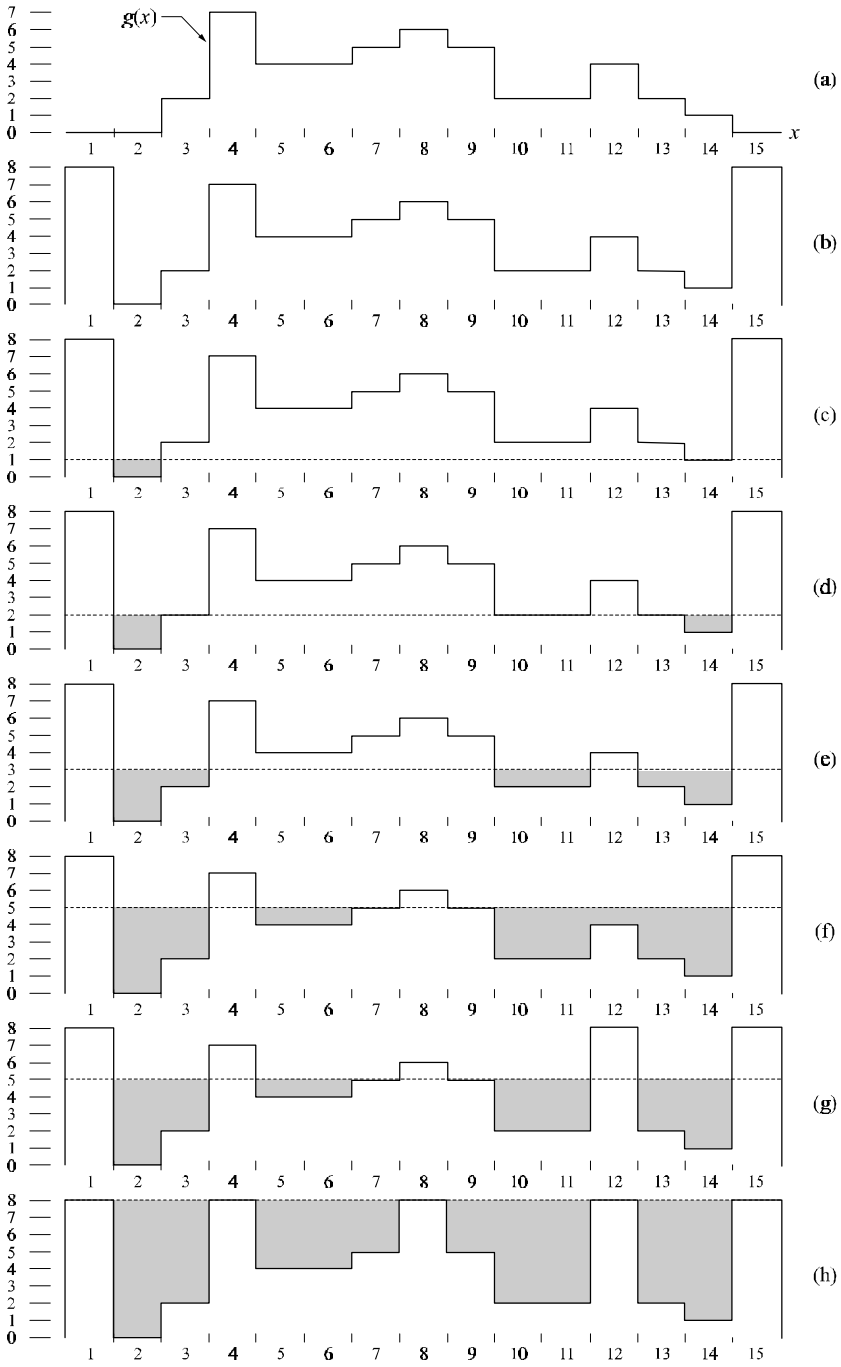


Figure P10.31

Problem 10.33

(a) True, assuming that the threshold is not set larger than all the differences encountered as the object moves. The easiest way to see this is to draw a simple reference image, such as the white rectangle on a black background. Let that rectangle be the object that moves. Since the absolute ADI image value at any location is the absolute difference between the reference and the new image, it is easy to see that as the object enters areas that are background in the reference image, the absolute difference will change from zero to nonzero at the new area occupied by the moving object. Thus, as long as the object moves the dimension of the absolute ADI will grow.

(b) True. The positive ADI is stationary and equal to the dimensions of the moving object because the differences between the reference and the moving object never exceed the threshold in areas that are background in the reference image (assuming as Eq. (10.6-3) that the background has lower values than the object).

(c) True. From Eq. (10.6-4), we see that difference between the background and the object will always be negative (assuming as in Eq. (10.6-4) that the gray levels in the object exceed the value of the background). Assuming also that the differences are more negative than the threshold, we see for the same reason as in (a) that all new background areas occupied by the moving object will have nonzero counts, thus increasing the dimension of the nonzero entries in the negative ADI (keep in mind that the values in this image are counts).

Problem 10.34

Consider first the fact that motion in the x-direction is zero. When all components of an image are stationary, $g_x(t, a_1)$ is a constant, and its Fourier transform yields an impulse at the origin. Therefore, Fig. 10.53 would now consists of a single impulse at the origin. The other two peaks shown in the figure would no longer be present. To handle the motion in the positive y-direction and its change opposite direction, recall that the Fourier transform is a linear process, so we can use superposition to obtain a solution. The first part of motion is in the positive y-direction at 1 pixel/frame. This is the same as in Example 10.2, so the peaks corresponding to this part of the motion are the same as the ones shown in Fig. 10.54. The reversal of motion is instantaneous, so the 33rd frame would show the object traveling in exactly the opposite direction. To handle this, we simply change a_2 to $-a_2$ in Eq. (10.6-7). Based on the discussion in

connection with Eq. (10.6-5), all this change would do is produce peaks at frequencies $u = -a_2 v_2$ and $K + a_2 v_2$. From Example 10.21 we know that the value of a_2 is 4. From the problem statement, we know that $v_2 = 1$ and $K = 32$. Thus, we have two new peaks added to Fig. 10.54: one at $u = -4$ and the other at $u = 36$. As noted above, the original peaks correspond to the motion in the positive y-direction given in the problem statement, which is the same as in Example 10.21. Note that the frame count was restarted from 0 to 31 with the change in direction.

Problem 10.35

(a) It is given that 10% of the image area in the horizontal direction is occupied by a bullet that is 2.5 cm long. Since the imaging device is square (256×256 elements) the camera looks at an area that is 25 cm $\times$ 25 cm, assuming no optical distortions. Thus, the distance between pixels is 25/256=0.098 cm/pixel. The maximum speed of the bullet is 1000 m/sec = 100,000 cm/sec. At this speed, the bullet will travel $100,000/0.98 = 1.02 \times 10^6$ pixels/sec. It is required that the bullet not travel more than one pixel during exposure. That is, (1.02×10^6 pixels/sec) $\times$ K sec $\leq$ 1 pixel. So, $K \leq 9.8 \times 10^{-7}$ sec.

b) The frame rate must be fast enough to capture at least two images of the bullet in successive frames so that the speed can be computed. If the frame rate is set so that the bullet cannot travel a distance longer (between successive frames) than one half the width of the image, then we have the cases shown in Fig. P10.35. In cases A and E we get two shots of the entire bullet in frames t_2 and t_3 and t_1 and t_2, respectively. In the other cases we get partial bullets. Although these cases could be handled with some processing (e.g., by determining size, leading and trailing edges, and so forth) it is possible to guarantee that at least two complete shots of every bullet will be available by setting the frame rate so that a bullet cannot travel more than one half the width of the frame, minus the length of the bullet. The length of the bullet in pixels is (2.5 cm)/(0.098 cm/pixel) $\approx$ 26 pixels. One half of the image frame is 128 pixels, so the maximum travel distance allowed is 102 pixels. Since the bullet travels at a maximum speed of 1.02×10^6 pixels/sec, the minimum frame rate is $1.02 \times 10^6/102 = 10^4$ frames /sec.

(c) In a flashing situation with a reflective object, the images will tend to be dark, with the object shining brightly. The techniques discussed in Section 10.6.1 would then be quite adequate.

(d) First we have to determine if a partial or whole image of the bullet has been obtained. After the pixels corresponding to the object have been identified using motion segmen-

tation, we determine if the object runs into the left boundary (see the solution to Problem 9.27) regarding a method for determining if a binary object runs into the boundary of an image). If it does, we look at the next two frames, with the assurance that a complete image of the bullet has been obtained in each because of the frame rate in (b). If the object does not run into the left boundary, we are similarly assured of two full shots in two of the three frames. We then compute the centroid of the object in each image and count the number of pixels between the centroids. Since the distance between pixels and the time between frames are known, computation of the speed is a trivial problem. The principal uncertainty in this approach is how well the object is segmented. However, since the images are of the same object in basically the same geometry, consistency of segmentation between frames can be expected.

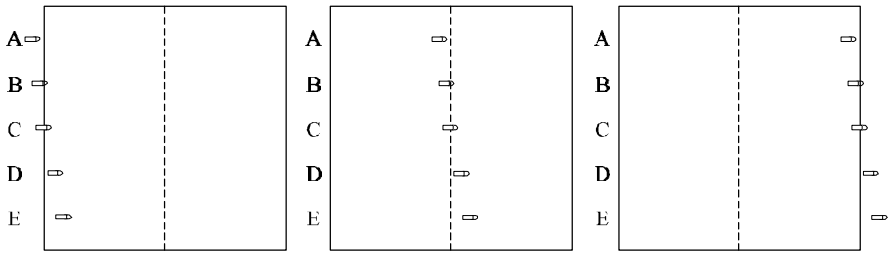


Figure P10.35

11 Problem Solutions

Problem 11.1

(a) The key to this problem is to recognize that the value of every element in a chain code is relative to the value of its predecessor. The code for a boundary that is traced in a consistent manner (e.g., clockwise) is a unique circular set of numbers. Starting at different locations in this set does not change the structure of the circular sequence. Selecting the smallest integer as the starting point simply identifies the same point in the sequence. Even if the starting point is not unique, this method would still give a unique sequence. For example, the sequence 101010 has three possible starting points, but they all yield the same smallest integer 010101.

(b) Code: 11076765543322. The starting point is 0, yielding the sequence

$$07676554332211.$$

Problem 11.2

(a) The first difference only counts the number of directions that separate adjacent elements of the code. Since the counting process is independent of direction, the first difference is independent of boundary rotation. (It is worthwhile to point out to students that the assumption here is that rotation does not change the code itself).

(b) Code: 010103030323232212111. Difference: 3131331313031313031300. (Note that the code was treated as a circular sequence, so the first element of the difference is the transition between the last and first element of the code, as explained in the text).

Problem 11.3

(a) The rubber-band approach forces the polygon to have vertices at every inflection of the cell wall. That is, the locations of the vertices are fixed by the structure of the

inner and outer walls. Since the vertices are joined by straight lines, this produces the minimum-perimeter polygon for any given wall configuration.

(b) If a corner of a cell is centered at a pixel on the boundary, and the cell is such that the rubber band is tightened on the opposite corner, we would have a situation as shown in Fig. P11.3. Assuming that the cell is of size $d \times d$, the maximum difference between the pixel and the boundary in that cell is $\sqrt{2}d$. If cells are centered on pixels, the maximum difference is $(\sqrt{2}d)/2$.

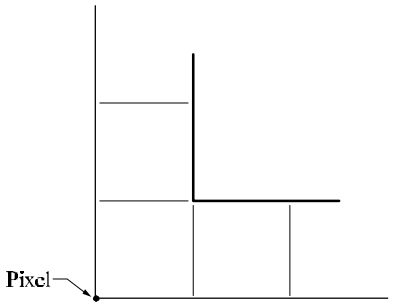


Figure P11.3

Problem 11.4

(a) The resulting polygon would contain all the boundary pixels.

(b) Actually, in both cases the resulting polygon would contain all the boundary pixels.

Problem 11.5

(a) The solution is shown in Fig. P11.5(b). (b) The solution is shown in Fig. P11.5(c).

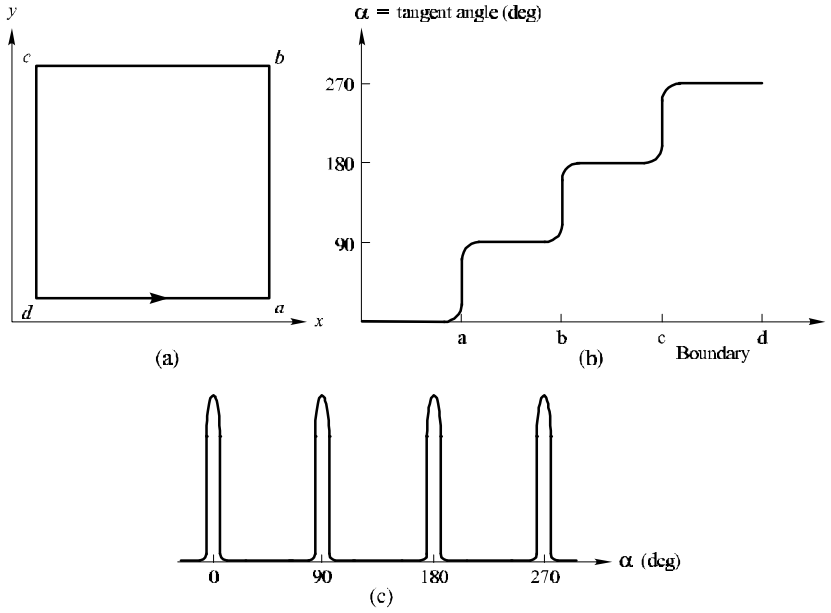


Figure P11.5

Problem 11.6

(a) From Fig. P11.6(a), we see that the distance from the origin to the triangle is given by

$$\begin{aligned} r(\theta) &= \frac{D_0}{\cos\theta} \qquad 0^\circ \le \theta < 60^\circ \\ &= \frac{D_0}{\cos(120^\circ - \theta)} \qquad 60^\circ \le \theta < 120^\circ \\ &= \frac{D_0}{\cos(180^\circ - \theta)} \qquad 120^\circ \le \theta < 180^\circ \\ &= \frac{D_0}{\cos(240^\circ - \theta)} \qquad 180^\circ \le \theta < 240^\circ \\ &= \frac{D_0}{\cos(300^\circ - \theta)} \qquad 240^\circ \le \theta < 300^\circ \\ &= \frac{D_0}{\cos(360^\circ - \theta)} \qquad 300^\circ \le \theta < 360^\circ \end{aligned}$$

where D_0 is the perpendicular distance from the origin to one of the sides of the triangle, and $D = D_0/\cos(60^\circ) = 2D_0$. Once the coordinates of the vertices of the triangle are given, determining the equation of each straight line is a simple problem, and D_0 (which is the same for the three straight lines) follows from elementary geometry.

(b) From Fig. P11.6(b),

$$\begin{aligned}
r(\theta) &= \frac{B}{2\cos\theta} && 0^\circ \le \theta < \varphi \\
&= \frac{A}{2\cos(90^\circ - \theta)} && \varphi \le \theta < 90^\circ \\
&= \frac{A}{2\cos(\theta - 90^\circ)} && 90^\circ \le \theta < (180^\circ - \varphi) \\
&= \frac{B}{2\cos(180^\circ - \theta)} && (180^\circ - \varphi) \le \theta < 180^\circ \\
&= \frac{B}{2\cos(\theta - 180^\circ)} && 180^\circ \le \theta < 180^\circ + \varphi \\
&= \frac{A}{2\cos(270^\circ - \theta)} && 180^\circ + \varphi \le \theta < 270^\circ \\
&= \frac{A}{2\cos(\theta - 270)} && 270^\circ \le \theta < 270^\circ + \varphi \\
&= \frac{B}{2\cos(360^\circ - \theta)} && 270^\circ + \varphi \le \theta < 360^\circ.
\end{aligned}$$

where $\varphi = \tan^{-1}(A/B)$.

(c) The equation of the ellipse in Fig. P11.6(c) is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1.$$

We are interested in the distance from the origin to an arbitrary point (x, y) on the ellipse. In polar coordinates,

$$x = r\cos\theta$$

and

$$y = r\sin\theta$$

where is the distance from the origin to (x, y):

$$r = \sqrt{x^2 + y^2}.$$

Substituting into the equation of the ellipse we obtain

$$\frac{r^2\cos^2\theta}{a^2} + \frac{r^2\sin^2\theta}{b^2} = 1$$

from which we obtain the desired result:

$$r(\theta) = \frac{1}{\left[\left(\dfrac{\cos\theta}{a}\right)^2 + \left(\dfrac{\sin\theta}{b}\right)^2\right]^{1/2}}.$$

When $b = a$, we have the familiar equation of a circle, $r(\theta) = a$, or $x^2 + y^2 = a^2$.

Plots of the three signatures just derived are shown in Fig. P11.6(d)-(f).

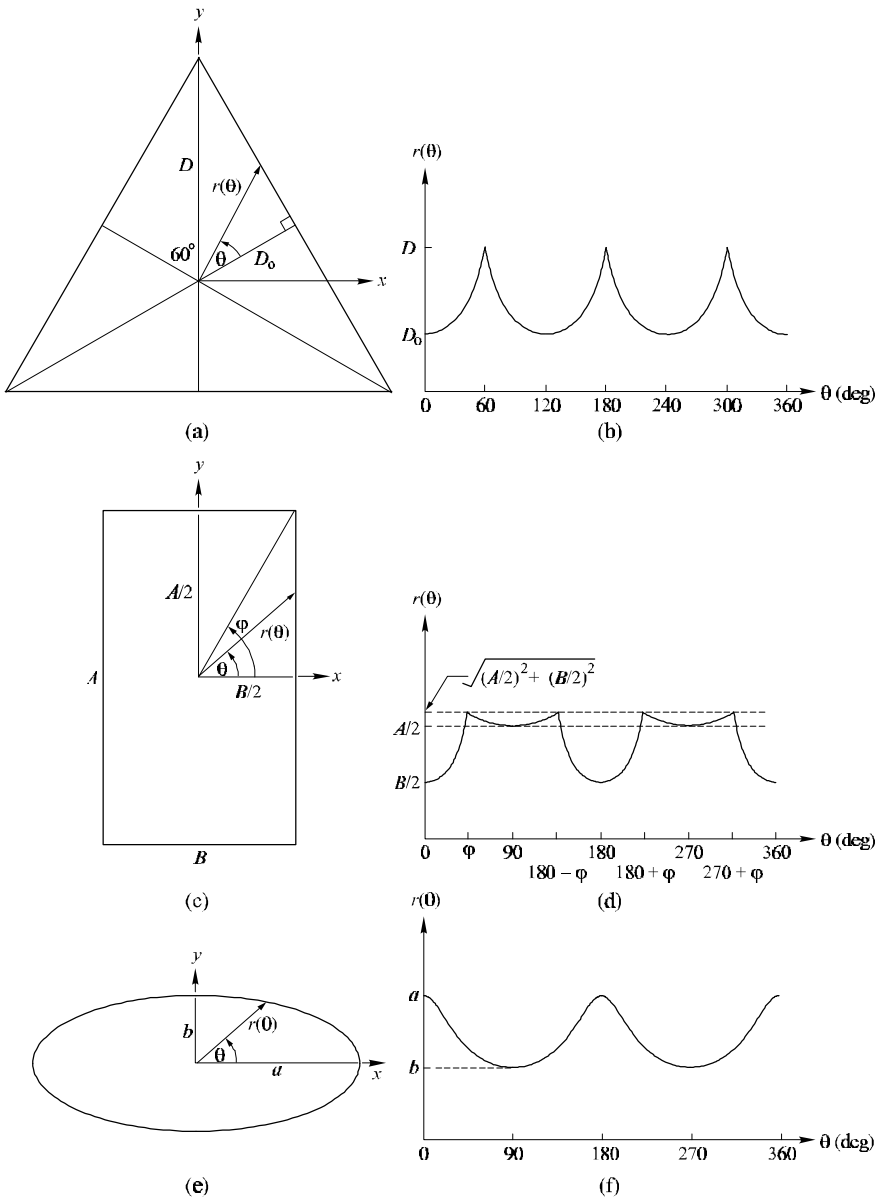


Figure P11.6

Problem 11.7

The solutions are shown in Fig. P11.7.

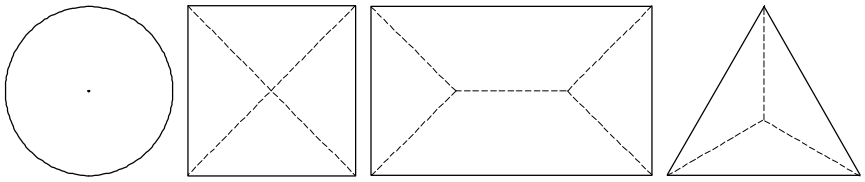


Figure P11.7

Problem 11.8

(a) In the first case, $N(p) = 5$, $S(p) = 1$, $p_2 \cdot p_4 \cdot p_6 = 0$, and $p4 \cdot p6 \cdot p8 = 0$, so Eq. (11.1-1) is satisfied and p is flagged for deletion. In the second case, $N(p) = 1$, so Eq. (11.1-1) is violated and p is left unchanged. In the third case $p2 \cdot p4 \cdot p6 = 1$ and $p4 \cdot p6 \cdot p8 = 1$, so conditions (c) and (d) of Eq. (11.1-1) are violated and p is left unchanged. In the forth case $S(p) = 2$, so condition (b) is violated and p is left unchanged.

(b) In the first case $p2 \cdot p6 \cdot p8 = 1$ so condition (d') in Eq. (11.1-3) is violated and p is left unchanged. In the second case $N(p) = 1$ so p is left unchanged. In the third case (c') and (d') are violated and p is left unchanged. In the fourth case $S(p) = 2$ and p is left unchanged.

Problem 11.9

(a) The result is shown in Fig. 11.9(b). (b) The result is shown in Fig. 11.9(c).

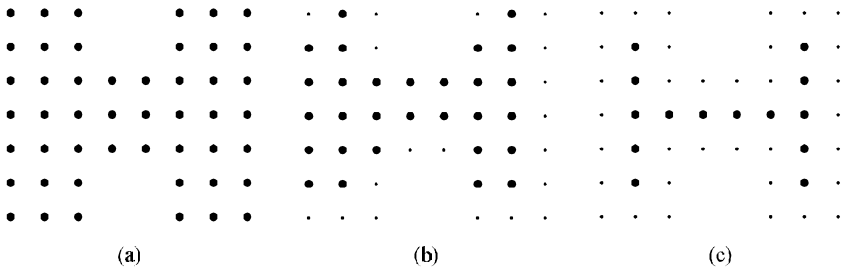


Figure P11.9

Problem 11.10

(a) The number of symbols in the first difference is equal to the number of segment

primitives in the boundary, so the shape order is 12.

(b) Starting at the top left corner,

Chain code:	000332123211
Difference:	300303311330
Shape no.:	003033113303

Problem 11.11

With reference to the discussion in Section 4.6.1, the DFT can be real only if the data sequence is conjugate symmetric. Only contours that are symmetric with respect to the origin have this property. The axis system of Fig. 11.13 would have to be set up so that this condition is satisfied for symmetric figures. This can be accomplished by placing the origin at the center of gravity of the contour.

Problem 11.12

The mean is sufficient.

Problem 11.13

Two ellipses with different, say, major axes, have signatures with the same mean and third statistical moment descriptors (both due to symmetry) but different second moment (due to spread).

Problem 11.14

This problem can be solved by using two descriptors: holes and the convex deficiency (see Section 9.5.4 regarding the convex hull and convex deficiency of a set). The decision making process can be summarized in the form of a simple decision, as follows: If the character has two holes, it is an 8. If it has one hole it is a 0 or a 9. Otherwise, it is a 1 or an X. To differentiate between 0 and 9 we compute the convex deficiently. The presence of a ”significant” deficiency (say, having an area greater than 20% of the area of a rectangle that encloses the character) signifies a 9; otherwise we classify the character as a 0. We follow a similar procedure to separate a 1 from an X. The presence of a convex deficiency with four components whose centroids are located approximately in

the North, East, West, and East quadrants of the character indicates that the character is an X. Otherwise we say that the character is a 1. This is the basic approach. Implementation of this technique in a real character recognition environment has to take into account other factors such as multiple "small" components in the convex deficiency due to noise, differences in orientation, open loops, and the like. However, the material in Chapters 3, 9 and 11 provide a solid base from which to formulate solutions.

Problem 11.15

We can use the position operator P: "$2m$ pixels to the right and $2m$ pixels below." Other possibilities are P: "2m pixels to the right," and P: "2m pixels below." The first choice is better in terms of retaining the "flavor" of a checkerboard.

Problem 11.16

(a) The image is

$$\begin{matrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{matrix}.$$

Let $z_1 = 0$ and $z_2 = 1$. Since there are only two gray levels the matrix $\mathbf{A}$ is of order 2×2. Element a_{11} is the number of pixels valued 0 located one pixel to the right of a 0. By inspection, $a_{11} = 0$. Similarly, $a_{12} = 10$, $a_{21} = 10$, and $a_{22} = 0$. The total number of pixels satisfying the predicate P is 20, so

$$\mathbf{C} = \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix}.$$

(b)In this case, a_{11} is the number of 0's two pixels to the right of a pixel valued 0. By inspection, $a_{11} = 8$. Similarly, $a_{12} = 0$, $a_{21} = 0$, and $a_{22} = 7$. The number of pixels satisfying P is 15, so

$$\mathbf{C} = \begin{bmatrix} 8/15 & 0 \\ 0 & 7/15 \end{bmatrix}.$$

Problem 11.17

When assigning this problem, the Instructor may wish to point the student to the review

of matrices and vectors in the book web site.

From Eq. (11.4-6),

$$\mathbf{y} = \mathbf{A}(\mathbf{x} - \mathbf{m_x}).$$

Then,

$$\begin{aligned} \mathbf{m_y} &= E\{\mathbf{y}\} = E\{\mathbf{A}(\mathbf{x} - \mathbf{m_x})\} \\ &= \mathbf{A}[E\{\mathbf{x}\} - E\{\mathbf{m_x}\}] \\ &= \mathbf{A}[\mathbf{m_x} - \mathbf{m_x}] \\ &= \mathbf{0}. \end{aligned}$$

This establishes the validity of Eq. (11.4-7).

To prove the validity of Eq. (11.4-8), we start with the definition of the covariance matrix given in Eq. (11.4-3):

$$\mathbf{C_y} = E\{(\mathbf{y} - \mathbf{m_y})(\mathbf{y} - \mathbf{m_y})^T\}.$$

Since $\mathbf{m_y} = \mathbf{0}$, it follows that

$$\begin{aligned} Cy &= E\{yyT\} \\ &= E\{[\mathbf{A}(\mathbf{x} - \mathbf{m_x})][\mathbf{A}(\mathbf{x} - \mathbf{m_x})]^T\} \\ &= \mathbf{A}\, E\{(\mathbf{x} - \mathbf{m_x})(\mathbf{x} - \mathbf{m_x})^T\}\mathbf{A}^T \\ &= \mathbf{A}\mathbf{C_x}\mathbf{A}^T. \end{aligned}$$

Showing the validity of Eq. (11.4-9) is a little more complicated. We start by noting that covariance matrices are real and symmetric. From basic matrix algebra, it is known that a real symmetric matrix of order n has n linearly independent eigenvectors (which are easily orthonormalized by, say, the Gram-Schmidt procedure). The rows of matrix $\mathbf{A}$ are the orthonormal eigenvectors of $\mathbf{C_x}$. Then,

$$\begin{aligned} \mathbf{C_x}\mathbf{A}^T &= \mathbf{C_x}[\mathbf{e}_1, \mathbf{e}_2, \ldots \mathbf{e}_n] \\ &= [\mathbf{C_x}\mathbf{e}_1, \mathbf{C_x}\mathbf{e}_2, \ldots \mathbf{C_x}\mathbf{e}_n] \\ &= [\lambda_1\mathbf{e}_1, \lambda_2\mathbf{e}_2, \ldots, \lambda_n\mathbf{e}_n] \\ &= \mathbf{A}^T\mathbf{D} \end{aligned}$$

where use was made of the definition of an eigenvector (i.e., $\mathbf{C_x}\mathbf{e}_i = \lambda_i\mathbf{e}_i$) and $\mathbf{D}$ is a diagonal matrix composed of the eigenvalues of $\mathbf{C_x}$:

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}.$$

Premultiplying both sides of the preceding equation by matrix a gives

$$\begin{aligned}\mathbf{A}\mathbf{C}_\mathbf{x}\mathbf{A}^T &= \mathbf{A}\mathbf{A}^T\mathbf{D} \\ &= \mathbf{D}\end{aligned}$$

where we used the fact that $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$ because the rows of $\mathbf{A}$ are orthonormal vectors. Thus, since, $\mathbf{C}_\mathbf{y} = \mathbf{A}\mathbf{C}_\mathbf{x}\mathbf{A}^T$, we have shown that $\mathbf{C}_\mathbf{y}$ is a diagonal matrix which is produced by diagonalizing matrix $\mathbf{C}_\mathbf{x}$ using a transformation matrix composed of its eigenvectors. The eigenvalues of $\mathbf{C}_\mathbf{y}$ are seen to be the same as the eigenvalues of $\mathbf{C}_\mathbf{x}$. (Recall that the eigenvalues of a diagonal matrix are its diagonal terms). The fact that $\mathbf{C}_\mathbf{y}\mathbf{e}_i = \mathbf{D}\mathbf{e}_i = \lambda_i\mathbf{e}_i$ shows that the eigenvectors of $\mathbf{C}_\mathbf{y}$ are equal to the eigenvectors of $\mathbf{C}_\mathbf{x}$.

Problem 11.18

The mean square error, given by Eq. (11.4-12), is the sum of the eigenvalues whose corresponding eigenvectors are not used in the transformation. In this particular case, the four smallest eigenvalues are applicable (see Table 11.5), so the mean square error is

$$e_{ms} = \sum_{j=3}^{6} \lambda_j = 280.$$

The maximum error occurs when $K = 0$ in Eq. (11.4-12) which then is the sum of all the eigenvalues, or 4421 in this case. Thus, the error incurred by using the two eigenvectors corresponding to the largest eigenvalues is only 6.3 % of the total possible error.

Problem 11.19

This problem is similar to the previous one. The covariance matrix is of order 4096×4096 because the images are of size 64×64. It is given that the covariance matrix is the identity matrix, so all its 4096 eigenvalues are equal to 1. From Eq. (11.4-12), the mean square error is

$$\begin{aligned}e_{ms} &= \sum_{j=1}^{4096} \lambda_j - \sum_{i=1}^{2048} \lambda_i \\ &= 2048.\end{aligned}$$

Problem 11.20

When the boundary is symmetric about the both the major and minor axes and both axes

intersect at the centroid of the boundary.

Problem 11.21

A solution using the relationship "connected to," is shown in Fig. P11.21.

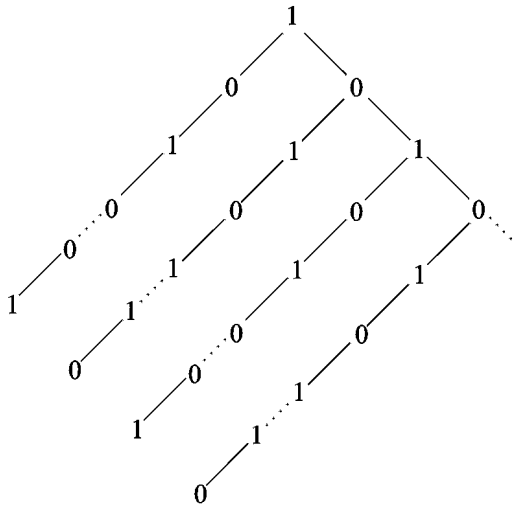


Figure P11.21

Problem 11.22

We can compute a measure of texture using the expression

$$R(x,y) = 1 - \frac{1}{1+\sigma^2(x,y)}$$

where $\sigma^2(x,y)$ is the gray-level variance computed in a neighborhood of (x,y). The size of the neighborhood must be sufficiently large so as to contain enough samples to have a stable estimate of the mean and variance. Neighborhoods of size 7×7 or 9×9 generally are appropriate for a low-noise case such as this.

Since the variance of normal wafers is known to be 400, we can obtain a normal value for $R(x,y)$ by using $\sigma^2 = 400$ in the above equation. An abnormal region will have a variance of about $(50)^2 = 2,500$ or higher, yielding a larger value of $R(x,y)$. The procedure then is to compute $R(x,y)$ at every point (x,y) and label that point as 0 if it is normal and 1 if it is not. At the end of this procedure we look for clusters of 1's using, for example, connected components (see Section 9.5.3 regarding computation of

connected components) . If the area (number of pixels) of any connected component exceeds 400 pixels, then we classify the sample as defective.

Problem 11.23

This problem has four major parts. (1) Detecting individual bottles in an image; (2) finding the top each bottle; (3) finding the neck and shoulder of each bottle; and (4) determining the level of the liquid in the region between the neck and the shoulder.

(1) *Finding individual bottles*. Note that the background in the sample image is much darker than the bottles. We assume that this is true in all images. Then, a simple way to find individual bottles is to find vertical black stripes in the image having a width determined by the average separation between bottles, a number that is easily computable from images representative of the actual setup during operation. We can find these stripes in various ways. One way is to smooth the image to reduce the effects of noise (we assume that, say, a 3×3 or 5×5 averaging mask is sufficient). Then, we run a horizontal scan line through the middle of the image. The low values in the scan line will correspond to the black or nearly black background. Each bottle will produce a significant rise and fall of gray level in the scan line for the width of the bottle. Bottles that are fully in the field of view of the camera will have a predetermined average width. Bottles that are only partially in the field of view will have narrower profiles, and can be eliminated from further analysis (but we need to make sure that the trailing incomplete bottles are analyzed in the next image; presumably, the leading partial bottle was already processed.).

(2) *Finding the top of each bottle*. Once the location of each (complete or nearly complete) bottle is determined, we again can use the contrast between the bottles and the background to find the top of the bottle. One possible approach is to compute a gradient image (sensitive only to horizontal edges) and look for a horizontal line near the top of the gradient image. An easier method is to run a vertical scan line through the center of the locations found in the previous step. The first major transition in gray level (from the top of the image) in the scan line will give a good indication of the location of the top of a bottle.

(3) *Finding the neck and shoulder of a bottle*. In the absence of other information, we assume that all bottles are of the same size, as shown in the sample image. Then, once we now where the top of a bottle is, the location of the neck and shoulder are known to be at a fixed distance from the bottle top.

(4) *Determining the level of the liquid*. The area defined by the bottom of the neck and the top of the shoulder is the only area that needs to be examined to determine acceptable vs. unacceptable fill level in a given bottle. In fact, As shown in the sample image, an area of a bottle that is void of liquid appears quite bright in an image, so we have various options. We could run a single vertical scan line again, but note that the bottles have areas of reflection that could confuse this approach. This computation is at the core of what this system is designed to do, so a more reliable method should be used. One approach is to threshold the area spanning a rectangle defined by the bottom of the neck, the shoulder, and sides of the bottle. Then, we count the number of white pixels above the midpoint of this rectangle. If this number is greater than a pre-established value, we know that enough liquid is missing and declare the bottle improperly filled. A slightly more sophisticated technique would be to actually find the level of the liquid. This would consist of looking for a horizontal edge in the region within the bottle defined by the sides of the bottle, the bottom of the neck, and a line passing midway between the shoulder and the bottom of the neck. A gradient/edge-linking approach, as described in Sections 10.1 and 10.2 would be suitable. Note however, that if no edge is found, the region is either filled (dark values in the region) or completely void of liquid (white, or near white values in the region). A computation to resolve these two possible conditions has to follow if the system fails to find an edge.

Problem 11.24

The key specification of the desired system is that it be able to detect individual bubbles. No specific sizes are given. We assume that bubbles are nearly round, as shown in the test image. One solution consists of (1) segmenting the image; (2) post-processing the result; (3) finding the bubbles and bubble clusters, and determining bubbles that merged with the boundary of the image; (4) detecting groups of touching bubbles; (5) counting individual bubbles; and (6) determining the ratio of the area occupied by all bubbles to the total image area.

(1) *Segmenting the image*. We assume that the sample image is truly representative of the class of images that the system will encounter. The image shown in the problem statement is typical of images that can be segmented by a global threshold. As shown by the histogram in Fig. P11.24, the gray level of the objects of interest is high on the gray scale. A simple adaptive threshold method for data that is that high on the scale is to choose a threshold equal to the mean plus a multiple of the standard deviation. We chose a threshold equal to $m + 2\sigma$, which, for the image in the problem statement, was

195. The segmented result is shown on the right of Fig. P11.24. Obviously this is not the only approach we could take, but this is a simple method that adapts to overall changes in intensity.

(2) *Post-processing*. As shown in the segmented image of Fig. P11.24, many of the bubbles appear as broken disks, or disks with interior black components. These are mostly due either to reflection (as in Fig. 9.16) or actual voids within a bubble. We could attempt to build a procedure to repair and/or fill the bubbles (as in Problem 9.23). However, this can turn into a computationally expensive process that is not warranted unless stringent measurement standards are required, a fact not mentioned in the problem statement. An alternative is to calculate, on the average (as determined from a set of sample images), the percentage of bubble areas that are filled with black or have black "bays" which makes their black areas merge with the background. Then, once the dimensions of each bubble (or bubble cluster) have been established, a correction factor based on area would be applied.

(3) *Finding the bubbles*. Refer to the solution to Problem 9.27. The solution is based on connected components, which also yields all bubbles and bubble clusters.

(4) In order to detect bubble clusters we make use of shape analysis. For each connected component, we find the eigen axes (see Section 11.4) and the standard deviation of the data along these axes (square root of the eigenvalues of the covariance matrix). One simple solution is to compute the ratio of the large to the small variance of each connected component along the eigen axes. A single, uniformly-filled, perfectly round bubble will have a ratio of 1. Deviations from 1 indicate elongations about one of the axes. We look for elliptical shapes as being formed by clusters of bubbles. A threshold to classify bubbles as single vs. clusters has to be determined experimentally. Note that single pixels or pixel streaks one pixel wide have a standard deviation of zero, so they must be processed separately. We have the option of considering connected components that consist of only one pixel to be either noise, or the smallest detectable bubble. No information is given in the problem statement about this. In theory, it is possible for a cluster to be formed such that its shape would be symmetrical about both axes, in which case the system would classify the cluster as a single bubble. Resolution of conflicts such as this would require additional processing. However, there is no evidence in the sample image to suggest that this in fact is a problem. Bubble clusters tend to appear as elliptical shapes. In cases where the ratio of the standard deviations is close to the threshold value, we could add additional processing to reduce the chances of making a mistake.

(5) *Counting individual bubbles*. A bubble that does not merge with the border of the image or is not a cluster, is by definition a single bubble. Thus, counting these bubbles is simply counting the connected components that have not been tagged as clusters or merged with the boundary of the image.

(6) *Ratio of the areas*. This ratio is simply the number of pixels in all the connected components plus the correction factors mentioned in (2), divided by the total number of pixels in the image.

The problem also asks for the size of the smallest bubble the system can detect. If, as mentioned in (4), we elect to call a one-pixel connected component a bubble, then the smallest bubble dimension detectable is the physical size of one pixel. From the problem statement, 700 pixels cover 7 cm, so the dimension of one pixel is 10 mm.

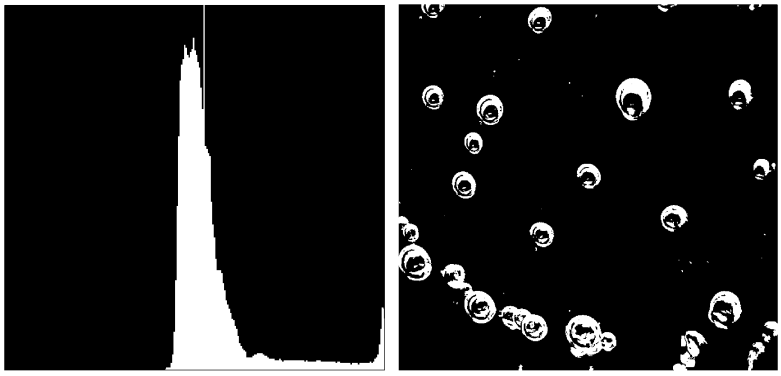


Figure P11.24

12 Problem Solutions

Problem 12.1

(a) By inspection, the mean vectors of the three classes are, approximately, $\mathbf{m}_1 = (1.5, 0.3)^T$, $\mathbf{m}_2 = (4.3, 1.3)^T$, and $\mathbf{m}_3 = (5.5, 2.1)^T$ for the classes Iris setosa, versicolor, and virginica, respectively. The decision functions are of the form given in Eq. (12.2-5). Substituting the above values of mean vectors gives:

$$\begin{aligned} d_1(\mathbf{x}) &= \mathbf{x}^T\mathbf{m}_1 - \frac{1}{2}\mathbf{m}_1^T\mathbf{m}_1 = 1.5x_1 + 0.3x_2 - 1.2 \\ d_2(\mathbf{x}) &= \mathbf{x}^T\mathbf{m}_2 - \frac{1}{2}\mathbf{m}_2^T\mathbf{m}_2 = 4.3x_1 + 1.3x_2 - 10.1 \\ d_3(\mathbf{x}) &= \mathbf{x}^T\mathbf{m}_3 - \frac{1}{2}\mathbf{m}_3^T\mathbf{m}_3 = 5.5x_1 + 2.1x_2 - 17.3 \end{aligned}$$

(b) The decision boundaries are given by the equations

$$\begin{aligned} d_{12}(\mathbf{x}) &= d_1(\mathbf{x}) - d_2(\mathbf{x}) = -2.8x_1 - 1.0x_2 + 8.9 = 0 \\ d_{13}(\mathbf{x}) &= d_1(\mathbf{x}) - d_3(\mathbf{x}) = -4.0x_1 - 1.8x_2 + 16.1 = 0 \\ d_{23}(\mathbf{x}) &= d_2(\mathbf{x}) - d_3(\mathbf{x}) = -1.2x_1 - 0.8x_2 + 7.2 = 0 \end{aligned}$$

A plot of these boundaries is shown in Fig. P12.1.

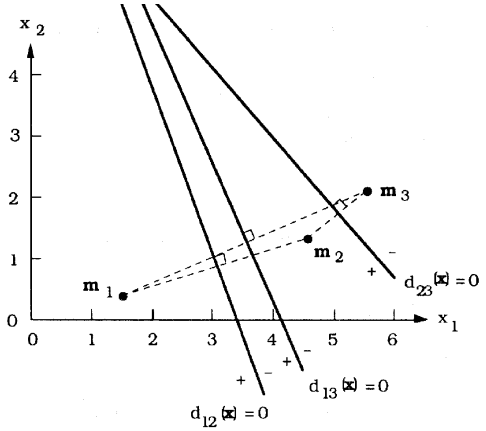


Figure P12.1

Problem 12.2

From the definition of the Euclidean distance,

$$D_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{m}_j\| = \left[(\mathbf{x} - \mathbf{m}_j)^T(\mathbf{x} - \mathbf{m}_j)\right]^{1/2}$$

Since $D_j(\mathbf{x})$ is non-negative, choosing the smallest $D_j(\mathbf{x})$ is the same as choosing the smallest $D_j^2(\mathbf{x})$, where

$$\begin{aligned} D_j^2(\mathbf{x}) &= \|\mathbf{x} - \mathbf{m}_j\|^2 = (\mathbf{x} - \mathbf{m}_j)^T(\mathbf{x} - \mathbf{m}_j) \\ &= \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{m}_j + \mathbf{m}_j^T\mathbf{m}_j \\ &= \mathbf{x}^T\mathbf{x} - \mathbf{2}\left(\mathbf{x}^T\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j\right) \end{aligned}$$

We note that the term $\mathbf{x}^T\mathbf{x}$ is independent of j (that is, it is a constant with respect to j in $D_j^2(\mathbf{x})$, $j = 1, 2, ...$). Thus, choosing the minimum of $D_j^2(\mathbf{x})$ is equivalent to choosing the maximum of $\left(\mathbf{x}^T\mathbf{m}_j - \frac{1}{2}\mathbf{m}_j^T\mathbf{m}_j\right)$.

Problem 12.3

The equation of the decision boundary between a pair of mean vectors is

$$d_{ij}(\mathbf{x}) = \mathbf{x}^T(\mathbf{m}_i - \mathbf{m}_j) - \frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j)$$

The midpoint between $\mathbf{m}_i$ and $\mathbf{m}_j$ is $(\mathbf{m}_i + \mathbf{m}_j)/2$ (see Fig. P12.3) . First, we show that this point is on the boundary by substituting it for $\mathbf{x}$ in the above equation and showing that the result is equal to 0:

$$\begin{aligned} \frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j) - \frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j) &= \frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j) \\ &\quad -\frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j) \\ &= 0 \end{aligned}$$

Next, we show that the vector $(\mathbf{m}_i - \mathbf{m}_j)$ is perpendicular to the hyperplane boundary. There are several ways to do this. Perhaps the easiest is to show that $(\mathbf{m}_i - \mathbf{m}_j)$ is in the same direction as the unit normal to the hyperplane. For a hyperplane with equation $w_1x_1 + w_2x_2 + ...w_nx_n + w_{n+1} = 0$, the unit normal is

$$\mathbf{u} = \frac{\mathbf{w}_o}{\|\mathbf{w}_o\|}$$

where $\mathbf{w}_o = (w_1, w_2, ..., w_n)^T$. Comparing the above equation for $d_{ij}(\mathbf{x})$ with the general equation of a hyperplane just given, we see that $\mathbf{w}_o = (\mathbf{m}_i - \mathbf{m}_j)$ and $w_{n+1} = -(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j)/2$. Thus, the unit normal of our decision boundary is

$$\mathbf{u} = \frac{(\mathbf{m}_i - \mathbf{m}_j)}{\|\mathbf{m}_i - \mathbf{m}_j\|}$$

which is in the same direction as the vector $(\mathbf{m}_i - \mathbf{m}_j)$. This concludes the proof.

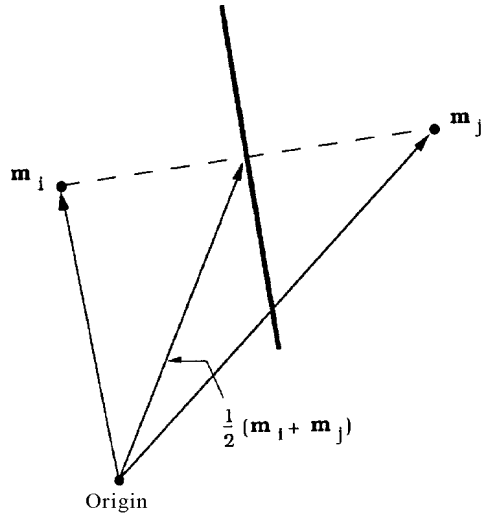


Figure P12.3

Problem 12.4

The solution is shown in Fig. P12.4, where the x's are treated as voltages and the Y's denote impedances. From basic circuit theory, the currents, I's, are the products of the voltages times the impedances.

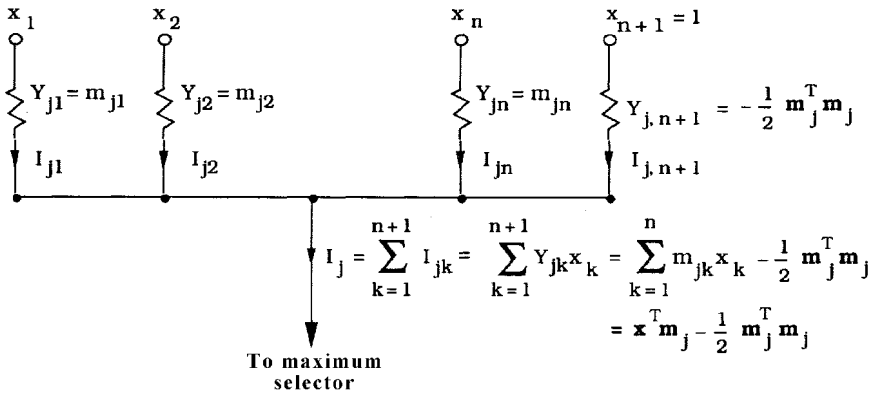


Figure P12.4

Problem 12.5

Assume that the mask is of size $J \times K$. For any value of displacement (s, t), we can express the area of the image under the mask, as well as the mask $w(x, y)$, in vector form by letting the first row of the subimage under the mask represent the first K elements of a column vector $\mathbf{a}$, the elements of next row the next K elements of $\mathbf{a}$, and so on. At the end of the procedure we subtract the average value of the gray levels in the subimage from every element of $\mathbf{a}$. The vector $\mathbf{a}$ is of size $(J \times K) \times 1$. A similar approach yields a vector, $\mathbf{b}$, of the same size, for the mask $w(x, y)$ minus its average. This vector does not change as (s, t) varies because the coefficients of the mask are fixed. With this construction in mind, we see that the numerator of Eq. (xx.3-8) is simply the vector inner-product $\mathbf{a}^T\mathbf{b}$. Similarly, the first term in the denominator is the norm squared of $\mathbf{a}$, denoted $\mathbf{a}^T\mathbf{a} = \|\mathbf{a}\|^2$, while the second term has a similar interpretation for $\mathbf{b}$. The correlation coefficient then becomes

$$\gamma(s, t) = \frac{\mathbf{a}^T\mathbf{b}}{\left[(\mathbf{a}^T\mathbf{a})(\mathbf{b}^T\mathbf{b})\right]^{1/2}}$$

When $\mathbf{a} = \mathbf{b}$ (a perfect match), $\gamma(s, t) = \|\mathbf{a}\|^2 / \|\mathbf{a}\| \|\mathbf{a}\| = 1$, which is the maximum value obtainable by the above expression. Similarly, the minimum value occurs when $a = -b$, in which case $\gamma(s, t) = -1$. Thus, although the vector $\mathbf{a}$ varies in general for every value of (s, t), the values of $\gamma(s, t)$ are all in the range $[-1, 1]$.

Problem 12.6

The solution to the first part of this problem is based on being able to extract connected components (see Chapters 2 and 11) and then determining whether a connected component is convex or not (see Chapter 11). Once all connected components have been extracted we perform a convexity check on each and reject the ones that are not convex. All that is left after this is to determine if the remaining blobs are complete or incomplete. To do this, the region consisting of the extreme rows and columns of the image is declared a region of 1's. Then if the pixel-by-pixel AND of this region with a particular blob yields at least one result that is a 1, it follows that the actual boundary touches that blob, and the blob is called incomplete. When only a single pixel in a blob yields an AND of 1 we have a marginal result in which only one pixel in a blob touches the boundary. We can arbitrarily declare the blob incomplete or not. From the point of view of implementation, it is much simpler to have a procedure that calls a blob incomplete whenever the AND operation yields one or more results valued 1.

After the blobs have been screened using the method just discussed, they need to be classified into one of the three classes given in the problem statement. We perform the classification problem based on vectors of the form $\mathbf{x} = (x_1, x_2)^T$, where x_1 and x_2 are, respectively, the lengths of the major and minor axis of an elliptical blob, the only type left after screening. Alternatively, we could use the eigen axes for the same purpose. (See Section 11.2.1 on obtaining the major axes or the end of Section 11.4 regarding the eigen axes.) The mean vector of each class needed to implement a minimum distance classifier is really given in the problem statement as the average length of each of the two axes for each class of blob. If‘ they were not given, they could be obtained by measuring the length of the axes for complete ellipses that have been classified a priori as belonging to each of the three classes. The given set of ellipses would thus constitute a training set, and learning would simply consist of computing the principal axes for all ellipses of one class and then obtaining the average. This would be repeated for each class. A block diagram outlining the solution to this problem is straightforward.

Problem 12.7

(a) Since it is given that the pattern classes are governed by Gaussian densities, only knowledge of the mean vector and covariance matrix of each class are required to specify the Bayes classifier. Substituting the given patterns into Eqs. (12.2-22) and (12.2-23) yields

$$\mathbf{m}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\mathbf{m}_2 = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$$

$$\mathbf{C}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{C}_1^{-1}$$

and

$$\mathbf{C}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{C}_2^{-1}$$

Since $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{I}$, the decision functions are the same as for a minimum distance classifier:

$$d_1(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_1 - \frac{1}{2}\mathbf{m}_1^T\mathbf{m}_1 = 1.0x_1 + 1.0x_2 - 1.0$$

and

$$d_2(\mathbf{x}) = \mathbf{x}^T\mathbf{m}_2 - \frac{1}{2}\mathbf{m}_2^T\mathbf{m}_2 = 5.0x_1 + 5.0x_2 - 25.0$$

The Bayes decision boundary is given by the equation $d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 0$, or

$$d(\mathbf{x}) = -4.0x_1 - 4.0x_2 + 24.0 = 0$$

(b) A plot of the boundary is shown in Fig. P12.7.

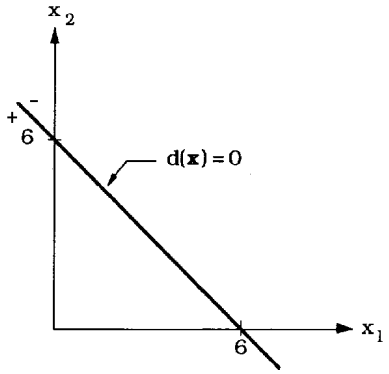


Figure P12.7

Problem 12.8

(a) As in Problem 12.7,

$$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{m}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{C}_1 = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{C}_1^{-1} = 2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad |\mathbf{C}_1| = 0.25$$

and

$$\mathbf{C}_2 = 2\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \mathbf{C}_2^{-1} = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad |\mathbf{C}_2| = 4.00$$

Since the covariance matrices are not equal, it follows from Eq. (12.2-26) that

$$\begin{aligned} d_1(\mathbf{x}) &= -\frac{1}{2}\ln(0.25) - \frac{1}{2}\left\{\mathbf{x}^T\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\mathbf{x}\right\} \\ &= -\frac{1}{2}\ln(0.25) - (x_1^2 + x_2^2) \end{aligned}$$

and

$$\begin{aligned} d_2(\mathbf{x}) &= -\frac{1}{2}\ln(4.00) - \frac{1}{2}\left\{\mathbf{x}^T\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}\mathbf{x}\right\} \\ &= -\frac{1}{2}\ln(4.00) - \frac{1}{4}(x_1^2 + x_2^2) \end{aligned}$$

where the term ln $P(\omega_j)$ was not included because it is the same for both decision functions in this case. The equation of the Bayes decision boundary is

$$d(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x}) = 1.39 - \frac{3}{4}(x_1^2 + x_2^2) = 0.$$

(b) A plot of the boundary is shown in Fig. P12.8.

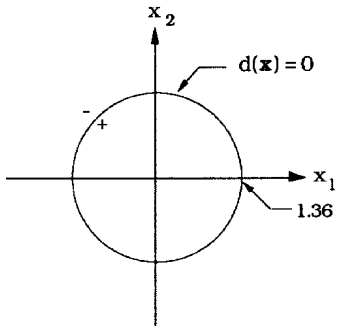


Figure P12.8

Problem 12.9

The basic mechanics are the same as in Problem 12.6, but we have the additional requirement of computing covariance matrices from the training patterns of each class.

Problem 12.10

From basic probability theory,

$$p(c) = \sum_{\mathbf{x}} p(c/\mathbf{x})p(\mathbf{x}).$$

For any pattern belonging to class ω_j, $p(c/\mathbf{x}) = p(\omega_j/\mathbf{x})$. Therefore,

$$p(c) = \sum_{\mathbf{x}} p(\omega_j/\mathbf{x})p(\mathbf{x}).$$

Substituting into this equation the formula $p(\omega_j/\mathbf{x}) = p(\mathbf{x}/\omega_j)p(\omega_j)/p(\mathbf{x})$ gives

$$p(c) = \sum_{\mathbf{x}} p(\mathbf{x}/\omega_j)p(\omega_j).$$

Since the argument of the summation is positive, $p(c)$ is maximized by maximizing $p(\mathbf{x}/\omega_j)p(\omega_j)$ for each j. That is, if for each $\mathbf{x}$ we compute $p(\mathbf{x}/\omega_j)p(\omega_j)$ for $j = 1, 2, ..., W$, and use the largest value each time as the basis for selecting the class from which $\mathbf{x}$ came, then $p(c)$ will be maximized. Since $p(e) = 1 - p(c)$, the probability of error is minimized by this procedure.

Problem 12.11

(a) For class ω_1 we let $\mathbf{y}(1) = (0,0,0,1)^T$, $\mathbf{y}(2) = (1,0,0,1)^T$, $\mathbf{y}(3) = (1,0,1,1)T$, $\mathbf{y}(4) = (1,1,0,1)T$. Similarly, for class ω_2, $\mathbf{y}(5) = (0,0,1,1)^T$, $\mathbf{y}(6) = (0,1,1,1)^T$, $\mathbf{y}(7) = (0,1,0,1)^T$, $\mathbf{y}(8) = (1,1,1,1)^T$. Then, using $c = 1$ and

$$\mathbf{w}(1) = (-1,-2,-2,0)^T$$

it follows from Eqs. (12.2-34) through (12.2-36) that:

$$\begin{aligned}
&\mathbf{w}(1)^T\mathbf{y}(1) = 0, && \mathbf{w}(2) = \mathbf{w}(1) + \mathbf{y}(1) = (-1,-2,-2,1)^T;\\
&\mathbf{w}(2)^T\mathbf{y}(2) = 0, && \mathbf{w}(3) = \mathbf{w}(2) + \mathbf{y}(2) = (0,-2,-2,2)^T;\\
&\mathbf{w}(3)^T\mathbf{y}(3) = 0, && \mathbf{w}(4) = \mathbf{w}(3) + \mathbf{y}(3) = (1,-2,-1,3)^T;\\
&\mathbf{w}(4)^T\mathbf{y}(4) = 2, && \mathbf{w}(5) = \mathbf{w}(4) = (1,-2,-1,3)^T;\\
&\mathbf{w}(5)^T\mathbf{y}(5) = 2, && \mathbf{w}(6) = \mathbf{w}(5) - \mathbf{y}(5) = (-1,-2,-2,2)^T;\\
&\mathbf{w}(6)^T\mathbf{y}(6) = -2, && \mathbf{w}(7) = \mathbf{w}(6) = (-1,-2,-2,2)^T;\\
&\mathbf{w}(7)^T\mathbf{y}(7) = 0, && \mathbf{w}(8) = \mathbf{w}(7) - \mathbf{y}(7) = (1,-3,-2,1)^T;\\
&\mathbf{w}(8)^T\mathbf{y}(8) = -3, && \mathbf{w}(9) = \mathbf{w}(8) = (1,-3,-2,1)^T.
\end{aligned}$$

Since a complete iteration through all patterns without an error was not achieved, the patterns are recycled by letting $\mathbf{y}(9) = \mathbf{y}(1)$, $\mathbf{y}(10) = \mathbf{y}(2)$, and so on, which gives

$$\begin{aligned}
&\mathbf{w}(9)^T\mathbf{y}(9) = 1, && \mathbf{w}(10) = \mathbf{w}(9) = (1,-3,-2,1)^T;\\
&\mathbf{w}(10)^T\mathbf{y}(10) = 2, && \mathbf{w}(11) = \mathbf{w}(10) = (1,-3,-2,1)^T;\\
&\mathbf{w}(11)^T\mathbf{y}(11) = 0, && \mathbf{w}(12) = \mathbf{w}(11) + \mathbf{y}(11) = (2,-3,-1,2)^T;\\
&\mathbf{w}(12)^T\mathbf{y}(12) = 1, && \mathbf{w}(13) = \mathbf{w}(12) = (2,-3,-1,2)^T;\\
&\mathbf{w}(13)^T\mathbf{y}(13) = 1, && \mathbf{w}(14) = \mathbf{w}(13) - \mathbf{y}(13) = (2,-3,-2,1)^T;\\
&\mathbf{w}(14)^T\mathbf{y}(14) = -4, && \mathbf{w}(15) = \mathbf{w}(14) = (2,-3,-2,1)^T;\\
&\mathbf{w}(15)^T\mathbf{y}(15) = -2, && \mathbf{w}(16) = \mathbf{w}(15) = (2,-3,-2,1)^T;\\
&\mathbf{w}(16)^T\mathbf{y}(16) = -2, && \mathbf{w}(17) = \mathbf{w}(16) = (2,-3,-2,1)^T.
\end{aligned}$$

Again, since a complete iteration over all patterns without an error was not achieved, the patterns are recycled by letting $\mathbf{y}(17) = \mathbf{y}(1)$, $\mathbf{y}(18) = \mathbf{y}(2)$, and so on, which gives:

$$\begin{aligned}
&\mathbf{w}(17)^T\mathbf{y}(17) = 1, && \mathbf{w}(18) = \mathbf{w}(17) = (2,-3,-2,1)^T;\\
&\mathbf{w}(18)^T\mathbf{y}(18) = 3, && \mathbf{w}(19) = \mathbf{w}(18) = (2,-3,-2,1)^T;\\
&\mathbf{w}(19)^T\mathbf{y}(19) = 1, && \mathbf{w}(20) = \mathbf{w}(19) = (2,-3,-2,1)^T;\\
&\mathbf{w}(20)^T\mathbf{y}(20) = 0, && \mathbf{w}(21) = \mathbf{w}(20) + \mathbf{y}(20) = (3,-2,-2,2)^T;\\
&\mathbf{w}(21)^T\mathbf{y}(21) = 0, && \mathbf{w}(22) = \mathbf{w}(21) - \mathbf{y}(21) = (3,-2,-3,1)^T.
\end{aligned}$$

It is easily verified that no more corrections take place after this step, so $\mathbf{w}(22) = (3,-2,-3,1)^T$ is a solution weight vector.

(b) The decision surface is given by the equation

$$\mathbf{w}^T\mathbf{y} = 3y_1 - 2y_2 - 3y_3 + 1 = 0$$

A section of this surface is shown schematically in Fig. P12.11. The positive side of the surface faces the origin.

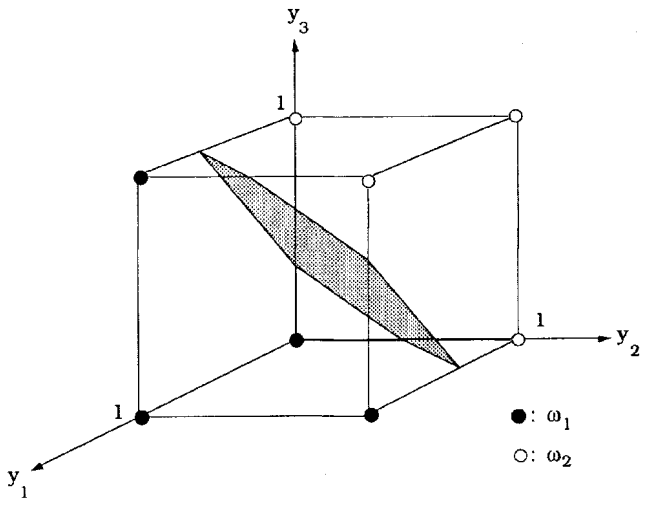


Figure P12.11

Problem 12.12

We start by taking the partial derivative of J with respect to $\mathbf{w}$:

$$\frac{\partial J}{\partial \mathbf{w}} = \frac{1}{2}\left[\mathbf{y}\text{sgn}(\mathbf{w}^T\mathbf{y}) - \mathbf{y}\right]$$

where, by definition, $\text{sgn}(\mathbf{w}^T\mathbf{y}) = 1$ if $\mathbf{w}^T\mathbf{y} > 0$, and $\text{sgn}(\mathbf{w}^T\mathbf{y}) = -1$ otherwise. Substituting the partial derivative into the general expression given in the problem statement gives

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{c}{2}\left\{\mathbf{y}(\mathbf{k}) - \mathbf{y}(\mathbf{k})\text{sgn}\left[\mathbf{w}(\mathbf{k})^T\mathbf{y}(\mathbf{k})\right]\right\}$$

where $\mathbf{y}(k)$ is the training pattern being considered at the kth iterative step. Substituting the definition of the sgn function into this result yields

$$\mathbf{w}(k+1) = \mathbf{w}(k) + c\begin{cases}\mathbf{0} & \text{if } \mathbf{w}(\mathbf{k})^T\mathbf{y}(\mathbf{k}) \\ \mathbf{y}(k) & \text{otherwise}\end{cases}$$

where $c > 0$ and $\mathbf{w}(1)$ is arbitrary. This expression agrees with the formulation given in the problem statement.

Problem 12.13

Let the training set of patterns be denoted by $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$. It is assumed that the

training patterns of class ω_2 have been multiplied by -1. If the classes are linearly separable, we want to prove that the perceptron training algorithm yields a solution weight vector, $\mathbf{w}^*$, with the property

$$\mathbf{w}^{*T}\mathbf{y}_i \geq T_0$$

where T_0 is a nonnegative threshold. With this notation, the Perceptron algorithm (with $c = 1$) is expressed as $\mathbf{w}(k+1) = \mathbf{w}(k)$ if $\mathbf{w}^T(k)\mathbf{y}_i(k) \geq T_0$ or $\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{y}_i(k)$ otherwise.

Suppose that we retain only the values of k for which a correction takes place (these are really the only indices of interest). Then, re-adapting the index notation, we may write

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mathbf{y}_i(k)$$

and

$$\mathbf{w}^T(k)\mathbf{y}i(k) \leq T_0$$

With these simplifications in mind, the proof of convergence is as follows: From the above equation,

$$\mathbf{w}(k+1) = \mathbf{w}(1) + \mathbf{y}_i(1) + \mathbf{y}_i(2) + \cdots + \mathbf{y}_i(k)$$

Taking the inner product of the solution weight vector with both sides of this equation gives

$$\mathbf{w}^T(k+1)\mathbf{w}^* = \mathbf{w}^T(1)\mathbf{w}^* + \mathbf{y}_i^T(1)\mathbf{w}^* + \mathbf{y}_i^T(2)\mathbf{w}^* + \cdots + \mathbf{y}_i^T(k)\mathbf{w}^*$$

Each term $\mathbf{y}_i^T(j)\mathbf{w}^*$, $j = 1, 2, ..., k$, is less than T_0, so

$$\mathbf{w}^T(k+1)\mathbf{w}^* \geq \mathbf{w}^T(1)\mathbf{w}^* + kT_0$$

Using the Cauchy-Schwartz inequality, $\|\mathbf{a}\|^2 \|\mathbf{b}\|^2 \geq (\mathbf{a}^T\mathbf{b})^2$, results in

$$\left[\mathbf{w}^T(k+1)\mathbf{w}^*\right]^2 \leq \left\|\mathbf{w}^T(k+1)\right\|^2 \|\mathbf{w}^*\|^2$$

or

$$\left\|\mathbf{w}^T(k+1)\right\|^2 \geq \frac{\left[\mathbf{w}^T(k+1)\mathbf{w}^*\right]^2}{\|\mathbf{w}^*\|^2}.$$

Another line of reasoning leads to a contradiction regarding $\left\|\mathbf{w}^T(k+1)\right\|^2$. From above,

$$\|\mathbf{w}(j+1)\|^2 = \|\mathbf{w}(j)\|^2 + 2\mathbf{w}^T(j)\mathbf{y}_i(j) + \|\mathbf{y}_i(j)\|^2$$

or

$$\|\mathbf{w}(j+1)\|^2 - \|\mathbf{w}(j)\|^2 = 2\mathbf{w}^T(j)\mathbf{y}_i(j) + \|\mathbf{y}_i(j)\|^2$$

Let $Q = \max_i ||\mathbf{y}_i(j)||^2$. Then, since $\mathbf{w}^T(j)\mathbf{y}_i(j) \leq T_0$,

$$\|\mathbf{w}(j+1)\|^2 - \|\mathbf{w}(j)\|^2 \leq 2T_0 + Q$$

Adding these inequalities for $j = 1, 2, \ldots, k$ yields

$$\|\mathbf{w}(j+1)\|^2 \leq \|\mathbf{w}(1)\|^2 + [2T_0 + Q]\, k$$

This inequality establishes a bound on $\|\mathbf{w}(j+1)\|^2 2$ that conflicts for sufficiently large k with the bound established by our earlier inequality. In fact, k can be no larger than k_m, which is a solution to the equation

$$\frac{\left[\mathbf{w}^T(k+1)\mathbf{w}^* + k_m T_0\right]^2}{\|\mathbf{w}^*\|^2} = \|\mathbf{w}(1)\|^2 + [2T_0 + Q]\, k_m$$

This equation says that k_m is finite, thus proving that the perceptron training algorithm converges in a finite number of steps to a solution weight vector $\mathbf{w}^*$ if the patterns of the training set are linearly separable.

Note: The special case with $T_0 = 0$ is proved in a slightly different manner. Under this condition we have

$$\mathbf{w}^T(k+1)\mathbf{w}^* \geq \mathbf{w}^T(1)\mathbf{w}^* + ka$$

where

$$a = \min_i \left[\mathbf{y}_i^T(j)\mathbf{w}^*\right]$$

Since, by hypothesis, $\mathbf{w}^*$ is a solution weight vector, we know that $\left[\mathbf{y}_i^T(j)\mathbf{w}^*\right] \geq 0$. Also, since $\mathbf{w}^T(j)\mathbf{y}_i(j) \leq \ (T = 0)$,

$$\begin{aligned} \|\mathbf{w}(j+1)\|^2 - \|\mathbf{w}(j)\|^2 &\leq \|\mathbf{y}_i(j)\|^2 \\ &\leq Q. \end{aligned}$$

The rest of the proof remains the same. The bound on the number of steps is the value of k_m that satisfies the following equation:

$$\frac{\left[\mathbf{w}^T(1)\mathbf{w}^* + k_m a\right]^2}{\|\mathbf{w}^*\|^2} = \|\mathbf{w}(1)\|^2 + Qk_m$$

Problem 12.14

The single decision function that implements a minimum distance classifier for two classes is of the form

$$d_{ij}(\mathbf{x}) = \mathbf{x}^T(\mathbf{m}_i - \mathbf{m}_j) - \frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j).$$

Thus, for a particular pattern vector $\mathbf{x}$, when $d_{ij}(\mathbf{x}) > 0$, $\mathbf{x}$ is assigned to class ω_1 and, when $d_{ij}(\mathbf{x}) < 0$, $\mathbf{x}$ is assigned to class ω_2. Values of $\mathbf{x}$ for which $d_{ij}(\mathbf{x}) = 0$ are on the boundary (hyperplane) separating the two classes. By letting $\mathbf{w} = (\mathbf{m}_i - \mathbf{m}_j)$ and $w_{n+1} = -\frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j)$, we can express the above decision function in the form

$$d(\mathbf{x}) = \mathbf{w}^T\mathbf{x} - w_{n+1}.$$

This is recognized as a linear decision function in n dimensions, which is implemented by a single layer neural network with coefficients

$$w_k = (m_{ik} - m_{jk}) \qquad k = 1, 2, \ldots, n$$

and

$$\theta = w_{n+1} = -\frac{1}{2}(\mathbf{m}_i^T\mathbf{m}_i - \mathbf{m}_j^T\mathbf{m}_j).$$

Problem 12.15

The approach to solving this problem is basically the same as in Problem 12.14. The idea is to combine the decision functions in the form of a hyperplane and then equate coefficients. For equal covariance matrices, the decision function for two pattern classes is obtained Eq. (12.2-27):

$$\begin{aligned} d_{ij}(\mathbf{x}) &= d_i(\mathbf{x}) - d_j(\mathbf{x}) = \ln P(\omega_i) - \ln P(\omega_j) + \mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_i - \mathbf{m}_j) \\ &\quad -\frac{1}{2}(\mathbf{m}_i - \mathbf{m}_j)^T\mathbf{C}^{-1}(\mathbf{m}_i - \mathbf{m}_j). \end{aligned}$$

As in Problem 12.14, this is recognized as a linear decision function of the form

$$d(\mathbf{x}) = \mathbf{w}^T\mathbf{x} - w_{n+1}$$

which is implemented by a single layer perceptron with coefficients

$$w_k = v_k \qquad k = 1, 2, \ldots, n$$

and

$$\theta = w_{n+1} = \ln P(\omega_i) - \ln P(\omega_j) + \mathbf{x}^T\mathbf{C}^{-1}(\mathbf{m}_i - \mathbf{m}_j)$$

where the v_k are elements of the vector

$$\mathbf{v} = \mathbf{C}^{-1}(\mathbf{m}_i - \mathbf{m}_j).$$

Problem 12.16

(a) When $P(\omega_i) = P(\omega_j)$ and $\mathbf{C} = \mathbf{I}$.

(b) No. The minimum distance classifier implements a decision function that is the perpendicular bisector of the line joining the two means. If the probability densities are known, the Bayes classifier is guaranteed to implement an optimum decision function in the minimum average loss sense. The generalized delta rule for training a neural network says nothing about these two criteria, so it cannot be expected to yield the decision functions in Problems 12.14 or 12.15.

Problem 12.17

The classes and boundary needed to separate them are shown in Fig. P12.17(a). The boundary of minimum complexity in this case is a triangle, but it would be so tight

in this arrangement that even small perturbations in the position of the patterns could result in classification errors. Thus, we use a network with the capability to implement 4 surfaces (lines) in 2D. The network, shown in Fig. P12.17(b), is an extension of the concepts discussed in the text in connection with Fig. 12.22. In this case, the output node acts like an AND gate with 4 inputs. The output node outputs a 1 (high) when the outputs of the preceding 4 nodes are all high simultaneously. This corresponds to a pattern being on the + side of all 4 lines and, therefore, belonging to class ω_1. Any other combination yields a 0 (low) output, indicating class ω_{21}.

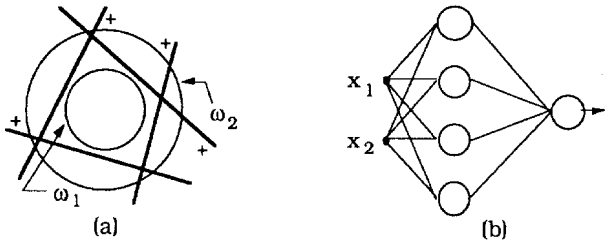


Figure P12.17

Problem 12.18

All that is needed is to generate for each class training vectors of the form $\mathbf{x} = (x_1, x_2)^T$, where x_1 is the length of the major axis and x_2 is the length of the minor axis of the blobs comprising the training set. These vectors would then be used to train a neural network using, for example, the generalized delta rule. (Since the patterns are in 2D, it is useful to point out to students that the neural network could be designed by inspection in the sense that the classes could be plotted, the decision boundary of minimum complexity obtained, and then its coefficients used to specify the neural network. In this case the classes are far apart with respect to their spread, so most likely a single layer network implementing a linear decision function could do the job.)

Problem 12.19

This problem, although it is a simple exercise in differentiation, is intended to help the student fix in mind the notation used in the derivation of the generalize delta rule. From Eq. (12.2-50), with $\theta_0 = 1$,

$$h_j(I_j) = \frac{1}{1 + e^{-\left[\sum_{k=1}^{N_K} w_{jk} O_k + \theta_j\right]}}.$$

Since, from Eq. (12.2-48),

$$I_j = \sum_{k=1}^{N_K} w_{jk} O_k$$

it follows that

$$h_j(I_j) = \frac{1}{1 + e^{-[I_j + \theta_j]}}.$$

Taking the partial derivative of this expression with respect to I_j gives

$$h'_j(I_j) = \frac{\partial h_j(I_j)}{\partial I_j} = \frac{e^{-[I_j + \theta_j]}}{\left[1 + e^{-[I_j + \theta_j]}\right]^2}.$$

From Eq. (12.2-49)

$$O_j = h_j(I_j) = \frac{1}{1 + e^{-[I_j + \theta_j]}}.$$

It is easily shown that

$$O_j(1 - O_j) = \frac{e^{-[I_j + \theta_j]}}{\left[1 + e^{-[I_j + \theta_j]}\right]^2}$$

so

$$h'_j(I_j) = O_j(1 - O_j)$$

This completes the proof.

Problem 12.20

The first part of Eq. (12.3-3) is proved by noting that the degree of similarity, k, is non-negative, so $D(A, B) = 1/k \geq 0$. Similarly, the second part follows from the fact that k is infinite when (and only when) the shapes are identical.

To prove the third part we use the definition of D to write

$$D(A, C) \leq \max\left[D(A, B), D(B, C)\right]$$

as

$$\frac{1}{k_{ac}} \leq \max\left[\frac{1}{k_{ab}}, \frac{1}{k_{bc}}\right]$$

or, equivalently,

$$k_{ac} \geq \min\left[k_{ab}, k_{bc}\right]$$

where k_{ij} is the degree of similarity between shape i and shape j. Recall from the definition that k is the largest order for which the shape numbers of shape i and shape j still coincide. As Fig. 12.24(b) illustrates, this is the point at which the figures "separate" as we move further down the tree (note that k increases as we move further down the tree). We prove that $k_{ac} \geq \min[k_{ab}, k_{bc}]$ by contradiction. For $k_{ac} \leq \min[k_{ab}, k_{bc}]$ to hold, shape A has to separate from shape C *before* (1) shape A separates from shape B, *and* (2) *before* shape B separates from shape C, otherwise $k_{ab} \leq k_{ac}$ or $k_{bc} \leq k_{ac}$, which automatically violates the condition $k_{ac} < \min[k_{ab}, k_{bc}]$. But, if (1) has to hold,

then Fig. P12.20 shows the only way that A can separate from C before separating from B. This, however, violates (2), which means that the condition $k_{ac} < \min[k_{ab}, k_{bc}]$ is violated (we can also see this in the figure by noting that $k_{ac} = k_{bc}$ which, since $k_{bc} < k_{ab}$, violates the condition). We use a similar argument to show that if (2) holds then (1) is violated. Thus, we conclude that it is impossible for the condition $k_{ac} < \min[k_{ab}, k_{bc}]$ to hold, thus proving that $k_{ac} \geq \min[k_{ab}, k_{bc}]$ or, equivalently, that $D(A, C) \leq \max[D(A, B), D(B, C)]$.

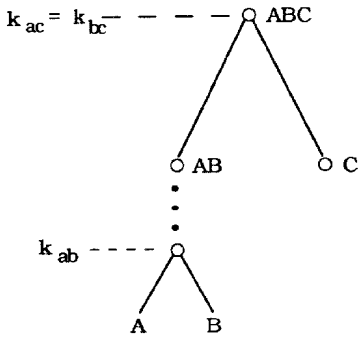


Figure P12.20

Problem 12.21

$Q = 0$ implies that $\max(|A|, |B|) = M$. Suppose that $|A| > |B|$. Then, it must follow that $|A| = M$ and, therefore, that $M > |B|$. But M is obtained by matching A and B, so it must be bounded by $M \leq \min(|A|, |B|)$. Since we have stipulated that $|A| > |B|$, the condition $M \leq \min(|A|, |B|)$ implies $M \leq |B|$. But this contradicts the above result, so the only way for $\max(|A|, |B|) = M$ to hold is if $|A| = |B|$. This, in turn, implies that A and B must be identical strings ($A \equiv B$) because $|A| = |B| = M$ means that all symbols of A and B match. The converse result that if $A \equiv B$ then $Q = 0$ follows directly from the definition of Q.

Problem 12.22

(a) An automaton capable of accepting *only* strings of the form $ab^n a \geq 1$, shown in Fig. P12.22, is given by

$$A_f = (Q, \Sigma, \delta, q_0, F),$$

with

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3, q_\emptyset\}, \\ \Sigma &= \{a, b\}, \end{aligned}$$

mappings

$$\begin{aligned} \delta(q_0, a) &= \{q_1\}, \\ \delta(q_1, b) &= \{q_1, q_2\}, \\ \delta(q_2, a) &= \{q_3\} \end{aligned}$$

and

$$F = \{q3\}.$$

For completeness we write

$$\delta(q_0, b) = \delta(q_1, a) = \delta(q_2, b) = \delta(q_3, a) = \delta(q_3, b) = \delta(q_\emptyset, a) = \delta(q_\emptyset, b) = \{q_\emptyset\},$$

corresponding to the null state.

(b) To obtain the corresponding grammar we use the procedure discussed in Section 12.3.3 under the heading *Automata as string recognizers*: 1. If q_j is in $\delta(q_i, c)$, there is a production $X_i \longrightarrow X_j$ in P; 2. If a state in F is in $\delta(q_i, c)$, there is a production $X_i \longrightarrow c$ in P. Normally, null state transitions are not included in the generation of productions. Using the results in (a) we obtain the grammar $G = (N, \Sigma, P, X_0)$, with $N = \{X_0, X_1, X_2\}$, $\Sigma = \{a, b\}$, and productions $P = \{X_0 \longrightarrow aX_1,\ X_1 \longrightarrow bX_1,\ X_1 \longrightarrow bX_2,\ X_2 \longrightarrow a\}$.

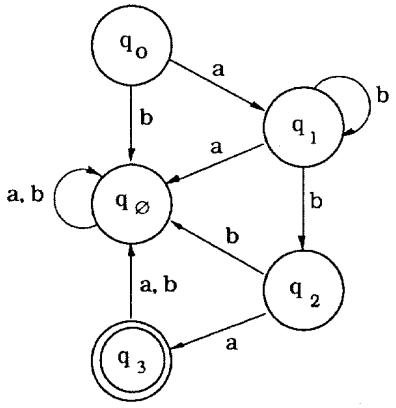


Figure P12.22

Problem 12.23

The patterns are of the form shown in the solution to Problem 11.2. (This problem is

not starred, so a solution in not included in the book web site. If the problem was not assigned, it might be a good idea to give the solution in class). A possible expansive tree grammar is $G = (N, \Sigma, P, r, S)$, with $N = \{S, X_1, X_2, ..., X_6\}$, $\Sigma = \{0, 1\}$, $r(0) = \{0, 1, 2\}$, $r(1) = \{0, 1, 2\}$, and the productions shown in Fig. P12.23:

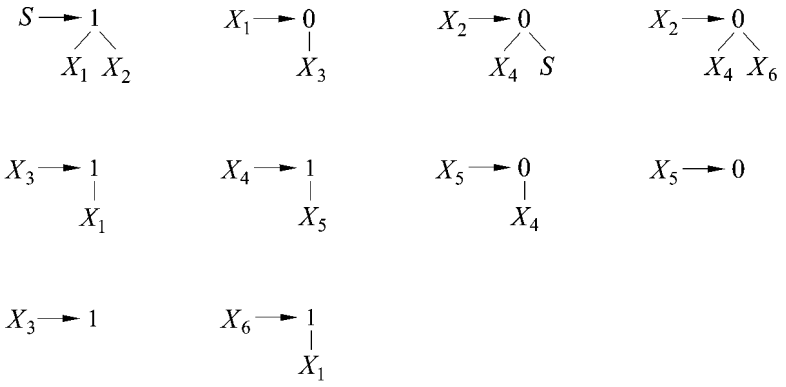


Figure P12.23

Problem 12.24

For the sample set $R^+ = \{aba, abba, abbba\}$ it is easily shown that, for $k = 1$ and 2, $h(\lambda, R^+, k) = \emptyset$, the null set. Since $q_0 = h(\lambda, R^+, k)$ is part of the inference procedure, we need to choose k large enough so that $h(\lambda, R^+, k)$ is not the null set. The shortest string in R^+ has three symbols, so $k = 3$ is the smallest value that can accomplish this. For this value of k, a trial run will show that one more string needs to be added to R^+ in order for the inference procedure to discover iterative regularity in symbol b. The sample string set then becomes $R^+ = \{aba, abba, abbba, abbbba\}$. Recalling that $h(z, R^+, k) = \{w \,|zw \text{ in } R^+,\ |w| \leq\ k\}$ we proceed as follows:

$$
\begin{aligned}
z = \lambda, \quad h(\lambda, R^+, 3) &= \{w \,|\lambda w \text{ in } R^+, |w| \leq 3\} \\
&= \{aba\} \\
&= q_0;
\end{aligned}
$$

$$
\begin{aligned}
z = a, \quad h(a, R^+, 3) &= \{w \,|aw \text{ in } R^+, |w| \leq 3\} \\
&= \{ba, bba\} \\
&= q_1;
\end{aligned}
$$

$$
\begin{aligned}
z = ab, \qquad h(ab, R^+, 3) &= \{w \,|abw \text{ in } R^+, |w| \leq 3\} \\
&= \{a, ba, bba\} \\
&= q_2; \\
z = aba, \qquad h(aba, R^+, 3) &= \{w \,|abaw \text{ in } R^+, |w| \leq 3\} \\
&= \{\lambda\} \\
&= q_3; \\
z = abb, \qquad h(abb, R^+, 3) &= \{w \,|abbw \text{ in } R^+, |w| \leq 3\} \\
&= \{a, ba, bba\} \\
&= q_2; \\
z = abba, \qquad h(abba, R^+, 3) &= \{w \,|abbaw \text{ in } R^+, |w| \leq 3\} \\
&= \{\lambda\} \\
&= q_3; \\
z = abbb, \qquad h(abbb, R^+, 3) &= \{w \,|abbbw \text{ in } R^+, |w| \leq 3\} \\
&= \{a, ba\} \\
&= q_4; \\
z = abbba, \qquad h(abbba, R^+, 3) &= \{w \,|abbbaw \text{ in } R^+, |w| \leq 3\} \\
&= \{\lambda\} \\
&= q_3; \\
z = abbbb, \qquad h(abbbb, R^+, 3) &= \{w \,|abbbbw \text{ in } R^+, |w| \leq 3\} \\
&= \{a\} \\
&= q_5; \\
z = abbbba, \qquad h(abbbba, R^+, 3) &= \{w \,|abbbbaw \text{ in } R^+, |w| \leq 3\} \\
&= \{\lambda\} \\
&= q_3;
\end{aligned}
$$

Other strings z in $\Sigma^* = (a, b)^*$ yield strings zw that do not belong to R^+, giving rise to another state, denoted $q_\emptyset$, which corresponds to the condition that h is the null set. Therefore, the states are $q_0 = \{aba\}$, $q_1 = \{ba, bba\}$, $q_2 = \{a, ba, bba\}$, $q_3 = \{\lambda\}$, $q_4 = \{a, ba\}$, and $q_5 = \{a\}$, which gives the set $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_\emptyset\}$.

The next step is to obtain the mappings. We start by recalling that, in general, $q_0 = h(\lambda, R^+, k)$. Also, in general,

$$\delta(q, c) = \{q' \text{ in } Q \,|q' = h(zc, R^+, k), \quad \text{with} \quad q = h(z, R+, k)\}.$$

In our case, $q_0 = h(\lambda, R^+, 3)$ and, therefore,

$$\delta(q_0, a) = h(\lambda a, R^+, 3) = h(a, R^+, 3) = \{q_1\} = q_1$$

and

$$\delta(q_0, b) = h(\lambda b, R^+, 3) = h(b, R^+, 3) = \{q_\emptyset\} = q_\emptyset,$$

where we have omitted the curly brackets for clarity in notation since the set contains only one element. Similarly, $q_1 = h(a, R^+, 3)$, and

$$\begin{aligned} \delta(q_1, a) &= h(aa, R^+, 3) = h(a, R^+, 3) = q_\emptyset, \\ \delta(q_1, b) &= h(ab, R^+, 3) = q_2. \end{aligned}$$

Continuing in this manner gives $q_2 = h(ab, R^+, 3) = h(abb, R^+, 3)$,

$$\begin{aligned} \delta(q2, a) &= h(aba, R^+, 3) = h(abba, R^+, 3) = q3, \\ \delta(q_2, b) &= h(abb, R^+, 3) = q_2, \end{aligned}$$

and, also,

$$\delta(q_2, b) = h(abbb, R^+, 3) = q_4.$$

Next, $q_3 = h(aba, R^+, 3) = h(abba, R^+, 3) = h(abbba, R^+, 3) = h(abbbba, R^+, 3)$, from which we obtain

$$\begin{aligned} \delta(q_3, a) &= h(abaa, R^+, 3) = h(abbaa, R^+, 3) \\ &= h(abbbaa, R^+, 3) = h(abbbbaa, R^+, 3) \\ &= q_\emptyset \\ \delta(q3, b) &= h(abab, R^+, 3) = h(abbab, R^+, 3) \\ &= h(abbbab, R^+, 3) = h(abbbbab, R^+, 3) \\ &= q_\emptyset; \end{aligned}$$

For the following state, $q_4 = h(abbb, R^+, 3)$,

$$\begin{aligned} \delta(q_4, a) &= h(abbba, R^+, 3) = q_3, \\ \delta(q_4, b) &= h(abbbb, R^+, 3) = q_5. \end{aligned}$$

Finally, for the last state, $q_5 = h(abbbb, R^+, 3)$, and

$$\begin{aligned} \delta(q_5, a) &= h(abbbba, R^+, 3) = q_3, \\ \delta(q_5, b) &= h(abbbbb, R^+, 3) = q_\emptyset. \end{aligned}$$

We complete the elements of the automaton by recalling that $F = \{q \,|q \,\text{in}\, Q, \lambda \,\text{in}\, q\} = q_3$. We also include two remaining mappings that yield the null set: $\delta(q_\emptyset, a) = \delta(q_\emptyset, b) = q_\emptyset$.

Summarizing, the state mappings are:

$$\begin{aligned} \delta(q_0, a) &= q_1, \delta(q_0, b) = q_\emptyset; \\ \delta(q_1, a) &= q_\emptyset, \delta(q_1, b) = q_2; \end{aligned}$$

$$\begin{aligned}
\delta(q_2, a) &= q_3, \delta(q_2, b) = \{q_2, q_4\};\\
\delta(q_3, a) &= q_\emptyset, \delta(q_3, b) = q_\emptyset;\\
\delta(q_4, a) &= q_3, \delta(q_4, b) = q_5;\\
\delta(q_5, a) &= q_3, \delta(q_5, b) = q_\emptyset;\\
\delta(q_\emptyset, a) &= q_\emptyset, \delta(q_\emptyset, b) = q_\emptyset.
\end{aligned}$$

A diagram of the automaton is shown in Fig. P12.24. The iterative regularity on b is evident in state q_2. This automaton is not as elegant as its counterpart in Problem 12.22(a). This is not unexpected because nothing in the inference procedure deals with state minimization. Note, however, that the automaton accepts only strings of the form $ab^n a$, $b \geq 1$, as desired. The minimization aspects of a design generally follow inference and are based on one of several standard methods (see, for example, Gonzalez and Thomason [1978]). In this particular example, even visual inspection reveals that states q_4 and q_5 are redundant.

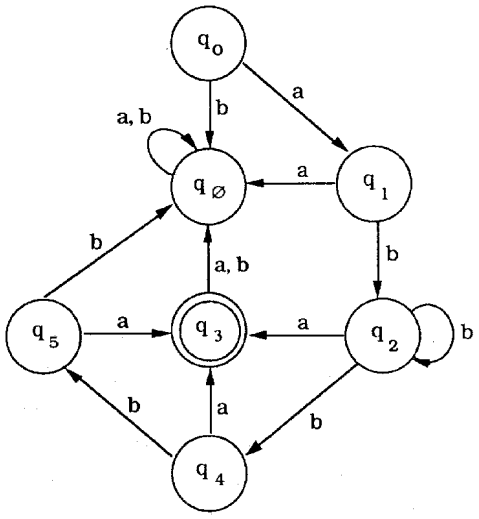


Figure P12.24

Problem 12.25

Consider the automaton related to Fig. 12.30, and the tree shown in Fig. 12.31(b). The explanation is simplified by moving up the tree one level at a time, starting at the lowest level. In this case the lowest level is in the innermost branch labeled with a's. We start at its frontier node and assign state X_1 to that node by virtue of f_a. The next level contains

an a along that same branch, but its offspring now has been labeled X_1. Assignment f_a again indicates an assignment of X_1. We move up the tree in this manner. The assignments along all the single branches of a's are X_1's, while those along the single branches of b's are X_2's. This continues until the automaton gets to the bottom of the single branch of a's at the center of the tree. This particular a now has three offspring labeled X_1 and three labeled X_2, which causes f_a to assign state S to that a. As the automaton moves up one more level, it encounters another a. Since its offspring is S, f_a assigns state S to it and moves up another level. It is evident that the automaton will end in state S when the last (root) node is processed. Since S is in F, the automaton in fact has accepted the tree in Fig. 12.31(b).

Problem 12.26

There are various possible approaches to this problem, and our students have shown over the years a tendency to surprise us with new and novel approaches to problems of this type. We give here a set of guidelines that should be satisfied by most practical solutions, and also offer suggestions for specific solutions to various parts of the problem. Depending on the level of maturity of the class, some of these may be offered as "hints" when the problem is assigned.

Since speed and cost are essential system specifications, we conceptualize a binary approach in which image acquisition, preprocessing, and segmentation are combined into one basic operation. This approach leads us to global thresholding as the method of choice. In this particular case this is possible because we can solve the inspection problem by concentrating on the white parts of the flag (stars and white stripes). As discussed in Section 10.3.2, uniform illumination is essential, especially when global thresholding is used for segmentation. The student should mention something about uniform illumination, or compensation for nonuniform illumination. A discussion by the student of color filtering to improve contrast between white and (red/blue/background) parts of an image is a plus in the design.

The first step is to specify the size of the viewing area, and the resolution required to detect the smallest components of interest, in this case the stars. Since the images are moving and the exact location of each flag is not known, it is necessary to specify a field of view that will guarantee that every image will contain at least one complete flag. In addition, the frame rate must be fast enough so that no flags are missed. The first part of the problem is easy to solve. The field of view has to be wide enough to encompass an

area slightly greater across than two flags plus the maximum separation between them. Thus, the width, W, of the viewing area must be at least $W = 2(5) + 2.05 = 12.1$in. If we use a standard CCD camera of resolution 640×480 elements and view an area 12.8 in. wide, this will give us a sampling rate of approximately 50 pixels/inch, or 250 pixels across a single flag. Visual inspection of a typical flag will show that the blue portion of a flag occupies about 0.4 times the length of the flag, which in this case gives us about 100 pixels per line in the blue area. There is a maximum of six stars per line, and the blue space between them is approximately 1.5 times the width of a star, so the number of pixels across a star is $100/([1 + 1.5] \times 6) \simeq 6$ pixels/star.

The next two problems are to determine the shutter speed and the frame rate. Since the number of pixels across each object of interest is only 6, we fix the blur at less than one pixel. Following the approach used in the solution of Problem 10.35, we first determine the distance between pixels as $(12.8\,\text{in})/640\,\text{pixels} = 0.02\,\text{in/pixel}$. The maximum speed of the flags is 21in/sec. At this speed, the flags travel $21/0.02 = 1,050$ pixels/sec. We are requiring that a flag not travel more than one pixel during exposure; that is $(1,050\,\text{pixels/sec}) \times T\,\text{sec} \leq 1\,\text{pixel}$. So, $T \leq 9.52 \times 10^{-4}$ sec is the shutter speed needed.

The frame rate must be fast enough to capture an image of every flag that passes the inspection point. Since it takes a flag $(21\,\text{in/sec})/(12.8\,\text{in}) \simeq 0.6$ sec to cross the entire field of view we take a frame every 0.3 sec in order to guarantee that every image will contain a whole flag, and that no flag will be missed. We assume that the camera is computer controlled to fire from a clock signal. We also make the standard assumption that it takes $1/30\,\text{sec} \simeq 330 \times 10^{-4}$ sec to read a captured image into a frame buffer. Therefore, the total time needed to acquire an image is $(330+9.5)\times10^{-4} \simeq 340\times10^{-4}$ sec. Subtracting this quantity from the 0.3 sec frame rate leaves us with about 0.27 sec to do all the processing required for inspection, and to output an appropriate signal to some other part of the manufacturing process.

Since a global thresholding function can be incorporated in most digitizers as part of the data acquisition process, no additional time is needed to generate a binary image. That is, we assume that the digitizer outputs the image in binary form. The next step is to isolate the data corresponding to a complete flag. Given the imaging geometry and frame rate discussed above, four basic binary image configurations are expected: (1) part of a flag on the left of the image, followed by a whole flag, followed by another partial flag; (2) one entire flag touching the left border, followed by a second entire flag, and then a gap before the right border; (3) the opposite of (2); and (4) two entire flags, with

neither flag touching the boundary of the image. Cases (2), (3), and (4) are not likely to occur with any significant frequency, but we will check for each of these conditions. As will be seen below, Cases (2) and (3) can be handled the same as Case (1), but, given the tight bounds on processing time, the output each time Case (4) occurs will be to reject both flags.

To handle Case (1) we have to identify a whole flag lying between two partial flags. One of the quickest ways to do this is to run a window as long as the image vertically, but narrow in the horizontal direction, say, corresponding to 0.35 in. (based on the window size 1/2 of $[12.8 - 12.1]$), which is approximately $(0.35)(640)/12.8 \simeq 17$ pixels wide. This window is used look for a significant gap between a high count of 1's, and it is narrow enough to detect Case (4). For Case (1), this approach will produce high counts starting on the left of the image, then drop to very few counts (corresponding to the background) for about two inches, pick up again as the center (whole flag) is encountered, go like this for about five inches, drop again for about two inches as the next gap is encountered, then pick up again until the right border is encountered. The 1's between the two inner gaps correspond to a complete flag and are processed further by the methods discussed below; the other 1's are ignored. (A more elegant and potentially more rugged way is to determine all connected components first, and then look for vertical gaps, but time and cost are fundamental here). Cases (2) and (3) are handled in a similar manner with slightly different logic, being careful to isolate the data corresponding to an entire flag (i.e., the flag with a gap on each side). Case (4) corresponds to a gap-data-gap-data-gap sequence, but, as mentioned above, it is likely that time and cost constraints would dictate rejecting both flags as a more economical approach than increasing the complexity of the system to handle this special case. Note that this approach to extracting 1's is based on the assumption that the background is not excessively noisy. In other words, the imaging set up must be such that the background is reliably segmented as black, with acceptable noise.

With reference to Fig. 1.23, the preceding discussion has carried us through the segmentation stage. The approach followed here for description, recognition, and the use of knowledge, is twofold. For the stars we use connected component analysis. For the stripes we use signature analysis. The system knows the coordinates of two vertical lines which contain the whole flag between them. First, we do a connected components analysis on the left half of the region (to save time) and filter out all components smaller and larger than the expected size of stars, say (to give some flexibility), all components less than 9 (3×3) pixels and larger than 64 (8×8) pixels. The simplest test at this point is to count the number of remaining connected components (which we assume to

be stars). If the number is 50 we continue with the next test on the stripes. If the number is less than 50 we reject the flag. Of course, the logic can be made much more complicated than this. For instance, it could include a regularity analysis in which the relative locations of the components are analyzed. There are likely to be as many answers here as there are students in the class, but the key objective should be to base the analysis on a rugged method such as connected component analysis.

To analyze the stripes, we assume that the flags are printed on white stock material. Thus, ”dropping a stripe” means creating a white stripe twice as wide as normal. This is a simple defect detectable by running a vertical scan line in an area guaranteed to contain stripes, and then looking at the gray-level signature for the number of pulses of the right height and duration. The fact that the data is binary helps in this regard, but the scan line should be preprocessed to bridge small gaps due to noise before it is analyzed. In spite of the $\pm 15^\circ$ variation in direction, a region, say, 1in. to the right of the blue region is independent enough of the rotational variation in terms of showing only stripes along a scan line run vertically in that region.

It is important that any answer to this problem show awareness of the limits in available computation time. Since no mention is made in the problem statement about available processors, it is not possible to establish with absolute certainty if a solution will meet the requirements or not. However, the student should be expected to address this issue. The guidelines given in the preceding solution are among the fastest ways to solve the problem. A solution along these lines, and a mention that multiple systems may be required if a single system cannot meet the specifications, is an acceptable solution to the problem.