



Digital Image Processing

Second Edition

Instructor's Manual

Rafael C. Gonzalez
Richard E. Woods

Digital Image Processing

Second Edition

Instructor's Manual

(Evaluation copy—Contains only representative, partial problem solutions)

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

Copyright ©1992-2002 by Rafael C. Gonzalez and Richard E. Woods

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 11 we 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.

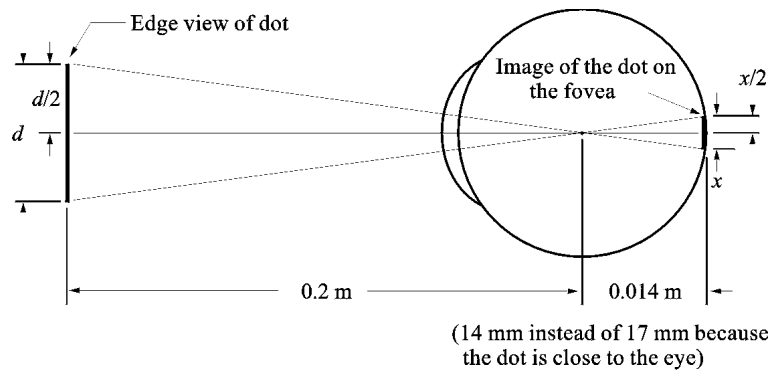


Figure P2.1

Problem 2.3

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

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 2.9

(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 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 2.11

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.

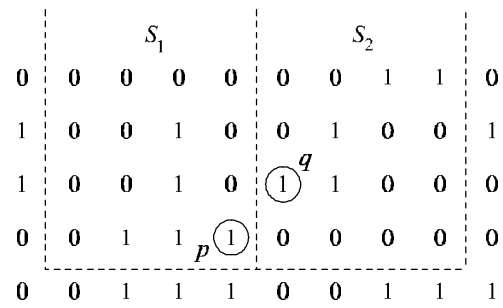


Figure P2.11

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.

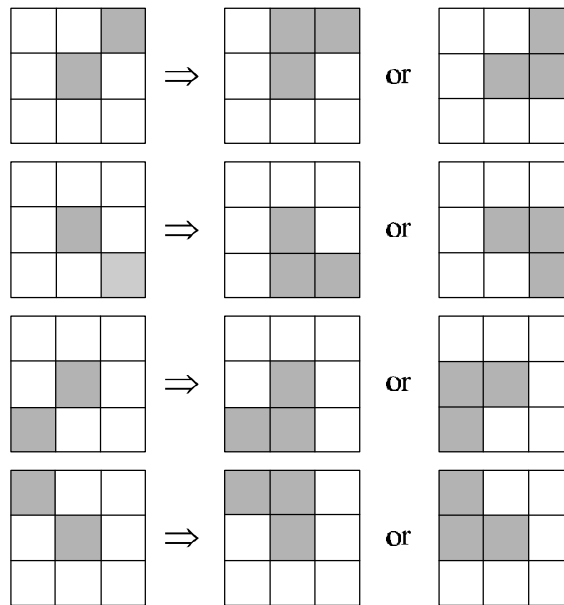


Figure P2.12

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. In this case the length of shortest m - and 8-paths is the same. 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 similarly is 4.

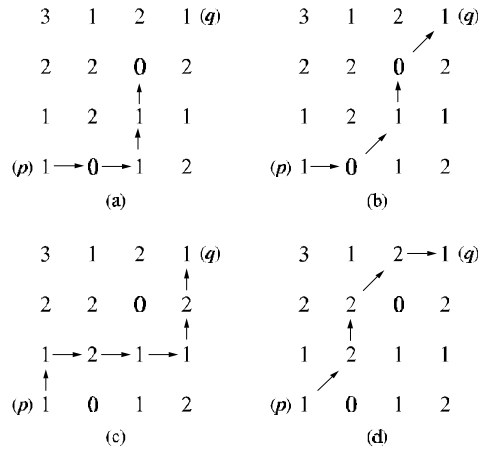


Figure P2.15

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 or may not be unique, depending on V and the values of the points along the way.

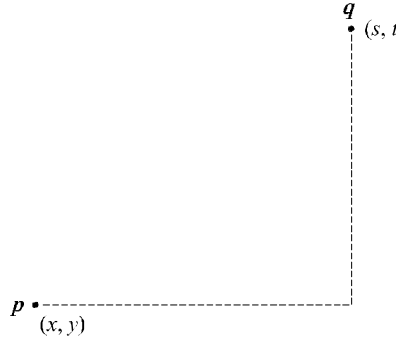


Figure P2.16

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 in 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.

3 Problem Solutions

Problem 3.2

(a)

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

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.

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 3.8

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 $[1/4, 3/4]$.

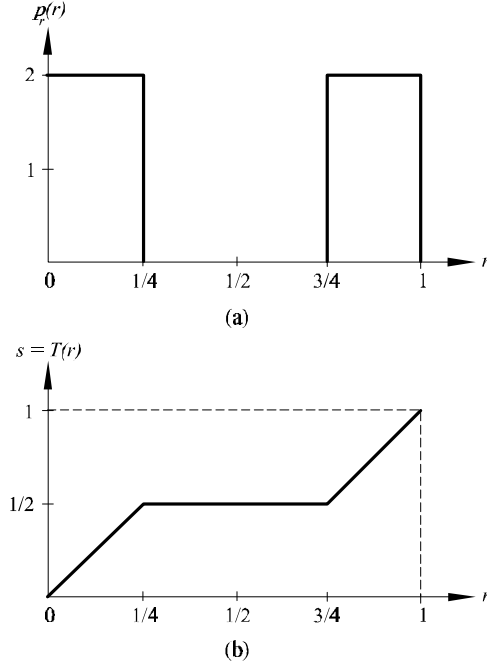


Figure P3.8.

Problem 3.9

(c) If none of the gray levels r_k , $k = 1, 2, \dots, L - 1$, 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 3.11

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 $k = 1, 2, \dots, K - 1$, 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 $k = 0, 1, \dots, K - 1$, 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 $k = 0, 1, \dots, K - 1$. 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 $k = 0, 1, \dots, K - 1$, 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.

$$\sigma_g^2 = \sigma_f^2 + \frac{1}{K^2} [\sigma_{\eta_1}^2 + \sigma_{\eta_2}^2 + \dots + \sigma_{\eta_K}^2]$$

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

$$\sigma_g^2 = \frac{K}{K^2} \sigma_\eta^2 = \frac{1}{K} \sigma_\eta^2$$

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

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 of 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 3.17

- (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.

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.

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 3.22

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 one 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 are 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.

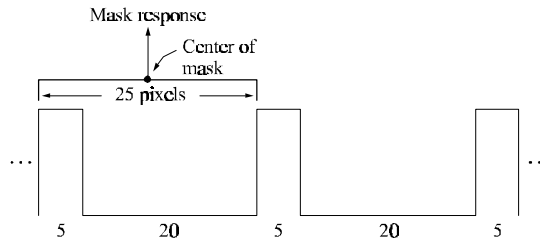


Figure P3.22

Problem 3.25

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 3.27

Consider the following equation:

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

where $\bar{f}(x, y)$ 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

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

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.

