Laboratory Projects

for

Digital Image Processing

by

Gonzalez and Woods

Prentice Hall

Upper Saddle River, NJ 07458 USA

www.prenhall/gonzalezwoods

The following sample laboratory projects are keyed to the material in the text. Several projects are designated as having "multiple uses" because their results are used in some of the other projects that follow them. They should be given assignment priority. The label [MULTIPLE USES] indicates that some or all the results of a project are used in subsequent projects.

A note on programming: The principal objectives of the following projects are (1) to teach the student how to manipulate images, and (2) to help in developing a sense of how image processing solutions are *prototyped* in software. To this end, the programming environment needed to implement these projects can consist of a truly general-purpose environment (e.g. a C++ approach) to an environment in which the projects can be implemented as a combination of existing functions with the capability to write code that can expand the capabilities of those functions. Perhaps the best exponent of that approach is MATLAB's Image Processing Toolbox (IPT). We prefer an approach that uses such a combination of capabilities because it is more representative of what the student is likely to find in practice. The software section of the book web site contains resources and links to resources that can be helpful in developing a programming environment for the projects.

Project No.	Title	Comments
Proj00-00	Suggested format for submitting project reports.	
Proj02-01	Image Printing Program Based on Halftoning.	
Proj02-02	Reducing the Number of Gray Levels in an Image.	
Proj02-03	Zooming and Shrinking Images by Pixel Replication.	
Proj02-04	Zooming and Shrinking Images by Bilinear Interpolation.	Multiple uses.

Proj03-01	Image Enhancement Using Intensity Transformations.	
Proj03-02	Histogram Equalization.	Multiple uses.
Proj03-03	Arithmetic Operations.	Multiple uses.
Proj03-04	Spatial Filtering.	Multiple uses.
Proj03-05	Enhancement Using the Laplacian.	
Proj03-06	Unsharp Masking.	
Proj04-01	Two-Dimensional Fast Fourier Transform.	Multiple uses.
Proj04-02	Fourier Spectrum and Average Value.	
Proj04-03	Lowpass Filtering.	
Proj04-04	Highpass Filtering Using a Lowpass Image.	
Proj04-05	Correlation in the Frequency Domain.	
Proj05-01	Noise Generators.	Multiple uses.
Proj05-02	Noise Reduction Using a Median Filter.	
Proj05-03	Periodic Noise Reduction Using a Notch Filter.	
Proj05-04	Parametric Wiener Filter.	
Proj06-01	Web-Safe Colors.	
Proj06-02	Pseudo-Color Image Processing.	
Proj06-03	Color Image Enhancement by Histogram Processing.	
Proj06-04	Color Image Segmentation.	
Proj07-01	One-Dimensional Discrete Wavelet Transforms.	Multiple uses.
Proj07-02	Two-dimensional Discrete Wavelet Transforms.	Multiple uses.
Proj07-03	Wavelet Transform Modifications.	
Proj07-04	Image De-Noising.	
Proj08-01	Objective Fidelity Criteria.	Multiple uses.
Proj08-02	Image Entropy.	
Proj08-03	Transform Coding.	
Proj08-04	Wavelet Coding.	
Proj09-01	Morphological and Other Set Operations.	Multiple uses.
Proj09-02	Boundary Extraction.	Multiple uses.

Proj09-03	Connected Components	Multiple uses.
Proj09-04	Morphological Solution to Problem 9.27	
Proj10-01	Edge Detection Combined with Smoothing and Thresholding.	
Proj10-02	Global Thresholding.	Multiple uses.
Proj10-03	Optimum Thresholding.	
Proj10-04	Region Growing.	
Proj11-01	Skeletons.	
Proj11-02	Fourier Descriptors.	Multiple uses.
Proj11-03	Texture.	
Proj11-04	Principal Components.	
Proj12-01	Generating Pattern Classes.	Multiple uses.
Proj12-02	Minimum Distance Classifier.	
Proj12-03	Bayes Classifier.	
Proj12-04	Perceptron Classifier.	

PROJECT 00-00

Suggested Format for Submitting Project Reports

Because laboratory projects are in addition to course work, it is suggested that project reports be kept short, and be organized in a uniform manner to simplify grading. The following format achieves these objectives.

Page 1. *Cover Page*. Typed or printed neatly.

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

Page 2. *Technical discussion*. One to two pages (max). This section should include the techniques used and the principal equations (if any) implemented.

Page 3 (or 4). *Discussion of results*. One to two pages (max). A discussion of results should include major findings in terms of the project objectives, and make clear reference to any images generated.

Results. Includes all the images generated in the project. Number images individually so they can be referenced in the preceding discussions.

Appendix. *Program listings*. Includes listings of all programs written by the student. Standard routines and other material obtained from other sources should be acknowledged by name, but their listings should not be included.

Layout. The entire report must be in standard sheet size format (8.5 x 11 inches in the U.S.) All sheets should be stapled in three locations to form a binding booklet-like support on the left margin. Alternatively, sheets can be assembled using a commercial plastic binding product with a clear plastic cover.

A note on program implementation: As noted earlier, the objective of the computer programs used in the following projects is to teach the student how to manipulate images. There are numerous packages that perform some of the functions required to implement the projects. However, the use of "canned" routines as the only method to implement an entire project is discouraged. For example, if the students are using MATLAB and the Image Processing Toolbox, a balanced approach is to use MATLAB's programming environment to write M functions to implement the projects, using some of MATLAB's own functions in the process. A good example is the implementation of the 2-D Fourier Fast Transform. The student should use the MATLAB function that computes the 2-D FFT directly, but write functions for operations such as centering the transform, multiplying it by a filter function, and obtaining the spectrum.

PROJECT 02-01

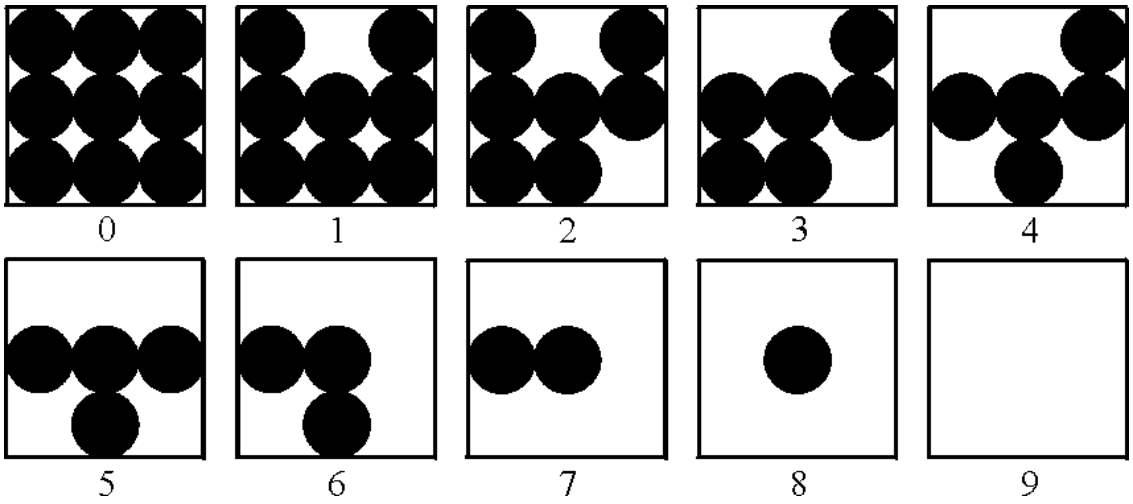
Image Printing Program Based on Halftoning

The following figure shows ten shades of gray approximated by dot patterns. Each gray level is represented by a 3 x 3 pattern of black and white dots. A 3 x 3 area full of black dots is the approximation to gray-level *black*, or 0. Similarly, a 3 x 3 area of white dots represents gray level 9, or *white*. The other dot patterns are approximations to gray levels in between these two extremes. A gray-level printing scheme based on dots patterns such as these is called "halftoning." Note that each pixel in an input image will correspond to 3 x 3 pixels on the printed image, so spatial resolution will be reduced to 33% of the original in both the vertical and horizontal direction. Size scaling as required in (a) may further reduce resolution, depending on the size of the input image.

(a) Write a halftoning computer program for printing gray-scale images based on the dot patterns just discussed. Your program must be able to scale the size of an input image so that it does not exceed the area available in a sheet of size 8.5 x 11 inches (21.6 x 27.9 cm). Your program must also scale the gray levels of the input image to span the full halftoning range.

(b) Write a program to generate a test pattern image consisting of a gray scale wedge of size 256 x 256, whose first column is all 0's, the next column is all 1's, and so on, with the last column being 255's. Print this image using your gray-scale printing program.

(c) Print book Figs. 2.22(a) through (c) using your gray-scale printing program. Do your results agree with the conclusions arrived at in the text in pgs. 61-62 and Fig. 2.23? Explain. You will need to download Figs. 2.22(a) through (c).



PROJECT 02-02

Reducing the Number of Gray Levels in an Image

(a) Write a computer program capable of reducing the number of gray levels in a image from 256 to 2, in integer powers of 2. The desired number of gray levels needs to be a variable input to your program.

(b) Download Fig. 2.21(a) and duplicate the results shown in Fig. 2.21 of the book.

PROJECT 02-03

Zooming and Shrinking Images by Pixel Replication

(a) Write a computer program capable of zooming and shrinking an image by pixel replication. Assume that the desired zoom/shrink factors are integers. You may ignore aliasing effects. You will need to download Fig. 2.19(a).

(b) Download Fig. 2.19 (a) and use your program to shrink the image from 1024 x 1024 to 256 x 256 pixels.

(c) Use your program to zoom the image in (b) back to 1024 x 1024. Explain the reasons for their differences.

PROJECT 02-04 [Multiple Uses]

Zooming and Shrinking Images by Bilinear Interpolation

(a) Write a computer program capable of zooming and shrinking an image by bilinear interpolation. The input to your program is the desired size of the resulting image in the horizontal and vertical direction. You may ignore aliasing effects.

(b) Download Fig. 2.19(a) and use your program to shrink this image from 1024 x 1024 to 256 x 256 pixels.

(c) Use your program to zoom the image in (b) back to 1024 x 1024. Explain the reasons for their differences.

PROJECT 03-01

Image Enhancement Using Intensity Transformations

The focus of this project is to experiment with intensity transformations to enhance an image. Download Fig. 3.8(a) and enhance it using

(a) The log transformation of Eq. (3.2-2).

(b) A power-law transformation of the form shown in Eq. (3.2-3).

In (a) the only free parameter is *c*, but in (b) there are two parameters, *c* and *r* for which values have to be selected. As in most enhancement tasks, experimentation is a must. The objective of this project is to obtain the best visual enhancement possible with the methods in (a) and (b). Once (according to your judgment) you have the best visual result for each transformation, explain the reasons for the major differences between them.

PROJECT 03-02 [Multiple Uses]

Histogram Equalization

(a) Write a computer program for computing the histogram of an image.

(b) Implement the histogram equalization technique discussed in Section 3.3.1.

(c) Download Fig. 3.8(a) and perform histogram equalization on it.

As a minimum, your report should include the original image, a plot of its histogram, a plot of the histogram-equalization transformation function, the enhanced image, and a plot of its histogram. Use this information to explain why the resulting image was enhanced as it was.

PROJECT 03-03 [Multiple Uses]

Arithmetic Operations

Write a computer program capable of performing the four arithmetic operations between two images. This project is generic, in the sense that it will be used in other projects to follow. (See comments on pages 112 and 116 regarding scaling). In addition to multiplying two images, your multiplication function must be able to handle multiplication of an image by a constant.

PROJECT 03-04 [Multiple Uses]

Spatial Filtering

Write program to perform spatial filtering of an image (see Section 3.5 regarding implementation). You can fix the size of the spatial mask at 3 x 3, but the coefficients need to be variables that can be input into your program. This project is generic, in the sense that it will be used in other projects to follow.

PROJECT 03-05

Enhancement Using the Laplacian

(a) Use the programs developed in Projects 03-03 and 03-04 to implement the Laplacian enhancement technique described in connection with Eq. (3.7-5). Use the mask shown in Fig. 3.39(d).

(b) Duplicate the results in Fig. 3.40. You will need to download Fig. 3.40(a).

PROJECT 03-06

Unsharp Masking

(a) Use the programs developed in Projects 03-03 and 03-04 to implement high-boost filtering, as given in Eq. (3.7-8). The averaging part of the process should be done using the mask in Fig. 3.34(a).

(b) Download Fig. 3.43(a) and enhance it using the program you developed in (a). Your objective is to choose constant *A* so that your result visually approximates Fig. 3.43(d).

PROJECT 04-01 [Multiple Uses]

Two-Dimensional Fast Fourier Transform

The purpose of this project is to develop a 2-D FFT program "package" that will be used in several other projects that follow. Your implementation must have the capabilities to:

(a) Multiply the input image by $(-1)^{x+y}$ to center the transform for filtering.

(b) Multiply the resulting (complex) array by a real function (in the sense that the the real coefficients multiply both the real and imaginary parts of the transforms). Recall that multiplication of two images is done on pairs of corresponding elements.

(c) Compute the inverse Fourier transform.

(d) Multiply the result by $(-1)^{x+y}$ and take the real part.

(e) Compute the spectrum.

Basically, this project implements Fig. 4.5. If you are using MATLAB, then your Fourier transform program will not be limited to images whose size are integer powers of 2. If you are implementing the program yourself, then the FFT routine you are using may be limited to integer powers of 2. In this case, you may need to zoom or shrink an image to the proper size by using the program you developed in Project 02-04.

An approximation: To simplify this and the following projects (with the exception of Project 04-05), you may ignore image padding (Section 4.6.3). Although your results will not be strictly correct, significant simplifications will be gained not only in image sizes, but also in the need for cropping the final result. The principles will not be affected by this approximation.

PROJECT 04-02

Fourier Spectrum and Average Value

(a) Download Fig. 4.18(a) and compute its (centered) Fourier spectrum.

(b) Display the spectrum.

(c) Use your result in (a) to compute the average value of the image.

PROJECT 04-03

Lowpass Filtering

(a) Implement the Gaussian lowpass filter in Eq. (4.3-7). You must be able to specify the size, $M \times N$, of the resulting 2D function. In addition, you must be able to specify where the 2D location of the center of the Gaussian function.

(b) Download Fig. 4.11(a) [this image is the same as Fig. 4.18(a)] and lowpass filter it to obtain Fig. 4.18(c).

PROJECT 04-04

Highpass Filtering Using a Lowpass Image

(a) Subtract your image in Project 04-03(b) from the original to obtain a sharpened image, as in Eq. (4.4-14). You will note that the resulting image does not resemble the Gaussian highpass results in Fig. 4.26. Explain why this is so.

(b) Adjust the variance of your Gaussian lowpass filter until the result obtained by image subtraction looks similar to Fig. 4.26(c). Explain your result.

PROJECT 04-05

Correlation in the Frequency Domain

Download Figs. 4.41(a) and (b) and duplicate Example 4.11 to obtain Fig. 4.41(e). Give the (x,y) coordinates of the location of the maximum value in the 2D correlation function. There is no need to plot the profile in Fig. 4.41(f).

PROJECT 05-01 [Multiple Uses]

Noise Generators

This is a generic project, in the sense that the programs developed here are used in several of the projects that follow. See Fig. 5.2 for the shapes and parameters of the following noise probability density functions.

(a) Find (or develop) a program to add Gaussian noise to an image. You must be able to specify the noise mean and variance.

(b) Find (or develop) a program to add salt-and-pepper (impulse) noise to an image. You must be able to specify the probabilities of each of the two noise components.

PROJECT 05-02

Noise Reduction Using a Median Filter

(a) Modify the program that you developed in Project 03-04 to perform 3 x 3 median filtering.

(b) Download Fig. 5.7(a) and add salt-and-pepper noise to it, with $P_a = P_b = 0.2$.

(c) Apply median filtering to the image in (b). Explain the major differences between your result and Fig. 5.10(b).

PROJECT 05-03

Periodic Noise Reduction Using a Notch Filter

(a) Write a program that implements sinusoidal noise of the form given in Problem 5.14. The inputs to the program must be the amplitude, A, and the two frequency components u_0 and v_0 shown in the problem equation.

(b) Download image 5.26(a) and add sinusoidal noise to it, with $u_0 = M/2$ (the image is square) and $v_0 = 0$. The value of A must be high enough for the noise to be quite visible in the image.

(c) Compute and display the spectrum of the image. If the FFT program you developed in Project 4.01 can only handle images of size equal to an integer power

of 2, reduce the size of the image to 512 x 512 or 256 x 256 using the program from Project 02-04. Resize the image before adding noise to it.

(d) Notch-filter the image using a notch filter of the form shown in Fig. 5.19(c).

PROJECT 05-04

Parametric Wiener Filter

(a) Implement a blurring filter as in Eq. (5.6-11).

(b) Blur image 5.26(a) in the $+45^{o}$ direction using $T = 1$, as in Fig. 5.26(b).

(c) Add Gaussian noise of 0 mean and variance of 10 pixels to the blurred image.

(d) Restore the image using the parametric Wiener filter given in Eq. (5.8-3).

PROJECT 06-01

Web-Safe Colors

In order to complete this project, it is necessary that you find a program capable of generating the RGB component images for a given jpg color image. For example, MATLAB's Image Processing Toolbox can do this, but you can also do it with image editing programs like Adobe's Photo-Shop or Corel's Photo-Paint. It is acceptable for the purposes of this project to convert an image to RGB (and back) manually.

(a) Write a computer program that converts an arbitrary RGB color image to a web-safe RGB image (see Fig. 6.10 for a definition of web-safe colors).

(b) Download the image in Fig. 6.8 and convert it to a web-safe RGB color image. Figure 6.8 is given in jpg format, so convert your result back to jpg (see comments at the beginning of this project). Explain the differences between your result and Fig. 6.8.

PROJECT 06-02

Pseudo-Color Image Processing

(a) Implement Fig. 6.23, with the characteristic that you can specify two ranges of gray-level values for the input image and your program will output an RGB image whose pixels have a specified color corresponding to one range of gray levels in the input image, and the remaining pixels in the RGB image have the same shade of gray as they had in the input image. You can limit the input colors to all the colors in Fig. 6.4(a).

(b) Download the image in Fig. 1.10(4) and process it with your program so that the river appears yellow and the rest of the pixels are the same shades of gray as in the input image. It is acceptable to have isolated specs in the image that also appear yellow, but these should be kept as few as possible by proper choice of the two gray-level bands that you input into your program.

PROJECT 06-03

Color Image Enhancement by Histogram Processing

(a) Download the dark-stream color picture in Fig. 6.35 (this image is labeled Fig. 6.35(05) in the image gallery for Chapter 6). Convert the image to RGB (see comments at the beginning of Project 06-01). Histogram-equalize the R, G, and B images separately using the histogram-equalization program and convert the image back to jpg format.

(b) Form an average histogram from the three histograms in (a) and use it as the basis to obtain a single histogram equalization intensity transformation function. Apply this function to the R, G, and B components individually, and convert the results to jpg. Compare and explain the differences in the jpg images in (a) and (b).

PROJECT 06-04

Color Image Segmentation

Download Fig. 6.28(b) and duplicate Example 6.15, but segment instead the darkest regions in the image.

PROJECT 07-01 [Multiple Uses]

One-Dimensional Discrete Wavelet Transforms

The purpose of this project is to build a rudimentary wavelet transform package using Haar wavelets that can be used in projects that follow. You will use an "averaging and differencing" approach that is unique to Haar basis functions. As an introduction to the method, consider the function in Example 7.8. The necessary "averaging and differencing" operations are:

Step 1: Compute two-point sums and differences across the function vector and divide the results by the square root of 2. Since f(x) = {1, 4, -3, 0}, we get

$$\{1 + 4, -3 + 0, 1 - 4, -3 - 0\} / 1.414$$

$$\{5, -3, -3, -3\} / 1.414$$

Note that the sums are positioned consecutively at the beginning of the intermediate result and followed by the corresponding differences.

Step 2: Repeat the process over the sums computed in the first step to get

$$\{[5 + (-3)] / 1.414, [5 - (-3)] / 1.414, -3, -3\} / 1.414$$

$$\{2 / 1.414, 8 / 1.414, -3, -3\} / 1.414$$

$$\{1, 4, -2.121, -2.121\}$$

The coefficients of the final vector match those in Example 7.8. The two-step computation generates a two-scale DWT with respect to Haar wavelets. It can be generalized to higher scales and functions with more than 4 points. Moreover, an inverse DWT can be computed by reversing the process.

(a) Write a program to compute j-scale DWTs with respect to Haar wavelets. Let scale be an input parameter and assume a $2M$ point discrete one-dimensional function. Use the averaging and differencing approach described above.

(b) Write a program to compute the inverse DWT of a j-scale DWT based on Haar wavelets.

(c) Test your programs using the function in Example 7.8.

PROJECT 07-02 [Multiple Uses]

Two-dimensional Discrete Wavelet Transforms

(a) Use the routines developed in Project 07-01 to write a program that computes *j*-scale two-dimensional DWTs with Haar wavelets. Base your routine on the discussion of separable wavelets and two-dimensional wavelet transforms in Section 7.5.

(b) Download the image of Fig. 7.1 and use your program to generate the three-scale DWT (or Haar transform) shown in Fig. 7.8(a). Label the various detail and approximation coefficients that make up the transform and indicate their scales.

(c) Write a program to compute the inverse two-dimensional DWT with respect to Haar wavelets and use it to reconstruct the original image from the wavelet decomposition in (b).

(d) Write a program to scale the detail coefficients of the DWT in (b) so that the underlying structure is more visible. The approximation coefficients do not need to be scaled.

PROJECT 07-03

Wavelet Transform Modifications

Download the image from Fig. 4.18(a) and pad it with 0s to obtain a 512 x 512 array. Use the two-dimensional DWT program developed in Project 07-02 to compute the transform of the padded image at a variety of scales between 1 and 9.

(a) Zero the approximation coefficients of the generated transforms and record your observations regarding subsequently reconstructed images. That is, compute the inverse transforms of the decompositions after the approximation coefficients have been zeroed and record the impact on the transform modifications.

(b) Repeat the process in (a) but zero the horizontal detail coefficients instead.

(c) Repeat the process in (a) but zero the vertical detail coefficients instead.

(d) Repeat the process in (a) but zero both the horizontal and vertical detail coefficients.

PROJECT 07-04

Image De-Noising

Download the noisy MRI image of Fig. 7.26(a) and de-noise it using a Haar-based DWT.

PROJECT 08-01 [Multiple Uses]

Objective Fidelity Criteria

(a) Write a program to compute the root-mean-square error [see Eq. (8.1-8)] and mean-square signal-to-noise ratio [per Eq. (8.1-9)] of a compressed- decompressed image. This project is generic in the sense that it will be used in other projects that follow.

(b) Download the image of Fig. 8.4(a) and write a program to generate the results in the (b) and (c) parts of the figure. Use your fidelity criteria program to characterize any loss of visual information and comment on your results.

PROJECT 08-02

Image Entropy

(a) Write a program to compute the first and second order entropy estimates of an image.

(b) Download the images of Figures 8.14(a) and (b) and use your program to estimate their entropies. Interpret the results in view of the compression results given in Tables 8.8 and 8.9.

PROJECT 08-03

Transform Coding

(a) Write a program to compute the information loss associated with the following transform coding schemes:

	Case 1	**Case 2**
Transform:	Fourier	Cosine
Subimage Size:	8 x 8	8 x 8
Bit Allocation:	8-largest coding	8-largest coding

Use the routines developed in Project 08-01 to quantify the loss of information. Download the image in Fig. 8.23 and use the program to compare Cases 1 and 2.

(b) Gradually decrease the number of retained coefficients until the reconstruction error for Case 2 becomes objectionable. That is, try 7-largest, 6-largest, … coding as the bit allocation method.

PROJECT 08-04

Wavelet Coding

Download and compress the image of Fig. 8.23 using the Haar-based DWT program of Project 07-02. Use various scales while truncating the detail coefficients to achieve compression. Quantify the reconstruction error using the program from Project 08-01 and compare both the computed error and wavelet-based compression performance to the results from Proj08-03.

PROJECT 09-01 [Multiple Uses]

Morphological and Other Set Operations

(a) Write a computer program capable of performing binary dilation and erosion with an arbitrary (i.e., specified) structuring element of size 3 x 3.

(b) Write a computer program for performing set intersection, differencing, and complementation.

PROJECT 09-02 [Multiple Uses]

Boundary Extraction

(a) Use your results from Project 09-01 to implement morphological boundary extraction as in Eq. (9.5-1).

(b) Download Fig. Prob9.20(left) and extract the boundaries of the characters.

PROJECT 09-03 [Multiple Uses]

Connected Components

(a) Use your results from Project 09-01 to write a computer program capable of extracting (and counting) the connected components from a binary image.

(b) Download Fig. Prob9.27 and use your program in (a) to count the number of connected components.

PROJECT 09-04

Morphological Solution to Problem 9.27

Use previous results plus any new required techniques and write a computer program to completely solve Problem 9.27. Solve the problem.

PROJECT 10-01

Edge Detection Combined with Smoothing and Thresholding

(a) Extend the program from Project 03-04 to compute the Sobel gradient using the masks in Fig. 10.8. Your program should implement Eq. (10.1-12), and have the option of outputting a binary image by comparing each gradient point against a specified threshold, T.

(b) Download Fig. 10.15(a). By combining smoothing with a 3 x 3 mask from Project 03-04 and your program from (a), process Fig. 10.15(a) and produce a binary image that isolates (segments) the large blood vessel in the center of the image. This will require repeated trials of smoothing and choices of T. Looking at the histogram (Project 03-02) of the gradient image before it is thresholded will help you select a value for T.

PROJECT 10-02 [Multiple Uses]

Global Thresholding

(a) Write a global thresholding program in which the threshold is estimated automatically using the procedure discussed in Section 10.3.3. The output of your program should be a segmented (binary) image.

(b) Download Fig. 10.27(a) and segment the object using your thresholding program.

PROJECT 10-03

Optimum Thresholding

(a) Implement the optimum thresholding approach discussed in Section 10.3-5. Assume Gaussian densities in which the variances of the objects and background are the same. In addition to an image, the inputs to your program are as shown in Eq. (10.3-14).

(b) Write a program that, given an image patch, computes the mean and variance of the pixels in the patch.

(c) Download Fig. 10.27(a) and select a small patch in the region of the object and also of the background to estimate the mean and variance of each. Compute the value of a single variance by averaging the two variances just obtained. Obtain the probabilities P_1 and P_2 by estimating (manually) the relative areas occupied by the object and background. Input the parameters thus obtained into your program and segment Fig. 10.27(a). If you were assigned Project 10-02 as well, compare the results.

PROJECT 10-04

Region Growing

(a) Implement a region-growing procedure (see Section 10.4.2) for segmenting an image into two regions. *Hint*: One approach is to grow only one region. Then, by default, the other region is the set of points left over after growth of the first region stops. At any step in the growth process, a new point is appended to the region only if a set of pre-established predicates is satisfied. For example, let s represent the *average* gray level of the region grown thus far at any iterative step in the procedure. One possibility is to append a new point to the region if (a) its gray level does not differ by more than a specified constant, k, from s, and (b) the point is connected (see Section 2.5.2) to the region grown thus far.

(b) If Project 10-03 was not assigned, write a program that, given an image patch, computes the mean and variance of the pixels in the patch.

(c) Download Fig. 10.27(a). Select a small patch in the object region and compute the mean and variance. Let the mean be the starting value of s. Use a multiple of the standard deviation as the value of k.

(d) Segment the image by region growing.

PROJECT 11-01

Skeletons

(a) Implement the skeletonizing procedure described in Section 11.1.5.

(b) Download Fig. 10.25, threshold it using your program from Project 10-02, and obtain its skeleton.

PROJECT 11-02 [Multiple Uses]

Fourier Descriptors

(a) Implement the Fourier descriptor scheme developed in Section 11.2.3.

(b) Download Figs. 12.18(a1) and (a2) and obtain the boundaries of each using your program from Project 09-02. Choose 256 points to represent each boundary. The 256 points should be as equally-spaced as reasonably possible.

(c) Obtain the Fourier descriptors of each figure (be sure to start at approximately the same point in each figure). Approximate the boundaries using the first 16, 32, 164 and 128 of the coefficients (see Example 11.3 for guidance). Plot each result.

PROJECT 11-03

Texture

(a) Implement the statistical texture measures described in Section 11.3.3.

(b) Download Figs. 1.14(a), (d) and (e), and extract a 100 x 100 segment from the lower, right quadrant of each image.

(c) Compute the statistical measures of the subimages using the measures described in Table 11.2. Present your results in the same table format and discuss.

PROJECT 11-04

Principal Components

(a) Implement the principal components transform discussed in Section 11.4. The objective is to be able to use Eqs. (11.4-11) and (11.4.-12). You will need routines to compute the eigenvalues and eigenvectors of the covariance matrix.

(b) Download Figs. 1.10(1) through 1.10(6) and duplicate the sequence of operations described in Example 11.10, including the images and tables. It is not necessary to duplicate Fig. 11.27.

PROJECT 12-01 [Multiple Uses]

Generating Pattern Classes

(a) Refer to your results in Project 11-02. Choose the smallest number of descriptors that preserved the basic differences between the two figures. This will give you two sequences of Fourier descriptors, one for each figure. Express each set of coefficients as a vector, $\mathbf{x}_A=(x_1, x_2, \ldots, x_n)^T$ and $\mathbf{x}_B=(x_1, x_2, \ldots, x_n)^T$, where n is the number of descriptors.

(b) Create two pattern *classes*, class *A* and class *B*, by adding noise (Project 05-01) to the components of each vector. Use Gaussian noise of zero mean and standard deviation equal to the maximum component of each vector divided by 10. Generate 100 samples each of class *A* and *B*. Call the aggregate of the two sets the *training set*. Generate an additional 100 samples of each class and call the aggregate of these two sets the *test set*.

PROJECT 12-02

Minimum Distance Classifier

(a) Implement the minimum distance classifier (for two classes) discussed in Section 12.2.1.

(b) Compute the classifier parameters using the training set developed in Project 12-01.

(c) Establish the classifier recognition performance by determining the percent of patterns from the test set that are classified correctly.

PROJECT 12-03

Bayes Classifier

(a) Implement the Bayes classifier (for two classes) discussed in Section 12.2.2. Assume Gaussian pattern classes.

(b) Compute the classifier parameters using the training set developed in Project 12-01.

(c) Establish the classifier recognition performance by determining the percent of patterns from the test set that are classified correctly.

PROJECT 12-04

Perceptron Classifier

(a) Implement the perceptron classifier (for two classes) discussed in Section 12.2.3.

(b) Compute the classifier parameters by training, using the training set developed in Project 12-01. Since training convergence can be guaranteed only if the classes are linearly separable, and this is not known a priori, establish a limit that stops the

algorithm after it executes 10n passes through the entire training set, where n is the dimensionality of the pattern vectors.

(c) Establish the classifier recognition performance by determining the percent of patterns from the test set that are classified correctly.