Image Processing I

EECE/CS 253 Fall 2011

Laboratory 5: Quantization, Sampling, Rotation, and Warping

Due at 12:00 midnight on Thursday 3 November 2011

The goals of this lab are to learn (1) about the effects of image quantization and the use of noise for the visual enhancement of quantized images, (2) how to perform a nearest-neighbor resampling of an image, (3) how to perform bilinear interpolation, (4) how to perform image rotation, and (5) how to perform 2D linear warping of an image

In this lab assignment (and all the others) you will be writing your own image processing functions. You may not use the functions from the `matlab` image processing toolkit by themselves or within your functions to solve the problems in the labs. You may compare the results of your own functions against those of the IP toolkit for your own information but those results may not be used in your lab reports.

Help can be gotten on any `matlab` command (or function) by typing `help command_name`, *e.g.* to get help on `imwrite()`, type `help imwrite` at the command prompt. Another way is to press the <F1> key, select the **index** tab in the window, and type the command name into the search box.

In completing this assignment (and the others) if specific images are not supplied, you may use any images you like providing that they are of the type specified by the problem description (*e.g.* 24-bit truecolor). Please do not use images that are obscene or gruesome. In general, an image is OK if you could show it to your grandmother without upsetting her or embarrassing yourself. A wide variety of images are available under creative commons license on the web site `www.flickr.com`. If you are not the photographer, be sure to credit the person who did take the picture.

1. Write a function to quantize a color image. That is, it should input a nominally 8-bit per pixel per band image (a truecolor image of class `uint8`) and return an image that is quantized to n bits where $n \in \{1, 2, \ldots, 7\}$. The function should input a color image of any size and return a new color image of the same size. The function also should accept an integer argument that specifies the number of bits per pixel per band in the output. The function call should look something like

   ```
   J = QuantImg(I,n);
   ```

   where `I` is the input image and `n` is the number of bits per pixel per band in the resulting image. The output image, `J`, should remain class `uint8` even though each pixel will have fewer than 8 bits of intensity resolution.

   (a) Test your function on a single original image and produce 7 new images from it, one for each quantization level (7 bits through 1 bit per pixel per band). Make a table of quantization levels and number of colors in the the result for a one band image and a 3-band image.

   Before you display or save the `n = 1` image you should first scale the results so that the individual pixels have values 0 and 255. Otherwise the image will appear dark. I suggest using a small image (say around 256×256 in linear dimension) for this part so that they can all be displayed on 1 or 2 pages in your report.

   (b) Experiment with adding noise to the image, both before and after quantization. For the experiments use a quantization level of 3 bits. To add the noise before quantizing do, for example,

   ```
   I = uint8(double(I)+sigma*randn(size(I)));
   ```

   before applying your quantization routine. When adding the noise after quantization of the image, be sure to quantize the noise to 3 bits before adding it to the 3-bit quantized image.

Determine which which value of sigma produces the best results in terms of the reduction of false contours and determine whether it is better to add the noise before or after quantization of the image. I suggest you first try powers of 2 for sigma then refine your search once you've bracketed the results.

2. Write a program to implement a nearest neighbor resampling of an image. The function's arguments should be the input image and the number of rows and columns in the output image.

   (a) Find a truecolor image, I, with dimensions in the 512 to 768 range. Use your function to reduce the image to 1/4 size and 11/16 size. Then use the function to increase the size of the 1/4 image to its original size. That is, if I is $R \times C \times 3$, nearest-neighbor downsample I to get a $\lfloor R/4 \rfloor \times \lfloor C/4 \rfloor$ image, J. Then use the function to upsample J to a $4\lfloor R/4 \rfloor \times 4\lfloor C/4 \rfloor$ image, K. Do the same for the 11/16 size. Display the results next to the original image and comment on their appearances.

   (b) Use your NN resampling function to increase the size of I by factors of 17/9 and and 3/2. Display the results and comment on their appearances.

3. Write a program to resize a color image using bilinear interpolation. Repeat the two sections of problem 2 using this new function. Hint reuse your nearest neighbor algorithm to select the actual (fractional) pixel locations that determine, for each output pixel, which four input pixels to interpolate.

4. Write a program to rotate an image an arbitrary angle in degrees. Positive rotation should be clockwise since the y-axis points downward in an image. The output image should be large enough to accommodate the complete result, but not larger. Use center of the image as the center of rotation. Hints: (1) Use backward mapping. First determine the size of the output image. Then for each pixel location in the output image determine the location of the pixel in the input image to copy. (2) To find the corresponding input location, rotate the the output pixel location in the opposite direction and round or truncate the results. If the computed location of the input pixel is out of range, zero the output pixel.

   Display both positively and negatively rotated versions of the same image.

5. Find a true-color image on which to perform linear warping. Display the image and select in it four or more points as warp-point origins using `[x y] = ginput()`. Create a set of valid target points. Generate a planar homography that maps the input points to the output points. Use the homography to remap the entire image.

6. **Extra problem for graduate students.** Problem 4 is an example of the two-step procedure for image warping in which (1) a map the size of the output image is constructed that associates an input pixel location with every output pixel and (2) for each output pixel, the image values from a neighborhood of the associated input pixel location are used to compute the output pixel value. This is known as sampling followed by interpolation. In the previous problem, the sample locations are the inversely rotated output pixel locations and the interpolation was *de facto* nearest neighbor. Why is that?

   For this extra problem replace, in the rotation function, the nearest neighbor interpolation with a bilinear interpolation of the neighbors of the fractional input location.

Rules for laboratory assignments

1. Perform all the tasks listed in the instructions.
2. Explain the tasks you performed in detail.
3. Answer in writing in your report all the questions asked in the instructions.
4. Include in the report the original images you used and those resultant images that were specified in the instructions.

5. Include all computer code that you wrote and used, clearly documented, in an appendix.

6. All work must be yours and yours alone. Collaboration on the laboratory assignments is forbidden, with the following exceptions:

   (a) You may obtain help on any aspect of the homework from either Prof. Peters or the TA for this course.

   (b) You may obtain technical help on `matlab` from anyone you wish. However *you may not get direct help on the implementation of the specific algorithm* from another person except as noted in (a) above.

   (c) You may get help in obtaining the *input* images for the assignments from anyone you wish.

   (d) You may get help in the formatting or storing or transmission of your reports, but *not the content*, from anyone you wish.

7. Write your results in a clear laboratory report format using MS Word, WordPerfect, LaTeX, or any other word processor with which you can embed images in text. I prefer that the reports be submitted in .pdf format, but that is not required. Submit your report to me as a file on Blackboard. If for some reason this does not work, you may submit your report on a CD-ROM.

8. Assignments are due at midnight on the day specified in the instructions or in class. The grade on a laboratory report will be reduced by 10 points (out of 100) for every day (24 hours) that it is late.