پروژه های عملی درس پردازش تصویر

دانشگاه خاتم، نیمسال ۲-۱۴۰۰

مهلت تحویل: دوشنبه دهم مرداد ۱۴۰۱ ساعت ۲۳:۵۵

- ✓ پروژه های زیر مربوط به فصول ۲ تا ۶ و فصب ۸ کتاب است.
- ✓ پروژه های فوق را با پایتون پیاده سازی کرده و با فرمت پیشنهادی زیر به آدرس ایمیل manzuri@sharif.edu ارسال کنید.
- ✓ هر دانشجو باید مستقلا پروژه ها را انجام دهد و انجام گروهی غیر قابل خواهد بود.

Suggested Format for Submitting Project Reports

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.

- ✓ **Create a main folder with your name and student number**
- ✓ **In main folder create several sub-folders with the name of Project number**
- ✓ **After completing the project, do as follows:**
  - **Write a technical report in which you should discuss the results. It should include major findings in terms of the project objectives, and make clear reference to any images generated.**
  - **Includes all the images generated in the project. Number images individually so they can be referenced in the preceding discussions.**
- ✓ **Include the file of your written program in the sub-folder.**

Projects:

PROJECT 02-01

Image Printing Program Based on Halftoning

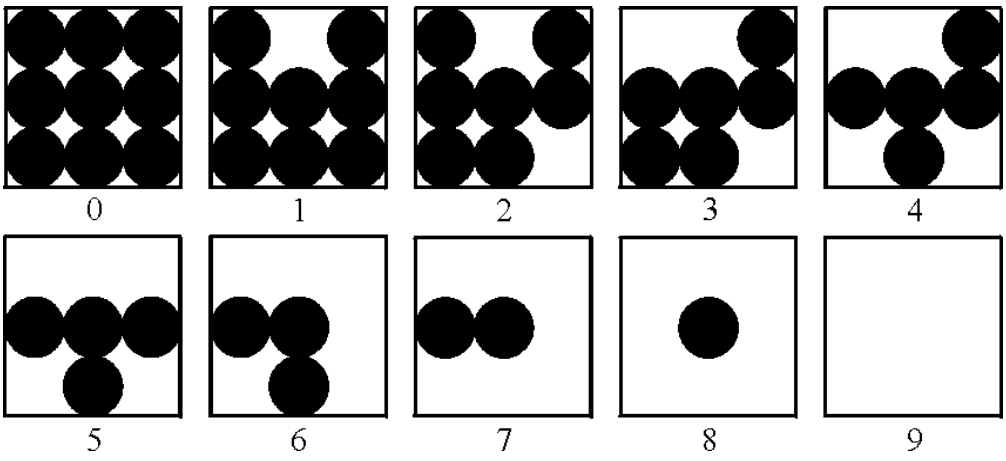
The following figure shows ten shades of gray approximated by dot patterns. Each gray level is represented by a 3 x 3 pattern of black and white dots. A 3 x 3 area full of black dots is the approximation to gray-level

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. 64-65 and Fig. 2.23? Explain. You can download the required figures from the book web site.



PROJECT 02-02

Reducing the Number of Intensity Levels in an Image

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

(b) Download Fig. 2.21(a) from the book web site 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.

(b) Download Fig. 2.20(a) from the book web site and use your program to shrink the image by a factor of 10.

(c) Use your program to zoom the image in (b) back to the resolution of the original. Explain the reasons for their differences.

PROJECT 02-04

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 resolution (in dpi) of the resulting image.

(b) Download Fig. 2.20(a) from the book web site and use your program to shrink this from 1250 dpi t0 100 dpi.

(c) Use your program to zoom the image in (b) back to 1250 dpi. Explain the reasons for their differences.

PROJECT 02-05

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 03-01

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) from the book web site 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

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) from the book web site 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

Spatial Filtering

Write program to perform spatial filtering of an image (see Section 3.4 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 03-04

Enhancement Using the Laplacian

(a) Use the programs developed in Project 03-03 to implement the Laplacian enhancement technique described in connection with Eq. (3.6-7).

(b) Duplicate the results in Fig. 3.38. You can download the original image from the book web site.

PROJECT 03-05

Unsharp Masking

(a) Use the program developed in Project 03-03 to implement high-boost filtering, as given in Eq. (3.6-9). The averaging part of the process should be done using the mask in Fig. 3.32(a).

(b) Download Fig. 3.40(a) from the book web site and enhance it using the program you developed in (a). Your objective is to approximate the result in Fig. 3.40(e).

PROJECT 04-01

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 filter 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 the steps in Section 4.7.3. 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. See the Software section of the book web site to find a 1-D FFT routine. Then use the method discussed in Sections 4.11.1 and 4.11.2 for computing the 2-D FFT.

An approximation: To simplify this and the following projects (with the exception of Project 04-05), you may ignore image padding (Section 4.6.6). 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. 4.41(a) from the book web site 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. (4.8-7). You must be able to specify the size, M x N, of the resulting 2D function. In addition, you must be able to specify the location of the center of the Gaussian function.

(b) Download Fig. 4.41(a) from the book web site and lowpass filter it to duplicate the results in Fig. 4.48.

PROJECT 04-04

Highpass Filtering

(a) Implement the Gaussian highpass filter of Eq. (4.9-4). (Note that, if you did project 04-03, you can use basically the same program to generate highpass filters.)

(b) Download Fig. 4.41(a) from the book web site and highpass filter it to duplicate the results in Fig. 4.56.

PROJECT 04-05

Highpass Filtering Combined with Thresholding

Download Fig. 4.57(a) from the book web site and use your program from Project 04-04 to approximate the results in Fig. 4.57 (note that you will be using a Gaussian, instead of a Butterworth, filter.

PROJECT 05-01

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.

Note: Your program must be capable also of generating random numbers organized as a 1-D array of specified size (including a single random number), as you will need it later in Chapter 12 to add noise to elements of a vector.

PROJECT 05-02

Noise Reduction Using a Median Filter

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

(b) Download Fig. 5.7(a) from the book web site and add salt-and-pepper noise to it, with Pa = Pb = 0.2.

(c) Apply median filtering to the image in (b). Explain any 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 u0 and v0 shown in the problem equation.

(b) Download image 5.26(a) from the book web site and add sinusoidal noise to it, with u0 = M/2 (the image is square) and v0 = 0. The value of A must be high enough for the noise to be clearly 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-degree 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 tif 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 PhotoPaint. 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 from the book web site and convert it to a web-safe RGB color image. Figure 6.8 is given in tif format, so convert your result back to tif (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) from the book web site 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 from the book web site. 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 from Project 03-02 and convert the image back to tif 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 tif images in (a) and (b).

PROJECT 06-04

Color Image Segmentation

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

PROJECT 08-01

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.